ICSSP’17
Proceedings of the 2017 International Conference on
Software and System Process

Edited by:
Reda Bendraou, David Raffo, Huang LiGuo, and Fabrizio Maria Maggi

Sponsored by:
ISSPA, UPMC, in-coop with ACM SIGSOFT

Supported by:
System X, Criteo, Frauenhofer IESE, USC CSSE
Message from the Chairs

Welcome to the International Conference on Software and System Processes (ICSSP) 2017 held in Paris, France, between July 5th and 7th, 2017. The ICSSP conference, continuing the success of the Software Process Workshop (SPW), the Workshop on Software Process Simulation Modeling (ProSim) and the International Conference on Software Process (ICSP) conference series, has become an established premier event in the field of software and systems engineering processes in collaboration with ACM/SIGSOFT. It provides a leading forum for the exchange of academic research results and industrial best-practices in process development and evolution on software and systems disciplines. ICSSP 2017 celebrates the 10-year anniversary of the conference series.

In whatever field (software, military, healthcare or business), processes are ubiquitous. Not only are processes used to design, produce, and deliver enterprise’s products and services, but they are also used to instantiate enterprise/government practices, policies and regulations. Fundamentally, processes are the means by which an enterprise satisfies customers and creates value, and they comprise the building blocks of all information systems. Given their importance to the success and profitability of the enterprise, it is more important than ever to seriously consider the design and improvement of these processes and to make sure that they are free of flaws and inconsistencies. Recent advances in the hardware domain have paved the way for more complex and critical systems making the development processes for these systems very challenging. The Internet of Things (IoT) applications and Cloud computing infrastructures have introduced an increasing need for distributed system applications and more intelligent system segmentation. Big data solutions offer the potential to learn from vast amounts of information generated by systems. This information can be used to satisfy business needs or simply to learn and to improve the systems. The ICSSP 2017 conference seeks to explore how these new trends impact and constrain the way we design processes of different types and working in different application domains. The conference also seeks to inquire how new visions and tools that incorporate cloud computing, big data, IoT, and DevOps can be used by processes to design the next generation of systems and to improve collaboration and coordination between teams. As a community, we explore
  
  - How will the next generation of process paradigms look like?
  - How will business and system stakeholders in the development and evolution of processes be supported?

The ICSSP 2017 program is featured by two keynote speeches, delivered by Dr. Tom Zimmermann (Senior Researcher in the Software Engineering group at Microsoft Research, Redmond, USA), Dr. Dieter Rombach (Head of the Research Group for Software Engineering (AGSE) and Director Business Development of the Fraunhofer Institute for Experimental Software Engineering (IESE) in Kaiserslautern, Germany). Both keynotes well address the
theme of ICSSP 2017. Many companies are considering understanding and improving productivity of individual software developers as well as software teams. Dr. Tom Zimmermann motivates the need for data analytics in software teams and describes how data scientists work in large software companies helping software teams to infer actionable insights. His talk also shows how data from software development can be used to learn more about the productivity of organizations, teams, and individuals and help them to become more effective in building software. Dr. Dieter Rombach’s talk entitled "New Software Engineering Challenges in the Digital Transformation Era" sheds light on how digital transformation involves the creation of new business models based on communication among all “things” and effective use of big data. This revolution affects all sectors of industry (e.g., industry 4.0) and society (e.g., health, energy) as well as private life (e.g., mobility). The resulting systems of systems pose completely new software engineering challenges such as complex open systems of systems with heterogeneous development environments, integration of safety & security issues, or run-time adaptivity. Dr. Rombach's presentation surveyed these challenges as well as some of the ongoing research and development activities to address them. In addition, ICSSP 2017 features a mixture of full paper and short paper presentations on closely relevant topics exploring various aspects of software and systems processes. We are convinced that there are many interesting ideas, corresponding to the conference theme, worth being introduced to the community.

In response to the Call for Papers, this year 42 valid submissions were received by authors from 23 countries, including 31 full papers and 11 short papers. Each paper was rigorously reviewed by 4 Program Committee members and held to very high quality standards. Finally, 11 full research papers and 5 short papers were accepted for presentation at the conference, representing an overall acceptance rate of 38%, with 35% for full papers and 45% for short papers, respectively.

The accepted papers present completed research, advanced work-in-progress and industrial experience reports in various areas of software and system processes as well as domains outside the traditional software process community such as business processes. We clustered these papers based on their topics to form five tracks for this conference. Besides, we have included one Journal First session which presents three selected papers from the Journal of Software: Evolution and Process (JSEP) Special issue on “Software Engineering for Connected Health Systems”, as well as one panel session on the topic “Systems and Software Processes Meet System Diversity”, during which panelists and conference attendees have an opportunity to discuss a wide variety of experiences in developing, determining, and applying various types of processes.

With ICSSP 2017 being the 10th anniversary of the ICSP/ICSSP Conference series, we decided to look back at the papers published from the special journal issue containing the best papers from ICSP 2007 which appeared in Software Process: Improvement and Practice in 2008. A panel composed of three members of the ICSSP Steering Committee reviewed these papers and
unanimously selected one paper. The criteria used were citations and relevance of the topic to the current practice and research of software engineering. The paper selected was:


David Anderson's presentation provided a summary of the paper and a 10-year retrospective of how work in Lean and Agile development has grown since the paper was published, the major developments in the field and how this paper and the ideas presented in it have contributed to this development. This paper also had the distinction of receiving the most downloads of any paper published by Wiley Interscience in 2008 across all technical disciplines.

The success of the conference would not have been possible without the dedication and professional supports of many colleagues. We would like to express our gratitude to all contributors who submitted papers. Their work formed the basis for the success of the conference. We would also like to thank the Program Committee members and reviewers for devoting their time and efforts to help assess the submissions and guarantee the high quality of the conference. We appreciate the keynote speakers for delivering their insightful speeches at the conference. We are also grateful to the Steering Committee members, Barry Boehm, LiGuo Huang, Ross Jeffery, Marco Kuhrmann, Mingshu Li, Leon Osterweil, Dewayne Perry, David Raffo, Reda Bendraou and Wilhelm Schäfer, for their continuous advice, encouragement and support. Finally, we would also like to extend our special thanks to the International Software Process Association (ISSPA), SIGSOFT, Université Pierre & Marie Curie (UPMC), System X, Criteo, Fraunhofer ISE, and USC CSSE for sponsoring this conference. We further appreciate ACM for their collaboration in publishing the conference proceedings.

For further information, please visit the conference website at http://icssp-conferences.org/.

July 2017

Reda Bendraou
General Chair

David Raffo
General Chair

LiGuo Huang
Program Chair

Fabrizio Maria Maggi
Program Chair
ICSSP 2017 Organization

Organizing Committee

General Chairs:
Reda Bendraou University of Pierre & Marie Curie, France
David Raffo Portland State University, USA

Program Chairs:
LiGuo Huang Southern Methodist University, USA
Fabrizio Maria Maggi University of Tartu, Estonia

Workshop Chair:
María Cecilia Bastarrica Universidad de Chile, Chile

Poster Chair:
Djamel Khelladi Johannes Kepler University Linz, Austria

Publicity Chairs:
Philipp Diebold Fraunhofer IESE, Germany
Regina Hebig University of Gothenburg, Sweden

Program Committee Members
Amin Jalali (Sweden)
Andrea Burattin (Austria)
Andrea Marrella (Italy)
Barbara Russo (Italy)
Bernard Coulette (France)
Chiara Di Francescomarino (Italy)
Christiane Gresse von Wangenheim (Brazil)
Chuanyi Li (China)
Claudio Di Ciccio (Austria)
Davide Fucci (Finland)
Daniel Houston (USA)
Daniel Rodriguez (Spain)
Danilo Caivano (Italy)
Dietmar Pfahl (Estonia)
Dietmar Winkler (Austria)
Dewayne Perry (USA)
Graham Hellestrand (USA)
Jason He Zhang (China)
Jens Heidrich (Germany)
Josep Carmona (Spain)
Juergen Muench (Finland, Germany)
Kenneth Nidiffer (USA)
Krysztof Wnuk (Sweden)
Leon Osterweil (USA)
Luciano García-Bañuelos (Estonia)
Maleknaz Nayebi (Canada)
Marco Kuhrmann (Denmark)
Marco Montali (Italy)
Marlon Dumas (Estonia)
Maya Daneva (Netherlands)
Meiru Che (USA)
Michael Felderer (Austria)
Minseok Song (South Korea)
Mingshu Li (China)
Murat Yilmaz (Turkey)
Ove Armbrust (USA)
Paul Clarke (Ireland)
Qi Li (USA)
Rafael Prikladnicki (Brazil)
Riccardo De Masellis (Italy)
Richard Turner (USA)
Rory O’Connor (Ireland)
Slinger Jansen (Netherlands)
Stanley Sutton (USA)
Supannika Koolmanojwong (USA)
Tom McBride (Australia)
Thomas Birkhoelzer (Germany)
Tijs Slaats (Denmark)
Walt Scacchi (USA)

Additional Reviewers
Andrea Fornaia (Italy)
Christian R. Prause (Germany)
Fahad Rafique Golra (France)
Guoping Rong (China)
Lukas Linsbauer (Austria)
Tobias Baum (Germany)
Contents

Frontmatter

Message from the Chairs ................................................................. iii
ICSSP 2017 Organization ................................................................. vi
ICSSP 2017 Sponsors and Supporters ............................................. viii

Invited Talks
Software Productivity Decoded: How Data Science Helps to Achieve More (Keynote)
  Thomas Zimmermann — Microsoft Research, USA ............................... 1
Software Engineering for Connected Health (Journal First Session)
  Noel Carroll, Craig Kuziemsky, and Ita Richardson — Lero, Ireland; University of Limerick, Ireland; University of Ottawa, Canada .......................... 3

Systems and Software Process Traceability
Improving Traceability Management through Tool Integration: An Experience in the Automotive Domain
  Domenico Amalfitano, Vincenzo De Simone, Anna Rita Fasolino, and Stefano Scala — Federico II University of Naples, Italy; Fiat Chrysler Automobiles, Italy .................................................. 5
Mining Collaboration Patterns of Software Development Processes Based on Trace Alignment
  Dongjin Yu and Jiaojiao Wang — Hangzhou Dianzi University, China .................. 15

Tracing Requirements in Software Design
Zeheng Li, Mingrui Chen, LiGuo Huang, Vincent Ng, and Ruili Geng — Southern Methodist University, USA; University of Texas at Dallas, USA; Spectral MD, USA .......................... 25

Empirical Studies and Experience Reports
Hybrid Software and System Development in Practice: Waterfall, Scrum, and Beyond
  Marco Kuhrmann, Philipp Diebold, Jürgen Münch, Paolo Tell, Vahid Garousi, Michael Felderer, Kitija Trekteere, Fergal McCaffery, Oliver Linssen, Eckhart Hanser, and Christian R. Praise — TU Clausthal, Germany; Fraunhofer IESE, Germany; Reutlingen University, Germany; IT University of Copenhagen, Denmark; University of Luxembourg, Luxembourg; University of Innsbruck, Austria; Dundalk Institute of Technology, Ireland; FOM University of Applied Sciences for Economics and Management, Germany; DHBW-Loerrach, Germany; DLR, Germany .................. 30
Requirements Volatility in Software Architecture Design: An Exploratory Case Study
Sanja Aaramaa, Sandun Dasanayake, Markku Oivo, Jouni Markkula, and Samuli Saukkonen — University of Oulu, Finland .......................................................... 40
A Systematic Map on Verifying and Validating Software Process Simulation Models
Haojie Gong, He Zhang, Dexian Yu, and Bohan Liu — Nanjing University, China ............ 50

Agile World
Rapid, Evolutionary, Reliable, Scalable System and Software Development: The Resilient Agile Process
Doug Rosenberg, Barry Boehm, Bo Wang, and Kan Qi — ICONIX Software Engineering, USA; University of Southern California, USA ........................................ 60
Searching for Common Ground: Existing Literature on Automotive Agile Software Product Lines
Philipp Hohl, Javad Ghofrani, Jürgen Münch, Michael Stupperich, and Kurt Schneider — Daimler, Germany; Leibniz Universität Hannover, Germany; Reutlingen University, Germany; University of Helsinki, Finland .................. 70
How Do Agile Practices Support Automotive SPICE Compliance?
Philipp Diebold, Thomas Zehler, and Dominik Richter — Fraunhofer IESE, Germany; IBM, Germany .......................... 80
Process Modeling and Simulation

White-Box Prediction of Process Performance Indicators via Flow Analysis
Ilya Verenich, Hoang Nguyen, Marcello La Rosa, and Marlon Dumas — Queensland University of Technology, Australia; University of Tartu, Estonia ................................................................. 85

Domain Modeling for Development Process Simulation
Ian J. De Silva, Sanjai Rayadurgam, and Mats P. E. Heimdahl — University of Minnesota, USA ................................. 95

Business Process Improvement

Leveraging Business Process Improvement with Natural Language Processing and Organizational Semantic Knowledge
Deniz Iren and Hajo A. Reijers — VU University Amsterdam, Netherlands .................................................. 100

Data Driven Credit Risk Management Process: A Machine Learning Approach
Mingrui Chen, Yann Dautais, LiGuo Huang, and Jidong Ge — Southern Methodist University, USA; GDS Link, USA; Nanjing University, China .................................................. 109

Process and Project Management

Using TRIZ to Balance Software Process Commonality and Diversity
Dan Houston — Aerospace Corporation, USA .................................................. 114

Process-Based Project Management and SPI
María Cecilia Bastarrica, Daniel Perovich, Jacqueline Marín, and Luis Rioseco — University of Chile, Chile 124

Impact of Task Switching and Work Interruptions on Software Development Processes
Alexey Tregubov, Barry Boehm, Natalia Rodchenko, and Jo Ann Lane — University of Southern California, USA; San Diego State University, USA .................................................. 134

Poster Papers

WebProcessPAIR: Recommendation System for Software Process Improvement
Mushtaq Raza, João Pascoal Faria, Luis Amaro, and Pedro Castro Henriques — INESC TEC, Portugal; University of Porto, Portugal; Strongstep - Innovation in Software Quality, Portugal .................................................. 139

Towards Capability-Oriented Business Process Management
Amin Jalali — Stockholm University, Sweden .................................................. 141

Workshops

First International Workshop on Verification of Business and Software Processes
Souheib Baarir and Kais Klai — Paris Nanterre University, France; University of Paris North, France .......................... 143

First International Workshop on Hybrid dEveLopmENt Approaches in Software Systems Development
Marco Kuhrmann, Philipp Diebold, and Jürgen Münch — TU Clausthal, Germany; Fraunhofer IESE, Germany; Reutlingen University, Germany; Herman Hollerith Center, Germany .................................................. 145

Author Index ........................................................................................................... 147
Software Productivity Decoded: How Data Science Helps to Achieve More (Keynote)

Thomas Zimmermann
Microsoft Research
One Microsoft Way
Redmond, WA 98004, USA
tzimmer@microsoft.com

ABSTRACT
Many companies are looking into understanding and improving productivity of individual software developers as well as software teams. In this talk, I will motivate the need for data analytics in software teams and describe how data scientists work in a large software company to help software teams to infer actionable insights. I will then show how the data from software development can be used to learn more about the productivity of organizations, teams, and individuals and help them to become more effective in building software.

CCS CONCEPTS
• Software and its engineering → Software development process management; Software development techniques; Software post-development issues; Collaboration in software development;

KEYWORDS
Software productivity, data science

ACM Reference format:
https://doi.org/10.1145/3084100.3087674

1 DATA SCIENCE IN SOFTWARE TEAMS
Software and its development generate and large amount of data. For example, check-ins, work items, bug reports and test executions are recorded in software repositories such as CVS, Subversion, GIT, and Bugzilla. Telemetry data, run-time traces, and log files reflect on how customers experience software, including application and feature usage, performance and reliability. The sheer amount is truly impressive.

Over the past years the field of software analytics has emerged with the goal to use data to improve software. Data scientists are now very common in software teams. They use data about software and its development and turn it into actionable insight to inform better decisions. I’ve been studying how data scientists work in industry and in this talk I will share some of observations and lessons learned, including

• What is the educational background of data scientists?
• How do data scientists work? What tasks do they work on, how do they spend their time, and what tools do they use?
• What challenges do data scientists face and what are the best practices and advice to overcome those challenges?
• What processes do data scientists use to increase confidence about the quality of their work?

Some of the work that I will present in the keynote has been previously published. For example, a catalog of 145 questions that software engineers have for data scientists [1] or five working styles of data scientists [2]:

• Polyath, data scientists who ‘do it all’
• Insight Provider, main task is to generate insights and to support and guide their managers in decision making
• Platform Builder, build shared data platforms used across several product teams
• Modeling Specialist, data scientists who act as expert consultants and build predictive models
• Team Leader, senior data scientists who run their own data science teams act as data science ‘evangelists’

I will also present unpublished results from a recent large-scale survey of data scientists in industry and highlight opportunities on how to leverage data science for software processes.

2 PRODUCTIVITY IN SOFTWARE TEAMS
There is an ever-growing demand of software being built and a shortage of software developers to satisfy this demand, despite the immense growth in the number of professional software developers. To address this demand, industry and research are looking into understanding and improving the productivity of individual software developers as well as teams. A substantial amount of work has examined the meaning of software productivity over the past four decades. Much of this work introduces particular definitions of productivity, considers organizational issues associated with productivity, or is focused on specific tools and approaches for improving productivity. In fact, many of the seminal works on software productivity are from the 80s and 90s (Peopleware, Mythical Man-Month, Personal Software Process).

At the same time, software development has changed significantly over the past decades with the rise of agile development, distributed development, more rapid release cycles and the high fragmentation of today’s work. The technology available to software engineers has improved with social coding tools like GitHub
and StackOverflow and better IDEs. The amount of data available about software development has grown significantly over the past decade. Software engineering data is collected

The availability of more fine-grained data about development allows to revisit many of the fundamental questions about software development with a much larger scale than in the past. I will talk about how to use data science to answer questions such as:

- How do developers work and how do they spend their time?
- How to measure software productivity?
- What are the impediments on productivity of developers?
- Can we learn patterns of highly productive developers?

In the talk, I will share some of our experiences on understanding and improving productivity of software developers at Microsoft, starting from perceptions of developer productivity [3] to how we can model developer productivity [4], including some unpublished work about physical work environments and developer satisfaction.

BIOGRAPHY

Thomas Zimmermann is a Senior Researcher in the Research in Software Engineering group at Microsoft Research, Redmond, USA. His research interests include software productivity, software analytics, recommender systems, and games research. He is best known for his research on systematic mining of software repositories to conduct empirical studies and to build tools to support developers and managers. His work received several awards, including Ten Year Most Influential Paper awards at ICSE 2014 and MSR 2014, 2015, and 2017, five ACM SIGSOFT Distinguished Paper Awards, and a CHI Honorable Mention. He currently serves as Program Co-Chair for ICSME 2017. He is Co-Editor in Chief of the Empirical Software Engineering journal and serves on the editorial boards of several journals, including the IEEE Transactions on Software Engineering. He received his PhD in 2008 from Saarland University in Germany. http://thomas-zimmermann.com

ACKNOWLEDGMENTS

The work presented in this keynote has been done with many of my collaborators Andrew Begel, Christian Bird, Jacek Czerwonka, Rob DeLine, Denae Ford, Thomas Fritz, Brittany Johnson, Miryung Kim, Gloria Mark, Andre Meyer, Gail Murphy, Nachi Nagappan, Peggy Storey, and many others.

REFERENCES

Software Engineering for Connected Health
(Journal First Session)

Noel Carroll
Lero – the Irish Software Research Centre
University of Limerick
Ireland
noel.carroll@lero.ie

Craig Kuziemsky
Telfer School of Management
University of Ottawa
Canada
kuziemsky@telfer.uottawa.ca

Ita Richardson
Lero – the Irish Software Research Centre
University of Limerick
Ireland
ita.richardson@lero.ie

ABSTRACT
Changes in health management, healthcare, society, demographics, economics and information and communications technology have challenged the healthcare delivery model globally. Healthcare demands are causing difficulties for healthcare professionals’ capabilities to deliver safe quality care in a timely manner. Preventative health management is increasing in importance. Therefore, health’s transformative efforts focus on the need to support care delivery across various providers and settings, and connected health solutions can support and resolve these efforts. This Journal First session presents 3 papers which discuss the role software engineering in supporting connected health.

CCS CONCEPTS
• Applied computing →Health informatics

KEYWORDS
Software Engineering; Connected Health; Health management; Healthcare

ACM Reference format:
DOI: 10.1145/3084100.3087675

1 WHY CONSIDER SOFTWARE ENGINEERING FOR CONNECTED HEALTH?
Societal, health and demographic changes coupled with recent economic challenges have challenged us to reconsider how we deliver health and social care in our communities [1]. Due to aging populations and medical advances, it is inevitable that healthcare demands will outgrow healthcare professionals’ capabilities to deliver safe quality care in a timely manner. Additionally, healthcare’s transformative efforts focus now support preventative health management. Studies have indicated that socio-economic factors play an important role in supporting changing healthcare requirements. We describe this as ‘healthcare accessibility’ and good primary care which delivers care and services close to home is associated with better population health, lower rates of unnecessary hospitalisations, and relatively lower socio-economic inequality[2]. The importance of healthcare accessibility becomes even more critical due to increasing costs associated with providing healthcare services. This is particularly relevant for an increasingly older population. Indeed, healthcare systems across the world are attempting to address the challenges that result from ageing populations, the growth in chronic diseases, burgeoning technical possibilities and public expectation [3]. Regardless of how they are funded, healthcare services continue to consume an increasing proportion of Gross Domestic Product across the OECD countries. Thus, we need to coordinate smarter care models as they apply to health care and management [4, 5], working within Connected Health (CH), a complex socio-technical model for healthcare management which we define as:

Connected Health enables the delivery of process-driven collaborative health management and healthcare practice by individuals, healthcare professionals, patients and/or carers through the support of technology (software and/or hardware). CH, which can be undertaken in the person’s home, care centre or hospital setting, challenges software engineers in new ways, and Software Engineering for Connected Health supports these challenges.

Why is Software engineering for CH different? In the first instance, medical software, a subset of connected technologies, is subject to regulation. Research by [6] has discussed how this should be done: regulation should be embraced and used as a guideline to CH solutions. Secondly, information technology spending within healthcare has been significantly lower than many other sectors, resulting in low quality software systems, requiring healthcare professionals to implement inefficient processes and practices. Thirdly, CH software is subject to
Medical Council and Ethical guidelines which vary between countries. Fourthly, healthcare professionals unquestioningly place their trust in (often inaccurate and unreliable software [7, 8]. Finally, healthcare connectivity is not linear or static, but rather is a complex system with many interrelated concepts [9]. Designing for complex systems introduces challenges such as dynamic processes (e.g. collaborative care delivery) that may evolve over time presenting with different needs. Thus trade-offs may be needed [10, 11]. In designing CH software, we need to incorporate this complexity into systems design as much as possible [12].

Evaluation can play an important part in solving problems that exist. There have been recent efforts to evaluating multiple perspectives e.g. clinical, business, technology of a CH system, identifying a robust means of justifying the need for CH solutions and identifying opportunities for such solutions [13]. including a CH Evaluation Framework (CHEF) introduced by [14]. [12] offer the Social Information System Connectivity Framework for supporting social information system (SIS) design in healthcare. Their framework defines the structure of a SIS as an integrated set social triads consisting of people, processes and technology and the behaviour of a SIS according to six behavioural dimensions. In addition, [15] focus on the practitioner’s perspective on a CH solution using clinical pathway support systems. Building on success factors for CH solutions, and drawing on interviews and case study synopses, [16] synthesize 10 key success factors for CH software innovations. These research developments, among others being presented at the Journal First session of ICSSP, play a key role in the development of software engineering for CH solutions.

To extend the CH field, we are publishing a Special Issue of the Journal of Software Evolution and Process on Software Engineering for Connected Health. The papers presented at this Journal First session have been selected from submissions to this, and include the topics of continuous improvement for medical device companies, performance management of community care services and requirements elicitation.

ACKNOWLEDGEMENT

This work was partially supported by Science Foundation Ireland grant 13/RC/2094 to Lero (www.lero.ie).

JOURNAL FIRST PAPERS

Eze, B., C. Kuziemsky, L. Peyton, Cloud-based Performance Management of Community Care Services.

Webber, P., J. Filho, M. Lee, R. Bachman, Automated Conflict Detection between Clinical Pathways.

Ozden Ozcan-Top and Fergal McCaffrey, A Continuous Improvement Approach for Medical Device Software Development Companies.

REFERENCES


Improving Traceability Management through Tool Integration: An Experience in the Automotive Domain

Domenico Amalfitano
Department of Electrical Engineering and Information Technologies
University of Naples Federico II, Italy
domenico.amalfitano@unina.it

Anna Rita Fasolino
Department of Electrical Engineering and Information Technologies
University of Naples Federico II, Italy
fasolino@unina.it

Vincenzo De Simone
Department of Electrical Engineering and Information Technologies
University of Naples Federico II, Italy
vincenzo.desimone2@unina.it

Stefano Scala
Fiat Chrysler Automobiles Pomigliano Technical Center
Via ex Aeroporto, Pomigliano d’Arco
Italy
stefano.scala@fcagroup.com

ABSTRACT

Despite the relevance of traceability in software processes is well-known, the activities of traceability creation and management are not always adequately supported in real software projects. The lack of integration between the tools adopted in the development processes is one of the main causes of such an ineffective management, where traceability relationships are still manually generated and maintained. In this paper we present an industrial experience we performed for improving the traceability management in a software development process performed in Fiat Chrysler Automobiles FCA company. We designed a software architecture for integrating the existing Application Lifecycle Management (ALM) platform with the tools used in the testing process. The architecture aimed at fully automating the execution of the testing process and at automatically generating the appropriate traceability links when they are established. It was implemented using a Continuous Integration Engine that allowed us to develop a modular, evolvable and reconfigurable integration architecture. The new architecture was validated by an experiment that showed its capability in correctly and completely generating and handling traceability links between artifacts involved in the testing process. The experiment demonstrated that the integration solution produced also beneficial effects on other quality attributes of the process.

KEYWORDS

Traceability Management; Tool Integration; Software process improvement; Automotive; Industrial Experience

1 INTRODUCTION

In 1994 Gotel et al. [9] defined the Requirement traceability as the ability to describe and follow the life of a requirement, in both a forwards and backwards direction. Intuitively, this definition is realized in practice by establishing relationships, called trace links, between the requirements and one or more artifacts of the system [14]. During the years this definition was extended and nowadays more in general we talk about Software traceability for indicating the creation and the use of links (or connections) between different kinds of software artifacts such as requirements, models, source code, test cases, or test results. These connections are called trace links and connect in a bidirectional way a source artifact to a target artifact. Many authors addressed the topic of the traceability management that is the planning, the organization, and the coordination of all the activities related to traceability, including the creation, maintenance, and use of trace links [26]. Traceability is required by different software development standards (SPICE, CMMI, ISO 26262, etc.) and its importance is well-recognized by software engineering community [17]. Traceability leads to improvements in software systems by supporting tasks related to maintenance, evolution, reuse and more. In practice it is very difficult to guarantee an effective traceability management since software projects are often developed by distributed teams, both software artifacts and trace links undergo constant change, and multiple stakeholders with different backgrounds are involved [26]. Moreover, traceability support in contemporary software engineering tools is not satisfactory [12].
Nowadays, an effective solution for managing the traceability between software artifacts is offered by Application Lifecycle Management (ALM) platforms. An ALM platform consists of a set of tools, technologies, or techniques that attempt to provide support for monitoring, controlling and managing software development over the whole application lifecycle [11]. At the same time ALM platforms offer a unified storage and management of every software artifact that is simply stored in version control repositories. They may also provide the automatic implementation and management of traceability relations among artifacts that are guaranteed via automatic change control of every requirement.

ALM tools certainly offer a valuable support to the software lifecycle. Unfortunately, the existing implementations of ALM do not provide tool support for every lifecycle process, neither the existing ALM are able to automatically integrate with any legacy tool and the artifacts it may produce [8]. As a consequence, in many organizations ALM are just used for storing the traceability links between artifacts, while traceability generation and update are manually performed by the personnel involved in the software lifecycle. This habit causes a great waste of time and resources and may yield many inconsistencies, errors, and problems with the manually produced links.

To overcome these limitations, more and more organizations are forced to address the problems of integrating the adopted ALM platform with their legacy software tools and of implementing the automatic generation and update of the traceability links.

A feasible solution to these problems requires the creation of toolchains that implement pipelines of existing tools. Several approaches have been proposed in the literature to implement toolchains, such as [2] but they are still not mature enough for being adopted.

In the practice, the organizations often implement ad-hoc solutions, based on glue code, wrappers, adapters, and other software integration mechanisms, which often result in scarcely maintainable and efficient systems.

In this paper we present an industrial experience we performed in collaboration with the Fiat Chrysler Automobiles (FCA) company within the research project APPS4SAFETY 1. The main aim of APPS4SAFETY was to improve the traceability management in a software testing process performed in FCA.

In this project, we designed a software architecture for integrating the existing ALM platform with the tools used in the testing process. This solution aimed at fully automating the execution of the testing process and at generating the appropriate traceability links when they are established (in-situ creation). These links had to be automatically stored in the ALM used in the company.

To implement this toolchain we exploited a Continuous Integration Engine that allowed us to develop a modular, evolvable and reconfigurable integration architecture, satisfying the design principles of low coupling and location transparency.

In order to validate the proposed solution, we conducted an experiment in the industrial organization. The experiment involved real software testing projects conducted in the company and the personnel involved in it. The experimental results showed the capability of the proposed solution of correctly and completely handling traceability links between software artifacts involved in a testing process. Moreover it demonstrated that the integration solution produced beneficial effects on several quality attributes of the process.

The remainder of the paper is organized as follows. Section 2 provides background information about ALM and solutions for software toolchain implementation. Section 3 presents the software testing process where we addressed the integration problem. Section 4 illustrates the software architecture of the proposed integration solution. Section 5 presents the validation experiment while Section 6 finally reports conclusive remarks.

2 BACKGROUND

Application Lifecycle Management has been proposed as a solution for monitoring, controlling and managing software process over the entire application lifecycle [10]. ALM can be seen as a supervisor which covers the whole development process from the initial idea to the end of the product lifecycle through different core aspects: governance, development and operations [6]. They combine the supporting tool such as Version Control Systems (VCS) to handle the artifacts to issue tracking applications [15]. Nowadays many companies adopt Application Lifecycle Management (ALM) platforms to follow the entire lifecycle of their software product. The goal of an ALM platform is to make software development and delivery more efficient, lower its costs and improve software quality. ALM is based on three main concepts, called the pillars of ALM: traceability, visibility and process automation [18]. In order to truly reach these goals, an ALM platform should provide a holistic view on all the software process. All the tools involved in the development process should be able to interact among them through the ALM platform.

Although ALM tool support has increased during the past decades, there is still a tool support ecosystem largely fragmented [8]. Many of the ALM tools available on the market (e.g. IBM Rational Team Concert, HP Application Lifecycle Management, Polarion ALM, etc.) are able to reach the defined goals, but only through the integration of tools of their ecosystems. Most companies already have a de-facto development process and tools. For these reasons ALM platforms offer mechanisms to interconnect most commonly used tools like Microsoft Excel, Eclipse, IBM DOORS and they also offer an Application Programming Interface (API) for the integration with other tools.

There is still the need to provide solution able to integrate the different tools used during the software development in order to automate its process and to interconnect artifacts created through the adopted heterogeneous tools.

The need to link tools that address different aspects of the development process, creating so called toolchains is emerged since 90s [23].

Tool integration is often achieved building automated tool-chains that take care of all the needed interactions between tools, from data exchange and manipulation, to the tools execution itself. Creating a toolchain is not an easy task since every toolchain is specific for a particular development process and environment. Based

---

1 Active Preventive Passive Solutions for Safety - PON03PE_00159_3
on the number of tools to integrate and on the breadth of the development sub-process that must be automated, it can be hard to properly describe, design and develop the needed toolchain.

Solutions to tool integration are based on the same principles: the presence of a central repository for data management, a framework based on a service oriented architecture or a component architecture for tool orchestration and a set of transformations to make data available to different tools. In [16], in the context of MOGENTES project (a project aimed at enhancing testing and verification of dependable embedded systems), a service-oriented, model-driven, process-centric approach for the definition of toolchains was proposed. Their approach leads to the creation of a tool manager (the broker) to which the tools should be registered to offer their functionality as services to the final user. However, each tool must expose a well defined interface to provide homogeneous programming interfaces and hide heterogeneity.

In [2], a way to properly describe a toolchain defining a domain specific modeling language (DSML), named Tool Integration Language, was proposed. This language aims at mapping tool integration domain concepts to language ones in order to fully describe a toolchain and to allow its automatic generation using a Service Component Architecture (SCA). Anyway such approach is not used in practice where usually ad-hoc solutions are developed, instead.

The tool integration was highly investigated in the automotive domain, where several research projects have been funded to investigate this problem. These projects aim to create a toolchain that covers the entire embedded software development process, starting from requirements specification, to model design, to its implementation and testing.

The Cesar Project [1] offers a toolchain that is divided in two interconnected parts, one for the requirements management and one for the system design, safety analysis and V & V. The two parts use different integration technologies and different repositories for their artifacts, but the toolchain is able to offer full traceability of the application. The integration is achieved by using a defined meta-model and transformations technologies.

The Amalthea Project [27] goal was to create an open source toolchain for embedded software development in the automotive domain based on the Eclipse Software development Framework and leaded to the Amalthea4Public tool. The proposed platform integrates a lot of existing tools already supported by the Eclipse Framework. These tools can be plugged in and executed automatically, but Amalthea platform adds to the standard framework the possibility to define models using the Eclipse Modeling Framework and the presence of an high performance model repository to store development models so that every tool in the framework can access project data.

Many tools were proposed to help companies keeping traceability between the artifacts involved in the development process and choosing which one to use is a complex task. The usually adopted tools to express and maintain traceability were general purpose tools such as word processors or, even better, spreadsheets. They are adopted to create a Traceability Matrix, a matrix showing dependencies between artifacts. This matrix is easily readable by an user and has the advantage of being a single repository for documenting both forwards and backwards traceability across all of the work products [24]. Traceability was also managed using Entity-Relationship (ER) Models and database technology, with the development artifacts represented by entities, while relationship instances represent links between artifacts [25]. But keeping such a matrixes or tables always updated requires considerable manual efforts. With the advent of hypertext technology and special-purpose tools it became easier to provide full traceability to a software project.

The importance of automatic traceability management tool has been extensively investigated by the authors in [13] who conducted a series of interviews with industrial and academic partners and software development stakeholders to study the impact of such tool on the quality of traceability links. The data from these interviews reveals two main challenges: traceability links are still mostly created manually and users must cope with interconnected but highly heterogeneous artifacts across tool boundaries.

### 3 THE ADDRESSED PROBLEM

In this Section we present the tool integration problem we addressed in the context of embedded software development process. More precisely, we focused on the Model In the Loop (MIL) testing process performed by FCA EMEA SW Factory, the unit responsible for the development of embedded software for two Electronic Control Units of the vehicle, i.e., the Instrument Panel Cluster and the Body Computer Module.

In the last years, the automotive field has been interested by the introduction of Model Based (MB) approaches and technologies in the software development processes.

MB technologies yield many advantages to the software processes, which become mostly focused on producing high-level models of the Software Components (SWC) that can be used for simulation in very early stages of the development process. These technologies enable the engineers to test the SWCs models in a virtual environment at a stage of the process where they are inexpensive to be fixed, i.e. before the code is actually implemented or integrated on the final hardware which is called Electronic Control Unit (ECU) [4].

This first stage of testing is called Model In the Loop (MIL) and consists of a testing activity where the model and its environment are simulated (interpreted) in the modeling framework (usually MATLAB/Simulink Stateflow) without any physical hardware component [4].

The MIL testing process executed in FCA requires several steps and involves six different testing artifacts having heterogeneous formats, i.e., Microsoft Word Files, Microsoft Excel Files, and MATLAB files. The process exploits MATLAB as the virtual environment where the models of the SWCs are tested and involves the use of an ALM platform. The ALM stores all the artifacts necessary for and produced by the testing process, manages their lifecycle and the traceability relationships between artifacts. Moreover, it handles the roles of users involved in the process and regulates how they can interact with the artifacts.

Figure 1 reports the model of the artifacts (a.k.a. documents or work products) stored in Polarion and the <<related work product>>
Figure 1: Artifacts Relationships in SPEM

Figure 2: The MIL Testing Process in SPEM

Figure 3: The MIL Monitoring Process in SPEM

relationships that may exist among them. Instances of these relationships are actually the traceability links that should be stored in the ALM.

As figure shows, a Test Run is a composition of one or more Test Case documents. Each Test Case relates to the Software Component Release work product to be tested. Any Software Component Release is linked to the MATLAB Model artifacts simulating its behavior. A Test Case document has its own Test Case Result work products. Issue work products are associated to the Test Case founding them.

The considered MIL testing process is described in detail by the SPEM model reported in Figure 2.

The Test Engineer manually \(<\text{performs}\>\) the four activities required for the process completion. Polarion ALM and MATLAB are the \(<\text{used tool}\>\) exploited for the accomplishment of these activities.

The MIL process starts with the Create Test Run activity. Here the Test Engineer exploits Polarion for creating a new Test Run. The Test Cases needed for simulating a given Software Component Release are linked to the newly created Test Run and these links are stored into the ALM.

After, the Select Test Cases activity is performed. In this activity the Test Engineer exploits the ALM for selecting the actual Test Case artifacts and choosing the MATLAB Models implementing the behavior of the SWC Release under test.

These Test Cases are executed for simulating the MATLAB Models in the Launch Test Cases activity, where the Test Engineer has to configure, manually, the MATLAB testing environment before starting the Test Cases execution. At the end of this activity a document containing the results of the Test Cases, i.e. the Test Case Results, is produced.

The last activity of the MIL testing process is the Import Test Case Results. Here the Test Engineer manually stores, into the ALM, all the results of the test case executions, along with the traceability links between Test Cases and Test Case Results. Polarion is configured to produce automatically an Issue work item for each failed Test Case, i.e., a Test Case that found at least a failure on a MATLAB Model. Polarion automatically links Test Cases and Issues.

The MIL Testing process is monitored by the Project Manager, using the ALM tool. This monitoring process is performed according to the SPEM model reported in Figure 3. The Project Manager accesses the information about a SWC Release navigating through its related artifacts. In this way he is able to evaluate the process progress.

As the above descriptions highlighted, many tasks of the MIL process were manually executed by the Test Engineer, essentially due to the lack of integration between the ALM and the remaining testing environment. This caused many obvious disadvantages for the overall process. We were able to identify its most important drawbacks:

- The slowness of the MIL process, due to the many manual and time-consuming activities (of interacting with the ALM repository) performed by the Test Engineer.
- The ineffectiveness of the traceability links management that are manually introduced into the ALM by the Test Engineers. The responsibility for the completeness and the
Our solution should provide four of these levels of integration. The presentation integration was necessary because the Test Engineer had to interact only with the GUI provided by the ALM. The data integration was needed to guarantee the automatic import of the test results from the MATLAB testing tool into the ALM and the automatic creation of the traceability links. The control integration was required to allow the automatic cooperation of the two tools.

Finally, the solution had to guarantee the aspect of process integration since the data of the process management tool, i.e., Polarion ALM, had to be automatically obtained from the development tool, i.e., MATLAB testing tool.

We did not have to satisfy the platform integration aspect since the involved tools did not run on heterogeneous hardware nodes.

4 THE PROPOSED SOLUTION

To overcome the limitations of the current MIL testing process, it was necessary to find a solution for reducing the manual intervention of the Test Engineer in the process. To this aim, we decided to improve the integration between the tools used in the process and to automate the most tedious activities conducted by the Test Engineer, e.g., Select Test Cases, Launch Test Cases and Import Test Results.

More in detail, we required the new testing process to be executed carrying out the following activities:

1. the Test Engineer will create the Test Run,
2. the Test Engineer will exploit a single point of access provided by the GUI of Polarion, where it:
   • configure the MATLAB testing environment,
   • launches the execution of the Test Cases composing the Test Run,
3. the test case results will be automatically imported into Polarion ALM at the end of the test cases execution,
4. the traceability links between test cases, test case results and issues will be automatically created and stored into Polarion ALM.

To satisfy these requirements we had to find a solution for integrating the ALM tool with the testing environment. According to Wasserman [23] and Thomas [22], tools integration can be achieved at five different levels:

• **presentation integration**: this integration solution provides a similar look to the tools that have to be integrated or a single point of access to all tools, to deliver a better user experience;
• **data integration**: such a solution manages data sharing between tools and format and semantic transformations to make data available to different tools;
• **control integration**: it makes available the functions offered by a tool to other tools in the environment;
• **process integration**: it provides process management tools with data from development tools;
• **platform integration**: it provides network and operating systems transparency for tools based on heterogeneous hardware.

4.1 Architectural Design

For developing the integration architecture, we had to choose a strategy able not only to interconnect the ALM to the testing tool, but also of automatically coordinating the execution of the process activities by the supporting tools.

A first option consisted in exploiting the mechanism offered by Polarion ALM for interconnecting it with external tools and for extending the ALM platform default features. This mechanism consists in adding new plugins to the ones already present in the standard distribution of the tool. Polarion plugins are actually Java projects exploiting the APIs6 provided by Polarion. Through these APIs it is possible to programatically execute all the features of Polarion, such as selecting an artifact, adding a new artifact, adding a new traceability link between two artifacts, etc. Moreover, using Java commands for launching executable files, these plugins have the capability of controlling third party tools.

However, we did not decide to follow this strategy that implements in the ALM all the business logic of the process and couples the ALM directly to the external tools. Viceversa, we decided to introduce an intermediate Coordinator Component between ALM and more testing tool and to embed all the business process logic into this component. Figure 4 describes the proposed integration solution. The coordinator component will exploit specific connectors for interfacing the ALM with one or more MATLAB testing environments, respectively.

Such an alternative solution is more flexible and less invasive since it does not require any modification neither in the ALM nor in the testing tool. This architecture can be easily extended or adapted when new tools have to be integrated with the ALM platform or the process workflow needs to change.

4.2 Implementing Components and Connectors

The Coordinator Component has the responsibility for the automatic execution of the two tasks of Launch Test Cases and Import Test Case Results. Basically, it acts as a broker between Polarion ALM and MATLAB testing environment and viceversa.

Thanks to this component, Polarion ALM launches automatically the selected Test Cases in the MATLAB testing environment.

---

Then, the Test Case Results are automatically imported into the ALM as soon as the test cases execution terminates.

We decided to implement this component by exploiting the features provided by the Jenkins automation engine\textsuperscript{5}. Jenkins is a valid support for building pipelines involving the execution of heterogeneous third-party tools in scenarios of continuous integration, automated testing, or continuous delivery. Polarion itself suggests to exploit Jenkins for building automated testing toolchains, to this aim Polarion provides a suitable connector, named PJ Connector\textsuperscript{6}, allowing the automatic interaction of the ALM with Jenkins.

Figure 5 shows details about the design of the overall architecture. The Coordinator component has been implemented as a Jenkins based Master/Slave architecture. The Jenkins Master component interacts with the Polarion ALM component by means of two connectors \textsuperscript{[21]}. The Execute Test Run connector is a call connector exposed by the Jenkins Master component and was realized exploiting the Jenkins Remote access API. This connector is exploited by the ALM for launching a Test Run. The Jenkins Master requires the data access connector Export Test Run provided by Polarion ALM for querying the Test Cases belonging to the Test Run to be launched.

The Jenkins Master delegates the execution of all the Test Cases belonging to a Test Run to one of the Jenkins Slaves by exploiting the Dispatch Jenkins connector. Each slave component is responsible for interacting with just one testing environment. This is guaranteed by the two connectors exposed by each testing environment. The Execute Test Cases call connector allows the Jenkins slave to launch one or more test cases on the MATLAB platform. The Read Test Results data access connector, is used by the slave to load, from the testing environment, the results of the tests execution. The slave exploits the two data connectors Update Test Runs and Update Test Case provided by the ALM for updating the status of both the Test Runs and the Test Cases.

All the connectors exposed by Polarion ALM are data access connectors and were developed exploiting the APIs exposed by Polarion itself\textsuperscript{7} to access its stored elements.

\textsuperscript{5}https://jenkins.io/
\textsuperscript{6}http://www.emerasoft.com/pj-polarionjenkins-connector/
\textsuperscript{7}https://wiki.jenkins-ci.org/display/JENKINS/Remote+access+API
\textsuperscript{8}http://almdemo.polarion.com/polarion/sdk/index.html

4.3 Extending the ALM GUI

In order to overcome the limitations of the previous process, where the Test Engineer had to interact with the GUIs of two different environments, we implemented a new GUI in the ALM offering a single point of access for the execution of the MIL testing process.

To this aim, we exploited the Wiki-based mechanism provided by Polarion ALM\textsuperscript{9} to extend its default Test Run GUI, used in the original process for defining the Test Run and for manually importing the Test Case Results.

We enriched this GUI by two additional interaction panels. The "Create Test Run" panel is reported in Figure 6(a). It shows the Test Cases available for a given SWC Release and allows the Test Engineer to select the Test Cases to be run. Moreover, the panel shows the status (i.e., Failed, Passed, Blocked, Waiting) of all these Test Cases.

The "Test Run Execution", shown in Figure 6(b), is structured in several sections. The testing environment can be configured through the "Test Environment Configuration" section and the Test Run execution can be launched by the "Execute Test Run" button. In addition, its "Test Run Status" section offers an overview about the status of all the launched Test Cases and the "Activity" section reports the last activities performed by the users on the current Test Run.

5 A VALIDATION EXPERIMENT

We conducted a case study \textsuperscript{[19]} in the considered automotive context to evaluate whether the proposed solution was able to mitigate
The process drawbacks highlighted in Section 3. More in detail, the study aimed at answering the following Research Questions:

RQ1 How does the adoption of the proposed solution affect the process rapidity?

RQ2 How does the adoption of the proposed solution influence the effectiveness of the traceability links management related to the process?

RQ3 How does the adoption of the proposed solution impact on the process visibility from the point of view of the Project Manager?

RQ4 How does the adoption of the proposed solution impact the process acceptability from the point of view of the Test Engineers?

To answer these RQs, we executed the MIL Testing process using the new toolchain (hereafter “new process”) on a number of SWC Releases that had been already tested in the past in the former infrastructure (from now on “original process”). Archival data about these process executions was available, such as the time needed for the execution of each activity of the process, or the number of traceability links produced every day.

5.1 Objects

As objects of the study we considered SWC Releases belonging to the IPC of a segment B mass market vehicle produced by FCA. The selected SWCs were: the Speedometer responsible for filtering and presenting to the driver the vehicle speed information; the Tachometer that filters and presents to the driver the engine speed information and the Trip providing features for calculating and presenting to the driver some information related to one or more “trips”, i.e. segments of travelling selected by the driver, including travel duration and length, travel average speed and estimation of distance that can be traveled until the fuel tank will be empty. Table 1 shows for each SWC the number of the available Test Cases for its testing.

<table>
<thead>
<tr>
<th>Software Component</th>
<th>Number of Available Test Cases</th>
</tr>
</thead>
<tbody>
<tr>
<td>Speedometer</td>
<td>34</td>
</tr>
<tr>
<td>Tachometer</td>
<td>75</td>
</tr>
<tr>
<td>Trip</td>
<td>150</td>
</tr>
</tbody>
</table>

5.2 Subjects

The subjects of the study were employees of the EMEA SW Factory. We selected a Project Manager and three Test Engineers involved in the past execution of the original process on the IPC considered in the study. However, they worked on SWC Releases different from the ones we considered in the case study. The selected Project Manager had more than 15 years of experience managing more than 15 different projects. The selected test engineers had on average more than 20 years of experience in the MIL testing process.

5.3 Considered Metrics

Now we report the metrics we used to collect the evidences necessary for answering the RQs.

5.3.1 Rapidity. According to the definition of process rapidity provided by Sommerville [20], the rapidity of the MIL testing process represents how fast can the process deliver the test case execution results.

To evaluate this process characteristic, we measured the execution time of the whole testing process and of its composing activities.

The execution time of the overall MIL Testing process is 
\[ T = \sum_{i=1}^{4} T(A_i), \]
where \( T(A_i) \) represent the \( A_i \) process activity execution time.

5.3.2 Effectiveness of Traceability Links Management. To evaluate the effectiveness of the traceability link management in a given MIL process execution, we adopted the following two metrics:

- \#ITL: the number of incorrect traceability links between Test Cases and Test Case Results stored into the ALM
- \#MTL: the number of missing traceability links between Test Cases and Test Case Results.

These metrics were measured by a company employee who analyzed all the traceability links stored in the ALM during the MIL Testing process executions.

5.3.3 Visibility. According to Sommerville, the visibility of a process represents to what extent the process activities culminate in clear results, so that the progress of the process is externally
visible. Usually the progress of a process can be evaluated by observing the availability of specific artifacts delivered by the process and measuring the quantity of these artifacts that are available at specific points of the process execution.

To evaluate the process visibility, we designed a semi-structured interview and submitted it to the Project Manager.

The interview aimed at understanding which artifacts of the MIL testing process the Project Manager exploits to monitor the MIL testing process and if this information can be obtained when needed.

5.3.4 Acceptability. The acceptability of the MIL Testing Process represents if the defined process is acceptable to and usable by the engineers responsible for its execution.

To evaluate the acceptability of a given MIL process, we designed a semi-structured interview with the Test Engineers. The interview aimed at understanding the usability requirements of the Test Engineers about the MIL process, which process factors impacted the process usability from their point of view, and whether these usability conditions were met by the considered process.

5.4 Case Study Procedure

The case study was performed following a four steps procedure.

5.4.1 Subjects Allocation Step. The Project Manager randomly assigned to each Test Engineer the execution of the MIL Testing process related to one of the selected SWC Releases.

5.4.2 Execution Step. Each Test Engineer executed the testing process exploiting the proposed tool architecture. They were asked to complete the assigned process execution in a fixed time frame of five days, according to the standard company practices. For the new process executions, they had just to Create the Test Runs and to launch their execution through Polarion ALM.

5.4.3 Data Collection Step. At the end of the new process executions, the data needed to answer the defined Research Questions was collected.

The data needed to calculate the $T_i$, $T(A_i)$, $\#ITL$ and $\#MTL$ metrics were available in Polarion ALM. One of the authors queried the ALM to obtain the data regarding the new process executions. Archival data, about the original process executions, stored in the ALM were also queried.

Moreover, another author conducted the designed interviews with the study subjects and collected their answers.

5.4.4 Data Analysis Step. The collected data was analyzed to obtain the evidences to answer the defined Research Questions. The values of the defined metrics for the original and the new process executions were compared. To evaluate how the rapidity of the process was influenced we calculated the Speedup Percentage for the entire process, $SP$, defined according to the following formula:

$$SP = \frac{T_{original} - T_{new}}{T_{original}} \times 100$$  \hfill (1)

Moreover the $SP(A_i)$ for each $A_i$ activity was evaluated too. $SP(A_i)$ is defined as:

$$SP(A_i) = \frac{T_{original}(A_i) - T_{new}(A_i)}{T_{original}(A_i)} \times 100$$  \hfill (2)

$T_{original}$ and $T_{original}(A_i)$ represent the execution times for the entire process and for its $A_i$ composing activities in the original process obtained through the archival data whereas $T_{new}$ and $T_{new}(A_i)$ refers to the ones related to new process executions.

Table 2 shows the comparison results we obtained regarding the entire MIL Testing process (MIL) and the Create Test Run (CTR), Select Test Cases (STC), Launch Test Cases (LTC) and Import Test Case Results (ITCR) activities.

<table>
<thead>
<tr>
<th>Software Component</th>
<th>MIL</th>
<th>CTR</th>
<th>STC</th>
<th>LTC</th>
<th>ITCR</th>
</tr>
</thead>
<tbody>
<tr>
<td>Speedometer</td>
<td>7%</td>
<td>0%</td>
<td>73%</td>
<td>0%</td>
<td>79%</td>
</tr>
<tr>
<td>Tachometer</td>
<td>9%</td>
<td>0.1%</td>
<td>81%</td>
<td>0%</td>
<td>90%</td>
</tr>
<tr>
<td>Trip</td>
<td>14%</td>
<td>0.1%</td>
<td>91%</td>
<td>0.3%</td>
<td>96%</td>
</tr>
</tbody>
</table>

To estimate the differences in effectiveness between the original and the new process two metrics named Incorrect traceability links reduction percentage (ITLRP), and Missing traceability links reduction percentage (MTLRP) were proposed.

$$IT LR P = \frac{\#IT L_{original} - \#IT L_{new}}{\#IT L_{original}} \times 100$$  \hfill (3)

$$MT LR P = \frac{\#MT L_{original} - \#MT L_{new}}{\#MT L_{original}} \times 100$$  \hfill (4)

$MT LR P$ measures the percentage reduction of the number of incorrect traceability links of the new process, $\#IT L_{new}$, with respect the ones of the original process, $\#IT L_{original}$. It is represented by the following formula.

Whether these comparison metrics assume negative values it means that the proposed solution decreases the process performance.

Table 3 reports the comparison results we obtained at the end of the study.

<table>
<thead>
<tr>
<th>Software Component</th>
<th>ITLRP</th>
<th>MTLRP</th>
</tr>
</thead>
<tbody>
<tr>
<td>Speedometer</td>
<td>100%</td>
<td>100%</td>
</tr>
<tr>
<td>Tachometer</td>
<td>100%</td>
<td>100%</td>
</tr>
<tr>
<td>Trip</td>
<td>100%</td>
<td>100%</td>
</tr>
</tbody>
</table>

The data collected through the interviews carried out to compare the visibility and the acceptability of the two processes was submitted to a data triangulation step. More in detail, the interviews data was independently analyzed by different authors and the results were compared.
5.5 RQ Answers

On the basis of the collected data we were able to answer the proposed RQs.

RQ1: How does the adoption of the proposed solution affect the process rapidity? We were able to answer RQ1 on the basis of the data reported in Table 2.

The data highlights that the execution time of the overall process was not essentially affected by the adoption of the proposed architecture since the SP was always less than 14%.

On the other hand, analyzing Speedup Percentage of each process activity, we were able to understand that the Select Test Cases and Import Test Results are the activities affected since they shows SP values always greater than 73% and 79%, respectively. No substantial differences in terms of execution time were measured for the other two activities of the process. These results showed us that the adoption of the proposed architecture leads to process rapidity improvement.

RQ2: How does the adoption of the proposed solution influence the effectiveness of the traceability links management related to the process? Thanks to the analysis of the data reported in Table 3 we answered to RQ2. All the incorrect and missing links introduced in the original process due to their manual insertion were avoided in the new process executions thanks to the automation of the Import Test Case Results activity.

The adoption of the proposed architecture led to 100% reduction for both Incorrect and Missing traceability links for all the objects of the study, improving the effectiveness of the process traceability links management.

RQ3: How does the adoption of the proposed solution impact on the process visibility? The interview showed that the Project Manager usually exploits the percentage of the executed test cases for monitoring the progresses of the MIL testing processes involving the Software Component Releases of its own responsibility.

By analyzing his answers we were able to understand that in the original process, the Project Manager was able to monitor the progresses only on two defined milestones (i.e., the third and fifth day of the testing process). The other time he wanted to obtain the needed information he had to directly inquire the assigned Test Engineers.

On the other hand, in the new process the Project Manager is able to have a real time monitoring since the data are stored into the ALM as soon they are produced.

We report in the following some excerpts of the Project Manager statements that lead us to our conclusions:

"... with the original process, I could not check the information about component testing progress in real time because the information in the ALM did not reflect the real testing progress. To get real time information I had to directly inquiry the involved Test Engineers...".

"... Test Engineers should report test results every time they finish a test execution, but often they don’t have the time to do so due to strict deadlines. So we have established some milestones to enforce in the ALM the executed test case results. When the milestone is reached, all the test cases executed until that point must be reported in Polarion ALM. This happens two times a week..."

The results of this qualitative analysis showed us the evidences that the adoption of the proposed architecture leads to significant improvements in the visibility of the MIL Testing process from the point of view of the Project Manager.

RQ4: How does the adoption of the proposed solution impact the process acceptability? From this analysis of the collected answers we understood that the main factors affecting the acceptability of the process from the point of view of the Test Engineer are the number of different tools to exploit and the tedious manual activities they have to perform for the accomplishment of their work.

The data showed us that in the original process the Tester Engineers need to switch between two different tools (Polarion ALM and MATLAB Testing environment) and had to execute many time consuming and boring tasks. On the contrary, thanks to the new architecture they use a single tool and the most tedious tasks are automatically executed without any manual intervention.

To give evidence of this, we report excerpts of statements given by one of the interviewed Test Engineer: "... I have to use two tools for selecting and executing test cases ... the import of the results is fully manual .... this needs a great effort and distract me from the analysis of test results... With the new approach I select the test cases to execute in Polarion and launch their execution in the same tool... at the end of the test execution, I do not need to insert the test results back in Polarion since they are stored automatically."

The results confirmed us that the adoption of the proposed architecture was successfully accepted by the Test Engineers.

5.6 Threats to Validity

5.6.1 Internal Validity. This aspect of validity needs to be evaluated when causal relationship are examined. It defines how sure we can be that the treatment actually caused the outcome. In our case, the subjects experience could be another factor influencing the outcomes of the study. To mitigate this threat we selected subjects with different levels of experience. Moreover, a wider experimentation with other subjects should be carried out.

5.6.2 External Validity. External validity is related to what extent it is possible to generalize the findings and if these are of interest to people outside the reported case. A possible threat could be related to the application of the approach in a specific industrial domain and for a specific process. In order to minimize this threat a further experimentation in different industrial domains and for different processes will be performed.

5.6.3 Reliability. The reliability is concerned with to what extent the data and analysis are dependent on the specific researchers. Hypothetically, if another researcher later on conducts the same study, the result should be the same. To minimize this threat, the case study protocol and the artifact produced was reviewed by a group of researchers of the university. We reported, to the extent possible, the followed case procedure in this paper. Furthermore data collected through the interviews and their analysis were reviewed by case subjects in order to avoid possible misinterpretations.
6 CONCLUSIONS & FUTURE WORK

The ecosystem of artifacts, tools and platforms used in software development processes executed in industrial organizations can be very complex and heterogeneous. Software traceability represents a feasible solution for managing the complexity of these ecosystems and ALM tools provide a professional solution for storing and maintaining the traceability links among artifacts.

In the practice, ALM and legacy tools adopted in real software processes are not coordinated and integrated, and software organizations are forced to adopt tool integration solutions if they want to automate and improve the traceability management in their processes.

In this paper we reported our experience in developing a solution for integrating an ALM tool with the software tools used in MIL testing processes executed in FCA. The proposed solution exploits a Continuous Integration platform to implement a toolchain aimed at automating the manual activities of the process and at automatically creating the traceability links between the involved artifacts. This architectural solution is based on Master/Slave components, where the Master implements the process workflow, while each Slave interfaces a single tool of the toolchain. This architecture can be easily extended and reconfigured both to integrate new tools with the ALM, both to implement new software processes. The integration of another tool will require an additional Slave component and the design of the connectors needed for interfacing the tool. The implementation of a new process will require the modification of the Master component too.

The proposed architecture was evaluated by a case study conducted in the FCA EMEA SW Factory where we compared the results of the MIL testing process execution before and after the introduction of the new toolchain. The experimental results showed that the traceability management, the testing process rapidity, visibility and acceptability improved thanks to the new toolchain.

In future work we plan to extend our architecture by integrating additional tools with the ALM and to reuse the architecture for implementing different processes carried out both in the same context and in different industrial domains.

ACKNOWLEDGMENTS

This work was carried out in the context of the research project APPS4SAFETY (Active Preventive Passive Solutions for Safety - PON03PE_001593), partially funded by the Italian Ministry for University and Research (MIUR). Authors want to acknowledge the students Pasquale Curci and Luca Iandolo for their contribution in the implementation of the Tool Integration Architecture.

REFERENCES


Mining Collaboration Patterns of Software Development Processes Based on Trace Alignment

Dongjin Yu
School of Computer Science and Technology
Hangzhou Dianzi University
Hangzhou, China
yudj@hdu.edu.cn

Jiaojiao Wang
School of Computer Science and Technology
Hangzhou Dianzi University
Hangzhou, China
jjWang_hdu@126.com

ABSTRACT
Developing large-scale software usually involves the interaction of a great number of engineers over a long period. To discover the collaboration patterns from developing logs helps improve the software development processes. Traditional techniques of process mining can be employed to identify such patterns. Unfortunately, due to the high uncertainty of software development process, they tend to obtain "spaghetti" models which are difficult to comprehend or even misleading. As a remedy, in this paper we propose an approach to the discovery of collaboration patterns existing in software development process by aligning development logs. It considers not only the sequence of activities, but also the collaboration of actors who perform activities. Instead of using time-consuming graph mining techniques, it employs the trace alignment, which is much more straightforward. Moreover, unlike some traditional approaches, the discovered patterns are determined because we do not depend on the mined process model that is usually uncertain due to the unstructured nature of software development process. The experimental results based on a large dataset generated from CPNTools demonstrate the effectiveness of our approach.

CCS CONCEPTS
• Applied computing → Business process management;
• Software and its engineering → Software development process management

KEYWORDS
collaboration patterns; software development process; process mining; trace alignment; activities; actors

ACM Reference format:
http://dx.doi.org/10.1145/3084100.3084103

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from Permissions@acm.org.

IC SSP’17, July 5-7, 2017, Paris, France
© 2017 ACM. ISBN 978-1-4503-5270-3/17/07...$15.00
http://dx.doi.org/10.1145/3084100.3084103

1 INTRODUCTION
Developing large-scale software usually involves the interaction of a great number of engineers over a long period to achieve a common goal. Processes of such software development, however, are different from each other and tend to be far less structured than production processes due to complicated context. In order to help project managers to manage, understand, analyze and optimize their development processes, it is necessary to extract the process logs from Software Configuration Systems, such as CVS and Subversion, for constructing explicit process models automatically. We call this software process mining.

Conventional process mining techniques focus on the analysis of the control flow, as represented in flow charts or Workflow UML Activity Diagrams. They usually obtain an understandable process model by tools like ProM (http://www.promtools.org) and Disco (https://fluxicon.com/disco/) first and then recover the hidden reusable patterns. Unfortunately, these traditional techniques have problems while dealing with less structured processes like the software development because they may obtain "spaghetti" models that are difficult to comprehend.

As Whitehead et al. suggest, collaboration is pervasive throughout software engineering because almost all non-trivial software projects require effort and talent of many people working together [1]. Regardless of all known benefits, however, achieving effective team collaboration remains a great challenge. For instance, in a large software company with hundreds or even thousands of engineers, engineers in different departments may need to collaborate to accomplish the same project. There would be many choices when assigning engineers to different tasks of the project, some of which may result in high efficiency while some may not.

In our opinion, the engineers frequently participate in the same task or the successive tasks usually indicate they have the high collaboration efficiency. The reason may lie in that they share mutual understanding after a prolonged period of interaction. In other words, such collaboration patterns reveal the best practice in a company and applying these collaboration patterns can therefore improve the collaboration efficiency. Thus, we propose the concept of collaboration pattern that reveals not only the frequently occurring sequence of tasks (activities) but also how these tasks (activities) are usually assigned to engineers (actors) based on event logs of a software development process.
For example, in case of software change in a software company, Tom and Mary may together interact with stakeholders to discover their change requirements, usually followed by Jason proposing the change solution, then Smith writing and testing code for the change, and Jack evaluating and delivering the change finally. This case demonstrates the usage of “collaboration pattern” which involve five engineers (Tom, Mary, Jason, Smith and Jack) and six activities (discovering change requirements, proposing change solution, writing code, testing code, evaluating change and delivering the change).

Uncovering how to allocate human resources from event logs in a process can derive an organizational model based on the hidden resource information [2]. In other words, the integration of control flow and organizational information help well enlighten the features like the distribution of the resources on activities (who performed a particular task). In this paper, we present an approach to discover these mentioned collaboration patterns for better analysis. Our approach is inspired by the trace alignment that is further based on sequence alignment [2]. Sequence alignment is an essential method in bioinformatics that assists in unrelaving the structures of proteins and molecules [3] [4]. Likewise, software process mining also deals with sequences, i.e., traces of events stored in software development logs. The collaborative patterns of software development processes can be therefore discovered based on trace alignment from event logs.

The contributions of this paper are as following: (1) We elaborate an approach to the discovery of collaboration patterns involving not only the collaboration of activities but also actors who perform these activities, so that the software development process can be analyzed effectively. (2) Instead of the time-consuming graph mining techniques that are frequently used by traditional ones, our approach employs the trace alignment that is much more straightforward. (3) The result is determined because it does not depend on the mined process model that is usually uncertain due to the unstructured nature of software development process.

The rest of paper is structured as follows. After Section 2 discusses the related work, Section 3 illustrates the problem that we try to resolve and introduces some related definitions. Section 4 is devoted to introduce the details of the approach proposed to discover hidden collaboration patterns of software development process based on development logs in Software Configuration Systems. Afterwards, Section 5 discusses the case study that demonstrates the applicability of our approach. Finally, the last section concludes the paper and outlines the future work.

2 RELATED WORK

Discovering the implicit collaboration patterns helps analyze and improve the software development processes. Project managers are able to exploit such discovered patterns to assign tasks in a more reasonable way. Recently, researchers have dedicated a great amount of work on the approaches to the discovery of collaboration patterns from software development logs. Here, we briefly review the state-of-the-art works related to it.

Many researchers employ the techniques of process mining to obtain the patterns. Process mining is a developed research field that has provided a large number of mature methods and techniques [5] [6]. The recent works mainly focus on how to deal with unstructured processes, since the adoption of typical mining algorithm on such processes can likely originate too complex models or, on the contrary, oversimplified models. Hence, Weijters et al. introduce a heuristics driven process mining algorithm called HeuristicsMiner in ProM, which is an application to deal with noise and can be used to express the main behavior registered in an event log [7]. Similarly, Günther et al. propose Fuzzy Mining approach that is also integrated in ProM and allows for different faithfully simplified views of a particular process through the concept of a roadmap used as a metaphor [8]. However, these methods filter out the activities and behaviors with low-frequency to obtain the elements with high-frequency while mining unstructured processes, which usually leads to the result models that are difficult to analyze. In addition, the obtained process models are only the abstraction of event logs and fail to differentiate the iterations of tasks.

As a remedy for analyzing the unstructured software development processes, some researchers focus on the subprocess that may be highly structural in the whole unstructured process. For example, Raykenler et al. propose to segment the aligned traces and to form representative groups of sub-processes [9]. They exploit the tree of building blocks obtained to reflect the hierarchical organization that is established between the subprocesses. Besides, Claudia et al. employ a hierarchical clustering technique that is able to extract relevant patterns representing valuable knowledge about the collaboration process to mine and analyze unstructured process like software development process [10]. However, they only consider the cluster of activities but neglect the related actors.

The work based on trace alignment is also frequently used to uncover the common patterns of process execution [2]. Sequence matching is widely employed in biology science for comparing gene sequences and discovering the similarity between them. As we know, event logs containing heterogeneous sets of execution traces can lead to complex process models that try to account for very different behaviors. Bose and Van der Aalst propose an approach to align traces in a way that event logs can be explored easily [11]. They highlight some of the challenges such as multiple trace alignment and believe that this will open a new area of research within unstructured process mining. Although the above-mentioned works can inspire the analyzing of software development process to some extent, to the best of our knowledge, none of them is able to discover the hidden collaboration patterns based on trace alignment.

Collaboration patterns of software development processes reveal the collaboration inside a project team. Alves et al. argue that, by understanding the way collaboration is performed, participants and managers can better understand the development process in order to conduct their activities [12]. They propose an approach based on Social Network Analysis (SNA) to identify collaboration patterns in software development process instances that can be used as a resource for collaboration awareness and understanding. Also, many researches employ SNA to improve the efficiency of community collaboration for Open Source
Software (OSS) development [21-25]. Besides, Fan et al.
investigate the effect of collaboration process patterns on
teamwork efficiency (e.g. time cost) in the software development
setting, and develop a framework to identify frequent interaction
structures referred to as collaboration processes stored in a
software project tracking system [13]. Likewise, Jorge A. Colazo
examines the association between collaboration patterns, project
productivity and product quality by using archival data from
electronic sources related to OSS projects [24]. However, these
researches only consider the community members of a team
rather than integrating the actors with the activity in a specific
process.

3 PROBLEM AND DEFINITIONS
In order to demonstrate the problem to be resolved as well as the
notion of collaboration pattern, we illustrate a general process
model in Fig. 1 at the high level of software development, as an
example throughout the paper, which is adopted from [14].

![Software development process model.](image)

### 3.1 A Motivation Example

The software development process given in Fig. 1 follows the
typical “waterfall” model that consists of four fundamental stages
of software engineering. In practice, some activities may iterate
for several times particularly for the agile development. It is worth
noting that the granularity of activity can be either large or small,
and the model does not cover the activities like discovering
requirements, architectural design and risk management, which
are usually not recorded in event logs. Here, the
RequirementsEngineering stage involves activities of
WriteRequirements and ReviseRequirements, whereas the
Designing stage consists of activities of DesignSoftware and
ReviseDesign. The last two activities are WriteCode in
Implementation stage and TestCode in Testing stage. Such a simple
model provides only an abstract and high-level description of the
software development process. Of course, there are some variants
in real world as the development process is mainly driven by
different actors (engineers). For example, some activities can be
repeated for several times or executed in a different order. For
another example, the programmer may find that the design of a
transaction is not proper when coding in the Implementation stage.
Therefore, he has to ask the designer to revise the design
documents.

As we know, the software development process is such a
collaborative process that many activities are performed by more
than one actor. Moreover, the collaborative situations are not
identical under different circumstances. Unfortunately, the
traditional process mining techniques are difficult to obtain a
complete process model from process instances by mining event
logs either from control flow or organization perspectives [15].

For example, a group of team actors can perform the
DesignSoftware activity in different ways: sometimes they work
together on the whole activity, sometimes they are assigned to
finish a certain part of the activity. Consequently, a single activity
itself is a complex structure for process analysts.

Because there exists too little commonality among different
process instances and the frequent patterns, such as a certain
activity always performed by a certain engineer, in some
fragments of these process instances, we take the actors who
perform the activity into account while mining collaboration
patterns. In other words, we expect that different instances of the
process share some frequent sub-processes, characterized by
specific work practices. Meanwhile, the sub-process is always
represented as the fragment sequence in process trace. Therefore,
we propose an approach based on the sequence of execution trace
to discover the frequently occurring collaboration patterns.

### 3.2 Conceptual Basic

**Definition 1:** A software development process consists of a
flow of activities which are performed by some resources (e.g.,
actors) while developing software. Each time we start to develop a
new software, an instance of software development process is
generated.

**Definition 2:** An event is an instance of a certain activity
within a process instance, described by a set of event attributes.
The event logs consist of a series of events for one certain process
instance. As we can see from Table 1, an event e with InstanceID of
1 and EventID of 5 has different attributes, such as timestamp (e
occurs at 10:11:01 on 2012.10.26), actor (e is performed by M7 and
M9), activity (e corresponds to WriteCode), and event type (e is a
complete event, i.e., the complete time of the WriteCode is at
10:11:01 on 2012.10.26).

**Table 1:** Event logs of software development process

<table>
<thead>
<tr>
<th>Instance Event ID</th>
<th>Timestamp</th>
<th>Activity</th>
<th>Event Type</th>
<th>Actor</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2012.10.12</td>
<td>WriteRequirements</td>
<td>start</td>
<td>M3, M4, M5, M6</td>
</tr>
<tr>
<td>2</td>
<td>2012.10.17</td>
<td>ReviseRequirements</td>
<td>complete</td>
<td>M3</td>
</tr>
<tr>
<td>3</td>
<td>2012.10.19</td>
<td>DesignSoftware</td>
<td>start</td>
<td>M1, M2, M5</td>
</tr>
<tr>
<td>4</td>
<td>2012.10.25</td>
<td>ReviseDesign</td>
<td>complete</td>
<td>M6</td>
</tr>
<tr>
<td>5</td>
<td>2012.10.26</td>
<td>WriteCode</td>
<td>complete</td>
<td>M7, M9</td>
</tr>
<tr>
<td>6</td>
<td>2012.11.17</td>
<td>TestCode</td>
<td>start</td>
<td>M7</td>
</tr>
<tr>
<td>7</td>
<td>2013.01.12</td>
<td>WriteRequirements</td>
<td>complete</td>
<td>M3, M4, M5</td>
</tr>
</tbody>
</table>

**Definition 3:** A trace represents a time ordered sequence of all
events occurring in a process instance for a process, denoted by
$$T_i = \langle e_{i,0}, e_{i,1}, \ldots, e_{i,j}, \ldots, e_{i,T_i.length-1} \rangle,$$
in which $$e_{i,j}$$ indicates a jth
occurring event in process instance \(i\). Besides, the event \(e_{ij}\) happens just after the event \(e_{i,j−1}\). It is obvious that a set of events with the same \(\text{InstanceID}\) but different \(\text{EventID}\) belong to the same process instance. Generally speaking, we use the activity attribute of event to represent the event itself, i.e., \(T_i = \langle a_{i,0}, a_{i,1}, \ldots, a_{i,j−1}, a_{i,j}, \ldots, a_{i,T_{\text{length}_i}} \rangle\). For example, we can denote the trace with \(\text{InstanceID}\) of 1 in Table 1 as \(\langle \text{WriteRequirements}, \text{ReviseRequirements}, \text{DesignSoftware}, \text{ReviseDesign}, \text{WriteCode}, \text{TestCode} \rangle\).

Definition 4: The activity and the actor(s) who perform(s) the activity can be integrated into an activity-actor entity, or entity in short, defined as \(a_r\) which indicates the activity \(a\) is performed by the actor(s) \(r\). For example, \(\text{WriteCode}\|\text{M7}, \text{M9}\), or \(\text{WC}\|\text{M7}, \text{M9}\) in short, indicates \(\text{M7}\) and \(\text{M9}\) perform the \(\text{WriteCode}\) activity.

Definition 5: An entity trace represents a time ordered sequence of all entities belonging to the same process instance, denote by \(ET_i = \langle a_{i,0}, a_{i,1}, \ldots, a_{i,j−1}, a_{i,j}, \ldots, a_{i,T_{\text{length}_i}} \rangle\). For example, we can describe the entity trace with \(\text{InstanceID}\) of 1 in Table 1 as \(ET_i = \langle \text{WR}\|\text{M3}, \text{M4}, \text{M5}, \text{M6}; \text{RR}\|\text{M3}, \text{DS}\|\text{M1}, \text{M2}, \text{M5}; \text{RD}\|\text{M6}; \text{WC}\|\text{M7}, \text{M9}; \text{TC}\|\text{M7} \rangle\).

3.3 Sequence Alignment

Trace alignment is a mapping method developed in the bioinformatics discipline for optimal alignment sequences of DNA and protein building blocks. An alignment is a sequence of pairs, either of elements of two sequences, or of an element of the first and a "gap" in the second sequence, or of a "gap" in the first and an element of the second sequence. For example, the sequences GCATTCA and GATTACA may be aligned as ("-" represents a gap):

\[
\begin{align*}
G & \text{C} & \text{A} & \text{T} & \text{T} & \text{C} & \text{A} \\
G & \text{A} & \text{T} & \text{T} & \text{A} & \text{C} & \text{A}
\end{align*}
\]

Similarly, the entity traces can also be aligned to discover the common sub-entity-traces. For example, a pair of two entity traces: \(\langle \text{WR}\|\text{M3}, \text{M4}, \text{M5}, \text{M6}; \text{RR}\|\text{M3}, \text{DS}\|\text{M1}, \text{M2}, \text{M5}; \text{RD}\|\text{M6}; \text{WC}\|\text{M7}, \text{M9}; \text{TC}\|\text{M7} \rangle\) and \(\langle \text{WR}\|\text{M3}, \text{M4}, \text{M5}, \text{M6}; \text{RR}\|\text{M3}, \text{DS}\|\text{M1}, \text{M2}, \text{M5}; \text{RD}\|\text{M6}; \text{WC}\|\text{M7}, \text{M9}; \text{TC}\|\text{M7} \rangle\) can be aligned to obtain the following alignment ("-" represents a gap):

\[
\begin{align*}
\text{WR}\|\text{M3}, \text{M4}, \text{M5}, \text{M6} & \text{RR}\|\text{M3}, \text{DS}\|\text{M1}, \text{M2}, \text{M5} & \text{RD}\|\text{M6} & \text{WC}\|\text{M7}, \text{M9} & \text{TC}\|\text{M7} \\
\text{WR}\|\text{M3}, \text{M4}, \text{M5} & \text{RR}\|\text{M3}, \text{DS}\|\text{M1}, \text{M2}, \text{M5} & \text{RD}\|\text{M6} & \text{WC}\|\text{M7}, \text{M9} & \text{TC}\|\text{M7}
\end{align*}
\]

Definition 6: An activity block is a set of entities related to the same activity, defined as:

\[ AB_a = \{ a_{r_1}, a_{r_2}, \ldots, a_{r_j}, \ldots, a_{r_n} \} \quad (1) \]

In general, an ordered and repeatable collection \(C_{AB}\) that event logs of a software development process. Since aligned entity traces form a matrix that consists of entities or gaps, any activity blocks can be obtained by dividing this matrix according to the different activities related to entities. Fig. 2 shows the matrix that can be divided into eight activity blocks, namely \(C_{AB} = \langle \text{AB}_W, \text{AB}_R, \text{AB}_D, \text{AB}_R, \text{AB}_D, \text{AB}_W, \text{AB}_C, \text{AB}_T \rangle\). Meanwhile, the activity block, such as \(\text{AB}_W\), denotes a set of entities sharing the same activity, such as \(\text{WR}\), or \(\text{WriteRequirements}\).

![Figure 2: Activity blocks of alignment matrix.](image_url)
4 APPROACH OVERVIEW

The overall architecture of our approach is shown in Fig. 4. First, we generate entity traces based on event logs of software development process, which are further aligned to establish an alignment matrix (Step A). Afterwards, we segment the alignment matrix for activity blocks (Step B). Finally, we discover the collaboration patterns of both activity blocks and entities from the alignment matrix (Step C). Our approach involves three main steps that are described as follows.

4.1 Standardizing and Preprocessing of Event Logs
Software Configuration Management (SCM) systems such as CVS and Subversion are used to track the whole software development process with abundant event logs. However, these event logs in real-life are usually neither structured nor standard, and even full of noise due to the improper operations. Hence, it is necessary to extract the key fields of the original event logs to derive the standard logs. Here, we employ ProM and Disco to perform the standardizing and preprocessing of event logs automatically. Firstly, the logs are marked if their attribute values are incorrect, which may include the following cases: the finish time of one activity exceeds the finish time of the whole process instance, the final activity of one process instance is not the terminal one, the execution time of two sequential activities of the same process instances are overlapped, and so on. Meanwhile, we collect all the logs with missing attribute values and try to supplement the missing ones based on their relevant logs. Afterwards, we delete all logs marked with incorrect and whose missing attribute values cannot be recovered. In this way, we can obtain a list of standardized and clean logs with values of all attributes (i.e., Instance ID, Event ID, Timestamp, Activity, EventType, and Actor) filled.

4.2 Alignment of Entity Traces
In our approach, we employ the Needleman-Wunsch algorithm [19], which is frequently used in bioinformatics for aligning protein or nucleotide sequences, to align two entity traces. The Needleman-Wunsch algorithm guarantees a mathematically optimal alignment, given a table of scores for matches and mismatches between the elements of aligning traces. However, as for our approach, more than two traces are required to align. Obviously, the alignment of multiple entity trace is not the same as that of pair traces. On the other hand, attempts at generalizing dynamic programming, such as extending the Needleman-Wunsch algorithm to multiple alignments, are limited to only small numbers of short traces [16].

Here, we adopt a progressive method for aligning multiple traces. More specifically, we build up a multiple alignment progressively by a series of pairwise alignments, following the branching order in a tree. In this way, we can first align the most closely related traces, gradually adding in the more distant ones. After that, we obtain the aligned longest trace, based on which we construct the alignment matrix. These steps can be described in detail as following.

In order to obtain a guide tree of multiple traces based on the similarity between them in terms of a progressive approach, we adopt the idea of Kruskal algorithm [17] by viewing the trace as the node and the similarity of Levenshtein distance as the weight of edge between nodes in an undirected graph, which is shown in Algorithm 1. The key of our approach is how to determine the
features used to characterize the similarity between traces. These features depend on the knowledge of the domain and the intention of analysis [18]. Here, we choose the Levenshtein distance [20] as the similarity metric in our approach. Based on the guide tree, we employ the Needleman-Wunsch algorithm to align traces for a longest trace. Moreover, a mapping table of entity-character encoding is derived. After that, we align the entity traces based on the longest trace. Fig. 5 shows an example of generating a guide tree and aligning multiple entity traces which involve 5 entity traces given in Fig. 2.

As Algorithm 1 shows, we first compute the edit distance between every two traces and sort them in DIST_{n×n} (Lines 1-3). Afterwards, we utilize the idea of Kruskal algorithm for generating minimum spanning tree to build the guide tree of process traces, and then apply Needleman-Wunsch algorithm to align traces by combining two clusters of traces progressively with the most-matched pairs first (Lines 4-16). Subsequently, we obtain the longest trace from the aligned traces (Line 17). Finally, we align entity traces based on the longest obtained traces and derive the matrix (Lines 18-29). Since the entity traces in alignment matrix has been aligned now, we can segment the alignment matrix for activity blocks as the entities with the same activity locate in the same column.

4.3 Discovering Collaboration Patterns

Based on the alignment matrix, we then search for the collaboration patterns represented as the hierarchical structure of activity blocks and entities, as Algorithm 2 illustrates. As we all know, developing different softwares may share the similar process model in the view of activity sequence. Therefore, we can segment the alignment matrix according to the shared activities of entities. In Algorithm 2, we firstly extract all the distinct successive activity block sequences represented as column index from the alignment matrix (Lines 2-5). Based on them, we then extract the distinct successive entity sequences and compute their

---

**Algorithm 1: Align Entity Traces**

**Input:** T = \{T_i\} = \{< a_{i,0}, a_{i,1}, ..., a_{i,j}, ..., a_{i,T_{i}+1} >\}, i = 1,2,...,n // n traces

ET = \{ET_i\} = \{< a_{i,0} | r_{i,0}; a_{i,1} | r_{i,1}; ...; a_{i,j} | r_{i,j}; ...; a_{i,T_{i}+1} >\}, i = 1,2,...,n

// n entity traces corresponding to above n traces

**Output:** AM_{n×n} // the alignment matrix of n entity traces

1: FOR i = 1 to n, j = 1 to n DO
2: \[\text{DIST}_{ij} \leftarrow \text{distance}(T_i, T_j)\] //calculate the distance between each pair traces (T_i, T_j)
3: END FOR

4: FOR i = 1 to n DO
5: \[\text{clusterList}[i] \leftarrow T_i\] //put each trace T_i to a distinct cluster clusterList[i]
6: END FOR

7: DO
8: \[\text{traceMap} \leftarrow \text{Tid,clusterId} \leftarrow < T_i, i \] //map the trace with the cluster
9: END DO

10: DO
11: \[\text{clusterNum} = n\] to 1 DO
12: \[< T_j, T_j >\] the trace pair with smallest distance from two clusters according to DIST_{n×n}
13: \[C_1 \leftarrow \text{traceMap.get}(T_j)\] //obtain the identifier of cluster of T_j
14: \[C_2 \leftarrow \text{traceMap.get}(T_j)\] //obtain the identifier of cluster of T_j
15: END DO

16: DO
17: \[\text{longestTrace} \leftarrow \text{getLongestTrace(clusterList[C_i])}\]
18: \[m \leftarrow \text{length}(\text{longestTrace})\]
19: \[\text{Initialize AM}_{n×n}\]
20: FOR i = 1 to n DO
21: \[\text{id}x \leftarrow 1\]
22: \[\text{FOR j = 1 to m DO}\]
23: \[\text{IF} idx \leftarrow \text{length}(ET_j) \&\& ET_i, a_i,idx = \text{longestTrace}_j \text{ THEN}\]
24: \[AM_{i,j} \leftarrow ET_{i,j}\]
25: \[idx + +\]
26: \[\text{ELSE}\]
27: \[AM_{i,j} \leftarrow " - " \] // add the gap
28: \[\text{END FOR}\]
29: \[\text{END FOR}\]
5.1 The Event Log
In order to obtain the entity traces of software development process, a Petri net with organization perspective and its execution event logs should be built at first. Here, we exploit CPNTools that supports modeling and simulating execution as well as colored Petri net analysis to create a colored Petri net with some constraints in Table 2. The colored Petri net represents the software development process introduced in Fig. 1. Additional, we only focus on a subset of all possible behaviors of each activity and their probabilities respectively. Because it is assumed possible to recognize some frequent patterns among all possible ones, we consider the cases of all alternative performing patterns for each activity of software development process. As shown in Table 2, the activity WriteRequirements can be performed by actors of M3, M4, M5 and M6 working together on the same document with a probability of 0.4, by M3 and M6 with a probability of 0.3. Furthermore, the collaboration patterns can be performed by actors of M3, M4 and M5 with a probability of 0.3 and by M3 and M6 with a probability of 0.3.

Based on the probability table and the expanded basic process model given in Fig. 1, we build a colored Petri net (CP-net) and then simulate it in CPNTools. To create the result MXML file (a file similar to XML file which can be used to represent event log), we extend CPNTools by implementing the ML functions firstly. Afterwards, we use the plug-in of ProMimport (http://www.promtools.org/promimport/) to bundle the logged files into a MXML file. In short, two steps are necessary to create MXML logs: (i) Make sure a CP-net invokes the set of ML functions that will create logs for every case. This step involves modifying the declarations of the CP-net and the input/output/action transition inscriptions. (ii) Use the plug-in of ProMimport to group the logs for the individual cases into a single MXML file of event logs. In this way, we can obtain the MXML logs whose statistics is shown in Table 3. In order to simulate the real-life software development process, we expand the original process model in Fig. 1 to a more complex model as shown in Fig. 6 and then build a CP-net based on it. We execute the built CP-net for 10,000 times, and

---

**Algorithm 2: Discover Collaboration Patterns**

**Input:** \( AM_{n \times m} \) // the alignment matrix of \( n \) entity traces

\( k_{AB}, k_{E} \) // the thresholds of frequently occurring activity block sequences and sub-entity-traces respectively

**Output:** \( M_{\text{blockSequence}} \) // the list of collaboration pattern of activity blocks

\( M_{\text{entitySequence}} \) // the list of collaboration pattern of entities

```
01: Initialize colIndexList[n], colSeqMap < colSeq, count >, entitySeqMap < entitySeq, count >
02: FOR i = 1 to n DO
03:     colIndexList[i] ← the column index of \( i^{th} \) entity trace in \( AM_{n \times m} \) with not all having ‘.’
04: END FOR
05: //store the mapping between distinct activity block sequences and their frequency
06: colSeqMap < colSeq, count > ← all activity block sequences according to colIndexList
07: //store the mapping between all distinct entity sequences and their frequency
08: entitySeqMap < entitySeq, count > ← entity sequences with their frequencies according to colSeq
09: END FOR
10: FOR each < colSeq, count > of colSeqMap DO
11:     IF count >= \( k_{AB} \) * n THEN
12:         \( M_{\text{blockSequence}} \) ← all activity block sequences of colSeq
13:     END FOR
14: FOR each < entitySeq, count > of entitySeqMap DO
15:     num ← the total of activity block sequences of entitySeq in colSeqMap
16:     IF count >= \( k_{E} \) * num THEN
17:         \( M_{\text{entitySequence}} \) ← entitySeq
18: END FOR
```

---

frequency (Lines 6-8). Afterwards, we filter out and store the frequent successive activity block sequences according to \( k_{AB} \) (Lines 10-11). Likewise, we filter out and store the frequent sub-entity-traces representing the expected collaboration patterns according to \( k_{E} \) in these frequent activity block sequences (Lines 12-16). These outputs can be described as the hierarchical graphs shown in Fig. 3.

5 A CASE STUDY
In this section, we present a case to demonstrate the applicability of our proposed approach. First, we exploit CPNTools (http://cpntools.org/), a tool for color Petri nets modeling and simulation, to generate the synthetic event logs based on Table 2. Then, we align the generated entity traces from event logs to obtain an alignment matrix, and segment it for activity blocks. Finally, we extract the collaboration patterns with regard to activities and actors from the alignment matrix based on activity blocks. The case runs on the server with six cores of E5-2620 2.00GHz, 64GB memory, and Windows.
Table 2: The activities and actors with their occurrence probabilities

<table>
<thead>
<tr>
<th>Activity</th>
<th>Actors</th>
<th>Occurrence probability</th>
</tr>
</thead>
<tbody>
<tr>
<td>WriteRequirements</td>
<td>M3, M4, M5, M6</td>
<td>0.4</td>
</tr>
<tr>
<td></td>
<td>M3, M4, M5</td>
<td>0.3</td>
</tr>
<tr>
<td></td>
<td>M3, M6</td>
<td>0.3</td>
</tr>
<tr>
<td>ReviseRequirements</td>
<td>M3</td>
<td>0.2</td>
</tr>
<tr>
<td></td>
<td>M4</td>
<td>0.3</td>
</tr>
<tr>
<td></td>
<td>M5</td>
<td>0.2</td>
</tr>
<tr>
<td></td>
<td>M6</td>
<td>0.3</td>
</tr>
<tr>
<td>DesignSoftware</td>
<td>M1, M2, M5</td>
<td>0.4</td>
</tr>
<tr>
<td></td>
<td>M5</td>
<td>0.3</td>
</tr>
<tr>
<td></td>
<td>M1, M2</td>
<td>0.3</td>
</tr>
<tr>
<td>ReviseDesign</td>
<td>M1</td>
<td>0.3</td>
</tr>
<tr>
<td></td>
<td>M2</td>
<td>0.3</td>
</tr>
<tr>
<td></td>
<td>M5</td>
<td>0.2</td>
</tr>
<tr>
<td></td>
<td>M6</td>
<td>0.2</td>
</tr>
<tr>
<td>WriteCode</td>
<td>M7, M8, M9, M10</td>
<td>0.5</td>
</tr>
<tr>
<td></td>
<td>M7, M9</td>
<td>0.3</td>
</tr>
<tr>
<td></td>
<td>M8</td>
<td>0.2</td>
</tr>
<tr>
<td>TestCode</td>
<td>M7</td>
<td>0.3</td>
</tr>
<tr>
<td></td>
<td>M8</td>
<td>0.3</td>
</tr>
<tr>
<td></td>
<td>M9</td>
<td>0.2</td>
</tr>
<tr>
<td></td>
<td>M10</td>
<td>0.2</td>
</tr>
</tbody>
</table>

Table 3: The statistics of synthetic event logs

<table>
<thead>
<tr>
<th>#Traces</th>
<th>#Events</th>
<th>#Entity specification</th>
</tr>
</thead>
<tbody>
<tr>
<td>10,000</td>
<td>106,500</td>
<td>21</td>
</tr>
</tbody>
</table>

Figure 6: The software development process model used in experiment.

obtain the MXML file with 10,000 traces and 106,500 events as well as 21 distinct entities.

5.2 Generating Entity Traces

Afterwards, we import the generated MXML file into ProM to obtain entity traces in the format of MXML displayed in Fig. 7. For this purpose, we need to exploit the specific format-conversion plugins to modify the attribute in MXML file and then capture the entity trace by integrating the resource attribute into the activity attribute.

5.3 Aligning Entity Traces

According to Algorithm 1, we obtain the guide tree (Fig. 8) and the alignment matrix (Fig. 9). As we can see from Fig. 9, the obtained trace alignment clearly shows that entities sharing the same activity locate in the same column. For example, the encoded entities n, b and r which correspond to WR|M3, M6, WR|M3, M4, M5 and WR|M3, M4, M5, M6 respectively share the same activity WR.

5.4 Discovering Collaboration Patterns

Finally, we exploit the proposed approach to detect the collaboration patterns and then represent them with the hierarchy diagrams as shown in Fig. 10. Here, we set $k_{AB}$ and $k_{P}$ to 0.65 and 0.05 respectively. In this way, we can obtain 10 collaboration patterns of activity blocks as well as 95 collaboration patterns of entities. The relationship between activity blocks displayed in Fig. 10(a) reveals the frequent orders of activities and some successive activities with high collaboration in software development process. For example, the ReviseRequirements activity often happens after the WriteRequirements activity (CP1) which may be further followed by the DesignSoftware activity (CP6), the WriteCode activity frequently happens after the accomplishment of ReviseDesign (CP2), and the ReviseRequirements activity may repeat again after the DesignSoftware activity under some circumstances (CP9). The collaboration patterns displayed in Fig. 10(b) demonstrate the cases that more than one actor often perform the same activity, which conforms to the distribution of the entity code.

Figure 7: The entity trace file in the form of MXML.

Figure 8: The part of obtained guide tree.

Figure 9: The part of alignment matrix.
the probabilities shown in Table 2. In fact, for two thirds of complicate activities (e.g. WriteRequirements, DesignSoftware and WriteCode) of the software development process, the probability of assigning an activity to a group of members is higher than that of assigning to only a single member.

From the perspective of domain analysis, it is also noteworthy that the ReviseRequirements activity is performed by only one actor, as shown in all collaboration patterns of entities. On the contrary, the WriteRequirements activity is often performed by a group of members. In addition, some more complex collaboration patterns can be also found. For example, M1, M2 and M5 perform the DesignSoftware activity firstly, followed by M1 or M2 who revises design and followed by M7, M8, M9 as well as M10 who write code finally. Moreover, we can discover that the ReviseRequirements activity repeat again if no one who performs the WriteRequirements activity joins the DesignSoftware activity by comparing CP6 with CP9 in Fig. 10(b). Obviously, these discovered frequently occurring patterns give the project manager the hints about the assignment of future tasks to ensure the high team efficiency.

### 5.5 Comparison and Discussion

Some state-of-the-art methods can also be employed to obtain the similar findings. However, most of them consider only the collaboration of activities but ignore the actors who perform activities. In addition, to the best of our knowledge, none of them investigates collaboration patterns based on entity trace alignment from event logs in software development process by integrating the actors with activities. Moreover, it is obvious that trace alignment in our approach runs much faster than those based on sub-graph mining. Last but not least, we can distinguish the same sub-entity-trace that locates in different positions in an entity trace.

A similar work is presented by Claudia et al. In [14], they introduce an approach based on the process instance graph as well as the casual relation obtained from the mined process model for unstructured processes like software development. They employ the Frequent Sub-graph Mining (FSM) algorithm to discover frequent sub-graphs representing collaboration patterns in these process instance graphs. However, as they point out, the building of process instance graphs needs to depend on the mined process model. Unfortunately, there is no algorithm widely accepted for mining models of unstructured processes. In addition, they acknowledge that the structure of mined process model can influence the structure of obtained collaboration pattern. For example, one will not derive the concurrent structure of collaboration pattern if the algorithm cannot mine the process model of concurrent structure. Hence, the result of their proposed approach is uncertain. Furthermore, the FSM method in a given process instance graph dataset can be a time-consuming task.

On the other hand, Fan et al. also propose an analytical framework to study the effect of collaboration patterns in software development processes based on the theory of frequent sub-graphs mining [13]. They define collaboration patterns as recurrent process substructures and obtain some sub-graphs with sequence as well as loop structure. However, the discovered collaboration patterns with complex structure are prone to over-fit the executed process instances.

As we know, one can find activity blocks as well as activity block sequences by loading the event log into a tool like Disco or Minit and computing the direct-follow relation between activities. However, as for the unstructured process like software development, such tools fail to discover collaboration patterns inside the iterations. In particular, they cannot find the patterns hidden from a certain round of iteration. On the contrary, our approach focuses on the ordered traces and therefore can retrieve what is inside the iteration by smoothing and extending the iteration. In addition, while dealing with unstructured processes like the software development, conventional process mining techniques may obtain “spaghetti” model which is prone to produce decentralized traces, and such traces are difficult to recover the original model again. Therefore, our approach starts from the trace logs for the discovery of collaboration pattern. It is
unnecessary to construct the process model that the traditional process mining techniques usually rely on.

6 CONCLUSION AND FUTURE WORK

In this paper, we propose a new approach based on trace alignment to discover collaboration patterns from development logs to improve the collaboration of a software development process to some extent. Firstly, we combine a series of both activities and actors into entity traces, and then construct an alignment matrix. Afterwards, from the alignment matrix, we obtain the frequently occurring collaboration patterns described in hierarchical graphs. Such derived patterns can be used to interpret and illustrate the collaborative cases in software development process, which will be of great significance for assignment of tasks in the effective management of software development.

The case based on the sequential structure is somewhat simple and high-level. Nevertheless, what we study can be also applicable to the complex processes because either simple or complex processes share the same characteristics by considering the trace logs. However, there are still some limits of our approach. For instance, it is lack of effective verification and comparison. In the future, we will apply our approach in different real-life scenarios to further evaluate its effectiveness. Meanwhile, we will introduce a compatibility model based on the discovered patterns. By using the compatibility model, we can then assign the actors to different tasks for maximum cooperation in future projects.

ACKNOWLEDGMENTS

The work is supported by National Natural Science Foundation of China (No. 61100043), the Key Science and Technology Project of Zhejiang Province (No. 2017C01010, No. 2016F50014 and No. 2015C01040). The authors would also like to thank anonymous reviewers who made valuable suggestions to improve the quality of the paper.

REFERENCES

[23] S A., Licorish. 2013. Collaboration patterns of successful globally distributed agile software teams: The role of core developers. Auckland University of Technology. DOI: http://hdl.handle.net/10922/5973
Tracing Requirements in Software Design

Zeheng Li
Southern Methodist University
Dallas, TX 75275-0122, USA
zehengl@smu.edu

Mingrui Chen
Southern Methodist University
Dallas, TX 75275-0122, USA
mingruic@smu.edu

LiGuo Huang
Southern Methodist University
Dallas, TX 75275-0122, USA
lghuang@smu.edu

Vincent Ng
University of Texas at Dallas
Richardson, TX 75083-0688, USA
vince@hlt.utdallas.edu

Ruili Geng
Spectral MD
Dallas, TX 75201, USA
rgeng@smu.edu

ABSTRACT
Software requirement analysis is an essential step in software development process, which defines what is to be built in a project. Requirements are mostly written in text and will later evolve to fine-grained and actionable artifacts with details about system configurations, technology stacks, etc. Tracing the evolution of requirements enables stakeholders to determine the origin of each requirement and understand how well the software’s design reflects to its requirements. Reckoning requirements traceability is not a trivial task, we focus on applying machine learning approach to classify traceability between various associated requirements. In particular, we investigate a 2-learner, ontology-based approach, where we train two classifiers to separately exploit two types of features, lexical features and features derived from a hand-built ontology. In comparison to a supervised baseline system that uses only lexical features, our approach yields a relative error reduction of 25.9%. Most interestingly, results do not deteriorate when the hand-built ontology is replaced with its automatically constructed counterpart.

CCS CONCEPTS
• Software and its engineering → Requirements analysis;
  Software design engineering;

KEYWORDS
Requirements Traceability, Software Design, Machine Learning

1 INTRODUCTION
Evolution and refinement of requirements is guiding the software system development process. Requirement specifications, mostly documented in natural language, are refined with additional details of design and implementation information as a software project move forwards in its development life cycle. An important task in software development process is requirements traceability, which is concerned with linking requirements in which one is a refinement of the other. Being able to establish traceability links allows stakeholders to find the origin of each requirement and track every change that has been made to it, and ensures the continuous understanding of the problem that needs to be solved so that the right system is delivered.

In practice, one is given a set of high-level (coarse-grained) requirements and a set of low-level (fine-grained) requirements, and requirements traceability aims to find for each high-level requirement all the low-level requirements that refine it. Note that the resulting mapping between high- and low-level requirements is many-to-many, because a low-level requirement can potentially refine more than one high-level requirement.

As an example, consider the three high-level requirements and two low-level requirements shown in Figure 1 about the well-known Pine email system. In this example, three traceability links should be established: (1) HR01 is refined by UC01 (because UC01 specifies the shortcut key for saving an entry in the address book); (2) HR02 is refined by UC01 (because UC01 specifies how to store contacts in the address book); and (3) HR03 is refined by UC02 (because both of them are concerned with the help system).

From a text mining perspective, requirements traceability is very challenging task. First, there could be abundant information irrelevant to the establishment of a link in one or both of the requirements. For instance, all the information under the Description section in UC01 is irrelevant to the establishment of the link between UC01 and HR02. Worse still, as the goal is to induce a many-to-many mapping, information irrelevant to the establishment of one link could be relevant to the establishment of another link involving the same requirement. For instance, while the Description section is irrelevant to linking UC01 and HR02, it is crucial to linking UC01 and HR01. Above all, a link can exist between a pair of requirements (HR01 and UC01) even if they do not possess any overlapping or semantically similar content words.

Virtually all existing approaches to the requirements traceability task were developed in the software engineering (SE) research

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

ICSSP'17, Paris, France
© 2017 ACM. 978-1-4503-5270-3/17/07... $15.00
DOI 10.1145/3084100.3084102
community. Related work on this task can be broadly divided into two categories. In manual approaches, requirements traceability links are recovered manually by developers. Automated approaches, on the other hand, have relied on information retrieval (IR) techniques, which recover links based on computing the similarity between a given pair of requirements. Hence, such similarity-based approaches are unable to recover links between those pairs that do not contain overlapping or semantically similar words or phrases.

In light of this weakness, we recast requirements traceability as a supervised binary classification task, where we classify each pair of high- and low-level requirements as positive (having a link) or negative (not having a link). We represent each pair of requirements using two types of features. First, we employ word pairs, each of which is composed of a word taken from each of the two requirements involved. These features will enable the learning algorithm to identify both semantically similar and dissimilar word pairs that are strongly indicative of a refinement relation between the two requirements, thus overcoming the aforementioned weakness associated with similarity-based approaches.

Next, we employ features derived from an ontology hand-built by a domain expert. The ontology contains only a verb clustering and a noun clustering: the verbs are clustered by the function they perform, whereas a noun cluster corresponds to a (domain-specific) semantic type.

There are at least two reasons why the ontology might be useful for identifying traceability links. First, since only those verbs and nouns that (1) appear in the training data and (2) are deemed relevant by the domain expert for link identification are included in the ontology, it provides guidance to the learner as to which words/phrases in the requirements it should focus on in the learning process. Second, the verb and noun clusters provide a robust generalization of the words/phrases in the requirements. For instance, a word pair that is relevant for link identification may still be ignored by the learner due to its infrequency of occurrence. The features that are computed based on these clusters, on the other hand, will be more robust to the infrequency problem and therefore potentially provide better generalizations.

Our main contribution in this paper lies in the proposal of a 2-learner, ontology-based approach to the task of traceability link prediction, where, for the sake of robustness, we train two classifiers to separately exploit the word-pair features and the ontology-based features. Results on a traceability dataset involving the Pine domain reveal that our use of two learners and the ontology-based features are both key to the success of our approach: it significantly outperforms not only a supervised baseline system that uses only word pairs features, but also a system that trains a single classifier over both the word pairs and the ontology-based features. Perhaps most interestingly, results do not deteriorate when the hand-built ontology is replaced with an automatically constructed ontology.

2 RELATED WORK

Automated or semi-automated requirements traceability has been exploited by many researchers. Pierce [11] designed a tool that maintains a requirements database to aid automated requirements tracing. Jackson [8] proposed a keyphrase-based approach for tracing a large number of requirements of a large Surface Ship Command System. More advanced approaches relying on information retrieval (IR) techniques, such as the tf-idf-based vector space model [13], Latent Semantic Indexing [5, 6, 10], probabilistic networks [3], and Latent Dirichlet Allocation [12], have been investigated, where traceability links were generated by calculating the textual similarity between requirements using similarity measures such as Dice, Jaccard, and Cosine coefficients [4]. All these methods were developed based on either matching keywords or identifying similar words across a pair of requirements. In recent years, Li [9] has studied the feasibility of employing supervised learning to accomplish this task.

3 DATASET

We employ the well known Pine system for evaluation. This dataset consists of a set of 49 (high-level) requirements and a set of 51 (low-level) use case specifications about Pine, an email system developed at the University of Washington. Out of the 2499 pairs of requirement and use case specification, only 10% (250) are considered traceability links.
4 APPROACH

In this section, we describe our supervised approach.

4.1 Classifier Training

Each instance corresponds to a high-level requirement and a low-level requirement. Hence, we create instances by pairing each high-level requirement with each low-level requirement. The class value of an instance is positive if the two requirements involved should be linked; otherwise, it is negative. Since we conduct 5-fold cross-validation experiments, we randomly partition the instances into five folds of roughly the same size, training only four folds and evaluate on the remaining fold in each fold experiment. Each instance is represented using seven types of features, as follows.

1. Same words. We create one binary feature for each word \( w \) appearing in the training data. Its value is 1 if \( w \) appears in both requirements in the pair under consideration. Hence, this feature type contains the subset of the word pair features mentioned earlier where the two words in the pair are the same.

2. Different words. We create one binary feature for each word pair \((w_i, w_j)\) collected from the training instances, where \(w_i\) and \(w_j\) are non-identical words appearing in a high-level requirement and a low-level requirement respectively. Its value is 1 if \(w_i\) and \(w_j\) appear in the high-level and low-level pair under consideration, respectively. Hence, this feature type contains the subset of the word pair features where the two words in the pair are different.

3. Verb pairs. We create one binary feature for each verb pair \((v_i, v_j)\) collected from the training instances, where \(v_i\) and \(v_j\) appear in a high-level requirement and a low-level requirement respectively, and both verbs appear in the ontology. Its value is 1 if \(v_i\) and \(v_j\) appear in the high-level and low-level pair under consideration, respectively. Using these verb pairs as features may allow the learner to focus on verbs that are relevant to traceability prediction.

4. Verb group pairs. For each verb pair feature described above, we create one binary feature by replacing each verb in the pair with its cluster id in the ontology. Its value is 1 if the two verb groups in the pair appear in the high-level and low-level pair under consideration, respectively. These features may enable the resulting classifier to provide robust generalizations in cases where the learner chooses to ignore certain useful verb pairs owing to their infrequency of occurrence.

5. Noun pairs. We create one binary feature for each noun pair \((n_i, n_j)\) collected from the training instances, where \(n_i\) and \(n_j\) appear in a high-level requirement and a low-level requirement respectively, and both nouns appear in the ontology. Its value is computed in the same manner as the verb pairs. These noun pairs may help the learner to focus on verbs that are relevant to traceability prediction.

6. Noun group pairs. For each noun pair feature described above, we create one binary feature by replacing each noun in the pair with its cluster id in the ontology. Its value is computed in the same manner as the verb group pairs. These features may enable the classifier to provide robust generalizations in cases where the learner chooses to ignore certain useful noun pairs owing to their infrequency of occurrence.

7. Dependency pairs. In some cases, the noun/verb pairs may not provide sufficient information for traceability prediction. For example, the verb pair feature \(\text{(delete, delete)}\) is suggestive of a positive instance, but the instance may turn out to be negative if one requirement concerns deleting messages and the other concerns deleting folders. As another example, the noun pair feature \(\text{(folder, folder)}\) is suggestive of a positive instance, but the instance may turn out to be negative if one requirement concerns creating folders and the other concerns deleting folders.

In other words, we need features that encode the verbs and nouns in isolation but the relationship between them. To do so, we first parse each requirement using the Stanford dependency parser [7], and collect each noun-verb pair \((n_i, v_j)\) connected by a dependency relation. We then create binary features by pairing each related noun-verb pair found in a high-level training requirement with each related noun-verb pair found in a low-level training requirement. The feature value is 1 if the two noun-verb pairs appear in the pair of requirements under consideration. To enable the learner to focus on learning from relevant verbs and nouns, only verbs and nouns that appear in the ontology are used to create these features.

We employ LIBSVM [2] as the learning algorithm for training a binary SVM classifier on the training set. We use a linear kernel, tuning the C value (the regularization parameter) to maximize F-score on the development (dev) set. All other learning parameters are set to their default values.

To improve performance, we perform feature selection (FS) using the backward elimination algorithm [1]. Starting with all seven feature types, the algorithm iteratively removes one feature type at a time until only one feature type is left. Specifically, in each iteration, it removes the feature type whose removal yields the largest F-score on the dev set. We picked the feature subset that achieving the largest F-score on the dev set over all iterations.

Note that tuning the C value (from libSVM) and selecting the feature subset both require the use of a dev set. In each fold experiment, we reserve one fold for development and use the remaining three folds for training classifiers. We jointly tune the C value and select the feature subset to maximize F-score on the dev set.

4.2 Two Extensions

Next, we present two extensions to our supervised approach.

4.2.1 Employing Two Views. Our first extension involves splitting our feature sets into two views (i.e., disjoint subsets) and training one classifier on each view. To motivate this extension, recall that the ontology is composed of words and phrases that are deemed relevant to traceability prediction according to a SE expert. In other words, the (word- and cluster-based) features derived from the ontology (i.e., features 3–7 in our feature set) are sufficient for traceability prediction, and the remaining features (features 1 and 2) are not needed according to the expert. While some of the word pairs that appear in features 1 and 2 also appear in features 3–7, most of them do not. If these expert-determined irrelevant features are indeed irrelevant, then retaining them could be harmful for classification because they significantly outnumber their relevant counterparts. However, if some of these features are relevant (because some relevant words are missed by the expert, for instance), then removing them would not be a good idea either.

Our solution to this dilemma is to divide the feature set into two views. Given the above discussion, a natural feature split would involve putting the ontology-based features (features 3–7) into one view and the remaining ones (features 1–2) into the other view.
Then we train one SVM classifier on each view as before. During test time, we apply both classifiers to a test instance, classifying it using the prediction associated with the higher confidence value.\textsuperscript{3} This setup would prevent the expert-determined irrelevant features from affecting the relevant ones, and at the same time avoid totally discarding them in case they do contain some relevant information.

A natural question is: why not simply use backward elimination to identify the irrelevant features? While we believe FS can help, it may not be as powerful as one would think because (1) backward elimination is greedy; and (2) the features are selected using a fairly small set of instances (i.e., the dev set) and may therefore be biased towards the dev set.

In fact, we view our 2-learner setup and FS as complementary rather than competitive solutions to our dilemma. In particular, we will employ FS in the 2-learner setup: when training the classifiers on the two views, we employ backward elimination in the same way as before, removing the feature type (from one of the two classifiers) whose removal yields the highest F-score on the dev set in each iteration.

4.2.2 Learning the Ontology. An interesting question is: can we learn instead of hand-build the ontology? Not only is this question interesting from a research perspective, it is of practical relevance: even if a domain expert is available, hand-constructing the ontology is a time-consuming and error-prone process. Below we describe the steps we propose for ontology learning, which involves producing a verb clustering and a noun clustering.

Step 1: Verb/Noun selection. We select the nouns, noun phrases (NPs) and verbs in the training set to be clustered. Specifically, we select a verb/noun/NP if (1) it appears more once in the training data; (2) it contains at least three characters (thus avoiding verbs such as be); and (3) it appears in the high-level but not the low-level requirements and vice versa.

Step 2: Verb/Noun representation. We represent each noun/NP/verb as a feature vector. Each verb \( v \) is represented using the set of nouns/NPs collected in Step 1. The value of each feature is binary: 1 if the corresponding noun/NP occurs as the direct or indirect object of \( v \) in the training data (as determined by the Stanford dependency parser), and 0 otherwise. Similarly, each noun \( n \) is represented using the set of verbs collected in Step 1. The value of each feature is binary: 1 if \( n \) serves as the direct or indirect object of the corresponding verb in the training data, and 0 otherwise.

Step 3: Clustering. To produce a verb clustering and a noun clustering, we cluster the verbs and the nouns/NPs separately. We experiment with two clustering algorithms. The first one, which we refer to as Simple, is the classical single-link algorithm. Single-link is an agglomerative algorithm where each object to be clustered is initially in its own cluster. In each iteration, it merges the two most similar clusters and stops when the desired number of clusters is reached. The second clustering algorithm is motivated by the following observation. We could produce a better verb clustering if each verb were represented using noun categories rather than nouns/NPs, because there is no need to distinguish between the nouns in the same category in order to produce the verb clusters we desire. Similarly, we could produce a better noun clustering if each noun were represented using verb categories.

In practice, we do not have the noun and verb categories (because they are what the clustering algorithm is trying to produce). However, we can use the (partial) verb clusters produced during the verb clustering process to improve noun clustering and vice versa. This motivates our Interactive clustering algorithm. Like Simple, Interactive is also a single-link clustering algorithm. Unlike Simple, which produces the two clusterings separately, Interactive interleaves the verb and noun clustering processes, as described below.

Initially, each verb and each noun is in its own cluster. In each iteration, we (1) merge the two most similar verb clusters; (2) update the noun’s feature representation by merging the two verb features that correspond to the newly formed verb cluster\textsuperscript{4}; (3) merge the two most similar noun clusters using this updated feature representation for nouns; (4) update the verb’s feature representation by merging the two noun features that correspond to the newly formed noun cluster. As in Simple, Interactive terminates when the desired number of clusters is reached.

For both clustering algorithms, we compute the similarity between two objects by taking the dot product of their feature vectors. Since we are using single-link clustering, the similarity between two clusters is the similarity between the two most similar objects in the two clusters. Since we do not know the number of clusters to be produced a priori, we produced three noun clusterings and three verb clusterings (with 10, 15, and 20 clusters each). We then select the combination of noun clustering, verb clustering, the C value, and the feature subset that maximizes F-score on the dev set, and apply the resulting combination on the test set.

5 EVALUATION

5.1 Experimental Setup

We employ as our evaluation measure F-score, which is the unweighted harmonic mean of recall and precision. Recall is the percentage of links in the gold standard that are recovered by our system. Precision is the percentage of links recovered by our system that are correct. We preprocess each document by removing stopwords and stemming the remaining words. All results are obtained via 5-fold cross validation.

5.2 Results and Discussion

5.2.1 Baseline Systems. We present two unsupervised and two supervised baselines.

Baseline 1: \( tf.idf \). Motivated by previous work, we employ \( tf.idf \) as our first unsupervised baseline. Each document is represented as a vector of unigrams. The value of each feature is its \( tf.idf \) value. Cosine is used to compute the similarity between two documents. Any pair of requirements whose similarity exceeds a given threshold is labeled as positive. We tested thresholds from 0.1 to 0.9 with an increment of 0.1 and report results using the best threshold, essentially giving an advantage to it in the performance comparison. As we can see in row 1 of Table 1, it achieves an F-score of 54.5%.

Baseline 2: LDA. Also motivated by previous work, we employ LDA as our second unsupervised baseline. We train an LDA on our data to produce n topics (where \( n=10, 20, \ldots, 60 \)). We then use

\textsuperscript{3}To compute the confidence value associated with a prediction, we take the absolute distance of the underlying test instance from the hyperplane.

\textsuperscript{4}This will reduce the number of features by one. The value of the "merged" feature will be the disjunction of the values of the original features.
Table 1: Five-fold cross-validation results.

<table>
<thead>
<tr>
<th>System</th>
<th>Feature Selection?</th>
<th>Recall</th>
<th>Prec.</th>
<th>F-score</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Baseline Systems</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TF.idf</td>
<td>N/A</td>
<td>73.6</td>
<td>43.3</td>
<td>54.5</td>
</tr>
<tr>
<td>LDA</td>
<td>N/A</td>
<td>30.4</td>
<td>39.2</td>
<td>34.2</td>
</tr>
<tr>
<td>Features 1&amp;2</td>
<td>No</td>
<td>50.0</td>
<td>66.5</td>
<td>75.5</td>
</tr>
<tr>
<td></td>
<td>Yes</td>
<td>62.4</td>
<td>73.9</td>
<td>71.0</td>
</tr>
<tr>
<td>Features 1&amp;2 + LDA</td>
<td>No</td>
<td>50.4</td>
<td>67.0</td>
<td>65.5</td>
</tr>
<tr>
<td></td>
<td>Yes</td>
<td>66.8</td>
<td>72.4</td>
<td>75.5</td>
</tr>
<tr>
<td><strong>Our Approach</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Single learner + manual clusters</td>
<td>No</td>
<td>54.0</td>
<td>73.0</td>
<td>62.1</td>
</tr>
<tr>
<td></td>
<td>Yes</td>
<td>66.8</td>
<td>79.1</td>
<td>72.5</td>
</tr>
<tr>
<td>Single learner + induced clusters</td>
<td>No</td>
<td>53.2</td>
<td>73.5</td>
<td>61.7</td>
</tr>
<tr>
<td></td>
<td>Yes</td>
<td>65.6</td>
<td>78.1</td>
<td>71.3</td>
</tr>
<tr>
<td>Two learners + manual clusters</td>
<td>No</td>
<td>61.6</td>
<td>84.6</td>
<td>71.3</td>
</tr>
<tr>
<td></td>
<td>Yes</td>
<td>68.4</td>
<td>84.2</td>
<td>75.5</td>
</tr>
<tr>
<td>Two learners + induced clusters</td>
<td>No</td>
<td>62.8</td>
<td>81.8</td>
<td>71.0</td>
</tr>
<tr>
<td></td>
<td>Yes</td>
<td>71.2</td>
<td>84.0</td>
<td>77.1</td>
</tr>
</tbody>
</table>

the n topics as features for representing each document, where the value of a feature is the probability the document belongs to the corresponding topic. Cosine is used as the similarity measure. Any pair of requirements whose similarity exceeds a given threshold is labeled as positive. We tested thresholds from 0.1 to 0.9 with an increment of 0.1 and report results using the best threshold, essentially giving an advantage to it in the performance comparison. As we can see in row 2 of Table 1, it achieves an F-score of 34.2%.

**Baseline 3: Features 1 and 2.** As our first supervised baseline, we train a SVM classifier using only features 1 and 2 (all the word pairs). As we can see from row 3 of Table 1, it achieves F-scores of 57.1% (without FS) and 67.7% (with FS). These results suggest that FS is indeed useful.

**Baseline 4: Features 1, 2, and LDA.** As our second supervised baseline, we augment the feature set used in Baseline 3 with the LDA features used in Baseline 2 and then train a SVM classifier. We select the best n (number of topics) using the dev set. As we can see in row 4 of Table 1, this is the best of the four baselines: it significantly outperforms Baseline 3 regardless of whether feature selection is performed\(^1\), suggesting the usefulness of the LDA features.

5.2.2 **Our Approach.** Next, we evaluate our 2-learner, ontology-based approach. In the single-learner experiments, a classifier is trained on the seven features described in Section 4.1, whereas in the 2-learner experiments, these seven features are split as described in Section 4.2.

**Setting 1: Single learner, manual clusters.** As we can see from row 5 of Table 1, this classifier significantly outperforms the best baseline (Baseline 4): F-scores increase by 4.6% (without FS) and 3.5% (with FS). Since the only difference between this and Baseline 4 lies in whether the LDA features or the ontology-based features are used, these results seem to suggest that features formed from the clusters in our hand-built ontology are more useful than the LDA features.

**Setting 2: Single learner, induced clusters.** As we can see from row 6 of Table 1, this classifier performs statistically indistinguishably from the one in Setting 1. This is an encouraging result: it shows that even when features are created from induced rather than manual clusters, performance does not significantly drop regardless of whether FS is performed.

---

\(^1\)All significance tests are two-tailed paired t-tests (\(p < 0.05\)).

Setting 3: Two learners, manual clusters. As we can see from row 7 of Table 1, this classifier performs significantly better than the one in Setting 1: F-scores increase by 9.2% (without FS) and 3.0% (with FS). As the two settings differ only w.r.t. whether one or two learners are used, the improvements suggest the effectiveness of our 2-learner framework.

Setting 4: Two learners, induced clusters. As we can see from row 8 of Table 1, this classifier performs significantly better than the one in Setting 2: F-scores increase by 9.3% (without FS) and 5.8% (with FS). It also performs indistinguishably from the one in Setting 3. Taken together, these results suggest that (1) our 2-learner framework is effective in improving performance, and (2) features derived from induced clusters are as effective as those from manual clusters.

Overall, these results show that (1) our 2-learner, ontology-based approach is effective, and (2) feature selection consistently improves performance.

6 CONCLUSIONS

We investigated a 2-learner, ontology-based approach to supervised traceability prediction. Results showed that (1) our approach is effective: in comparison to the best baseline, relative error reduces by 25.9%; and (2) results obtained via induced clusters were as competitive as those obtained via manual clusters.

REFERENCES

Hybrid Software and System Development in Practice: Waterfall, Scrum, and Beyond

Marco Kuhrmann
Clausthal University of Technology
kuhrmann@acm.org

Philipp Diebold
Fraunhofer ISE
philipp.diebold@iese.fhg.de

Jürgen Münch
Reutlingen University
j.muench@computer.org

Paolo Tell
IT University Copenhagen
pate@itu.dk

Vahid Garousi
University of Luxembourg
garousi@svv.lu

Michael Felderer
University of Innsbruck
Michael.Felderer@uibk.ac.at

Kitija Trektere
Dundalk Institute of Technology
kitija.trektere@dkit.ie

Fergal McCaffery
Dundalk Institute of Technology
fergal.mccaffery@dkit.ie

Oliver Linssen
FOM University of Applied Sciences
oliver.linssen@fom.de

Eckhart Hanser
DHBW Lörrach
hanser@dhw-loerrach.de

Christian R. Prause
German Aerospace Center
christian.prause@dlr.de

ABSTRACT
Software and system development faces numerous challenges of rapidly changing markets. To address such challenges, companies and projects design and adopt specific development approaches by combining well-structured comprehensive methods and flexible agile practices. Yet, the number of methods and practices is large, and available studies argue that the actual process composition is carried out in a fairly ad-hoc manner. The present paper reports on a survey on hybrid software development approaches. We study which approaches are used in practice, how different approaches are combined, and what contextual factors influence the use and combination of hybrid software development approaches. Our results from 69 study participants show a variety of development approaches used and combined in practice. We show that most combinations follow a pattern in which a traditional process model serves as framework in which several fine-grained (agile) practices are plugged in. We further show that hybrid software development approaches are independent from the company size and external triggers. We conclude that such approaches are the results of a natural process evolution, which is mainly driven by experience, learning, and pragmatism.

CCS CONCEPTS
• General and reference → Surveys and overviews; • Software and its engineering → Software development methods; Software organization and properties; Designing software; Software development techniques; Programming teams;

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

ICSSP'17, Paris, France
© 2017 ACM. 978-1-4503-5270-3/17/07... $15.00
DOI: 10.1145/3084100.3084104

1 INTRODUCTION
Software development is diverse, and companies have to adopt to new technologies and markets quickly. According to Brooks [2], there is no “Silver Bullet” in software development. Hence, software engineers are on the quest for suitable development approaches, yet facing a huge variety of contextual factors influencing the definition of appropriate development processes [4, 14, 26]. To address these factors as well as the increasing number of domains for which software has become a vital part, different software and system development methods have been proposed. These methods implement different philosophies, e.g., stage-gate or plan-based, iterative-incremental, and lean, and range in their lifecycle coverage from small task-specific practices (e.g., daily stand-up) to large and comprehensive process frameworks (e.g., the family of V-shaped processes). Furthermore, software development has become key to system development and, therefore, increasingly addresses safety-critical and reliable systems, such as Automotive software, Aerospace systems, or Medical Devices. These domains add standards, norms, and regulations to software and its development.

Hybrid Software Development Approaches. We aim to study hybrid software development approaches (short: hybrid approaches), which we define as follows: A hybrid software development approach...
is any combination of agile and traditional (plan-driven or rich) approaches that an organizational unit adopts and customizes to its own context needs (e.g., application domain, culture, processes, project, organizational structure, techniques, technologies, etc.).

Problem Statement. In 2011, West et al. [24] coined the term “Water-Scrum-Fall” and hypothesized that hybrid development methods will become the standard. A systematic review by Theocharis et al. [20] aimed at collecting evidence to confirm West’s claim. They revealed a gap in literature: while research on agile software development is rich, traditional processes are widely ignored in recent research. Hence, data (and evidence) on combination patterns and contextual factors driving the creation of hybrid approaches is missing, e.g., how do standards affect the use of agile methods, or do company size and industry sector matter?

Objective. The goal of our research is to close this gap and to collect data to help determining combination patterns, i.e., which development approaches are used in practice and how are these approaches combined in company- or project-specific development approaches. We further aim to identify context factors influencing the creation of hybrid approaches. We also aim to investigate which software development problems motivate hybrid approaches and if hybrid approaches help solving such problems.

Contribution. The paper at hand presents results from the HELENA study. Differently from the study by Theocharis et al. [20], HELENA is an internationally conducted survey that aims at collecting data to study the use of hybrid approaches. Based on the analysis of 69 responses, we present a list of development approaches as used in practice. We analyze these development approaches for patterns, and test our data for different context attributes. Based on cluster analyses, we identified five major combination patterns. Furthermore, we present the data confirming that using hybrid approaches has become mainstream, and that their use is independent from company size, industry sector, and external standards.

Outline. The remainder of the paper is organized as follows: Section 2 provides an overview of related work. Section 3 describes the research design, and Section 4 presents the results. Finally, we conclude the paper in Section 5.

2 RELATED WORK

There is a long history of studies on the use of software processes. An important step was a special issue of IEEE Software in 2003 in which several authors started collecting knowledge about process use in general and combining processes in particular. For instance, Cusumano et al. [5] surveyed 104 projects and found many projects using and combining different development approaches. Jones’ findings [13], which are based on an analysis of data from approx. 12,000 projects, indicate a certain diversity in the development methods used. Despite the variety observed, a pattern seems to be recurrent. Neill and Laplante [18] found in their study that approx. 35% of the participants use the Waterfall model, yet, projects also use evolutionary/incremental approaches—even within particular lifecycle phases. Starting in 2005, a series of independently conducted studies in Germany investigated, among other things, the use of software processes from different perspectives [3, 11, 15] and showed numerous development approaches applied and combined with each other. Recently, Garousi et al. [12] and Vijayasarathy and Butler [23] showed that “classic” approaches like the Waterfall model are increasingly combined with agile/lean development approaches. These studies confirm the observation that a huge bandwidth of processes exist and that they are combined with each other.

In 2011, West et al. [24] coined the term “Water-Scrum-Fall” to describe that very combination pattern, which was studied by Theocharis et al. [20]. Their major finding is that few data is available about combined process use in general. Moreover, Theocharis et al. [20] found an extensive knowledge base on agile software development, e.g., Dybå and Dingsøyr [9] or the continuously updated State of Agile Survey [22]. However, missing so far is a big picture due to the lack of information about traditional processes.

This missing big picture motivates the research presented in the paper at hand. The goal is to study the use of (hybrid) software development processes in general with respect to the development context (e.g., industry sector, company size) and the different constraints companies face (e.g., standards, norms, and regulations). This paper thus fills a gap in literature by (i) contributing to the body of knowledge on process use, but (ii) providing a more holistic perspective covering traditional and agile/lean approaches alike.

3 RESEARCH DESIGN

Section 3.1 presents research questions and hypotheses, followed by a description of the instrument in Section 3.2. Data collection and analysis procedures are presented in Section 3.3 and 3.4. Section 3.5 describes the procedures to increase the validity of our results.

3.1 Research Questions and Hypotheses

Our research is driven by accepting West’s “Water-Scrum-Fall” hypothesis, and we accept that hybrid approaches have become reality and shape today’s software system development [20]. Therefore, this research aims at studying what a hybrid approach is after all, how and why those approaches are developed, and whether hybrid approaches fulfill the expectations of practitioners. For this, we define the fine-grained research questions in Table 1. In addition, we aim to study the context factors company size and external triggers
Table 2: The HELENA questionnaire lists the questions and question groups (conditional questions for the different paths are omitted in this table). The table lists the question scales, and if applicable the number of options and free-text options.

<table>
<thead>
<tr>
<th>No.</th>
<th>Group</th>
<th>Question</th>
<th>Scale</th>
<th>#opt</th>
<th>FT</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>M</td>
<td>What is your organization’s size?</td>
<td>SC</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>2.</td>
<td>M</td>
<td>What is the main business area of your company?</td>
<td>MC</td>
<td>7</td>
<td>✓</td>
</tr>
<tr>
<td>3.</td>
<td>M</td>
<td>Do you participate in distributed software projects?</td>
<td>SC</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>4.</td>
<td>M</td>
<td>In which country are you personally located?</td>
<td>FT</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5.</td>
<td>M</td>
<td>In which application domain are you most frequently involved?</td>
<td>MC</td>
<td>17</td>
<td>✓</td>
</tr>
<tr>
<td>6.</td>
<td>M</td>
<td>Which role are you most frequently assigned to?</td>
<td>SC</td>
<td>9</td>
<td>✓</td>
</tr>
<tr>
<td>7.</td>
<td>M</td>
<td>In your projects, a software failure potentially can: (option list)</td>
<td>MC</td>
<td>8</td>
<td></td>
</tr>
<tr>
<td>8.</td>
<td>PU</td>
<td>Does your company define a company-wide standard process for software and system development?</td>
<td>SC</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>9.</td>
<td>PU</td>
<td>Which of the following development approaches and practices do you use?</td>
<td>MC</td>
<td>40</td>
<td>✓</td>
</tr>
<tr>
<td>10.</td>
<td>PU</td>
<td>Do you combine different development approaches?</td>
<td>YN</td>
<td></td>
<td></td>
</tr>
<tr>
<td>11.</td>
<td>PU</td>
<td>For the following standard activities in your projects, please indicate to which degree you carry out these activities in a more traditional or more agile manner. (option list comprises 11 categories)</td>
<td>LI</td>
<td></td>
<td></td>
</tr>
<tr>
<td>12.</td>
<td>PU</td>
<td>What is the main motivation for this particular combination of development approaches?</td>
<td>FT</td>
<td></td>
<td></td>
</tr>
<tr>
<td>13.</td>
<td>PU</td>
<td>How were the combinations of development approaches in your company developed?</td>
<td>MC</td>
<td>3</td>
<td>✓</td>
</tr>
<tr>
<td>14.</td>
<td>PU</td>
<td>How do you select your project-specific development approach?</td>
<td>MC</td>
<td>6</td>
<td></td>
</tr>
<tr>
<td>15.</td>
<td>PUS</td>
<td>Which external standards are implemented in your company?</td>
<td>FT</td>
<td></td>
<td></td>
</tr>
<tr>
<td>16.</td>
<td>PUS</td>
<td>Why have you implemented the aforementioned standards?</td>
<td>MC</td>
<td>3</td>
<td>✓</td>
</tr>
<tr>
<td>17.</td>
<td>PUS</td>
<td>How is the compliance of the development process assessed?</td>
<td>MC</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>18.</td>
<td>PUS</td>
<td>Does agility challenge the implementation of the standards you have to apply?</td>
<td>YN</td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>19.</td>
<td>PUL</td>
<td>Is your development approach continuously improved?</td>
<td>SC</td>
<td>5</td>
<td>✓</td>
</tr>
<tr>
<td>20.</td>
<td>PUL</td>
<td>What is your motivation for implementing an improvement program?</td>
<td>MC</td>
<td>5</td>
<td>✓</td>
</tr>
<tr>
<td>21.</td>
<td>PUL</td>
<td>Is your company, unit or project certified?</td>
<td>YN</td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>22.</td>
<td>PUL</td>
<td>What are the goals of your improvement program? (option list comprises 12 categories)</td>
<td>LI</td>
<td></td>
<td></td>
</tr>
<tr>
<td>23.</td>
<td>E</td>
<td>Based on your experience, please rate the following statements: (option list comprises 6 categories)</td>
<td>LI</td>
<td></td>
<td></td>
</tr>
<tr>
<td>24.</td>
<td>E</td>
<td>Based on your experience, can you name problems occurred regarding your current process and your current application domain?</td>
<td>FT</td>
<td></td>
<td></td>
</tr>
<tr>
<td>25.</td>
<td>E</td>
<td>Do you have any further comments or issues not addressed so far?</td>
<td>FT</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Legend for scales: YN=yes/no, SC=single choice, MC=multiple choice/select, LI=5-item Likert scale, FT=free text.

Legend for groups: M=metadata, PU=process use, PUS=process use and standards, PUL=process use in the lifecycle, E=experience.

Table 3: Hypotheses and variables (ref. to questions (Q) from Table 2 and hypothesis (H) in which the variable is used).

### Hypotheses

H1: The use of hybrid approaches depends on the company size.

H2: The use of external standards depends on the company size.

H3: The use of hybrid approaches depends on external standards.

H4: The use of hybrid approaches depends on external triggers.

### Variable Q H

<table>
<thead>
<tr>
<th>cs ∈ {1 . . . 5} (company size)</th>
<th>1</th>
<th>H1, H2</th>
</tr>
</thead>
<tbody>
<tr>
<td>hbruc ∈ {0, 1} (hybrid approach use)</td>
<td>10</td>
<td>H1, H3, H4</td>
</tr>
<tr>
<td>stdext ∈ {1 . . . 3} (ext. standard use)</td>
<td>8</td>
<td>H2, H3</td>
</tr>
<tr>
<td>External trigger:</td>
<td>16</td>
<td>H4</td>
</tr>
</tbody>
</table>
| \(\begin{cases} 
2 & \text{if } \text{trig}_{\text{ext}} = \text{true} \\
1 & \text{if } \text{trig}_{\text{prj}} = \text{true} \land \text{trig}_{\text{pel}} = \text{true} \\
0 & \text{otherwise}
\end{cases} \) |

for HELENA is mainly grounded in [15], but also borrows from the annual State of Agile Survey [22], which we used to enrich the collection of development approaches (the 3ProcSurvey only contained a limited list). The study as such targets practitioners and aims at (i) collecting quantitative data regarding the (general) process use and (ii) collecting qualitative data on experiences. The context is further set by constraints originating from external standards, norms, and regulations to be applied to organizations as a whole as well as to projects. Hence, the questionnaire covers organization-, project- and personal experience levels.

The questionnaire was developed incrementally to increase the validity of the instrument. In the first stage, the questionnaire from the 3ProcSurvey was used and analyzed for reusable assets. In the second step, the questionnaire was initially crafted and tested in two external organizations\(^2\) with in total 15 subjects [16].

After the trial phase, the questionnaire was finalized and published. The final questionnaire consists of max. 25 questions (excluding conditional selectors; Table 2) of which: seven questions aim at collecting metadata; seven questions aim at collecting information about general process use; four questions to study process use in the context of standards, norms, and regulations; four questions to investigate the use of processes and standards in the process lifecycle; and, three questions to gather general experiences. The questionnaire was designed to be manageable within 10–20 minutes, depending on the actual path through the questionnaire.

\(^2\)The German Aerospace Center (DLR) and FOM University of Applied Sciences for Economics and Management. Researchers from both institutions were involved in the questionnaire’s development and quality assurance before running the tests locally at their institutions.
3.3 Data Collection Procedure

The first stage of the HELENA survey was accepting answers from May to June 2016. A simple questionnaire design based on Google Forms was used. We opted for a convenience sampling strategy [25], and posted the survey to a number of mailing lists of IT clusters and networks, and we used LinkedIn, Twitter, Facebook, Xing, and ResearchGate to promote the survey within the relevant communities. Since one of our major goals in this first and exploratory stage was gaining broad visibility, we intentionally sacrificed the ability to calculate response rates (Section 3.5).

3.4 Data Analysis Procedure

To analyze the data, we utilized several methods to provide answers to the research questions. For all research questions, we use descriptive statistics, e.g., to provide tables and charts for process use and process selection. Furthermore, we applied cluster analysis and hypothesis testing to analyze our data set. To answer RQ2, we applied a multi-staged cluster analysis: As a first step, we applied an Affinity Propagation Clustering (AP; [10]) to search for general structures in the result set. In the second step, we applied a Spectral Clustering (SC; [19]) to split the result set in two subsets: $A_f^{high}$ and $A_f^{low}$ to sort those development approaches with high affinity and those with little/no affinity. $A_f^{low}$ was excluded from further analyses. In the third step, we applied AP($A_f^{high}$) to determine the number of centers $c$ in $A_f^{high}$ for further analysis, and applied SC($A_f^{high}$, $c$) to determine the final clusters. In addition, we applied SC($A_f^{high}$, 2) to investigate, whether the two ‘opposite worlds’, i.e., traditional and agile, can be constructed from the result set. Finally, to answer RQ3 and to test the hypotheses H1-4, we applied the Pearson’s $\chi^2$ test, with $p \leq 0.05$ to find (no) support for the respective hypotheses.

3.5 Validity Procedures

To improve the validity and to mitigate risks, we implemented different measures: First, our research is grounded in previously conducted studies, in particular [11, 15] that provided a basic set of questions. To find a set of development approaches of interest, we also ground our research in lists of approaches (e.g., offered by the State of Agile Survey [22]), which have been combined to complement the findings of our previously conducted research. A design group consisting of three researchers developed the initial version of the instrument (Sect. 3.2). Two more researchers performed a quality assurance and conducted an external test of the questionnaire to test the general feasibility [16]. Finally, the data analysis was handed over to another team of two researchers.

Another risk is related to the data collection strategy: Since one of the main goals of the study is to build a quantitative basis for our research, we accepted the risk of loosing full control in terms of sampling, response rate, and so forth, and—for HELENA’s first stage—we intentionally opted for an open call for participation (convenience sampling; [25]) to maximize the number of datapoints.

4 RESULTS

We present the results from the survey. We start with an overview of the study population in Section 4.1, before we present the findings according to the research questions. Finally, we discuss our findings in Section 4.5.

<table>
<thead>
<tr>
<th>Company Size</th>
<th>Micro</th>
<th>Small</th>
<th>Medium</th>
<th>Large</th>
<th>Very Large</th>
</tr>
</thead>
<tbody>
<tr>
<td>%</td>
<td>17.4</td>
<td>20.3</td>
<td>17.4</td>
<td>20.3</td>
<td>24.6</td>
</tr>
</tbody>
</table>

Table 4: Overview of the company size and the roles participants have (n=69).

4.1 Study Population

In total, we received 69 complete responses from approx. 15 countries. Responses regarding the company size cover all categories, i.e., the result set contains answers ranging from micro-sized companies to very large companies (Table 4). Furthermore, approx. 2/3 of the respondents state that they work in a distributed fashion, in particular: 18.8% work distributed in the same country and 20.3% in the same region, and 26.1% work globally distributed. As found in previous studies [15], distributed work has become reality for companies of all sizes. Participants were asked for the roles that they are most frequently assigned to. Table 4 shows the outcomes and relates the roles to the company size. Project/team manager is the most frequently stated role, followed by architects, testers, product managers/owners, and developers.

Another 8.7% was categorized as “other” (e.g., safety managers, compliance managers, and C-level managers).

Figure 1 provides an overview of the industry sectors in which the participants are active. In total, the survey returned 167 selections, i.e., several participants are engaged in multiple sectors. The figure shows approx. 29% of the participants are engaged in Web Applications and Services, followed by Medical Devices and Health Care, and Public Sector/Public Contracting. Among the sectors categorized as “Other”, participants named Energy, Traffic Management Systems, or Industrial Control Systems.

4.2 RQ1: Development Approach Use

The first research question aims to study the use of different development approaches in general. As described in Sect. 3.2, the participants were presented a list of 40 development approaches to select—including a free text field. Figure 2 shows the participants’ selection. In total, 729 selections have been made by the 69 participants. The figure shows (i) a ranking of the different approaches based on the frequency and that (ii) the participants use the different approaches in combination.

---

3 A codebook and raw data are available from: https://goo.gl/MK0mYZ. Note that the raw data is available in an anonymized from only.
To further analyze the result set, we defined the two main categories \textit{method} and \textit{practice}, and within each category, we used the three sub-categories \textit{traditional}, \textit{agile}, and \textit{both}. The categorization is based on the definitions provided by Diebold and Zehler [8]. Figure 2 shows the resulting classification of the different approaches (main categories marked along side the label, sub-categories color coded), and shows that—starting with 20% share—the participants use development approaches of all kinds, i.e., traditional like the Waterfall model, agile like Scrum, and generic (both) approaches, such as code reviews. Among others, Figure 2 shows that more than a half of the participants (53.6%) implement Scrum and more than a third (34.8%) implement a Waterfall/Phase Model. That is, West’s claim that the “Water-Scrum-Fall” will become reality for software system development [24] is confirmed by our result set. This large-scale combination is complemented by a number of small practices, which also supports our claim that practices have become the building blocks of process customization [20].

4.3 RQ2: Combination of Development Approaches

The second research question aims at studying the way of implementing combinations of different development approaches. For this, we categorized the different development approaches (Section 4.2) and, based on this categorization, analyzed the dataset from different perspectives.

4.3.1 Process Use in Projects (Self-Evaluation and Company Size). To analyze how the different approaches are combined and used in practice, we analyzed the result set from two perspectives: the first perspective is given by a self-evaluation of the participants on how they implement a set of given project disciplines. In the second perspective, we analyze the process use in the context of the different company sizes.
Table 5: Use of development approaches (in % incl. mean, mean absolute deviation; categorized, company size; normalized by selected approaches n).

<table>
<thead>
<tr>
<th>Comp. Size</th>
<th>Method</th>
<th>Practice</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>T</td>
<td>A</td>
</tr>
<tr>
<td>Micro</td>
<td>6.6</td>
<td>15.4</td>
</tr>
<tr>
<td>Small</td>
<td>3.2</td>
<td>16.7</td>
</tr>
<tr>
<td>Medium</td>
<td>4.9</td>
<td>16.2</td>
</tr>
<tr>
<td>Large</td>
<td>4.3</td>
<td>16.0</td>
</tr>
<tr>
<td>Very Large</td>
<td>8.5</td>
<td>15.8</td>
</tr>
<tr>
<td>Mean</td>
<td>5.5</td>
<td>16.0</td>
</tr>
<tr>
<td>MAD</td>
<td>1.6</td>
<td>0.3</td>
</tr>
</tbody>
</table>

Legend: T=Traditional, A=Agile, B=Both.

Table 6: Low-affinity cluster excluded after applying first spectral analysis with two centers.

<table>
<thead>
<tr>
<th>Development Approach</th>
<th>#</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rational Unified Process (custom variant)</td>
<td>1</td>
</tr>
<tr>
<td>Rational Unified Process (standard version)</td>
<td>0</td>
</tr>
<tr>
<td>Spiral Model</td>
<td>1</td>
</tr>
<tr>
<td>Disciplined Agile Delivery (DAD)</td>
<td>0</td>
</tr>
<tr>
<td>Feature-Driven Development (FDD)</td>
<td>3</td>
</tr>
<tr>
<td>Crystal Family</td>
<td>0</td>
</tr>
</tbody>
</table>

Approaches, e.g., 28 participants state that they combine Scrum and Continuous Integration. Using this combination matrix and as described in Section 3.4, we performed the AP and SC cluster analyses. The first AP analysis resulted in six clusters, which revealed development approaches with little or no affinity to other approaches. Therefore, we split the result set into two subsets \( A_{high} \) and \( A_{low} \) using an SC analysis with two centers. Table 6 summarizes the approaches that were sorted into \( A_{low} \), and which we excluded from further analyses.

Running the AP analysis on the subset of the remaining 33 development approaches in \( A_{high} \) resulted into five centers and, therefore, we applied the trained SC algorithm again searching for clusters around the five centers. The resulting clusters are illustrated in Table 7 with sizes ranging from four to eight elements. Given the categorization (traditional/agile) of the approaches (Figure 2) all five clusters show a mixture of traditional, agile, and generic (both) methods and practices, e.g., Cluster 5-4. However, remarkable is Cluster 5-5, which shows a collection of approaches as one would expect from a practically applied Scrum adaptation [7].

Therefore, in order to investigate if we are able to construct the two “extremes” (i.e., agile and traditional approaches) from our result set, we executed the SC algorithm again with two centers to split the result set. Table 8 shows the outcome: while on the one hand, the algorithm isolated elements characterizing an “almost pure” Scrum-centered development process (Cluster 2-2), one the other hand, it was not able to construct a pure traditional cluster. In fact, Cluster 2-1 represents mixture of development approaches from all categories.
Table 7: High-affinity clusters of development approaches after applying spectral analysis with five centers.

<table>
<thead>
<tr>
<th>Cluster 5-1</th>
<th>Cluster 5-2</th>
<th>Cluster 5-3</th>
<th>Cluster 5-4</th>
<th>Cluster 5-5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Size=4</td>
<td>Size=5</td>
<td>Size=8</td>
<td>Size=8</td>
<td>Size=8</td>
</tr>
<tr>
<td>XP</td>
<td>V-Model Derivates(s)</td>
<td>Iterative Development</td>
<td>Waterfall/Phase Model</td>
<td>Scrum</td>
</tr>
<tr>
<td>SAFe</td>
<td>APM</td>
<td>Continuous Deployment</td>
<td>BDD</td>
<td>Collective Code Ownership</td>
</tr>
<tr>
<td>Formal Estimation</td>
<td>Lean Development</td>
<td>Continuous Integration</td>
<td>DevOps</td>
<td>Daily Standups</td>
</tr>
<tr>
<td>On-Site Customer</td>
<td>Definition of Ready</td>
<td>Iteration Planning</td>
<td>Kanban</td>
<td>Definition of Done</td>
</tr>
<tr>
<td></td>
<td>Expert-/Team-based Estimation</td>
<td>Unit Testing</td>
<td>TDD</td>
<td>Digital Task Board</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Retrospectives</td>
<td>Prototyping</td>
<td>Pair Programming</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Release Planning</td>
<td>Formal Specification</td>
<td>Refactoring</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Coding Standards</td>
<td>LeSS</td>
<td>Code Review</td>
</tr>
</tbody>
</table>


Table 8: High-affinity clusters after applying spectral analysis with two centers.

<table>
<thead>
<tr>
<th>Cluster 2-1</th>
<th>Cluster 2-2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Size=17</td>
<td>Size=16</td>
</tr>
<tr>
<td>V-Model Derivates(s)</td>
<td>Scrum</td>
</tr>
<tr>
<td>Waterfall/Phase Model</td>
<td>Iterative Development</td>
</tr>
<tr>
<td>APM</td>
<td>Collective Code Ownership</td>
</tr>
<tr>
<td>BDD</td>
<td>Continuous Deployment</td>
</tr>
<tr>
<td>DevOps</td>
<td>Continuous Integration</td>
</tr>
<tr>
<td>Extreme Programming</td>
<td>Daily Standups</td>
</tr>
<tr>
<td>Kanban</td>
<td>Definition of Done</td>
</tr>
<tr>
<td>LeSS</td>
<td>Iteration Planning</td>
</tr>
<tr>
<td>Lean Development</td>
<td>Pair Programming</td>
</tr>
<tr>
<td>SAFe</td>
<td>Refactoring</td>
</tr>
<tr>
<td>Prototyping</td>
<td>Retrospectives</td>
</tr>
<tr>
<td>Formal Estimation</td>
<td>Expert-/Team-based Estim.</td>
</tr>
<tr>
<td>Formal Specification</td>
<td>Code Review</td>
</tr>
<tr>
<td>Definition of Ready</td>
<td>Coding Standards</td>
</tr>
<tr>
<td>Digital Task Board</td>
<td>Unit Testing</td>
</tr>
<tr>
<td>On-Site Customer</td>
<td>Release Planning</td>
</tr>
<tr>
<td>TDD</td>
<td></td>
</tr>
</tbody>
</table>

Table 9: Does the company have a company-wide standard process for software development? (n=69)

<table>
<thead>
<tr>
<th>Option</th>
<th>#</th>
<th>%</th>
</tr>
</thead>
<tbody>
<tr>
<td>All projects are operated according to the same (customized) standard process(es)</td>
<td>36</td>
<td>52.2</td>
</tr>
<tr>
<td>Each business unit has its own approaches, which all projects of this unit have to follow</td>
<td>14</td>
<td>20.3</td>
</tr>
<tr>
<td>Each project can individually select the process to be used</td>
<td>19</td>
<td>27.5</td>
</tr>
</tbody>
</table>

Table 10: How was the particular combination of development approaches created? (n=56)

<table>
<thead>
<tr>
<th>Option</th>
<th>#</th>
<th>%</th>
</tr>
</thead>
<tbody>
<tr>
<td>Planned in a process improvement program</td>
<td>11</td>
<td>19.6</td>
</tr>
<tr>
<td>Evolved from past projects over time</td>
<td>47</td>
<td>83.9</td>
</tr>
<tr>
<td>Situation-specific</td>
<td>13</td>
<td>23.2</td>
</tr>
</tbody>
</table>

Table 11: Overview of the actual process selection and tailoring in particular projects. (n=56)

<table>
<thead>
<tr>
<th>Option</th>
<th>#</th>
<th>%</th>
</tr>
</thead>
<tbody>
<tr>
<td>Practices and methods are selected in the project on demand</td>
<td>30</td>
<td>53.6</td>
</tr>
<tr>
<td>Practices and methods are selected according to customer demands</td>
<td>14</td>
<td>25.0</td>
</tr>
<tr>
<td>A project manager tailors the process in the beginning of a project</td>
<td>19</td>
<td>33.9</td>
</tr>
<tr>
<td>Project-specific process selection and tailoring follows defined rules</td>
<td>20</td>
<td>35.7</td>
</tr>
<tr>
<td>Project-specific process selection and tailoring is supported by tools</td>
<td>13</td>
<td>23.2</td>
</tr>
<tr>
<td>The process is not tailored at all</td>
<td>12</td>
<td>21.4</td>
</tr>
<tr>
<td>Other</td>
<td>2</td>
<td>3.6</td>
</tr>
</tbody>
</table>

4.3.3 Process Use (Institutionalization). Finally, we study the way the development approach is composed and used in projects. Therefore, we asked the participants for company-wide policies, development of their particular development approach, and decision-making in regard to the process selection. Table 9 shows that more than the half of the participants (52.2%) state to be obliged to follow a standard process. Another 20.3% have a standard at the level of business units, and the remaining 27.5% state to select the actual process on demand.

That is, more than 3/4 of the participants state to have rules concerning the process use. After collecting the data regarding the development approaches used (cf. Section 4.2), we asked the participants if they combine the different approaches within projects, and 56 participants stated to explicitly combine different development approaches. We asked the participants how their particular combination of the different development approaches was developed, and a majority of 83.9% (Table 10) stated that the development approach emerges from experience and learning from past projects.

---

6Rationale: Accepting that companies might run projects for different clients, different methods might be known at the company level, but there is the possibility of an exclusive use on a per-project or per-client basis, i.e., multiple approaches but no combined use. Table 2, question 10 addresses this situation.
Finally, we were interested in the decision-making process, i.e., how are development approaches selected and by whom. The results summarized in Table 11 show that more than the half of the participants (53.6%) state selecting the methods and practices in the project on demand. Furthermore, 35.7% state that the project-specific tailoring follows defined rules and that a project manager carries out the tailoring in the beginning of a project (33.9%). Hence, this shows that standards exist and that they are (at least initially) applied to a project. However, during the project, the configuration of methods and practices might change.

As shown in Table 10, we asked participants, whether their companies implement a Software Process Improvement (SPI) program to investigate if the development approach used emerges from a planned and/or controlled procedure. Table 12 shows that only 15.9% have implemented an SPI program based on standardized approaches like ISO/IEC 15504 or CMMI, but the majority states implementing an in-process improvement (37.7%) or no SPI program at all (18.8%).

## 4.4 RQ3: Triggers for Hybrid Approaches

While the first two research questions are concerned with studying which development approaches are used and how they are combined with each other, the third research question aims to investigate the triggers for creating a hybrid approach. For this, we argued that external standards, norms, and regulations are major triggers introduced by the increasing complexity of software-intensive systems, especially by emerging industry sectors engaged in Cyber-physical systems (see also the hypotheses in Table 3). In this section, we study standards and whether standards foster (or even enforce) the creation of hybrid development approaches.

### 4.4.1 Use of Standards

We asked the participants, if they have to adhere to external standards (and to which ones), and if agility is a challenge in the light of those standards. Figure 4 illustrates the result as a word cloud. The participants named 45 different standards and standards-related methods of which ISO 9001 was the most frequently mentioned (12 mentions), followed by ISO 27001 (5 mentions), and ISO 13485 (4 mentions). The summary in Figure 4 shows a considerable variety of standards, and Table 13 shows whether using standards, norms, and regulations challenges the companies in implementing agility.

In total, 41 participants stated external standards, norms, and regulations relevant for their companies—even micro-sized companies have to adhere to standards. The column “Challenged by Agility?” of Table 13 refers to the 41 participants stating that they have to use standards and norms, whereas 23 participants (approx. 53%) are not triggered by external standards. That is, we have no evidence for H2, i.e., the obligation to use external standards does not depend on the company size. Our data (Table 13) suggests that the creation and use of hybrid approaches does not depend on the size of the company. As also shown in Table 5, our data suggests that companies use hybrid approaches—regardless of their size. We have no evidence for H3, i.e., the use of hybrid approaches does not depend on the company size. Our data (Table 13) suggests that standards have become relevant to companies of all kinds.

### 4.4.2 What Triggers Hybrid Approaches?

Finally, we aim to investigate the triggers in more detail. In Table 14, we formulate four hypotheses to test. Table 14 presents the results from the tests using Pearson’s $\chi^2$. The table shows that we found no support for any of the hypotheses. In particular, we have no evidence for H1, i.e., the use of hybrid approaches does not depend on the company size. As also shown in Table 5, our data suggests that companies use hybrid approaches—regardless of their size. We have no evidence to support H2, i.e., the obligation to use external standards does not depend on the company size. Our data (Table 13) suggests that standards have become relevant to companies of all kinds. Data also suggests that the creation and use of hybrid approaches is not triggered by external standards. That is, we have no evidence supporting H3. Finally, we also have no evidence for H4, i.e., the use of hybrid approaches does not depend on external triggers. Especially for H3 and H4, we can find rationale in the open questions. For instance, Table 10 shows 83.9% of the participants stated that the hybrid approach has emerged from experience.
analysis of the open questions reveals that 18 participants state the current hybrid approach being a result of pragmatically applied development approaches that evolved over time.

In our dataset there was no proof of dependance for any of the hypotheses. Hence, we conclude that the use of hybrid approaches can be considered independent of the company size (also of the industry sector, cf. Section 4.3.1), and external standards do not trigger the creation and use of hybrid approaches. Moreover, our data suggests that a hybrid approach is a result of a natural evolution of the different development approaches used by companies. But, even though the data presented supports this claim, a deeper investigation remains subject to future work.

4.5 Discussion
This study is grounded in observations made in Theocharis et al. [20]. In particular, this study aimed at collecting data to allow for closing a gap identified, namely missing data about process use in general including agile and rich processes. In addition—and in the light of modern software system development—we added an extra dimension by including standards, norms, and regulations to our research.

In this context, the HELENA study produced a list of development approaches as used in practice (Table 1, RQ1). Compared to related studies, e.g., [5, 11, 15, 22, 23], our results show a good match. Traditional as well as agile methods and practices are present, and the trends emerging from our data are in line with previous studies, e.g., [15, 23]. West et al. [24] named this trend “Water-Scrum-Fall”, and our results support West’s claim that the “Water-Scrum-Fall” has become reality. For instance, based on the descriptive analysis of our categorized data, we could show that companies tend to implement a balanced software development approach that includes traditional as well as agile elements (Figure 3). The overall tendency shows risk management and configuration management implemented in a more traditional way, while the activities around requirements engineering, implementation, integration, and testing tend to be implemented in a more agile fashion.

Our data shows no evidence supporting the claim that the implementation of hybrid approaches depends on the company size (Table 5 and H1) or on the industry sector (Table 1, RQ3). In the detailed analysis, we also found no indication that standards or external triggers drive the development of hybrid approaches (Section 4.4.2, H2-4). However, our data also shows that companies are active in business areas enforcing requirements to adhere to standards on the companies, and that notably implementing agility in such standards-driven environments challenges companies (Table 13). While several industry studies (e.g., [17, 21]) argue that hybrid approaches are caused by a reluctance of the management to buy-in agile, we argue that implementing hybrid approaches is also an attempt to address multiple challenges, such as balancing management and developer expectations regarding the development process or implementing (rigorous) standards by, at the same time, keeping high levels of flexibility. A specific challenge to be addressed by hybrid approaches is scalability of agile methods. For instance, Murphy et al. [17] found Scrum to be considered the most favorite development approach at Microsoft. However, the suitability of agile methods for large projects was considered critical (substantial communication effort and overhead, reluctance of the management to accept the agile approach). This finding is supported by Melo et al. [6], who found the management-related agile practices to be either adopted to a large extent or completely rejected, while development-centered practices have become well-accepted.

From our data, we conclude that hybrid approaches can be considered a good compromise that helps balancing the needs of different stakeholder groups. As for instance found in [6, 17], management has different requirements and expectations concerning the development process than developers. A cluster analysis (Sect. 4.3.2, Tables 7 and 8) shows that hybrid approaches include development approaches from both worlds (traditional and agile). This combination addresses the needs of managers (more traditional methods and frameworks to support “classic” management tasks) and developers (freedom to select those practices best fitting the respective context). Tables 7 and 8 show that hybrid approaches are not limited to combinations of traditional and agile methods. Moreover, especially Table 8 shows that even agile methods are not implemented by the book. Rather, different practices are combined to address practical needs (Table 1, RQ2; see also [7]). To a large extent, such combinations are developed on a per-project basis (Table 11 and [15]) and are continuously improved within the projects (Table 12). The inspection of the pair-wise coincidence matrix shows that the fine-grained development-related practices (e.g., unit testing, code reviews, pair programming, and retrospectives) are extensively combined with each other. In our previous study [20], we claimed that practices have become the major building blocks of process customization. A claim that is supported by the study at hand.

4.6 Threats to Validity
Despite the rigorous development procedure of the survey instrument (Sect. 3.2), still, our study faces some threats to validity, which we discuss in the following.

Internal Validity. The internal validity might be threatened by the questionnaire as such. To increase the internal validity, we used questionnaires from previously conducted studies (e.g., [3, 11, 15]) as reference for the instrument development. Furthermore, we conducted an iterative validation phase, and internal and external reviews to increase the internal validity via researcher triangulation.

External Validity. The external validity might be threatened by low number of participants, the participants’ self-reporting, and the limited number of regions included in the study, which might affect the generalizability of our results. To increase the external validity, external reviews and trails with industry practitioners were conducted prior to the study’s launch. Furthermore, results were compared with previous studies to find a reference for data interpretation. However, in order to generalize the results, further research in more regions is necessary.

5 CONCLUSION
This paper presents findings from the HELENA project with which we study the use of hybrid software development approaches. An internationally conducted survey provided 69 complete responses from which we extracted a list of software development approaches used in practice. We categorized and analyzed the processes used
and found hybrid approaches to be widely used in practice. Our study revealed that hybrid approaches have become mainstream and are used by companies regardless of company size and industry sector. While standards, norms, and regulations challenge companies, even in regulated domains, companies adopt agile methods. An empirical analysis confirmed that there is no evidence to claim that the development and use of hybrid approaches are triggered by company size or external standards. Hybrid approaches used in practice today emerge from pragmatic process selection and evolve over time. The cluster analysis supports West’s “Water-Scrum-Fall” hypothesis by showing that combinations of development approaches follow a pattern in which a traditional process serves as framework refined by (multiple) fine-grained practices. We further argue that individual practices, rather than large methods, have become the building blocks for process customization.

Limitations. The main limitation of the study presented in the paper at hand is the population. The data reported and analyzed is mostly coming from participants that are either located or involved in projects within Europe. Furthermore, the selected sampling strategy was a convenience sampling. That is, the aim has been to collect as many data points as possible with little regards to neither controlling the response rate nor the distribution over, e.g., industry sector and roles. Industry sectors and roles cover a broad spectrum, but are not evenly distributed. These aspects limit the generalizability of our result set. However, at this point, our analysis does not yet attempt to provide a generalizable and complete picture. Rather, the present study aims to improve our understanding of the software system development approaches in practice, yet calls for future work to eventually allow for generalizability.

Future Work. Future—already ongoing—steps of the HELENA project are in line with the aforementioned limitations. The survey instrument (Table 2) is receiving a revision (refined scope based on findings obtained in the first stage) before initiating the second stage of the project. Furthermore, by growing the international network (more than 20 countries from various continents) for a second stage, the study is no longer limited to the European context. Finally, due to expected larger data base, it will be possible, e.g., to develop statistical models grounded in evidence, test further hypotheses such as multivariate analyses, and, eventually, create focused research groups to study specific areas if interest more thoroughly, e.g., via interviews.

REFERENCES
Requirements Volatility in Software Architecture Design: An Exploratory Case Study

Sanja Aaramaa
M3S
Faculty of Information Technology and Electrical Engineering
University of Oulu
Finland
sanja.aaramaa@oulu.fi

Sandun Dasanayake
M3S
Faculty of Information Technology and Electrical Engineering
University of Oulu
Finland
sandun.dasanayake@oulu.fi

Markku Oivo
M3S
Faculty of Information Technology and Electrical Engineering
University of Oulu
Finland
markku.oivo@oulu.fi

Jouni Markkula
M3S
Faculty of Information Technology and Electrical Engineering
University of Oulu
Finland
jouni.markkula@oulu.fi

Samuli Saukkonen
M3S
Faculty of Information Technology and Electrical Engineering
University of Oulu
Finland
samuli.saukkonen@oulu.fi

ABSTRACT
Requirements volatility is a major issue in software (SW) development, causing problems such as project delays and cost overruns. Even though there is a considerable amount of research related to requirement volatility, the majority of it is inclined toward project management aspects. The relationship between SW architecture design and requirements volatility has not been researched widely, even though changing requirements may for example lead to higher defect density during testing. An exploratory case study was conducted to study how requirements volatility affects SW architecture design. Fifteen semi-structured, thematic interviews were conducted in the case company, which provides the selection of software products for business customers and consumers. The research revealed the factors, such as requirements uncertainty and dynamic business environment, causing requirements volatility in the case company. The study identified the challenges that requirements volatility posed to SW architecture design, including scheduling and architectural technical debt. In addition, this study discusses means of mitigating the factors that cause requirements volatility and addressing the challenges posed by requirements volatility. SW architects are strongly influenced by requirement volatility. Thus understanding the factors causing requirements volatility as well as means to mitigate the challenges has high industrial relevance.

CCS CONCEPTS
Software and its engineering~Requirements analysis, Software and its engineering~Software design engineering, Software and its engineering~Software development process management

KEYWORDS
Requirement management, software architecture

ACM Reference format:
DOI: 10.1145/3084100.3084105

1 INTRODUCTION
Requirements changes affect throughout all software (SW) development lifecycle, for example leading to higher defect density in testing phase [51]. Due to various internal and external factors, changes in individual requirements or in sets of requirements are inevitable. The changing nature of requirements in requirements engineering (RE) is denoted by requirements volatility. [10]. Since they are responsible for system structure, SW architects are the stakeholders that are greatly affected by requirements volatility [8], [17]. SW architecture design plays a prominent role in SW development, as it acts as the foundation of the SW systems and shapes the final outcome. SW architects must make critical decisions based on requirements [24]. Making sub-optimal architecture decisions can decrease system quality and cause problems later on [14]. Changes in requirements may necessitate the redesign of the SW to accommodate those changes, leading to an unstable SW system [26]. Correcting system shortcomings after completing
system development is very complicated and far costlier than identifying and addressing them during initial design [51]. Existing literature discusses requirements volatility mainly from the project management viewpoint [40] and it has been studied related to SW maintenance [52] and coding [25]. But empirical evidence about requirements volatility from the SW architects’ viewpoint and interplay between RE and SW architecture design in industrial environments is not extensively addressed. To provide empirical insights into requirements volatility from SW architects’ point of view, an industrial case study was conducted to explore requirements volatility in the context of SW architecture design. Fifteen SW architects involved in architecture design were interviewed for the case, the objective of which was to identify challenges that SW architects face due to requirements volatility and to propose means to address those challenges. The following research questions were derived from the objectives of the case.

RQ1: What are the factors that cause requirement volatility?

RQ2: What challenges does requirement volatility pose to SW architecture design?

RQ3: What are means to address the identified challenges in SW development?

This paper is structured as follows. Section 2 outlines related scientific literature. Section 3 describes the research approach. Section 4 provides an overview of the software development process in the case company and answers RQ1, describing the challenges identified based on the interviews. Section 5 discusses the means to address the identified challenges, based on existing scientific literature and the researchers’ experience, thereby answering RQ2. Finally, Section 6 discusses conclusions.

2 RELATED WORK

RE is a systematic way to elicit, organise and document system requirements as well as a process to establish and maintain agreement between a customer and a systems provider [32]. Many authors emphasise the nature of change in the RE process [59], [35], [31]. In addition, the requirements management process is defined as a process of managing changes in the requirements [29]. Nurmuliani [43] defines requirements volatility as ‘the tendency of requirements to change over time in response to the evolving needs of customers, stakeholders, the organisation and the work environment’. Requirements volatility may also be defined as a change that could occur to a requirement [53]. This is in line with Nurmuliani’s operationalisation definition of requirements volatility [43]. Several internal and external factors, such as stakeholder feedback, technology, the market situation and customer needs, can cause changes in requirements [38]. Christel [10] groups requirements elicitation problems into three categories: scoping (information mismatch), understanding (inter- and intra-group) and volatility (requirement changes). The major reason for requirements volatility is changing user needs [48], which primarily lead to changes in individual requirements. Other important factors affecting volatility are conflicting stakeholder views and complexity of organisation [18], which lead to changes in the content of forthcoming releases. In addition, problems in understanding and scoping cause volatility [10]. Requirement information mismatch is also denoted requirement uncertainty [40], [39]. Existing literature on requirements volatility emphasises management viewpoint [58], [45]. These studies sought factors causing requirements volatility [10], its effects on projects [61] and means to mitigate those effects [1]. In addition, requirements volatility has been investigated in relation to SW maintenance [52] and coding [25], but not extensively in the context of SW architecture design. In short, requirements volatility is a phenomenon caused by various internal and external factors that lead to changes in the single requirements or sets of requirements. Changes in the sets of requirements have been claimed to result in more severe consequences than those in one requirement [16]. Applying an iterative RE process model has been proposed as a means to tackle requirements volatility [34]. For example, the twin peaks model is an iterative approach in which requirements and software architecture are developed in parallel from the very beginning of the software development life cycle [44].

SW architecture is the foundation that the SW system is built upon. The purpose of designing the SW architecture is to provide a unified vision about the system and improve understanding of its behaviour. The architecture, including diagrams, use cases and semantics, reduces ambiguities and shortens the time it takes for stakeholders to understand the constraints, behaviour, timing, layout for instance. Stakeholders involved in SW architecture design must make various decisions throughout the SW system life cycle regards to development, evolution and integration [24]. SW architecture design is inherently complex, and complexity is further increased because the architecture must address various stakeholder concerns [55] that may conflict with achieving SW system development goals. According to recent empirical studies on SW architecture design decision-making, in industrial environments, SW architects use experience, intuition and other informal approaches rather than using formal tools and techniques [15], [57]. SW architecture design is considered primarily as the SW architect’s responsibility [13]; nevertheless, the active involvement of other stakeholders, such as SW developers, product managers and customers, is crucial for achieving the better understanding about the criteria the architecture must meet. An important element of the architecture design process is recording architecture design rationale, as understanding the reasons behind a certain architecture design decision can be critical during SW system maintenance and evolution [56]. Architectural technical debt refers to the consequences faced late in the SW development process due to sub-optimal architecture decisions and trade-offs [33], [14]. SW teams accumulate architectural technical debt due to their own actions and due to external events related to natural software aging and evolution. Even though technical debt related to coding issues can be detected using various tools, architecture technical debt mostly remains invisible and grows over time. [30] Factors that contribute to accumulating architectural debt include uncertainty about requirements at the beginning of development, the introduction
of new requirements during the SW development process, time pressure, feature-oriented prioritisation and specification issues with critical architectural requirements [36].

The RE process and SW architecture design are called "twin peaks" and considered equally important [37]. Since they are closely connected, decisions made regarding one can affect the other [44]. Even though the traditional waterfall model leads to freezing requirements before moving into design and making hard to change architecture decisions, in reality the changes can occur in both areas and they can affect one another [11] Non-functional requirements (NFRs) constitute the majority of stakeholder concerns and greatly influence shaping up the architecture. The interactions between NFRs are among the main factors that should be taken into consideration during architecture design, as architecture either allows or precludes almost all NFRs of a system [12]. SW architecture design decisions are primarily based on Architecturally Significant Requirements (ASRs), which are critical to shaping system architecture [7]. While NFRs take the large share of ASRs due to their ability to affect the whole SW system, it can also include functional requirements. Correctly identifying and classifying architecturally significant functional requirements (ASFRs) is also critical for an architect to make informed decisions [41], [6]. The modern iterative software development approaches facilitate close interaction between requirements and architecture and help making rapid adjustments.

3 RESEARCH METHOD

The research method selected for this study was the case study, because the objective of the study was to research the requirements volatility in SW architecture design in the industrial setting [47]. The case study method is well suited for the researching real life phenomena in its natural setting [60]. Figure 1 depicts the phases of the case study.

Case context: The case study was conducted in a company with more than 900 employees in 25 offices worldwide. However, the product development is mainly done in three countries. The case company is a comprehensive SW solutions provider to both private and business customers and SW products for private customers. The SW solutions offered for business customers include a number of management tools and company services for the worldwide market. In addition, the case company provides various SW products for private customers to be used on an array of devices, ranging from desktops to handheld mobile devices. The SW development activities in the case company are carried out mainly in three countries.

In the case company there are three parallel business lines: U1, U2 and U3. The units are divided based on their business focus and each unit have their own financial responsibilities. There also is a horizontal business line, U4, which is responsible for providing services that are commonly shared by other three business units. The business lines operate as independent entities and within the business lines, there is a flat hierarchy of teams mostly organised based on projects. Along with the business units, a special unit, U5, consists of technical experts who take company-wide decisions related to technical matters. Individual teams are mostly self-organising and typically consist of four to eight people. While teams are free to operate according to their own agendas, they might have to interact and align with other teams, depending on the nature of the project. Some teams have their own architect or scrum master, but this is not the case for every team.

<table>
<thead>
<tr>
<th>ID</th>
<th>Unit</th>
<th>Responsibilities</th>
<th>The years of experience (in the case company)</th>
</tr>
</thead>
<tbody>
<tr>
<td>11</td>
<td>U1</td>
<td>SW dev./ Architect</td>
<td>18 (17)</td>
</tr>
<tr>
<td>12</td>
<td>U1</td>
<td>SW dev.</td>
<td>7 (2)</td>
</tr>
<tr>
<td>13</td>
<td>U1</td>
<td>SW dev./ Architect</td>
<td>5 (3)</td>
</tr>
<tr>
<td>14</td>
<td>U1</td>
<td>SW dev.</td>
<td>20 (2)</td>
</tr>
<tr>
<td>15</td>
<td>U2</td>
<td>SW dev./ Team lead</td>
<td>15 (15)</td>
</tr>
<tr>
<td>16</td>
<td>U2</td>
<td>SW dev./ Team lead</td>
<td>20 (20)</td>
</tr>
<tr>
<td>17</td>
<td>U2</td>
<td>SW dev./ Architect</td>
<td>14 (8)</td>
</tr>
<tr>
<td>18</td>
<td>U3</td>
<td>Program/ Project lead</td>
<td>13 (7)</td>
</tr>
<tr>
<td>19</td>
<td>U3</td>
<td>SW dev./ Architect</td>
<td>13 (3)</td>
</tr>
<tr>
<td>10</td>
<td>U3</td>
<td>SW dev.</td>
<td>22 (15)</td>
</tr>
<tr>
<td>11</td>
<td>U3</td>
<td>SW dev./ Architect</td>
<td>13 (9)</td>
</tr>
<tr>
<td>12</td>
<td>U3</td>
<td>SW dev./ Architect</td>
<td>24 (4)</td>
</tr>
<tr>
<td>13</td>
<td>U4</td>
<td>SW dev.</td>
<td>15 (5)</td>
</tr>
<tr>
<td>14</td>
<td>U4</td>
<td>SW dev.</td>
<td>19 (6)</td>
</tr>
<tr>
<td>15</td>
<td>U5</td>
<td>Lead architect</td>
<td>15 (14)</td>
</tr>
</tbody>
</table>

Table 1. The details of the interviews.

Case study design and preparation. An interview guide, which consisted of the set of open ended questions grouped into different themes, was prepared to guide the interviews. The interview guide consisted of the following themes; a general background, SW development process, requirements engineering...
and SW architecture design, SW architecture design challenges, solutions and expectations. The background questions covered the general information such as the experience of the interviewee, typical project and team composition as well as company description. The SW development process theme covered the overall development life cycle. The rest of the themes were designed to go into a deep discussion about requirement engineering and SW architecture design activities. The final part targeted, the various challenges faced by the interviewees and the possible solutions from their perspectives.

A pilot interview was conducted to test the interview guide as well as the interviewers approach. The participant of the pilot interview had a long history working in industry as a SW developer and a SW architect. In addition, the feedback received from the company representatives was also used to improve the interview guide. The reviewers of the questionnaire and the pilot interview participant did not participate in the actual case study.

Data collection. Fifteen semi-structured, thematic [20] interviews were conducted with SW experts during November and December 2014 as the primary method of data collection. All the interviewees acted as software architects in their respective teams even though not all of them are titled as architects (cf. Table 1). The interviewees were selected to represent all five units in the company’s technical organisation. Interviewees have different levels of work experience, are active in various projects and are located at three company sites. The experience given in Table 1 reflects the years of experience in SW development in industry. The number without brackets is about total while the one in brackets is for the years in the case company. Twelve interviews of 1-2 hours in length were conducted by two researchers face to face (F2F) at case company sites, and three interviews of the same duration were conducted via Skype. Interviewees were provided in advance with the list of detailed questions emerged during the interviews.

Data analysis. The interviews were professionally transcribed and fed into NVivo, a qualitative data analysis tool, as the starting point of the data analysis. The researchers went through each of the transcribed interviews and labelled the relevant information using a pre-defined set of themes based on the interview guide. Initially, there were the limited set of themes based on the research questions and as the process going forward, new themes were emerged and the sub themes were also created. The themes used during the data analysis are shown in Figure 2. The themes are not mutually exclusive and the same data can be labelled with multiple labels belongs to different themes.

This case study adapted the reporting guidelines provided by [47]. A strict privacy policy was followed that described all necessary elements of the case study while protecting the integrity of the company and individuals [4]. Interviewees’ identities are protected by, for example, using aggregated information instead of presenting interview excerpts and by avoiding use of corresponding IDs in tables.

The results are based on the interviews and publicly available information about the company. The topics represent the aggregated viewpoints of the respondents throughout the interviews. The issues mentioned only by a single interviewee have not been included in the results, as they were considered not expressing shared understanding among SW architects. At first, an overview of SW development practices are outlined, which helps to understand the development context. This is followed by the description of requirements volatility challenges in the case company.

4 SW DEVELOPMENT PROCESS

SW development practices in the case company are quite informal and vary from team to team. Even though the company has a history of experimenting with various lean and agile SW development approaches on the company level, at the moment, there is no company-wide SW development approach. SW teams are free to select their own approaches unless there are specific restrictions such as customer preferences. SW teams tend to create their own approach by selecting and combining various agile and lean practices, including following sprints, maintaining product backlog and using Kanban boards. Although interviewees’ have responsibility for SW architecture design, their involvement in RE is limited to occasionally providing expert opinion. Thus, interviewees’ understanding about RE is not as extensive as is their knowledge of SW architecture design.

4.1 Requirements Engineering

SW architects are not directly involved in customer requirement elicitation and analysis. In the case company, elicitation is accomplished using techniques such as focus groups, beta testers, and direct customer communication. Occasionally, customers’ ideas are expressed at so abstract a level that they can hardly be translated as requirements. On the other hand, it is possible that requirements will be stated as full-scale technical specifications. Customer needs are clarified during the requirement analysis. The SW product owner (PO) is a link between the SW team and product management (PM). The PO
has the responsibility to communicate with the PM to clarify what must be implemented. For the majority of interviewees, the PO is the sole connection point to requirements elicitation and analysis phases. During the clarification process, SW architects are occasionally consulted to define the technical feasibility of requirements and choose the best implementation solution. At this phase, requirements volatility factors include changing customer needs and evolving technological understanding.

A project management tool (PMT) is the main medium for documenting requirements, which usually are expressed as features or backlog items. In addition to the primary PMT, legacy and team-specific tools and sticky notes are used to communicate tasks, store customer information and document decision rationales. The level of details in information on requirements varies depending on a product and a team. Sometimes, only a feature name is recorded, but at the other extreme, descriptions include even the contact information of the relevant technical specialist on the customer side. In the case of private customer products, requirements are created by experts based on a foreseen market. Usually, the creator of a requirement is recorded, but sometimes it is not known whether the requirement originated with a customer or an internally identified technology gap. Interviewees related that they sometimes needed more technical details or contextual descriptions to be able to choose the best implementation alternative. Big corporate customers may have strict requirements about formal documentation to be delivered to the customer. Interviewees pointed out that a requirement description is always a compromise between level of detail and time available for the task.

The PM is responsible for prioritising requirements, taking into account, for example, company strategies and the importance of the customer. The dominant factor when setting requirement priority is the customer: The bigger the customer, the higher the priority of its requirements. Even though some features are technically feasible and could contribute improving the product quality in the long run, it may be very difficult to say no to features that a customer wants, especially if the customer is big. Other factors taken into account when setting requirement priorities include development cost, feature size, product roadmap, criticality and external audit results, which are publicly available and used to rank the solution providers in the domain. Interviewees were involved in requirement prioritisation, proposing product improvements and project scoping meetings. Most interviewees mentioned having faced challenges with changing requirement priorities. As the backlog is updated frequently, changing priorities contribute to requirements volatility.

4.2 Software Architecture Design
Since the majority of the teams follow agile and lean approaches, the design and implementation are done iteratively, leading to a shorter design phase than in the traditional waterfall approach. The SW architecture design process typically starts with backlog review meetings between the team and other relevant stakeholders. The objective of these meetings is to reach a consensus what needs to be developed to fulfil requirements. While the team is generally represented by senior members during the initial meeting, it is possible that the whole team is involved from the beginning. Once the basic ASRs are understood, the team creates a design proposal, which is delivered for review. The review is done at different levels, depending on the scale of the project or its dependencies to other products. Once decisions are made, the team is free to begin development and has the flexibility to make minor changes to the design. If the design must be altered considerably, the evaluation of the alterations is escalated. SW architecture designs and decisions taken during discussions at various levels are recorded using several methods. Even though the interviewees claimed that they have maintained some type of design documentation, attention to documenting design appears to be inadequate. The majority of teams use tools such as Wiki or PMT instead of traditional design documents to store their architecture decisions.

5 REQUIREMENT VOLATILITY – ORIGINS AND CHALLENGES
Through data analysis, it was possible to identify factors that cause requirements volatility in the case company as well as the challenges that requirements volatility poses to SW architecture design. This section first describes the factors of requirements volatility identified in the case company and then the consequences of requirements volatility. Thus the subsection Error! Reference source not found. answers RQ1 and the subsection Error! Reference source not found. answers RQ2. Quotes from interviews provide insights for collected empirical evidence. Each quote is from a different interview. However, to protect the integrity of the respondents the quotes are not labelled by the respondent ID. None of the respondents was a native English speaker. Therefore, it has been necessary to correct some minor language errors to ensure a proper message.

5.1 Factors Causing Requirements Volatility
Several factors that contribute to requirements volatility were identified in the case company.

5.1.1 Requirement Uncertainty. An important factor for causing requirements volatility is the uncertainty of requirements, which is realised for example as inadequate descriptions of backlog items. Often, features or backlog items lack detailed information; for example, a backlog item may have only a name, but no one knows why the item is there. The tool includes a customer acceptance criteria field as part of the requirement description. Most of the time, something is recorded in this field. However, often, the description is a couple of lines of text at an abstract level. This means that architects and testers must guess what must be fulfilled.

"It [description] can be just couple lines of text and that’s all and we need to guess what shall we do... quality engineers always
complain about it because they don’t know how to test because it’s not so clear how it should work."

5.1.2 Changing User Needs. The case company provides multiple SW products for various customer groups and frequently comes across changing customer needs. As most of the requirements for private customer products are decided within the company, corporate customers are the main source of requirements changes. The long-term business relationship between corporate customers and the company makes it difficult to refuse to adapt to changing customer needs.

“Well, since we are doing this project with, constantly changing requirements. I don’t see much chance for improving the process because we are just, basically adapting. And not planning ahead.”

5.1.3 Dynamic Business Environment. The company operates in a dynamic business domain and must adjust its strategies for accommodating development in that domain to stay ahead of the competition. The severe market situation requires constant changes in requirement priorities. As most of the company’s private customer products run on smartphones where the operating systems are highly fragmented and subjected to frequent changes, the company has to make frequent changes to their products to accommodate those changes.

“This list we see for quarter is something that we can work on. Whatever in future is, at least, that is subject to change because market change, situation change and stuff like that. So we wouldn’t know.

5.1.4 Stakeholder Dependencies. The company is structured along business lines, each of which runs its projects independently. However, sometimes delivering a solution requires collaboration among teams from different business units. For example, a team that has developed a mobile application might have to interact with teams that have developed the same application for different platforms and with teams that provide server-side support for those applications. In this situation, changes in requirements in one unit lead to changes in another one.

“When we have external dependencies on the teams in, especially if they are another location, it’s sometimes quite hard to make sure that everything happens in time.”

5.1.5 Communication Issues. As the case company has a globally distributed customer base, multiple development sites and virtual SW teams, it is challenging to communicate requirements. Communication issues may begin during the elicitation and customer negotiation phases. Typically, this is due to the fact that the terminology and semantics differ between customers and developers. These differing domains bring to the table different terminology and concepts. Later in the process, SW teams may face language barriers and cultural differences that pose communication challenges. Communication issues are present, even though the company has tried several supporting tools and approaches to improve communication both within the company and with customers.

“And we often need clarification all the way from customers..., but we do have this kind of feedback cycle from us to the customers, so that we can find out what exactly is wanted. Because that’s really not that clear always.”

5.2 Requirements Volatility Challenges in Software Architecture

Interviews revealed several challenges that requirements volatility poses to SW architecture design. It should be noted that requirements volatility is not the sole reason for these challenges, but requirements volatility is the scope of this study.

5.2.1 Scheduling. Typically, requirement clarification takes so much time in the case company that architects receive requirements very late in the project. This causes challenges in scheduling both on the team level and the organisational level. On the team level, the later the architects receive the requirements, the less time they have to design architecture, and some steps must be omitted. The company creates development plans quarterly. The aim is to have two forthcoming quarters planned to provide an overall idea about what should be achieved in six months. However, quarterly plans are subject to change. Often, when a quarter begins, the schedule applies for only a couple of weeks, and then the content must be re-planned.

The most important reason for scheduling issues is requirements volatility. In the worst cases, priorities change daily, in which case, architects have no choice but to work on the item at hand and then take the next one on the backlog list, which might be different the next morning.

“Actually, we can’t use Scrum basically because our priorities change all the time, like.. maybe, sometimes daily so, yeah, we basically get the next item and work on it.”

5.2.2 Synchronisation. Stakeholder dependencies are one factor causing requirements volatility, which, in turn, causes synchronisation issues. Interviewees noted that sometimes they are unable to deliver products on time due to delays in other units. Beside inter-unit synchronisation issues, teams in the same business unit but at different development sites have synchronisation challenges caused by lack of physical proximity. Synchronisation needs and development dependencies also influence the tools used. PMT was used for project management, requirement descriptions and bug fixing. These activities require very different tool functionalities, which are supported only partially. For example, maintaining the backlog within a project and following feature development work well, but cross-project management, such as moving a feature from one project to another, is not supported.

“Probably the most complex thing is that, we need to somehow synchronize the requirements between the different teams and, that’s why, having some leadership team would be beneficial because they would synch up together what they are gonna do, what resources they have, how they would transfer their stories between the teams and et cetera.”

5.2.3 Architectural Technical Debt. The company’s requirement prioritisation criteria are strongly business driven, favouring market needs over architectural considerations. Overlooking architectural aspects when prioritising requirements accumulates architectural technical debt. As architects are overwhelmed with volatile requirements, they are not necessarily able to find out the optimal architectural design choices. Specifically, prioritising functional requirements over
NFRs is a major issue, as NFRs are the majority of ASRs, and neglecting them leads to sub-optimal architecture design.

“I think the biggest driver usually for getting something prioritized really fast is money. So if the customer is big enough and the expected income is big enough, [case company] will run through hoops... for smaller customers even if the smaller customer is asking for something that makes much more sense. So I think that, the primary driver is economic. So instead of doing feature development, we kind of get overridden by the [business customer] delivers all the time, because that’s where the money comes from.”

5.2.4 Tracing Design Rationale. The interviewees understand the importance of recording design rationale while making an architecture decision and tracing back to it during the later stage of the software development process. However, they find it difficult to maintain design documents, the most common way of recording design decisions, because they require frequent updates due to the volatile requirements. Even though PMT supports current need to some extent, there are major issues with outdated and unstructured information, broken links and inability to trace decision rationale. Typically, PMT is not used to store customer-related information, since it is considered too insecure. Thus, there must be some other means to record customer data.

“So usually the reasoning [behind design decisions], happens, it’s kind of like corridor discussion where we have a meeting where we talk about it and then we report what we decided to do in the PMT items but of course there’s lot of stuff that we miss, so we don’t really document the why, we document the what. And of course these discussions in the meeting, kind of contain the why also but if you are not in the meeting then that information is not available, usually.”

The following Figure 3 summarises the identified factors causing requirements volatility in the case company as well as the challenges it poses to SW architects during the definition of SW architecture.

![Figure 3: The factors and challenges causing requirements volatility.](image)

6 ADDRESSING REQUIREMENT VOLATILITY CHALLENGES

This section discusses the requirements volatility challenges in SW architecture design in relation to existing scientific knowledge and in the light of the empirical data gathered from the case company, thereby answering RQ3. In general, there are two main approaches to addressing requirements volatility challenges: mitigate their causes or find means to manage their consequences to SW architecture design.

Mitigating the causes of requirements volatility. Interviewees provided contradictory opinions about how much information is available to them. On one hand, it was reported that at the beginning of a project, a significant amount of time is spent negotiating technical feasibility and clarifying actual needs, the intended behaviour of the product and dependencies with other features. On the other hand, it was noted that if the available information is not too detailed it leaves room for creativity and allows discovering the best technical solution. Missing requirements early in the process causes the costliest fixes later in the development process [43]. The starting point for addressing requirement uncertainty is to evaluate what information is crucial to whom, why and when. This should be stated explicitly in the information fields provided for describing the requirement. Unnecessary default requirement fields should be removed. However, this is not sufficient, since the quality of the descriptions depends on the expertise of the writer. According to interviewees, POs or marketing personnel do not have enough technical understanding about the product to write detailed requirement descriptions. In addition, it would be a waste of time to write an extensive requirement description just to find out later that the requirement is not technically feasible. The twin peaks approach where requirements and architecture are developed in parallel could be used to address this issue [44]. The impact caused by the lack of technical understanding of the people responsible for eliciting requirements can be mitigated through supportive means. One such way is asking probe questions, to identify ASRs from software requirement specifications [5], [6].

Maintaining a strong, mutually beneficial relationship between the customer and the development team is crucial to successfully managing changing customer needs [22]. While this helps to understand the customers’ true needs and derive well-defined requirements from the beginning, it also allows developers to communicate the consequences of accommodating changes rather than blindly accepting them. Approaches to mitigate the effects of changing customer needs include eliciting gaps in requirement changes [46] and reusing existing requirements to identify the gap between elicited requirements and true user needs [2].

Following shorter development cycles can help address requirement prioritisation issues caused by a dynamic business environment [23]. However, following short design cycles can affect project scheduling and synchronisation. So it should be carefully considered.

As the SW teams in the case company sometimes depend on each other’s work, their requirements are interconnected. Therefore, it was alarming to observe that identifying the dependencies between the requirements of various stakeholders sometimes was more wishful thinking than practice. Managing
requirement dependencies is acknowledged as a challenging task [9] that may be addressed by supporting impact analysis [58]. One approach could be to establish a product management team across business units that would have overall responsibility for managing projects that span business units, including resources, scheduling, priorities and such.

Introducing advanced collaborative and communication mechanisms can overcome some of the communication issues among distributed software teams [3] as well as customer communication issues [28]. At the same time, using multiple communication media rather than a single communication channel can help avoid misunderstandings caused by cultural and language differences [50]. In the context of the case company, the personnel responsible for customer communication play a crucial role in effective communication.

Managing the consequences of requirements volatility. As previous sections have described, there are no direct causal relationships between one factor and one consequence, but relationships can be recognised. Therefore, there is no definite solution to address each identified consequence. As scheduling issues appear at the team level and at the organisational level, addressing them must be undertaken at both levels. At the team level, the situation could be improved by, for example, assessing the suitability of the elicitation techniques used and the adequacy of the requirement information collected. According to Hickey and Davis, an elicitation technique should be chosen based on problem, solution and project domain characteristics as well as known requirements [21]. On the organisational level, one solution could be to include a sufficient buffer [19] for planned releases as a response to requirements volatility. According to feedback from interviewees, interaction among team members located at various sites is not adequate, despite using various communication tools. When it comes to distributed teams, just maintaining work communication among team members is insufficient. The performance of distributed teams is affected by networking within the team and trust among members [54].

Lack of visibility among business units was mentioned constantly during interviews as hindering synchronisation among business units. While separations among business units may be necessary to organisational management, they cause several negative results in SW architecture design, the main one being the possibility of duplication of work, as the teams are not aware of each other’s work. Considering the amount of human resources and talent in the case company, there are good opportunities for knowledge-sharing among engineers. Even though the technology council and company-wide steering group can prevent the large-scale duplication of effort, work still can be duplicated on the micro-level. Closer interaction among architects in various business lines will facilitate the identification of resources suitable for a given task and, hence, get it done more efficiently. Since individual business lines evaluate their own performance, collaboration with other business lines might not be high on their agendas. However, in the long run, business lines and the company as whole can benefit from a transparent approach.

Taking the views of software architects and developers into consideration during the prioritisation process can contribute to reducing architectural technical debt considerably. Since it is the SW architects’ responsibility to recognise the ASRs that include NFRs and their effects on overall SW system architecture, architects are in the best position to identify factors causing debt and manage them to minimize accumulation of architectural technical debt [57]. In the context of iterative SW development, addressing architectural technical debt as a separate backlog item can ensure that it is addressed properly, as otherwise, the cross-cutting nature of NFRs makes it difficult to address them properly at any given point [42]. Improving traceability across the SW development process is important to understand the implications of changing requirements for SW architecture and assessing possible architectural debt [27].

SW architects should be encouraged to record design rationale by being given adequate tools [56]. In addition, a company-wide methodological approach to recording design rationale and maintaining necessary documentation should improve the quality of documentation and traceability of design rationale. Using the set of tools, including PMT, Wiki and a version control system, is a good move, as it helps maintain consistency and makes tool maintenance and support easier. However, to get the maximum benefit from the tools, it is important to educate engineers and provide necessary training. In addition, filling gaps in the existing tool chain or introducing a new tool chain that provides end-to-end tool solutions would help address identified challenges.

7 THREATS TO VALIDITY

According to Yin a construct and external validity as well as reliability are necessary conditions that have to be taken into account when conducting case studies. Internal validity has to be considered when conducting exploratory case studies. [60] Yin suggests using multiple sources of evidence, establishing the chain of events and having key informants to review case study report as tactics for ensuring construct validity in case studies. Internal validity can be addressed for example through considering a rival explanation and using logical models.

In this case study, threats to construct validity were mitigated by interview guide reviews done by professors and representatives of the case company. At the beginning of each interview, key terms related to the study were defined and discussed to ensure a common vocabulary among researchers and interviewees. In addition, a pilot interview was conducted to get feedback from an expert. The case study results were presented in the case company in a workshop, where practitioners had an opportunity to give feedback to the researchers about the study results. The case study report was delivered to the case company representatives, including interviewees, and they were asked point out any corrections needed for the report.

Threats to internal validity must be taken into account when studying causal relationships. This study aimed to explore the challenges to SW architecture posed by requirements volatility.
Since SW development is affected by several other factors, too, there is no clear causal relation between requirements volatility and SW architecture challenges. The examples of these factors are technological changes and company strategies. However, this study addressed requirements volatility only.

External validity relates to the generalisability of results. Traditionally, it has been suggested that the generalisability of results from a single case study is rather poor. [47] However, the results of case studies may be extended to other cases that have common characteristics. [47] Seddon & Scheepers, suggest that generalisation of results can be done based on a single case study as long as 1) a sample is carefully analysed, 2) relevant factors, which are true in a sample can be argued being true in larger similar context and 3) researchers seeking to generalise results discuss their findings in relation to prior studies. [49] Considering the context of the study it is expected that similar finding can be drawn when the following characteristics are present: a) globally distributed software development teams, b) a company operating in a dynamic market, c) a large company structured as autonomous units and d) serving a diverse customer base. Threats to external validity were taken into account by collecting data that can be used to characterise the subjects and case context. Examples of these data are experience of the interviewees in their field and in the company, team sizes, organisational structures and roles and the responsibilities related to them.

Threats to reliability relate mostly to means of collecting and processing data. These threats were addressed by the review of the interview guide and by agreeing upon a coding scheme prior to analysis.

8 CONCLUSIONS AND FUTURE WORK

This industrial case study was conducted to explore the challenges that requirements volatility poses to SW architecture design. Fifteen SW experts involved in SW architecture design in various business units were interviewed using a semi-structured interview as a guide.

This study revealed factors causing requirements volatility as well as the challenges posed by the requirement volatility in the case company, which provides SW solutions for companies and SW products for consumers in a global market. Through the study important challenges that requirements volatility poses to SW architecture design were identified. Finally, the means to address the identified challenges were discussed. Some factors, such as requirements uncertainty or missing information, the constant change of priorities and shifting and competing goals, are recognised as preventing architects from analysing available options and taking the optimal course of action in complex, real-world scenarios. As discussed, the context in which architecture is designed is very demanding, and architects prefer to compromise quality rather than increase overhead. However, there is a possibility that the same factors that are considered advantages also affect architecture design negatively. For example, even though light-weight documentation is considered an advantage, it can contribute to requirements volatility and increase the risk of creating architectural technical debt.

As software engineering researchers are increasingly interested in the “twin peaks” of the software development requirements and architecture design. This study provided empirical evidence about the relationship between them and how the process changes in one can affect another. The ultimate goal was to understand the complexity of the development environment and issues the practitioners face daily and thus propose feasible solutions for industry. This case study provides an example for practitioners how research may help to expose challenges, their reasons and impacts in the company. Practitioners may consult the results of the case study to identify similarities and differences in their practices. This in turn helps to find improvement directions.

In future research, another case study will be conducted in a company of different size and in a different domain to investigate whether the same challenges are present there. Cross analysis between the case studies will provide new insights and help increasing the generalisability of the findings. Based on the results of those case studies, it is planned to develop a framework that provides means for practitioners to identify the presence of challenges posed by requirement volatility and take necessary steps to mitigate the risks.

ACKNOWLEDGMENTS

This work is funded by ITEA2 and Tekes - The Finnish Funding Agency for Innovation, via the MERgE project.

REFERENCES


[10] Christel, M. G. and Kang, K. C. Issues in requirements elicitation. CMU/SEI-92-


A Systematic Map on Verifying and Validating Software Process Simulation Models

Haojie Gong, He Zhang, Dexian Yu, Bohan Liu
Software Institute, Nanjing University
Nanjing, Jiangsu, P.R. China
ghj9222@gmail.com, hezhang@nju.edu.cn, dexiayu@foxmail.com, hifibohan@gmail.com

ABSTRACT
Verification and Validation (V&V) is a critical step in software process modelling to secure the model’s quality and credibility. Software Process Simulation Models (SPSMs) that are based on descriptive process models offer the executability that is able to demonstrate the dynamic changes of software process over time. The V&V of process simulation models go beyond static process models and turn to be more complex and challenging to software modelers. This study aims to identify what aspects of process simulation models are verified and validated by using which V&V methods in what conditions in software engineering research. We conducted a systematic literature review (mapping study) on the studies of software process simulation that report of their V&V activities. We identified 72 relevant studies from a pool of 331 papers on SPSM until 2015. These studies can be mapped to ten V&V methods applied for five aspects of process models to be verified and validated, i.e., syntactic quality, semantic quality, pragmatic quality, performance, and value. A systematic map is presented to illustrate the relationships between the identified V&V methods and their supporting aspects of process models. This mapping will provide the community reference value when developing, verifying, and validating software process (simulation) models.

CCS CONCEPTS
• Computing methodologies → Model verification and validation; • Software and its engineering → Software development process management.

KEYWORDS
model verification and validation, process modeling, process simulation, systematic mapping study

ACM Reference format:
https://doi.org/10.1145/3084100.3084106

1 INTRODUCTION
Software Process Model (SPM) is a general model for software developers using any software development approach and provides a framework for measuring, analysing, and understanding the software development process [17].

Software process model can be a static model (e.g., COCOMO [6]) or a dynamic model whose behaviour changes over time. The process simulation (dynamic) models, which are based upon the static descriptive process models, are considered valuable for the management and improvement of processes in engineering [15] and enable the representation of complex dynamics of the process. Hence, Software Process Simulation Model (SPSM) offers the executability beyond the static process model.

Verification and Validation (V&V) is a critical step in any modeling project to secure the model’s quality and credibility. The model development teams are all rightly concerned with whether models and simulations are correct and reliable [24]. Model verification ensures that the computer program of the computerized model and its implementation are correct so that they fully satisfy the developer’s intent (as indicated in specifications) [35]; while model validation is defined as "substantiation that a computerized model within its domain of applicability possesses a satisfactory range of accuracy consistent with the intended application of the model" [35], which ensures the model being able to adequately support its intended use.

Since SPSM offers executability, it becomes more complex to be verified and validated than the static process model. It is often costly and time-consuming to justify that an SPSM is sufficiently valid over the wide domain of its intended applicability. Besides, there is no set of specific tests can be easily applied to determine the correctness of a SPSM. Although Lindland and other pioneers [3, 6, 19, 21] have proposed several frameworks about the V&V of software process models, their frameworks are commonly at high abstraction levels and were rarely applied in modeling practice. It is still challenging for software process modelers to systematically and comprehensively verify and validate their process (simulation) models in accordance with these frameworks.

This study aims to identify what aspects of process simulation models are verified and validated by using which V&V methods in what conditions in software engineering research. We conducted a systematic literature review (mapping study) following the guideline [16]. As a result we identified 72 relevant studies from a pool of 331 papers on SPSM until 2015. Based upon the pre-existing frameworks and the evidence collected from the reviewed studies, we provide a systematic map to illustrate the relationships between the ten identified V&V methods and five supporting aspects of process models.
The rest of this paper is organized as follows: Section 2 discusses the related V&V frameworks; Section 3 describes our research method and presents the descriptive results; the preliminary discussions to the research questions and mapping are reported in Section 4; Section 5 discusses possible threats to validity and draws the conclusions.

2 RELATED FRAMEWORKS

There exist several frameworks for the V&V of software process models. This section briefly reviews and tabulates them in terms of the aspects, goals, model properties and means (as shown in Table 1). We only listed three frameworks [3, 19, 21] (Lindland, Kitchenham and Ahmed) here because others are in the high level.

### 2.1 Richardson’s Framework on System Dynamics Modeling

Richardson and Pugh [32] proposed a framework that allocates confidence-building activities into a two-by-two matrix, having verification and validation on one axis, and structure and behaviour on the other. A verification and validation plan was developed using these sources.

**Verification** uses parameter sensitivity/variability analysis to verify model dimensional consistency; uses structural insensitivity to verify model extreme conditions in equations; uses traces to verify structural adequacy (modeler review).

**Validation** uses statistical tests to validate face validity (expert review) and does experimental case to check parameter validity.

**Evaluation** checks for unexpected behaviour to evaluate appropriateness of model characteristics for the intended audience (user review).

### 2.2 Boehm’s Criteria of Cost Model Evaluation

Boehm et al. [6] present a list of criteria (i.e., definition, fidelity, objectivity, constructiveness, detail, stability, scope, ease of use, prospectiveness, parsimony) that would be helpful in evaluating the utility of software cost model for estimation purposes. Boehm’s cost model evaluation raised macroscopic dynamic process model, the V&V of process simulation models can also be beneficial with reference to these criteria.

### 2.3 Lindland’s Framework for Quality in Conceptual Modelling

Lindland et al. [21] present a framework comprising quality goals for each linguistic concept together with methods for achieving these goals that include model properties and modelling attribute.

The only goal of **Syntactic Quality** is to assure model’s syntactic correctness, which means that all the statements in the model are syntactically correct.

There are two goals of **Semantic Quality**: validity and completeness. Lindland et al. interpret them as feasible validity and feasible completeness because they are unattainable if the goals are underproof. Feasible validity means that there may be invalid statements in the model, but it is unworthy eliminating them. In a similar way, feasible completeness means there may be statements missing from the model, but it is not worthwhile trying to find them [19].

**Pragmatic quality** touches upon the issue about how to convey a model. Lindland et al. [21] suggested that the only pragmatic goal is comprehension. They differentiated ‘comprehension’ (the model has been understood) from ‘comprehensibility’ (the model can be understood), because the pragmatic quality ultimate goal is that the model be understood rather than it is understandable; and quantifying understandability is hard to achieve.

### 2.4 Raffo’s Empirical Analysis in Process Simulation Modelling

To establish input parameters to a SPSM and compare the results of the model with actual organizational results, Raffo and Kellner [28] used actual data as input parameters to their SPSM. They addressed two kinds of challenges respectively from the input side and output side in the experimental case. The former kind of challenge includes lack of desired data, considerable variability and outliers, loosely defined metrics, small sample sizes, and so on. The latter kind of challenge includes the V&V of the model and quantitative methods of evaluating model outputs which support managers to make decision (e.g. multi-criteria utility functions, financial performance using Net Present Value (NPV) and Data Envelopment Analysis (DEA)). Raffo and Kellners paper [28] concentrates on the stochastic modelling which use Monte Carlo simulation [11]. The major points of their paper are illustrated by examples based on their practical application experiences.

### 2.5 Kitchenham’s Framework for Model Evaluation

Kitchenham et al. [19] aims to develop an extended evaluation framework and an associated evaluation process that can be used to evaluate their bidding model.

**Syntactic Quality** Lindland et al.’s [21] point of view demonstrate various different criteria, such as unambiguity, correctness, consistency, unambiguity, is actually equivalent to either validity or completeness in Kitchenham’s Framework [19].

**Semantic Quality** indicates that the model properties are syntax definition rather than syntactically correct by Kitchenham et
al. [19], because syntax is related to the modelling language. Syntax can depict the relations among language constructs without considering their meaning.

Different from Lindland’s framework [21], in which the only goal of Pragmatic Quality is comprehension because a simulation model is not useful if no-one understands it. In Kitchenham et al.’s view [19], understandability is a model property, but comprehension is an attribute of the audience groups themselves. Kitchenham et al. [19] pointed out that developers should consider two elements at the time assessing the model pragmatic quality: comprehension and understandability.

Test Quality is one more quality aspects proposed in Kitchenham et al.’s framework [19] that have been missed by Lindland et al. [21]. According to Kitchenham’s definition, its quality goal is feasible test coverage and the property to be analysed for this is executability. To assess a model’s test quality, stability, fidelity and sensitivity need to be taken into account.

Kitchenham et al. further indicated that another quality aspect missed by other researchers is the model’s Value and the quality goal is model’s utility.

2.6 Ahmed’s Framework for Simulation Model Evaluation

The result of Ahmed et al.’s research [3] is a consolidated evaluation framework for software process simulation models, the majority of which was an extension of the evaluation framework developed by Kitchenham et al. [19]. Most of the quality aspects for evaluating a bidding model were covered by Kitchenham’s framework [19]. It was noticed that Maintainability is a quality aspect but neither discussed in Kitchenham’s framework [19] nor in Lindland’s framework [21]. However, in Lindland’s framework [21] formal semantic modifiability is mentioned a property to be evaluated for model semantic quality, which is relevant to maintainability. From Ahmed et al.’s point of view [3], maintainability should be paid more attention as a quality aspect for the evaluation of a model.

3 METHOD: MAPPING STUDY

We carried out a mapping study (a specific type of systematic literature review) on the V&V of SPSM by following the SLR guideline developed by Kitchenham and Charters [16]. Three research students and their supervisor were mainly involved in this study.

3.1 Research Questions

The primary research question (RQ) of this mapping study is “How did software process modelers adopt methods for model V&V so as to secure the quality and validity of their process (simulation) models?” This question can be further elaborated into the following three research questions in order to drive this study.

RQ1. What methods were ever used in model V&V for SPSM? This question aims to summarize the V&V methods we can identify in the SPSM related studies. What are the purposes of these V&V methods? What V&V methods were applied to verify and validate the model in the studies.

RQ2. What aspects of model were verified and/or validated by these V&V methods? Although Kitchenham and other researchers have proposed in their frameworks what aspects of a process model need to be checked, to our knowledge no systematic review or mapping study was reported to investigate how those model aspects are evaluated in the process modelling studies. The answer to this question would reflect what aspects were verified and validated in process (simulation) modeling, based on the evidence extracted from the SPSM studies, and further the possible impact of those model V&V frameworks.

RQ3. How did software process modelers use these methods in their (simulation) modeling? Based on the reviewed studies, we would reach an understanding on how the model developers applied the identified V&V methods with what conditions in verifying and validating their process (simulation) models.

3.2 Search Strategy

Driven by the research questions, we adopted QGS (Quasi-Gold Standard) based search strategy [39], which systematically integrates manual search and automatic search together to capture as many relevant studies as possible.

3.2.1 Manual Search. The data source for manual search was borrowed from our previous search of SPSM studies [11, 40, 41] (as shown in Table 2), which is a combination of conference proceedings or academic journals. The manual search of this study
extended our previous literature search at those venues until the recent (the end of 2015).

3.2.2 Automatic Search. The search string used in our previous searches was re-evaluated in this study and refined with reference to the most recent studies identified in the manual search. The final search string for the automatic search of this study is as follows:

\[(\text{software process}) \text{ OR } (\text{software project}) \text{ OR } (\text{software product}) \text{ OR } (\text{software evolution}) \text{ AND } (\text{simulation} \text{ OR } \text{simulator} \text{ OR } \text{simulate} \text{ OR } (\text{dynamic model}) \text{ OR } (\text{system dynamics}))\]

The search string was coded into different syntax formats and imported into five electronic databases (IEEEExplore, ACM DL, SpringerLink, ScienceDirect and WileyInterScience) for retrieving SPSM related studies until 2015.

### Table 2: Study search strategy

<table>
<thead>
<tr>
<th>Conference</th>
<th>Journal</th>
</tr>
</thead>
<tbody>
<tr>
<td>The proceedings of ICSE (incl. related workshops, excl. ProSim)</td>
<td>IEEETransactions on software engineering (TSE)</td>
</tr>
<tr>
<td>The proceedings of PROFES conference</td>
<td>ACM Transactions on Software Engineering and Methodology (TOSEM)</td>
</tr>
<tr>
<td>The proceedings of ESEM conference</td>
<td>Journal of Systems &amp; Software (JSS)</td>
</tr>
<tr>
<td>The proceedings of SEKE conference</td>
<td>Journal of Software: Evolution and Process (JSEP)</td>
</tr>
<tr>
<td>The proceedings of ICSE/ICSSP conference</td>
<td>Journal of Information and Software Technology (IST)</td>
</tr>
<tr>
<td>The proceedings of ICSE/ICSSP conference</td>
<td>International Journal of Software Engineering and Knowledge Engineering (IJSEKE)</td>
</tr>
</tbody>
</table>

3.3 Search Process
The literature search is basically an extension and update of our previous searches [10, 40, 41] until the end of 2015 but for which explicitly report model V&V in the papers. Figure 1 shows the search process of this study that consists of three stages.

Stage I and II (the extension and update of our previous searches) finished in the beginning of this mapping study. In stage III, we reviewed the SPSM related studies from Stage I and II, and the papers were selected if they report any V&V activity during process modeling.

3.4 Study Selection
The selection criteria were identified from the existing criteria in Gao et al.’s study [11] before defining the search strategy. The inclusion and exclusion criteria are presented in Table 3 and were applied in all the search stages.

3.5 Data Extraction
Three research students and their supervisor were involved in this study. The study selection and data extraction were mainly conducted by two research students independently, and then their results were checked by the third research students. Any inconsistencies and disagreements were discussed among them until a consensus reached or escalated to their supervisor for final decision.

3.6 Results
By applying the selection criteria, we identified 72 primary studies related to model V&V from a pool of 331 papers found through Stage I and II (cf. Figure 1). Given the page limit of the paper, the final included study list is accessible at the Dropbox1.

The figure 2 shows that the majority (over 77%) of primary studies are distribute between 1999 and 2007. During the period, several frameworks for process model evaluation were proposed such as Kitchenham’s evaluation framework [19] and Ahmed’s updated edition [3]. Nevertheless, we found few studies that conducted and reported a systematic model V&V process by following the proposed frameworks in the primary studies.

When come to the venues, Software Process Improvement & Practice (SPIP2) and Journal of Systems & Software (JSS) published more papers with process model V&V than other venues. (The possible reason is that the special issues of PROSIM3 were published in these two journals).

1http://t.cn/RMCcjRb
2which was later merged into Journal of Software: Evolution and Process (JSEP)
3International Workshop on Software Process Simulation Modeling
4 DISCUSSION AND MAPPING

This section discusses the preliminary answers to the research questions based on the data extracted from 72 selected primary studies.

4.1 V&V Methods (RQ1)

The Verification and Validation (V&V) methods that were applied in the 72 studies can be identified and grouped into ten major categories, i.e., dimensional consistency test, syntax test, parameter confirmation test, face validity, extreme conditions test, comparison with other models, sensitivity analysis, questionnaire, experimental case, and comparison with actual data. The frequencies of the methods used in the reviewed studies are presented in Figure 3. It shows that experimental case is the most used method in 23% studies. In order to facilitate researchers’ reading, we listed the frequencies of the methods used from high to low, the following paragraph is distributed in accordance with this frequencies.

### Experimental case

As the most reported V&V method, experimental case was used in 22 studies to test the performance of the model’s fidelity. Beyond that, an experimental case can also be conducted to evaluate if the model provides a wide range of value (e.g., [1]).

### Sensitivity analysis

Since the input parameters of an SPSM may be associated with uncertainty (like stochastic simulation in other domains), they may have a great impact on the model sensitivity over time. This method applies uncertainty test and sensitivity test (reported in 25 studies), while the former checks model stability (e.g., [34]), and the latter examines whether the model exhibits realistic behaviours and parameters have effect on model behaviours (e.g., [38]).

### Comparison with actual data

We found 22 studies that compared the simulation results with the actual process data, such as historical data from open source projects, for validating their SPSMs (e.g., [15]).

### Parameter confirmation test

An SPSM may contain many parameters and variables that deserve to check. This method aims to evaluate the consistency between constant parameters and the knowledge in reality from conceptual and numerical aspects. It is observed in 22 studies to measure the models’ parameters against real-world attributes to ensure them correct (e.g., [20]).

### Face validity

The process model can not be verified and validated by the numeric data only, in most cases face checking (reported in 17 studies) may be necessary for the process model V&V. Interview and inspection are two common methods for face validity. Experienced developers were invited to review the process diagrams, model parameters and their relations to ensure the structure correctness and model usability (e.g., [4]).

### Extreme conditions test

The performance of an SPSM in extreme conditions can reflect whether the process model output domain is within the scope of the real domain. This method was used in 13 studies to test whether the process model behaves logically under extreme conditions (e.g., [27]).

### Dimensional consistency test

In process modeling, the consistency of the dimension and the unit has to be taken into account. This method verifies that whether the dimensions of the variables and constants used in the model are consistent and the units are correct. This method, i.e., unit check and consistency check, was used in eight studies (e.g., [27, 33]).

### Comparison with other model

SPSMs can often be developed based upon an existing process model which has been verified and validated. By comparing the simulation results of the model under development with the results of the validated model(s), this method (found in four studies) offers an effective approach to evaluate the quality of the new one (e.g., [25]).

### Syntax test

The process (simulation) models are built by modeling languages that may vary in syntax. The use of this method is explicitly found in three studies to find out syntactic errors in the behaviour-governing equations of the model. Besides manual syntax checking, some simulation tools (e.g., Vensim®) offer automation syntax-checking function (e.g., [15]).

### Questionnaire

In order to investigate the satisfaction level of the model’s performance, questionnaires were used in four studies to collect views and comments from the process modelers, contributors and target users (e.g., project manager). For example, a survey can be taken to validate whether the model can be comprehended by its target users (e.g., [12]). It can also be used to check the value of the model (e.g., [14]).

Note that some of these V&V methods may be related to each other. For example, comparison with other model and comparison with actual data are both able to check the model’s outputs, while the former is chosen when a related and validated model is available but the real process data is hard to accessible, and the latter is used to check whether the model’s outputs and the actual data are alike when real (empirical) data is available.

Both experimental case and comparison with actual data need real (empirical) process data. However, they differ in the purposes of verification and validation in modeling practice. In modeling practice, comparison with actual data is used to support model verification against its specifications when a model is incomplete and under development; whereas experimental case that also demands the real data can be used for both verification and validation of a complete SPSM by checking if the simulated model behaves consistently with the reality [9]. From the reviewed studies, we also observed that model developers specifically split an experimental case to validate different parts of the model.

---

2http://vensim.com
4.2 V&V Aspects (RQ2)

Based on the existing frameworks introduced in Section 2, we identified five aspects that were verified and validated in the reviewed studies, i.e., syntactic quality, semantic quality, pragmatic quality, performance, and value. The frequencies of the aspects that have been verified or validated in the reviewed studies are presented in Figure 4.

![Figure 4: Model to be verified and validated](image)

#### 4.2.1 Syntactic Quality

The correctness of model syntax is the only goal of the syntactic quality verification. Hurtado et al. [15] indicated that most model development tools have syntactic detection functions, e.g., the ‘Check Syntax’ function of Vensim, can be used to check whether the behaviour governing equations have syntax error. It should be noticed that syntactic verification is not explicitly reported in most studies, because model’s syntactic correctness is the basic requirement of any process model.

#### 4.2.2 Semantic Quality

Kitchenham et al. [19] indicated that two goals of the semantic quality are feasible validity and feasible completeness. The former checks whether the semantics (statements) of the process model is correct, while the latter checks any components missing in the process model.

Feasible validity denotes that all variable statements of the model are correct and consistent with the expectations [19]. In [2], Aguilar-Ruiz et al. indicated that feasible validity refers to the correctness of elements to the domain. In addition, the relationships among model elements should be also considered [15]. In other words, the goal of feasible validity is to secure the model can be represented accurately and veritably. For instance, Matalonga et al. [22] compared the empirical data they collected with their model outputs and indicated that securing the model’s consistence with reality is one of the goals of semantic quality.

In order to achieve feasible validity, it is also important to examine the structure of the model, consisting of the connections (relations) among the elements, which can be either represented with model statements or visualized in model diagrams. In [25], Pfahl et al. checked the structure of their simulation model in five views (similar to GENESISM2.0 [18]), each view contains several variable elements. The relations among these elements were checked, such as the quality view: defect injection → defect propagation → defect detection → defect correction. The correct structure with these elements allows the process model to follow the established process to implement the simulation.

Feasible completeness at a high level means that the model contains variable statements (components) about the domain that are correct, relevant to the problem and complete [21]. In process modeling, missing model parameter(s) or variable(s) will result in an incomplete model, which causes model’s outputs deviating from the expected results. Rus et al. suggest the relations in a process model must be checked for completeness and consistency [33]. All the elements of the model diagram that may influence the result have to be represented. In the reviewed studies, most process modelers recognized that the model’s completeness is the basis of model operation.

In summary, feasible validity and feasible completeness have to be checked to ensure the semantic quality of an SPSM. In Figure 4 we can notice that more than half of the reviewed studies report their semantic quality (in 38 studies). The verification of computerized model level and the validation of conceptual model level [15] are the main contents of semantic quality.

#### 4.2.3 Pragmatic Quality

The model pragmatic quality can be composed of two perspectives, i.e., the developer’s perspective and the user’s perspective.

**Understandability:** One common objective and responsibility of software process modelers is that the process model, as the outcome of modeling practice, should be understood by the stakeholders involved in the process modeling. Feasible understandability can be promoted by expressive structure of the model. In [30], Raffo et al. mentioned that the advantage of their Statistical Process Control (SPC) model is that it is easy to be understood. In addition, high understandability would encourage the involvement of more model stakeholders, which helps to improve model’s accuracy and validity.

**Comprehension:** From model user’s perspective, whether a process model is easy to use depends on its comprehension level by users. Different representations of the modeled process, e.g., visualization and explanation, can enhance the user’s comprehension. In [12], Hanakawa et al. sent questionnaires to project managers for their feedbacks as an indicator for evaluating comprehension. Most questions in their questionnaire are about the use of the model. With their regards, the comprehension of a model is an important quality aspect of process modeling.

#### 4.2.4 Performance

Performance, as an new aspect identified in the reviewed studies, was not included in the existing model V&V frameworks [19, 29]. The model’s performance can be evaluated in the test phase (model tuning and calibration) of model development process where Kitchenham and Ahmed’s framework [3, 19] suggested the test quality in this phase. In their frameworks, test quality aims to ensure the coverage of the test, which is important to the model V&V but was not mentioned in any reviewed study. To our mind, in many model V&V process, model builders would pay more attention to the quality of the model rather than the coverage of the test, which does not mean test quality is inconsequential. Alternatively, based on the reviewed studies, here we only focus on what need to be verified and validated but not the
### Table 5: Mapping between V&V methods and model aspects

<table>
<thead>
<tr>
<th>Aspect</th>
<th>V&amp;V Method</th>
<th>Condition</th>
<th>Papers &amp; Articles</th>
</tr>
</thead>
<tbody>
<tr>
<td>syntactic Quality</td>
<td>Syntax test</td>
<td>Simulation tools have &quot;Syntax check&quot;</td>
<td>[Agu02], [Hur15], [Ahm05].</td>
</tr>
<tr>
<td></td>
<td></td>
<td>There are adequate real data</td>
<td>[Kel99], [Rus14], [Sta01], [Hon15], [Cha00], [Mun03], [Leh05], [Sto06], [Leh02], [Smi06], [Lak03], [Set07a], [Pfa01b], [Don01], [Kou07], [Car02], [Syc99].</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Actual simulation process is available</td>
<td>[Lak03].</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Model may not be free-of-measurement errors</td>
<td>[Mat14].</td>
</tr>
<tr>
<td>semantic Quality</td>
<td>Comparison with actual data</td>
<td>There are adequate real data</td>
<td>[Kel99], [Rus14], [Sta01], [Hon15], [Cha00], [Mun03], [Leh05], [Sto06], [Leh02], [Smi06], [Lak03], [Set07a], [Pfa01b], [Don01], [Kou07], [Car02], [Syc99].</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Actual simulation process is available</td>
<td>[Lak03].</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Model may not be free-of-measurement errors</td>
<td>[Mat14].</td>
</tr>
<tr>
<td></td>
<td></td>
<td>There are adequate real data</td>
<td>[Kel99], [Rus14], [Sta01], [Hon15], [Cha00], [Mun03], [Leh05], [Sto06], [Leh02], [Smi06], [Lak03], [Set07a], [Pfa01b], [Don01], [Kou07], [Car02], [Syc99].</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Actual simulation process is available</td>
<td>[Lak03].</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Model may not be free-of-measurement errors</td>
<td>[Mat14].</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Not specified</td>
<td>[Rus99], [Rui02], [And02].</td>
</tr>
<tr>
<td></td>
<td>Parameter confirmation test</td>
<td>There are adequate real data</td>
<td>[Kel99], [Rus14], [Sta01], [Hon15], [Cha00], [Mun03], [Leh05], [Sto06], [Leh02], [Smi06], [Lak03], [Set07a], [Pfa01b], [Don01], [Kou07], [Car02], [Syc99].</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Actual simulation process is available</td>
<td>[Lak03].</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Model may not be free-of-measurement errors</td>
<td>[Mat14].</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Not specified</td>
<td>[Rus99], [Rui02], [And02].</td>
</tr>
<tr>
<td></td>
<td>Dimensional consistency test</td>
<td>Equations may be inconsistent</td>
<td>[Qud05], [Pfa01a], [Hur15], [Hon15], [Rus99], [Ahm05].</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Not specified</td>
<td>[Pfa00a], [Car02].</td>
</tr>
<tr>
<td></td>
<td>Sensitivity analysis</td>
<td>There are uncertain factors affect the output sensitivity</td>
<td>[Rus03], [Leh05], [Kel99], [Wern07], [Hur15].</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Not specified</td>
<td>[Pfa00a], [Car02].</td>
</tr>
<tr>
<td></td>
<td>Face validity</td>
<td>Experienced people is available in the validation process</td>
<td>[Pfa00a], [Sto06], [Raf05a], [Agu02], [Hur15], [Abd89], [Cao10], [Leh05], [Kol98a], [Kou07], [Syc99], [Men02], [Ahm05], [Car02].</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Not specified</td>
<td>[Cao10].</td>
</tr>
<tr>
<td></td>
<td>Extreme conditions test</td>
<td>Parameters have a clear range</td>
<td>[Sto01], [Non07], [Qud05].</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Not specified</td>
<td>[Cao10].</td>
</tr>
<tr>
<td>pragmatic Quality</td>
<td>Questionnaire</td>
<td>Questions with ease of use are available</td>
<td>[Han02], [Mat14].</td>
</tr>
<tr>
<td></td>
<td>Face validity</td>
<td>Questions with ease of use are available</td>
<td>[Han02], [Mat14].</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Not specified</td>
<td>[Raf03], [Zaw13].</td>
</tr>
<tr>
<td></td>
<td>Experimental case</td>
<td>Parameters have a clear range</td>
<td>[Cang04], [Ber03], [Agu02], [Pfa00c], [Qud05], [Abd89], [Smi06], [San05].</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Ensure the rationality of the model</td>
<td>[Sto01], [Hur15].</td>
</tr>
<tr>
<td></td>
<td>Extreme conditions test</td>
<td>Uncertainties may exist in the model</td>
<td>[Ber03], [Zaw13], [Cang04], [Pfa00c], [San05], [Sto06], [Wern99], [Tur06], [Leh02], [Qud05], [Rui04], [Pow07].</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Models may be sensitive</td>
<td>[Raf03], [Men02], [San05], [Raf05a], [Smi06].</td>
</tr>
<tr>
<td></td>
<td>Sensitivity analysis</td>
<td>Execution validity is difficult to assess</td>
<td>[Non07].</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Not specified</td>
<td>[Cao10].</td>
</tr>
<tr>
<td></td>
<td>Experimental case</td>
<td>There are several clear cases</td>
<td>[Pfa01a], [Raf05a], [Pfa00a], [Pad02], [Far12], [Abd89], [Raf05b], [Mar01b], [Agu02], [Ngu15].</td>
</tr>
<tr>
<td></td>
<td>Questionnaire</td>
<td>Questionnaire is available to Check the satisfaction of users and particip-</td>
<td>[Hou03], [Ahm05].</td>
</tr>
</tbody>
</table>

quality of testing a process model, and suggest the replacement of test quality with performance. From the 72 studies, the performance was evaluated from two perspectives, i.e., fidelity and robustness. Fidelity is subjective to evaluate the performance of the model and answer the question that is the model simulation close to actual behaviour. The ultimate goal of the model development is to simulate the real situation accurately, hence fidelity checking is one of the significant steps towards a valid software process model.

In [5], for example, Berling et al. conducted an extreme conditions test to validate the model fidelity. If the model is able to keep maintain its fidelity in the extreme conditions, this model can simulate the actual behaviour in the gamut.

Robustness: Of process models was firstly introduced by Sandrock et al. in [34]. They suggest that model robustness can be supported by extreme behaviour tests and sensitivity analysis. The
sensitivity analysis can be performed in two steps: firstly, the uncertain test is used to determine whether the model is stable; next, sensitivity test is used to examine the sensitivity of the model.

During modeling, we may find some uncertain factors with the process being modelled, resulting in that the model’s behaviours might deviate from the reality because of the varying of model parameters. Process modelers need to figure out which elements of the model strongly affect the model’s behaviours and the degree of the possible impact. In Kitchenham’s framework [19] and Boehm’s model evaluation criteria [6], they introduced the sensitivity and stability of software process models. As an instance, Cangussu et al. [7] indicated that the model robustness is reflected by the sensitivity and stability.

In Figure 4, we can notice that the verification and validation of process model’s semantic quality and performance accounted for 88% of the total V&V practices.

4.2.5 Value. The value of a process model lies in its practical utility, which means the purpose of the model value validation is to make the model can be widely used in practical utility [19]. The value can be validated by multiple experimental cases. Pfah et al. [25] conducted a number of cases to check whether their SPSM can simulate the real software process in a variety of scenarios.

Although maintainability was proposed in Ahmed’s framework [3] for SPSMs, we failed to find any study reporting the validation of maintainability in our review. Most of the reviewed studies do not discuss the validation of their models after deploying them.

4.3 Implementing V&V Methods (RQ3)

This subsection presents how the software process modelers use the identified V&V methods to verify and validate the aspects of the models. We establish a mapping between the V&V methods and the model aspects as well as with the application conditions (shown in Table 5). It illustrates what method can be used to verify and validate which aspects of process model under specific conditions based on the evidence. Some articles did not unambiguously provide the conditions when to use these methods. So we take notes about these articles and use “Not specified” to mark this situation.

Syntax test. As shown in Table 5, syntax test is the means to verify the syntactic quality of process models. Aguilar-Ruiz et al. [2] indicated that software process modelers can use syntax test when they need to check the syntactic correctness of the process model. The syntactic quality can be verified manually or automated. When the simulation tool has “Syntax Check” function (such as in the System Dynamics simulation package—Vensim), developers can use the feature to perform the syntax test. Beside automation checking, manual checking can also be used to verify the syntactic issues of process models as one of the most primitive verification methods.

Dimensional consistency test. This test focuses on the semantic quality of process models and verifies the dimensions of the variables and constants. Hurtado et al. [15] indicated that dimensional consistency should be checked to secure the variables in the equations correct in the input/output process (such as process “variable-rate-variable”). In dynamic process models, the dimension of input ($R_i$) and output ($R_o$) rates of the variable ($\int_0^T (R_i - R_o) \, dt$) should be consistent. Model developers may use this method on the condition that they need to check the consistency of dimensions and units or check the correctness of model’s equations.

Parameter confirmation test. An SPSM may contain a lot of parameters. It is necessary to ensure that the input parameters of the process model are realistic. Parameter confirmation test can be used to verify the semantic quality of SPSMs. If modelers expect to check the parameters’ correctness or the influence of parameters, parameter confirmation test would be useful in process model development. For example, Qudrat-Ullah et al. [27] estimated the effectiveness of the input parameters to secure their process model’s input parameters can correctly reflect the real situations.

Face validity. Face validity is a flexible but not easy model V&V method. Semantic quality and pragmatic quality can be verified and validated by using this approach, which is able to check the structure and completeness of the process model, but model developers need to be familiar with the structure of the process model, or the effectiveness of face validity would be reduced [2]. There are many parameters and (auxiliary) variables in an SPSM. The order between the variables, whether the relations among the process model’s elements are appropriate, and whether these relations are consistent with the actual situations, have all to be checked by face validity. This was confirmed by the software project managers involved in Abdel-Hamid and Madnick’s study [1].

There are several specific face validity methods such as interview, invitation, etc. Hurtado et al. [15] recommended adopt interview for face validity. They interviewed several international usability experts in their field. Each expert checks the usability of the model and scores them, including the structure of the process simulation model and others.

Extreme conditions test. The test can be used to verify the semantic quality and validate the model performance, which means this method can be applied to check whether the model is valid when developers need to guarantee model’s reasonable behaviours. This method enables developers to have confidence in the process model when the performance of the model falling within reasonable ranges.

Nonaka et al. [23] were confident with that their process simulation model is reasonably implemented. They investigated the simulation results by using the histogram which can show the results of the model. Firstly, they identified two important parameters in their process model, ‘the strength of the dependency’ (DEP$\in (0.2, 1.0)$) and ‘the number of residual defects in core assets’ (NRD$\in (10, 40)$), after that they added the extreme values to the process model. As time (t) goes on, four kinds of behaviours ($2^4$) are output under extreme conditions.

As another example, Smith et al. [36] checked the model behaviours under the extreme conditions. They exhaustively inspected the model’s parameter space by determining the range of possible values for nine controlling parameters in the process model, and each parameter has maximum and minimum values (2 values). They input these extreme values to their process model, which resulted in 256 ($2^8$) distinct combinations of parameter values. They logged these results following each time points, and then examined
whether these model outputs (against the extreme input values) in
a reasonable range of the real system.

Comparison with other model. In the reviewed studies, many model-
ers developed their process models based on other pre-existing
models. When real (historical) data could not be obtained, process
modelers compare theirs with other models to validate the seman-
tic quality. In general, comparison with other model is used to
check whether the parameters, outputs, and structure of model are
similar or comparable with other models. The COCOMO model is
a static software process model that provides constraints on the re-
results of the model. In [25] Pfahl et al. based their dynamic process
model on the COCOMO model. The constraints of the COCOMO
model can limit the results of their dynamic process model, accord-
ingly the accuracy of their model was identified through the com-
parison of the simulation results with the estimates produced by
the COCOMO model.

Sensitivity analysis. When the process model contains a lot of
uncertain factors or random parameters. It is necessary for mod-
ers to perform sensitivity analysis to test the model’s sensitivity
which is related to semantic quality and performance.

Sensitivity analysis are performed with a full factorial design
for Berling et al.’s model [5]. First, they determined which param-
eters may contain uncertainties and analysed the possible devia-
tions. After that, they identified five parameters that need to be
detected; the possible deviation was divided into two levels; and
the analysis index named ‘A’. According to full factorial method
definition there dynamic process model should be resulted in 32
runs ($2^5$). Then they defined the degree of uncertainty indicators,
usually $\Delta F = \pm 10\%$. Ultimately the sensitivity coefficient is $E = \Delta A$.

Nonaka et al. [23] suggested that this approach is not applicable
in the case that the process model have a number of parameters,
however these parameters are deterministic but not stochastic.

Questionnaire. Questionnaire is a direct and effective method to
examine the pragmatic quality and value of process models for
users. Typically, this method was feasible for model validation if
the model development team is able to design a questionnaire and
expects to check the satisfaction of users.

Hanakawa et al. [12] sent questionnaires to project managers for
the feedback. This allows the process modeling team to be aware of
what needs to be modified. In addition, questionnaire can be used to
investigate user’s satisfaction and whether the process simulation
model can realize its value in real process management. As
reported by Houston et al. in [14], a questionnaire can be sent to
the managers who can make recommendations and evaluation of
the model behaviours in the development process. The evaluation
is conducted for checking whether the process model can simulate
the real behaviour.

Experimental case. In the reviewed studies, we found that exper-
imental case was widely used to test the performance and value of
process model. This method can be used to validate the process
model by using the data from a specific case (instance) of the mod-
elled process. In Cao’s and other pioneers’ research [8, 26, 31, 37],
they set their models with the data from one or more special cases.
The simulated results by executing the process models were com-
pared against the data from the real cases to test whether the pro-
cess model results were consistent with the real case. The purpose
of the experimental case is to test the process model performance
in a special case(s) with real data.

Comparison with actual data. This method is used to empirically
validate the semantic quality of SPSMs when real process data is
available. Stallinger et al. [37] applied empirical data from four real
projects to test whether the behaviours of their process model fol-
lowing the time (t) can reflect the reality.

5 CONCLUSIONS

This paper reports a systematic mapping study on Verification and
Validation (V&V) of software process simulation models. To reduce
the reviewers’ bias, each paper was reviewed by two researchers
independently and we held regular meetings to discuss any dis-
agreements until reach a consensus.

One possible threat to validity is that the definitions of the V&V
method vary among studies, which might introduce bias when distin-
guishing and classifying the V&V methods since definitions in
some studies are vague. Moreover, several studies did not provide
the details about their V&V processes, which disables us to identify
the actual V&V methods used in them. Another potential threat is
that some studies only provided suggestions about which model
aspects need to be validated and which method can be used, but
without any empirical data to support their suggestions. Hence,
the qualities of these studies as evidence might be uneven.

Based on the reviewed results, we have identified what model
V&V methods can be used to verify and validate software process
simulation model, what aspects need to be verified and validated,
and how these V&V methods were used in modeling practice. As
the outcome of this research, a systematic map is presented to il-
lustrate the relationships between the identified V&V methods, the
supporting aspects to be verified and validated. Although with ref-
ence to Kitchenham’s framework [19], our mapping is built upon
software process modelers’ practices reported in the 72 studies on
SPSM, and first establish the solid links from the model aspects
for V&V at the high abstraction level to the specific V&V methods
at the implementation level. Moreover, the model’s performance is
an extended replacement of the text quality in Kitchenham’s frame-
work [19].

Nevertheless, it does not necessarily imply that this mapping
applies to every simulation model V&V situation, such as a special
situation as an exception from the reviewed studies. Thereafter,
the future research work can be carried out in two directions: (1)
to improve the existing frameworks or propose an updated version
of the framework for the V&V of SPSMs; and (2) to develop guidelines
of performing the model V&V in process simulation studies and
extend the instructive value of the model V&V frameworks.

ACKNOWLEDGMENTS

This work is supported by the National Natural Science Foundation
of China (Grant No.61572251).
Rapid, Evolutionary, Reliable, Scalable System and Software Development: The Resilient Agile Process

Doug Rosenberg
ICONIX Software Engineering and USC
United States
resilientagile@iconixsw.com

Bo Wang
University of Southern California
United States
wang756@usc.edu

Barry W. Boehm
University of Southern California
United States
boehm@usc.edu

Kan Qi
University of Southern California
United States
kqi@usc.edu

ABSTRACT
The increasing pace of change in competition, technology, and complexity of software-intensive systems has increased the demand for rapid, reliable, scalable, and evolvable processes. Agile methods have made significant contributions to speeding up software development, but often encounter problems with reliability, scalability, and evolvability. Over the past 3 years, we have been experimenting with an approach called Resilient Agile (RA), which addresses these problems while also speeding up development by finding enablers for parallel systems engineering, development, and test. This paper summarizes our experience in defining and evolving RA by applying it to three representative emergent-technology applications: Location-Based Advertising, Picture Sharing, and Bad Driver Reporting. In comparison with the mainstream Architected Agile process that we had been using on similar systems, the RA process achieved fewer defects and significant speedups in system development and evolution.

The paper summarizes the overall challenge of software schedule compression; identifies managed parallel development as generally the most powerful and least-practiced strategy for schedule compression; summarizes the key elements required to support parallelism, including specific model-driven system development techniques, automatic generation of key elements and realistic schedule and effort estimation. It then summarizes the three successful Resilient Agile projects to date, provides criteria for selecting a Resilient Agile process, and summarizes the key techniques for scaling up Resilient Agile, using a previous million-line command and control project as an example.

CSCS CONCEPTS
• Software and its engineering → Agile software development; Rapid application development; Software architectures;

KEYWORDS
agile development, schedule compression, rapid delivery, NoSQL, REST, code generation, MVC, parallel development, scalable software development, ICSM, microservice architecture, resilient software, UML modeling, use case driven development

1 INTRODUCTION
Our continuing interactions with our government and industry affiliates have indicated that earlier forms of agile methods can be rapid, but often achieve speed by addressing only the sunny-day use cases, causing serious reliability problems. Some progress in improving agile development reliability has resulted from the Architected Agile[1] process, but often the build-to architecture remains fixed and non-responsive as the marketplace and technology evolve. Approaches such as Scrum and Kanban address such problems, but tend to focus on project management and less on design for parallel development.

For the last 3 years, we have been experimenting with the Resilient Agile approach. It involves a highly skilled Systems Engineering team exploring options with clients via storyboards, workflows, and prototypes to define both sunny-day and rainy-day use cases; defining test requirements and plans for each usage scenario; decomposing each case into a conceptual Model-View-Controller (MVC) pattern, and continuing to evolve the architecture as client needs and opportunities evolve.

Resilient Agile achieves significant schedule compression by leveraging parallelism; as many developers as necessary can independently and concurrently develop the scenarios from initial concept through code. Further, a system test team can use the MVC decompositions to quickly generate a test suite to support continuous testing of the evolving system. We have used this approach to develop a working Location-Based Advertising system (LBA).
involving more than 70 developers overall and 29 parallel scenario developers, a working Picture Sharing system (PicShare) involving 10 scenario developers, and a proof-of-concept Bad Driver Reporting system (BDR) involving 15 scenario developers to-date (ongoing project). In each case, the scenario developer effort has been roughly 80 hours, or a 2-week time commitment.

Resilient Agile is fully compatible with the Incremental Commitment Spiral Model (ICSM) [2], which involves a 3 team concurrent effort of a Systems Engineering team, a Development team, and a dedicated System Test team. The stabilization afforded by Resilient Agile’s “conceptual MVC” scenario decomposition enables the Development team to operate in a highly parallel mode, enabling more rapid development and evolution while benefiting from the ICSM’s scalability to larger projects.

1.1 The Challenge: Compressing Schedule by Adding Developers

From the beginning of software-time, people have wondered why it isn’t possible to accelerate software projects by simply adding staff. This is sometimes known as the “why can’t we make a baby with nine women in one month?” problem. The most famous treatise declaring this to be impossible is Fred Brooks’ book in 1975, “The Mythical Man-Month”, in which he declares that “adding more programmers to a late software project makes it later” and indeed this has proven largely true over the decades. However, since 1975 a number of strategies for accelerating systems and software development schedules have been found. A DoD Systems Engineering Research Center study[3] categorized these into product, process, project, and people strategies, which led to the development of a quantitative model for estimating the effects of the strategies on a project’s schedule[4]. This study identified a number of good sources [5–13] for such strategies. One particular strategy for schedule acceleration with high payoff potential but relatively little application was to organize projects in ways that enabled developers to work in parallel, within a framework where the components were ensured to successfully interoperate. The following sections describe our approach to successfully execute this strategy on several relatively different applications.

1.2 Resilient Agile VS. Other Agile Methods

Resilient Agile has some commonalities and some differences with other agile approaches. Like most agile methods, RA “gets to code early” and uses feedback from executable software to drive requirements and design. Specifically, RA uses technical prototyping as a risk-mitigation strategy, for user interface refinement, to help “sanity-check” requirements for feasibility, and to evaluate different technical architectures and technologies. In our student projects, we have used technical prototyping to help develop background-capable, power-safe geofencing, to compare multiple voice-recognition toolkits, and to evaluate different toolkits for developing augmented reality applications.

Unlike many agile methods, RA does not support “design by refactoring,” nor does it drive designs from unit tests. Instead, RA uses a minimalist UML-based design approach (Agile/ICONIX) [14] that starts out with a domain model to facilitate communication across the development team, and partitions the system along use case boundaries, which enables parallel development. Within each use case, RA follows a standard set of steps including a complete sunny-day/rainy-day description for all use cases (confronting rainy-day scenarios early adds resilience to software designs). Each use case is then “disambiguated” using a conceptual Model-View-Controller description, and then carefully designed. Typically, designs are shown on sequence diagrams; however, TDD may be used as an alternate detailed design process. Requirements are modeled and allocated to use cases, and traceability between requirements and design is verified during design reviews, further increasing resilience of the software.

Having a small, standardized set of design steps and artifacts facilitates communication among team members who are working in parallel and also makes developers more interchangeable. One of the characteristics of student projects that run across multiple semesters is nearly 100% staff turnover every 3 months since students typically don’t take the same class in multiple semesters. This turnover rate has not caused a problem on our student projects.

Having a UML model allows RA to leverage automation to further accelerate development. Domain models are made executable by code generation of database collections, CRUD functions and RESTful APIs. RA drives test plans from the design artifacts (Design Driven Testing) [15] and puts greater emphasis on acceptance testing as opposed to unit testing.

Resilient Agile does not specify a management process for delivering code but focuses strictly on the engineering design of the software. RA is flexible with respect to management process. It’s possible to use RA with two-week Scrum sprints, but this process is not mandatory. If the sprint/backlog paradigm is desired, it’s straightforward to populate the backlog with controllers from the MVC use case decompositions. On most of our student projects, we have worked with semester-long “sprints” but with developers working in parallel (one per use case) and each developer delivering either design artifacts or code on a weekly basis.

Finally, RA attempts to eliminate ceremony from the development process. Both agile and waterfall processes include significant amounts of ceremony (e.g. daily stand-up meetings) and ceremony costs time and money. For example, requiring all team members to participate in an hour-long standup meeting every day imposes a 12.5% tax (assuming an 8-hour workday) across the team.

However, working in parallel does require some form of organized communication between team members, as interfaces need to be negotiated. We have found that one 90-minute meeting per week (class time) is beneficial in this regard.

2 LEVERAGING PARALLEL DEVELOPMENT - THE HIGH-PAYOFF KEY TO SCHEDULE COMPRESSION

Our experiments at USC CSSE with Resilient Agile began with an effort that we initially called “Massively Parallel Use Case Modeling”,
where we partitioned a Location Based Advertising system into a large number of use cases and assigned one use case to each of 47 students in Prof. Boehm’s CS577 Software Engineering class. Two students from that class subsequently were admitted to the PhD program at USC and are co-authors of this paper. The lead author was the project owner, architect, instigator, troublemaker, “Keeper of The Holy Vision” (KOTHV), and also the author of Agile Development with ICONIX Process[14].

The project originated as two homework assignments to give students a hands-on learning exercise with UML and use case driven development, with the first homework assignment called “build the right system” and the second assignment called “build the system right”. Homework 1 (Build The Right System) consisted of each student identifying requirements, storyboarding screens, writing use case narratives, and disambiguating their use cases via “robustness diagrams” (conceptual model-view-controller decomposition of a use case), while Homework 2 (Build The System Right) consisted of detailed design using class diagrams, sequence diagrams, database schemas, and occasionally state machines. While grading Homework 2, the chief troublemaker got the bright idea of offering an extra credit assignment for students to write prototype code for their use case, and 29 out of 47 students accepted the challenge.

Our original expectations for this exercise were that we would wind up with a fairly detailed UML model (which we did) and not much in the way of working code. The expectation of a decent UML model came from a couple of decades of ICONIX “JumpStart” training workshops where it is standard practice to work a real industry project with multiple lab teams where each team worked on a different subsystem of the project. In those classes we typically limit to 3 lab teams per instructor, so whether this could be stretched to 47 parallel threads of development was an unknown quantity.

The expectation of “not much in the way of working code” was based on the presumption that it was a complete impossibility to successfully integrate the work of 47 developers working in parallel. This expectation was exceeded for reasons we explain below, and we presented the results at the USC Annual Research Review under the title of “Massively Parallel Use Case Modeling”. The results of CS577 were interesting enough that we decided to continue the project using Directed Research students as scenario developers, and work continued over a summer session, and the following fall and spring semesters, once again with students working in parallel. At the end of the exercise, we had evaluated multiple technical approaches and refined the prototype code into a fully functional system. We then continued the experiment with two other student projects and began gathering productivity data on the various projects. This experimentation and data gathering is presently continuing with additional student projects.

3 KEY ELEMENTS REQUIRED TO SUPPORT PARALLELISM

3.1 Just Enough Planning

Software processes can be rated on a formality scale ranging from "feedback driven" (less formal) to "plan-driven" (more formal). In Balancing Agility and Discipline[16] the case is made that some parts of the system may be feedback-driven (e.g., the user interface) and that some parts of the system may be plan-driven (e.g., the security kernel), but that for most complete systems the extremes on either side of this scale are expensive, and that there is a cost-minimum somewhere in the middle. In Agile Development with ICONIX Process[14], the case is made that “just enough planning” in the form of a minimalist, use case driven approach gets close to this cost minimum, and in fact, Resilient Agile has its roots in Agile/ICONIX. For CS 577 students we simplified the approach to Build the Right System (i.e. get the requirements right) and Build the System Right (i.e. careful design).

3.2 Anticipation of Dependencies

When planning for parallel development, it’s critically important to understand where dependencies between use cases exist, partition the project into appropriate subsystems, and ensure adequate collaboration and communication between developers whose work is interdependent. On the location based advertising (geofenced coupon delivery) project, we developed a cross-platform mobile app, a customer facing website, and a business-partner facing website which included coupon publishing and billing. A well-formed UML model with careful and continuous architectural review of independently developed designs was critical to a successful outcome.

3.3 Rapid and Flexible Database and API Development

One of the big reasons that we had low expectations for successful integration of independently developed student code was that it seemed to present an intractable database problem. With independent development, we had no right to expect that database schemas would be compatible. In this aspect we were aided by our choice of a NoSQL database rather than a relational database. NoSQL databases are very flexible by virtue of each record carrying its own JSON schema, and thus well-suited for "sandbox" development. Parallel development can then proceed in the sandbox, and when it reaches
maturity, the database schema can be unified and promoted out of
the sandbox into a production instance of the database.

On the location based advertising project, the initial CS577 stu-
dent team that developed the billing system had requested to work
in PHP/MySQL because those students had more experience with
that technology. Our success in other parts of the system using
NoSQL was good enough that in the following semester we re-did
this part of the system using Cassandra and JavaScript (jQuery
Mobile) so that we had a unified code baseline. During this exercise
we developed a design pattern where students could rapidly ‘clone’
some sample database code that implemented the database “CRUD
functions” (Create, Read, Update, Delete), and this proved successful
enough that it became the seed for a student-developed database
code generator that we describe below. We have piloted this data-
base code generator on a subsequent RA project with excellent
success.

3.4 Use of RESTful APIs (Microservice
Architecture)

NoSQL databases did not exist when Brooks wrote Mythical Man
Month. Neither did RESTful APIs and microservice architectures.
These technologies are both helpful in integrating work performed
by independent developers and development teams. In particular
the ability to rapidly publish a RESTful API that makes it possible to
create and read collections from a NoSQL database from both mobile
apps and web apps allows for a great flexibility in development.

3.5 Prototyping Areas of Technical Risk

One of the luxuries of massively parallel development is that it’s pos-
sible to have developers explore alternate technologies and mitigate
risk early on, because you have lots of programmers available. This
prototyping occurs in parallel, so you’re not burning up months
on the calendar to evaluate alternatives. On the location based
advertising project we used this to our advantage by prototyping
systems that used two different commercial geofencing products,
in parallel with developing two homegrown geofencing plugins
(one for Android and one for iOS). In the end our homegrown so-
lution outperformed both commercial solutions. On a subsequent
RA project (Bad Driver Reporting) multiple students prototyped
voice activation frameworks until we found one that worked, and
prototyped dual map/video display code for both iOS and Android.
Meanwhile other parts of the project continued without waiting for
these prototyping threads to complete.

3.6 Better Communication through Visual
Models

One of the major factors behind Brooks’ Law was that bringing
new programmers onto a project placed an additional communica-
tion burden on the more experienced developers, so that schedule
sometimes slipped because those who knew what was going on
spent too much time training the newcomers.

Visual modeling was another technology that didn’t exist in
the days of OS 360. All of our student projects experienced nearly
100% personnel turnover every 3 months (occasionally we get lucky
with a few returning students but for the most part it’s a new class
coming in every semester). We had great success handing the UML
When you have a large pool of developers available it’s not too involved in the proposed system. It also shows the attributes and variety of model artifacts such as use cases. Having the system consists of two steps: in parallel with a live database immediately. The general process happens very early in the project, enabling developers to work input for generating the database backend and RESTful API. This all developers who need that object.

It’s usually the case that different use cases will need different attributes. Each developer can start building immediately with the attributes they need, and worry about integration later. All that’s required is that the correct name of the domain object is used by all developers who need that object.

In Resilient Agile, the conceptual domain model is processed as input for generating the database backend and RESTful API. This happens very early in the project, enabling developers to work in parallel with a live database immediately. The general process consists of two steps:

4.1.1 Extraction. A domain model contains all the entities involved in the proposed system. It also shows the attributes and behaviors for each entity. The code generation process will first extract this information from the domain model by analyzing its profile.

4.1.2 Generation. The code generator then processes and organizes the information. On the client side, it generates corresponding models and wraps basic functions in an adapter for utilizing. On the server side, it provides DB schema and basic CRUD functions for data queries, and basic RESTful APIs for resource accessing.

4.2 Instant Microservice Architecture

Microservice architecture has become increasingly popular in today’s software development. Services can be accessed via URL using RESTful APIs. Using microservice architecture, conceptual MVC decompositions of use cases can be quickly realized.

Our goal is to accelerate project progress by rapid integration and release of what has been designed. At the very beginning, a draft design of the domain model can be directly translated into source code. The source code provides the whole backend, which covers both server side and client side, for the developers to access. At any point in time, when the domain model evolves, the entire backend can be regenerated immediately. In this way, database code generation enables true evolutionary development (i.e. it’s true to the goals of agility, whether or not we use Scrum and TDD).

4.3 Databases Targeted

It’s clear that some projects will find a NoSQL model most effective, but others will find a SQL model to be a better fit to their usage patterns. Some might need both for different parts of their business. Moreover, there are many flavors of database whatever model is chosen. For the above reasons, in order to be most effective across a wide range of projects, the code generation should be re-targetable towards a variety of databases. The decision varies from project to project.

To help make the best choice, several criteria should be taken into consideration:

4.3.1 Scalability. If the system tends to contain big data, or to be massive with a large number of users, the NoSQL model should be considered.

4.3.2 Flexibility. If the data schema change frequently in order to better fit the requirements, or tends to be complex, the NoSQL model should be first considered.
4.3.3 **Functionality.** If the database is expected to serve complex queries, the SQL model should be chosen.

4.3.4 **Relation Complexity.** If there are complex relationships existing between different sets of data, SQL model’s join capability would be the better choice.

When attempting schedule compression via massively parallel development, flexibility is key. The data schema changes frequently because individual developers are working in “sandbox mode” and may require different attributes on the database collections. Of course, attributes must be merged when moving out of the sandbox into a production system. Because flexibility is so important, current research has focused on NoSQL databases, with all scripts and generated code written in JavaScript. Specifically the initial implementation of the code generator generates code compatible with Mongo DB and Node JS (i.e. MEAN stack). Future versions of the code generator will target additional databases.

### 4.4 Adapting to Changing Requirements via Database and API Regeneration

Responding to rapidly changing requirements is one of the key reasons that organizations choose to adopt agile processes. In a more formal agile process we first refactor the design based on new requirements, then implementation follows. Large amounts of time may be spent refactoring. In less formal agile processes, code is modified without documentation, which makes the code mismatch the design. The mismatch is untraceable and will lead to a series of other problems.

With RA we can instantly regenerate the backend (both database and REST API) from the domain model, giving us a way to incrementally and quickly prototype the design. Design changes to accommodate shifting requirements are reflected in the design. In this way, executable domain models enable evolutionary development within the context of a model-driven project.

### 5 SUMMARY OF RESILIENT AGILE STUDENT PROJECTS TO DATE

Our experimentation with RA at the USC Center for Software Engineering started with a Location Based Advertising (LBA) project. This project consisted of an initial team of 47 students in a Masters level Software Engineering class as two homework assignments and was continued over 3 semesters using different teams of Directed Research (DR) students. DR students are given a time budget of 5 hours per week per unit, and most students take only 1 unit per semester. Approximately 75 students overall worked on the LBA project, with a 90% staff turnover every 3 months. During this time we did extensive prototyping of different geofencing solutions, and introduced a test-team operating concurrently with the developers to evaluate the performance of the various prototypes and conduct in-the-field acceptance testing.

Our next attempt was with a photo-sharing app called PicShare, where we originally wanted to implement the same project with two independent teams, with one using Architected Agile and one using RA. The productivity comparison effort was hampered by the RA team initially getting some incorrect guidance on how to do MVC decomposition which required extensive re-work by the 10 student RA team. However, once the team got straightened out they produced excellent results.

The third experiment was with a Crowdsourced Bad Driver Reporting System (BDR) which involved 15 students working in parallel for approximately 12 weeks at 5 hours per week per student. This was the first project where we deployed the database code generator. The BDR team developed a proof-of-concept system consisting of voice-activated “dashboard camera” mobile apps for iOS (Swift) and Android (Java), and a web app implemented in Angular JS for filing, reviewing and querying video-centric bad driver reports. This project is about to be continued by a new student team who will move it from proof-of-concept into a more robust system.

A new augmented-reality game project will also be attempted in the upcoming semester.

### 6 RESILIENT AGILE AND SCRUM/TDD

RA is a flexible process in that development can proceed using the now ubiquitous Scrum-sprint-backlog paradigm or each developer can design and implement the use case they have specified. The authors are fully aware that many people will not regard any process as “agile” unless it has Scrum and TDD in it.

With the Scrum-sprint approach, modeling is focused on requirements and conceptual design, then controllers from the MVC decompositions are added to the backlog, and TDD can be used as an alternative design approach to UML sequence and class diagrams. Once the use cases have been elaborated through conceptual design which includes both sunny-day and rainy-day behavior descriptions, the detailed design process can be left to the discretion of the Scrum teams with little risk of scope creep. Our work on student projects to date has not used the Scrum/TDD development strategy, as we have focused on compressing schedules using parallelism. This remains an area for future exploration and data collection.
while being calibrated to the actual effort on the student projects. An important to avoid understaffing or overstaffing the number to apply RA in a corporate rather than a student environment, it important to be able to generate these first-order estimates quickly. requirements are very fluid at this early stage of the process it's putting cyclomatic complexity of the activity diagram. Since system systematically by elaborating the use case on an activity diagram) and computing cyclomatic complexity of the activity diagram. Since system |

7 ESTIMATION OF RESILIENT AGILE

To apply RA in a corporate rather than a student environment, it will be important to avoid understaffing or overstaffing the number of developers needed. This made it important to come up with an effort estimation model that could be used for corporate RA projects, while being calibrated to the actual effort on the student projects. An initial cost model for such corporate RA projects has been developed and calibrated, with reasonable accuracy. It is actually a series of three models, based on the information that will be available at the three decision points: Early Use Cases, Extended Use Cases, and Function Point Sizing. Tables 1, 2 and 3 provide a summary of the inputs, estimates, and actuals for the 3 projects. The inputs include not just sizing parameters, but also first-order technical complexity factors and environmental complexity factors, roughly comparable to the function point General System Characteristics. Future extensions will provide more detailed rating scales for these, and comparisons with estimates based on other models[17–25].

7.1 Early Use Case Points (EUCP) enable Predictive Complexity Estimation

Because RA projects follow a well-defined sequence of modeling steps before coding begins, and because these models are captured electronically, we can mine the models using scripts for information that is useful in predicting size and cost drivers before coding. Ideally we would like to be able to get rough estimates during the "build the right system" (requirements exploration) phase, and then refine our estimates during the "build the system right" (design) phase. Because RA is use case driven at its core, we have started from previous work on Use Case Points and extended the concept to include a first estimate deriving Early Use Case Points (EUCP) from cyclomatic complexity of use case paths, and then a more detailed estimate (driven from MVC decompositions) based on Transactional Complexity within the use cases. Finally during detailed design we can derive more accurate estimation by counting model elements such as the number of messages on a sequence diagram.

Early Use Case Points (EUCP) can be derived by analyzing the number of sunny-day and rainy-day paths through a use case (typically by elaborating the use case on an activity diagram) and computing cyclomatic complexity of the activity diagram. Since system requirements are very fluid at this early stage of the process it’s important to be able to generate these first-order estimates quickly.

<table>
<thead>
<tr>
<th>Table 1: Effort estimation by Early Use Case Point (EUCP)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Project</td>
</tr>
<tr>
<td>----------</td>
</tr>
<tr>
<td>BDR</td>
</tr>
<tr>
<td>PicShare</td>
</tr>
<tr>
<td>LBA</td>
</tr>
</tbody>
</table>

**Table 2: Effort estimation by Extended Use Case Point (EX-UCEP)**

<table>
<thead>
<tr>
<th>Project</th>
<th>UXEXUCP(^6)</th>
<th>UAW TCF ECF</th>
<th>EXUCP(^7)</th>
<th>Estimate (Man-hour)</th>
<th>Actual (Man-hour)</th>
</tr>
</thead>
<tbody>
<tr>
<td>BDR</td>
<td>62</td>
<td>14</td>
<td>1.14 1.21</td>
<td>104.83</td>
<td>1523.78</td>
</tr>
<tr>
<td>PicShare</td>
<td>132</td>
<td>8</td>
<td>1.06 1.03</td>
<td>152.85</td>
<td>1858.96</td>
</tr>
<tr>
<td>LBA</td>
<td>270</td>
<td>12</td>
<td>1.12 1.32</td>
<td>416.91</td>
<td>3702.10</td>
</tr>
</tbody>
</table>

Fortunately a substantial amount of automation exists for generating activity diagrams from use case descriptions in the Enterprise Architect CASE tool, using automatic activity diagram generation from “structured scenarios”. This approach is detailed in Design Driven Testing[16]. The estimation process follows the steps below:

- Convert Use Case narratives to structured scenarios.
- Convert structured scenarios to activity diagrams.
- Count the cyclomatic complexity of each activity diagram.
- Based on the cyclomatic complexity of each activity diagram, count the Unadjusted Early Use Case Weight (UEUCW) for each identified use case.

7.2 Extended Use Case Points (EXUCP) measure Transactional Complexity within Use Cases

In RA, use cases are decomposed as conceptual-MVC (robustness) diagrams which break each use case into Model (entity), View (boundary), and Controller (control) elements. Controllers on the robustness diagram represent transactions within a use case. This process can be automated using scripts on the UML models.

We can compute transaction weight using the following steps:

- Identify the transactions by counting controllers on robustness diagrams.
- Count the number of UI elements (boundary elements) involved in each identified transaction.
- Count the number of domain objects (entity objects) involved in each identified transaction.
- Evaluate the complexity level for each transaction.

Using these methods we can automatically generate a use case complexity estimate. Note that this method is predicated on users producing well-formed robustness diagrams. This presented a practical difficulty on our second RA project (PicShare) when the team received some incorrect mentoring guidance instructing them to show system behavior on associations rather than using controllers. In general the conceptual MVC technique for elaborating use cases is not as well understood as techniques such as activity and sequence diagrams. An additional cost driver to account for the use of the automatic code generator use on the BDR project reduced its effective size, and explained how the 15 BDR developers could use less effort than the 10 PicShare developers.

---

\(^1\)UEUCW: Unadjusted Early Use Case Weight
\(^2\)UAW: Unadjusted Actor Weight
\(^3\)TCF: Technical Complexity Factor
\(^4\)ECF: Environmental Complexity Factor
\(^5\)EUCP: Early Use Case Point
\(^6\)UXEXUCP: Unadjusted Extended Use Case Weight
\(^7\)EXUCP: Extended Use Case Point
Table 3: Effort estimation by Application Function Point (AFP)

<table>
<thead>
<tr>
<th>Project</th>
<th>$AFP^8$</th>
<th>$VAF^9$</th>
<th>$AFPC^{10}$</th>
<th>Estimate (Man-hour)</th>
<th>Actual (Man-hour)</th>
</tr>
</thead>
<tbody>
<tr>
<td>BDR</td>
<td>403</td>
<td>1.16</td>
<td>467.48</td>
<td>1251.15</td>
<td>1392</td>
</tr>
<tr>
<td>PicShare</td>
<td>714</td>
<td>1.14</td>
<td>813.96</td>
<td>2273.26</td>
<td>2016</td>
</tr>
<tr>
<td>LBA</td>
<td>1054</td>
<td>1.19</td>
<td>1254.26</td>
<td>3572.15</td>
<td>3680</td>
</tr>
</tbody>
</table>

7.3 Application Function Point (AFP) Estimation

Sequence and class diagrams are available by elaborating the behavioral and structural aspects of the system in UML. With information available at this level of detail, we can count actual Function Points for each use case, following the method suggested in the study of measuring Function Points by UML Design Specification[26]. This provides a means to achieve the goal of fast measurement of function points.

7.4 Cost and Schedule Estimation Summary

The overall effort numbers for Location Based Advertising (3680 hours) add up to a little under 2 man-years, distributed across approximately 75 students over several semesters, to build a fully-functional system. It was this result that sparked our interest and has led to further study of how to leverage parallelism in development to reduce cost and compress schedules.

One of the main drivers behind Resilient Agile is the compression of schedule and reduction of overall cost of system development. Thus it is important to obtain unbiased and objective cost and schedule data on RA projects. This work is in a very early stage as we have only been able to study a small number of RA projects to date, and because the first RA project (Location Based Advertising) was mostly complete before we started the estimation work.

Since we are introducing a database and REST API code generator as part of the RA process, it’s important to quantify the effect of “executable domain models” on development. As seen in Table 1, a cost driver for automatic code generation was added to reduce the effective size of BDR, as is often done to account for other effort reduction strategies such as code reuse.

Finally the RA effort estimation model provides an opportunity to develop predictive corporate RA estimates, and to use these estimates to help guide project management in allocating development resources for development.

8 SCALABILITY AND COMPATIBILITY WITH THE INCREMENTAL COMMITMENT SPIRAL MODEL (ICSM)

The Resilient Agile (RA) process is consistent with the four principles of the incremental Commitment Spiral Model (ICSM):

- Stakeholder value-based guidance.
- Incremental commitment and accountability.
- Concurrent multi-discipline engineering.
- Evidence and risk-based decisions.

Stakeholder value-based guidance is built up via continuing involvement of stakeholders in iterating and discussing storyboards, prototypes, analyses, constraints, and concepts of operation. Incremental commitment and accountability is built up via periodic stakeholder reviews of the system’s increasingly complete definition, prototypes, future plans, and resource requirements. Concurrent multi-discipline engineering is built up via participation of stakeholder experts, and often of experts from second-order or third-order stakeholders. Evidence and risk-based decisions are made as stakeholders see that shortfalls in evidence are probabilities of stakeholder loss, which when multiplied by size of loss becomes Risk Exposure. Sometimes stakeholders will accept that the higher risk is worth the benefits of early system fielding, and decide to go forward, but in other cases will decide that obtaining further evidence of system feasibility is preferable.

Such risks are encountered in Pure Agile, in which developers choose easiest-first initial capabilities. This often achieves early user satisfaction, but often different users and developers make incompatible usage and implementation choices, leading to extensive refactoring and delays. The Architected Agile[1] approach addresses this problem, but at the expense of a longer startup.

RA can be considered as a new common case of the ICSM table of common process cases in Chapter 11 of the ICSM book[2], p.202. The RA common case is similar to the Architected Agile common case, with an additional criterion of a High-Very High premium on Rapid Fielding and Modification, and wider ranges on the criteria for Size/Complexity (Low to Very High) and Criticality (Low to Very High), giving RA the advantages of rapid development and high assurance.

The keys to RA superiority are its ability to enable many developers to work in parallel, and the three-team approach to have a continuing system engineering team handling the change traffic and emerging risks and stabilizing the developers, and the continuing independent verification and validation (V&V) team quickly finding problems for quick resolution and reducing technical debt, as shown in Figure 5 from[2], p. 245.

For two out of the three small and very quick and well-V&Ved RA projects summarized above, there was a single "keeper of the holy vision" performing the continuing systems engineering function, creating the architecture supporting multiple parallel developers, and working out stakeholder-coordinated feature content. There were relatively small but well-qualified V&V teams performing continuous testing.

To scale up the RA approach to large projects, alternative approaches for system engineering, development, and V&V can be applied. An excellent example is the million-line, 3-increment TRW Command Center Processing and Display System-Replacement (CCPDS-R) project summarized in[2], pp. 101-103, and described in detail in[27], pp. 299-362. Instead of a single system engineer, there was a team averaging around 10 system engineers working out an architecture enabling many programmers to work in parallel, not only developing Ada package specifications for the various modules, but also having them consistency-checked by the Ada compiler.
We have described an ongoing series of experiments in achieving Rapid, Evolutionary, Reliable, Scalable System and Software Development: The Resilient Agile Process. The process we have followed (Resilient Agile) includes extensive prototyping to mitigate technical risk, a minimalist UML-based process (Agile/ICONIX), and a code generator that enables “executable domain models” to be rapidly realized with a NoSQL database (Mongo DB) and a REST API (Node JS) that supports common database functions.

The experiments have been conducted with more than 100 Masters Computer Science students over a 3 year period on 3 projects to-date, and are still ongoing. The class projects have mostly been “directed research” students with a time budget of around 5 hours per week for a 3 month semester, and team sizes ranging from 10 to 47 students working in parallel. We have tried to do “just enough planning” to ensure that work developed independently by large numbers of students working in parallel integrates successfully, and in fact, integration has not proven to be a problem on any of the projects attempted.

Empirical results from the projects attempted (e.g. less than 2 man-years total effort for 75 students to build a complete Location-Based Advertising system) have been encouraging and detailed effort tracking and productivity measurements are being deployed on new projects in an attempt to quantify the empirical results more precisely. More data is being gathered to facilitate accurate calibration of cost models.

More generally, the USC and CCPDS-R projects identify some generally underutilized key enablers for speeding up software deliveries by enabling project participants to operate in parallel rather than sequentially. These include:

- Well-defined interface definitions between system components (REST APIs for the USC projects; Ada package specs for CCPDS-R);
- Having systems engineers, developers, and testers operating concurrently rather than sequentially;
- Just Enough Planning (balancing the costs and risks of doing too little vs. too much);
- Pinching the “cone of uncertainty” during the early op concept phase, using a mix of prototyping to evaluate technical alternatives and MVC decompositions to disambiguate behavior specs;
- Leveraging code generation of databases and APIs to rapidly respond to changing requirements;
- Organizing deliveries to be incremental or continuous;
- Getting project personnel and their other stakeholders to agree to the approach and to keep each other informed about impending changes;
- Providing rewards to participants for continuous performance and providing useful improvement suggestions.

9 CONCLUSIONS

We have described an ongoing series of experiments in achieving schedule compression by exploiting parallelism in development. The process we have followed (Resilient Agile) includes extensive prototyping to mitigate technical risk, a minimalist UML-based process (Agile/ICONIX), and a code generator that enables “executable domain models” to be rapidly realized with a NoSQL database (Mongo DB) and a REST API (Node JS) that supports common database functions.

The experiments have been conducted with more than 100 Masters Computer Science students over a 3 year period on 3 projects to-date, and are still ongoing. The class projects have mostly been “directed research” students with a time budget of around 5 hours per week for a 3 month semester, and team sizes ranging from 10 to 47 students working in parallel. We have tried to do “just enough planning” to ensure that work developed independently by large numbers of students working in parallel integrates successfully, and in fact, integration has not proven to be a problem on any of the projects attempted.

Empirical results from the projects attempted (e.g. less than 2 man-years total effort for 75 students to build a complete Location-Based Advertising system) have been encouraging and detailed effort tracking and productivity measurements are being deployed on new projects in an attempt to quantify the empirical results more precisely. More data is being gathered to facilitate accurate calibration of cost models.

More generally, the USC and CCPDS-R projects identify some generally underutilized key enablers for speeding up software deliveries by enabling project participants to operate in parallel rather than sequentially. These include:

- Well-defined interface definitions between system components (REST APIs for the USC projects; Ada package specs for CCPDS-R);
- Having systems engineers, developers, and testers operating concurrently rather than sequentially;
- Just Enough Planning (balancing the costs and risks of doing too little vs. too much);
- Pinching the “cone of uncertainty” during the early op concept phase, using a mix of prototyping to evaluate technical alternatives and MVC decompositions to disambiguate behavior specs;
- Leveraging code generation of databases and APIs to rapidly respond to changing requirements;
- Organizing deliveries to be incremental or continuous;
- Getting project personnel and their other stakeholders to agree to the approach and to keep each other informed about impending changes;
- Providing rewards to participants for continuous performance and providing useful improvement suggestions.

REFERENCES


Searching for Common Ground: Existing Literature on Automotive Agile Software Product Lines

Philipp Hohl  
Daimler AG, Research and Development, Germany  
philipp.hohl@daimler.com

Javad Ghofrani  
Leibniz Universität Hannover, Germany  
javad.ghofrani@inf.uni-hannover.de

Jürgen Münch  
Reutlingen University, Germany and University of Helsinki, Finland  
juergen.muench@reutlingen-university.de

Michael Stupperich  
Daimler AG, Research and Development, Germany  
michael.stupperich@daimler.com

Kurt Schneider  
Leibniz Universität Hannover, Germany  
kurt.schneider@inf.uni-hannover.de

ABSTRACT
The digital transformation of the automotive industry has a significant impact on how development processes need to be organized in future. Dynamic market and technological environments require capabilities to react on changes and to learn fast. Agile methods are a promising approach to address these needs but they are not tailored to the specific characteristics of the automotive domain like product line development. Although, there have been efforts to apply agile methods in the automotive domain for many years, significant and widespread adoptions have not yet taken place. The goal of this literature review is to gain an overview and a better understanding of agile methods for embedded software development in the automotive domain, especially with respect to product line development. A mapping study was conducted to analyze the relation between agile software development, embedded software development in the automotive domain and software product line development. Three research questions were defined and 68 papers were evaluated. The study shows that agile and product line development approaches tailored for the automotive domain are not yet fully explored in the literature. Especially, literature on the combination of agile and product line development is rare. Most of the examined combinations are customizations of generic approaches or approaches stemming from other domains. Although, only few approaches for combining agile and software product line development in the automotive domain were found, these findings were valuable for identifying research gaps and provide insights into how existing approaches can be combined, extended and tailored to suit the characteristics of the automotive domain.

CCS CONCEPTS
-Software and its engineering → Software product lines;

1 INTRODUCTION
Agile development within Software Product Lines (SPLs) is not new. Several studies treated this issue before. Perez and Silva [15] conducted a mapping study and a structured literature review on agile software product line engineering. Our study extends the findings from the survey and includes references published after 2011. We primarily focus on literature which addresses the automotive domain. The aim of our study is to increase the understanding on how agile software development (ASD) might be integrated within an automotive software product line. We limited the scope to automotive embedded software development. In this context it is important to consider deep integration between hardware and software, strong focus development processes, strong supplier involvement, and safety-critical functionality [17]. Furthermore, specific testing conditions, like tests in the real car, must be considered.

The presented study examines existing challenges which arise when agile development and SPLs are introduced in the automotive domain and how they can be overcome.

Existing literature deals with (1) the use of agile methods in the automotive domain, (2) the use of product line development in the automotive domain, and (3) the combination of agile methods and product line development. This study offers an intersection of all three topics. Furthermore, the literature review helps to identify already existing solutions in published literature.

The remainder of this paper is structured as follows. Section 2 describes the method used for the literature review. Section 3 reports the contributions of the collected studies. Section 4 summarizes key findings as well as implications for practitioners. Section 5 presents conclusions due to the findings. Finally, Section 6 summarizes the paper and gives an outlook on future work.
2 RESEARCH DESIGN

In order to conduct a literature review, two main research methods are applicable: A mapping study as well as a structured literature review. Both methods share some commonalities. For the presented study, the researchers conducted a mapping study as this method enables to identify gaps in literature to the specific research field [16, 56]. In addition, the mapping study allows to gain a wide overview of the research area [36]. The research process described by Petersen et al. [56] served as a basis for our research. Researcher 1 (P. Hohl) was following the process primarily. This process was extended by the data collection of Researcher 2 (J. Ghofrani). He conducted a snowballing search to find related literature which helps to answer the research questions. By including this literature and combining it with the results from Researcher 1, we achieved a broader view on the topic. With this, related research areas are taken into account. Whenever we deviated from the process, we describe our actions in the following subsections.

The process by Peterson et al. [56] is illustrated in Figure 1, whereas solid lines and white background define Process Steps and dashed lines with grey background the corresponding Outcome to each process step.

![Research process by Petersen et al. [56]](image)

In the first step of the research process, research questions were defined. In the next step, we conducted the search within suitable reference databases. Therefore, a search string was constructed. We extended the results by two different searches, conducted by two researchers independently. In the third step, the papers were screened by applying defined inclusion and exclusion criteria. Whenever, it was not clear how to categorize a paper, it was discussed by Researcher 1 and Researcher 2 in a review meeting. The abstracts were screened in detail and a mapping process was executed as a last step in the mapping process.

2.1 Research Questions

The general question that guides this mapping study is: What is the state-of-the-art to combine agile software development and software product lines in the automotive domain, according to published literature? The question is divided into three research sub-questions to provide different views on the topic. Figure 2 summarizes the questions.

**RQ 1:** In what ways can software product lines be combined with existing agile software development in the automotive domain, according to published literature? The goal is to investigate how ASD in the automotive domain could be extended by a SPL. Ideas, found in the existing literature, are evaluated, which foster an integration and focus on existing challenges for the introduction of a SPL.

**RQ 2:** In what ways can agile software development be combined with existing software product lines in the automotive domain, according to published literature? This question investigates how current SPL development in the automotive domain could be extended by agile development approaches. Ideas found in the existing literature are evaluated, which foster an integration and focus on existing challenges to introduce ASD.

**RQ 3:** Are there any suitable concepts from agile software product lines to adapt within the automotive domain, according to published literature? To complete the picture, this question identifies how concepts for Agile Software Product Lines (ASPLs) and agile embedded software development can be adapted to the automotive domain.

2.2 Conducting the Search

This section describes the research process for the mapping study executed by Researcher 1 and, furthermore, describes the research process applied by Researcher 2. The individual results were combined and screened as described in Section 2.3.

2.2.1 Search Process Applied by Researcher 1. The used research process based on Petersen et al. [56] is divided into five steps. These steps were executed in a sequential order:

**Selection of databases.** The selection of databases is essential to make the research outcome as comprehensive as possible. Eight databases were selected for the mapping study: Scopus, Science Direct, IEEE, ACM DL, Springer Link, Wiley Online Library, World Scientific, and Google Scholar. With these databases, a comprehensive search was conducted to avoid bias [56]. Although Scopus covers IEEE Xplore and Elsevier, the two databases were included to verify the quality of the search strings.

**Definition of the search string.** The construction of the search string for the mapping study was conducted as follows: First, three main keywords were selected. For these three keywords, synonyms and a corresponding word family were defined (cf. Table 1).

![Overview search keywords](image)

<table>
<thead>
<tr>
<th>Search keyword 1</th>
<th>Synonyms</th>
<th>Word family</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;software product line(s)&quot;</td>
<td>&quot;software product line engineering&quot;</td>
<td>&quot;software product line engineering&quot;</td>
</tr>
<tr>
<td>&quot;product line(s)&quot;</td>
<td>&quot;software product line development&quot;</td>
<td>&quot;software product line engineering&quot;</td>
</tr>
</tbody>
</table>

**Search keyword 2** | "automotive" | "regulated domain(s)" |
| Synonyms | "car manufacturing" | "car development" |

**Search keyword 3** | "agile software development" | "agile methods" |
| Synonyms | "agile practices" | "agile software development" |
| Word family | | |
The shape of the search string was adapted to the selected databases, since search options differ and are specific for each search engine. The search was conducted using titles, abstracts and keywords.

2.2.2 Search Process Applied by Researcher 2. Researcher 2 conducted a systematic literature review using the snowballing method as a systematic approach. Therefore, he was following the “Guidelines for Snowballing in Systematic Literature Studies and a Replication in Software Engineering” by Wohlin [72].

Identify a start set of papers. This step identifies a start set of papers in order to use it for the snowballing procedure. The start set consists of literature which is related to the research. Wohlin [72] mentioned, that there exists no silver bullet for identifying the start set. To conduct the search it is necessary to include relevant papers. Those papers may come from different communities.

Selection of databases. Selecting databases which contain the related scientific studies for ASD methods and software product line engineering is an important factor to achieve valid search results. Five databases were selected for the creation of the start set: ACM Digital Library, Web of Science, IEEE Xplore, Science Direct, and Wiley InterScience.

Building a start set. An initial search string is built from the combination of related keywords to software product line development and agile software development. Keywords and corresponding words are defined in Table 2. This results in the following search string: (“Agile” OR “SCRUM” OR “Kanban”) AND (“software”) AND (“product line” OR “product family” OR “reuse” OR “variability”) AND (“methode” OR “development” OR “progress” OR “engineering”)

The shape of the search string was adapted to the selected databases. In total, 26 studies were identified and build up the start set for the Systematic Literature Review using the snowballing approach.

Performing the snowballing. For snowballing, the ReviewR tool, was used. This tool is an in-house development of the Software Engineering Group of Leibniz Universität Hannover to conduct snowballing. The tool manages the search iterations and performs the snowballing within a Scopus database. Five iterations with forward and backward snowballing were performed. If the title and the abstract of the publication contains any combination of agile methods and SPLs it is included for the next iteration. We excluded duplicates, studies older than 2001 and studies which are only mention one keyword.

2.2.3 Definition of Inclusion and Exclusion Criteria. To eliminate studies which are not relevant for answering the research questions, exclusion criteria were applied [56].

Inclusion criteria:

- The paper discusses agile software product line development in the automotive domain.
- The paper discusses product line engineering in the automotive domain.
- The paper discusses techniques to speed up feature development in the automotive software development.

Table 2: Overview search keywords for building the start set

<table>
<thead>
<tr>
<th>Search keyword 1</th>
<th>Agile</th>
</tr>
</thead>
<tbody>
<tr>
<td>Related keywords</td>
<td>“SCRUM”</td>
</tr>
<tr>
<td></td>
<td>“Kanban”</td>
</tr>
<tr>
<td>Search keyword 2</td>
<td>“software”</td>
</tr>
<tr>
<td>Related keywords</td>
<td>no related keyword selected</td>
</tr>
<tr>
<td>Search keyword 3</td>
<td>“product line”</td>
</tr>
<tr>
<td>Related keywords</td>
<td>“product family”</td>
</tr>
<tr>
<td></td>
<td>“feature”</td>
</tr>
<tr>
<td></td>
<td>“reuse”</td>
</tr>
<tr>
<td></td>
<td>“variability”</td>
</tr>
<tr>
<td>Search keyword 4</td>
<td>“methode”</td>
</tr>
<tr>
<td>Related keywords</td>
<td>“development”</td>
</tr>
<tr>
<td></td>
<td>“progress”</td>
</tr>
<tr>
<td></td>
<td>“engineering”</td>
</tr>
</tbody>
</table>

OR “car manufacturing” OR “car development”) AND (“agile software development” OR “agile methods” OR “agile practices”)
- The paper discusses agile software development for embedded software.
- The paper discusses agile software development in combination with the use of SPLs.
- The paper is written in English or German.

Exclusion criteria:
- The paper was written before 2001.
- The paper has no accordance with at least two of the search keywords.
- The paper focuses on a specific tool. (This study refers to methods and not to specific tools.)
- The paper is not accessible.
- Duplicated papers.

2.2.4 Selection of Researchers to Conduct the Study. Due to the fact that the mapping study is part of a PhD project, Researcher 1 conducted the mapping study. Researcher 2 conducted a separate study and added his literature to extend the dataset and to give a broader view with respect to the related research areas.

2.2.5 Execute the Research. As described in the previous steps, the research was executed in the selected databases.

2.3 Screening of Papers

In this step, the previously defined inclusion and exclusion criteria were used to select the most suitable set of papers. The screening process was divided into several filtering steps as shown in Figure 3. Papers were included if they fulfilled the inclusion criteria. This filtering was applied to the results of Researcher 1 and Researcher 2.

The search of Researcher 1 resulted in 5630 identified contributions. In the first step of the screening process, the inclusion and exclusion criteria were applied on the title and keywords of the identified studies. Duplicate papers were excluded as well. Whenever Researcher 1 was not able to include or exclude the paper, he marked the paper as “unclear”. In this filtering step, 23 papers were classified as unclear. These papers were reviewed by Researcher 2 and categorization was discussed with Researcher 1 in a review meeting. If the paper was still unclear after this process, another researcher reviewed the paper and decided to include it to one of the given categories or exclude it. This filtering step leads to 181 remaining papers.

In addition, Researcher 2 separately searched with a broader approach (cf. Section 2.2.2) to find successful implementations and experience reports examining the combination of ASD and the use of SPLs. This search resulted in a total number of 91 resources.

Both data sets were combined and Researcher 1 deleted the double references in the dataset. For the remaining 246 references, the abstracts were extracted and stored in the reference management and citation tool Citavi. Researcher 1 evaluated the abstracts. This last filtering step led to a final set of 68 papers.

2.4 Categorizing the Studies

For later data extraction, the results of this study were categorized. To analyze, categorize and to sort the collected literature, we used the tagging and coding functionality in the reference management and citation tool Citavi. The first step in Citavi was to read abstracts and identify keywords and concepts reflecting the contribution of the papers [56]. The selected keywords were used for clustering the papers into categories [56]. According to Figure 2, three main categories were defined. The output of this step gave an overview of the covered topics and areas of the selected papers and defined the current state of research at a granular view.

2.5 Data Extraction

For each specified category, the data of each paper was extracted. The first step of data extraction gathered standardized information like author, type of paper and publication channel. This information was used to give an overview about the active participants in the considered area of research. Furthermore, the results showed frequencies of papers for each category. This enables us to identify gaps and possibilities for future research [56]. After getting an overview of the papers, detailed data was extracted in order to answer the research questions.

2.6 Threats to Validity

This section treats the identified threats to the validity of our study.

Research questions: We defined the research questions to be unambiguously understandable. Therefore, we reviewed the wording of the research questions several times. It would be false to affirm that our research questions cover the complete existing research on ASPLs, containing all related research areas. However, our research questions address an urgent issue in that field.

Publication bias: Due to the fact, that only Researcher 1 was doing the screening process, we can not guarantee that all relevant primary studies were selected. It might be possible that papers were excluded in the filtering steps. For the data collection, we mitigated this by adding the literature from Researcher 2. Furthermore, a review session was introduced, to discuss the “unclear” marked publications.

Search conducted: The shape of the search string was adapted to the selected databases, since search options differ and are specific for each search engine. We mitigate the threats introduced by this adaption by using eight databases. Although the database Scopus covers IEEE Xplore and Elsevier, the two databases were included to verify the quality of the search strings. This threat was mitigated further, by adding the literature from Researcher 2.

Data extraction: Data extraction and categorization of the papers was done by Researcher 1. This could introduce bias and incorrectly classified papers. Researcher 1 conduct this classification two times to avoid a wrong categorization.
3 REPORTING THE REVIEW
To visualize the results, we divided them into the three categories from Figure 2. The following Table 3 contains the number of studies according to the publishing date. As shown in Table 3, a lot of research focus on related research areas. The number of papers have increased since 2004. It furthermore illustrates that papers are rare with respect to the automotive domain. Only in 2014, four papers focus on ASD and address the automotive domain.

Table 3: Publications by year

<table>
<thead>
<tr>
<th>Year</th>
<th>Automotive &amp; Agile</th>
<th>Automotive &amp; SPL</th>
<th>Related Research Area</th>
</tr>
</thead>
<tbody>
<tr>
<td>2004</td>
<td>[40, 41]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2005</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2006</td>
<td>[37, 55, 66, 68]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2007</td>
<td>[31, 50]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2008</td>
<td>[70]</td>
<td>[23, 26, 44, 45, 49, 58, 67]</td>
<td></td>
</tr>
<tr>
<td>2009</td>
<td>[64]</td>
<td>[4, 6, 25, 42, 51]</td>
<td></td>
</tr>
<tr>
<td>2010</td>
<td>[22, 24, 47, 59]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2011</td>
<td>[39]</td>
<td>[5, 7–9, 11, 14, 15, 28, 38, 57]</td>
<td></td>
</tr>
<tr>
<td>2012</td>
<td>[30]</td>
<td>[10, 17, 32, 35, 43, 52, 61, 69]</td>
<td></td>
</tr>
<tr>
<td>2013</td>
<td>[18]</td>
<td>[3, 33, 53]</td>
<td></td>
</tr>
<tr>
<td>2014</td>
<td>[1, 34, 62, 63]</td>
<td>[13]</td>
<td>[2, 19–21, 27, 46, 48, 65]</td>
</tr>
<tr>
<td>2015</td>
<td>[29, 71]</td>
<td>[73]</td>
<td>[12]</td>
</tr>
<tr>
<td>2016</td>
<td>[60]</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

However it seems, that no paper handles the combination of ASD in the automotive domain with respect to SPL development. This will be analyzed in more detail. The following subsections will derive results from the literature review to answer the research questions:

3.1 Research Question 1
In what ways can software product lines be combined with existing agile software development in the automotive domain, according to published literature?

The published literature does not provide significant information or approaches on how to adapt existing ASD approaches to SPL development in the automotive domain. Therefore, it is difficult to answer the research question based on information from existing literature. However, several publications were found that focus on the adaptation of ASD approaches to the specifics of the automotive domain. As SPLs are a common approach in the automotive domain it could be argued that this adaptation might at least implicitly consider also product line development. The findings can be summarized in the following: (1) Adapting ASD in the automotive typically concentrates only on selected agile elements such as Continuous Integration (CI). The published literature does not show any recommendations to use a comprehensive set of agile elements and practices together in the automotive domain. (2) The majority of the published literature suggests that agile models and processes should get customized to the specifics of the automotive domain before they are implemented in practice. (3) Several agile models and processes that are already customized to the specifics of the automotive domain are proposed in the published literature. Examples are the Feedback Loop Model that especially considers the collaboration between different organizations (such as OEMs and suppliers) and the Agile Hybrid Assessment Method that especially considers safety-criticality constraints in the automotive industry. These methods and processes include interesting new concepts such as virtual integration on the system level.

Thiel et al. [64] analyze the combination of ASD and plan-driven processes which could be seen as a typical characteristic of current automotive software development. They emphasize that a combination could be beneficial under certain conditions such as rigid quality and safety requirements [64]. As a solution, Thiel et al. [64] suggest to introduce selected agile approaches to automotive systems engineering. This suggestion is also supported by the “Agile in Automotive Survey” from Kugler and Maag [71]. The survey highlights that many companies recommend to only introduce single agile elements into existing and proven development cycles. In addition, they recommend not to apply a purely SCRUM approach in practice. Instead, they emphasize that pure SCRUM is often not applicable due to the context factors and the development environment.

Schlosser et al. [60] define an upcoming challenge in the automotive software development. They identify the necessity of shorter development time for multiple software variants by considering constraints of a high cost pressure. They further argue that more incremental software deliveries are necessary for a quick response from customer which shortens the development time and save money [34, 60]. In addition, the agile element of CI provides a quick response with respect to functional aspects [1, 60, 70]. In order to integrate further agile elements into the automotive domain, different models and processes are introduced including a (1) Feedback Loop Model, the (2) Agile, Hybrid Assessment Method for the Automotive Industry and the (3) Mega Scale Software Product Line Engineering:

The concurrent Feedback Loop Model (1) introduces feedback loops for different organizations to enable cooperation [29]. The cooperation (including communication) between different organizations is managed by a new architect role to shorten the development time. The reduction of time is validated in a case study. In turn, the case study reveals several drawbacks [29].

The Agile, Hybrid Assessment Method for the Automotive Industry (AHAA) (2) considers safety critical development. Software quality is important for software critical companies like in the automotive domain to ensure high reliability and maturity of software. Plan-driven practices are assessed with respect to software process improvement like CMMI and ASPICE [63].

Additionally, the assessment suggests agile based improvement solutions within the automotive domain. Furthermore, the System Architecture Virtual Integration (SAVI) initiative helps to improve the agile development. In SAVI, CI is concurrently operated within the
software development process. Integration starts with the earliest available system models into a virtual integration environment. With a virtual integration strategy vendors are more closely tied to the project [62].

3.2 Research Question 2

In what ways can agile software development be combined with existing SPLs in the automotive domain, according to published literature?

Research Question 2 could not be answered by the selected findings. The majority of the published literature is concerned with SPLs within the automotive domain without considering an adaptation to ASD. Several variant management approaches and appropriate methods and processes to handle variants are discussed. Some product line approaches are presented that might be helpful in understanding what to consider when combining with agile development. Understanding binding times, for instance, might help to better organize CI mechanisms.

According to Wozniak et al. [73], the automotive domain is the most challenging environment for systems and SPL engineering. Millions of different software variants exist, where each encompasses a large complexity. The complexity is based on the large number of variation points within the product [21]. Four different binding times may be distinguished: programming, integration, assembly, and runtime. All of them are used in the automotive domain. This leads to a "very large number of individually complex products with incomparably rich feature variation among them" [73, p.336]. In order to handle the complexity, the Mega Scale Software Product Line Engineering (MS-SPLE) (3) approach is introduced. This approach is applied for large product sets with complex products and complex feature variations like the automotive domain. MS-SPLE is a possibility to manage, e.g., calibration parameters for the software variation mechanism and the complexity management. However, ASD is not taken into account within MS-SPLE [73].

3.3 Research Question 3

Are there any suitable concepts from agile software product lines to adopt within the automotive domain, according to published literature?

The published literature provides several sources of information on concepts for the combination of ASD and SPLs. These concepts could be used as candidates for the adoption in the automotive domain or they might reveal helpful insights for an adaptation in the automotive domain. The findings can be summarized as follows: (1) Several concrete methods and models are described that combine agile and product line concepts. An example is ScrumPL which supports iterative domain and application engineering. (2) Specific cross-cutting aspects such as architecture, scoping, or communication are considered in the published literature with respect to the combination of agile and product line development. An example is a scoping approach that allows conducting a commonality and variability analysis in each iteration in order to constantly understand the implications of new or changing requirements. (3) Several publications present expected or experienced benefits of agile product line development. Among others, frequently mentioned benefits are increased flexibility, shortened development cycles, and in consequence a reduction of technological, customer, and market risks. (4) Different strategies exist to introduce a combination of agile and product line development. Although it is difficult to say with certainty, the primary strategy seems to be introducing agile into well-established SPL development. (5) The combination of agile and product line development significantly depends on customer needs (such as availability or cost efficiency).

Besides concepts from agile product line development the published literature includes information on agile development in embedded domains. This information might be helpful for transitioning towards agile product line development in the automotive domain.

In 2007, Hummel and Atkinson [31] stated, that agile development and systematic software reuse have rarely been attempted in the same project. Pohjalainen [57] describes problems with the combination of agile and product line development based on the required upfront planning in SPLs, while agile method practitioners inherently want to avoid any heavy up-front planning [57]. To overcome this incompatibility, Pohjalainen [57] describes a bottom-up approach to combine product line engineering and formal modeling with ASD.

Martini [43] mentioned, that agile and reuse practices do not hinder each other. Some sources emphasize the benefits of introducing ASD practices to SPLs [26]. Potential benefits include the improved change management for requirements, the increased product quality with coincident decrease of development costs, or a reduced time to market [20]. The development enriched by agile practices is often referred to as Agile Product Line Engineering (APLE) [48].

Mainly two approaches for introducing APLE are reported: Either take an existing SPL process and introduce agility or take an agile process in order to tailor it for SPLs [66]. The focus is primarily on introducing agile into well-established SPL rather than on the implementation of software reuse into an agile-oriented development [43]. It is essential to develop a reuse strategy that complements the principles of agile development and to foster software reuse in combination with the practices of agile methods [31]. Hanssen [28] concludes that a combination makes the organization more flexible and thus capable of serving a volatile market with fast-changing technologies. Furthermore, this enables the organization to collaborate better with suppliers [28]. The published literature can be structured by two different viewpoints: The technical viewpoint and the business viewpoint.

Regarding the business viewpoint, it is often mentioned that it is essential to deliver high-quality software in time and within estimated cost and effort as time to market is an increasingly important success factor for companies. It is, therefore, important to know, which parts of the software development process are able to become more agile and how to apply the agile practices to speed up the development [40].

One main goal with respect of the SPL is to satisfy a wide range of customer needs [3]. Atherton and Collins [3] describe challenges of product lines and compare a planned development strategy against a haphazard reuse strategy for software components in aircraft engines. They describe different needs of military customers, who focus on capability and system availability. In addition, they consider commercial customers, who focus on minimizing life cycle cost. McGregor [45] emphasizes, that an entire customer satisfaction always requires a tradeoff between flexibility and software.
waste due to late rework of code and a late binding time of variability [38, 46]. The technical viewpoint can be divided into agile methods and models, which are already in use and agile aspects like documentation or scoping, which should be considered for a successful combination.

3.3.1 Agile Methods and Models. Different agile methods are described to combine agile software development and a reuse strategy of software components. Well-known agile methods such as Scrum and XP are described in the literature. Furthermore, various papers present a tailored approach to use well-known agile methods. Lindvall et al. [40] describe a project which utilizes the XP approach. This is achieved by auxiliary processes in order to define the scope and to adjust delivery time in advance. With respect to Scrum, a new method called ScrumPL is introduced. Scrum lifecycle phases and the SPL sub-processes are combined to form ScrumPL [59]. In this tailored Scrum, domain engineering as well as application engineering processes are performed in an iterative way separated in different sprints [14].

Another approach of combining SPLs and agile development is the Feature Driven Approach with the use of Feature Models [2, 58]. This hybrid method captures the initial view of the results of the commonality and variability analysis [45]. In this approach, both disciplines are merged. As a result, feature orientation blends the benefits of product line engineering with those of agility [35]. As an extension of the feature model, the composite feature model is described by Urii et al. [69]. Their solution relies on the definition of composite feature models and the use of a model-driven evolution process to support it on large real systems [69]. In addition, Trinidad et al. [68] evaluate the possible reasons why a feature model has dead features and how to evaluate feature models in general.

Neves and Vilain [48] present the idea of Test-Driven-Development (TDD) as the basis to combine ASD and SPLs. The usage of refactoried agile practices lead to a reactive SPL. Therefore, SPL evolves and acquires variability points on demand [48]. Due to the speed up of the development, it is necessary to retest and certificate some parts repeatedly [3]. It is, therefore, essential to automate test procedures. Continuous Integration and Continuous Testing (CT) are necessary to test new working products in each iteration in a flexible and rapid way. However, in agile methodologies, testing tools should be capable to test the developed software. It is a significant effort to set up tools and to maintain the test environment [42]. Ghanam et al. [26] extend the idea of TDD and describe tests not only in form of regression tests. They rather describe the test approach as a way to determine commonalities in software systems of the same domain [26].

3.3.2 Agile Aspects. This section presents cross-cutting aspects like architecture, scoping, communication and safety regulations. These aspects should be considered in detail when combining ASD and SPLs.

Architecture. McGregor [45] suggests an architecture definition practice that replaces the product line and agile practices focusing on the structure, commonality and variability. Typically, agile development disregards a detailed consideration of the architecture. However, in product line development, architecture is an important aspect to consider [4]. It is necessary to find the right tradeoff between the management of architecture evolution and refactoring without sacrificing the principles of agility. Furthermore, architecture erosion should be avoided at any time [8]. The use of proprietary architectures could help to avoid this. In contrast, Chong [9] promotes the adoption of open standards, rather than closed proprietary architectures. He mentions this as an essential economic driver for opening closed markets to competition [6, 9].

Scoping. Ghanam et al. [22] introduce a new lightweight variability analysis technique. This enables to determine commonalities and variations between the new requirements and the existing ones. The analysis is conducted in every agile iteration and not as traditional up-front [22]. In addition, another lightweight approach is presented by Atherton and Collins [3]. They divide requirements into small pieces of functionality, categorize and priorities them [3].

Communication. An important key factor for a successful combination of ASD and SPLs is communication [27]. Intra- and inter-organizational communication practices and the awareness for software reuse are essential [43]. Good communication is necessary within development teams, between teams, and the units these depend on, e.g., validation and business units [43]. It is important to keep the communication overhead low [35].

Safety regulations. Kirchner and Hofman [35] conduct a study based on findings revealed in the area of medical device development. In this area, they have to fulfill certain pre-requisites with regards to patient safety. Development organizations have to adhere regulations to comply with the law [35]. Regarding regulations and standards, process descriptions and development documentation are required. However, this may become inefficient if the overhead to organize document writing, signing, reading, reviewing, and revising increase. A production of waste such as functionality that potentially never gets shipped to a customer should be avoided [35].

Regarding the combination of agile and embedded development, the published literature reveals that the motivation for using agile methods is often characterized by market uncertainties, competitive pressures, and the need to shorten development cycles [33].

In the embedded domain, safety aspects are a prime goal to satisfy [32]. Therefore, the use of agile practices in regulated and safety-critical domains is still limited [32]. With respect to the literature, the study of Kaisti et al. [33] finds no reason why agile methods could not be used in the embedded domain. XP and Scrum or a mixture of both are the wide-spread used methods [61]. Eklund et al. [19] point out the necessity to consider different development cycle-times for hardware and mechanics. It is important to synchronize these cycles and to freeze the design at a quality gate and milestone. Traditionally, no or minimal changes are allowed. Subsequent processes like optimizing the manufacturing and sales are often seen as more important. Nevertheless, software is strongly dependent on mechanical structures [19]. They further mention a long feedback loop with customers and management. Furthermore, a long-term predictability is hard to achieve with short-term agility [19].

4 DISCUSSION

We approached the issue of ASD in combination with SPLs in the automotive domain from three different perspectives.
The first direction was treated by Research Question 1. The question focuses on existing ASD in the automotive domain. It further examines, if SPL development could be combined with agile development processes so that they can benefit from both, agility and systematic reuse through product lines. Moreover, our work concludes that there is an existing gap in the literature with respect to agile SPL development in the automotive domain. Altogether, our work reveals many fields for further research, such as organizational, process, and product-oriented challenges. It further helps to identify means for addressing specific characteristics and restrictions of the automotive domain.

For future work, we aim at transferring the presented solutions to the automotive domain and evaluate them in existing development projects. As a result, we plan to create and evaluate a model for the adaption of ASD in the context of existing automotive SPL development.

REFERENCES


Table 4: Challenges identified in related research areas and potential solutions for the automotive domain

<table>
<thead>
<tr>
<th>Identified Challenge</th>
<th>Possible Solution</th>
<th>Potential Transfer</th>
</tr>
</thead>
</table>
| - Market uncertainties and fast changing requirements [2, 5, 6, 20–22, 37, 38, 41, 47, 48, 51, 54, 65–67] | - Faster customer feedback and agile development to be more flexible in the development. | - Set up a new feedback culture for customers. This leads to a restructuring in the organization and the possibility for the project managers to act as internal customers to promote ideas and new features.  
- Get direct customer feedback in beta-test drives on a separate test route. |
| - Competitive pressure and the need to shorten development cycles [4, 8, 20, 37, 38, 41, 47, 49, 50, 52, 54] | - Innovative features implemented in an agile way could be delivered faster to foster business advantages. | - A deployment of beta-software to the customer is hard to apply. Restrictions given by the law and relevant safety issues are of highest priority. The use of SPLs helps to reuse already certificated software parts and to save time.  
- Auxiliary tasks like CI help to ensure high quality software due to automated software test at a high pace. |
| - Different development cycle-times for related development systems (e.g. hardware or mechanics) [3, 15, 17, 19, 27, 33, 54, 61] | - Synchronize development cycles. | - Embed the agile development and establish an interface between the agile and the surrounding processes.  
- Synchronize important dates like the hardware release with the software development release dates. |
| - Long-term predictability is hard to achieve [5, 7, 23, 28, 46, 52, 58] | - More responsibilities for lower hierarchy level to get a flexible development. | - Flatten the hierarchy to shift decisions about implementation to lower hierarchy levels. Less communication effort shortens the decision time and therefore the overall time for development. |
| - It is unclear how to manage software reuse and agile development [4, 7, 10–12, 15, 21–25, 27, 31–33, 35, 42, 44, 45, 47, 49–51, 55, 57–59, 66, 68, 69] | - New lightweight variability analysis technique for faster analysis. | - Resolve the hierarchies and give responsibilities to lower hierarchy levels.  
- Set up a lightweight process to do the agile scoping.  
- Resolve problems with world-wide coordination. |
| - The usage of open source software [6, 9] | - The usage of open source software could foster competition and innovation. | - Not desired in the automotive domain. Every OEM wants to protect their intellectual property. Open source software is not considered here. |
| - Communication overheads and problems caused by outsourcing and attitudinal (cultural) incompatibilities in agile projects [3, 6, 8, 12, 23, 28, 35, 43, 61] | - New lightweight process and a mindset change of employees. | - Increase the amount of in-house development for less communication.  
- It is not possible to force a mindset change. Sensitize employees and give them enough time to familiarize with the situation.  
- Define interfaces, first, between traditional SPLs and agile development, as well as, second, between supplier and OEM. |
ICSSP’17, July 2017, Paris, France
Philipp Hohl, Javad Ghofrani, Jürgen Münch, Michael Stupperich, and Kurt Schneider
How Do Agile Practices Support Automotive SPICE Compliance?

Philipp Diebold  
Fraunhofer ISE  
Kaiserslautern, Germany  
philipp.diebold@iese.fhg.de

Thomas Zehler  
Fraunhofer ISE  
Kaiserslautern, Germany  
thomas.zehler@iese.fhg.de

Dominik Richter  
IBM Deutschland GmbH,  
Frankfurt, Germany  
d_richter10@cs.uni-kl.de

ABSTRACT

In the automotive domain, Automotive SPICE® is a common requirement for software and system development. Most automotive companies face the problem of complying with this model, especially, when dealing with agile development in order to benefit from advantages, such as higher customer collaboration or faster development. This paper describes the evaluation of how agile practices support Automotive SPICE requirements, based on literature reviews and expert opinion and including 772 mappings. 103 of 155 agile practices including Scrum and XP were used to cover 173 of 185 Automotive SPICE requirements (base practices and work products). Most Automotive SPICE base practices are supported (96%), while there is less support for work products (87%). The results are coherent with existing more high-level mappings but show more details. These results enable companies to benefit from agile without compromising Automotive SPICE conformity.

CCS CONCEPTS
- Software and its engineering → Software development methods;

KEYWORDS
Agile Development, Agile Practices, Automotive SPICE, SPI.

ACM Reference format:

1 INTRODUCTION

In the last decades, many companies adopted agile development approaches such as the most common ones Scrum, XP or their combinations / adaptions [1]. The reasons for this change are expected benefits such faster development, responding to change, customer collaboration and better quality software.

However, this trend focuses on the information system domains [1] and is only very slowly covering all project phases of embedded software development domains, such as Automotive, Avionics, Railway, or Defense. These embedded domains only use a small number of single practices from agile development, such as pair programming or burn-down charts, and only in some initial phases, e.g. pre-development [2].

The reason for their sparse use of agile development is most often the need to comply with mandatory standards and models, mainly common capability models, such as CMMI for Development v1.3, ISO/IEC15504 (SPICE) or domain-specific adaptions as Automotive SPICE v3.0 (registered trademark of VDA”) or MDevSPICE. A common misunderstanding wrt. agile development is that agile is incompatible with most of these capability models [3] and thus, is not usable in such domains.

In our opinion, agile development and the above mentioned standards and models are not contradicting each other since the standards define a high-level “What” to do, and agile development practices detail the “How” to do. A number of publications address this combination, such as [4, 5, 6]. They focus on a very limited and high-level scope. They do not provide the necessary detailed guideline for the practitioners from industry about how to implement the standards when making use of agile development.

Our work answers the research question (RQ) of how practices agile map/support necessities from capability models in a practical way for the automotive domain. It is motivated due to many companies containing OEMs and suppliers asking for guidance on the combination of Agile and Automotive SPICE® version 3.0. Thus, our results can be used to develop processes with agile elements in an Automotive SPICE compliant way. Our detailed mapping of Agile Practices to Automotive SPICE does not cover Safety aspects.

2 AUTOMOTIVE SPICE IN A NUTSHELL

Automotive SPICE [7] is an adaption of the ISO/IEC15504 to the specific characteristics of the automotive domain. The current version (3.0), from July 2015, was developed by an initiative in the German Association of Automotive Industry, which includes the most influencing OEMs and automotive suppliers. Additionally, the ”Hersteller Initiative Software” (HIS) of the manufacturers
Audi, BMW Group, Daimler, Porsche and VW defined the so-called HIS-scope, which comprises the minimal set of processes for assessments. It includes all engineering aspects from systems (SYS.2-5) to software engineering (SWE.1-6) and further management and supporting aspects, such as Project Management (MAN.3), Quality Assurance (SUP.1), Configuration Management (SUP.8), and Change Request Management (SUP.10). The Automotive SPICE Process Assessment Model (PAM) refined each of these processes by base practices and work products. The HIS-scope, extended by Requirements Elicitation (SYS.1), used for our mapping, results in a set of 133 base practices (BPs) and 52 work products (WPs). Since Capability Level 1, dealing with the implementation of the specific base practices and work products, is a prerequisite for the following Capability Levels with the generic goals, we focus only on Capability Level 1 with its 185 elements. Thus, the mapping does not contain generic practices and resources, essential for achieving higher capability levels.

3 AGILE IN A NUTSHELL

Agile development distinguishes between cultural aspects (with core values and principles) on the one hand and technical aspects on the other hand. Within the technical aspects, agile methods like Scrum and XP, as the most common ones [1], are built on a set of predefined agile practices, which are established instructions with a specific focus [8]. Common examples are daily stand-ups, retrospectives, feature-driven development or pair programming. We identified 155 agile practices [8][9]. Furthermore, agile methods are commonly adapted or changed to the specific companies’ context [10]. This is especially the case for embedded systems, since regulations require the adaption of practices in order to ensure compliance to standards.

4 RELATED WORK

The analysis of the current state showed that in general the combination of capability models and agile development is possible. Nonetheless, the majority of comparisons or mappings stays on the level of the processes for the capability models and complete agile methods. This results in generic statements. Only a few cover a very limited set of processes on the levels of base/best practices [3][11][13] which is necessary for an agile implementation in practice. The only known combination of a capability model and agile practices, which does not work with a detailed mapping, is in the medical domain covering MDevSPICE [13].

Considering the field of capability models in general and not looking to some specific models, such as CMMI or SPICE, the majority of authors agree that these models can be fulfilled by (adapting) agile methods without violating the idea of agile development [4][5][6][11]. Only some authors believe that they contradict each other [3]. When getting into detail and having a look at the capability levels of the models the results differ. Within CMMI, Capability Level 2 can be reached easily [3][11][12], but for Capability Level 3 agile development needs more changes [12], “especially in Organizational Process Focus and Decision Analysis and Resolution” [12]. In contrast to this, reaching CMMI Levels 4 and 5 is “practically impossible with XP and Scrum without making changes to the methods that contradict agility” [12]. Considering SPICE, Level 2 can be reached easily as most of the engineering process group is covered by Scrum [4][5], similar to Level 3 “using supporting auxiliary processes” [4].

In several discussions and workshops (e.g. Gate4SPICE-meetings) with consultants and practitioners from different automotive companies (OEMs and suppliers), we identified that the level of granularity and completeness of the existing mappings do not address the practitioners’ perspective and needs.

Because of the following different reasons and use cases within their daily work, different stakeholders, such as engineering process groups / process owners, consultants, and assessors, need an in-depth mapping of agile development and the respective capability models: (1) Implementing Automotive SPICE by using agile, (2) Checking Automotive SPICE BPs and WPs that are supported by implemented agility, (3) Providing possibilities and ideas for process improvement initiatives by integrating agile development to get a better Automotive SPICE compliance, (4) Giving reasons for using specific agile practices, and (5) Make assessment results more objective and comparable.

5 MAPPING APPROACH

With the mapping approach, we are going to combine Automotive SPICE with agile practices to answer which practices support the standard. Our mapping approach contains five steps (Fig. 1).

Figure 1. Process of the overall mapping approach
5.1 Set-Up
In the "set-up" we prepared the input for the following steps: We listed both input sets for the mapping, the 155 agile practices [9] and 185 Automotive SPICE requirements (all BPs and WPs from Automotive SPICE 3.0 HIS-scope processes + Requirements Elicitation), to cover most practitioners’ interests and needs.

5.2 Literature Mapping
Within the mapping approach, existing literature that provided some detailed comparison (at least on process-level), e.g. [11][12][13], was defined as relevant source (Fig. 1, Step 2). Besides literature, we also included web sources for better coverage, e.g. blogs such as sebastian-schneider.eu. Within the systematic search for relevant sources, we checked them for their confidence e.g. based on authors’ experience. The search ended with ten sources that contained mappings that we extracted.

The integration of the extracted literature mappings was straightforward. Either the agile practice was directly mapped to a requirement (BP/WP) or it was only mapped to a process with its candidate list. Afterwards, we mapped these candidates in the expert mapping to the detailed requirements of the specific process. Additionally, most of the literature could not be used directly because they do not cover the latest version of Automotive SPICE. Thus, we built a workaround using an indirect mapping, first from Automotive SPICE to the mentioned source models (mainly CMMI) and then to the mapped practices. This literature step finally included 461 mappings (connections from agile practices to Automotive SPICE requirements).

5.3 Expert Mapping
The integrated mappings based on the literature contained a number of large gaps, e.g. most of the BPs regarding communication and traceability. Thus, we added another step to extend the literature mapping and fill these gaps based on our experience of the usage of all the different agile practices and on our Automotive SPICE experiences (Fig. 1, Step 3).

First, one expert in agile development assigned the practices from the candidate list (from literature) to all requirements which they support in this particular process. For traceability reasons, we kept the initial literature mapping annotated.

Second, the same expert also performed an experience-based check: In one direction, we checked whether for each of the single base practices/work products some of the agile practices support the requirement. Here, the major focus was on base practices/work products that are not covered before to find initial gaps in Automotive SPICE that are not at all supported by agile practices. In the other direction, we independently checked the list of agile practices to find out where they could contribute/support at least to some extent. Here we mainly focused on practices of Scrum or XP (as the commonly used). Finally, and as a consistency check, we compared similar base practices, e.g. all those concerned with establishing traceability, and their respective mapped agile practices with each other. Based on these two different directions and the consistency check, we ensured completeness of the initial mapping.

5.3 Quality Assurance & Validation Procedure
Since we performed all the mapping iteratively, we started with quality assurance (QA) very early. After finishing all the mapping steps for two Automotive SPICE processes (SUP.1 and MAN.3), we discussed this preview results in a workshop with three experts (two related to agile and two to SPICE) and external partners from the automotive industry. Here we asked for feedback regarding the later usage of the results in daily work and whether we address the mentioned use cases.

Although we performed the different systematic procedures and different stages of QA, we are aware of the fact that there are some threats to validity: our structured search for information sources cannot guarantee that we found all relevant literature.

We used a two-fold approach for the quality assurance of the mapping over all the processes of our scope: First, two domain experts independently performed a high-level review by checking whether the different agile practices make sense for the respective processes and work products (Fig. 1, Step 4). We consolidated these data by integrating the mappings where both agreed on adding or removing agile practices to the process or work product. In the case, that only one expert added or deleted an agile practice within his review; we added them to the candidate list and used it for the second quality assurance step. Second, we reviewed each of the processes of our defined scope in detail (Fig. 1, Step 5). This included checking each BP individually and if necessary changing (adding or deleting) the mapped agile practices. In general, two experts performed these detailed reviews consecutively. Differences were discussed and resolved in a final workshop reaching consensus.

Overall, we tried to avoid being subjective by the aforementioned quality assurance steps with a multi-expert agreement. Additionally, the resolution of conflicts between literature and our experience is a possible threat. Within the final review, we discussed them in detail, looked in the literature again, checked its quality (e.g. author and evidence), and finally decided on a majority vote (two experts and literature).

6 RESULTS AND DISCUSSION
Overall, based on the 185 Automotive SPICE requirements (BPs and WPs) and the 155 agile practices (not only from Scrum and from XP), our mapping approach resulted in 772 mappings (461 from literature, 311 from experts) from agile practices to support Automotive SPICE. 612 of these mappings (79%) deal with Automotive SPICE base practices and only 160 cover the work products (21%). However, a large number of mappings or supporting agile practices per Automotive SPICE base practice does not necessarily mean that this base practice is fully achieved. However, it supports the specific base practice at least to some extent, e.g. only addressing a sub-aspect.

From all 185 Automotive SPICE requirements only 12 (6%) are not supported to at least some extent, 5 from the base practices (4%) and 7 from the work products (13%). From the other point of
view, out of the 155 agile practices in our list we used 103 practices (66%) in the mapping; 33 of all the 38 Scrum or XP practices (89%) show a better usage. The most commonly mapped agile practices and thus, the ones with the highest contribution within our results are "backlog" (n=61), "taskboard" (n=37), "product owner" (n=30), "definition-of-done", "user-stories", and "planning meeting" (each with n=25).

### 6.1 Automotive SPICE: Work Products

We considered the work products as one set and not as part of the different individual processes, because some of them are important in more than one process. In general there were only 7 work products that were not supported by agile practices from our point of view: "software item" (which is the integrated software), "recovery plan", "risk management plan", "risk mitigation plan", "communication record", "quality record", and "evaluation report". The number of mappings for the remaining 45 work products varies significantly, from only one supporting agile practice up to ten. The ones with the highest number of supporting practices are "stakeholder requirements" (n=10), "progress status record" (n=5), "test specification" (n=7), "project plan" (n=6), "SW architectural design" (n=6), "test result" (n=6), and "system requirements specification" (n=6). The most commonly used agile practices for supporting all work products are the "backlog" (n=14), "configuration management" (n=12), "taskboard" (n=10), and "planning meeting" (n=8). These results show a focus on three aspects of the development life cycle: project management, requirements and testing aspects.

The overall numbers show that all processes are supported to some extent, from only 17 mappings up to 104. The processes with the highest number of mappings are:

- Project Management (MAN.3, n=104),
- Software Requirements Analysis (SWE.1, n=46),
- Requirements Elicitation (SYS.1, n=44),
- SW Detailed Design and Unit Construction (SWE.3, n=40),
- Quality Assurance (SUP.1, n=39)
- System Requirements Analysis (SYS.2, n=39).

In contrast, there are processes with a small number of mappings:

- Configuration Management (SUP.8, n=17),
- Software Qualification Test (SWE.6, n=23),
- System Architectural Design (SYS.3, n=23),
- System Qualification Test (SYS.6, n=25).

When taking into account the average number of each base practices in every process (Fig. 2, Ø per BP) and the sole number of agile practices (Fig. 2, #APs) mapped to the processes, the results look quite similar but not the same. Within the first case, agile practices show a better support of Quality Assurance (SUP.1) and Supplier Monitoring (ACQ.4). In the second case with sole numbers of mapped agile practices, Supplier Monitoring (ACQ.4) are better supported. When comparing the number of mappings and number of mapped agile practices, it is interesting that there are two processes, ACQ.4 and SWE.6, having almost a ratio of 1. The complete opposite is MAN.3, which has a rate of using each practice on average three times. This implies that most practices supporting this process are crosscutting over several base practices.

Finally, comparing all base practices, independent of their processes, only five are unsupported by agile practices: SYS.2.BP4, SYS.3.BP5, SWE.2.BP5, SWE.3.BP3, and SUP.9.BP1. The other base practices are supported by at least one (18 base practices) up to 19 agile practices. In 1/5 of all base practices, three agile practices are mapped as supporting them. To represent how the specific base practices of each process are supported by agile practices, Fig. 2 presents a heat map of the number of supporting agile practices (darker blue = more agile practices). Even if Project Management (MAN.3) shows the highest supported base practices, there are other base practices that are supported in the same way, e.g. SUP.1.BP2, SYS.1.BP1, SWE.2.BP1, or SWE.3.BP8. Nonetheless, we are aware of the fact that “support” is hard to measure. For example, some practices like taskboard or backlog support base practices on traceability, but tracing thousands of requirements seems to be impossible only with this practice.

### 6.3 Agile Practices

Since all the previous results covered the viewpoint of Automotive SPICE, we now discuss the agile practices in some more details. For this reason, we visualized how often the different agile practices appear in the 17 processes of our scope. Fig. 3 (each line representing an agile practice) shows the most common agile practices from Scrum and XP by their overall occurrence (BPs and WPs) and a distribution over the processes.
Automotive SPICE-compliance in a few areas. This implies that introducing these agile practices only supports one or two requirements, e.g., common workspace or refactoring. Retrospectives. Unfortunately, some agile practices support only spread over some processes, such as Incremental Design or are other practices that we mapped less often, but that are still large coverage of Automotive SPICE with a mature process. There standards. They seem to be the most promising to reach a fast and process, such as backlog, product owner or templates and. Nonetheless, they need to be enhanced later on by other useful (agile or traditional) practices, e.g., for architectural aspects. Future work will concentrate on how to use the mapping in different use cases. Our main goal is to use these results defining a roadmap for introducing agile development in automotive:

1. Defining “Must Haves” of agile practices as the most important and supporting ones for Automotive SPICE.
2. Identifying gaps where agile practices do not support Automotive SPICE. Comparing these gaps with the common industry practice (not only agile).
3. Creating concepts to fill remaining practical gaps with respect to the characteristics and requirements of automotive.
4. Defining a guideline or standard for agile development in automotive (software) industry.
5. Defining a new automotive specific agile method by defining extensions that address the gaps.

REFERENCES
White-Box Prediction of Process Performance Indicators via Flow Analysis

Ilya Verenich∗
Queensland University of Technology
Brisbane, Australia
ilya.verenich@qut.edu.au

Hoang Nguyen
Queensland University of Technology
Brisbane, Australia
huanghuy.nguyen@hdr.qut.edu.au

Marcello La Rosa
Queensland University of Technology
Brisbane, Australia
m.larosa@qut.edu.au

Marlon Dumas†
University of Tartu
Tartu, Estonia
marlon.dumas@ut.ee

ABSTRACT
Predictive business process monitoring methods exploit historical process execution logs to provide predictions about running instances of a process, which enable process workers and managers to preempt performance issues or compliance violations. A number of approaches have been proposed to predict quantitative process performance indicators, such as remaining cycle time, cost, or probability of deadline violation. However, these approaches adopt a black-box approach, insofar as they predict a single scalar value without decomposing this prediction into more elementary components. In this paper, we propose a white-box approach to predict performance indicators of running process instances. The key idea is to first predict the performance indicator at the level of activities, and then to aggregate these predictions at the level of a process instance by means of flow analysis techniques. The paper specifically develops this idea in the context of predicting the remaining cycle time of ongoing process instances. The proposed approach has been evaluated on four real-life event logs and compared against several baselines.

CCS CONCEPTS
•Information systems → Information systems applications; Decision support systems;

KEYWORDS
Process Mining, Predictive Process Monitoring, Flow analysis

1 INTRODUCTION
Predictive business process monitoring techniques seek to determine the future state or properties of ongoing process instances based on models extracted from historical event logs. A wide range of predictive monitoring techniques have been proposed to predict, for example compliance violations [13, 14], the next activity or the remaining sequence of activities of a process instance [8, 23], or quantitative process performance indicators, such as the remaining cycle time of a process instance [18, 19, 22]. These predictions can be used to alert process workers to problematic process instances or to support resource allocation decisions, e.g. to allocate additional resources to instances that are at risk of a deadline violation.

This paper addresses the problem of predicting quantitative process performance indicators, with a specific focus on predicting the remaining cycle time of ongoing process instances. Existing approaches to this problem adopt a “black-box” approach by building stochastic models or regression models which, given a process instance, predict the remaining execution time as a single scalar value, without seeking to explain this prediction in terms of more elementary components. Yet, quantitative performance indicators such as cost or time are aggregations of corresponding performance indicators of the activities composing the process. In particular, the cycle time of a process instance consists of the sum of the cycle time of the activities performed in that process instance. In this respect, existing techniques allow us to predict the aggregate value of a performance indicator for a running process instance, but they do not explain how each activity contributes to this aggregate prediction.

Motivated by this observation, this paper proposes a “white-box” approach to predicting quantitative performance indicators of running process instances based on a general technique for quantitative process analysis known as flow analysis. The idea of flow analysis is to estimate a quantitative performance indicator at the level of a process by aggregating the estimated values of this performance indicator at the level of the activities in the process, taking into account the control-flow relations between these activities. Accordingly, in order to predict the remaining cycle time of a process instance, we propose to first estimate the cycle time of each activity
that might potentially be executed within this process instance, and then to aggregate these estimates using flow analysis.

In addition to providing predictions that can be traced down to the level of individual activities, we show via an empirical evaluation with real-life business process event logs, that the proposed technique achieves comparable and sometimes higher prediction accuracy relative to several state-of-the-art “black-box” baselines.

The remainder of the paper is structured as follows. Section 2 presents the related work on process prediction, with an emphasis on the prediction of remaining time. Section 3 introduces the important concepts and notations used in the paper. Section 4 outlines the details of the proposed approach. Next, Section 5 presents an experimental evaluation of our approach and compares it with the baseline techniques. Finally, Section 6 concludes the paper and outlines future work directions.

2 RELATED WORK
A wide range of predictive business process monitoring problems have been studied in previous work, including the prediction of delays and deadline violations, remaining cycle time, outcome, and future events of a running case.

The problem of predicting delays and deadline violations in business processes has been addressed by different authors. Pika et al. [16] propose a technique for predicting deadline violations by identifying process risk indicators that cause the possibility of a delay. Metzger et al. [15] present techniques for predicting “late show” events (i.e. delays between the expected and the actual time of arrival) in a freight transportation process by finding correlations between “late show” events and external variables related to weather conditions or road traffic. Finally, Senderovich et al. [20] apply queue mining techniques to predict delays in case executions.

Another group of works address the prediction of the remaining cycle time of running cases. Van Dongen et al. predict the remaining time by fitting non-parametric regression models based on the frequencies of activities within each case, their average durations, and case attributes [24]. Van der Aalst et al. [22] propose a remaining time prediction method by constructing a transition system from the event log using set, bag, or sequence abstractions of observed events. Polato et al. [17] refine this method by proposing a data-aware transition system annotated with classifiers and regressors. Rogge-Solti and Weske [18, 19] model business processes as stochastic Petri nets and perform Monte Carlo simulation to predict the remaining time of a process instance. De Leoni et al. [5, 6] propose a general framework to predict various characteristics of running instances, including the remaining time, based on correlations with other characteristics and using decision and regression trees. The remaining time prediction problem has also been extensively studied in the context of software development processes. For example, Kikas et al. [10] predict issue resolution time in Github projects using static, dynamic and contextual features. In this paper, we show that the remaining cycle time of a process instance can be decomposed into a sum of the cycle times of the activities that are yet to be performed in that process instance. Thus, estimating cycle times of individual activities, we can estimate the entire remaining time of a case.

Another category of techniques aim to predict the outcome of running cases. For example, Maggi et al. [14], propose a framework to predict the outcome of a case (normal vs. deviant) based on the sequence of activities executed in a given case and the values of data attributes of the last executed activity in a case. This latter framework constructs a classifier on-the-fly (e.g. a decision tree or random forest) based on historical cases that are similar to the (incomplete) trace of a running case. Other approaches construct a collection of classifiers offline. For example, [13] construct one classifier for every possible prediction point (e.g. predicting the outcome after the first event, the second one and so on). Conforti et al [4] apply a multi-classifier (decision trees) at each decision point of the process, to predict the likelihood of various types of risks, such as cost overruns and deadline violations.

A final group of techniques aim to predict future event(s) of a running case. Lakshmanan et al. [12] use Markov chains to estimate the probability of future execution of a given task in a running case; Breuker et al. [3] use probabilistic finite automata to predict the next activity to be performed while Tax et al [21] predict the entire continuation of a running case as well as timestamps of future events using long short-term memory (LSTM) neural networks.

In this paper, we do not address the problems of case outcome prediction and future events prediction, although our approach could in principle be extended in these directions.

3 BACKGROUND
In this section, we introduce concepts used in later sections of this paper.

3.1 Event Logs, Traces and Sequences
For a given set A, A* denotes the set of all sequences over A and σ = ⟨a1, a2, . . . , an⟩ a sequence of length n; ┐ is the empty sequence and σ1 · σ2 is the concatenation of sequences σ1 and σ2. ℎd(σ) = ⟨a1, a2, . . . , an⟩ is the prefix of length k (0 < k < n) of sequence σ and ℎd(σ) = ⟨a1, a2, . . . , ak⟩ is its suffix. For example, for a sequence σ1 = ⟨a, b, c, d, e⟩, ℎd(σ1) = ⟨a, b⟩ and ℎd(σ1) = ⟨c, d, e⟩.

Let E be the event universe, i.e., the set of all possible event identifiers, and T the time domain. We assume that events are characterized by various properties. One of these properties is the timestamp of an event1, meaning that there is a function πT ∈ E → T that assigns timestamps to events. Other properties of an event include its activity, resource performing the event, etc.

Definition 3.1 (Trace). A trace is a finite non-empty sequence of events σ ∈ E* such that each event appears only once and time is non-decreasing, i.e., for 1 ≤ i < j ≤ |σ| : σ(i) ≠ σ(j) and πT(σ(i)) ≤ πT(σ(j)). A trace in a log represents the execution of one case.

Definition 3.2 (Event log). An event log is a set of events, each linked to a particular trace and globally unique, i.e., the same event cannot occur twice in a log.

3.2 Flow Analysis
Flow analysis is a family of techniques that enables estimation of the overall performance of a process given knowledge about the

1Hereinafter, we refer to the event completion timestamp unless otherwise noted.
performance of its activities. For example, using flow analysis one can calculate the average cycle time of an entire process if the average cycle time of each activity is known. Flow analysis can also be used to calculate the average cost of a process instance knowing the cost-per-execution of each activity, or calculate the error rate of a process given the error rate of each activity [7]. The main advantage of the flow analysis is that the estimation can be easily explained in terms of its elementary components.

**Definition 3.3 (Cycle time of an activity).** A cycle time of an activity $i$ is the average time it takes between the moment the activity is ready to be executed and the moment it completes. By “ready to be executed” we mean that all activities upon which the activity in question depends have completed. Formally, cycle time is the difference between the timestamp of the activity and the timestamp of the previous activity. i.e. $\pi_T(\sigma(i)) - \pi_T(\sigma(i-1))$ for $1 \leq i \leq |\sigma|$. Here, $\pi_T(\sigma(0))$ denotes the start time of the case.

The cycle time of an activity includes the processing time of the activity, as well as all waiting time prior to the execution of the activity. Processing time refers to the time that actors spend doing actual work. On the other hand, waiting time is the portion of the cycle time where no work is being done to advance the process. This may include time spent in transferring information about the case between process participants, for example when documents are exchanged by post, as well as time when the case is waiting for an actor to process it. In many processes, the waiting time makes up a considerable proportion of the overall cycle time. This situation may, for example, happen when the work is performed in batches. In a process related to the approval of purchase requisitions at a company, the supervisor responsible for such approvals in a business unit might choose to batch all applications and check them only once at the start or the end of a working day [7].

To understand how flow analysis works, we start with an example of a process with sequential fragments of events as in Figure 1a. Each fragment has a single entry flow and a single exit flow and has a cycle time $T_i$. Since the fragments are performed one after the other, we can intuitively conclude that the cycle time $CT$ of a purely sequential process with $N$ event fragments is the sum of the cycle times of each fragment [7]:

$$CT = \sum_{i=1}^{N} T_i \quad (1)$$

Let us consider a process model with a decision point between $N$ mutually exclusive fragments, represented by an XOR gateway (Figure 1b). In this case, the cycle time of a process model is

$$CT = \sum_{i=1}^{N} p_i \cdot T_i, \quad (2)$$

where $p_i$ denote the branching probabilities, i.e. frequencies with which a given branch $i$ of a decision gateway is taken.

In case of parallel, or AND gateways where activities can be executed concurrently as in Figure 1c, the combined cycle time of multiple fragments is determined by the slowest of the fragments, that is:

$$CT = \max_{i=1,...,\pi} T_i \quad (3)$$

Another recurrent pattern is the one where a fragment of a process may be repeated multiple times, for instance because of a failed quality control. This situation is called rework and is illustrated in Figure 1d. The fragment is executed once. Next, it might be repeated each time with a probability $r$ referred to as rework probability. The average number of times that the rework fragment is expected to be executed can be obtained via the geometric series [7], and the cycle time of the fragment in this case is:

$$CT = \frac{T}{1 - r} \quad (4)$$

**Figure 1: Typical process model patterns: sequential (a), XOR-block (b), AND-block (c) and rework loop (d).**

Besides cycle time, flow analysis can also be used to calculate other performance measures. For instance, assuming we know the average cost of each activity, we can calculate the cost of a process more or less in the same way as we calculate cycle time. In particular, the cost of a sequence of activities is the sum of the costs of these activities. The only difference between calculating cycle time and calculating cost relates to the treatment of AND-blocs. The cost of an AND-block such as the one shown in Figure 1c is not the maximum of the cost of the branches of the AND-block. Instead, the cost of such a block is the sum of the costs of the branches. This is because after the AND-split is traversed, every branch in the AND join is executed and therefore the costs of these branches add up to one another [7].
In case of block-structured process models that can be represented as a sequence of event fragments with a single entry and a single exit, we can relate each fragment to one of the four described types and use the aforementioned equations to estimate the required performance measure. However, in case of an unstructured process model or if a model contains other modeling constructs besides AND and XOR gateways, the method for calculating performance measures becomes more complicated.

A major limitation of flow analysis is that it does not consider the fact that a process behaves differently depending on the load, i.e. the number of process instances that are running concurrently. For example, the cycle time of a process for handling insurance claims would be much slower if the insurance company was handling thousands of claims at once, due for example to a recent natural disaster as compared to the case where the load is low and the company may be handling only a hundred claims at once. When the load increases and the number of process workers remains constant, the waiting times tend to increase. This phenomenon is referred to as resource contention. It occurs when there is more work to be done than resources available to perform the work. In such scenarios, some tasks will be in waiting mode until a required resource is freed up. Flow analysis does not take into account the effects of increased resource contention. Instead, the estimates obtained from flow analysis are only applicable if the level of resource contention is relatively stable over the long term.

4 APPROACH
In this section, we describe the proposed approach to predict the remaining time. We first provide an overview of the entire solution framework and then focus on the key parts of our approach.

4.1 Overview
Our approach exploits historical execution traces in order to discover a structured process model. Once the model has been discovered, we identify its set of activities and decision points and train two families of machine learning models: one to predict the cycle time of each activity, and the other to predict the branching probabilities of each decision point. To speed up the performance at runtime, these steps are performed offline (Figure 2).

At runtime, given an ongoing process instance, we align its partial trace with the discovered process model to determine the current state of the instance. Next, we traverse the process tree obtained from the model starting from the state up to the process end and deduce a formula for remaining time using rules described in Section 3.2. The formula includes cycle times of activities and branching probabilities of decision points that are reachable from the current execution state. These components are predicted using previously trained regression and classification models. Finally, we evaluate the formula and obtain the expected value of the remaining cycle time.

4.2 Discovering Process Models from Event Logs
The proposed approach relies on a process model as input. However, since the model is not always known or might not conform to the real process, generally we need to discover the model from event logs. For that, we use a two-step automated process discovery technique proposed in [2] that has been shown to outperform traditional approaches with respect to a range of accuracy and complexity measures. The technique has been implemented as a standalone tool as well as a ProM plugin, namely StructuredMiner.

The technique in [2] pursues a two-phase "discover and structure" approach. In the first phase, a model is discovered from the log using a heuristic process discovery method that has been shown to consistently produce accurate, but potentially unstructured or even unsound models. In the second phase, the discovered model is transformed into a sound and structured model by applying two techniques: a technique to maximally block-structure an acyclic process model and an extended version of a technique for block-structuring flowcharts. This approach has been shown to outperform traditional "discover structured" approaches with respect to a range of accuracy and complexity measures.

A structured model is internally represented as a process tree. A process tree is a tree where each leaf is labeled with an activity and each internal node is labeled with a control-flow operator: sequence, exclusive choice, non-exclusive choice, parallelism, or iteration.

4.3 Replaying Partial Traces on the Process Model
For a given partial trace, to predict its remaining time, we need to determine the current state of the trace relative to the process model. For that, we map, or align, a trace to the process model using the technique described in [1] which is available as a plugin for the open-source process mining platform Apromore.

The technique treats a process model as a graph that is composed of activities as nodes and their order dependencies as arcs. A case replay can be seen as a series of coordinated moves, including those over the model activities and gateways and those over the trace events. In that sense, a case replay is also termed an alignment of a process model and a trace. Ideally, this alignment should result in as many matches between activity labels on the model and event labels in the trace as possible. However, practically, the replay may choose to skip a number of activities or events in search of more matches in later moves. Moves on the model must observe the semantics of the underlying modeling language which is usually expressed by the notion of tokens. For example, for a BPMN model, a move of an incoming token over a XOR split gateway will result in a single token produced on one of the gateway outgoing branches.

\[\text{Available at http://apromore.org/platform/tools}\]
while a move over an AND split gateway will result in a separate token produced on each of the gateway outgoing branches. The set of tokens located on a process model at a point in time is called a marking. On the other hand, a move in the trace is sequential over successive events of the trace ordered by timestamps, one after another. Thus, after every move, either on the model or in the trace, the alignment comes to a state consisting of the current marking of the model and the index of the current event in the trace.

In [1], cases are replayed using a heuristics-based backtracking algorithm that searches for the best alignment between the model and a partial trace. The algorithm can be illustrated by a traversal of a process tree starting from the root node, e.g. using depth-first search, where nodes represent partial candidate solution states (Figure 3). Here the state represents the aforementioned alignment state of the case replay. At each node, the algorithm checks whether the alignment state till that node is good enough. If so, it generates a set of child nodes of that node and continues down that path; otherwise, it stops at that node, i.e. it prunes the branch under the node, and backtracks to the parent node to traverse other branches.

Consider, we have a partial trace $hd(\sigma) = \langle A,D,B \rangle$. Replaying this trace on the given model as described in the Section 4.3, we find the current marking to be in the states $B$ and $D$ within the AND-block. Traversing the process model starting from these states until the process end, we obtain the following formula:

$$CT_{rem} = \max(T_B + T_C, T_D) + T_F + p_2 \left( T_G + \frac{T_H}{1-r} \right)$$  \hfill (6)

Since the activity $A$ has already been executed, it does not contribute to the remaining cycle time. Thus, it is not a part of the formula. Furthermore, $T_B$ and $T_D$ have been executed, however, since they form one of the terms of the formula whereas $T_C$ is still unknown, they cannot be omitted, but their actual cycle times should be taken. All the other formula terms need to be predicted using the data from $hd(\sigma)$.

Similarly, if a current marking is inside a XOR block, its branching probabilities need not be predicted. Instead, the probability of the branch that has actually been taken is set to 1 while the other probabilities are set to 0.

A more complex situation arises when the current marking is inside the rework loop. In this case, we “unfold” the loop as shown in the Figure 5. Specifically, we separate the already executed occurrences of the rework fragment from the potential future occurrences and take the former out of the loop. Let us consider a partial trace $hd(\sigma) = \langle A,D,B,C,F,G,H \rangle$. Since $H$ has occurred once, according to the process model (Figure 4), with a probability $r$, it may be repeated, otherwise, the rework loop is exited. To signal this choice, we take the first occurrence of $H$ out of the loop, and place a XOR gateway after it. One of the branches will contain a rework loop of future events with the same probability $r$, while the other one will reflect an option to skip the loop altogether. Thus, the cycle time of the whole fragment can be decomposed as follows:

$$CT_H = T_H^+ + r \frac{T_H}{1-r} \right)$$  \hfill (7)

where $T_H$ refers to the cycle time of already executed occurrence(s) of $H$. It is highlighted in bold font, meaning that we should take the actual cycle time rather than the predicted.

### 4.5 Computing the Remaining Time

We can use the flow analysis formulas produced by the method described in Section 4.4 to compute the remaining cycle time of a case, given: (i) an estimate of the cycle time of each activity reachable from the current execution state; and (ii) an estimate of the branching probability of each flow stemming from a reachable XOR-split (herein called a reachable conditional flow). Given an execution state, these estimates can be obtained in several ways including:

1. By using the prediction models produced for each reachable activity and for each reachable conditional flow, taking into account only traces that reach the current execution state. We herein call this approach predictive flow analysis.
2. By computing the mean cycle time of each reachable activity and the traversal frequency of each reachable conditional flow, again based only on the suffixes of traces that reach the execution state in question. We call this approach mean flow analysis.
a class starting from 0, and the model makes predictions about the probability of each class. The predictive models are trained for prefixes $hd^n(\sigma)$ of all traces $\sigma$ in the training set for $2 \leq k < |\sigma|$. We do not train and make predictions after the first event, since for those prefixes there is no sufficient data available to base the predictions upon.

As an example, let us consider a snapshot of the log with one completed case in Table 1 that corresponds to the process model in Figure 4. The events are ordered according to their completion timestamp.

<table>
<thead>
<tr>
<th>Case ID</th>
<th>Case Attributes</th>
<th>Event Attributes</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Email 37</td>
<td>A 9:13:00</td>
</tr>
<tr>
<td>1</td>
<td>Email 37</td>
<td>B 9:14:20</td>
</tr>
<tr>
<td>1</td>
<td>Email 37</td>
<td>D 9:16:00</td>
</tr>
<tr>
<td>1</td>
<td>Email 37</td>
<td>C 9:18:00</td>
</tr>
<tr>
<td>1</td>
<td>Email 37</td>
<td>F 9:18:10</td>
</tr>
<tr>
<td>1</td>
<td>Email 37</td>
<td>G 9:18:50</td>
</tr>
<tr>
<td>1</td>
<td>Email 37</td>
<td>H 9:19:00</td>
</tr>
</tbody>
</table>

To encode traces as feature vectors, we include both case attributes and event attributes. Thus, the first case in the log will be encoded as such:

$$\bar{X} = (\text{Email, 37; A, B, D, C, F, G, H}; 9:13:00, R03; 9:14:20, R12; 9:16:00, R07; 9:18:00, R03; 9:18:10, R21; 9:18:50, R12; 9:19:00, R12)$$

Now, to create the training set for $hd^n(\sigma)$, we cut the feature vectors to include the event attributes up to the $k$-th event and case attributes (which are usually known since the beginning of the case). Furthermore, we add the value of the target variable $y$ to be learned. For example, if we are to predict the cycle time of activity $G$ for prefixes $k = 2$, the training sample based on the data extracted from the first case in Table 1 would be created as follows:

$$D^2_G = \{\bar{X}^2, y_G\} = \{\text{Email, 37; A, B}; 9:13:00, R03; 9:14:20, R12; 9:16:00, R07; 9:18:00, R03; 9:18:10, R21; 9:18:50, R12; 9:19:00, R12\}$$

Here 40 is the cycle time of $G$ for the first case, determined as the time difference (in seconds) between the completion timestamp of $G$ and the completion timestamp of the previous activity $F$. It should be noted that for a case that follows the upper branch of the gateway $x21$, the process terminates after $F$, thus $G$ is never executed and its cycle time is undefined. Therefore, we exclude such cases from the training data. Conversely, if an activity occurs multiple times in a case, we take its average cycle time.

The rationale for the mean flow analysis is that the prefix size can have two opposite effects on prediction accuracy. If a prefix is too short, there might not be enough information in it to predict cycle times of some activities and gateways’ branching probabilities, especially those that are executed near the process end. On the other hand, if the prefix is long, for activities and gateways that are usually executed at the beginning of the process, we will not have enough training data to fit the model. As an example, let us consider an activity that, according to the process model, usually occurs in the 4th or 5th position in the process, but in a few cases can occur in the 8th position. Then, to fit a model for a prefix length 5, as training data we can only use these few cases, since for most other cases, the activity will not occur after the 5th event. In cases where the accuracy of the produced predictive models is insufficient, we can then use the mean historical activity cycle times instead.

In order to make use of predictive models, we need to encode process execution traces in the form of feature vectors. In this paper, we use index-based encoding as described in [13] that concatenates the case attributes and, for each position in a trace, the event occurring in that position and the value of each event attribute in that position. This type of trace encoding is lossless and has been shown to achieve a relatively high accuracy and reliability when making early predictions of binary process properties [13, 25].

For each activity in the process model, to predict its cycle time, we train a regression model, while for predicting branching probabilities we fit classification models for each corresponding XOR gateway. In the latter case, each branch of a gateway is assigned
We conducted the experiments using four real-life event datasets. The datasets consisted of three subprocesses: one that tracks the state of the application (BPIC'12 A), one that tracks the state of the offer (BPIC'12 O), and a third one tracks the states of work items associated with the application (BPIC'12 W). For the latter subprocess, we retain only events of type complete. The fourth dataset is based on the log that contains events from a ticketing management process of the help desk of an Italian software company. Each case starts with the insertion of a new ticket into the ticketing management system and ends when the issue is resolved and the ticket is closed.

As mentioned in Section 3.2, flow analysis technique cannot readily deal with unstructured models. Even though the tool described in Section 4.2 aims to mine maximally structured models, it does not always succeed in doing so. Specifically, it sometimes produces models with overlapping loops which our current implementation is unable to deal with. One solution to this problem could be to simplify the process model by removing the transitions that cause overlapping loops. However, this may severely decrease the accuracy of the discovered model, which will, in turn, negatively affect the accuracy of the flow analysis-based predictions of remaining time. Hence, instead, we remove the cases that cause overlapping loops from the event log (up to 15% of cases in each log).

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Number of cases</th>
<th>Mean activities per case</th>
<th>Mean variants per case</th>
<th>Mean duration, days</th>
</tr>
</thead>
<tbody>
<tr>
<td>BPIC'12 A</td>
<td>12,007</td>
<td>10</td>
<td>10</td>
<td>4.49</td>
</tr>
<tr>
<td>BPIC'12 O</td>
<td>3,487</td>
<td>7</td>
<td>6</td>
<td>4.56</td>
</tr>
<tr>
<td>BPIC'12 W</td>
<td>9,650</td>
<td>6</td>
<td>2,263</td>
<td>7.50</td>
</tr>
<tr>
<td>helpdesk</td>
<td>3,218</td>
<td>5</td>
<td>8</td>
<td>3.30</td>
</tr>
</tbody>
</table>

5.2 Experimental Setup
To assess the quality of the prediction of continuous variables, well-known error metrics are Mean Absolute Error (MAE), Root Mean Square Error (RMSE) and Mean Percentage Error (MAPE) [9], where MAE is defined as the arithmetic mean of the prediction errors, RMSE – as the square root of the squared prediction errors, while MAPE measures error as the average of the unsigned percentage error. We observe that the value of remaining time tends to be highly varying across cases, with values at different orders of magnitude. RMSE would be very sensitive to such outliers. Furthermore, the remaining time can be very close to zero, especially near the end of the trace, thus MAPE is skewed in such situations. Hence, we use MAE to measure the error in predicting the remaining time.

We employ several baselines to compare our approach to. Firstly, we use a transition system (TS) based method proposed by van der Aalst et al. [22] applying both set, bag and sequence abstractions. Secondly, we use a method proposed by Leontjeva et al. [13] who compared several types of business process sequence encodings for prediction of the boolean case outcome. This method can be naturally adjusted to predict the remaining time by replacing the classification task with the regression task. For the purpose of this paper, we will reproduce only two types of the original encodings – index-based and frequency-based, as the others were shown to have either very similar or inferior performance. Next, we evaluate against the stochastic Petri-net (SPN) based approach proposed by Rogge-Solti and Weske [18, 19]. Specifically, we use the method based on the constrained Petri net, as it was shown to have the lowest prediction error. However, their original approach makes predictions at fixed time points, regardless of the arriving events. To make the results comparable to our approach, we modify the method to make predictions after each arrived event. Finally, we used a combined estimator along the lines of [24] where the feature set includes the frequencies of activities within each case, their average durations, and case attributes.

In our experiments, we order the cases in the logs based on the time at which the first event of each case has occurred. Then, we split the logs into two parts. We use the first part (2/3 of the cases) as a training set, i.e. as historical data to train the predictive models. The remaining 1/3 of the cases are used to evaluate the accuracy of the predictions. Furthermore, we perform a five-fold cross-validation on the training set in order to select the optimal values of the training parameters such as the number of trees and the number of variables at each split for a random forest model.
5.3 Results
Table 3 summarizes the performance of the predictive and mean flow analysis techniques, as well as the baselines approaches for each dataset. We make predictions for prefixes $hd^k(\sigma)$ of traces $\sigma$ in the test set starting from $k = 2$. However, since for very long prefixes, there are not enough traces with that length, and the error measurements become unreliable, we stop the predictions after $k$ reaches the 70th-percentile length of the traces in the log, i.e. at least 70% of the traces in the log have a length smaller than $k$. Thus, since the BPIC’12 W log contains longer traces, the prefix sizes evaluated are higher for this log. Additionally, we report the average performance across all prefixes, weighted over the relative frequency of traces with that prefix (i.e. longer prefixes get lower weights, since not all traces reach that length).

We observe that for most logs, the prediction accuracy of flow analysis-based techniques is at least as good as that of the baselines. At the same time, for all logs except BPIC’12 O, mean flow analysis, on average, provides the best results among all the methods. Specifically, it outperforms the predictive flow analysis. The latter is due to the lack of data attributes in the event logs that would be able to accurately explain the variation in the cycle times of individual activities and branching probabilities of each conditional flow. To further investigate this issue, for each activity in the BPIC’12 A and BPIC’12 O logs, we analyze the performance of regressors trained to predict its cycle time and compare it with a constant regressor used in the mean flow analysis. In Table 4 we report MAE of cycle times for each activity and each technique, as evaluated on the test set. Since for each prefix length we have a separate regressor, we report weighted average values, as in Table 3. In addition, we report the actual average cycle time values of each predicted activity based on the test set.

As can be seen from Table 4, in the BPIC’12 O log, prediction-based cycle times are more accurate than the constant ones for longer activities which make up the largest portion of the remaining cycle time. Furthermore, the difference between the two approaches is higher for BPIC’12 O. Hence, for this log, we can estimate the remaining time more accurately with the predictive flow analysis.

Another observation is related to the very low accuracy of the predictive flow analysis on the BPIC’12 W log. Having closely inspected this log, we found that it contains sequences of two or more events in a row of the same activity. In other words, activities are frequently reworked multiple times. As mentioned in Section 3.2, flow analysis techniques assume a constant rework probability $r$. However, in many real-life processes $r$ subsequently decreases after each execution of the rework loop, meaning that the rework becomes less and less likely. Thus, if $r$ is inaccurately predicted in predictive flow analysis, this error propagates further. To verify our hypothesis, we modify the log keeping only the first occurrence of each repeated event in a sequence. To keep the remaining time calculations correct, we retain the last event of a case, even if it is a repeated event. Having run the experiments on the modified log (Table 5), we notice that predictive flow analysis becomes almost as accurate as mean flow analysis, thus proving our hypothesis.

Summing up, the experiments suggest that flow analysis-based techniques provide relatively accurate estimations of the remaining cycle time across all logs. Thus, we can positively answer RQ1.

Our experiments also show that flow analysis-based techniques are able to provide relatively accurate predictions starting from the early stages of an ongoing case. The general trend is a stable reduction in MAE values as a case progresses. This is due to the increasing amount of attributes in the prefix to base the predictions upon. Furthermore, the actual remaining times intuitively decreases at later stages of a case, thus its prediction error also decreases. We can then provide a positive answer to RQ2.

Execution Times. The execution time of the proposed approach is composed of the execution times of the following components: (i) training the predictive models; (ii) replaying the partial traces on the process model (finding an alignment) and deriving the formulas; (iii) applying the models to predict the cycle times and branching probabilities and calculating the overall remaining time. For realtime prediction, it is crucial to output the results faster than the mean case arrival rate. Thus, we also measured the average runtime overhead of our approach. All experiments were conducted on a laptop with a 2.4 GHz Intel Core i5 processor and 8 Gb of RAM.

For a given prefix length $k$, training all the models takes between 20 and 200 seconds depending on the prefix size and the number of models to train. Replaying the test traces takes between 5 and 45 seconds, for a given length of the prefix. Finally, making the predictions takes less than 4 seconds per prefix length. This shows that our approach performs within reasonable bounds for most online applications.

5.4 Threats to Validity
The datasets used in this evaluation, except for the BPIC’12 W, have only the completion timestamps, but not the start timestamps. Thus, it is impossible to discern the actual processing time from the waiting time. The latter can have a significant impact on the overall cycle time depending on the case arrival rate and the resource load. As these factors are not accounted for in the predictive models, their accuracy is rather low.

We reported the results with a single learning algorithm (random forest). With decision trees and gradient boosting, we obtained qualitatively the same results, relatively to the baselines. However, our approach is independent of the learning algorithm used. Thus, using a different algorithm does not in principle invalidate the results. That said, we acknowledge that the goodness of fit, as in any machine learning problem, depends on the particular classifier/regressor algorithm employed. Hence, it is important to test multiple algorithms for a given dataset, and to apply hyperparameter tuning, in order to choose the most adequate algorithm with the best configuration.

The proposed approach relies on the accuracy of the branching probability estimates provided by the classification model. It is known however that the likelihood probabilities produced by classification methods are not always reliable. Methods for estimating the reliability of such likelihood probabilities have been proposed in the machine learning literature [11]. A possible enhancement of the proposed approach would be to integrate heuristics that take into account such reliability estimates.
### Table 3: MAE values (in days) for prefixes of different lengths.

<table>
<thead>
<tr>
<th>Method</th>
<th>Prefix length</th>
<th>Avg</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>BPIC’12 A</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Predictive flow analysis</td>
<td></td>
<td>9.48</td>
<td>9.60</td>
<td>10.38</td>
<td>9.68</td>
<td>7.04</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Mean flow analysis</td>
<td></td>
<td>8.32</td>
<td><strong>7.89</strong></td>
<td>9.62</td>
<td>8.81</td>
<td><strong>6.88</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TS set abstraction [22]</td>
<td></td>
<td>9.16</td>
<td>8.39</td>
<td>10.53</td>
<td>10.31</td>
<td>8.02</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TS bag abstraction [22]</td>
<td></td>
<td>9.16</td>
<td>8.39</td>
<td>10.53</td>
<td>10.31</td>
<td>8.02</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TS sequence abstraction [22]</td>
<td></td>
<td>9.16</td>
<td>8.39</td>
<td>10.53</td>
<td>10.31</td>
<td>8.02</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Frequency-based encoding [13]</td>
<td></td>
<td>9.16</td>
<td>8.40</td>
<td>10.52</td>
<td>10.28</td>
<td>8.02</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Constrained SPN [19]</td>
<td></td>
<td>8.44</td>
<td>9.15</td>
<td><strong>8.47</strong></td>
<td><strong>7.41</strong></td>
<td>6.89</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Combined estimator [24]</td>
<td></td>
<td>9.05</td>
<td>8.19</td>
<td>10.48</td>
<td>10.32</td>
<td>8.03</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>BPIC’12 O</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Predictive flow analysis</td>
<td></td>
<td>5.96</td>
<td>7.46</td>
<td><strong>6.40</strong></td>
<td>2.55</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Mean flow analysis</td>
<td></td>
<td>6.33</td>
<td>8.00</td>
<td>6.81</td>
<td>2.53</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TS set abstraction [22]</td>
<td></td>
<td>6.05</td>
<td>8.03</td>
<td>6.81</td>
<td>2.54</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TS bag abstraction [22]</td>
<td></td>
<td>6.05</td>
<td>8.03</td>
<td>6.81</td>
<td>2.54</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TS sequence abstraction [22]</td>
<td></td>
<td>6.05</td>
<td>8.03</td>
<td>6.81</td>
<td>2.54</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Index-based encoding [13]</td>
<td></td>
<td>6.36</td>
<td>8.06</td>
<td>6.82</td>
<td>2.52</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Frequency-based encoding [13]</td>
<td></td>
<td>6.33</td>
<td>8.02</td>
<td>6.81</td>
<td>2.54</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Constrained SPN [19]</td>
<td></td>
<td>5.49</td>
<td><strong>6.46</strong></td>
<td>6.46</td>
<td><strong>2.42</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Combined estimator [24]</td>
<td></td>
<td>6.34</td>
<td>8.04</td>
<td>6.80</td>
<td>2.52</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>BPIC’12 W</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Predictive flow analysis</td>
<td></td>
<td>14.48</td>
<td>15.38</td>
<td>15.33</td>
<td>15.83</td>
<td>14.32</td>
<td>16.08</td>
<td>11.62</td>
<td>12.52</td>
<td>13.67</td>
<td>12.21</td>
</tr>
<tr>
<td>Mean flow analysis</td>
<td></td>
<td>7.35</td>
<td><strong>8.58</strong></td>
<td><strong>8.01</strong></td>
<td><strong>7.49</strong></td>
<td><strong>7.20</strong></td>
<td>6.87</td>
<td>6.70</td>
<td>6.61</td>
<td>6.36</td>
<td>6.21</td>
</tr>
<tr>
<td>TS set abstraction [22]</td>
<td></td>
<td>7.99</td>
<td>9.04</td>
<td>8.70</td>
<td>8.20</td>
<td>7.93</td>
<td>7.50</td>
<td>7.34</td>
<td>7.35</td>
<td>6.94</td>
<td>6.75</td>
</tr>
<tr>
<td>TS bag abstraction [22]</td>
<td></td>
<td>7.95</td>
<td>8.84</td>
<td>8.71</td>
<td>8.22</td>
<td>7.95</td>
<td>7.42</td>
<td>7.26</td>
<td>7.27</td>
<td>6.93</td>
<td>6.83</td>
</tr>
<tr>
<td>TS sequence abstraction [22]</td>
<td></td>
<td>7.91</td>
<td>8.84</td>
<td>8.70</td>
<td>8.22</td>
<td>7.91</td>
<td>7.40</td>
<td>7.21</td>
<td>7.22</td>
<td>6.84</td>
<td>6.74</td>
</tr>
<tr>
<td>Index-based encoding [13]</td>
<td></td>
<td>7.64</td>
<td>8.71</td>
<td>8.29</td>
<td>7.86</td>
<td>7.50</td>
<td>7.24</td>
<td>7.02</td>
<td>6.95</td>
<td>6.69</td>
<td>6.53</td>
</tr>
<tr>
<td>Combined estimator [24]</td>
<td></td>
<td>7.66</td>
<td>8.74</td>
<td>8.30</td>
<td>7.91</td>
<td>7.39</td>
<td>7.28</td>
<td>7.02</td>
<td>6.93</td>
<td>6.64</td>
<td>6.43</td>
</tr>
<tr>
<td><strong>Helpdesk</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Predictive flow analysis</td>
<td></td>
<td>5.97</td>
<td>5.24</td>
<td>9.36</td>
<td>2.76</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Mean flow analysis</td>
<td></td>
<td>5.27</td>
<td><strong>5.10</strong></td>
<td>6.10</td>
<td>3.28</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TS set abstraction [22]</td>
<td></td>
<td>5.52</td>
<td>5.44</td>
<td><strong>5.92</strong></td>
<td>5.14</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TS bag abstraction [22]</td>
<td></td>
<td>5.59</td>
<td>5.49</td>
<td>6.15</td>
<td>3.08</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TS sequence abstraction [22]</td>
<td></td>
<td>5.59</td>
<td>5.49</td>
<td>6.15</td>
<td>3.08</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Index-based encoding [13]</td>
<td></td>
<td>5.58</td>
<td>5.39</td>
<td>6.54</td>
<td>3.26</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Frequency-based encoding [13]</td>
<td></td>
<td>5.61</td>
<td>5.50</td>
<td>6.17</td>
<td>3.28</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Constrained SPN [19]</td>
<td></td>
<td>5.54</td>
<td>5.34</td>
<td>6.53</td>
<td>4.29</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Combined estimator [24]</td>
<td></td>
<td>5.54</td>
<td>5.39</td>
<td>6.34</td>
<td>3.27</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

## 6 CONCLUSION AND FUTURE WORK

The paper has put forward some potential benefits of a “white-box” approach to predicting quantitative process performance indicators. Rather than predicting single scalar indicators, we demonstrated how these indicators can be estimated as aggregations of corresponding performance indicators of the activities composing the process. In this way, the predicted indicators become more explainable, as they are decomposed into elementary components. Thus, business analysts can pinpoint the bottlenecks in the process execution and provide better recommendations to keep the process compliant with the performance standards.

We implemented and evaluated two approaches – one where the formulas’ components are predicted from the trace prefix based on the models trained on historical completed traces, and the other that instead uses constant values obtained from the historical averages of similar traces. We evaluated the approaches to predict the remaining cycle time, as one of common process performance indicators. The empirical evaluation has shown that the proposed techniques are, on average, able to yield more accurate predictions at different stages of running cases than the surveyed baselines.

We identified a limitation of flow analysis-based approaches when dealing with traces with rework loops, i.e. multiple occurrences of the same fragment of activities in a row. A direction for future work is to further investigate the factors affecting the performance of the proposed approaches in order to better understand their strength and weaknesses. Furthermore, we plan to extend...
Table 4: MAE of cycle time predictions of individual activities and their actual mean cycle times (in days).

<table>
<thead>
<tr>
<th>Activity</th>
<th>Predictive MAE</th>
<th>Predictive Mean</th>
<th>Predictive Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>A_CANCELLED</td>
<td>11.97</td>
<td>12.02</td>
<td>14.36</td>
</tr>
<tr>
<td>A_APPROVED</td>
<td>7.61</td>
<td>7.51</td>
<td>7.36</td>
</tr>
<tr>
<td>A_DECLINED</td>
<td>3.72</td>
<td>3.74</td>
<td>3.74</td>
</tr>
<tr>
<td>A_REGISTERED</td>
<td>5.92</td>
<td>5.96</td>
<td>3.70</td>
</tr>
<tr>
<td>A_ACTIVATED</td>
<td>4.46</td>
<td>4.47</td>
<td>2.88</td>
</tr>
<tr>
<td>A_ACCEPTED</td>
<td>0.43</td>
<td>0.78</td>
<td>0.76</td>
</tr>
<tr>
<td>A_PREACCEPTED</td>
<td>0.04</td>
<td>0.13</td>
<td>0.09</td>
</tr>
<tr>
<td>A_FINALIZED</td>
<td>0.01</td>
<td>0.01</td>
<td>0.01</td>
</tr>
<tr>
<td>BPIC’12 A</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BPIC’12 O</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>O_CANCELLED</td>
<td>8.68</td>
<td>9.75</td>
<td>18.20</td>
</tr>
<tr>
<td>O_SENT_BACK</td>
<td>2.79</td>
<td>4.01</td>
<td>9.42</td>
</tr>
<tr>
<td>O_ACCEPTED</td>
<td>2.60</td>
<td>2.59</td>
<td>4.22</td>
</tr>
<tr>
<td>O_DECLINED</td>
<td>2.50</td>
<td>2.43</td>
<td>3.54</td>
</tr>
<tr>
<td>O_SENT</td>
<td>&lt; 0.01</td>
<td>&lt; 0.01</td>
<td>&lt; 0.01</td>
</tr>
</tbody>
</table>

Table 5: MAE values (in days) for prefixes of different lengths for the modified BPIC’12 W log with excluded event duplicates.

<table>
<thead>
<tr>
<th>Method</th>
<th>Prefix length</th>
<th>Avg</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Predictive flow analysis</td>
<td></td>
<td>6.70</td>
<td>8.22</td>
<td>5.86</td>
<td>4.45</td>
<td>4.27</td>
</tr>
<tr>
<td>Mean flow analysis</td>
<td></td>
<td>6.15</td>
<td>7.69</td>
<td>5.14</td>
<td>3.88</td>
<td>4.14</td>
</tr>
<tr>
<td>TS Set abstraction</td>
<td></td>
<td>6.70</td>
<td>8.40</td>
<td>5.82</td>
<td>3.94</td>
<td>4.27</td>
</tr>
<tr>
<td>TS Bag abstraction</td>
<td></td>
<td>6.69</td>
<td>8.40</td>
<td>5.82</td>
<td>4.04</td>
<td>3.97</td>
</tr>
<tr>
<td>TS Sequence abstraction</td>
<td></td>
<td>6.69</td>
<td>8.40</td>
<td>5.82</td>
<td>4.04</td>
<td>3.97</td>
</tr>
<tr>
<td>Index-based encoding</td>
<td></td>
<td>6.54</td>
<td>8.14</td>
<td>5.66</td>
<td>4.15</td>
<td>4.04</td>
</tr>
<tr>
<td>Freq-based encoding</td>
<td></td>
<td>6.71</td>
<td>8.44</td>
<td>5.83</td>
<td>4.03</td>
<td>4.00</td>
</tr>
<tr>
<td>Constrained SPN</td>
<td></td>
<td>7.82</td>
<td>7.76</td>
<td>8.14</td>
<td>7.70</td>
<td>7.46</td>
</tr>
<tr>
<td>Combined estimator</td>
<td></td>
<td>6.50</td>
<td>8.12</td>
<td>5.59</td>
<td>4.09</td>
<td>3.99</td>
</tr>
</tbody>
</table>

the proposed approaches so that they would be able to deal with more complex models with overlapping loops, using structuring techniques such as the one proposed in [26].

With some modifications in the derivation of the flow analysis formulas, the proposed approaches can be extended to predict other quantitative performance indicators. In future work, we aim to extend and evaluate the approaches to predict the process cost or error rate.

ACKNOWLEDGMENTS

This research is funded by the Australian Research Council under Grant No.: DP150103356 and the Estonian Research Council under Grant No.: IUT20-55.

REFERENCES


Domain Modeling for Development Process Simulation

Ian J. De Silva
Dept. of Comp. Sci. and Engr.
University of Minnesota, USA
desilva@cs.umn.edu

Sanjai Rayadurgam
Dept. of Comp. Sci. and Engr.
University of Minnesota, USA
rsanjai@cs.umn.edu

Mats P. E. Heimdahl
Dept. of Comp. Sci. and Engr.
University of Minnesota, USA
heimdahl@cs.umn.edu

ABSTRACT
Simulating agile processes prior to adoption can reduce the risk of enacting an ill-fitting process. Agent-based simulation is well-suited to capture the individual decision-making valued in agile. Yet, agile’s lightweight nature creates simulation difficulties as agents must fill-in gaps within the specified process. Deliberative agents can do this given a suitable planning domain model. However, no such model, nor guidance for creating one, currently exists.

In this work, we propose a means for constructing an agile planning domain model suitable for agent-based simulation. As such, the domain model must ensure that all activity sequences derived from the model are executable by a software agent. We prescribe iterative elaboration and decomposition of an existing process to generate successive internally-complete and -consistent domain models, thereby ensuring plans derived from the model are valid. We then demonstrate how to generate a domain model and exemplify its use in planning the actions of a single agent.

CCS CONCEPTS
•Computing methodologies → Modeling methodologies; Agent / discrete models; Planning and scheduling; •Software and its engineering → Agile software development;

KEYWORDS
Domain Modeling, Software Development Process Evaluation

ACM Reference format:
DOI: 10.1145/3084100.3084111

1 INTRODUCTION
Software development processes do not ensure project success, yet adopting an ill-fitting process may hinder the team’s ability to complete the project on-time, on-budget, and with the required functionality. Prior to adoption, we want to evaluate candidate processes, particularly agile processes, to ensure they satisfy the project goals. Agent-based simulation is well-suited to such evaluation [3].

Because of their people-focused, lightweight nature, agile processes present unique difficulties for simulation. Rather than specifying all actions, lightweight processes specify some actions or quality criteria and allow individuals to supplement the process (choosing additional steps that both satisfy the criteria and reach the goal). Existing simulations cannot replicate this behavior, and, instead, rely on full process specification [3]. We desire a simulation that can model lightweight processes and capture the impact of individual decisions on a project and its outcomes.

Deliberative software agents1 can model process supplementation by composing plans (sequences of process activities) expected to reach a goal state, then selecting and executing the most desirable (and process compliant) plan. Execution stops when the agent reaches the goal state or the plan becomes impractical—wherein it forms a new plan. A planner module within the agent forms these plans over a domain model (the set of activities the agent can perform and their sequencing constraints). Unfortunately, there are no existing planning domain models nor guidance for creating them in the literature. The objective of this paper, is to provide guidance for creating a planning domain model for agent-based simulation.

What are the properties of a suitable planning domain model? For agents, planning is performed in situ, and planning failure results in simulation failure, thereby wasting valuable analysis time. These failures occur if no valid plan can be constructed in the planning domain. To prevent this, we desire a planning domain model where we cannot construct an invalid plan if the planner cooperates in proactively ensuring plan validity. A plan, which is essentially a narrowly-defined process, is valid if it is both internally-complete (all activity dependencies are present in the plan) and -consistent (for each plan activity, all activities it depends on precede it in the plan).2 Thus, with a cooperative planner, invalid plans are only generated when the domain model contains internal incompleteness or inconsistency. We, therefore, desire a planning domain model that is both internally-complete and -consistent.

As a step towards multi-agent, agile-process simulation, we prescribe a method for constructing an internally-complete and -consistent (ICC), single-agent3 planning domain model using iterative elaboration. We show that, given an initial ICC domain model and ICC process fragments, each elaboration produces an ICC domain model. Further, we demonstrate this approach by constructing an example planning domain model and use it to generate goal-reaching plans without a specified process.

1 A deliberative agent has an “explicitly represented, symbolic model of the world, and in which decisions (for example, about which actions to perform) are made via [logical] reasoning, based on pattern matching and symbolic manipulation” [14].
2 There are other situation-independent process quality criteria [6, 7]. However, they address process desirability rather than validity, and are unrelated to plan formation.
3 We assume the agent has perfect knowledge (full world observability).
2 RELATED WORK

Constructing Processes. Situational method engineering is a means for constructing situation-specific, fully-specified processes (methods) from process fragments stored in a repository. A process may be generated using method assembly, method configuration, or paradigm-based approaches [7]. Method assembly composes fragments from a repository according to the requirements/goals they satisfy [10]. Method configuration, including process tailoring, transforms an existing process specification by adding, removing, or elaborating it, according to guides/patterns, using fragments from a repository [7]. Paradigm-based construction generates/uses a meta-model that it transforms to meet the situational requirements then instantiates with fragments from a repository [10]. While all of these approaches generate situation-specific processes, they rely on the repository data’s quality. If needed fragments are missing, it may not be possible to construct a valid process.

Situational method engineering aims to generate good processes for human actors to follow. While useful as guidance, this is insufficient for approximating human behaviors as humans are not well-behaved [9]. Humans, due to external factors, perform redundant activities or compose activities in unusual ways. We want to capture realistic behavior as we expect it will improve a simulation’s predictive qualities. Rather than describing all acceptable ways in which an agent may deviate from the ideal, which may miss realistic behaviors, we want to permit all valid behaviors and allow the agent to reason over them.

Structured Process Transformation. Supporting process construction, Chroust introduces a model calculus to express process transformations [2], yet he does not show that these operations result in valid process models. Lee and Wyner define formal semantics for extending (specializing and refining) existing processes captured as data flow diagrams [8]. However, their approach is flow-preserving. Rather than adhering to fixed, idealized flows, we want agents to use any valid, process-constraint-complying activity sequencing to achieve the goal.

3 PLANNING DOMAIN MODEL CREATION

We describe an approach for generating a planning domain model and show that following this method ensures the result is ICC. We also provide guidance for specifying achievable intermediate goals despite possible activity output non-determinism within the model.

3.1 Constructing the Domain Model

A process (or fragment) is a sequence of activities (data transforms) executed by an actor to some end. These activities consume and produce data objects (artifacts and resources), which make up the agent’s world model at a given point. Data objects may be related to each other (e.g., by composition). Further, processes contain initial and goal states that are satisfied by one or more worlds.

Software development can be expressed as a process with a single activity: transforming a problem statement into software that addresses the problem. However, this lacks an operational description. We wish to iteratively elaborate this, our initial domain model, until we have enough detail to express all processes under consideration and provide alternative actions to the agent. Existing process repositories—literature and fragment repositories (e.g., [1, 96])—contain a wealth of method information that can aid our elaboration. Using these as fragment sources, we have summarized our method in Listing 1.

Listing 1: Our approach expressed algorithmically.

```
while(not shouldStop(domainModel)):
    (activity, fragment) = locateReplacement(domainModel)
    fragment = generalize(fragment)
    domainModel = replace(domainModel, activity, fragment)
```

We begin by selecting an activity to replace within the domain model and identifying a fragment from the repository that preserves the dependencies in the domain model; specifically, a fragment whose initial state is a subset of the activity’s inputs and whose goal state is a superset of the activity’s outputs. Further, the fragment must not contain an activity that removes data as part of its transformation.4

Next, we transform the fragment; generalizing it by removing all constraints (e.g., control flow constraints) except those inherent to its constituent activities (their data dependencies). As we will show, this allows agents to combine activities in ways that are valid, but may not have been captured in the process repository. We then replace the previously selected activity with the generalized fragment; resulting in a new ICC domain model.

We repeat this process until we have a model that can express all of the processes that we wish to evaluate using simulation and we have enough detail to provide alternative actions to the agent that are not specified in the process.

3.2 Ensuring Domain Model Suitability

We wish to construct a domain model such that, given an initial ICC domain model and a set of ICC process specifications, the iterative elaboration of the domain model will also be ICC. We will show that the generalize and replace transforms preserve the process specifications’ and domain model’s ICC properties.

3.2.1 Definitions. A process (or fragment) specification is a set of activities and sequencing constraints. Similarly, a domain model is a collection of activities, sequenced according to their dependencies, with initial and goal states to define/limit its scope. For model simplicity, we assume activities non-destructively consume data objects. Further, we omit input/output object cardinality from the planning domain model and leave it to activity execution.

In the planning domain, an execution is a non-empty sequence of activities. An execution is valid if it is ICC. A valid execution is goal-reaching (or domain-model-goal-reaching) if it connects the initial and goal states.

A domain model is internally complete if, for each activity in the domain model, its inputs are generated by another activity within the model. A domain model is internally consistent if, for a given activity, there is a sequence of activities from the initial state that provide the inputs required by the activity. Thus, a domain model (or, correspondingly, a process specification) is ICC if it is composed of a set of activities and sequencing rules such that the sequencing

4Rather than removing data objects, we model removal of artifacts as changes to scope information kept by the agent.
rules together with the activities make up a non-empty set of goal-reaching executions and every activity within the model lies on at least one execution.

To simplify our discussion, assume the domain model includes an activity with no inputs to produce the initial state \( v_{\text{start}} \) and another that consumes the goal state with no outputs \( v_{\text{end}} \). If more than one goal state exists, define \( v_{\text{end},1}, \ldots, v_{\text{end},m} \) such that these nodes generate a token artifact (indicating the goal state has been reached) that is consumed by \( v_{\text{end}} \). Do a similar thing if there are multiple initial states. Thus, without loss of generality, assume there is one \( v_{\text{start}} \) and one \( v_{\text{end}} \).

3.2.2 Generalization. For a given ICC process fragment specification, \( P \), with a set of activities \( V_{p} \), we generalize \( P \) (call it \( G(P) \)) by removing all constraints except the innate dependencies of its constituent activities (the data dependencies). Here, we show that the result of this transform is an ICC generalized fragment.

Let \( P(V_{p}) \) be the set of all executions (valid or not) over the activities in \( P \) and including \( V_{\text{start}} \) and \( V_{\text{end}} \) where each of the sequences begins at \( V_{\text{start}} \) and terminates at \( V_{\text{end}} \). We produce \( K(V_{p}) \) by removing all executions from \( P(V_{p}) \) where the dependencies of an activity do not precede it. Thus \( K(V_{p}) \) is the set of all executions within \( G(P) \).

To show that \( G(P) \) is ICC, we must show that \( K(V_{p}) \) is non-empty; \( K(V_{p}) \) contains only valid, goal-reaching executions; and every activity in \( V_{p} \) is in some execution in \( K(V_{p}) \).

Let \( e(P) \) be the set of all goal-reaching executions in \( P \). Because \( P \) is ICC and by the construction of \( K(V_{p}) \), we know that \( e(P) \subseteq K(V_{p}) \), \( e(P) \) is non-empty, and \( e(P) \) contains all activities in \( V_{p} \). Further, because \( K(V_{p}) \subseteq P(V_{p}) \), \( K(V_{p}) \) contains no activities besides those in \( V_{p} \). Thus, \( K(V_{p}) \) is non-empty and every activity in \( V_{p} \) is in some execution in \( K(V_{p}) \).

In the construction of \( K(V_{p}) \), we removed all executions from \( P(V_{p}) \) where any activity’s dependencies do not precede it in the execution. Thus, each execution in \( K(V_{p}) \) is ICC. Further, since each execution in \( P(V_{p}) \) begins and terminates at \( v_{\text{start}} \) and \( v_{\text{end}} \) respectively, each execution is a valid, goal-reaching execution.

Since \( K(V_{p}) \) is a non-empty set containing all valid, goal-reaching executions within the generalized fragment, \( G(P) \), and every activity is contained in at least one execution, we know \( G(P) \) is ICC.

3.2.3 Replacement. Generalization leaves us with both an ICC domain model, \( D \), and a generalized, ICC process fragment, \( G(P) \). We wish to compose them into a new domain model.

Previously, we selected a process fragment, \( P \), such that, for some activity \( a \in D \), the fragment’s initial state is a subset of the inputs of \( a \) and the fragment’s goal state is a superset of the outputs of \( a \). By the construction of \( G(P) \), \( P \), and \( G(P) \) have the same initial and goal states. As \( G(P) \) is ICC, its inputs are provided by \( a \)’s dependencies, and its outputs satisfy \( a \)’s dependents, \( G(P) \) can replace \( a \in D \) and the result is a new ICC domain model, \( D' \).

3.3 Additional Properties for Simulation

Simply being able to generate plans from an ICC domain model is not enough for simulation. In this section, we show both that agents can replan on-the-fly using this model, and that, with guidance, modelers can specify other achievable goals.

3.3.1 Replannability. Thus far, we have ignored output non-determinism: an activity may produce one of multiple output sets upon execution (e.g., a test-run may emit a success message or fail, providing an error report). This has no bearing on the ICC of the model; however, it does trigger replanning. We want to ensure that the agent can still generate a valid plan when starting from the current world (replannability).

Assume that we have an execution, \( e \), beginning at \( v_{\text{start}} \), that led us to the current, non-goal world. Because artifacts are never removed from the world, once we execute an activity, all of its outputs are available from that point on. Thus, we must supplement \( e \) so that it reaches \( v_{\text{end}} \). Because the domain model is ICC, there is at least one goal-reaching execution, \( e \), in the model. By removing all completed activities in \( e \) (call it \( f \)) and appending \( e \) to \( f \), we know the result will be a goal-reaching execution as it complies with the dependency constraints and connects \( v_{\text{start}} \) and \( v_{\text{end}} \). The sequence, \( f \), is the agent’s new plan to reach the goal.

Thus, from any world reached by performing a valid sequence of activities, we can create a plan from that world to the goal state.

3.3.2 Intermediate Goal Planning. By replannability, we can reach the domain model goal state from any reachable world. However, not all teams want to reach that goal. We’d like to define other, intermediate goals and plan to them.

Intermediate goals (IGs) are worlds in which desired data objects exist. When all activities are deterministic, an IG is expressible as a non-unique set of activities that produce the IG state. Using the same technique used to show replannability, we can generate a plan to reach each activity comprising the IG. Since activity output may be non-deterministic during plan generation, we can express IGs only over those data objects we could arrive at deterministically. This severely limits our model. Is there a way for us to treat non-deterministic activities as deterministic for planning purposes?

Activities that produce both expected and exceptional output sets result in output non-determinism. Exceptional activity output in the planning model stem from expected exceptions during activity execution (e.g., a defect in compiled code), or imperfect world knowledge (e.g., an artifact that is unexpectedly missing). By our earlier assumptions—that we are simulating a single agent with perfect knowledge—the latter cannot be true; thus exceptional output must stem from errors.

To ensure IG reachability, we want to prevent our IG state from including any artifact that can only be reached through an unexpected output of a non-deterministic activity. As illustrated by the run tests of activities of test-driven development (TDD; Figure 1), the expected output depends on the world’s state (e.g. tests pass if the increment’s code is present). To deal with this sort of non-determinism, some planners specify policies (over a control-flow-based planning domain) directing the planner to select a specific action when in a given state [5]. We could similarly guide the planner by specifying expected output based on the current state. In the TDD example, such a policy would expect test failure if the implementation is not present and test success if it is. This makes planning deterministic, and allows us to include additional data objects in our definition of an IG state.

Several forms of workflow faults exists [11]; however, in terms of expected activity output, only these apply.
4 THEORY IN PRACTICE: SIMULATING TDD

Having presented and supported an approach for constructing a domain model, we use our proposed approach to generate a domain model and show, in a simple simulation, that

(1) the domain model can be used to generate valid plans and
(2) agents can reach the goal even with a lightweight process specification.

4.1 Constructing a Planning Domain Model

Having defined an approach for generating a domain model, we want to put it into practice, applying it to create a domain model suitable for expressing test-driven development (TDD; Figure 1).

Figure 1: TDD – Detailed Control Flow [12]

Initial Domain Model. In order to simulate, we require a domain model. Here, we outline the steps needed to create one suitable for representing TDD.

Begin with an initial domain model: software development. Of the numerous possible elaborations, we select the V-model [13] since TDD is focused on testing. We expand implementation using IEEE Standard 1074 [1] and introduce activities for declaring, implementing, and integrating interfaces as prescribed by TDD for testing [12]. Spillner et al. [13] provides a general testing process that elaborates each of the testing activities. To support TDD’s test automation, we replace test coding with the implementation process from earlier and test execution with TDD’s definition [12]. Finally, we introduce refactoring according to Fields et al.’s description [4]. Listing 2 shows a subset of our domain model.

Model Validation. Before simulating, we want to verify the generated model can express TDD. To check this, we mapped the different portions of TDD to our generated model (Table 1).

One benefit of our approach is that we can generate activity sequences that were not considered during the construction of the domain model. For example, the activities in the generated domain model can be easily used to specify test last development (TLD).

Listing 2: Domain Model Subset – Activities are defined as <activityName>: <preconditions> -> <postconditions>, where capitalized terms are variables to unify. Activities starting with trans split data objects to constituent parts.

technicalSysDesign: artifact(requirement) -> artifact(sysArch), artifact(componentDefinition), artifact(componentIntegrationModel)
trans_ArchToComp: artifact(sysArch) -> artifact(componentDefinition), artifact(componentIntegrationModel)
testPlanning: artifact(requirement) -> artifact(testDesign)
specifyComponent: artifact(componentDefinition) -> artifact(componentImpl), artifact(incrementRequirement)
declareInterface: artifact(incrementRequirement), artifact(componentDesign) -> artifact(incrementInterface)
implementInterface: artifact(incrementRequirement), artifact(componentDesign) -> artifact(incrementInterface)
integrateIncrements: artifact(incrementImpl), artifact(componentImpl) -> artifact(sysImpl)
implementComponents: artifact(sysImpl), artifact(componentImpl) -> artifact(sysImpl)
compileSys: artifact(sysImpl), artifact(impl) -> artifact(compilationSys)

specifyUnitTests: artifact(testBasis), artifact(incrementInterface), artifact(testDesign) -> artifact(logicalUnitTest)
genConcreteUnitTests: artifact(logicalUnitTest) -> artifact(concreteUnitTest)
implUnitTest: artifact(incrementInterface), artifact(concreteUnitTest), artifact(testingTool) -> artifact(automatedUnitTestScript)
executeTests: artifact(automatedUnitTestScript) -> artifact(testSuiteImpl)
compileTests: artifact(testSuiteImpl), artifact(testCompilationSys) -> artifact(compiledTestSuite)
executeTests: artifact(compiledTestSuite) -> artifact(testExecutionResult)

Table 1: Mapping from TDD to the Domain Model Subset

<table>
<thead>
<tr>
<th>TDD Activity</th>
<th>Domain Model Activity</th>
</tr>
</thead>
<tbody>
<tr>
<td>Define Behavior</td>
<td>specifyComponent</td>
</tr>
<tr>
<td>Divide into Increments</td>
<td>specifyComponent</td>
</tr>
<tr>
<td>Design &amp; Declare Interfaces</td>
<td>declareInterface</td>
</tr>
<tr>
<td>Code Unit Test</td>
<td>implementInterface</td>
</tr>
<tr>
<td>Run Tests</td>
<td>executeTests</td>
</tr>
<tr>
<td>Refactor Code</td>
<td>implIncrement</td>
</tr>
</tbody>
</table>

4.2 Simulating

Having constructed an ICC domain model, we demonstrate its use with a simple simulation.

4.2.1 Simulator Set-up. To model an actor working to develop and unit-test a feature, we wrote a single-agent simulator. In it the agent deliberates—forming a plan—and executes the planned activities, simply producing the plan-predicted activity output.

Deliberation. Our planner generates all possible plans from the data model by forward-chaining activities to a fixed depth (here, three). Plans are rank-sorted according to a utility function and the plan in the first position (the highest utility) is selected, even if
there is a tie. On plan completion, the agent replans. This repeats until the agent reaches the goal state.

Further simplifying the planner, we omitted rework and non-deterministic activity outputs from the domain model. We will address them in future work.

**Utility.** Utility functions help the agent determine plan desirability. To capture process adherence, we biased behavior towards quickly completing orderings (pairs that represent activity partial orderings; i.e., succession), with the following utility function:

\[
U(a) = \frac{2}{3} W(a) + \frac{1}{3} \max(0, U(\text{successor}(a)))
\]

where \(a\) is the current activity in the plan, \(\text{successor}(a)\) is the current activity’s immediate successor in the plan, and \(W(a)\) is the activity’s weight based on initiating or completing an ordering (arriving at a node on the left or right side of a pair, respectively); defined as:

\[
W(a) = \begin{cases} 
1, & \text{if } a \text{ completes an ordering} \\
0.5 & \text{if } a \text{ initiates an ordering but does not complete one} \\
0 & \text{if } a \text{ does not initiate or complete an ordering}
\end{cases}
\]

To evaluate that an agent can reach the goal without full process specification, we defined a constant utility function and ran the agent with no specified process. Under these conditions, the agent should choose plans at random until it reaches the goal.

4.2.2 Results. We ran the simulation three times with the same goal: once each for no process (constant utility), fully-specified TDD, and fully-specified TLD. The resultant executions (Listing 3) indeed show that the agent was able to use the domain model to plan and execute TDD and TLD. Further, it generated a valid plan to reach the goal even without a specified process.

As this is part of ongoing work, we are laboring towards a larger example that incorporates these concerns and extends to multi-agent simulations.

## 5 Conclusion and Future Work

In this work, we presented a means for constructing a planning domain model; an important step towards simulating agile processes for a priori evaluation. We showed that iteratively elaborating a domain model preserves its internal-completeness and -consistency, ensuring plan validity when plans are generated by a cooperative planner. We then used the prescribed approach to construct an example domain model able to express test-driven development, and showed an agent can use the model to both adhere to a fully-specified process and achieve a goal absent a specified process.

Our approach generates domain models for an agent with perfect knowledge. As part of our on-going work, we expect to enhance the domain model to support multi-agent planning, paying particular attention to difficulties introduced by scaling (both in number of agents and domain model size) and partial world observability (imperfect knowledge). To this end, we are exploring a means to scale using intermediate goals to support hierarchical planning.

Our domain modeling approach lays the foundation for agent deliberation of software processes; providing a means for both generating plans without full process specification and for simulating agile processes.

## REFERENCES


---

**Listing 3: Simulations by Utility Function**

(a) **Constant-value**

- defineRequirements
- technicalSysDesign
- testPlanning
- trans_ArchToComp
- specifyComponent
- declareInterface
- compileSys
- implIncrement
- specifyUnTests
- integrateComponents
- genConcreteUnTests
- implUnTests
- integrateTests
- compileSys
- executeTests

(b) **Weighted: TDD**

- defineRequirements
- technicalSysDesign
- testPlanning
- trans_ArchToComp
- specifyComponent
- declareInterface
- compileSys
- specUnitTests
- testPlanning
- genConcreteUnTests
- implUnTests
- integrateTests
- compileSys
- executeTests

(c) **Weighted: TLD**

- defineRequirements
- technicalSysDesign
- trans_ArchToComp
- specifyComponent
- declareInterface
- compileSys
- specUnitTests
- intImplement
- integrateIncrements
- compileSys
- specUnTests
- testPlanning
- genConcreteUnTests
- implUnTests
- integrateTests
- compileSys
- executeTests

---

**Threats to Validity.** Even though we were able to demonstrate our method’s use in planning and simulation, this controlled experiment was done with a simple example, leaving out much of the complexity discussed earlier (specifically the non-determinism and rework).
Leveraging Business Process Improvement with Natural Language Processing and Organizational Semantic Knowledge

Deniz Iren
Vrije Universiteit Amsterdam, Business Informatics Group
De Boelelaan 1105
Amsterdam, the Netherlands 1181HV
y.d.iren@vu.nl

Hajo A. Reijers
Vrije Universiteit Amsterdam, Business Informatics Group
De Boelelaan 1105
Amsterdam, the Netherlands 1181HV
h.a.reijers@vu.nl

ABSTRACT
Contemporary organizations need to adapt their business processes swiftly to cope with ever-changing requirements. Requirement changes originate from a wide variety of sources. Business analysts gather these requirements, resolve conflicts, analyze impacts, and prepare actionable improvement plans. These tasks require a comprehensive knowledge of business processes and other entities within the organization. Business process model repositories, which may contain hundreds of models, are important sources of such cross-functional information. In this study, we introduce an approach which facilitates business process improvement by utilizing the comprehensive information covered by process models. Specifically, we associate requirements with other organizational entities based on their transitive relations with process models. To infer these associations, our approach makes use of natural language processing techniques and enterprise semantics. A quantitative evaluation of our approach, which took place within a major telecommunication company, displayed that it accurately detects associations between requirements and process models. Furthermore, semi-structured interviews with business analysts revealed that their expectations are high on efficiency increases due to the usage of this approach.

CCS CONCEPTS
•Computing methodologies → Natural language processing;
•Applied computing → Business process management;

KEYWORDS
Business process improvement, process model repository, requirements management, enterprise semantics

1 INTRODUCTION
Business Process Improvement (BPI) is not only necessary for organizations to gain a competitive advantage but also to survive. In contemporary business settings, organizations constantly face the challenge of rapidly changing business requirements [15]. Organizations need to redesign and improve their business processes continually to cope with this volatility [22]. Three challenges can be distinguished that are associated with requirements management in BPI initiatives.

The first challenge is related to the elicitation of requirements. The requirements may originate from a variety of stakeholders in a multitude of forms [7, 29]. Conflicts among requirements usually occur due to the contradicting interests of different stakeholders [16]. Conflict discovery and resolution are key problems in requirements engineering [18, 33]. The second challenge is to identify the potentially impacted entities among an extensive set of organizational assets. A seemingly isolated requirement may, in fact, have an impact on a large number of processes, IT systems, and business units within an organization [10]. Performing a comprehensive impact analysis requires the identification of all such organizational entities. Existing research mostly focus on defining the guidelines for impact analysis rather than providing mechanisms for the detection of impacted entities [6, 30]. The third challenge is the complexity associated with the task of deriving an actionable BPI plan [1]. Business analysts use the results of the impact analysis in decision making for the scope definition of the BPI initiative [24]. BPI plans need to be created in an actionable way because, generally, the BPI plans are executed by professionals other than business analysts. However, current BPI frameworks do not provide sufficient guidance for developing actionable plans [5].

Business process models play a central role in organizations by capturing essential process information such as the activities conducted, the actors performing these activities, the IT systems utilized, the business rules in play, and the decision points which affect the flow of these activities [17]. Employing information which traverses the organizational structure makes process models good candidates to act as pivotal entities in BPI initiatives [8, 12]. Many organizations develop process model repositories, which extensively capture organizational knowledge [31]. In this paper, we introduce an approach that provides business analysts with analytical capabilities to perform conflict resolution, impact analysis, and actionable BPI planning for the elicited requirements. Our approach differs from existing research by defining a concrete mechanism to relate requirements with organizational entities, based on business processes. We use and integrate multiple Natural Language Processing (NLP) techniques to categorize the requirements,
detect potential conflicts among them, and identify impacted organizational units, and organizational units. We further incorporate enterprise semantics to improve these matching techniques.

We applied our approach in a major telecommunication company, which has 650 process models within its repository and over 200 business requirement statements. All of these were gathered from multiple stakeholders. We quantitatively evaluated the accuracy of our technique by comparing the outcome of the technique against a gold standard set, which is manually prepared by local business analysts. Furthermore, we conducted semi-structured interviews with these analysts to evaluate the potential increase in efficiency achieved by using the proposed approach. Our findings suggest significant improvements in efficiency as well as a notable decrease of errors.

This paper is organized as follows. In Section 2, we provide the background of this study and introduce a running example. Section 3 covers a detailed description of our solution approach. In Section 4, we present our results and evaluate them. Section 5 provides the related research. Finally, in Section 6 we provide a discussion of the results and conclude this paper.

2 BACKGROUND
In this section, we provide background information for this study. Section 2.1 highlights the problem while describing the common BPI stages. In Section 2.2, we give a running example to clarify the problem and exemplify the operation of our approach.

2.1 BPI Stages and Highlighting the Problems
Although there is a consensus on the lack of guiding principles for BPI in the literature [32], and the practice of BPI varies in different organizations, we acknowledge that certain stages are characteristic to most BPI initiatives [24]. In this subsection, we summarize these common BPI stages and highlight the challenges associated with them.

2.1.1 Requirements Elicitation and Conflict Resolution. The first phase in BPI is the elicitation of business requirements. Business analysts gather a set of requirements from a wide variety of sources, such as employees in different departments, customers, and other external stakeholders. Each stakeholder has specific goals and a unique perspective. Moreover, elicitation from different parties is usually done separately. As a result, the set of collected requirements may include duplicates and conflicting statements. Business analysts identify these conflicts among a large set of requirements.

2.1.2 Impact Analysis. After the requirements elicitation and conflict resolution, business analysts conduct an impact analysis to identify the organizational entities that are potentially affected by the new requirement set. Then, such analysts typically try to assess the magnitude of the impact on the identified organizational entities. Impact analysis requires comprehensive knowledge regarding the organizational entities such as business processes, organization units, information assets, and IT systems. Due to its complexity, impact analysis is an effort-intensive and error-prone task.

2.1.3 Actionable BPI Planning. To set the boundaries of the BPI initiative, the responsible business analysts would define the scope.

Table 1: An example set of requirements

<table>
<thead>
<tr>
<th>ID</th>
<th>Requirement</th>
<th>Owner</th>
</tr>
</thead>
<tbody>
<tr>
<td>R1</td>
<td>Customers shall be able to select multiple promotional packages for subscription.</td>
<td>Marketing Department</td>
</tr>
<tr>
<td>R2</td>
<td>StudentPack promotional package subscribers cannot subscribe to Senior package at the same time.</td>
<td>Legal Department</td>
</tr>
<tr>
<td>R3</td>
<td>Easy-CRM must display customers’ contact information and the history of promotional packages subscribed, to the sales representative.</td>
<td>Customer Relations Department</td>
</tr>
</tbody>
</table>

Table 2: An example set of enterprise semantic local context

<table>
<thead>
<tr>
<th>ID</th>
<th>Entity</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>E1</td>
<td>Easy-CRM</td>
<td>IT System</td>
</tr>
<tr>
<td>E2</td>
<td>TelePortal</td>
<td>IT System</td>
</tr>
<tr>
<td>E3</td>
<td>Online P.O.S</td>
<td>IT System</td>
</tr>
<tr>
<td>E4</td>
<td>StudentPack</td>
<td>Product</td>
</tr>
<tr>
<td>E5</td>
<td>Sales Representative</td>
<td>Organizational Role</td>
</tr>
</tbody>
</table>

Along with other stakeholders, business analysts will examine the outcome of the impact analysis, define which requirements will be included in the scope, and prioritize those requirements. Then, business analysts define the steps to prepare an actionable plan for the BPI. This plan includes the as-is and the to-be depictions of the processes, highlighting the change, the impact, and the steps to execute.

2.2 A Running Example
In this subsection, we initiate a running example to clarify the problem we address in this work and the solution we propose to address it. The example is set within a large organization that is conducting a BPI project covering many organizational units and business processes.

Figure 1 gives a layout of the business entities which are directly related to this study (i.e. stakeholders, requirements, process models, and organizational entities). A large number of requirements are elicited from customers, external stakeholders, and employees in various departments. Table 1 lists a small subset of these requirement statements. It should be noted that in a realistic setting there may be hundreds of such requirements.

The organization also has a process model repository in place, which contains many process models. For the sake of simplicity, for this example, only one of these models is displayed in Figure 2. This figure depicts standard process model constructs of the Subscription Process. Process tasks are represented in a flow in the middle lane. This lane covers the tasks which are carried out by the process actor Customer. Two IT systems are associated to the tasks; namely TelePortal and Online P.O.S.

Additionally, Table 2 displays a catalogue of organizational entities and their types, which we refer to as the semantic local context. In Section 3 we will use this example while describing the solution approach step by step.

3 APPROACH
In this section, we describe an approach to facilitate the BPI stages outlined in Section 2 by providing business analysts with analytical capabilities and aggregated information.
The TelePortal checks the eligibility of the customer.

The TelePortal displays the list of promotional packages. The customer selects a package.

Figure 1: A layout of the related business entities

Figure 2: A process model example

Business process models play a key role in our approach due to their overarching nature, i.e., they employ cross-functional information and are associated with many types of organizational entities. Thus, in our approach, we infer the relationship among certain organizational entities from their transitive relations over business process models, as displayed in Figure 3. In other words, we establish the existence of a relationship between a requirement and an IT system, if the requirement is associated with a process in which the IT system is utilized. For example, although the requirements $R_1$ and $R_2$ in Table 1 do not directly address the TelePortal IT system in Figure 2, we infer a potential relationship between them based on their mutual association with the Subscription Process. The following subsections describe the details of our approach.

3.1 Addressing the BPI Problems

Figure 4 highlights how the proposed approach supports the stages of requirements elicitation and conflict resolution, impact analysis, and planning. The left lane depicts the BPI stages, while the right lane shows the steps of our approach. The dashed arrows represent the flow of the information aggregated by our approach. We use the term “VIEW” to represent the aggregated information provided by our approach.

3.1.1 Requirements Elicitation and Conflict Resolution. In this stage, a business analyst identifies conflicting requirements among a large number of statements. This cumbersome task can be facilitated by dividing the entire set of requirements into categories based on commonalities. Our approach assists business analysts by categorizing the requirements based on their similarities to the process models. We use NLP techniques to reveal these similarities. The details of these techniques are described in Section 3.2. These categorized requirements are referred to as VIEW1 in Figure 4. In Figure 5a we present an example visualization of this aggregated information.

Consider again our running example. When a large number of requirements are given as input, based on our approach, these are categorized according to their associations with process models. One of these categories consists of the statements in Table 1 which are associated with the Subscription Process (Figure 2). The business analyst reviews this category and detects the conflict between the requirements $R_1$ and $R_2$. Such conflicts are typical when the requirements are elicited from multiple sources since these may have different goals, viewpoints, and nomenclature.

3.1.2 Impact Analysis. It can be expected that the implementation of the established requirements has an impact on various organizational entities. In our approach, the outcome of the matching task from the previous stage is used. Based on the inferred relations between the requirements and the organizational entities, a list of potentially impacted organizational entities (i.e., IT systems, roles, and units, and information assets) is provided. This aggregated information is referred to as VIEW2 in Figure 4. This view can be visualized as an interactive interface to assist the business analyst in reviewing and evaluating the impact, as shown in Figure 5b.

In our running example, $R_1$, $R_2$, and $R_3$ are mapped to the Subscription Process, which has a role (i.e., customer) and two IT systems (i.e., TelePortal and Online P.O.S.). Our approach infers the relation between the requirement set ($R_1$, $R_2$, and $R_3$) and the organizational entities (Customer, TelePortal, and Online P.O.S.). It will then provide the business analyst with the latter list.

3.1.3 Actionable BPI Planning. The outcome of the impact analysis is used for the planning of the BPI project. Our approach supports this activity by providing information which can be used to highlight the impact on the corporate process map and the organizational architecture. We refer to this aggregated information as VIEW3 in Figure 4. An example visualization is given in Figure 5c. This information is used by the business analyst to analyze the gap between the as-is and to-be processes, define the overall scope, and determine the actionable steps in the BPI plan.

The following subsection describes the details of the NLP procedure which is used for the purpose of matching requirements to processes.

3.2 Matching Procedure

In the conflict resolution and the impact analysis steps described in Section 3.1, our approach requires the matching of requirements with process models. To accomplish this task, we designed an automated solution, which primarily uses NLP techniques and enterprise semantics. Figure 6 depicts the steps of this procedure.
3.2.1 Prepare Inputs. The inputs to this automated solution consist of textual business requirements, process models, and organizational semantic local context. At the initial step, business requirements and process models are transformed into a canonical form; this is a set of natural language sentences. Process model elements (i.e. task descriptions, activity labels, actors, IT systems, and information assets) are extracted from each process model and represented as a set of sentences and phrases. Optionally, at this point, an abbreviation list can be manually provided to handle the variations in expressions of the local context entities. Consider our running example:

R1: “Customers shall be able to select multiple promotional packages for subscription.”

Subscription Process: {{Subscription Process},{The TelePortal displays the list of promotional packages...},{Select promotional package},{TelePortal},{Customer}...}

3.2.2 Identify Objects-Subjects-Verbs. We use the Stanford Parser [19] to identify the grammatical structure of the requirement statements and the activity descriptions which are extracted from the process models. The Stanford Parser provides the typed dependencies among the words which we use to identify the objects, subjects, and verbs in the requirement statements and the activity descriptions. In the running example, the following sets are formed from the statement R2 in Table 1:

Objects: {package, promotional packages, multiple promotional packages}
Subjects: {customers}
Verbs: {able, select, subscribe}

3.2.3 Lemmatization. Lemmatization is the process of transforming a word into its dictionary form, called a lemma. We use the Stanford Parser to transform the words into lemmas. Thus, we represent the sentences and phrases as bags-of-words. In the running example, the following bag-of-words is prepared for the requirement statement R2.


3.2.4 Remove Stop Words. In the next step, we sanitize the bags-of-words by removing the stop words. Stop words are the most common words used in English, thus having low descriptive power such as “the”, “a”, and “at”. The bag-of-words in the running example is updated as follows.

R2: {“StudentPack”, “promotional”, “package”, “subscriber”, “subscribe”, “to”, “Senior”, “package”, “at”, “the”, “same”, “time”}

3.2.5 Calculate Term Frequency. This step consists of determining the descriptive power of words in each bag-of-words in all requirement statements, activity labels, and activity descriptions.
We use term frequency - inverse document frequency (TF-Idf) to calculate a value which represents the descriptive power of a word. TF-Idf incorporates two concepts: Tf is the frequency of a word within a bag-of-words, a sentence or a document; Idf is the measure of how common a word is within a set of bag-of-words, sentences or documents. For Tf, we use the normalized raw frequency, which is the number of occurrences of a word w in a bag-of-words s divided by the total number of words in that bag-of-words. We calculate the term frequency of words in each requirement statement, activity label, and activity description. We use Equation (1) to calculate Idf of a word w in a set of bag-of-words S. S, in this equation, is selected based on the origin of the word w. For example, if w is a word in a requirement statement, then S is the whole set of bag-of-words constructed from the requirements.

\[ Idf(w, S) = \log \frac{|S|}{\sum_{s \in S} \min(1, \frac{|s|}{|S|})} \]  

(1)

Then, we calculate the TF-Idf of each word by multiplying the Tf and the Idf values.

3.2.6 Named Entity Recognition. In order to identify the terms with specific meanings within the organization, such as the names given to the IT systems (e.g. TelePortal), we use entity recognition based on the Semantic Local Context (Figure 6). For each requirement statement and sentence extracted from process models, we construct a set of identified local context entities. The approach also considers the acronyms if an abbreviation list is provided.

Note that the requirement statement R3 in the running example (Table 1) includes the terms “Easy-CRM”, “sales representative”, and “promotional package”. These are already defined in the semantic local context catalog, which is shown in Table 2. Thus, we construct the following set, which includes the entities mentioned in R3:

Entities: “Easy-CRM”, “sales representative”, “promotional package”

3.2.7 Compare Object-Subject-Verb Sets. In natural language statements, business objects may appear as objects or subjects. The same business object can be the object of a phrase and the subject of another. Therefore, for comparison purposes, we use the union set of objects and subjects. The verbs in a phrase represent the actions. We compare the verb sets to identify the similar actions.

We use the Jaccard Index variant shown in Equation 2 to compare the sets of objects/subjects and verbs of the requirement statements and the process models. A and B in the equation denote the sets to be compared. The reason for using a variant instead of the standard Jaccard Index [27] is that the two sets being compared differ significantly in size [11]. This may lead to a coverage relationship, that is a larger set covering a smaller one. The standard Jaccard Index falls short to identify such relationships. There is a need for a technique which gives more importance to the existence of a common element than the absence of it [9]. Therefore, the Jaccard Index variant is more suitable to identify a match between a small set of objects-subject-verbs of a requirement statement, and a larger one generated from a process model.

\[ JI'(A, B) = \frac{|A \cap B|}{\min(|A|, |B|)} \]  

(2)

We will refer to the similarity measures which are calculated by comparing the object/subject sets and verb sets as SimOS and SimV, respectively.

3.2.8 Calculate Cosine Similarity. We calculate the Cosine similarity [25] of the requirements and the process models using the TF-Idf values. For each sentence, we construct a vector using the TF-Idf values. For each comparison operation, we create a vector A to represent the requirement statement and another vector B to represent the compared process activity label or activity description. Both vectors have the exact number of dimensions as the words in the bag-of-words of the requirement statement. We use the TF-Idf values to populate the vectors’ corresponding dimensions. Then
we calculate the Cosine similarity by using Formula 3, where $A$ and $B$ are two n-dimensional vectors.

$$Sim_{\text{Cos}} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2} \sqrt{\sum_{i=1}^{n} B_i^2}}$$ (3)

Since we use Tf-Idf values to construct the vectors, the Cosine similarity gives us how similar the two text instances are, considering the descriptive power of the words. We will refer to the Cosine similarity measure as $Sim_{\text{Cos}}$.

3.2.9 Compare Semantic Local Context Entity Sets. In this comparison step, we compare the local context entity sets of the requirements and the process models. For the reasons explained in Section 3.2.7 we use the Jaccard Index variant for this comparison. This similarity measure will be referred to as $Sim_{\text{LC}}$.

3.2.10 Aggregate Similarity Measures. Lastly, we aggregate the four similarity measures, namely $Sim_{\text{OS}}$, $Sim_{\text{V}}$, $Sim_{\text{Cos}}$, and $Sim_{\text{LC}}$. To calculate a final similarity value, we use a linear equation with a weight constant assigned to each measure (Equation 4). The use of multiple similarity values provides flexibility for business analysts to adjust the importance of these dimensions according to their specific needs.

$$Sim = \frac{C_1 \cdot Sim_{\text{Cos}} + C_2 \cdot Sim_{\text{LC}} + C_3 \cdot Sim_{\text{OS}} + C_4 \cdot Sim_{\text{V}}}{C_1 + C_2 + C_3 + C_4}$$ (4)

In the following section, we present the evaluation of our approach in a real-life organizational setting.

4 DATA ANALYSIS AND RESULTS

In this section, we present the evaluation of the accuracy of our matching technique and the efficiency gains provided by the overall approach. We evaluated the accuracy of the matching technique quantitatively by comparing the matching results produced by our technique against a gold standard. To evaluate the efficiency gains of the overall approach, we conducted semi-structured interviews with the business analysts and inquired after their insights regarding the benefits of using such an approach. Sections 4.1 to 4.4 provide the information about the research setting, the evaluation method we used, and the results of this evaluation.

4.1 Evaluation Environment

This evaluation was conducted in a major telecommunications company. The company had extensive experience in using process models. Their process model repository included 620 BPMN diagrams, which were consistently prepared in accordance with the company’s standards. The models in the repository were stored in a hierarchical structure. Most of the process models included textual descriptions of the tasks and had links to elements such as actors, IT systems, and documents. The company had 8048 entries in the semantic local context registry, which consisted of IT systems, roles, documents, and business units. The requirements set is taken from a cross-functional project which heavily involves process improvement. The project recently passed the elicitation phase, and 200 unique requirement statements were collected from various business units. These requirements were mapped against the company’s high-level processes to emphasize the potentially impacted areas.

4.2 Evaluation Method

Within this setting, we evaluated our approach in two dimensions. First, we compared the results of the matching technique against a gold standard set. The gold standard set consists of requirements and matching high-level process models, which were prepared manually by the business analysts. This set is not specifically created for this study but prepared as a part of the analysts’ work.

We prepared five configurations with different weight constants in Equation 4 to examine the accuracy of each technique together and separately. These are displayed in Table 3. The first one (Config-1) has constants larger than zero for all similarity values; thus, taking all into consideration. We selected the weight constants of Config-1 intuitively, aiming at a configuration that has non-zero weight on each parameter, and favoring the word similarity more than the others. Our aim was not to optimize the configuration to provide the best matching results, but merely simulating a scenario in which the business analyst selects the parameters intuitively. Other configurations only use one of the similarity values. Config-2 uses Cosine Similarity ($Sim_{\text{Cos}}$), Config-3 uses semantic local context similarity ($Sim_{\text{LC}}$), Config-3 uses the object and subject set similarity ($Sim_{\text{OS}}$), and Config-4 uses verb set similarity ($Sim_{\text{V}}$).
4.3 Accuracy of the Matching Technique

Figure 7 shows the precision-recall graph of our matching technique in five configurations. As shown in the figure, Config-1, a combination of four similarity values, provides more accurate results than any of them separately. For the recall values lower than 0.8, Config-1 consistently outperforms the other configurations. The maximum F1 score of Config-1 is 0.63, with a precision of 0.50 and a recall of 0.85. Around the recall values of 0.9, all configurations display a similar F1 score. The reason is that the gold standard set consists of matching between the requirements and the high-level processes as explained in Section 4.2. This causes all configurations to produce most of the possible matchings, thus providing a high recall and a similar precision value.

4.4 Efficiency of the Overall Approach

Our findings from the semi-structured interviews that we described in 4.2 are summarized below in three groups based on the related process improvement stage. The interviewees provided these information based on expert judgment and working experience in the projects which the data have been collected from.

Requirements elicitation and conflict resolution requires an overall understanding of the organization. The conflict resolution part is usually performed by more experienced business analysts. This task normally requires 30 to 45 person-minutes per requirement statement. By using our approach, the business analysts foresee a direct effort gain of 20 person-minutes on average per requirement statement, in a project of similar complexity and size. Moreover, an indirect efficiency gain is also possible due to the increased task consistency and quality, and the decreased number of errors. Each error occurred in this stage leads to many hours of rework.

Impact analysis is effort-intensive, time-consuming, and prone to errors. The actual effort spent in the project for this stage was 120 person-hours. Since this stage mostly consists of examining a large number of documents and IT systems to identify their dependencies, there is significant room for improvement by automation. The senior business analysts estimated a 50% to 70% efficiency increase in the identification of dependencies if an automated solution provides a list of possible impacted entities.

The quality of BPI planning depends on a proper categorization of the requirements and the impact analysis. For a small or medium-sized project, this stage takes approximately 80 person-hours. This effort requirement increases for larger and more complex projects. According to the judgment of the business analysts, at this stage, a 30% to 40% gain in time, and a 20% decrease in errors are expected with the usage of our approach. Moreover, less-experienced personnel will be enabled to perform this task. Such significant efficiency gains are made possible by automating the aggregation of a large amount of complex information. In addition to these effort savings, rework may be avoided due to the decreased number of human errors. Furthermore, the business analysts are enabled to develop actionable BPI plans by means of a finer level matching with respect to what they normally have.

### Table 3: Configurations with different weight constants

<table>
<thead>
<tr>
<th>Config</th>
<th>Cs</th>
<th>LC</th>
<th>OS</th>
<th>Y</th>
</tr>
</thead>
<tbody>
<tr>
<td>Config-1</td>
<td>0.5</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>Config-2</td>
<td>1.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>Config-3</td>
<td>0.0</td>
<td>1.0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>Config-4</td>
<td>0.0</td>
<td>0.0</td>
<td>1.0</td>
<td>0.0</td>
</tr>
<tr>
<td>Config-5</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>1.0</td>
</tr>
</tbody>
</table>

We quantitatively evaluated the accuracy of the matching technique by calculating the $F_1$ score, precision, and recall. Precision shows the ratio of correctly identified matching pairs, while recall describes the ratio of actual matching pairs identified by our technique. $F_1$ score is an acknowledged measure of the accuracy of information retrieval systems, which is the harmonic mean of precision and recall. We use equations 5 and 6 to calculate precision and recall respectively. $S_A$ denotes the set of actual matching pairs in the gold standard, and $S_P$ describes the set of matching pairs identified by our technique.

\[
\text{precision} = \frac{|S_A \cap S_P|}{|S_P|} \tag{5}
\]

\[
\text{recall} = \frac{|S_A \cap S_P|}{|S_A|} \tag{6}
\]

Secondly, we conducted semi-structured interviews with three business analysts to understand the perceived value of this approach, and how much efficiency they expect to gain from it, if any. Two of the interviewees were senior analysts having 17 and 5 years of experience in the BPI field, while the third was more junior (two years of experience). During each interview, which took approximately an hour, we performed a walkthrough of the process improvement tasks and the preparation of the deliverables. Furthermore, we addressed the following topics: (1) description of business analysts’ work in different BPI stages, (2) the effort requirements and the complexity of these tasks, (3) the level of knowledge and skills needed to carry out their work, (4) the risks associated with the tasks, and (5) the interviewee’s opinions regarding the improvement opportunities they see in these activities.

In the following subsections, we present our evaluation results.
5 RELATED WORK

Organizations establish process model repositories which may contain hundreds or even thousands of models [21]. Finding the relevant process information is a complex task in such large repositories. The motivation to overcome this challenge led the way to the development of many process query techniques [23].

A few querying techniques use visual process model fragments as inputs such as BPMN-Q [4], and VMQL [26]. Others incorporate a textual query language which is independent of the modeling notation [28]. All these techniques require the user to build sophisticated queries.

Many related studies exist in the literature which involve the use of NLP in the BPM field for a diverse set of purposes such as model matching [3, 20], process search [14], and model analysis [2]. In this study, we utilize the NLP-based matching techniques as proven to be useful in the BPM field in such studies. However, this study differs from the existing literature with respect to the application and integration of such techniques. The need for a different approach stems from the nature of the requirement statements. Most importantly, requirement statements are written by different stakeholders. Therefore, they lack a consistent structure. Secondly, the statements are often short and usually do not include activity descriptions. However, the requirements may mention local context entities, namely, IT systems, documents, roles, business units, and products and services in the organization. To deal with the challenge posed due to these characteristics of requirement statements, we used multiple similarity measures in this study. Our results show that these measures can be used synergistically to achieve higher matching accuracy.

Over the last years, NLP techniques reached a sophisticated level of maturity. However, there is still room for improvement of NLP-based matching based on domain ontologies [13]. Our findings support this claim by showing that traditional NLP-based matching techniques can be improved by using a registry of organizational entities in a semantic local context.

6 DISCUSSION AND CONCLUSION

In this paper, we introduced an approach to facilitate BPI by providing analytical capabilities to detect and resolve requirement conflicts, analyze impacts, and develop actionable BPI plans. Business process models are essential components in this approach because they employ cross-functional information. This information is used for inferring associations among a wide variety of organizational entities. Our approach uses and integrates multiple NLP techniques for matching business requirements to process models. Our matching technique works by calculating four similarity measures (i.e., Cosine similarity, and the similarities of object/subject sets, verb sets, and local context entity sets) and aggregates these to detect a matching requirement-process pair. We evaluated our approach in a major telecommunications company. Our results show that our technique was able to accurately match business requirements and process models. Furthermore, the semi-structured interviews we conducted with the business analysts revealed that the proposed approach may lead to significant gains in efficiency and a decrease in errors.

The gold standard matching set used for evaluation was prepared as an actual business deliverable, which consists of business requirements mapped to high-level processes. Although this is a realistic usage scenario, it hindered a fair evaluation of the accuracy of our matching technique, specifically for the high values of recall. Nevertheless, the evaluation results showed that our matching technique accurately detected the matching pairs for relatively lower recall values.

A limitation of our matching approach is that it does not take lexical semantics (i.e. synonyms) into account. This limitation can be addressed by introducing a new step in the solution which consists of using WordNet or Lin’s approach for lexical semantic identification. We also acknowledge that our matching technique can be further improved by handling organization-specific stop words, synonyms, and acronyms.

In future work, we plan to evaluate our approach using another gold standard set, which comprises a matching between requirements and fine-level process models. We also plan to improve our approach by implementing the capability to process word semantics as well as the organization-specific important words. Our observations suggest that the utilization of organizational semantics improves the matching results. We are interested in examining how our approach may also benefit from other contextual information sources such as the implicit social relationships among requirement owners and related process actors.

ACKNOWLEDGEMENTS

This research has been supported by The Netherlands Organisation for Scientific Research (NWO). The authors thank Chris Bergman, Mark Cloesmeijer, Roy Goverde, Vitali Habif, and the personnel of Precedence for their contributions and support throughout this study.

REFERENCES


Data Driven Credit Risk Management Process: A Machine Learning Approach

Mingrui Chen
Southern Methodist University
6425 Boaz Lane
Dallas, Texas, USA 75205
mingruic@smu.edu

LiGuo Huang
Southern Methodist University
6425 Boaz Lane
Dallas, Texas, USA 75205
lghuang@smu.edu

Yann Dautais
GDS Link, LLC
5307 East Mockingbird Lane
Dallas, Texas, USA 75275
yann.dautais@gdslink.com

Jidong Ge
State Key Laboratory for Novel Software Technology
Software Institute, Nanjing University
Nanjing, China 210093
gjd@nju.edu.cn

ABSTRACT
Credit scoring process, the most important part in credit risk management, aims at estimating the probability that an applicant will perform bad credit behaviors (e.g., loan default). Managing and developing effective and reliable risk assessment procedures in order to mitigate potential loss caused by new applicants heavily relies on the performance of scoring process. Traditionally this process is manually developed, which is time-consuming. In this paper, we propose an automated credit risk management process based on machine learning to ease the scoring process in order to reduce the human effort. This process is data driven: it leverages machine learning to automatically analyze vast amounts of historical data and build predictive model. We evaluate our process with a real-world proprietary dataset and achieved good performance, which shows the feasibility of using machine learning to facilitate the credit risk management process.

CCS CONCEPTS
- Computing methodologies → Machine learning; - Information systems → Decision support systems; Data analytics;

KEYWORDS
Credit risk management, Data driven, Machine learning, Process improvement

ACM Reference format:
DOI: 10.1145/3084100.3084113

1 INTRODUCTION
The scoring process is an essential part of the credit risk management system used in financial institutions to predict the risk of loan applications. This process uses a statistical model that takes into account the application data and performance data of a credit or loan applicant and estimates the defaulting probability, which is the most important factor used by the lender to rank applicants for decision making.

Developing an effective scoring process depends on two factors: (1) appropriate selection of key attributes from historical data, and (2) accurate estimation of defaulting probability produced by predictive model. The traditional way of developing this process is known as scorecard, which is a manual approach that subjectively select attributes and creating rules based on personal experience of analysts. However, when the dimension of data is increasing, manual analysis becomes more and more time consuming and error-prone, and the results are often subjective or even biased depending on experience. To meet these challenges, we propose an automated scoring process that simplifies the task of estimating the defaulting probability.

In this study, we employ Support Vector Machine (SVM) as the core algorithm to automate the scoring process due to two reasons: (1) SVM can solve both linear and non-linear classification problems and yield promising classification performance on multi-dimensional data with the capability to avoid the curse of dimensionality problem [14], (2) SVM has been well developed for probability estimation task [17]. The SVM algorithm was first proposed by Vladimir N. Vapnik [15] and we use LibSVM library [2].

The advantages of our approach are twofold: (1) all attributes are taken into account by machine learning algorithm so that the predictive model is less subjective (2) it eliminates the need to pick attributes and create rules manually so that it can be easily and effectively adapted to different datasets with different attributes.

We organize the paper as follows: Section 2 presents some related work of credit scoring. Section 3 introduces the basic idea of the traditional scoring process. Section 4 introduces our automated process based on machine learning algorithm. Section 5 describes
the experiment setup and evaluation results. Section 6 summarizes our work and proposes future work.

2 RELATED WORK

Machine learning algorithms have been used by practitioners and researchers for several credit risk management tasks. Worrachartdetchai, U. and Sooraksa, P. applied Least Squares Support Vector Machine to classify credit applicants into four groups [16]. Gestel et al. proposed SVM-based approach to classify credit applicants into even more groups [7]. Zhang et al. investigated a hybrid approach in building credit scoring model by combining genetic programming with support vector machine [18]. Kiani, M. and Mahmoudi, F. proposed another hybrid method for credit scoring by combining clustering and support vector machine in order to increase the accuracy of classification [11]. Neural network have been used in credit scoring before SVM was introduced. Huang et al. demonstrate that SVM can outperform neural networks in building risk classification model [9].

However, to our best knowledge, existing works employing machine learning algorithms mainly study how to increase the accuracy of classification models that classify applicants into discrete risk levels, but none of them is focusing on automating the complete scoring process to estimate defaulting probability. In addition, based on our industrial experience, the scoring process is still manually developed in financial institutions nowadays.

3 TRADITIONAL PROCESS

First, we introduce some definitions to help understanding the process.

• Good applicants: existing customers that have been observed to have good credit behaviors and classified as good
• Bad applicants: existing customers that have been observed to have bad credit behaviors and classified as bad
• Accepted applicants: applicants that have been accepted. This population consists of both good and bad applicants mentioned above
• Rejected applicants: applicants that have been rejected. This population has no observation of credit behaviors thus cannot be classified as good or bad when collecting data

The traditional way to estimate defaulting probability is accomplished by building scorecard. The common procedures are:

1. Collect data from existing customers and rejected applicants who are classified as Good, Bad and Reject
2. Do reject inference to assign inferred probability to applicants who have no history
3. Manually select a subset of attributes from the dataset based on analyst’s experience
4. Manually create rules based on selected attributes and estimate credit score for new applicants
5. Estimate final defaulting probability using statistical models based on the distribution of Good/Bad samples and associated credit score

Reject inference (Step 2) is a widely applied technique[6][13] which is an essential step to determine whether the model is generalizable to the entire population of applicants. When building scorecard, samples need to be identified as Good or Bad. However, in practice, only the performance (Good or Bad) of existing customers who have been accepted is available in the modeling data. Thus the model built exclusively on accepted samples may incur critical bias when it is applied on the entire “through-the-door” population, which includes new applicants who may have significantly different behaviors than existing customers. To resolve this problem, we can leverage the information of rejected applicants in the model. However, their performance has never been observed, therefore it is not able to identify which category (Good or Bad) they belong to. Reject inference is a technique to infer the performance, typically the probability of being Bad of rejected applicants, so that the “performance” of rejected applicants can be included in the model construction.

4 PROPOSED PROCESS

Figure 1 shows the framework of our automated scoring process. It has four major components connected as a workflow. It is different from the traditional process in that attribute selection, model construction and probability estimation are all automated in order to eliminate intensive human effort.

4.1 Data Preparation

In the traditional process, analysts need to examine the attributes in the historical data and pick a subset of attributes based on their experience. However, in our automated process, we can keep all the attributes and feed the complete dataset to the learning algorithm. During preparation, each data sample shall be classified into one of the following categories, namely “Good”, “Bad” and “Reject” which are mentioned in Section 3. In addition, since all the samples labeled as Good and Bad come from accepted applicants, they are also assigned a label “Accept”.

4.2 Predictive Models

The predictive model aims at estimating defaulting probability using SVM, instead of using human-generated rules. First we introduce some notations:

- \( P(B|A) \) denotes the probability of a sample being Bad in accepted samples
- \( P(R) \) denotes the probability of a sample being Reject
- \( P(B|R) \) denotes the inferred probability of a sample being Bad in rejected samples
- \( P(B) \) denotes the probability of a sample being Bad in all samples

The predictive model comprises of four components:

1. An initial Good/Bad model estimates the probability \( P(B|A) \)
2. An Accept/Reject model estimates the probability \( P(R) \)
3. A reject inference model infers \( P(B|R) \) based on \( P(B|A) \) and \( P(R) \)
4. A final scoring model estimates the probability \( P(B) \), which is equivalent to the defaulting probability

4.2.1 Initial Good/Bad Model. An initial Good/Bad model is first trained using SVM to estimate \( P(B|A) \). Before training, we construct a training set using the samples labeled Good and Bad from collected data. SVM is a kernel-based algorithm, so selecting the effective
kernel is important to achieve good results. We choose the RBF kernel in our experiments because RBF kernel has been proven to achieve very promising classification performance [10, 12]. We tune two parameters for RBF kernel, namely cost (c) and gamma (g). During the training process, the algorithm selects the (c, g) parameters using an exhaustive searching algorithm implemented in LibSVM [3]. Suppose there is a search space \( p = \{c, g\} \), in which c is cost and g is gamma. The search algorithm is to select the pair of (c, g) which results in the lowest rate of erroneous classifications in the training data set by conducting five-fold cross validation internally. The resulting classifier is then used to estimate \( P(B|A) \) of each sample by enabling the probability output option in LibSVM.

### 4.2.2 Accept/Reject Model

An Accept/Reject model is then trained using SVM as well to estimate \( P(R) \). Before training, we construct a training set using the following way: if the sample is labeled Good or Bad, it is assigned Accept, otherwise it is Reject.

The Accept/Reject model is used not only to differentiate Accept and Reject samples, but also to rank Reject samples in such a way that those who are likely to have been bad had they been accepted get a high probability of being rejected. This implies we need to emphasize the attributes that are important to classify samples into Accept or Reject, as well as to differentiate the Bad samples from Good ones in the Accept/Reject model. Thus, when training the Accept/Reject model, we take into account the importance of attributes from the initial Good/Bad model. We use a F-score based feature selection approach implemented in [4] to find the most significant attributes in the initial Good/Bad model and increase the weights of these attributes when training the Accept/Reject model. The resulting classifier is then used to estimate \( P(R) \) by enabling probability output in LibSVM.

### 4.2.3 Reject Inference

The purpose of reject inference has been explained in section 3. In this paper, we use a standard reject inference method which is called extrapolation. The extrapolation is performed by adjusting the probabilities \( P(B|A) \) produced by initial Good/Bad model based on the probabilities \( P(R) \) produced by Accept/Reject model such that \( P(B|A) \) are extrapolated onto Reject samples. The lower \( P(R) \) is, the more similar is the rejected sample to a good one, so that it has a lower inferred probability \( P(B|R) \). The extrapolated probability is factored in the following way:

\[
P(B|R) = f(P(B|A) \cdot (P(R)))
\]

where \( P(B|R) \) is the inferred probability of being Bad for rejected samples, \( P(B|A) \) and \( P(R) \) are estimated probabilities mentioned in section 4.2.1 and 4.2.2., and \( f() \) is a scaling function that scales \( P(B|A) \) to the entire rejected samples. As this is a standard method used in credit scoring industry, we don’t explain it in every detail and more details can be found in these two papers [6][13]. When \( P(B|R) \) is available for rejected samples, we can assign Good or Bad labels based on a cutoff percentage \( p \). For instance, we can rank rejected samples by their inferred probability \( P(B|R) \) in ascending order, and set \( p \) to 20%. Then we label the top 20% samples in the ranked list as Good and the remaining samples as Bad. Note that \( p \) can be different, and should be smaller than the percentage of original Good samples in Accept category.

### 4.2.4 Final Scoring Model

Now that we have a dataset which consists of both original and inferred Good/Bad samples, we can train a final scoring model on this less biased dataset to estimate the defaulting probability. We use the same setting of SVM mentioned in 4.2.1 to train the final scoring model.
Table 1: Distribution of data samples

<table>
<thead>
<tr>
<th>Label</th>
<th>Good</th>
<th>Bad</th>
<th>Reject</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td># of Samples</td>
<td>1019</td>
<td>379</td>
<td>4226</td>
<td>5624</td>
</tr>
</tbody>
</table>

4.3 Risk Assessment & Decision Making

After the final scoring model is built, we can use it to estimate the credit risk of new applicants. Decision can be made according to the outcome of the final scoring model and specific guidelines of financial institutions (i.e. to which level of defaulting probability will an applicant be accepted or rejected).

4.4 Update Data

Capturing the change of new and past applicants’ performance data is essential to keep the predictive model effective and reliable. Our approach leverages automatic feature selection which provides the capabilities of handling attributes that are frequently changing and new data samples that are added from new applicants. For example, if some rejected applicants are accepted and credit observations become available, or some Bad applicants become Good, we can update the data and retrain the predictive model.

5 EVALUATION RESULTS

This section evaluates our approach. The evaluation is based on a statistical model using accuracy and Gini coefficient as the primary evaluation metrics.

5.1 Experiment Setup

The dataset used in our study contains 5624 data samples which are provided by a bank in Bulgaria in collaboration with GDS Link, LLC. Each sample has 256 attributes (e.g. age, income, number of on-time payments). Originally, data is annotated by three categories, namely Good, Bad and Reject respectively. Table 1 shows the distribution of data samples in our dataset. Among the 5624 samples, there are 1019 labeled as Good, 379 labeled as Bad and 4226 labeled as Reject.

Recall that when training the initial Good/Bad model and final scoring model, we use data samples that are labeled as Good and Bad. Thus when preparing the dataset, we split the samples into 10 folds, each of which contains 10% Good samples and 10% Bad samples. If the sample belongs to Good, its class value is +1; otherwise, its class value is -1. We perform the jack knife by training the classifier based on 9 out of the 10 folds and test the classifier on the remaining one to get the probability \(P(B|A)\) of test fold. We performed 10-fold cross validation in order to get a complete set of probability \(P(B|A)\) on the whole dataset.

When training the Accept/Reject model, on the other hand, we use data samples that are labeled as Accept and Reject. If the training instance belongs to Accept, its class value is +1; otherwise, its class value is -1. Again we use the 10-fold cross validation aforementioned to get a complete set of Probability \(P(R)\) on the whole dataset.

5.2 Classification Accuracy

In order to estimate reasonable probabilities, we should first ensure a good classification accuracy of initial Good/Bad mode and

<table>
<thead>
<tr>
<th>Levels</th>
<th>Goods</th>
<th>Bads</th>
<th>Cumulative Goods</th>
<th>Cumulative Bads</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>575,796</td>
<td>238,204</td>
<td>33.23%</td>
<td>6.12%</td>
</tr>
<tr>
<td>0.1</td>
<td>267,358</td>
<td>113,642</td>
<td>48.66%</td>
<td>9.04%</td>
</tr>
<tr>
<td>0.2</td>
<td>147,798</td>
<td>77,202</td>
<td>57.19%</td>
<td>11.03%</td>
</tr>
<tr>
<td>0.3</td>
<td>93,542</td>
<td>67,458</td>
<td>62.58%</td>
<td>12.76%</td>
</tr>
<tr>
<td>0.4</td>
<td>71,336</td>
<td>64,664</td>
<td>66.70%</td>
<td>14.42%</td>
</tr>
<tr>
<td>0.5</td>
<td>75,396</td>
<td>78,604</td>
<td>71.05%</td>
<td>16.44%</td>
</tr>
<tr>
<td>0.6</td>
<td>55,442</td>
<td>84,558</td>
<td>74.25%</td>
<td>18.61%</td>
</tr>
<tr>
<td>0.7</td>
<td>67,898</td>
<td>106,102</td>
<td>78.17%</td>
<td>21.34%</td>
</tr>
<tr>
<td>0.8</td>
<td>32,011</td>
<td>152,989</td>
<td>81.17%</td>
<td>25.27%</td>
</tr>
<tr>
<td>0.9</td>
<td>79,331</td>
<td>320,67</td>
<td>85.75%</td>
<td>33.51%</td>
</tr>
<tr>
<td>1</td>
<td>246,952</td>
<td>2587.05</td>
<td>100%</td>
<td>100%</td>
</tr>
</tbody>
</table>

Accept/Reject model before doing reject inference and building the final scoring model. The classification accuracy of these three models are listed below. The accuracy is obtained by combining the prediction results of 10 test folds.

1. The initial Good/Bad model and final scoring model yields overall accuracy of 77.8% and 87.1% respectively
2. The Accept/Reject model yields overall accuracy of 93.1%

From the results we can see that SVM provides satisfactory performance for us to build predictive models.

5.3 Gini Coefficient

Gini coefficient [8] is a statistical method which is commonly used to measure the ability of a predictive model [5]. The higher Gini coefficient is, the better the model is. Formally, it is defined as:

\[
Gini = 1 - \sum_i (G_{i+1} - G_i) \cdot (B_{i+1} - B_i)
\]

(2)

where \(i\) is a set of cut-off values to select Good and Bad applicants, \(G_i\) is the proportion of Good applicants passing cut-off value \(i\), and \(B_i\) is the proportion of Bad applicants passing cut-off value \(i\). More specifically, the Gini calculation is done with each rejected applicant assigned to both the Bad and Good categories with a probability \(P(B)\) and \(1 - P(B)\) respectively and plotted based on 10 levels ranging from 0 to 1, with an interval of 0.1. For example, for a rejected applicant, if the probability of being Bad is 0.65, the probability of being Good will be 1 - 0.65 = 0.35. Hence on the level of 0.6, which is the 9th row of Table 2, 0.65 will be added to the Bad category and 0.35 will be added to the Good category. Table 2 shows the cumulative values used to plot the Gini curve.

Fig. 2 visualizes the Gini coefficient. The red dotted line represents the ability of discrimination of an actual predictive model, the diagonal represents no discrimination. In the graph, the larger the area between the red line and the diagonal, the better the predictive model is at discriminating Good and Bad. As a rule of thumb, predictive models that have 50% or more Gini coefficient are considered to have good predictive power [1]. Our approach achieves 64% Gini coefficient, which is acceptable.
The probabilities are sorted in ascending order based on the ones predicted by SVM and keep the inferred one tied with each sample. We can see that both models have similar trend, whereas after employing reject inference, the final scoring model generates a more flat curve, which indicates more rejected applicants are predicted to have lower defaulting probability.

6 CONCLUSION AND FUTURE WORK

In this paper, we propose an automated credit scoring process based on SVM. Our approach is fully automated so that it can largely reduce the human effort in developing the scoring process and effectively adapt to different datasets and financial institutions. We evaluate and validate our SVM-based predictive model on a real-world data set from a bank in Bulgaria and it achieves high accuracy and 64% Gini coefficient. Future work includes testing our approach on other credit datasets, exploring ways to further increase the performance and comparing with other machine learning algorithms.

REFERENCES


5.4 Effect of Reject Inference

This section does not evaluate the accuracy of predictive model, but shows a general idea about how reject inference can affect the predictive model on rejected samples. In Fig. 3 and Fig. 4, the X-axis represents all the rejected samples and the Y-axis represents the probabilities of being Bad for each sample. They reflect the trend of probabilities of being Bad on the rejected samples by applying the initial Good/Bad model before doing reject inference and the final scoring model after doing reject inference. The red dotted lines represent probability estimated by SVM and the blue lines represent the probability directly inferred by reject inference model.
Using TRIZ to Balance Software Process Commonality and Diversity

Dan Houston
The Aerospace Corporation
2310 E. El Segundo Blvd M/S M1/112
El Segundo, CA, 90245 USA
dxhouston@ieee.org

ABSTRACT
A normal tension exists between a corporate need for process commonality and local needs for process customization. Unifying diverse software processes to produce process commonality is challenging. A group seeking to create a common development process must consider issues such as a rationale for process commonality, how to design a process for balancing commonality and customization, how to analyze existing diverse processes, and whether and how to measure process commonality. Altschuller’s Theory of Inventive Problem Solving (TRIZ) is explored for resolving the contradiction between process commonality and diversity and examples from the literature are cited.

CCS CONCEPTS
• Software and its engineering→Software creation and management→Software development process management→Software development methods

KEYWORDS
software process commonality; software process customization; common software process; software process diversity

1 INTRODUCTION
Large, software-producing companies often have many development programs and projects that lacked coordinated process guidance early enough to avoid software process proliferation. Software groups within such companies may have spent many years independently addressing development process issues. Process diversity can be locally efficient and globally expensive. This expense has been amplified by several phenomena.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

ICSSP’17, July 2017, Paris, France
© 2017 ACM ISBN 978-1-4503-5270-3/17/07…$15.00
DOI: 10.1145/3084100.3084114

As the economics of many products has shifted over recent decades, hardware often becomes a commodity and software becomes a primary revenue source and/or product differentiator.

Globalization has led to more distributed development, resulting in stronger needs to reduce the cost and delay effects of process diversity on large programs.

Development process diversity is part of the challenge when integrating acquisitions that use entirely different development processes.

While executives may see common processes as a goal, others see software process diversity as inevitable, a phenomenon to be managed rather than replaced with process uniformity (Beck 2001).

This paper takes a balanced look at uniformity and diversity in software processes, proposing a means of achieving a significant degree of process commonality. No one solution can be proposed. Rather, this paper is intended to provide software process leaders and engineers with options to consider, tools for understanding how to arrive at a balanced approach, and some lessons from across the fields of software research and industry.

The first section of this paper discusses two sets of benefits: those of a common process and those of process diversity. The discussion then turns to framing the practical problem, a prelude to resolving the conflict between process commonality and diversity using Altschuller’s Theory of Inventive Problem-Solving, or TRIZ. A number of Altschuller’s inventive principles are considered and examples of their use in the field of software development process are listed and discussed. The approach is taken that an architectural principle for process commonality should be selected and followed by detailed analysis of diverse processes to be subsumed. Means of analyzing the commonality of the diverse processes are discussed. Measurement of process diversity is also treated so that process engineers who undertake a process commonality investigation/development/enhancement might have a basis for reporting status and progress. Finally, a proposal is offered for achieving a degree of process commonality from process diversity and some desirable characteristics of a common process are listed.

2 PROCESS COMMONALITY BENEFITS
When management realizes that completely diverse processes are expensive, they often seek a common development process for a variety of reasons.
Better management of human resources. With the cost of software development almost entirely dependent on intellectual assets, the ability of developers to move between projects and quickly become productive is advantageous. Knowledge of domain and product must be gained through study and experience, but process knowledge is one transition obstacle that can be managed preemptively. Common process is expected to lower transition costs, not only for individual process users, but also for managers and leaders responsible for process deployment and compliance.

Reduced communication costs. A common development process provides a common language for communicating across a company and with customers. From their experience with distributed development, Procuniar et al. (2001) report that a common language of development process not only minimizes training costs, but also reduces chances for miscommunication, especially as commonality is driven “into procedures, tools, and terminology.”

Improved results in geographically-distributed projects. This benefit, cited by owners of projects that employ a common process across sites, is partially a result of the communication cost reduction, but is also achieved through the deployment of common software process assets (Kaltio 2001). “Process, procedures, tools, and terminology commonality provides the distinct advantage of being able to move work seamlessly across multiple sites” (Procuniar et al., 2001). Delphi reported similar findings following an SEI assessment.

The assessment of Delphi Delco Electronics was significant because of the sheer size of the organization and because the assessment looked across the organization’s product lines and determined that a common software process is followed across its many sites. The assessment was one of the first to cover a single process across multiple continents and provide a single rating for the entire enterprise.

As a result of our common process, time zones become meaningless, global teamwork becomes easier,” said Les Wilkinson, Delphi Delco Electronics’ general director of Engineering. “We are able to leverage our global resources to achieve shorter cycle time, improved quality, reduced cost and cutting-edge technology – the factors most important to our manufacturing customers as they seek to differentiate their vehicles with the end consumer. (Delphi 2003)

Lower cost of process maintenance and improvement. Documented development processes require maintenance, ordinarily by designated process experts. A common process is reasonably expected to reduce the cost of process expert staffing for updating and deployment. Also, the benefits of automation improvements based on a common process are multiplied by the number of programs utilizing it.

Deeper understanding of development dynamics. Software process modeling, whether through static diagrams or through dynamic simulation, has enabled better understanding for strategic management, project planning and management, process improvement, and engineering training (Kellner et al., 1999). Process modeling is facilitated by commonality (limited variation) and its cost is easier to justify for a widely used process rather than for multiple, varied processes.

Software reuse. Based on a survey of software reuse practices, Frakes and Fox (1995) concluded that a defined software process is a factor that promotes reuse among the users of the process.

3 PROCESS DIVERSITY BENEFITS

The July/August 2000 issue of IEEE Software focused on the theme of software process diversity. In the editor’s column of that issue, Robert Glass discussed the increasing awareness in the software industry of the necessity for process diversity. He presents some project-based reasons.

Size: Large projects require much more methodological discipline and formality than small ones.

Domain: Business applications are very different from systems programming applications, scientific applications, real-time applications and edutainment applications.

Criticality: Projects where lives or huge amounts of money are at stake must be treated differently from projects that are less critical.

Innovativeness: Projects that tax the programmer’s mind and creativity inevitably will result in methodological restraints being lifted.

In addition to these project factors, the following benefits of process diversity can be cited.

Business strategy. In discussing ways to accelerate software measurement, Stan Rifkin (1999) demonstrates the effect of marketing focus on software development. The choices made between operational excellence, customer intimacy, and product innovation will strongly influence software process. Products for markets needing operational excellence are typically developed in organizations that rely on mature processes, such as those appraised using the SEI CMMI. MIS success is often found in organizations that use techniques and methods for emphasizing customer intimacy. Microsoft consumer products address markets desiring feature innovation.

Project priorities. In addition to business strategy, projects are driven by multiple concerns that include quality-based technical trade-offs and regulatory constraints. Prioritizing these concerns provides direction for process employment (Cockburn 2000). The Personal/Team Software Process (Humphreys 1997) emphasizes predictability. Agile methods
emphasize responsiveness to change and productivity of working software. Government regulations set priorities for software produced for the pharmaceutical industry, resulting in processes that emphasize the role of traceability.

- **Program resources.** A program’s size, the knowledge and experience of its people, and its hardware resources often shape the development and application of software processes.

- **Product lifecycle.** Development processes are often written for new products and must be adapted for technology development cycles and for product maintenance releases.

- **Product source.** When a product is developed from reusable or acquired software, the process must accommodate the type of sourcing. Variant processes must be employed for acquiring and integrating subsystems or functionality obtained from code generators, from a prior product, from a product line, from a contracted third-party, from a commercial library, or from open sources.

- **Process limitations.** A process may have limitations built into it that prevent it from being applied to all projects. These project-specific limitations include project roles and organizational structure, assumptions about supporting tools and technology, and specific policies. (Perry 1996)

## 4 THE PRACTICAL PROBLEM

Good cases can be made for both process commonality and process diversity. As mentioned earlier, many large companies find themselves with a history of process diversity and they want more process commonality. But the two are often held in tension. Particularly when approaching the goal of commonality from a history of diversity, management and developers may polarize around this issue, managers wanting a common process and developers wanting the freedom of process diversity. This presents a challenge to process leaders and experts, who may be tempted to take one of three avenues.

- a) Create and deploy an entirely new process that replaces existing processes.
- b) Take the process that is most mature or is used by the majority and deploy it in the remaining groups.
- c) Study the existing processes, identify what is common, and call it the common process.

Each of these approaches has its benefits and drawbacks. Creating a new process provides a fresh start, but it takes away something from everyone and is an expensive undertaking that may or may not satisfy stakeholders. The second approach is clearly the least expensive for deployment, but McMahon (2001) offers the following lesson gained from working with distributed development programs.

One of the most common pitfalls witnessed on distributed projects is referred to as the “let’s use the most mature process available” pitfall. This pitfall usually starts with the project leadership’s decision to mandate that all project sites use a common process. While at the appropriate level a common process makes sense, the pitfall is tied to what often happens next. Rather than define the common process at the appropriate level and allow individual sites the appropriate freedom to leverage site-specific procedures, oftentimes a mandate is sent across the distributed sites to drive procedure commonality (different from process commonality) as well. The common set of procedures chosen is usually supplied by the highest software maturity-rated organization on the project.

It is natural to look to your teammate with the highest process maturity for software guidance. However, procedures represent only a small part of the complete process maturity picture. They are often too site-specific to make sense for application across multiple organizations (each with their own culture and history) in conducting development activities.

The third approach is the most promising, but is also the most challenging. Procuniar et al. (2001) provide a good description of the challenge based on their experiences with distributed projects.

Many collaborative ventures have struggled when trying to establish the right level of process/procedure commonality across a virtual project. When process commonality is driven too low, site-specific culture clashes often lead to intense conflicts and leadership struggles. But when left too high, ambiguous terminology, divergent tools and inconsistent work instructions can cause a dramatic rise in the integration risk.

Of the three approaches, the third one requires the most knowledge to ensure that the final process is complete, coherent, and consistent. If undertaken properly, it can be both economical and effective. The remainder of this paper discusses this approach.

The question, then, is how to undertake process commonality in the face of process diversity. Innovation from this kind of contradiction has become the hallmark of a methodology called TRIZ, a Russian acronym for “Theory of Inventive Problem Solving.” (Terninko et al., 1988) Though TRIZ uses many methods, it is best known for the algorithm that identifies product parameters, forms a contradiction between them, and resolves the contradiction innovatively. The next section briefly discusses how TRIZ has been applied in software development, then applies TRIZ to the problem of producing a common software process for an organization that uses diverse processes.

## 5 PROCESS COMMONALITY DESIGN

### 5.1 Applications of TRIZ to Software Work

TRIZ was developed by Genrich Altshuller after years of studying patents and describing levels of innovation. Altshuller found that the most innovative solutions were produced when a physical contradiction could be resolved and that underlying principles were at work in the resolutions. He set about defining product parameters that describe useful functionality and inventive principles for
resolving contradictions between the parameters. Although the inventive principles can be employed without reference to the parameters, the task of employing them is facilitated by identifying a contradiction in parameters and using a contradiction matrix that shows which principles have been used most often in resolving each pair of contradictory parameters.

Although the inventive principles may sound like solutions, they are not. They do provide a “systematic and potent means of breaking out of current paradigms into often exciting and beneficial new ones” (Mann and Domb, 1999). They assist in stimulating creative thinking and directing it toward optimal solutions. The hard work of detailing design options and systematically analyzing them for a particular situation is still required. Thus the exercise demonstrated in this paper is not meant to provide an answer for those who face the problem of producing process commonality, but it is meant to provide stimulus for pursuing a good solution.

Altschuller’s work was done prior to the rise of computing and widespread software development, so it focused on physics-based products. In recent years, TRIZ has been applied to other fields, such as business process management (Mann and Domb, 1999; Ruchti and Livotov, 2001) and software development (Rea 2001, 2002, 2005; Mann 2004; van den Tillaart 2006). The approaches to applying TRIZ in other fields include software patent research (Mann 2004), analogies (Rea 2001), and revision of principles to a target field (Ruchti and Livotov 2001).

Software applications of TRIZ have been oriented toward software product problems rather than software process problems. The abstractions of software process problems bear more similarity to business process problems than to product design problems, so the following analysis relies on analogies and on the business process applications of TRIZ.

5.2 TRIZ and Process Commonality

The first step in using the TRIZ algorithm is the formation of a contradiction. In many problems, forming a contradiction requires considerable study. In this case, the contradiction is quite apparent and the task is to generate alternatives for architecting a common process that serves both common and diverse project development needs. In other words, the contradiction is that the process must be both common and diverse.

We can take two approaches in applying TRIZ to this problem. One is analogous application in each step of the algorithm, the other is use of TRIZ application to related fields. Both are discussed here for the sake of illustration.

Starting with the 39 engineering parameters of the original TRIZ (Rantanan and Domb 2002), we look for parameters that can describe process commonality and diversity analogously. Though multiple parameters may apply, two are chosen here. The “adaptability or versatility” parameter is closely related to diversity. Finding an analogy for the commonality parameter is more challenging. However, if we think in terms of the dimension of a number of process user groups, we are concerned about how many of them are covered by process commonality. Considering a measure of coverage in one dimension suggests a “breadth” or “length” parameter. Length appears in two of the 39 parameters, for stationary and moving objects. A documented process can be treated as a relatively stationary object. Thus we have a contradiction between the “length of stationary object” parameter and the “adaptability or versatility” parameter.

The contradiction matrix relates conflicting parameters to inventive principles by citing the principles that have been used most often to resolve parameter conflicts. The matrix places the parameter being improved on the vertical axis and the worsening parameter on the horizontal axis. Table 1 is a section of the contradiction matrix. Considering process breadth/length as the improving feature and adaptability as the worsening feature, inventive principles 1 and 35 are suggested (see the Appendix A.2 for a list of the principles). Table 2 contains descriptions of these two principles, examples of their application (Domb 1997), and questions that suggest ways of resolving the process commonality-diversity contradiction. The suggestions for applying the inventive principles to the process commonality/diversity problem are phrased as questions because they are intended to elicit responses from those who understand the diverse processes being considered as well as the organizational needs for process commonality.

Table 1. Section of Contraction Matrix

<table>
<thead>
<tr>
<th>Worsening \ Improving</th>
<th>34. Ease of repair</th>
<th>35. Adaptability or versatility</th>
<th>36. Device complexity</th>
</tr>
</thead>
<tbody>
<tr>
<td>3. Length of moving object</td>
<td>1, 28, 10</td>
<td>14, 15, 1, 16</td>
<td>1, 19, 26, 24</td>
</tr>
<tr>
<td>4. Length of stationary object</td>
<td>3</td>
<td>1, 35</td>
<td>1, 26</td>
</tr>
<tr>
<td>5. Area of moving object</td>
<td>15, 13, 10, 1</td>
<td>15, 30</td>
<td>14, 1, 13</td>
</tr>
</tbody>
</table>

Another approach to using TRIZ is proposed by Ruchti and Livotov (2001). They adapted the parameters to business and management, identified twelve pairs describing organizational contradictions, and have related them to the inventive principles. Two of the pairs that relate closely to the commonality-diversity problem are Homogeneity-Diversity and Standardization-Specialization, described in Table 3.

TRIZ inventive principles vary in their utility for particular problems. Three of the principles cited in Table 3 are applicable to this problem and are elaborated in Table 4.

The selection of inventive principles in Tables 2 and 4 is based on analogous use of TRIZ and on an application of TRIZ to business management, respectively. Since these are adaptations of the original, physics-based TRIZ, they may not cover all the applicable principles. Sometimes when considering a new field of application, it is helpful to browse the list of Inventive Principles (see Appendix A.2), consider physical examples the principles, and think of direct analogies to the physics-based principles. Table 5 is the result of
such an exercise. This table includes examples suggested by the process questions.

Table 2. TRIZ Inventive Principles 1 and 35

<table>
<thead>
<tr>
<th>Question</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Segmentation. Divide an object into independent parts or sections or segments.</td>
<td>Modular furniture allows for various living room configurations.</td>
</tr>
<tr>
<td>5. Avoid ‘gaps’ (personnel, information, finances etc.) in systems or organizations.</td>
<td>Immediately fill every available ‘gap’ with up-to-date information, tasks, etc.</td>
</tr>
<tr>
<td>3. Remov...</td>
<td>Vulcanizing rubber makes it more flexible and durable.</td>
</tr>
</tbody>
</table>

Questions for application to process commonality:
- Can a process be modularized by activity such that the process can be adapted for each project by choosing appropriate modules?
- Can the process be segmented into practices, application notes, templates, examples, checklists, and so forth that provide varying degrees of specificity and adaptability?
- Can process templates be modularized so that only applicable parts are utilized?

Examples:
- To facilitate the production of liquid-filled candies, freeze the centers before dipping them in melted chocolate. (state change)
- To make the dispensing of liquid hand soap more economical, increase its concentration. (density change)
- Vulcanizing rubber makes it more flexible and durable. (flexibility change)

Questions for application to process commonality:
- Can a process be documented in one form (for example, project-specific templates) and should be individually matched to the local requirements?
- Can a concise, concentrated process description be delivered to all projects and then expanded by each project to fit specific needs?

The questions for applying inventive principles to process commonality/diversity can be used to develop an organizing or architectural principle for a common process. For example, Principle 1, Segmentation, may suggest a layered process architecture, such as a process description for each activity common to every project, accompanied by an application note that describe how the activity can be tailored. Using peer reviews as a concrete example, the process description might discuss the purposes of peer reviews, the entry and exit criteria for a peer review, and types of peer reviews with general guidelines for the applicability of each type. The corresponding application note could detail the procedure for each type of peer review, the metrics collected for each, and the detailed guidelines for using each.

Though one principle might be considered the primary architectural one, other principles may also be utilized. For example, a layered process could also be packaged for familiarity as suggested by the Homogeneity principle (# 33).

Table 3. Business and Management Contradictions (Ruchti and Livottov 2001)

| Homogeneity – Diversity. Change from homogeneous structures, systems or environments to compound structures, and dissimilar systems or environments. Reduce the degree of diversity in the structures, systems or the environment. |
|---|---|
| Recommendation | 1. The different components are to perform different functions and should be individually matched to the local requirements. |
| 2. Minimize losses (cash, personnel, energy) during restructuring or reorganization processes. |
| 3. Remove all “foreign bodies” from the systems and organizations wherever possible. |
| 4. Try to remove the unwanted interaction between two elements of a system not by introducing foreign “separating substances” but rather by modifying the existing resources. Every system defends itself against “foreign” elements. |
| 5. Avoid ‘gaps’ (personnel, information, finances etc.) in systems or organizations. Immediately fill every available ‘gap’ with up-to-date information, tasks, etc. |

| Standardization – Specialization. Use more standardized processes, procedures, methods and products. Gain an advantage by utilizing special processes, products or methods. |
|---|---|
| Recommendation | 1. Replace an expensive, purpose-built system by an assortment of cheap and easily available components. Use recyclable disposable products. |
| 2. Make the application of additional systems and processes redundant by creating a universal system or procedure that can perform several different functions. |
| 3. Repetitive actions should be carried out with standard procedures but allow originality and individuality within the creative fields of activity. |
| Related technical TRIZ Principles | 6. Multifunctionality; 27. Disposable Objects |

5.3 Examples

To demonstrate the application of the inventive principles to the process commonality/diversity problem, this section briefly describes cases from the software engineering literature that exemplify specific inventive principles.
Table 4. TRIZ Inventive Principles 3, 6, and 33

<table>
<thead>
<tr>
<th>Principle</th>
<th>Description</th>
<th>Examples</th>
</tr>
</thead>
</table>
| 3. Local quality | Change an object’s structure or external environment/influence so that the object will have different features or influences in different places or situations. Provide a transition from a homogeneous structure to a heterogeneous structure. Have different parts of an object carry out different functions. Place each part of the object under conditions most favorable for its operation. | - A fractionating column uses a temperature gradient to produce different distillates  
- Smoking and non-smoking sections in restaurants  
- Lunch box with separate compartments for hot and cold foods |

Questions for application to process commonality:
- Can a process or parts of a process be defined on a continuum, from rigorous to flexible, or from necessary to advisable?
- Can each part of a common process be made to emphasize a specific set of practices that meets the needs of different classes of projects?

<table>
<thead>
<tr>
<th>Principle</th>
<th>Description</th>
<th>Examples</th>
</tr>
</thead>
</table>
| 6. Multifunctionality | Have the object perform multiple functions, thereby eliminating the need for other objects. | - Child’s car seat converts to a stroller  
- Swiss army knife  
- Cross-functional training  
- “One-stop” shopping stores that provide groceries, financial services, toys, linens, lamps, and so forth |

Questions for application to process commonality:
- Can a process be defined that does everything that is done by diverse processes?

<table>
<thead>
<tr>
<th>Principle</th>
<th>Description</th>
<th>Examples</th>
</tr>
</thead>
</table>
| 33. Homogeneity | Make objects that interact of the same material, or material that has identical properties. | - Ice cream cones and taco shells are edible containers.  
- Pipes carrying abrasive materials may be coated with the materials so that the surface is replenished. |

Questions for application to process commonality:
- Can a common process take advantage of familiarity with diverse predecessor processes by packaging it so that it has the external features common to the diverse processes?

---

Table 5. TRIZ Inventive Principles 7, 16, and 40

<table>
<thead>
<tr>
<th>Principle</th>
<th>Description</th>
<th>Examples</th>
</tr>
</thead>
</table>
| 7. Nesting | Contain an object inside another. | - Set of measuring cups  
- Extending radio antenna  
- Seat belt retraction mechanism |

Questions for application to process commonality:
- Can a common process have levels of abstraction/specificity? Example: local procedures are instantiations of corporate processes.
- Can a process be defined based on a hierarchy of project needs, from basic (common to all) to transcendent (for high-performing projects)? Example: levels of quality assurance discipline, each level assuming the preceding level.

<table>
<thead>
<tr>
<th>Principle</th>
<th>Description</th>
<th>Examples</th>
</tr>
</thead>
</table>
| 16. Partial or excessive action | When a 100% solution cannot be achieved easily, simplify the problem by doing somewhat more or less. | - A paint stencil catches overspray.  
- Perforate paper to make it easy to remove. |

Questions for application to process commonality:
- Can less than a fully defined common process be provided, one that allows projects to add process discipline where needed? Example: corporate process is minimal allowing for local embellishments.
- Can more process definition be provided than is required for any one project and allow projects tailoring choices? Example: a complete organizational process with options for selecting practices for a specific project.

<table>
<thead>
<tr>
<th>Principle</th>
<th>Description</th>
<th>Examples</th>
</tr>
</thead>
</table>
| 40. Composite materials | Change uniform objects to composite objects. | - Composite shingles provide low cost roofing by putting inert materials on top of light-sensitive, durable material.  
- Fiberglass is lightweight, durable, and formable. |

Questions for application to process commonality:
- Can a common process be composed of both moderately effective and highly effective quality-inducing activities that can be combined in proportions pertinent to a specific project? Examples: combine lean practices with early quality assurance practices (may not be regarded as lean).

---

Multifunctionality. Ocampo et al. (2003) describe a process for analyzing the common elements of two processes and producing a reference model based on the commonality. This approach leverages the multifunctionality principle. It ensures that all required process functionality is available in the reference model by starting with the common processes, negotiating process differences, then identifying process gaps and filling them with best practices.

Development of all three components shared a common requirements analysis phase and porting and finalization phase.

Local quality. This principle is exemplified in the application of object-oriented inheritance and polymorphism to exploit commonalities and accommodate variabilities in software products (Coplien et al. 1998). Inheritance and polymorphism can be applied to software process activities by abstracting and exploiting the commonalities while instantiating variants, as in process tailoring.

Both Ginsberg and Quinn (1995) and Fitzgerald et al. (2003) recommend tailoring at two levels, organizational and project. Ginsberg and Quinn recommend listing process elements in a table and populating the table with tailorable attributes for each element, the range for each attribute, and the considerations for selecting values in the range. The tailoring guidelines should grant programs the flexibility to operate efficiently, while preserving the maximum commonality possible. Thus, tailoring limits changes to degrees so that changes of kind are not introduced. Applying the local quality principle dynamically, Fitzgerald et al. describe not only levels of tailoring, but refinements during a project lifecycle.

Segmentation. In the early 1990s, the Motorola Cellular Infrastructure Group implemented process specifications that distinguished the essentials from the details (Smith 1997). The essential process activities are required by each organization in their development procedures. McMahon (2001) describes a mechanism that performs a similar function, the “process freedom line.”

A process freedom line is defined to be the point in the process where a site/organization is free to make process-related decisions. For example, a project level procedure may call for a design document to be produced with specific design artifacts, but may not require a specific document format. I recommend that virtual projects define process freedom lines at the point where products and people must come together across divergent sites/organizations. It is not recommended that the project attempt to dictate how specific sites/organizations accomplish their tasks internally.

The freedom line definition essentially tells each site where they are free to leverage site-specific procedures (that can include site-specific support organizations and company-owned assets) in implementing solutions. This strategy has proven effective at balancing the management of the project's integration risk, while at the same time leveraging the strengths of individual sites/organizations.

In a novel approach to managing process diversity, Beck (2001) advocates using a critical software quality factor to segment processes. In his example, a project produced three types of software and segmented the processes for developing them based on degrees of reliability required.

Parameter changes (transformation). Perry (1996) reported from his experience at AT&T on the problems of developing and deploying a company-wide software development process. He found that there were cases where a common process could not work. So rather than define a common process, his group defined the requirements for best in class software development processes. They also provided example processes that had met the requirements and that were in use in some part of the company. The advantage of this approach is that it allows autonomy while setting goals for individual processes. The problem that had to be avoided was writing the requirements at such a high level that any development process could satisfy them.

Another example of transformation is offered by Jiang et al. (2004). They chose to provide a methodology for defining a process. Their methodology included a process template, process building blocks, and process development guidelines. The template specifies generic activities and their relationships, the building blocks provide the specific activities, and the guidelines are a set of rules for choosing the specific activities.

Composite materials. Baldassarre et al. (2002) propose the use of a repository of process patterns. Composing a process for a specific project involves assessing the operational context of the project and choosing patterns known to be effective in the project’s context.

5.4 Procedure for Designing Process Commonality

This section proposes a procedure for a group charged with developing process commonality in the face of process diversity. This procedure goes as far as choosing an architectural principle for producing process commonality. It assumes that additional, in-depth analysis will be needed to design the process.

1. Identify the group responsible for developing process commonality. The membership should represent the interests of those who currently use diverse processes that will be characterized by process commonality.

2. Review the benefits of process commonality and of process diversity. Develop a consensus as to the priority of the benefits.

3. Use a prioritization matrix (QFD1) to translate the prioritized benefits into prioritized process characteristics. See the Appendix A.1 for a list of desirable process characteristics proposed by Siebel (2003).

4. Use the TRIZ algorithm to identify applicable inventive principles. Use the inventive principles to produce questions and suggestions for process architectures that support commonality and diversity.

5. Use various combinations of the questions and suggestions to describe several distinct process architectures. Exemplify these architectures by applying them to the elements of the current diverse processes.

6. Evaluate the process architectures in a concept selection exercise using the prioritized characteristics from Step 3 as criteria. In the usual manner of this exercise, develop hybrids after each evaluation until a desirable process architecture is produced.
6 PROCESS COMMONALITY ANALYSIS

The existing processes must be analyzed for their commonality and mapped to the selected process architecture. This analysis may be performed concurrently with the architecture selection, or it may be performed afterward. Concurrent analysis may inform the architectural selection exercise. On the other hand, the amount of detailed work required to complete the analysis may distract from the creative thinking required for architectural selection.

The simplest approach to analyzing process commonality comes from the field of social services (Maximus 2003). In an effort to standardize processes in a state social services department, analysts studied business flow diagrams for similarity of overall purposes. Processes were identified as either identical, similar, or unique. Identical processes are those in which there is no variation in the purpose of the process and the functions that are performed. Similar processes are those that have the same overall purpose, but for which there is some variation in the actual functions involved. Unique processes are those in which the overall purpose of the process is required in only one program area.

Coplien et al. (1998) describe the scope, commonality, and variability (SCV) analysis for software products, described by identifies common aspects of a family. SCV analysis, which can also be applied to software processes, results in a document that contains (a) a list of assumptions true for the family members, (b) a list of variabilities with a range of values for each, and (c) the time at which the value is fixed.

Ocampo et al. (2003) describe a combination of manual and semiautomated techniques for comparing two software development processes. The manual comparison uses one or more of three strategies for comparison.

- Compare each process phase-by-phase.
- Level-based analysis looks at the different aggregation levels of each process, following them in a top-down manner.
- Concept-based analysis compares elementary products descriptions, product hierarchies, elementary process descriptions, and process hierarchies.

Using a strategy or combination of strategies, the process engineer compares the processes and produces a table of similarities and differences. The results of this comparison are strongly influenced by the process engineer’s familiarity with the processes and knowledge of their contexts. The analysis is enhanced with a tool that reads the table and applies a set of simple rules that relate product structure, process hierarchy, process structure, and structure compatibility. In the final result, the activities of one process are presented on one axis of a matrix and the activities of the other process on the other axis. Each cell in the matrix has a value indicating the degree of similarity between each pair of activities.

As for the results that a process commonality analysis may yield, Smith (1997) found that the most common process element at Motorola CIG was inspections. This commonality was due to common training, use of a common database for reporting results, and senior management support for the activity.

7 MEASURING PROCESS COMMONALITY

An important element in undertaking an effort as large as developing process commonality from diverse processes is the ability to report progress. The measures described in this section provide a means of demonstrating increasing process commonality.

Bleckner et al. (2003) use a production process commonality metric as part of a suite of metrics for evaluating the effectiveness of mass production of customer-focused products, or mass customization. This commonality metric is a simple ratio of common production processes to all production processes. Adapting this formula to a software process may mean counting process activities rather than production processes.

\[
\text{Process commonality metric} = \frac{\text{Number of common process activities}}{\text{Number of all process activities}}
\]

This formula requires rules for counting process activities. However if an architecture utilizes a common level of abstraction and multiple distinct instantiations, as in the peer reviews example above, then the question arises as to how to count the abstract, common activities versus the instantiated, differing activities. One method would be to average the number of instantiations for each activity. Assuming that each abstraction represents a process activity, a value of 1 represents complete commonality and the value increases with the number of instantiations.

Jiao and Tseng (2000) proposed a process commonality index based on utilization. Utilization is 1 when an item is produced using a process, otherwise it is 0. Summing the utilization of every item produced, whether component or assembly, relative to every available process, then dividing the sum by the number of processes yields a number between 1 and the number of items. A PCI value of 1 means that none of the items has a process in common; a PCI value of the number of items means that all the items are produced with a single set of processes. The PCI can easily be adapted to indicate software process commonality by indicating which products utilize which process activities.

\[
\text{Process Commonality Index (PCI)} = \frac{\sum_{p=1}^{n_p} \sum_{j=1}^{n_d} \lambda_{pj}}{n_p} \quad 1 \leq \text{PCI} \leq n_d
\]

- \(n_p\) total number of processes
- \(P\) the index of a particular process, \(1, 2, \ldots, n_p\)
- \(n_d\) total number of items produced
- \(j\) the index of an item produced, \(1, 2, \ldots, n_d\)
- \(\lambda_{pj}\) a 0-1 variable: 1 when item \(j\) utilizes process \(p\), otherwise 0
8 SUMMARY
This paper provides background information for software process leaders and engineers charged with the responsibility of implementing a common software development process for groups using diverse processes. To this end, the benefits of both process commonality and process diversity have been discussed. Treating the two as contradictory, TRIZ was used to discover applicable inventive principles and suggest ways to apply the principles, both through questions and examples from the software engineering literature. A procedure for selecting a process architectural principle has been proposed, as well as means of analyzing and measuring process commonality. Though a complete set of software process inventive principles is beyond the scope of this paper, reviews of this study suggest that further investigation for developing such a set is warranted.

APPENDIX
A.1 Desirable Factors in Process Commonality
(Siebel 2003)
- Scalability. This mostly affects the software design process. Use good object-oriented structure, avoid global functions.
- Portability. Use of ISO and other standards and minimize the use of non-portable software and hardware functions.
- Generality. Use as few prior assumptions on input and output data as possible, and use abstract concepts during design (e.g. when programming a people tracker, find a layer of abstraction for tracking methods—the system might be used to track animals in the future).
- Interoperability. Use open standards (e.g. XML) for data exchange. This reduces platform dependencies for data, documents, and configuration information.
- Maintainability. Probably the most important point, and connected to several of the above. We recommend the following measures to attain a high level of maintainability: carefully choose, document, and follow a suitable software process to achieve a high software process consistency over time; create all software artifacts (e.g. programmer’s and user manuals) and keep them synchronized with the implementation; do refactoring ‘on the fly’—this can involve incremental reverse engineering using design patterns; use metrics periodically to monitor design quality.

A.2 The 40 Inventive Principles (Altschuller 2000; Rantanen and Domb 2002)
1. Segmentation: divide an object into independent parts.
2. Extraction: extract from an object a part or property either because it is necessary or it does not fit well.
3. Local quality: allow for heterogeneous structure.
4. Asymmetry: increase the degree of asymmetry.
5. Consolidation: consolidate objects in space or in time.
6. Multifunctionality or universality: an object performs multiple functions; extend functionality as needed.
7. Nesting: place an object inside another object or have an object pass through a cavity in another object.
8. Weight compensation: compensate for the weight of an object by combining it with another object that lifts.
10. Prior action: perform required action partially or completely in advance of required readiness.
12. Equipotentiality: change the condition of work such that potential actions are brought to the same level.
13. Do it in reverse: do something the other way around.
14. Spheroidality or curvature increase: replace straight or linear with curved or rotational.
15. Dynamic parts: alter the properties as needed.
16. Partial or Excessive action: achieve more than or less than 100% of a desired effect.
17. Transition to a new dimension: utilize another dimension.
18. Mechanical vibration: utilize oscillation or change the frequency of oscillation.
19. Periodic action: replace continuous action with a periodic action; change the frequency of a periodic action.
20. Continuity of useful action: continue action without breaks; eliminate unnecessary action.
21. Rushing through: perform risky operations at high speed.
22. Convert harm into benefit: utilize harmful factors for a positive effect.
23. Feedback: add feedback in a system; change feedback.
24. Mediator: use an intermediate object.
25. Self-service: make an object serve itself; use waste.
26. Copying: use simple or inexpensive copies.
27. Dispose: replace an expensive object with a cheap one, trading longevity (or other property) for cost.
28. Mechanical substitution: replace mechanical system or interaction with optical, acoustical, thermal, or magnetic
29. Pneumatic or hydraulic constructions: replace solid parts with gas or liquid, particularly for inflation or cushioning.
30. Flexible membranes or thin films: isolate an object from its environment with flexible membranes or thin films.
31. Porous material: add porosity to an object.
32. Changing the color: change the color of an object or its environment; change the degree of translucency.
33. Homogeneity: make interacting objects of the same material or with similar properties.
34. Rejecting and regenerating parts: discard, modify, or restore parts of an object that have fulfilled their function.
35. Transformation of properties: change properties such as physical state, concentration, density, or temperature.
36. Phase transition: use the phenomena of phase change such as changes in volume or temperature.
37. Thermal expansion: use thermal expansion/contraction.
38. Accelerate oxidation: with oxygen-enriched air.
39. Inert environment: use inert environment or neutralizing additives; perform process in a vacuum.
40. Composite materials: replace homogeneous materials with composite ones.
ACKNOWLEDGMENTS
The Aerospace Corporation supported publication of this research.

REFERENCES
Process-Based Project Management and SPI

María Cecilia Bastarrica
CS Department, Universidad de Chile
Santiago, Chile
cecilia@dcc.uchile.cl

Jacqueline Marin
CS Department, Universidad de Chile
Santiago, Chile
jamarin@dcc.uchile.cl

Daniel Perovich
CS Department, Universidad de Chile
Santiago, Chile
dperovic@dcc.uchile.cl

Luis Rioseco
CS Department, Universidad de Chile
Santiago, Chile
rioseo.luis@gmail.com

ABSTRACT

Software process definition and improvement are frequent strategies followed by software companies in order to enhance software quality and boost development productivity. Software processes are used for guiding development teams while executing projects, and also as a basis for project planning and monitoring. There are some tools available for software process definition and a plethora of tools for project management with support for project planning and monitoring. These tools are usually not integrated so project plans are made manually possibly introducing inconsistencies with respect to the process they are based on. Moreover, when project management is not performed using an integrated tool, plan and trace also differ. Measuring process execution is a common path for SPI. To this end, matching tasks in the process, plan and trace is needed to understand where the process fails and how it might be improved. Inconsistencies among these artifacts hinders SPI since manually matching tasks demands a large effort. In this paper we define an approach for process-based project management that reduces these inconsistencies. We present CASPLE, a tool that supports this approach, and we illustrate its application in PowerData, a small Chilean company. A case study of applying the approach and CASPLE in four of our industrial partners is also presented.

CCS CONCEPTS

- Software and its engineering → Software development process management;

KEYWORDS

Software process improvement, project management

1 INTRODUCTION

Software companies define their software development processes in order to guide software project execution avoiding improvisation. Better processes tend to yield higher quality software products and more predictive and productive projects: fewer errors, less rework, shorter development time, and improved public image for the company [22].

There are many tools available for software process specification which allow the process engineer to create, edit, manage and publish software processes and their elements. Rigorously defining a software process also allows for analyzing and evaluating it. For example, identifying waste artifacts allows process engineers to take them out of the process so that effort is not spared in useless tasks. Similarly, activities where there are no responsible roles assigned may result in nobody performing them. These and other common error patterns [16] cannot be identified unless the process is formally specified.

Software Process Improvement (SPI) is a means for improving quality and time-to-market, knowledge democratization and participation, as well as customer involvement [34]. SPI programs are advised to be based on tools for both process definition and maintenance, and project measurement [13]. SPI may follow the strategy of implementing process areas such as those suggested by CMMI or ISO. However, it can also be based on process enactment [14], by monitoring project execution and recording project traces. In this way we can identify the activities that deviate the most from the plan, or those that never get to be executed as they should because required resources are never available. For example, software design always starts before the final requirements document is signed by all parties, otherwise the deadlines cannot be reached even though the defined process indicates otherwise.

A software development process is expected to guide project planning and execution. While the project manager tailors the process to accommodate the project-specific context, the project team relies on the process as guidance for their job. Companies use project management tools for planning, monitoring progress, simplifying reports, and providing visibility of project status, allowing project managers to maintain control and thus make decisions based on actual execution data.

Although there are a plethora of tools for process specification and project planning and monitoring, to the best of our knowledge there are no tools that integrate all of these views. Therefore, even in the best scenario where a software company uses tools for process
specification and project management, execution trace data cannot be used directly for SPI [28].

We have found this situation in several Chilean software companies. Even though some of them have their processes formally specified and they rely on project management tools, the lack of an integrated tool forces project managers to create project plans manually or by replicating a template that is manually synchronized with the process specification. This task is error prone, and we have found that the activities in project plans and project traces tend to differ significantly from those specified in the process. There are some tools that help generating an initial project plan, but they are still not integrated with monitoring tools. This hinders software process improvement, as identifying a correspondence between activities in the process, in the plans and in the traces is not a simple task.

In this paper we propose an integrated approach and supporting tool for process definition and project management that enables SPI. The tool is developed integrating well established open source tools so that adoption is made easier. We detail the whole approach and we present CASPLE, the tool that supports it. We illustrate their application in the case of PowerData both, as a motivation and as a running example. We also present a case study studying the approach and tool in four of our industrial partners.

The rest of the paper is structured as follows. Section 2 presents some related work. The problem motivation is described in Sec. 3 where we present the case of PowerData. The proposed approach is described in Sec. 4 along with the CASPLE tool. Section 5 presents the application case study. Finally, some conclusions are presented in Sec. 6.

2 RELATED WORK

Several studies and proposals have been conducted for the improvement of software processes during the last decade. SPI increases competitiveness and productivity, reduces defects, costs and rework, as well as improves customer satisfaction.

The specification of software processes requires languages that allow us to comprehensively describe the complex nature of the software. Works such as [10, 11, 32] present ISO/IEC 24744 [17] and SPEM 2.0 [25] as the most widely used languages for the software domain. Essence 1.0 is compared to SPEM 2.0 in [8] and the authors find it even more appropriate for certain agile environments.

Formalizing the process in a particular language requires supporting tools. Tools like EPF Composer and Rational Method Composer (RMC) facilitate the creation, edition, adaptation, configuration and publication of software processes. Several research works like [2–4] describe the advantages and extended use of SPEM 2.0 and EPF Composer in the industry. In our work we use EPF Composer; it is a free tool that was well received by our industrial partners that are all small software companies.

Lately, tools specially designed for modeling business processes such as BonitaSoft or Bizagi have also been used for defining software processes, even though they present serious limitations mainly for modeling multi-role activities [5] that are essential in software development.

Initiatives such as NDTQ-framework have been created for the adoption of specific processes and/or methodologies for software development. It proposes a language and a supporting tool [10]. Although the authors claim that the framework is flexible, it has been conceived to support their own methodology. As our work is based on EPF Composer that implements SPEM, it does not present such limitation.

Fazal et. al [9] present an approach for process-based project management based on situational method engineering. It is similar to our approach though we address tailoring following a software process line strategy [15].

The support for process execution is another aspect related to SPI. Project planning, monitoring and control through project management tools like Redmine, Microsoft (MS) Project or Process Dashboard, among others, provide real-time data on project performance and progress. These traces of the execution facilitate the evaluation of the process and an efficient decision making. We follow this strategy in our approach, though we do not provide support for automatic process execution.

Redmine is a free tool that includes Gantt charts to aid visual representation of project deadlines and facilitate the monitoring. It is an open source project management tool, and supports a variety of plugins that can be added for enhancing it. It can also be customized and configured according to specifications and industrial needs [1, 23, 37]. MS Project is a proprietary tool that allows for planning and monitoring projects. Despite its wide use, scientific literature about its application for managing software projects is scarce. dotProject is a very simple open source project planning tool; it does not provide monitoring features. On the other hand, Process Dashboard is also an open source tool that allows the data collection of times, defects, project size, tracking developed projects under PSP and TSP standards. In contrast to our work, this tool is only suitable for dealing with specific processes.

Works such as [20] describe the usefulness of the Application Lifecycle Management (ALM) tools in support of SPI. Tools like Rational Team Concert (RTC) and Team Foundation Server (TFS) facilitate the execution of agile or formal projects with control and planning tools and process templates, e.g. Kanban [24], Scrum [30] and CMMI [31]. Despite of the advantages of these tools, they only allow the adaptation of predefined processes.

ALM tools applicability can be extended to be used in other software processes through their integration with another tools. In [26] is highlighted the integration with RTC and RMC, which facilitates and speeds project planning and work assignment supported by the described process. An alternative for TFS is the integration with Process enactment Tool Framework (PET). In [19], it is described how PET takes an input software development process model and transforms it into an intermediate format that serves as the basis for a second transformation step into data formats understandable for tools such as ALM frameworks. However, this approach does not contribute to process improvement by using project execution traces.

In [7], a tool chain based on eSPEM [6] and integrated in Eclipse is proposed to solve industry-specific problems. This tool chain is composed of three different integrated components that allow for modeling, simulating, deploying, managing and publishing software processes. This project has similar goals to ours, but there are no
In our work we intend to integrate a variety of different tools so that SPI can be addressed from different points of view: process specification, project planning and monitoring, and process discovery, conformance and enhancement.

3 MOTIVATION

In this section we introduce the case of PowerData, a Chilean software company that defined its software process so that company’s results could be improved through better project management, but that faced new problems derived from using non integrated project management tools.

3.1 The Case of PowerData

PowerData is the Chilean branch of a Spanish software consulting company, whose main goal is the development of Extract Transform and Load (ETL) projects. The company has offices in Spain, Argentina, Mexico, Peru and Chile. Its mission is to create value for its customers providing them with high quality and reliable solutions through expert support in data management, so that they could achieve lasting improvements in their business results. The vision is to offer a simple access to the information, minimizing risks and delays, even for heterogeneous and disparate data, in real time, achieving total costs much lower than manual solutions. Each country office is autonomous, although the mission and the vision are common for the whole organization.

In Chile, PowerData counts on 40 employees among administrative, commercial and IT professionals. The core business of PowerData in Chile is selling licenses of a third party company, Informática, for which it is the official representative and only partner in the country. PowerData also sells consulting services in the form of software development projects. To this end, and depending on the size of the project, working teams are formed by consultants that elicit requirements, develop solutions, execute tests and quality control, all with the support of a project manager also in charge of management. PowerData is currently involved in projects relating banking, retail, insurances, utility companies and state-owned companies.

3.2 Process-based Project Management

Between 2013 and 2015, the company lost almost 23% of its earnings in the area of consultancy and, what is even more serious, it lost loyal customers due to not being able to reach the promised goals and deadlines for contracted projects. Bad results brought a series of other consequences such as losses in economic revenue, labor instability and a discouraging work climate that does not favor professional development.

PowerData did not count on a defined software process that was transversely applied within the company, and milestones were not commonly defined either, making it difficult to monitor project progress. As a consequence, deviations were not detected until they were far from the original plan. Also there was low capacity for correctly estimating project expected time and cost because there
was no established methodology for this purpose, e.g., based on similar past projects or historical data.

In order to deal with the identified situation, the company decided to define its software development process, and apply it in two pilot projects in order to measure its quality before making it official for all software development within the company [27]. Figure 1 shows the process breakdown and Figure 2 the high level activity diagram of the software process, both defined using EPF Composer.

PowerData usually uses MS Project for planning projects. So they followed the same approach in the two pilot projects: SIETE and Multichannel Datamart. In these projects, the project manager used the specified process as an inspiration to manually produce the project plans. Figure 3 shows an excerpt of each plan.

When projects were executed, the corresponding project plan was followed by the development team to guide work and by the project manager to monitor the project schedule. To this end, the project manager used Excel spreadsheets for registering the project progress, and thus, recording the project execution trace or log. Figure 4 shows an excerpt of the logs of each project.

3.3 Software Process Improvement

After executing both pilot projects, and having captured measurements and traces on such execution, PowerData aimed to carry out a software process improvement effort. This endeavor took a whole year and they actually obtained some results that allowed them to improve the defined process. We point out some of the findings:

- most delayed tasks during pilot projects were found to be the responsibility of the project manager. For example, the Requirements activity was one of the most delayed in both projects, and the project manager was involved in all the tasks in it. Lowering his responsibilities in the process definition made him less a bottleneck.
- the requirements specification document was defined as an input to too many tasks, and therefore none of them did formally start before the document was completely approved. This constrain was relaxed in order to allow development to start even though negotiation about requirements could have not reached an end.

Manually mapping these delayed project tasks to process tasks allowed Power Data to improve its process. However, this was achieved with a huge human effort. Counting on a tool that can integrate project management with process definition would make this activity easier and less prone to errors.

3.4 The Problem

The benefits of the software process improvement effort, however, had a huge cost for PowerData. Even though they had captured relevant data about the process plan and execution, such data presented a lot of inconsistencies and using it for detecting improvement opportunities could only be done with an enormous manual effort. Moreover, although some of the findings were apparent when finishing the first pilot project, improvements could not be incorporated into the process for the second pilot because trace analysis had to be postponed given the time it would have taken. The inconsistencies in the captured data have two main causes that we describe next.

On the one hand, manually producing project plans from the process specification, unintentionally or not, the project manager introduced changes both in the names of some of the activities and in their structure and dependencies. For instance, in Figure 3 we can see a series of interesting issues relating tasks corresponding to the Analysis. First, the name of the activity in the process in Figure 1 is Requirements and not Analysis. Second, the set of activities in both projects does not match: the first project involves four tasks while the second involves only three. And also there are tasks that, even though we can assume to be referring to the same activity, they have different names: Create Requirements Book and Create Requirements Document. These are just three examples of inconsistencies introduced in manual planning, but a rapid look at the two plans makes clear that these situations are all over.

On the other hand, the lack of an integrated tool for project planning and for project monitoring also introduced inconsistencies between the activities in the project plans and in the project traces. As PowerData uses MS Project to define the project plan and MS Excel spreadsheets to record project traces, both manually, inconsistencies were introduced by using different names for the same activity and changing the structure of activities. For instance, in Figure 4 shows for example the tasks corresponding to the Analysis activity in the project trace. In the figure, we can see that in the Multichannel Datamart project trace this activity includes a task named Development Environment Preparation that is not a task of the Analysis activity of the project plan, but rather it is an activity by itself with one task, as shown in Figure 3. Also, while the log
recorded the activity Test Plans Preparation including two tasks, the project plan for project SIETE has not such activity but just the tasks, and the project plan for project Multichannel Datamart has an activity named Test Cases instead.

Other subtle inconsistencies such as differences in lower-upper case in the names of the tasks show that the mapping was performed manually, and could have included even more serious errors than it did. All these errors and mismatches between artifacts are also common in other software development companies.

4 PROCESS, MANAGEMENT AND SPI

In order to deal with the kind of inconsistency problems described in the previous section, we propose an approach to integrate different tools to process definition, project management and SPI. In this section we describe the proposed approach and the tool that has been implemented so far.

4.1 Integrating Tools

Figure 5 describes the envisioned roadmap for the complete integrated proposal for SPI based on process-based project management.

The goal is to integrate, within a unique framework, a series of new tools with others that software companies are already using for different purposes, but articulating them so that redundant work can be avoided and inconsistencies are not introduced or at least reduced to a minimum. Whenever possible, already available tools were integrated so that industrial adoption could be easier. In other
cases, research tools are being developed as part of the integrated approach.

4.1.1 Process Specification and Validation. The process must be formally specified so that it can be automatically analyzed and manipulated. Tools such as EPF Composer or IBM Rational Method Composer can be used to this end.

Static analysis can be performed so that simple syntactic and semantic errors can be identified. We have seen that it is quite common in practice to introduce some underspecifications such as not indicating a responsible role for a task or not specifying its input or output work product. Other underspecifications may undermine the correct execution of the process, even though they are not errors in themselves, e.g., not providing templates for certain work products or guidance for a role. We have developed a tool -AVISPA [16]- that has been quite successful for process static analysis. PowerData applied AVISPA for analyzing its process and was able to identify some of the aforementioned problems that were lately confirmed as part of the pilot experiences [27]. AVISPA has proven to be a good predictor for process inefficiencies. The tool is currently undergoing a reengineering process so that it can be smoothly integrated into the framework.

4.1.2 Process Tailoring. Process tailoring is the activity of adjusting the process to the particularities of each project context [18]. Several approaches may be followed, from having a specific process defined for each project, to a general process that is adjusted, either manually or automatically, according to the project context. Some companies also define a series of processes, each one targeting a type of project, and they are selected accordingly each time. All strategies may be useful. However, in order to make SPI feasible, all particular processes may be members of a software process line [36]. We have developed a process selector [12] and a MDE-based tailoring tool [35] that will be integrated with this purpose.

4.1.3 Initial Project Plan Generation. Once the appropriate process for the project is tailored, the project plan is automatically generated. In this way, all the tasks, and only those tasks included in the process form part of the plan. The initial plan is built following the dependencies established in the process specification, but without duration.

4.1.4 Project Task Duration. Task duration is then decided based on the specific scope of the product to be developed, helped by the historical data recorded for projects with similar development contexts and scenarios. In this stage the project manager can adjust the task duration plan before starting as well as adding some tasks specifically planned for this particular project. This plan is used as a basis for measuring project progress and deviation.

4.1.5 Project Execution and Monitoring. As the software project is executed, every finished task is registered in the form of an event with two different purposes: project monitoring and SPI. In the short term, monitoring is intended to identify deviations from the project plan as soon as possible. Similar to the strategy we followed for project planning, project monitoring is also performed in external tools, but we also built a dashboard that allows the project manager to monitor the complete company project portfolio.

4.1.6 Software Process Improvement. Project traces are also to be stored for SPI for further process mining. We have developed the v-algorithm [29] that is able to discover the software process line from a set of project traces. Provided that these traces are registered according to tasks in the plan, and these correspond to tasks in the process, they can be automatically mapped back for process conformance and SPI. This spot is an active research within our group.

4.2 The CASPLE Tool

CASPLE\(^1\) is the first implementation of our proposed approach. Figure 6 describes the portions that are already integrated (black), those that are currently developed but not yet integrated (blue), and those that are still research tools (red). Figure 7 shows the concrete external tools that are integrated so far.

4.2.1 Process Definition and Versioning. CASPLE acts as a software process repository for companies, and it is able to keep track of multiple process versions. We have built a plugin for EPF Composer that allows a process engineer in a company to retrieve any

of the stored process versions, use EPF Composer to update it, and to publish new versions in CASPLE.

4.2.2 Project Management. CASPLE is not a project management tool in itself. Since most software companies use tools for project planning, we adopted the strategy of not building yet another tool for project planning and monitoring, but to integrate with several existing tools. In this way, we have already integrated with Redmine [21], and we are currently also working on integrating with MS Project.

However, CASPLE captures project management information from the external tools it is integrated with. By now, CASPLE records information regarding the project plan and the project execution traces. To this end, a project manager in a company can create new projects in CASPLE based on a specific version of a process definition published in CASPLE by the company’s process engineer.

4.2.3 Project Plan Generation. Once the project is created in CASPLE, the project manager can generate an initial project plan. CASPLE creates the project in Redmine, creates a task (specifically a Redmine issue in a tracker named Tasks), preserving the nested structure of activities in the Work Breakdown Structure defined in the process and the precedence relationships between these activities.

Then, the project manager adjusts the project plan and assigns a duration and a responsible team member for each task. Figure 8 shows the project plan for the Multichannel project of PowerData using the process definition shown in Figure 1. The tasks in the project plan were automatically generated by CASPLE, and the task duration and responsible were stated by the project manager. We can notice that the activities and tasks included in the project plan have the same names, structure and precedence as those stated in the process definition.

Once the project plan is finished, CASPLE automatically imports it from Redmine, preserving it as the baseline for the project. By recording the project plan, CASPLE can then compare the process and the plan.

4.2.4 Project Monitoring. Redmine allows developers to register events whenever a new task is completed thus building a project trace. CASPLE is then able to retrieve these traces, compare actual and planned task duration, and display warnings in the dashboard whenever appropriate. Figure 9 shows the a dashboard in the CASPLE tool for the Multichannel Datamart project, which compares the process, the plan and the trace.

4.3 Applying CASPLE

In order to validate our tool in the context of a software company that had carried out a SPI effort without the appropriate tool support, we visited PowerData. The visit consisted of a two-hour meeting between members of our team and the process engineering and a project manager of PowerData. The meeting involved a tool demonstration and an structured interview. The tool demonstration consisted of a brief introduction to CASPLE’s envisioned roadmap and the current support, and of the execution of a complete user history using the tool, from publishing the process definition from EPF Composer, to generating the initial project plan, to monitoring the deviation between the process, plan and trace using the dashboard. After the demonstration, we performed an interview asking whether using an integrated tool for project planning and monitoring would be beneficial, and whether they would adopt a tool such as CASPLE.

The project manager agreed that it would have been highly beneficial to count on an integrated tool for project management, instead of relying on MS Project for project planning and on MS
We also asked them about their perception on the feasibility of PowerData adopting a tool such as CASPLE. They both agree that the most salient characteristic that makes CASPLE easily adopted is the fact that integrates already existing tools, and mainly one that looked promising - Redmine - with respect to the tools in their current practice. Also, and provided that it is a tool intended for small or medium size companies, they agreed that once managers acknowledge its value, it is not too difficult to align software professionals in using it.

The process engineer added that, a process-based project management approach as the one we proposed, with the appropriate tool support like CASPLE, would have significantly reduced the effort of analyzing the project logs to detect opportunities for improvement when they executed the pilot projects.

The validation with PowerData showed us that there is a perceived value for both our approach and our tool for performing SPI in small and medium software companies, which is encouraging to continue integrating additional tools to extend the current support.

### 5 CASE STUDY

We conducted a case study [33] to explore the practical applicability and usefulness of our approach and its supporting tool.

#### 5.1 Case Design

The main objective of the case study was to capture feedback in order to set a roadmap for the evolution of the approach and tool. To this end, we stated two research questions:

- **RQ1** Which are the perceived advantages and shortcomings of the approach and tool to apply them in practice?
- **RQ2** Which are the perceived limitations for adopting the approach and tool in practice?

The case study was conducted in small software enterprises in Chile, specifically in four of our industrial partners, namely Ki Teknology, Amisoft, Arbol Logika and Mobius. These companies have their development process defined, and they use tools for project planning and monitoring.

Data collection involved a structured questionnaire, a tool demonstration, and a non-structured interview with the CEO or COO along with a technical professional of each company. The goal of the questionnaire was to capture the actual and the desired practices regarding process-based project management and SPI. The tool demonstration was analogous to that presented to PowerData. After the demonstration, we performed an interview asking about the practical use of the tool and their disposition to adopt it. The interview aimed also to capture general recommendations regarding the tool.

#### 5.2 Results

Three of the partners have their processes specified in EPF Composer and one of them uses textual documents. Each company uses the same tool for planning and monitoring, unlike PowerData, but all of them use different tools, and even one of them uses a tool developed in house. They follow different approaches for generating project plans: either using the defined process or the project plans for previous projects. However, in all of these companies, project plans are generated manually. All partners claimed that automating the generation of the project plan from the process definition is desirable. However, none of them had an approach or tool to achieve it. They all emphasize that the integration of CASPLE with the tools already in use is highly desirable and adoption would be very difficult otherwise.

Ki Teknology and Amisoft were strongly interested in the adoption of CASPLE to complement the functionality provided by the tools they already use. As Ki Teknology uses Redmine, the only limitation they remarked is the support for process tailoring in the tool. In the case of Amisoft, while tailoring was not a limitation, it was the integration to the project management tool developed in-house. Mobius and Arbol Logika recognized the usefulness of the tool. However, they stated that adopting the tool would be difficult for them. Mobius has a defined process and uses Jira for project management, but stated that the company operation has a continuous and high demand of effort from the teams which would hinder the adoption of a new tool. Arbol Logika has a defined process and an ISO certification, but, as it is not formalized in EPF Composer, generating project plans from the process would not be possible.

Excel spreadsheets for recording project logs. Moreover, the project manager claimed that just counting on Redmine, and with no automatic plan generation, would have been an improvement for managing the projects. However, the process engineer insisted that the alignment of the three artifacts is the most valuable contribution because it is the only way they can count on actual data for SPI.
using our tool, at least for the moment. Provided that Arbol Logika is small - they have a development team of 8 members -, planning and monitoring is simple for them and they did not perceive an important benefit by introducing a tool like CASPLE. However, they recognized that, if the company grows, such tool would be very useful.

5.3 Analysis

The case study showed that our approach and tool are welcomed by the small software companies we consulted, finding it useful in practice but requiring extended support to fit their scenarios.

With respect to RQ1, the greatest perceived value of the tool is its ability to automatically generate project plans, to preserve historical information of project plans and traces, and to compare them with the process. The major benefit perceived of the approach is that it integrates the various stages of SPI and provides tool support for them, as oppose to the current practice in these companies where SPI, if done, is ad hoc and lacks historical information. Also, the strategic decision of integrating CASPLE with existing tools, instead of replacing them, was backed up unanimously. However, the main shortcoming reported is that current integration is limited to EPF Composer and Redmine. Also, the fact that a formal specification of the process is a prerequisite for the tool is seen as a disadvantage, as ISO certification does not require such level of formality.

With respect to RQ2, while the benefits of the approach and tool were recognized by all of these companies, they identified some limitations that hinder their adoption. First, the approach demands additional effort from managers and developers which may not be practical for micro-entreprises with fewer than 10 employees, or for small and medium companies were the normal operation demands the whole personnel full time. Second, as process tailoring, discovery and enhancement are not yet supported by the tool, the full potential of the approach cannot be achieved yet. Also, the current integration with few tools is seen as a limitation.

The case study allowed us to set a roadmap for the evolution of the approach and tool, mainly, to keep focusing in SPI while complementing current tool support used in industry. Thus, we set the following future goals:

- To improve the coverage of the tool with respect to SPI stages. Prototypes and ongoing research are being developed in our group.
- To improve the integration with other tools.

5.4 Threats to Validity

Internal validity. Even though those involved in the survey were two high rank people in each company, the actual adoption will depend on the whole team working at that company.

External validity. The case study was conducted in small software companies that already have their processes defined. We cannot generalize the results for small companies that lack defined processes or large companies. Also results cannot be generalized to companies that do not plan whole projects up front, e.g., companies that follow agile practices.

Construct validity. Both, the questionnaire and the interviews where designed for directly answering the research questions, therefore we think the case study presents a high construct validity.

Reliability. The case study was conducted by members of the team that developed the approach and the tool, so they may have introduced bias. If other people would have performed the case study, results could have been different.

6 CONCLUSIONS

In industry, the effort of formally specifying the software development process provides concrete benefits in the short term. However, enacting these process specifications in software development projects makes companies realize that the prescriptive processes are not always the actual processes they follow in practice. As a consequence, these companies start SPI projects to align the specified process and the practice.

We have found, however, that most SPI projects are hard to adopt, mainly due to the lack of accurate data regarding project plans and project execution traces. Some companies do not use purpose-specific tools for project management, and rely on spreadsheets for project planning and progress monitoring. Some other companies use project management tools and thus they capture project plans and traces. In this latter scenario, the project manager manually generates the project plans inspired in the activities specified in the process. However, this manual task not only is error prone, but also allows for improvisation. While a degree of freedom is required to accommodate the process to the project context, such freedom is usually abused: activities names are changed, dependencies are changed or missing, sub-activities are misplaced, new concrete project-specific activities are introduced at the top level instead of refining existing activities already defined in the process (such as “Design the solution”), etc. We showed a real-world example from PowerData, a Chilean software company that underwent a SPI effort and faced the complexity introduced by the lack of accurate data that required a enormous effort to match activities in the process, the plans and the traces.

We claim that automating the generation of project plans directly from the process specification provides the project manager a solid starting point for the project plans, yielding more accurate data in plans and traces that simplify matching the recorded activities to those specified in the process. To support this claim, and hence to deal with the identified problems, in this paper we presented an approach for process-based software project management that facilitates SPI. This approach is based on coordinating the process, the project plan and monitoring, process discovery and improvement, all of it supported by tools. CASPLE is the tool that supports the whole approach. Only part of it is already publicly available, but most other parts are under development and are planned to be integrated in the near future.

Counting just on the parts of CASPLE that are available, concrete gains have already been achieved. While companies rely on their project management tools for project monitoring, evaluating the deviation of the project execution with respect to the project plan, CASPLE allows for monitoring the deviation of the project plan and execution with respect to the process. This allows project managers
to make more informed decisions. Also, the process engineer counts on more accurate data to detect where the deviations take place.

We have assisted small Chilean software companies in specifying their processes during the last seven years. Recently, we extended our effort and tools to cope with software process improvement as such need arose in our industrial partners. The approach and tool we presented in this paper is mainly aimed to address this need. We validated our approach and tool with PowerData and other industrial partners, obtaining as a result that both our approach and tool are useful in practice for small and medium companies with a defined process that aim to SPI efforts, and most of them are willing to adopt them.

ACKNOWLEDGMENTS

The work of Jacqueline Marín was partly funded by Conicyt, Chile.

REFERENCES


Impact of Task Switching and Work Interruptions on Software Development Processes

Alexey Tregubov  
University of Southern California  
941 W 37 Pl, SAL 327  
Los Angeles, California 90089-0781, USA  
tregubov@usc.edu

Barry Boehm  
University of Southern California  
941 W 37 Pl, SAL 328  
Los Angeles, California 90089-0781, USA  
boehm@usc.edu

Natalia Rodchenko  
University of Southern California  
1050 Childs Way, RRI 316B  
Los Angeles, California 90089-0781, USA  
rodchenk@usc.edu

Jo Ann Lane  
San Diego State University  
5500 Campanile Drive  
San Diego, California 92182-7720, USA  
jalane@mail.sdsu.edu

ABSTRACT

Software developers often work on multiple projects and tasks throughout a work day, which may affect their productivity and quality of work. Knowing how working on several projects at a time affects productivity can improve cost and schedule estimations. It also can provide additional insights for better work scheduling and the development process. We want to achieve a better productivity without losing the benefits of work interruptions and multitasking for developers involved in the process. To understand how the development process can be improved, first, we identify work interruptions that mostly have a negative effect on productivity, second, we need to quantitatively evaluate impact of multitasking (task switching, work context switching) and work interruptions on productivity. In this research we study cross-project multitasking among the developers working on multiple projects in an educational setting. We propose a way to evaluate the number of cross-project interruptions among software developers using self-reported work logs. This paper describes the research that found: a) software developers involved in two or more projects on average spend 17% of their development effort on cross-project interruptions, b) the amount of effort spent on interruptions is overestimated by the G. Weinberg’s heuristic, c) the correlation between the number of projects and effort spent by developers on cross-project interruptions is relatively weak, and d) there is strong correlation between the number of projects and the number of interruptions developers reported.

KEYWORDS

Work interruptions, effort estimation, multitasking overhead, cost estimation, software development

ACM Reference format:

1 INTRODUCTION

Many have observed that software developers need to switch between different tasks quite often throughout a work day [1]. Parmin and Rugaber [1] studied 10,000 sessions of developers’ work and concluded that work interruptions and task resumptions are a frequent problem for developers. Another study [7] found that information-technology workers spent an average of only 3 minutes per task before switching to another task. DeMarco and Lister [3] found that developers work only 30% of their time alone and spend another 70% on collaboration and interaction with each other. Often such collaboration introduces work interruptions and requires developers to work on several projects at a time. In general, working on multiple tasks at a time and switching between them can have both positive and negative effects.

In order to better estimate development costs and achieve a better productivity we want to know the negative impact of work interruptions. We chose to study cross-project interruptions, because this type of interruptions introduces a time-consuming context switching which is often viewed as a waste of time. To achieve a better productivity cross-project interruptions can be reduced in development processes (e.g. Lean, Agile processes).

In this research, we quantitatively evaluated the impact of cross-project context switching and work interruptions on cost overhead in several software development projects in an educational setting. We used observational data collected from developers’ self-reported work logs.

We answer two research questions in this study: (1) whether there is a linear correlation between the number of cross-project work interruptions and the number of projects and (2) whether
there is a linear correlation between the amount of time spent on cross-project interruptions and the number of projects involved.

2 BACKGROUND

Dzubak [4] defines multitasking as the engagement in individual tasks that are performed sequentially and implies that there is necessarily some time spent switching between tasks. In other words, parallel tasks interrupt each other.

Jett and George [5] distinguish four major types of interruptions: intrusion, break, distraction, discrepancy. All of them have positive and negative effects on productivity; different resumptions time, different time of switching to a new work context, different underlining causes and sources of interruptions.

Positive effects of work interruptions in software development include a better team collaboration and communication (e.g. interruptions for design/peer review and architecture review meetings, which help developers to define a “big picture” of the project and obtain a feedback from their colleagues [4]), a stimulating work environment (e.g. reduced “the boredom effect” and the inclination to procrastinate [4]), a reduced “struggle time” trying to solve complex problems for too long [6]. Positive interruptions are typically related to software design work, problem solving, and debugging activities that often depend upon the creation and evolution of mental models. These mental models are what facilitate concurrent engineering and development.

Along with positive effects, studies [7-8] found that multitasking can reduce work productivity. In particular, interruptions introduce necessary time spent switching between the tasks, called “attention switching” in [8] and make an individual less productive by reducing the amount of short-term memory dedicated to a single task. Salvucci et al. [9] performed a set of experiments and observations to measure the time cost of task switching and determined that it depends on the complexity of the task, time length between task switches, and causes of interruptions (e.g. external or self-inflicted interruptions).

DeMarco and Lister [7] introduced a metric called reimmersion time as a measure of the effect of task interruption. Reimmersion time is an extra effort to complete the task after an interruption (Fig. 1). This metric is also known as resumption lag [1, 7, 9, 10]. G. Weinberg [11] introduced a heuristic measure of the effect of multitasking on reimmersion time that is a simple linear function of the number of projects assigned to an individual at the same time (see Fig. 6). The heuristic predicts a 100% increase of multitasking overhead for every additional task assigned.

In this paper, we study only cross-project interruptions and cross-project multitasking. We define cross-project multitasking as an involvement of software developers in multiple engineering activities with different or minimally overlapping contexts over a certain period. Cross-project interruptions are characterized by relatively long reimmersion times, longer resumption times, and significantly different work contexts of interrupting tasks [9]. Often such interruptions are external (not self-inflicted); therefore, could be limited by the development process (e.g. Agile and Lean development processes acknowledge the problem and introduce measures to reduce work interruptions [12, 13]).

Cross-project multitasking and interruptions may appear in different forms. A few examples are:

- Developers are often shared between projects in organizations with matrix structure
- Developers are shared between multiple releases within a single project (if several releases are maintained)
- In System of Systems environments, developers are often shared between several contexts such as customer-specific requirements, software releases for clients, etc.[13].

The focus of the rest of this paper is on negative productivity impacts of cross-project multitasking and interruptions.

3 CROSS-PROJECT MULTITASKING AND INTERRUPTIONS

3.1 Research Questions

In this paper, we measure a negative impact of cross-project interruptions as the total effort (person-hours) each developer spends on reimmersion time after each interruption that was caused by tasks from other activities.

The following research questions are addressed in the analysis below: is there a linear correlation between the number of projects each developer was involved with and

- the number of cross-project interruptions they experience?
- the amount of effort spent on cross-project interruptions?

There were two types of hypotheses tested for each question:

H.1 Correlation between subject means: developers working on more projects tend to experience more cross-project (a) interruptions, (b) spent more effort.

H.2 Correlation within subjects: an increase of the number of projects per subject is associated with an (a) increase of the number of cross-project interruptions, (b) increase of time spent on interruptions.

Additionally, we also compared the effort evaluation results with the predictions of G. Weinberg’s heuristic (Fig. 6).

3.2 Data Collection

In order to evaluate the impact of cross-project multitasking, we observed 10 software projects for three months. All projects were developed as part of a graduate level software engineering class at the University of Southern California. Clients of these projects represent various non-profit organizations, entrepreneurs and private companies. Clients joined teams of students of software engineering class to work on their projects. Most of the projects were scoped
We used reimmersion time as described in Fig. 1 to evaluate the impact of project progress and reporting project progress is part of the class curriculum.

In total, we collected work logs and weekly progress reports of 68 students for one semester of work. All students had degrees in computer science related fields and experience in software development.

Daily work logs were part of the work process, which allowed us to collect data on development effort and acquire development effort distribution across projects and individuals without interfering with their work flow. Project tasks were assigned to developers via the project tracking system used in the class. Every student reported their daily and weekly progress (time spent) on each task, that he/she performed on that day. In addition to the work logs, students also provided weekly information about work interruptions and self-evaluated reimmersion time. They were instructed how to evaluate their reimmersion time and how to use the project tracking system.

Some of the students also took one or two other classes and had no outside jobs; therefore, they had to distribute their time between the following three types of activities: (1) a project in the software engineering class, (2) group projects/tasks/assignments in classes other than the software engineering class, (3) individual assignments in the software engineering class and all other classes. These three types of activities have very different context of work, and independent deadlines. We viewed a switching between these three types of activities as a cross-project interruption.

In this study, we collected detailed information about projects in the software engineering class via the project tracking system (Atlassian Jira). The project tracking system tracked tasks’ statuses. A task’s status, reported by developers, can show if it is completed, in progress or in the backlog. A task’s labels also allowed us to distinguish different types of engineering activities (requirement engineering, coding, testing, documentation, etc.).

We collected information about the impact on the class project from the other two types of activities via weekly progress reports, where students provided self-evaluated reimmersion time and types of activities they were involved with throughout the week.

For the purpose of consistent terminology in this paper, we will call the three types of activities listed above as projects (activities with different context of work). We also will count switching between them as cross-project switching.

In order to analyze the impact of interruptions on effort, we selected a subset of 29 full-time students who worked in similar conditions. They only took two classes (one of them was the software engineering class) and had no outside jobs. At least 25% of their time in the class project was dedicated to development activities (programming, bug fixing, etc.).

Work interruptions were counted by analyzing developers’ work logs and weekly progress reports. These two sources of information were used to verify the information about the number of projects students were involved with and the number of interruptions they experienced. Weekly progress reports provide the number of interruptions evaluated by developers, and work logs allow us to see how developers worked on a software engineering class project throughout the week. Since all 29 developers spent their work time only on up to three projects at a time (including the software engineering class project), their work logs of tasks from the software engineering class projects showed when work was interrupted. In some cases work logs can only provide a lower bound estimate on the number of interruptions, because we were only able to track one out of three projects they worked on.

Reimmersion time may depend on many factors: the complexity of the task, length of the interruption, complexity of the other task that interrupted work, individual’s personality (closure-oriented vs. curiosity-oriented), etc. To account for all the factors that impact the reimmersion time we asked developers to evaluate their weekly average reimmersion time for tasks that they did for the class project. The total weekly effort spent on cross-project interruptions (i.e. multitasking overhead) is computed as the total number of interruptions times the reimmersion time.

4 RESULTS

In our observational data we collected repeated measures for the same set of projects over the course of 12 weeks (12 weeks = 12 times); therefore, we had to account for repeated observations in our linear regression analysis. For both effort and interruptions we tested two types of hypotheses: (1) developers working on more projects tend to have more interruptions and tend to spend more effort on multitasking overhead; (2) the increase of the number of projects worked on is associated with increase of the number of interruptions and effort.

To answer the first group of questions we used the linear correlation between subject means (mean values of the number of projects each developer was involved in, effort and the number of cross-project interruptions of each developer). Fig. 2 shows the correlation between the average number of interruptions and the average number of projects per week per developer. The total number of data points is 29. The linear correlation is relatively strong ($R^2 = 0.60699$). It supports the hypothesis H1.a that developers working on more projects tend to experience more interruptions.

In our observational data we collected repeated measures for the same set of projects over the course of 12 weeks (12 weeks = 12 times); therefore, we had to account for repeated observations in our linear regression analysis. For both effort and interruptions we tested two types of hypotheses: (1) developers working on more projects tend to have more interruptions and tend to spend more effort on multitasking overhead; (2) the increase of the number of projects worked on is associated with increase of the number of interruptions and effort.

To answer the second group of questions we used a multiple linear regression analysis to compute correlation coefficients. We treated developers as a categorical factor (a predictor). Fig. 4 shows the number of interruptions versus the number of projects per week. The total number of data points is 348 (repeated observations

3.3 Calculation of Work Interruptions and Reimmersion Time

We used reimmersion time as described in Fig. 1 to evaluate the impact of cross-project interruptions on developers’ productivity. In order to calculate the total effort each developer spends on reimmersion, we need to know the number of interruptions (obtained from work logs) and the reimmersion time after each interruption (evaluated by developers).
The linear correlation is strong (multiple $R^2 = 0.7779$). It supports the hypothesis $H_2.a$. This means that when a developer works on three projects in one week, he/she experiences more interruptions compared to weeks when he/she works on two projects. If developers work on one project, there is no cross-project interruptions at all. Fig. 5 shows the effort spent on context switching versus the number of projects per week. The total number of data points is 348 (repeated observations included). The linear correlation is strong (multiple $R^2 = 0.6934$). It supports the hypothesis $H_2.b$ It means that when a developer works on three projects in one week, he/she has more multitasking overhead compared to weeks when he/she works on two projects.

To better see how multitasking overhead affects productivity, we converted effort spent on cross-project interruptions (measured in person hours) into the percentage of the full-time equivalent (FTE). $FTE = 40$ person hours per week.

Fig. 6 shows average effort each developer spent on cross-project multitasking overhead per week. It also compares multitasking overhead results from observations with the G. Weinberg’s heuristic, which predicts a larger multitasking overhead.

Different sources estimate that the reimmersion time for information/knowledge workers varies from 20 minutes to 1-2 hours [1, 2, 7, 9]. We evaluated a lower bound estimate for the multitasking overhead using a constant 0.5 hour value of the reimmersion time for all interruptions (see Fig. 6). The evaluated lower bound underestimates the actual average value of the multitasking overhead.

Additionally, we wanted to see if there was any impact of cross-project multitasking on quality of work. Students were evaluated during architecture reviews meetings and by the clients at the end
we need to collect more data. However, visual observation of the
This is an on-going research which has limitations discussed below.
the same classes (e.g. students who took 3 classes were all enrolled
in other classes, students in the selected sample were enrolled in
neering educational environment. To limit the impact of deadlines
overhead.
here can be viewed as a lower bound of the overall multitasking
and self-in/licted interruptions. Therefore, evaluations presented
controlled experiment.
For the reimmersion time, we plan to conduct a more
projects and the number of interruptions. To get a more accurate
effect our conclusions about the correlation between the number of
in terms of grade points deducted. Fig. 7 shows grade deduction
of the semester. The impact on the quality of work was measured
in terms of grade points deducted. Fig. 7 shows grade deduction
depending on the amount of multitasking overhead.
In order to make quantitative conclusions about correlation be-
tween the quality of work and the amount of multitasking overhead,
we need to collect more data. However, visual observation of the
data points suggests that projects where multitasking overhead
took less than 40% of the overall effort had a better quality of work
compare to other projects. Table 1 summarizes all other results.

5 TREATS TO VALIDITY
This is an on-going research which has limitations discussed below.
Self-tracking of the reimmersion time and hours devoted to
projects may not be accurate. However, these values do not af-
fact our conclusions about the correlation between the number of
projects and the number of interruptions. To get a more accurate
evaluation of the reimmersion time, we plan to conduct a more
controlled experiment.
Counting only cross-project interruptions might underestimate
the overall multitasking overhead caused by internal (within project)
and self-inflicted interruptions. Therefore, evaluations presented
here can be viewed as a lower bound of the overall multitasking
overhead.
Participating subjects were graduate students in software engi-
eering educational environment. To limit the impact of deadlines
in other classes, students in the selected sample were enrolled in
the same classes (e.g. students who took 3 classes were all enrolled
in software engineering class, algorithms class and AI class). The
method used in this research can be applied to industry projects in
other domains.

6 CONCLUSIONS
In this study, we evaluated the impact of cross-project multita-
kling on software development effort by analyzing work logs and
weekly progress reports. Results show that developers who work
on 2 or 3 projects spend on average 17% of their effort on context
switching between tasks from different projects. Developers who
were involved in more projects tend to have more cross-project
work interruptions. However, the linear correlation between the
number of projects each developer worked on in one week and the
amount of effort developers spend on context switching is weak.
This partially can be explained by excessive amount of switching
between two most active projects.
The number of interruptions can be used successfully to evaluate
the quantitative impact of multitasking on effort. The number of
interruptions that people experience can also be used as a metric
and a measure of multitasking in teams. Additionally, the impact
of multitasking on effort can be integrated into software develop-
ment parametric cost and schedule estimation models such as the
Constructive Cost Estimation Model (COCOMO®) for better effort
estimation.
Finally, the work log analysis tools that we used in our research
can be integrated with project tracking systems such as Atlassian
Jira to provide real-time information about work interruptions and
their impact on productivity.

REFERENCES
[1] Parnin, Chris, and Spencer Rugaber. “Resumption strategies for interrupted pro-
ness: managing multiple working spheres.” In Proceedings of the SIGCHI confer-

Table 1: Hypotheses summary.

<table>
<thead>
<tr>
<th>Hypothesis</th>
<th>$R^2$ / multiple $R^2$</th>
<th>p</th>
<th>Accepted / Rejected</th>
</tr>
</thead>
<tbody>
<tr>
<td>H1.a</td>
<td>0.60699</td>
<td>&lt;0.01 Accepted</td>
<td></td>
</tr>
<tr>
<td>H2.a</td>
<td>0.7779</td>
<td>&lt;0.01 Accepted</td>
<td></td>
</tr>
<tr>
<td>H1.b</td>
<td>0.1622</td>
<td>&lt;0.03 Rejected</td>
<td></td>
</tr>
<tr>
<td>H2.b</td>
<td>0.6934</td>
<td>&lt;0.01 Accepted</td>
<td></td>
</tr>
</tbody>
</table>

Figure 7: Quality of work decrease depending on multitask-
ing overhead.
ABSTRACT

ProcessPAIR is a novel tool for helping software developers analyzing their personal performance. Based on a performance model calibrated from the anonymized performance data of many developers and the performance data submitted by an individual developer, it automatically identifies and ranks potential performance problems and their root causes for that developer. In this work we present WebProcessPAIR, which extends ProcessPAIR with the ability to recommend improvement actions to address the root causes identified, based on a crowdsourcing approach. A case study illustrates WebProcessPAIR usage.

KEYWORDS

• Software and its engineering → Software development methods;

1 MOTIVATION AND BACKGROUND

Collecting product and process measures in software development projects is important as a basis for self-assessing current performance, identifying performance problems and root causes and reacting to those causes. However, analyzing the collected data manually is challenging because of the expertise required, the lack of benchmarks for comparison, the amount of data to analyze, and the effort required. Although tools exist to automate data collection and help in problem identification and interactive performance analysis (e.g., [4] [5] [6]), practically no tool support exists for automated comparison with benchmarks and causal analysis.

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).

ICSSP ’17, July 5-7, 2017, Paris, France
© 2017 Copyright held by the owner/author(s).
https://doi.org/10.1145/3084100.3084365
To that end, we follow a crowdsourcing approach. Experts and users of the development process under consideration (called contributors in Fig. 1b) are invited to propose improvement actions for each PI defined in the corresponding PM (Fig. 2 - right). They are also invited to vote (like or dislike) on improvement actions previously proposed; those votes are used for ranking the improvement actions, from the most popular to the less popular.

Then, when a user uploads its performance data to be automatically analyzed (Fig. 2 - left), WebProcessPAIR will recommend the most popular improvement actions for each root cause identified. In case the user accepts a recommendation, its popularity goes up. In case the user is not satisfied with the recommendations given, he/she can also propose new improvement actions. Hence, the catalogue will automatically evolve based on user feedback.

3 CASE STUDY

To illustrate and assess the usefulness of WebProcessPAIR we conducted a case study using the Personal Software Process (PSP) [1]. First, we loaded to WebProcessPAIR the PM for the PSP that we developed in previous work [2] (step 1 in Fig. 1a), and calibrated it automatically based on a large PSP data set from the Software Engineering Institute referring to 31,140 projects concluded by 3,114 engineers during 295 classes of the classic PSP for Engineers I/II training courses (step 2a in Fig. 1a).

Then, two of the authors, with a good knowledge of the PSP, populated the catalogue of improvement actions (step 2b in Fig. 1b). As illustrated in Fig. 3, registered users can recommend, comment or vote on improvement actions for each PI.

Finally, the performance data of a PSP student from Tec de Monterrey, in Mexico, stored in Process Dashboard [6], was submitted to WebProcessPAIR for analysis. The results are illustrated in Fig. 4. In this case, there is a clear performance problem with 3 top-level PIs analyzed (Process Quality Index, Productivity and Time Estimation Accuracy). Fig. 4 shows the causes identified for the poor productivity of the user and the recommendations provided.

4 CONCLUSIONS

WebProcessPAIR is a novel platform that takes advantage of crowdsourcing and analytical techniques to help software engineering students and professionals analyzing their personal performance. In the near future, we intend to deploy WebProcessPAIR in an integrated SaaS project and process management platform. We also intend to take advantage of the user data submitted for analysis to recalibrate and evolve the performance model.

ACKNOWLEDGMENTS

This work is partially financed by the European Regional Development Fund (ERDF) through the Operational Programme for Competitiveness and Internationalisation (COMPETE 2020) and Portugal 2020, within project AIMS2 with the reference POCI-01-0247-FEDER-006405, and research grant with the reference SFRH/BD/85174/2012.

REFERENCES

Towards Capability-Oriented Business Process Management

Amin Jalali
Stockholm University, Sweden
aj@dsv.su.se

ABSTRACT
Capability management is an important area of research in information systems which focuses on managing ability and capacity in organizations. Business processes play a very important role in capability management, so it is vital to consider the notion of capability when managing business processes. Thus, this paper proposes a new approach to Business Process Management, which is based on considering the notion of capability as a first-class element in the BPM lifecycle. The new approach introduces future directions for the development of systems, techniques, methods and guidelines that empower organizations to manage their capabilities when managing their processes.

CCS CONCEPTS
• Information systems → Information systems applications;

KEYWORDS
Business Process Management, Capability Management, Lifecycle

1 INTRODUCTION
Supporting capability management using information systems has been recently positioned as a new paradigm in the area of Enterprise Modeling (EM). The capability is defined as “the ability and capacity that enable an enterprise to achieve a business goal in a certain context” [1].

For example, a car manufacturer might have a capability to produce thousand cars per month. This capability is the ability and capacity that is the result of how the process is managed in regards to resources and other organizational settings. It is important not only to capture whether organization functions as expected but also to be aware of the organization’s capability when applying changes in processes and their organizational settings. The lack of proper capability management can result in losing the competitive advantage in the market. To manage capabilities, there are needs for new methods, guidelines, frameworks, and applications to support modeling, monitoring and adjusting capabilities in organizations.

Currently, there is a lack of support in managing capabilities in the area of Business Process Management (BPM). Thus, this paper proposes a new approach that considers the notion of capability as a first-class element in the BPM lifecycle. The lifecycle is adopted based on the Capability Driven Development (CDD) methodology [6].

2 APPROACH
Managing business processes entails enacting a set of different activities according to some rules in an organization. A BPM lifecycle is a process that describes how different sets of actions support managing business processes effectively and efficiently. There are different BPM life-cycles proposed by different researchers and practitioners, which have been adapted for different processes, e.g. managing risk, cost, etc.

To enable managing business processes based on the notion of capability, we extend the lifecycle proposed by zur Muehlen [7] since it supports the integration of other enterprise models like goals, KPIs, etc. to process models. We name the extension “Capability-Oriented BPM lifecycle”. In this paper, we describe how this lifecycle can support different phases of capability management as defined by the Capability Driven Development (CDD) methodology [6]. Figure 1 and Figure 2 illustrate the lifecycle and the CDD methodology respectively.

We mapped each phase in our lifecycle to a phase in the CDD methodology, which is specified in these figures by the same numbers. We describe the requirements for each phase of the Capability-Oriented BPM lifecycle as follow.

Phase ➊: the first step in the CDD methodology is to match capabilities to enterprise models and patterns. In the process driven projects, we need to gather requirements of processes based on the notion of capabilities in the Goal Specification, Environment Analysis phase. Thus, a “Capability-Oriented Requirement Engineering” approach is needed for such a purpose.

Phase ➋: the second step in the CDD methodology is to composite several services that support delivery of different capabilities. In the process driven projects, the composition is equivalent to the definition of process models, so a Capability-oriented Business Process Modeling (CO-BPM) approach needs to be defined to support such composition. Note that capabilities are a kind of cross-cutting concerns [3, 5] that are not limited to one process models, so it is important to separate and encapsulate this concern and relate them to related process models.

Phase ➌: the third step in the CDD methodology is to integrate and develop systems which are aware of capabilities. There are different issues regarding how the systems should be integrated to capture the notion of capability [2]. The main obstacle in implementing capability oriented processes are the lack of tools that support such implementation. Here, we need Capability-Oriented Business Process Management Systems (CO-BPMS) to fulfill this goal.
After the third step, we can deliver capability applications, which is a part of a third cycle depicted below Figure 2 in the CDD methodology.

Phase 4: the fourth step in the CDD methodology is the enactment of systems supporting the notion of capability. This step is equivalent to Process Enactment step in the BPM lifecycle.

Phase 5: the fifth step in the CDD methodology is monitoring KPIs, which is equivalent to Process Monitoring in our lifecycle. Monitoring capabilities at runtime is an essential point in managing capability-oriented business processes - like other capability-driven approaches [6]. This feature can be supported by considering the capability as a new perspective when monitoring business processes [4].

Phase 6: the last step in the CDD methodology is to capture context and perform prediction, which is equivalent to Process Evaluation in our lifecycle. Here, new techniques are required in process mining, and data-based analysis approaches to incorporate the notion of capability when analyzing business process data. The result of the last three phases enables us to update the capability models and patterns.

In this life-cycle, the requirements which are defined using Capability-Oriented Requirements Engineering (CORE) techniques can be used as an input for Capability-oriented Business Process Modeling (CO-BPM). The process models which incorporate the notion of capability can be used for implementation and configuration of systems to enact process models. The capabilities need to be monitored at runtime, which can provide inputs for evaluating the processes. The result can adjust the processes for further improvement. The improvement can be performed by repeating the whole life-cycle.

3 CONCLUSION

In this paper, we proposed a new approach to supporting capability-oriented business process management. This approach suggests a new business process management life-cycle that incorporates the notion of capability in different steps including requirement specification, modeling, implementation, enactment, monitoring, and evaluation of business processes. It suggests capabilities be treated as a first-class element while managing business processes.

This proposal opens many directions for future works. It would be interesting to investigate how the notion of capability can be encapsulated as a new perspective when modeling business processes. Since these models need to be enacted, it is needed to define both syntax and semantics for such models. In addition, operational semantics of these models needs to be defined to enable implementing Business Process Management Systems that support enactment of these models. It also needed to specify how capabilities should be monitored and how the result of enacting these models can be used for further evaluating business processes. The adjustment of business processes to support new capabilities would be an interesting future direction for research.

REFERENCES

First International Workshop on Verification of Business and Software Processes

Souheib Baarir
University of Paris Nanterre
200 Avenue de la République
Nanterre 9200, France
souheib.baarir@lip6.net

Kais Klai
University of Paris 13
Institut Galilée Avenue J.B. Clément
Villetaneuse 93430, France
kais.klai@lipn.univ-paris13.fr

ABSTRACT
Processes, whatever the field (e.g. software, military or healthcare), are everywhere. They represent the building block of any information system nowadays. Business processes are used to represent the enterprise’s business and services it delivers. They are also used as a mean to enforce customer’s satisfaction and to create an added value to the company. Software processes are critical as well since they represent the guaranty to respect development process’s deadlines and to ensure a certain quality of the delivered software, which in some cases will end up being the company’s information system itself. It is then more than critical to seriously consider the design of such processes and to make sure that they are free of any kind of inconsistencies. One possible way to unsure that the developed processes are safe is to apply formal verification. Hence, we propose this workshop to investigate the novelties and advances concerning the application of formal methods in Business/Software (BS) processes design and execution.

CCS CONCEPTS
• Software and its engineering → Process management; Process management; Model-driven software engineering.

KEYWORDS

ACM Reference format:
https://doi.org/10.1145/3084100.3087676

1 ORGANIZERS
• Souheib Baarir (Main Contact): He is currently an assistant professor at the University of Paris Nanterre, France. His researches mainly concern Formal verification of discrete complex systems.
• Kais Klai: He is currently an assistant professor at the University Paris 13, France. His researches mainly concern the specification and the formal verification of concurrent systems.

2 THE WORKSHOP
2.1 Topics
We will encourage submissions that push the state of the art and practice in the following topics (but not limited to):
• Formal specification of Business/Software (BS) processes modeling languages.
• Formal approaches in the development of BS processes.
• Ability of formal methods to handle real-world problems.
• Formal Tools and technologies for BS process verification.
• Impact of formal methods on BS process based applications.
• Integration of Formal methods in the BS process life cycle.
• Case studies.
• Experimental validation.

2.2 Goals
• To provide a meeting point for researchers and practitioners in the area.
• To identify and gather a corpus of case studies and benchmarks to benefit the research and practitioner community.

2.3 Intended Audience
The workshop is intended for students and academics interested in getting involved or that are currently carrying out research in the application of formal methods for BS processes verification and validation.

3 WORKSHOP FORMAT
The workshop is intended to run for a half-day. The accepted paper will be presented by its author(s) and discussed by two other workshop participants (e.g. authors of other papers, PC members, organizers). Also, the workshop will be supplied by some keynotes and invited presentations related to its general field. This will initiate a possible collaboration between the participants and prepare the future editions of the workshop.

4 PC COMMITTEE MEMBERS
• Prof. Pascal Poizat, University Paris Nanterre, France
• Prof. Gwen Salaun, University of Grenoble Alpes, France
• Prof. Farouk Toumani, University Blaise Pascal, France
• Dr. Marco Kuhrmann, The M. M. Moller Institute, Denmark
• Dr. Maximilien Colange, EPITA, France
• Prof. Walid Gaaloul, TELECOM SudParis, France
• Dr. Ekkart Kindler, Technical University of Denmark, Denmark
• Dr. Mohamed Graiet, ISIM, Tunisia

• Prof. Dominique Rieu, University of Grenoble Alpes, France
• Dr. Tewfik Ziadi, University P. & M. Curie, France
• Dr. Lom M. Hillah, University Paris Nanterre, France
First International Workshop on Hybrid dEveLopmENt Approaches in Software Systems Development

Marco Kuhrmann
Clausthal University of Technology
Institute for Applied Software Systems Engineering
Goslar, Germany
kuhrmann@acm.org

Philipp Diebold
Fraunhofer Institute for Experimental Software Engineering
Process Engineering
Kaiserslautern, Germany
philipp.diebold@iese.fhg.de

Jürgen Münch
Reutlingen University
Business Informatics & Herman Hollerith Center
Böblingen, Germany
j.muench@computer.org

ABSTRACT

A software process is the game plan to organize project teams and run projects. Yet, it still is a challenge to select the appropriate development approach for the respective context. A multitude of development approaches compete for the users’ favor, but there is no silver bullet serving all possible setups. Moreover, recent research as well as experience from practice shows companies utilizing different development approaches to assemble the best-fitting approach for the respective company: a more traditional process provides the basic framework to serve the organization, while project teams embody this framework with more agile (and/or lean) practices to keep their flexibility. The first HELENA workshop aims to bring together the community to discuss recent findings and to steer future work.

CCS CONCEPTS
- General and reference → Surveys and overviews; - Software and its engineering → Software development methods; Software organization and properties; Designing software; Software development techniques; Programming teams;

KEYWORDS
Agile software development; software process; hybrid development approaches; survey

ACM Reference format:

1 INTRODUCTION

A software process is the game plan to organize project teams and run projects. Yet, it still is a challenge to select the appropriate development approach for the respective context. Since there is no “Silver Bullet” [1] in software development, software engineers are on the quest for suitable development approaches, yet facing a huge variety of contextual factors influencing the definition of appropriate development processes [2, 7]. Consequently, a variety of development approaches compete for the users’ favor, but there is no “Silver Bullet” serving all possible setups. Moreover, recent research as well as experience from practice shows companies utilizing different development approaches to assemble the best-fitting approach for the respective company: a more traditional process provides the basic framework to serve the organization, while project teams embody this framework with more agile (and/or lean) practices to keep their flexibility.

1.1 The HELENA Project

Accepting West’s claim that the “Water-Scrum-Fall” has become reality [6], in 2015 we conducted a systematic review to investigate the current state-of-practice in software process use [5]. Among others, we found a considerable imbalance in the research concerning traditional and agile software & system development. Eventually, we teamed up founding the HELENA initiative, which aims to study the use of “Hybrid dEveLopmENt Approaches in software systems development”. This initiative grew to a real project involving more than 60 researchers from (currently) 21 countries (cf. Table 2). Initial results—in particular from the HELENA trails and the first stage of the study—have been presented at the annual meeting of the Software Process special interest group of the German Computer Society [4], and will be presented at the International Conference on Software System Process (ICSSP) 2017 [3].

1.2 The 1st HELENA Workshop

This 1st HELENA workshop primarily focuses on the community work initiated at ICSSP 2016 (Austin, Texas); in particular, the HELENA survey. Currently, the HELENA community comprises more than 60 (academic) contributors from (currently) 21 countries. In this workshop, we aim at bringing together all academic and industry contributors and further interested people to:

1. Report the current state and (tentative) outcomes of the HELENA survey (from a global and regional perspective)
2. Develop a work program and define next steps within the whole community
3. Build working groups, which work on selected (sub-)topics of interest
4. Create a research agenda for hybrid development
2 WORKSHOP ORGANIZATION

The 1st HELENA workshop is a 1-day workshop aiming at bringing together all the contributors of the HELENA project. Table 1 shows the general workshop schedule. Besides the reports on the current state of the work in the different regions all across the globe, a key activity in the workshop is working in Breakout Sessions. These sessions aim at identifying topics of interest that allow for (i) continuing the survey research, and (ii) to form working groups within the HELENA team. Eventually, this workshop will also develop a research agenda to steer further work on the use of hybrid development approaches.

Table 1: Overview of the workshops topics and schedule.

<table>
<thead>
<tr>
<th>Topic</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Introduction (Organizers) Report of the current state from a global perspective (Organizers) Setup of working groups Working groups breakout session Working groups formation and consolidation (group-specific work plan) Development of the HELENA Agenda and next steps Closing (Organizers)</td>
<td></td>
</tr>
</tbody>
</table>

3 HELENA CONTRIBUTORS

HELENA involves a huge number of academic and industry contributors from (currently) 21 countries as shown in Table 2.

Table 2: Overview of the countries and the number of researchers (#Col) currently participating in HELENA (in alphabetical order, status May 5, 2017).

<table>
<thead>
<tr>
<th>No.</th>
<th>Country</th>
<th>#Col</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Argentina</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>Austria</td>
<td>3</td>
</tr>
<tr>
<td>3</td>
<td>Brazil</td>
<td>5</td>
</tr>
<tr>
<td>4</td>
<td>Canada</td>
<td>2</td>
</tr>
<tr>
<td>5</td>
<td>Chile</td>
<td>1</td>
</tr>
<tr>
<td>6</td>
<td>Costa Rica</td>
<td>1</td>
</tr>
<tr>
<td>7</td>
<td>Denmark</td>
<td>4</td>
</tr>
<tr>
<td>8</td>
<td>Estonia</td>
<td>2</td>
</tr>
<tr>
<td>9</td>
<td>Finland</td>
<td>1</td>
</tr>
<tr>
<td>10</td>
<td>Germany</td>
<td>11</td>
</tr>
<tr>
<td>11</td>
<td>Ireland</td>
<td>7</td>
</tr>
<tr>
<td>12</td>
<td>Italy</td>
<td>4</td>
</tr>
<tr>
<td>13</td>
<td>Luxembourg</td>
<td>3</td>
</tr>
<tr>
<td>14</td>
<td>New Zealand</td>
<td>3</td>
</tr>
<tr>
<td>15</td>
<td>Norway</td>
<td>2</td>
</tr>
<tr>
<td>16</td>
<td>Spain</td>
<td>3</td>
</tr>
<tr>
<td>17</td>
<td>Sweden</td>
<td>7</td>
</tr>
<tr>
<td>18</td>
<td>Switzerland</td>
<td>2</td>
</tr>
<tr>
<td>19</td>
<td>The Netherlands</td>
<td>1</td>
</tr>
<tr>
<td>20</td>
<td>Turkey</td>
<td>2</td>
</tr>
<tr>
<td>21</td>
<td>UK</td>
<td>5</td>
</tr>
</tbody>
</table>

Each of these 21 sites has a local head supporting the general organization team, and we owe special thanks to all our colleagues, who helped us quality assuring the survey instrument, translating the instrument, and spreading the word among their local peers.

The full list of all HELENA contributors can be depicted from: https://helenastudy.wordpress.com/helena-team.

4 FURTHER WORK

This HELENA workshop is only the first one, and we plan to organize further editions. Therefore, we cordially invite all interested researchers and practitioners to join us in Paris and to discuss opportunities for further collaboration. HELENA is an open and still growing project, and we appreciate all kinds of support.

ACKNOWLEDGMENTS

We want to thank the ICSSP 2017 Chairs and organization board for providing us with the opportunity to held this first workshop in conjunction with ICSSP 2017. We look forward to a fruitful and long-term collaboration with the ICSSP community.

REFERENCES

Author Index

Aaramaa, Sanja ............... 40
Amalfitano, Domenico ........ 5
Amaro, Luis ................. 139

Baarir, Souheib ............... 143
Bastarrica, María Cecilia ..... 124
Boehm, Barry ............... 60, 134

Carroll, Noel ................. 3
Chen, Mingrui ............... 25, 109
Dasanayake, Sandun ........... 40
Dautais, Yann ................. 109
De Silva, Ian J. ............... 95
De Simone, Vincenzo ......... 5
Diebold, Philipp ............. 30, 80, 145
Dumas, Marlon ............... 85

Faria, João Pascoal .......... 139
fasolino, Anna Rita .......... 5
Felderer, Michael .......... 30

Garousi, Vahid ............... 30
Ge, Jidong ................... 109
Geng, Ruili .................. 25
Ghofrani, Javad .............. 70
Gong, Haojie ................ 50

Hanser, Eckhart .............. 30
Heimdahl, Mats P. E. ......... 95
Henriques, Pedro Castro ..... 139

Hohl, Philipp ................ 70
Houston, Dan ............... 114
Huang, LiGuo ............... 25, 109

Iren, Deniz .................. 100
Jalali, Amin ................. 141
Klai, Kais .................... 143
Kuhrmann, Marco ........... 30, 145
Kuziemsky, Craig .......... 3

Lane, Jo Ann .................. 134
La Rosa, Marcello ........... 85
Li, Zeheng ................... 25
Linssen, Oliver .............. 30
Liu, Bohan ................... 50

Marín, Jacqueline .......... 124
Markkula, Jouni ............. 40
McCaffery, Fergal ........... 30
Münch, Jürgen ............. 30, 70, 145

Ng, Vincent ................ 25
Nguyen, Hoang ............. 85
Oivo, Markku ............... 40

Pervovich, Daniel .......... 124
Prause, Christian R. .......... 30
Qi, Kan ..................... 60
Rayadurgam, Sanjai ......... 95
Raza, Mushtaq ............... 139
Reijers, Hajo A. ............. 100
Richardson, Itha ........... 3
Richter, Dominik .......... 80
Rioseco, Luis .............. 124
Rodchenko, Natalia ....... 134
Rosenberg, Doug .......... 60
Saukkonen, Samuli .......... 40
Scala, Stefano ............... 5
Schneider, Kurt ............. 70
Stupperich, Michael ....... 70

Tell, Paolo .................. 30
Tregubov, Alexey .......... 134
Treter, Kitija ............... 30

Verenich, Ilya ............... 85
Wang, Bo ................... 60
Wang, Jiaojiao ............. 15
Yu, Dexian ................ 50
Yu, Dongjin ................ 15
Zehler, Thomas ............ 80
Zhang, He .................. 50
Zimmermann, Thomas ....... 1