GI-Edition

Lecture Notes in Informatics

Matthias Tichy, Eric Bodden, Marco Kuhrmann, Stefan Wagner, Jan-Philipp Steghöfer (Hrsg.)

Software Engineering und Software Management 2018

5.–9. März 2018
Ulm

M. Tichy, E. Bodden, M. Kuhrmann, S. Wagner, J.-P. Steghöfer (Hrsg.):
Software Engineering und Software Management 2018

279

Proceedings

Matthias Tichy, Eric Bodden, Marco Kuhrmann,
Stefan Wagner, Jan-Philipp Steghöfer (Hrsg.)

Software Engineering und Software Management 2018

5.-9. März 2018
Ulm, Deutschland

Gesellschaft für Informatik e.V. (GI)

Lecture Notes in Informatics (LNI) - Proceedings
Series of the Gesellschaft für Informatik (GI)

Volume P-279

ISBN 978-3-88579-673-2
ISSN 1617-5468

Volume Editors
Prof. Dr. Matthias Tichy
Universität Ulm
James-Franck-Ring, 89069 Ulm, Deutschland
matthias.tichy@uni-ulm.de

Prof. Dr. Eric Bodden
Heinz Nixdorf Institut, Universität Paderborn & Fraunhofer Institut für Entwurfstechnik Mechatronik (IEM)
Fürstenallee 11, 33102 Paderborn, Deutschland
eric.bodden@upb.de

PD Dr. Marco Kuhrmann
Technische Universität Clausthal
Julius-Albert-Str. 4, 38678 Clausthal-Zellerfeld, Deutschland
marco.kuhrmann@tu-clausthal.de

Prof. Dr. rer. nat. Stefan Wagner
Universität Stuttgart
Universitätsstraße 38, 70569 Stuttgart, Deutschland
stefan.wagner@informatik.uni-stuttgart.de

Dr. Jan-Philipp Steghöfer
Chalmers University of Technology
SE-412 96 Gothenburg, Schweden
jan-philipp.steghofer@cse.gu.se

Series Editorial Board

© Gesellschaft für Informatik, Bonn 2018
printed by Köllen Druck+Verlag GmbH, Bonn

Vorwort

Wir begrüßen Sie herzlich zu den Tagungen Software Engineering (SE) des Fachbereichs Softwaretechnik der Gesellschaft für Informatik (GI) sowie Software Management (SWM) des GI-Fachausschusses WI-MAW hier an der Universität Ulm.

Die jährlich stattfindende Tagung des Fachbereichs Softwaretechnik der Gesellschaft für Informatik dient als Plattform für den Austausch von Erfahrungen und Erkenntnissen aus dem Bereich der Softwaretechnik. Die Tagung richtet sich sowohl an Softwareentwickler und Softwareentwicklerinnen aus der Praxis, als auch an Forscherinnen und Forscher aus dem akademischen Umfeld.

Die seit 1995 stattfindende Tagung des GI-Fachausschusses WI-MAW fokussiert auf die vielfältigen Herausforderungen des modernen Produktmanagements und der Produktinnovation an der Schnittstelle von Softwaretechnik und der wirtschaftlichen Verwertung software-intensiver Produkte und Dienstleistungen.

Das wissenschaftliche Hauptprogramm der Tagungen setzt das sehr erfolgreiche Format der letzten Jahre fort und enthält Beiträge zu exzellenten Forschungsergebnissen der letzten beiden Jahre, die auf den internationalen Topkonferenzen und in den Topzeitschriften der Softwaretechnik veröffentlicht wurden. Ziel ist die Stimulation des wissenschaftlichen Diskurses innerhalb der deutschsprachigen Softwaretechnik sowie die Erhöhung der Sichtbarkeit bereits veröffentlichter Ergebnisse. Das wissenschaftliche Hauptprogramm wird ergänzt durch Beiträge zu Fallstudien in der industriellen Praxis sowie neuen Ideen. Es wird abgerundet durch einen Special Track zu „Erklärbarer Software“ – einem aktuell spannenden Thema in der Softwareentwicklung. Alle Einreichungen wurden von mindestens drei Gutachtern begutachtet und anschließend durch die jeweiligen Programmkomitees ausgewählt. Schließlich ergänzen Workshops zu aktuellen Themen in Forschung, Praxis und Lehre das Tagungsprogramm.

Wir danken allen Beteiligten, insbesondere den Autoren und Vortragenden für die Bereicherung des Programms, den Programmkomitees für die gewissenhaften Reviews, den Keynote-Speakern für die spannenden Beiträge, den Sponsoren, der Universität Ulm und der Stadt Ulm, sowie nicht zuletzt allen Unterstützern bei der lokalen Organisation, die diese Veranstaltung möglich gemacht haben.

Wir freuen uns auf eine interessante Konferenz und hoffen auf viele spannende Diskussionen und tolle neue Ideen!

Ulm, im Februar 2018

Matthias Tichy, Eric Bodden, Marco Kuhrmann, Stefan Wagner und Jan-Philipp Steghöfer

Sponsoren

Wir danken den folgenden Unternehmen und Institutionen für die Unterstützung der Konferenz.

Goldsponsoren

Daimler AG

adesso AG

Silbersponsoren

e.solutions GmbH

Lokale Partner

Universität Ulm

Stadt Ulm

Tagungsorganisation

Tagungsleitung:	Matthias Tichy, Universität Ulm
Programmkomiteeleitung:	Eric Bodden, Universität Paderborn Stefan Wagner, Universität Stuttgart Marco Kuhrmann, Technische Universität Clausthal
Workshops:	Jan-Philipp Steghöfer, Chalmers University of Technology
Social Media:	Stephan Krusche, Technische Universität München
Local Chair:	Alexander Raschke, Universität Ulm
Web Chair:	Alexander Breckel, Universität Ulm
Tagungsband:	Stefan Kögel, Universität Ulm

Programmkomitees

SE - Wissenschaftliches Hauptprogramm

Eric Bodden	Universität Paderborn
Andreas Zeller	Universität des Saarlandes
Sven Apel	Universität Passau
Lutz Prechelt	Freie Universität Berlin
Steffen Becker	Universität Stuttgart
Christian Kästner	Carnegie Mellon University
Gregor Engels	Universität Paderborn
Lars Grunske	Humboldt-Universität zu Berlin
Christian Hammer	Universität Potsdam
Michael Pradel	Technische Universität Darmstadt
Michael Goedicke	Universität Duisburg-Essen
Bernhard Rumpe	RWTH Aachen

SWM - Wissenschaftliches Hauptprogramm

Marco Kuhrmann	Technische Universität Clausthal
Jens Borchers	Sopra Steria GmbH
Jürgen Münch	Hochschule Reutlingen
Thorsten Spitta	Gesellschaft für Informatik
Oliver Linssen	Training und Beratung
Masud Fazal-Baqaie	S&N CQM GmbH
Birgit Demuth	Technische Universität Dresden
Ruediger Weissbach	Hochschule für Angewandte Wissenschaften (HAW) Hamburg
Georg Herzwurm	Universität Stuttgart
Christopher Jud	Universität Stuttgart
Stefan Eicker	Universität Duisburg-Essen

SE – Track: „Erklärbare Software"

Stefan Wagner	Universität Stuttgart
Andreas Vogelsang	Technische Universität Berlin
Matthias Riebisch	Universität Hamburg
Michael Felderer	Universität Innsbruck
Florian Matthes	Technische Universität München
Dirk Beyer	Ludwig-Maximilians-Universität München
Joel Greenyer	Leibniz Universität Hannover
Dirk Nowotka	Christian-Albrechts-Universität zu Kiel
Kurt Schneider	Leibniz Universität Hannover
Barbara Paech	Universität Heidelberg

Inhaltsverzeichnis

Keynotes

Workshops

Hauptprogramm

SE Session 1 - Software Product Lines

SE Session 2 - Security

SE Session 3 - Processes and Evolution of Software Engineering

SE Session 4 - Testing

SE Session 5 - Empirical 1

SE Session 6 - Modeling

SE Session 7 - Empirical 2

SE Session 8 - Model evolution and transformation

SE Session 9 - Program analysis and failure prediction

SE Session 10 - Software Process

SE Session 11 - Requirements and Traceability

SE Session 12 - Design

SE Session 13 - Program editing and comprehension

SE Session 14 - Software Architecture

SWM Session 1 - Wissenschaftliches Programm

SWM Session 2 - Fallstudien in der Industrie

SWM Session 3 - Neue Ideen

Erklärbare Software 1 - Understandable Verification

Erklärbare Software 2 - Understandable Decisions

Erklärbare Software 3 - Understandable Software

Keynotes

Crowdsourcing Software Development: Silver Bullet or Lead Balloon

Brian Fitzgerald[1]

1 Abstract

Crowdsourcing is emerging as an alternative outsourcing strategy which is gaining increasing attention in the software engineering community. However, crowdsourcing software development involves complex tasks which differ significantly from the micro-tasks that can be found on crowdsourcing platforms such as Amazon Mechanical Turk—the latter are much shorter in duration, and typically very simple and do not involve any task interdependencies. To achieve the potential benefits of crowdsourcing in the software development context, companies need to understand how this strategy works, what challenges arise, and what factors might affect crowd participation. Research to date on crowdsourcing software development has tended to focus on the 'crowd' or the technical platform, with little research from the perspective of the customer who is seeking to leverage the crowdsourcing development model. The findings from an in-depth case study of crowd-sourcing software development in a Fortune 500 company are augmented with an analysis of over 13,000 crowdsourcing competitions over a ten-year period on the Topcoder crowdsourcing platform, one of the most popular platforms for software development, are drawn on to evaluate the effectiveness of crowdsourcing in a software development context.

2 Bio

Professor Brian Fitzgerald is Director of Lero – the Irish Software Research Centre, where he previously held the role of Chief Scientist. Prior to that he served as Vice-President Research at the University of Limerick. He also holds an endowed professorship, the Krehbiel Chair in Innovation in Business & Technology, at the University of Limerick. His research interests lie primarily in software development, encompassing open source and inner source, crowdsourcing software development, agile and lean software development, and global software development. His publications include 15 books, and over 150 peer-reviewed articles in the leading international journals and conferences in both the Information Systems and Software Engineering fields, including MIS Quarterly (MISQ), Information Systems Research (ISR), IEEE Transactions on Software Engineering (TSE) and ACM Transactions on Software Engineering Methodology (TOSEM). Prior to taking up an academic position, he worked in the software industry for about 12 years, in a variety of sectors (including

[1] Lero, University of Limerick, brian.fitzgerald@lero.ie

finance, telecommunications, manufacturing, bespoke software development) in a number of countries (Ireland, Belgium, Germany).

Understanding Organizational Evolution of Software Projects

Sven Apel[1]

1 Abstract

The role of the organizational structure of large-scale, distributed software projects and its relation to project success has been gaining considerable attention in the research and practice of software engineering. Research has shown that analyzing the organizational structure reveals a great extent of information relevant for project evolution and success, including quality, productivity, and delays. However, despite encouraging results, the knowledge on which organizational patterns are desirable and how we can elicit and improve them is often anecdotal, and implications thereof are transferred only rarely systematically.

In this talk, I will report on our ongoing endeavor of studying real-world software projects to provide deep insights into the nature and role of organizational structure for understanding and ensuring project success as well as to drive the development and evaluation of efficient software-engineering practices and tools. In the long run, we aim at answering a number of scientifically and practically relevant research questions, including how we can extract accurate information on a project's organizational structure, which organizational patterns arise in practice, how they vary over time, and how they relate to project success. Methodologically, we base our research on a rigorous network approach, which includes a representation of organizational structures, called a developer network, as well as a state-of-the-art network-analysis framework.

2 Bio

Prof. Dr. Sven Apel holds the Chair of Software Engineering at the University of Passau, Germany. The chair is funded by the esteemed Emmy-Noether and Heisenberg Programs of the German Research Foundation (DFG). Prof. Apel received his Ph.D. in Computer Science in 2007 from the University of Magdeburg, Germany. His research interests include software product lines, software analysis, optimization, and evolution, as well as empirical methods and the human factor in software engineering. He is the author or co-author of over a hundred peer-reviewed scientific publications. He serves regularly in program committees of top-ranked international conferences and he is a member of the editorial boards of IEEE Transactions on Software Engineering, IEEE Software, and Empirical Software Engineering. He was/will be program-committee co-chair of the 31st

[1] Chair of Software Engineering University of Passau, apel@uni-passau.de

International Conference on Automated Software Engineering (ASE) and the 12th Joint Meeting of the European Software Engineering Conference and the ACM SIGSOFT Symposium on the Foundations of Software Engineering (ESEC/FSE). His work has received two Best Paper Awards, one ACM SIGSOFT Distinguished Paper Award, as well as awards by the Ernst-Denert Foundation, the Karin-Witte Foundation, and the State of Saxony-Anhalt. Sven Apel is a member of the Young Academy of Europe.

Revolution vs. Evolution: Model-Based Engineering and the Industry - The Potential of MBE Micro Injections

Gerald Stieglbauer[1]

1 Abstract

By now, model-based engineering (MBE) has a long tradition in academics and research. In contrast to this long tradition, however, adoption of MBE principles in the industry still remain limited. This led to corresponding debates within the modelling community about the root causes of this limited adoption. In this talk, the speaker highlights the importance of these debates and shares his experience gained during six years of technology knowledge transfer activity from research to industrial applications. As its central hypothesis, the talk will be about so-called MBE Micro Injections, for which the speaker has observed the potential to make the adoption of MBE principles in industry more successful. Since the idea of MBE Micro Injections comprises not only technical issues but as well socio-cultural aspects, the speaker will talk about concrete observations in these directions during past and ongoing research projects.

2 Bio

Dr. Gerald Stieglbauer, (male) is AVL's expert on model-based approaches such as modeling languages and modeling language design (with special focus on DSLs), model transformation, and system engineering. He studied Applied Computer Science at the University of Salzburg, graduated 2003 and received his PhD in 2007. He was awarded with the BMW Scientific Award (diploma thesis) and the Award of Excellence of the Austrian Government (PhD thesis). The focus of both theses was in the area of model-based development and simulation of safety-critical software for embedded systems. In 2007, he joined the AVL to become responsible for the development of a model transformation engine for a testbed configuration system. In 2011, he joined the Research & Technology department for Instrumentation and Test Systems at AVL's headquarter in Graz. Since then, he is technology scout for model-based engineering (MBE) in support of knowledge transfer from science to AVL and vice versa. In this context, his focus of interest goes beyond the technological aspects of MBE and includes especially observations about socio-cultural aspects of successful strategies aiming at a sustainable introduction of innovations in industry.

[1] Technology Scout for Model-based Engineering AVL, gerald.stieglbauer@avl.com

Algorithm Accountability, Algorithm Literacy and the hidden assumptions from algorithms

Julien Siebert[1]

1 Abstract

Our societies are facing problems that are more and more complex so that decision making is often helped or even delegated to algorithms. Algorithmic decision making (ADM) processes are complex socio-technical systems which interact with society on a large scale. Credit scoring, automatic job candidate selection, predictive policing, or recidivism risk assessment are examples, among others, of already used ADM systems. In this talk, I will start with an overview of what is so far understood as Algorithm Accountability and Algorithm Literacy. I will then focus on algorithms that carry with them modeling assumptions (e.g., machine learning, data-mining algorithms...) and show what effects this has on the interpretation of the algorithms' results and how we could, from a software engineering point of view, bring more explainability.

2 Bio

Dr.-Ing. Julien Siebert holds a master degree in Engineerring and a master degree in Artificial Intelligence. He holds a PhD in computer science, from INRIA - Université de Lorraine (Nancy, FR), on the topic of meta-modeling for distributed simulations of complex systems[2]. He later spent 2 years of postdocs in the Theoretical Physics Institute from the TU Berlin[3] working on numerical simulations of non-linear dynamics of complex systems (wave propagation, patterns formation and coupled oscillators). Before joining the Algorithm Accountability Lab, in the TU Kaiserslautern, he spent several years at Zalando in Berlin as software engineer and data scientist.

[1] Technische Universität Kaiserslautern, siebert@cs.uni-kl.de
[2] http://www.mecsyco.com/
[3] SFB 910

Workshops

3rd Workshop on Continuous Software Engineering

Horst Lichter [1], Stephan Krusche [2], Dirk Riehle[3], Andreas Steffens[4]

In order to develop and deliver high-quality products to their customers, software companies have to adopt state-of-the-art software development processes. To face this challenge, companies are applying innovative methods, approaches and techniques like agile methods, DevOps, continuous delivery, test automation, infrastructure as code or container-based virtualization.

These new approaches have a significant impact on the specification, design, development, maintenance, operation and the evolution of software systems. Therefore, common software engineering activities, organizational forms and processes have to be questioned, adapted and extended to ensure continuous and unobstructed software development: continuous software engineering (CSE). So far, there is a lack of systematic approaches to face these challenges.

The goal of this workshop is to present and discuss innovative solutions, ideas and experiences in the area of CSE. The workshop aims to cover the following topics:

Processes and workflows

- Change management and handling user feedback
- Software development lifecycle for CSE
- Continuous delivery for requirements engineering/early prototyping
- Lean agile processes and practices

Technologies and tools

- Infrastructure as code
- Provisioning of software and infrastructure

[1] RWTH Aachen University, Research Group Software Construction
horst.lichter@swc.rwth-aachen.de
[2] Technische Universität München, Chair for Applied Software Engineering
krusche@in.tum.de
[3] Friedrich-Alexander-University Erlangen-Nürnberg, Open Source Research Group
dirk.riehle@fau.de
[4] RWTH Aachen University, Research Group Software Construction, steffens@swc.rwth-aachen.de

- Engineering of deployment pipelines

Architecture

- Design for scalability
- Software architecture for CSE
- Model driven architecture for CSE

Quality and testing

- Test automation and optimization
- Monitoring and performance
- Security for DevOps
- Metrics for DevOps

Microservices and DevOps

- Pattern and best practices
- Domain specific languages for microservices
- Distributed persistence
- Containerization

Culture and business

- Teaching CSE approaches
- Organizational issues for CSE
- Digital transformation and innovation

We intended to have contributions from industry and academia presented and discussed in the workshop. Therefore, we asked for original and evaluated research as well as for papers describing novel ideas, identified challenges, and especially experience reports related to the workshop's theme.

The presented papers cover different topics of CSE like dedicated process models and their application in CSE, new architectural styles like microservices and their integration with existing methodologies, and approaches to improve DevOps in organizations.

Program Committee

Lukas Alperowitz
TU München

Jan Bosch
Chalmers, Sweden

Michael Goedicke
Universität Duisburg-Essen

Willi Hasselbring
Universität Kiel

Martin Jung
develop group, Erlangen

Stephan Krusche,
TU München (Organizer)

Horst Lichter
RWTH Aachen
(Organizer)

Christian Nester
Google Inc.

Dirk Riehle
FAU Nürnberg
(Organizer)

Daniel Fogl
FAU Nürnberg

Heinz-Josef Schlebusch
Kisters AG, Aachen

Andreas Steffens
RWTH Aachen
(Organizer)

Christian Uhl
Matmatch GmbH,
München

André van Hoorn
Universität Stuttgart

Heinz Züllighoven
WPS/Uni Hamburg

Stefan Wagner
Universität Stuttgart

Thomas Kurpick
Trusted Shops, Köln

Matthias Tichy
Universität Ulm

5th Collaborative Workshop on Evolution and Maintenance of Long-Living Systems

Robert Heinrich[1] Reiner Jung[2] Marco Konersmann[3] Eric Schmieders[3]

Langlebige softwareintensive Systeme sind häufigen Anforderungsänderungen ausgesetzt. Das führt unter anderem zu inkonsistenten Anforderungsspezifikationen, Architekturerosion und SLA-Verletzungen. Die Relevanz dieser Problematik ergibt sich vor allem in der industriellen Praxis, in der solche Systeme ständig weiterentwickelt werden müssen. Besonders im Kontext von DevOps und Informationssystemen erfolgen viele Änderungen in kurzer Zeit. Aber auch für eingebettete Systeme wird es immer wichtiger mit zahl-reichen Änderungen umgehen zu können.

Traditionelle Methoden zur Entwicklung und zum Betrieb von Softwaresystemen sind nur begrenzt auf diese neuen Herausforderungen vorbereitet und bedürfen der weiteren Integration. In der Industrie und Forschung werden in diesem Umfeld viele neue Technologien und Ansätze entwickelt, welche eine kontinuierliche Evolution und Adaption von Softwaresystemen ermöglichen. Durch die hohe Dynamik des Themengebiets ist es notwendig, die Diskussion und Kooperation zwischen allen Beteiligten zu intensivieren, auch gerade um eine Integration der Ideen zu erreichen.

Ziel der EMLS-Workshopreihe ist es den Austausch zu Themen der Software-Evolution und -Wartung zu fördern. Der Workshop bietet dazu ein Forum um Herausforderungen, Lösungsansätze und Erfahrungsberichte zu diskutieren. Wie schon in den vergangenen Jahren setzt der Workshop auch dieses Jahr auf themenbezogene Kleingruppen. Dies hat sich in der Vergangenheit als eine sehr produktive Umgebung erwiesen.

Zu Beginn jeder Session stellen die Autoren ihre Themen in kurzen Impulsvorträgen vor. Anschließend werden zu diesen Themen Kleingruppen gebildet. Für jede Gruppe stellen die Organisatoren einen Moderator um die Diskussion zu leiten. Die Gruppen werden mit passendem Diskussionsmaterial, wie Flipchart und Diskussionskarten, ausgestattet. Diese werden zur Steuerung der Diskussion und zur Vorbereitung der Ergebnispräsentation genutzt. Am Ende der Session werden die Ergebnisse der Diskussionen in den Klein-gruppen zusammengetragen und anschließend im Plenum vorgestellt. Dies ermöglicht es allen Teilnehmern die Ergebnisse der anderen Gruppen zu erfahren. Zur Ergebnisdokumentation

[1] Karlsruher Institut für Technologie, Am Fasanengarten 5, 76131 Karlsruhe, robert.heinrich@kit.edu
[2] Universität Kiel, Christian-Albrechts-Platz 4, 24118 Kiel, reiner.jung@email.uni-kiel.de
[3] Universität Duisburg-Essen, Gerlingstraße 16, 45127 Essen,
{marco.konersmann, eric.schmieders}@paluno.uni-due.de

werden die erarbeiteten Flipchart-Präsentationen fotografiert und auf der Seite des Workshops bereitgehalten (*emls.paluno.uni-due.de*).

1st Workshop on Innovative Software Engineering Education

Stephan Krusche[1] Marco Kuhrmann[2] Kurt Schneider[3]

Abstract: Due to the growing numbers of students, courses can no longer be offered in high quality without systematic approaches. Hence, this workshop aims at presenting and discussing innovative teaching approaches in software engineering education, which are highly relevant for teaching at universities, colleges, and in online courses.

Keywords: Project Courses, Active Learning, Large Courses, Digital Teaching, Online Courses

1 Introduction

The number of students grows continuously and, thus, software engineering teachers face more and more challenges. Motivating students to actively participate in a course is especially difficult for large classes. Even though practice-oriented and project-based training becomes increasingly important, such project courses in cooperation with industry often come along with high effort. To compensate this situation, digital teaching, online courses, and other new teaching concepts complement the curriculum. They offer a wide range of possibilities for modern and attractive teaching, yet introduce further methodical, technical and organizational challenges to be considered by the teachers.

2 Goals

The aim of the *1st Workshop on Innovative Software Engineering Education* is to bring software engineering instructors together to actively work and discuss the most important topics, challenges, and solution approaches. The goal is to create a platform for sharing experiences and identifying common topics of interest to foster collaboration. The workshop discusses which specific challenges have not yet been solved, so that an agenda for the improvement of software engineering education can be developed taking into account changing social, economic and political conditions.

The workshop provides an interactive forum with paper and poster presentations, and room for discussion. Authors give short talks about their contributions (Section 3), which are

[1] Technische Universität München, krusche@in.tum.de
[2] Clausthal University of Technology kuhrmann@acm.org
[3] Leibniz Universität Hannover kurt.schneider@inf.uni-hannover.de

followed by intensive discussions. The discussions are moderated by selected supporters, who prepare (critical) questions thus stimulating and guiding the discussion. It is the overall goal of each workshop session to use the presented papers as starting point to enter the plenary discussion and shape the topics for interactive group discussions.

3 Contributions

The workshop received 20 submissions covering a wide range of topics in the field of software engineering education of which 12 submissions (1 full paper, 7 short papers, 4 posters) have been accepted and selected for presentation. The accepted papers address different topics, such as tool-support for automating parts of the education thus reducing effort, e.g., assessment of code quality in programming assignments. Other topics are the management of change in project courses, the combination of hardware and software development, and the teaching of pattern-based development. Furthermore, using reflection techniques, monitoring of student frustrations levels, and teaching domain-specific requirements engineering to industry are discussed in the workshop. Finally, the use of competition between software projects in education is presented in the workshop.

Jürgen Börstler (Blekinge Institute of Technology) starts the workshop by giving his keynote on the current challenges in software engineering education. The keynote is followed by the authors presenting their papers and posters briefly to initiate the discussion. Furthermore, for each regular paper a poster is presented in a dedicated poster session, which allows for building small groups discussing topics of interest.

All participants contribute to the identification of challenges and problems, and they also contribute to the development of ideas and solution approaches. For this, different working sessions will be part of the workshop in which, among other things, the most interesting challenges are brainstormed, outlined, and prepared for a plenary discussion. A desired outcome of the workshop is a set of appropriate solutions for the challenges identified including new approaches, technologies, methods, and tools needed to improve software engineering education. Outcomes culminate in a collection of topics to motivate and guide the final discussion, which will focus on future joint areas of work, new ideas, and further activities.

4 Conclusion

It is gratifying how many papers have been submitted to the workshop, even from an international audience. This underlines the importance of new and innovative approaches in software engineering education and motivates for additional workshops in the future.

Software Engineering für E-Learning-Systeme (SEELS)

Michael Striewe[1], Sven Strickroth[2], Ulrike Lucke[3]

Abstract: Der Workshop „Software Engineering für E-Learning-Systeme“ (SEELS) widmet sich softwaretechnischen Fragestellungen rund um die Entwicklung von E-Learning-Systemen und die Realisierung von vernetzten E-Learning-Landschaften an Schulen und Hochschulen. Ziel des Workshops ist es, aktuelle Forschungsfragen z. B. zu Schnittstellen von E-Learning-Systemen, Sicherheit in heterogenen E-Learning-Landschaften und das Management fachspezifischer Anforderungen in universellen E-Learning-Systemen zu identifizieren und zu diskutieren.

Keywords: E-learning, Software Engineering, Systemdesign, Standards

1 Hintergrund und Ziel des Workshops

Die Entwicklung von E-Learning-Systemen und die Zusammenstellung vernetzter E-Learning-Landschaften mit einer Vielzahl heterogener Systeme muss eine große Vielfalt an verschiedenartigen Anforderungen berücksichtigen. In Teilen gleichen E-Learning-Systeme klassischen Informationssystemen einschließlich der damit verbundenen Fragen nach Skalierbarkeit und Erweiterbarkeit. Im Zuge von Ubiquitous Learning benötigen sie aber auch umfangreiche Expertise im Mobile Software Engineering, stellen im Kontext von rechtssicheren Prüfungen hohe Anforderungen an die Sicherheit und nehmen in innovativen Szenarien Anleihen aus dem Games Engineering und der Gestaltung von virtueller Realität. Schon bei der Entwicklung einzelner Systeme erfordern diese Anforderungen ein sorgfältiges Vorgehen, während der Trend zur Bildung von vernetzten E-Learning-Landschaften innerhalb einer Bildungsinstitution oder sogar über mehrere Institutionen hinweg die Komplexität der Entwicklung und des Betriebs noch einmal erhöht.

Während sich nationale und internationale E-Learning-Konferenzen schwerpunktmäßig (medien-)didaktischen, fachspezifischen und sozio-technischen Fragen (einschließlich Usability) widmen und daher der Frage nachgehen, *wie man gute E-Learning Systeme baut*, nimmt dieser Workshop explizit softwaretechnische Fragen in den Fokus und geht damit der Frage nach, *wie man E-Learning-Systeme gut baut*. Das Ziel des Workshops ist es, die Besonderheiten der Domäne sichtbar zu machen und gleichzeitig den Blick für softwaretechnische Fragenstellungen innerhalb der Community zu schärfen.

[1] paluno – The Ruhr Institute for Software Technology, michael.striewe@paluno.uni-due.de
[2] Universität Potsdam, Institut für Informatik und Computational Science, sven.strickroth@uni-potsdam.de
[3] Universität Potsdam, Institut für Informatik und Computational Science, ulrike.lucke@uni-potsdam.de

2 Arbeitsthemen und Einreichungen

Der Call for Papers des Workshops machte bewusst keine enge Einschränkung der möglichen Arbeitsthemen und reichte somit von Fragen nach universellen Schnittstellen zur Etablierung von E-Learning-Landschaften über (Daten-)Sicherheit in heterogenen E-Learning-Landschaften sowie das Management fachspezifischer Anforderungen in universitätsweiten E-Learning-Landschaften bis hin zur Nutzung von Microservice-Architekturen.

Es konnten vier Beiträge als Full Paper angenommen werden, die tatsächlich eine größere Spannbreite an Themen abdecken: Ein Beitrag befasst sich auf Basis eines Literaturreviews mit typischen Komponenten und Architekturpattern für die Integration von E-Assessment-Features in E-Learning Systeme; zwei Paper präsentieren und diskutieren das Design jeweils eines konkreten E-Learning-Systems mit verschiedenen Schwerpunktsetzungen und ein Beitrag widmet sich einer speziellen Frage der (system-unabhängigen) Spezifikation von variablen Aufgaben im Bereich der Programmier-ausbildung. Wertvolle Kommentare der Reviewer im Programmkomitee konnten dabei zur Schärfung der relevanten Aspekte beitragen und halfen den Autoren, ihre Paper signifikant zu verbessern.

3 Durchführung

Alle Themen der akzeptierten Beiträge versprechen Potenzial für interessante Diskussionen, mit denen das halbtägige Workshop-Programm gut gefüllt werden kann. Dabei wird jeder Beitrag mit einem ca. 20-minütigen Vortrag vorgestellt und anschließend diskutiert. Zusätzlich wird auch noch Raum für eine Arbeitsphase bleiben, in der weitere, übergreifende Themen diskutiert und weitere Forschungsfragen identifiziert werden können.

1st Workshop on Software Engineering for Applied Embedded Real-Time Systems (SEERTS)

Robert Höttger [1], Jörg Teßmer [2]

Abstract: Software engineering for research-intensive domains such as driver-assisted systems or autonomous driving in the automotive industry increasingly require highly sophisticated architectures as well as an optimized, safe, and secure interaction of a large number of actuators, sensors, and networked software components. In addition, connectivity, electric mobility, and heterogeneous development processes introduce new challenges for developers. The corresponding requirements in terms of real-time, causality, security, modularity, scalability or the use of various standards require appropriate domain-specific tools. Model-driven software development often plays an essential role for such tools. The SEERTS Workshop presents adequate technologies for the consideration of diverse and specific requirements within the embedded real-time domain (e.g. for robotics, automotive systems, etc.) and gives insights into their utilization in industrial applications.

Keywords: SEERTS, AUTOSAR, Real-time, Automotive, Software Engineering, Embedded Systems

1 Introduction and Workshop Goals

Embedded real-time systems still undergo development challenges driven by computation, energy, timeliness, or safety demands whilst addressing various design decisions, standards, or architectures. Industrial and research developments continuously have to face those challenges and adapt existing technologies or develop new innovative approaches. For instance, the automotive domain evolves towards an electrified and connected infrastructure intertwined with advanced driver assistant systems or autonomous driving. The dense existence of tools, standards, or frameworks exacerbate to understand systems comprehensively and reusing legacy code may further have spurious influence during the various development cycles.

The SEERTS workshop focuses on technologies to alleviate the spurious influence of diverse constraints in the embedded real-time domain and presents innovative solutions applicable in industrial applications.

[1] Dortmund University of Applied Sciences and Arts, IDiAL Institute, Otto-Hahn-Str. 23, 44227 Dortmund, Germany robert.hoettger@fh-dortmund.de
[2] Robert Bosch GmbH, Stuttgart, Germany Joerg.Tessmer@de.bosch.com

By participating at the SEERTS Workshop, developers get insights into state-of-the-art tools, technologies and applications that meet specific requirements. These special requirements give the SEERTS workshop its individual character, so that domain, real-time, tool, or technology experts can exchange experience, knowledge and know-how. For example, participants involved in parallel automotive systems would be able to learn model-based partitioning and mapping strategies for AMALTHEA models under APP4MC and take advantage of automated software parallelization processes under certain constraints.

Typical technologies involve parallelization strategies [Cu17] [HKI15], scheduling analyzes, tracing methods [HI15], simulation technologies, WCET analysis, resource blocking analysis and protocols [HIS17], models and domain specific languages [Hö17a], and domain specific extensions of existing embedded real-time technologies [Hö17b].

2 Talks and Contributions

With seven talks and eight SEERTS contributions, the workshop covers the above mentioned topics presented by experts from industry and research namely Robert Bosch GmbH, Daimler AG, Vector Informatik GmbH, HS Regensburg, FH Dortmund, DLR e.V., TWT GmbH, and Ulm University. The accepted submissions are entitled:

1. Reasoning on the Length of Trace Recordings for Reverse Engineering of AUTOSAR-compliant Models
2. On Latencies in Automotive Real-time Multi-core Control Systems
3. Quality Indicators for Automotive Test Case Specications
4. Improving the Efficiency of Dislocality Constraints for an Automated Software Deployment in Safety-Critical Systems
5. Multicore Performance Optimization with Open Source Technology Support
6. Software Mapping to Heterogeneous Hardware Using Dynamic Flow and Resource Tracing in Distributed Safety Constrained Many-core Architectures
7. Combining Eclipse IoT Technologies for a RPI3-Rover along with Eclipse Kuksa
8. Design Validation for Embedded Multi-core Systems in the Context of ISO 26262

The submissions were selected by the SEERTS program committee along with three rated reviews for each submission. The accepted submissions cover a comprehensive insight into state-of-the-art development activities within the automotive domain.

References

[Cu17] Cuadra, P.; Krawczyk, L.; Höttger, R.; Heisig, P.; Wolff, C.: Automated Scheduling for Tightly-Coupled Embedded Multi-Core Systems using Hybrid Genetic Algorithms. In: Proceedings of the 2017 International Conference on Information and Software Technologies. ICIST '17, Springer, Oct. 2017.

[HI15] Höttger, R.; Igel, B.: Utilizing Vector-Clocks in an Iterative Tracing Approach for Distributed Embedded Systems. In: 18th International Workshop on Software and Compilers for Embedded Systems. SCOPES'15, ACM, June 2015.

[HIS17] Höttger, R.; Igel, B.; Spinczyk, O.: On Reducing Busy Waiting in AUTOSAR via Task-Release-Delta-based Runnable Reordering. In: Proceedings of the 2017 Design, Automation & Test in Europe Conference & Exhibition. DATE '17, IEEE, pp. 1510–1515, Mar. 2017.

[HKI15] Höttger, R.; Krawczyk, L.; Igel, B.: Model-Based Automotive Partitioning and Mapping for Embedded Multicore Systems. In: International Conference on Parallel, Distributed Systems and Software Engineering. Vol. 2. ICPDSSE'15, World Academy of Science, Engineering and Technology, pp. 2643–2649, Jan. 2015.

[Hö17a] Höttger, R.; Mackamul, H.; Sailer, A.; Steghöfer, J.-P.; Tessmer, J.: APP4MC: Application Platform Project for Multi- and Many-core Systems. In: IT - Information Technology. Methods and Applications of Informatics and Information Technology. Vol. 59, Issue 5, ISSN: 2196-7032, De Gruyter Oldenbourg, Nov. 2017.

[Hö17b] Höttger, R.; Özcelikörs, M.; Heisig, P.; Krawczyk, L.; Wolff, C.; Igel, B.: Constrained Mixed-Critical Parallelization for Distributed Heterogeneous Systems. In: Proceedings of the 2017 Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications. IDAACS '17, IEEE, Sept. 2017.

Hauptprogramm

SE Session 1 - Software Product Lines

M. Tichy, E. Bodden, M. Kuhrmann, S. Wagner, J.-P. Steghöfer (Hrsg.): SE 2018,
Lecture Notes in Informatics (LNI), Gesellschaft für Informatik, Bonn 2018 51

Transformation of Software Product Lines: A Generalizing Framework Based on Category Theory

Gabriele Taentzer[1] Rick Salay[2] Daniel Strüber[3] Marsha Chechik[4]

Keywords: software product lines; variability; model transformation; graph transformation; category theory

1 Abstract

We present our paper from the proceedings of the 2017 edition of the IEEE/ACM International Conference on Model-Driven Engineering Languages and Systems (MODELS) [Ta17].

A *software product line* (SPL) is a portfolio of software products created from a set of common core assets. SPLs enable enterprises to produce custom-tailored products according to their customers' needs. However, a large amount of variability in an SPL can lead to great complexity: From n configuration options or *features*, up to 2^n products can be produced. To cope with this complexity, systematic methods for managing an SPL are required.

The systematic management of an SPL can be supported by expressing potential changes as model transformations. Three kinds of SPL transformations has been considered in the literature: 1. *Changes to feature models* aim to support reasoning about feature additions, deletions, and modifications. Thüm et al. [TBK09] distinguish four categories of changes based on their impact: refactorings, generalizations, specializations, and arbitrary edits. 2. *Lifting of model transformations* [Sa14] aims to make transformation rules from the single-product setting applicable to an entire SPL, represented as a feature-annotated domain model. The intended effect of the transformation is the same as considering each product separately. 3. *SPL refinement* [BTG12] aims to support safe evolution: modifications of the SPL may be restricted so that all or a well-defined subset of all products remain unchanged. None of these approaches allows specifying the *combined* transformation of feature models

[1] Philipps-Universität Marburg, Hans-Meerwein-Str. 1, 35032 Marburg, Germany, taentzer@mathematik.uni-marburg.de
[2] Department of Computer Science, 10 King's College Road, University of Toronto, Toronto, Ontario, Canada, M5S 3G4, rsalay@cs.toronto.edu
[3] Universität Koblenz-Landau, Universitätsstr. 1, 56070 Koblenz, Germany, strueber@uni-koblenz.de
[4] Department of Computer Science, 10 King's College Road, University of Toronto, Toronto, Ontario, Canada, M5S 3G4, chechik@cs.toronto.edu

and domain models. However, such combined transformations are important, since the addition or deletion of features usually entails the corresponding changes in the domain model. Developers need to know whether the overall result is a well-formed SPL again.

To address this need, in this paper, we present a formal framework for rule-based transformations of SPLs based on category theory. Specifically, we make the following contributions:

- We formalize the *category of software product lines*. Based on this category, we obtain a framework that abstracts from the type of model being considered, which makes it applicable to a great variety of models, including UML models and Petri nets.
- We formally define *transformations over SPLs* and show their soundness, i.e., for a given matching site of a rule to an input SPL, each rule application yields a well-defined and unique SPL. We discuss how our framework paves the way for new kinds of SPL analyses with sound tool support.
- We demonstrate the *applicability* of our formalization in three running examples. The flexibility of our framework follows from its capability to support almost all modifications proposed in the aforementioned works. In addition, we support entirely new kinds of modifications resulting from the combinations of the existing ones.

In the future, we will further extend our formal framework so we can use relevant formal results to enable new kinds of product line analyses. Specifically, to show that our framework is amenable to conflict and dependency analysis, we intend to show that our category is $\mathcal{M}$-adhesive, as would then give rise to a theory of product line transformations. Furthermore, we will apply our framework to a set of real-life product lines and provide tool support to make its benefits available to developers.

References

[BTG12] Borba, Paulo; Teixeira, Leopoldo; Gheyi, Rohit: A Theory of Software Product Line Refinement. Theor. Comput. Sci., 455:2–30, 2012.

[Sa14] Salay, Rick; Famelis, Michalis; Rubin, Julia; Di Sandro, Alessio; Chechik, Marsha: Lifting model transformations to product lines. In: International Conference on Software Engineering (ICSE). ACM, pp. 117–128, 2014.

[Ta17] Taentzer, Gabriele; Salay, Rick; Strüber, Daniel; Chechik, Marsha: Transformations of Product Lines: A Generalizing Framework based on Category Theory. In: International Conference on Model Driven Engineering Languages and Systems (MODELS). ACM, pp. 101–111, 2017.

[TBK09] Thüm, Thomas; Batory, Don; Kästner, Christian: Reasoning About Edits to Feature Models. In: International Conference on Software Engineering (ICSE). IEEE, pp. 254–264, 2009.

M. Tichy, E. Bodden, M. Kuhrmann, S. Wagner, J.-P. Steghöfer (Hrsg.): SE 2018,
Lecture Notes in Informatics (LNI), Gesellschaft für Informatik, Bonn 2018 53

Is There a Mismatch between Real-World Feature Models and Product-Line Research?

Alexander Knüppel[1] Thomas Thüm[2] Stephan Mennicke[3] Jens Meinicke[4] Ina Schaefer[5]

Abstract: This work has been presented at the joint meeting of the European Software Engineering Conference and the ACM SIGSOFT Symposium on the Foundations of Software Engineering in Paderborn, Germany [Kn17].

Feature modeling has emerged as the de-facto standard to capture variability of a software product line in a compact and understandable fashion. Multiple feature modeling languages that evolved over the last decades to manage industrial-size product lines have been proposed. However, less expressive languages, solely permitting require and exclude constraints, are permanently and carelessly used in product-line research. We address the problem whether those less expressive languages are sufficient for industrial product lines. We developed an algorithm to eliminate complex cross-tree constraints in a feature model, enabling the combined usage of tools and algorithms working with different feature model dialects in a plug-and-play manner. However, the scope of our algorithm is limited. Our evaluation on large feature models, including the Linux kernel, gives evidence that require and exclude constraints are not sufficient to express real-world feature models. Hence, we promote that research on feature models needs to consider arbitrary propositional formulas as cross-tree constraints prospectively.

Keywords: Software product lines, feature modeling, cross-tree constraints, model transformation, expressiveness, require constraints, exclude constraints

Overview

In this talk, we discuss the role of arbitrary propositional formulas as cross-tree constraints in feature modeling. We argue that the two most prominent kinds of cross-tree constraints, namely *require* and *exclude constraints*, are not enough to capture product lines created for real-world projects. Unfortunately, even nowadays, part of product-line research gives the impression that those simple constraints suffice to describe industrial software product lines.

The main results on this mismatch between newly proposed product-line research on the one hand and feature models of industrial product lines on the other hand have been

[1] TU Braunschweig, Germany, a.knueppel@tu-bs.de
[2] TU Braunschweig, Germany, t.thuem@tu-braunschweig.de
[3] TU Braunschweig, Germany, mennicke@ips.cs.tu-bs.de
[4] University of Magdeburg, Germany; Carnegie Mellon University, USA, meinicke@ovgu.de
[5] TU Braunschweig, Germany, i.schaefer@tu-braunschweig.de

presented at the joint meeting of the European Software Engineering Conference and the ACM SIGSOFT Symposium on the Foundations of Software Engineering in Paderborn, Germany [Kn17]. We showed that, to this day, a significant portion of novel contributions on feature models (e.g., algorithms or empirical evaluations) do not discuss complex constraints. The consequence is that applicability of these contributions on large-scale feature models remains questionable. Our evaluation on real-world feature models of considerable size portraits that simple constraints are not enough.

Although Schobbens et al. [Sc07] were the first to indicate that tree-based feature modeling languages with simple constraints are expressively incomplete, our work extends their prior theory and highlights the differences in expressive power with real numbers. To help the community to overcome this limitation, we developed a semantics-preserving transformation between languages with complex constraints and languages with simple constraints and gave proof of its correctness. The algorithm is based on the notion of abstract features [Th11]. Moreover, we successfully evaluated the benefits of this algorithm on a total of 127 real-world feature models: four monthly snapshots of an obfuscated automotive product line from our industrial partner with up to 18,616 features and 1,369 cross-tree constraints, eight models translated from the KConfig variability language (including the linux kernel), and 116 models translated from the component definition language (CDL).

On the upside, our algorithm is able to eliminate complex constraints in a feature model while preserving the represented set of products. The downside of our algorithm is that, for large feature models, our algorithm may render feature model applications (e.g., SAT analysis) infeasible due to a potential blow-up in the numbers of new features and constraints. However, the elimination of complex constraints is indispensable for practical product-line engineering. We thus advocate that product-line research should consider complex constraints as default in the future. We further think that a community effort is needed to evaluate which and how approaches tailored to basic feature models can be applied to complex constraints.

References

[Kn17] Knüppel, A.; Thüm, T.; Mennicke, S.; Meinicke, J.; Schaefer, I.: Is there a mismatch between real-world feature models and product-line research? In: Proceedings of the 2017 11th Joint Meeting on Foundations of Software Engineering. ACM, pp. 291–302, 2017.

[Sc07] Schobbens, P.-Y.; Heymans, P.; Trigaux, J.-C.; Bontemps, Y.: Generic semantics of feature diagrams. Computer Networks 51/2, pp. 456–479, 2007.

[Th11] Thüm, T.; Kästner, C.; Erdweg, S.; Siegmund, N.: Abstract features in feature modeling. In: Software Product Line Conference (SPLC), 2011 15th International. IEEE, pp. 191–200, 2011.

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

Philipp Hohl[1] Javad Ghofrani[2] Jürgen Münch[3] Michael Stupperich[4] Kurt Schneider[5]

This summary refers to the paper *Searching for Common Ground: Existing Literature on Automotive Agile Software Product Lines* [Ho17]. This paper was published in the *Proceedings of the International Conference on Software and System Process (ICSSP 2017)*.

Keywords: software product line; agile software development; automotive domain

The need for creating digitally enhanced products, services, and experiences as well as the emergence of new or modified business models has a significant impact on the automotive domain. Innovative solutions and new topics such as Smart Mobility or Connectivity require current automotive development processes to undergo major changes. They need to be redesigned in a way that it is possible to learn and adapt continuously at a fast pace. Agile methods are promising approaches to address these new challenges. However, agile methods are not tailored to the specific characteristics of the automotive domain such as software product line (SPLs) development. Although, there have been efforts to apply agile methods in the automotive domain, widespread adoptions have not yet taken place.

We conducted a comprehensive literature study [Ho17] that gives an overview of agile methods for embedded software development in the automotive domain, especially with respect to SPLs. We limited the scope to automotive embedded software development. Typical characteristics of the automotive domain that need to be considered are the deep integration between hardware and software, a strong focus on development processes, a close supplier involvement, and the importance of safety-critical functionality. Furthermore, specific testing conditions like tests in real cars must be considered. The general research question that guides the 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?* This question is divided into three research sub-questions to provide different views on the topic (see Fig. 1).

[1] Daimler AG, Wilhelm-Runge-Straße 11, 89013 Ulm, Germany, Philipp.Hohl@daimler.com
[2] Leibniz Universität Hannover, Welfengarten 1, 30167 Hannover, Germany, javad.ghofrani@inf.uni-hannover.de
[3] University of Reutlingen, Danziger Str. 6, 71034 Böblingen, Germany, juergen.muench@reutlingen-university.de
[4] Daimler AG, Wilhelm-Runge-Straße 11, 89013 Ulm, Germany, Michael.Stupperich@daimler.com
[5] Leibniz Universität Hannover, Welfengarten 1, 30167 Hannover, Germany, kurt.schneider@inf.uni-hannover.de

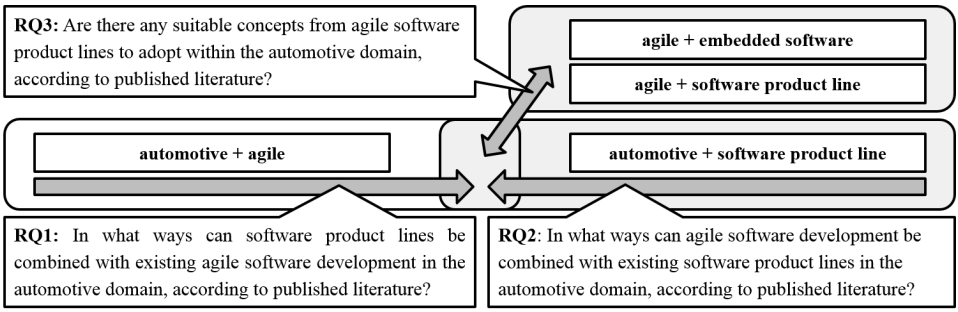


Fig. 1: What is the state-of-the-art to combine agile software development and software product lines in the automotive domain, according to published literature?

The literature review revealed that there is no specific approach tailored to the automotive domain handling the combination of agile software development (ASD) and SPLs. A possible explanation is that combining agile development with existing product lines is challenging in the automotive domain (consider, for instance, the need for synchronization with mechanical and electrical engineering). Searching for existing literature about the introduction of agile methods in the context of automotive product line development was also not fruitful. The study identified areas in the automotive domain where it seems to be comparably easy to introduce agile development without product line development (e.g., entertainment software for the car). Searching for literature that deals with introducing product line engineering to agile development in the automotive domain also did not reveal significant results. However, looking at related domains provided many insights into how the combination of agile methods and product line development could look alike in the automotive domain.

The study identified challenges and possible solutions to combine ASD with SPL. Examples for such solutions are approaches for architecture evolution, lightweight variability management, new communication structures and practices, lean and incremental safety-related procedures, and advanced synchronization mechanisms for hardware and software development cycles. The insights from the study might help as starting points to modify automotive software development processes so that they can benefit from both, agility and systematic reuse through product lines.

References

[Ho17] Hohl, P.; Ghofrani, J.; Münch, J.; Stupperich, M.; Schneider, K.: Searching for common ground: Existing literature on automotive agile software product lines. In (Bendraou, R.; Raffo, D.; LiGuo, H.; Maggi, F. M., eds.): Proceedings of the 2017 International Conference on Software and System Process - ICSSP 2017. ACM Press, New York, New York, USA, pp. 70–79, 2017, ISBN: 9781450352703.

SE Session 2 - Security

M. Tichy, E. Bodden, M. Kuhrmann, S. Wagner, J.-P. Steghöfer (Hrsg.): SE 2018,
Lecture Notes in Informatics (LNI), Gesellschaft für Informatik, Bonn 2018 59

Optimized Cloud Deployment of Multi-tenant Software Considering Data Protection Concerns – Abridged Version

Zoltán Ádám Mann[1] and Andreas Metzger [1]

Abstract: To ensure that a cloud tenant's data cannot be accessed by malicious code from another tenant, critical software components of different tenants are traditionally deployed on separate physical machines, leading to inefficient resource usage. In this paper, we show how secure hardware enclaves can be employed to address data protection concerns of cloud tenants, while optimizing hardware utilization. We provide a model, formalization and experimental evaluation of an efficient algorithmic approach to compute an optimized deployment of software components and virtual machines, taking into account data protection concerns and the availability of secure hardware enclaves.

Keywords: Cloud computing; data protection; privacy; resource optimization; secure enclave

1 Introduction

Ensuring the protection of sensitive data is key for the adoption of cloud services [SMM17]. Data protection concerns may especially arise in a multi-tenant setting, in which the confidentiality of a tenant's data may be breached by malicious code from another tenant. Even if different tenants' code and data are deployed in different virtual machines (VMs), if these separate VMs are deployed on the same physical machine (PM), malicious code in one VM may still breach the confidentiality of data in another VM. To address such security risks, the traditional solution is to physically separate critical code and data of one tenant from the code and data of other tenants by deploying each on different PMs. However, physical separation reduces the opportunity for sharing resources, thus leading to limited hardware utilization and increased costs [Ma15a, Ma15b].

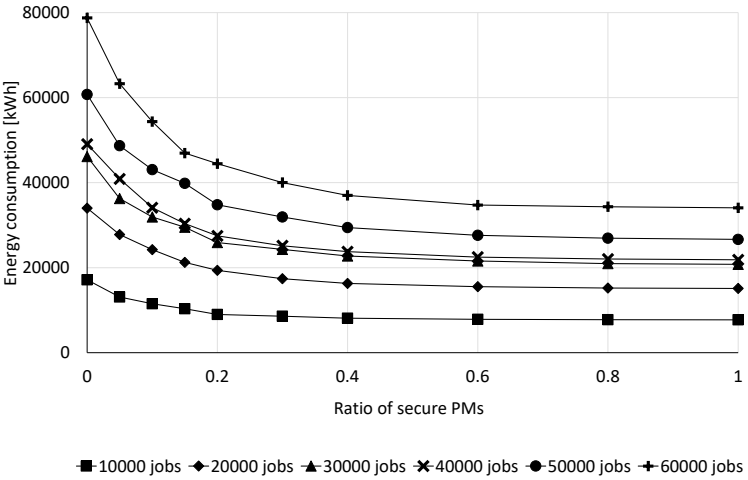
Secure enclaves (such as offered by Intel's SGX technology[2]) provide hardware mechanisms to protect critical code and data, maintaining confidentiality even when an attacker has physical control of the hardware platform and can conduct direct attacks on memory. Secure enclaves thereby make it possible to protect code and data within a PM, thus offering an alternative to physical separation. Since PMs offering secure enclaves are likely to remain a scarce resource in data centers in the near future, a combination of secure hardware and physical separation on traditional hardware appears to be a good compromise to achieve data protection goals while aiming to optimize resource usage.

[1] paluno – The Ruhr Institute for Software Technology, University of Duisburg-Essen, Essen, Germany. {zoltan.mann, andreas.metzger}@paluno.uni-due.de

[2] Software Guard Extensions, see `https://software.intel.com/en-us/sgx`

2 Approach and results

In our original paper [MM17], we defined and formalized a cloud deployment model considering data protection concerns. We introduced efficient heuristic algorithms to compute an optimized cloud deployment for tenant components (i.e., code and data) and VMs, taking into account data protection concerns and capacity constraints. By means of a comprehensive empiric evaluation with real workload data, we analyzed how cost savings depend on data security properties. In particular, we found that even if only 20% of the PMs offer secure hardware enclaves, savings of energy consumption (which is a major cost driver) may be as high as 47.5%. This is also exemplified by the following figure.



Acknowledgement. This work has received funding from the European Union's Horizon 2020 research and innovation programme under grant no. 731678 (RestAssured).

References

[Ma15a] Mann, Zoltán Ádám: Approximability of virtual machine allocation: much harder than bin packing. In: Proceedings of the 9th Hungarian-Japanese Symposium on Discrete Mathematics and Its Applications. pp. 21–30, 2015.

[Ma15b] Mann, Zoltán Ádám: Modeling the virtual machine allocation problem. In: Proceedings of the International Conference on Mathematical Methods, Mathematical Models and Simulation in Science and Engineering. pp. 102–106, 2015.

[MM17] Mann, Zoltán Ádám; Metzger, Andreas: Optimized Cloud Deployment of Multi-tenant Software Considering Data Protection Concerns. In: Proceedings of the 17th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing. pp. 609–618, 2017.

[SMM17] Schoenen, Stefan; Mann, Zoltán Ádám; Metzger, Andreas: Using Risk Patterns to Identify Violations of Data Protection Policies in Cloud Services. In: 13th International Workshop on Engineering Service-Oriented Applications and Cloud Services. 2017.

Detecting Information Flow by Mutating Input Data

Björn Mathis[1] Vitalii Avdiienko[2] Ezekiel O. Soremekun[3] Marcel Böhme[4]
Andreas Zeller[5]

1 Summary

This article is an abridged version of our article "Detecting Information Flow by Mutating Input Data" which is published in the proceedings of the thirty-second IEEE/ACM International Conference on Automated Software Engineering [Ma17].

When assessing the security of applications, *information flows* play an essential role: Which information sources does the application access, and to which sinks does it send these to? Static and dynamic analyses of information flow are easily challenged by non-available or obscure code.

Static flow detection tools like the state-of-the-art FlowDroid [Ar14] are effective but they suffer from the principal limitations of static analysis, notably that *all* code must be available for analysis. One example is native code, a piece of code that runs directly on the processor and whose source is written in C or C++—in contrast to the Dalvik byte code derived from the Java source. FlowDroid will miss a flow through native code, since it cannot analyze it and needs to either under- or overapproximate.

In contrast to static analysis, dynamic analysis allows to analyze concrete executions rather than abstract code. *Dynamic tainting*, for instance, tracks data throughout the execution. Finding a flow through native code, though, would require to track data through the hardware or a hardware interpreter, which is no simple feat. Since existing dynamic and static tools need to analyze the concrete path along which the information travels, they cannot detect deliberately hidden flows.

In this work, we investigate a lightweight *mutation-based* alternative to detect information flows called MutaFlow. Rather than statically analyzing application code or dynamically tracking data flow, we use an *experimental* approach: We systematically *mutate the*

[1] Saarland University, Saarbrücken, Germany mathis@st.cs.uni-saarland.de
[2] Saarland University, Saarbrücken, Germany avdiienko@st.cs.uni-saarland.de
[3] Saarland University, Saarbrücken, Germany soremekun@st.cs.uni-saarland.de
[4] Monash University, Melbourne, Australia marcel.boehme@acm.org
[5] Saarland University, Saarbrücken, Germany zeller@st.cs.uni-saarland.de

information sources of a program to assess whether the mutation impacts its information sinks. Specifically, our MutaFlow prototype *a)* takes an Android app as well as a set of test cases (given or generated), *b)* instruments *sensitive data sources and sinks* in the app to *mutate* input values at sources and track output values at sinks, *c)* executes tests on unmutated and mutated app versions, *d)* records the values passed to sensitive sinks, reporting a flow if the value changes due to a mutated sink value.

The advantage of MutaFlow over static analysis approaches is that it hardly overapproximates: It is *sound* in the sense that if it detects a flow, this flow is most likely real. The previous problem that data flows through a native method has no consequences for MutaFlow; it only cares about how changes in sources affect sinks, without having to track the path. All changes made by MutaFlow simulate changes in the external input data; the actual program functionality is never altered. However, as any dynamic approach, it is also *incomplete*. In general, we can only find information flows in executed code, and the number of found flows depends on the test suite's code coverage.

In our evaluation, we assessed the effectiveness of MutaFlow using two sets of Android apps, namely 20 real-world applications and *DroidBench* (a collection of 120 small Android applications with several categories of information flows that are obfuscated in one way or another).

Even though MutaFlow requires less than 10% of source code compared to FlowDroid, it has a similar accuracy. On the real-world applications MutaFlow achieves an accuracy of 81%, 60% more than FlowDroid and is able to detect 75 flows that FlowDroid misses. FlowDroid finds 17 unique flows and 2 flows are found by both tools. On Droidbench, MutaFlow has an accuracy of 72%, 14% higher than FlowDroid.

Hence, MutaFlow could be seen as a *complement* to static analysis approaches, focusing on problem areas such as non-analyzable code. However, mutation-based flow analysis can also be seen as an *alternative* analysis, should static analysis not be available or possible. This is because the *accuracy of MutaFlow in detecting flows is similar, if not superior, to static analysis tools such as FlowDroid*. This poses mutation-based flow analysis as a new and promising alternative in our portfolio of program analysis techniques.

References

[Ar14] Arzt, Steven; Rasthofer, Siegfried; Fritz, Christian; Bodden, Eric; Bartel, Alexandre; Klein, Jacques; Le Traon, Yves; Octeau, Damien; McDaniel, Patrick: FlowDroid: Precise Context, Flow, Field, Object-sensitive and Lifecycle-aware Taint Analysis for Android Apps. In: Proceedings of the 35th ACM SIGPLAN Conference on Programming Language Design and Implementation. PLDI '14, pp. 259–269, 2014.

[Ma17] Mathis, Björn; Avdiienko, Vitalii; Soremekun, Ezekiel O.; Böhme, Marcel; Zeller, Andreas: Detecting Information Flow by Mutating Input Data. In: Proceedings of the 32Nd IEEE/ACM International Conference on Automated Software Engineering. ASE 2017, IEEE Press, Piscataway, NJ, USA, pp. 263–273, 2017.

Integrating BPMN- and UML-based Security Engineering via Model Transformation

Qusai Ramadan[1], Mattia Salnitri[2], Daniel Strüber[3], Jan Jürjens[4], Paolo Giorgini[5]

Keywords: model-driven security; security by design; BPMN; UML; model transformation

1 Abstract

We present our paper from the proceedings of the 2017 edition of the IEEE/ACM International Conference on Model-Driven Engineering Languages and Systems (MODELS) [Ra17].

Most software systems today are embedded into Socio-Technical Systems (STSs), in which organizational and technical components are coordinated to accomplish shared objectives. Establishing security is particularly challenging in STSs since security issues may be propagated along these components, which are generally autonomous and hard to control. Previous work supports the management of organizational and technical security requirements in the early stages of the development process. *BPMN-based approaches* such SecBPMN2 [SDG14] abstract from technical details to support business analysts in specifying organizational security requirements as part of business processes for the target STS. *UML-based approaches* such as UMLsec [Jü05] support system developers in specifying technical security requirements and security-related assumptions in architectural models. The resulting models can be validated against pre-defined technical security policies.

Since these existing BPMN- and UML-based approaches address security in distinct development phases and from distinct stakeholders' perspectives, they deal with security requirements separately; vulnerabilities may arise from misunderstandings between these stalkeholders, in particular due to the divergent use of terminology. For instance, without the knowledge that BPMN's *swimlanes* are commonly used to express internal roles in an organization, a system developer may forgo the use of an appropriate role enforcement mechanism. Such loopholes may be hard to detect, since traceability mechanisms for security requirements across the different phases are usually not available.

[1] Universität Koblenz-Landau, Universitätsstr. 1, 56070 Koblenz, Germany, qramadan@uni-koblenz.de
[2] University of Trento, Via Sommarive 9, I-38050, Povo (Trento) Italy, mattia.salnitri@unitn.it
[3] Universität Koblenz-Landau, Universitätsstr. 1, 56070 Koblenz, Germany, strueber@uni-koblenz.de
[4] Universität Koblenz-Landau, Universitätsstr. 1, 56070 Koblenz, Germany and Fraunhofer Institute for Software and Systems Engineering, Emil-Figge-Str. 91, 44227 Dortmund, Germany, juerjens@uni-koblenz.de
[5] University of Trento, Via Sommarive 9, I-38050, Povo (Trento) Italy, paolo.giorgini@unitn.it

To address these issues, we propose a framework for designing secure STSs by integrating existing BPMN-based and UML-based security engineering approaches. Our framework involves four main tasks: (i) Organizational security requirements are modeled by business analysts using SecBPMN2, (ii) the SecBPMN2 model is transformed to a preliminary UMLsec-annotated architectural model automatically, using a Henshin-based model transformation [St17], (iii) the generated UMLsec model is manually refined by system developers, and (iv) the resulting UMLsec model is verified against the contained security policies automatically using a tool called CARiSMA [Ah17].

The main contributions of the current paper are:

- a *semi-automated process* that enforces security management throughout the development process in an integrated manner,
- a *model transformation* that supports the translation of security-annotated business models to security-annotated architectural models while establishing traceability, and
- a *case study* featuring an air traffic management system, in which our framework was used to establish integrated management and traceability of security requirements.

Our framework automatically establishes traceability between high-level security requirements and verifiable technical security policies. Doing so, it integrates the perspectives of business analysts and system developers, the main expert stakeholders in STS development.

An accompanying artifact for our paper (`http://www.remodd.org/v1/content/models2017-bpmn2uml-artifacts`) was accepted at MODELS's Artifact Evaluation track.

References

[Ah17] Ahmadian, Amir Shayan; Peldszus, Sven; Ramadan, Qusai; Jürjens, Jan: Model-based privacy and security analysis with CARiSMA. In: Joint Meeting on Foundations of Software Engineering (ESEC/FSE). pp. 989–993, 2017.

[Jü05] Jürjens, Jan: Secure systems development with UML. Springer Science & Business Media, 2005.

[Ra17] Ramadan, Qusai; Salnitri, Mattia; Strüber, Daniel; Jürjens, Jan; Giorgini, Paolo: From Secure Business Process Modeling to Design-Level Security Verification. In: ACM/IEEE International Conference on Model Driven Engineering Languages and Systems (MODELS). pp. 123–133, 2017.

[SDG14] Salnitri, Mattia; Dalpiaz, Fabiano; Giorgini, Paolo: Modeling and verifying security policies in business processes. In: Enterprise, Business-Process and Information Systems Modeling (BMMDS/EMMSAD), pp. 200–214. Springer, 2014.

[St17] Strüber, Daniel; Born, Kristopher; Gill, Kanwal Daud; Groner, Raffaela; Kehrer, Timo; Ohrndorf, Manuel; Tichy, Matthias: Henshin: A Usability-Focused Framework for EMF Model Transformation Development. In: International Conference on Graph Transformation (ICGT). pp. 196–208, 2017.

SE Session 3 - Processes and Evolution of Software Engineering

M. Tichy, E. Bodden, M. Kuhrmann, S. Wagner, J.-P. Steghöfer (Hrsg.): SE 2018,
Lecture Notes in Informatics (LNI), Gesellschaft für Informatik, Bonn 2018 67

Predictive Business Process Monitoring unter Berücksichtigung von Prognoseverlässlichkeit und Risiko

Andreas Metzger[1], Philipp Bohn[2], Felix Föcker[3]

Abstract: Wir stellen Techniken und Experimentergebnisse für die Berücksichtigung der Prognoseverlässlichkeit und des Risikos beim Predictive Business Process Monitoring vor. Die Berücksichtigung dieser Größen bietet zusätzliche Entscheidungsgrundlagen für die proaktive Prozessanpassung.

Keywords: Prognose, Machine Learning, Business Process Management, Risiko.

1 Überblick

Motivation. Predictive Business Process Monitoring zielt darauf ab, potenzielle Probleme in der Prozessausführung vorherzusagen. Predictive Business Process Monitoring ermöglicht den frühzeitigen Eingriff und somit die proaktive Anpassung laufender Prozesse [MLI17].

Die bisherige Forschung betrachtete die aggregierte Genauigkeit aller Vorhersagen (z. B. berechnet mittels Precision oder Recall). Die Genauigkeit und somit die Verlässlichkeit („Reliability“) einer einzelnen Prognose und damit auch das Risiko einer Entscheidung auf Basis dieser Prognose wurden nicht untersucht. Unsere Beiträge auf der 29th Int'l Conference on Advanced Information Systems Engineering (CAiSE) und der 15th Int'l Conference on Service Oriented Computing (ICSOC) adressieren diese Lücke.

Berücksichtigung von Prognoseverlässlichkeit. In unserem CAiSE-Beitrag (siehe [MeF17]) konnten wir zeigen, wie Verlässlichkeitsschätzungen für einzelne Prognosen mittels Ensemble-Vorhersagetechniken berechnet werden können. Unsere experimentelle Analyse anhand eines Industriedatensatzes aus der Logistik zeigt, dass die Berücksichtigung der Verlässlichkeit bei der proaktiven Prozessanpassung Kostenersparnisse bis zu 54% (14% im Mittel) ermöglicht.

Berücksichtigung von Risiko. In unserem ICSOC-Beitrag (siehe [MeB17]) wurde der obige Ansatz dahingehend erweitert, dass das Risiko der Adaptionsentscheidung berechnet und berücksichtigt wird. Abbildung 1 gibt einen Überblick über den erweiterten Ansatz. Die Prozessüberwachungsdaten werden in ein Ensemble-Vorhersagemodell einge-

[1] paluno, Univ. Duisburg-Essen, 45127 Essen, andreas.metzger@paluno.uni-due.de
[2] paluno, Univ. Duisburg-Essen, 45127 Essen, philipp.bohn@paluno.uni-due.de
[3] paluno, Univ. Duisburg-Essen, 45127 Essen, felix.foecker@paluno.uni-due.de

geben, welches die Abweichung der Prozessinstanz zur geplanten Ausführungsdauer vorhersagt (δ), sowie die Verlässlichkeit der Vorhersage schätzt (ρ). Die Verlässlichkeitsschätzung quantifiziert die Wahrscheinlichkeit des Risikos. Mittels einer parametrierbaren Kostenfunktion, cost(δ), wird die Vertragsstrafe bei einer Überschreitung der erwarteten Prozessdauer berechnet und somit das Ausmaß des Risikos *r* quantifiziert.

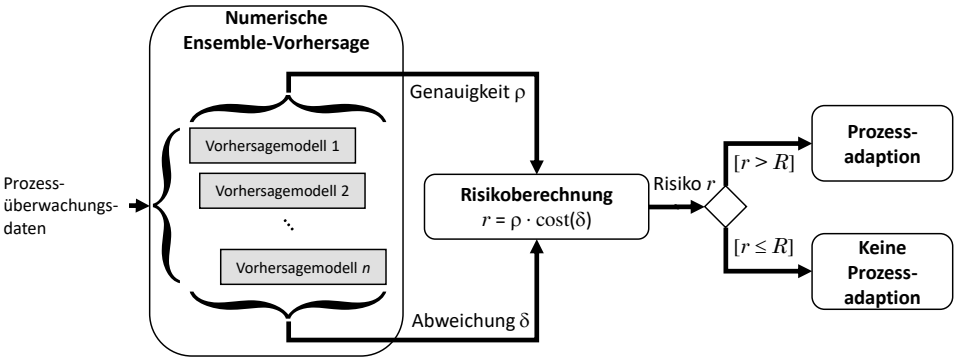


Abbildung 1: Überblick des Ansatzes zur risikobasierten Prozessadaption

Im Vergleich zum CAiSE-Beitrag konnten zusätzliche Kostenersparnisse bis zu 31% (15% im Mittel) gemessen werden. Werden lediglich die Ergebnisse nicht-konstanter Kostenfunktionen berücksichtigt, ergibt sich sogar eine Kostenersparnis von 23% im Mittel.

Ausblick. Als zukünftige Arbeiten planen wir weitergehende empirische Studien mit Prozessen und Datensätzen aus der Hafenlogistik und dem E-Commerce. Darüber hinaus wollen wir untersuchen, inwiefern der Prognosezeitpunkt bei der Adaptationsentscheidung berücksichtigen werden kann.

Die Ergebnisse entstanden im Rahmen des europäischen H2020-Projekts „Transforming-Transport“ (Förderkennzeichen 731932; https://transformingtransport.eu/).

Literaturverzeichnis

[MLI17] Metzger, A. et al.: Vergleich und Kombination von Techniken des Predictive Business Process Monitoring. In (Jürjens, J.; Schneider, K.; Hrsg.): Software Engineering 2017, Konferenz des GI-Fachbereichs Softwaretechnik, Hannover, LNI 267, GI, 2017.

[MeF17] Metzger, A.; Föcker, F.: Predictive business process monitoring considering reliability estimates. In (Dubois, E.; Pohl. K.; Hrsg.): Int’l Conference on Advanced Information Systems Engineering (CAiSE 2017), Essen, LNCS 10253, Springer, 2017.

[MeB17] Metzger, A.; Bohn, P.: Risk-based proactive process adaptation. In (Maximilian, M.; Vallecillo, A.; Wang, J.; Oriol, M.; Hrsg.): Int’l Conference on Service Oriented Computing (ICSOC 2017), Málaga, LNCS 10601, Springer, 2017.

Softwareentwicklung braucht mehr Gestaltungkompetenz: Digital Design als neues Rollenideal im Software Engineering

Kim Lauenroth[1]

1 Herleitung

In der Softwareentwicklung geschieht Gestaltung meist implizit, bspw. durch die Formulierung von Anforderungen eines sogenannten Fachbereichs / Product Owners als Vorgabe für eine umsetzende IT-Organisationen / ein Team. In klassischen Entwicklungssituationen (d.h. der Übertragung verstandener analoger Prozesse in IT-Systeme) war diese Form der Arbeitsteilung zweckmäßig und sinnvoll. Die Herausforderung bestand im Wesentlichen darin, die verstandene Fachlichkeit geeignet einem Entwicklungsprozess zuzuführen.

Für neue Entwicklungssituationen, gerne durch Digitalisierung oder digitale Transformation umschrieben, funktioniert dies nicht mehr (vgl. [La17]): Bei der Digitalisierung existieren zwar analoge Vorbilder, aber es ist keinesfalls sichergestellt, dass Nutzer das Digitale dem Analogen vorziehen. Beispielsweise war nicht klar, dass Nutzer den Bücherkauf im Internet dem Buchladen vorziehen. Folglich müssen digitale Prozesse nicht nur technisch gut gemacht sein, die Nutzer müssen diese auch akzeptieren. Bei der digitalen Transformation existieren keine analogen Vorbilder, da vollkommen neue Strukturen entstehen. Damit ist die Akzeptanz mindestens genauso unsicher. Ein Beispiel hierfür ist Second Life. Dieses Projekt war in den Anfangsjahren revolutionär, hat aber im Laufe der Zeit die Erwartungen nicht erfüllt und ist mittlerweile im Vergleich zu anderen Entwicklungen uninteressant geworden.

Digitalisierung oder digitale Transformation sind folglich eine Herausforderung für viele Unternehmen [Br17]. Die Verantwortlichen müssen nicht nur Software, sondern meist ein ganzes Ökosystem gestalten. Weiterhin ist das Risiko des Scheiterns wesentlich größer, da Digitalvorhaben von ihrer Natur wesentlich stärker auf Vermutungen und Annahmen über den Erfolg und die Auswirkungen des Vorhabens beruhen.

2 Digital Design als neues Rollenideal

Um diesen Herausforderungen zu begegnen, hat die Bitkom-Taskforce „Software-Gestalter“ den Digital Designer [Bi17] als ein Rollenideal erarbeitet, dass eine aktive

[1] adesso AG, Stockholmer Allee 20, 44269 Dortmund, kim.lauenroth@adesso.de

Gestaltungsrolle in der Softwareentwicklung einnehmen soll:

Digital Designer gestalten und optimieren digitale Produkte, Systeme und Dienstleistungen. Sie berücksichtigen dabei das Spannungsfeld zwischen den Wünschen und Bedürfnissen der Nutzer, den wirtschaftlichen Rahmenbedingungen und den technischen Möglichkeiten. Digital Designer führen den Entwicklungsprozess durch Skizzen, Modelle, Spezifikationen und Prototypen. Sie arbeiten dabei in multidisziplinären Gruppen mit dem Management, dem Marketing, der Entwicklung und dem Betrieb von Software.

Der Digital Design folgt dem Vorbild des Industriedesigners und ist eine dritte Führungsrolle in der Softwareentwicklung. Er führt auf Augenhöhe mit Projektleitern und Software-Architekten den Entwicklungsprozess von Software. Er bringt eigene Ideen und Vorschläge ein, um die potenziale von Software auszuschöpfen. Das Kompetenzprofil des Digital Designers hat zwei Schwerpunkte: Gestaltungskompetenz und Materialkunde. Die Materialkunde ist dem Industriedesign entlehnt. Nur durch ein profundes Verständnis verschiedener Materialen können Industriedesigner hochwertiger und nachhaltiger Produkte gestalten (vgl. [Se07]). Digital Design muss analog über ein fundiertes Materialwissen um die Fähigkeiten und Grenzen von Software und IT-Technik verfügen, um dieses Wissen frühzeitig in die Gestaltung von Software einzubringen.

Eine wichtige Komponente der Gestaltungskompetenz ist Methodenkompetenz. Traditionell passt sich das RE/UX an die im jeweiligen Vorhaben vorgefundene Vorgehensweise an (bspw. V-Modell oder Agile). In innovativen Kontexten hängt die Qualität der entwickelten Systeme maßgeblich vom Prozess der Ideenentwicklung ab. Hierfür haben sich eigene Vorgehensweisen entwickelt (bspw. Design Thinking, Lean Startup oder Design Sprint). Das Digital Design kennt diese Vorgehensweisen und kann das passende Vorgehen definieren, um möglichst hochwertige Ideen zu entwickeln.

3 Ausblick

Das Spannungsfeld zwischen Gestaltung und Ingenieurwesen in der Softwareentwicklung ist kein neues Phänomen (vgl. [Sc96]). Vor dem Hintergrund aktueller Entwicklungen (Stichwort Digitalisierung) müssen sich Informatik und Softwareentwicklung neu erfinden [Br17]. Die Etablierung des Digital Designer ist ein Teil dieses Prozesses und soll die Softwareentwicklung hinsichtlich Gestaltungskompetenz vervollständigen.

4 Quellenangaben

[Bi17] Bitkom – Leitfaden Rollenideal „Digital Design“, 2017.

[Br17] Brenner, W., Broy, M., Leimeister, J.M.: Auf dem Weg zu einer Informatik neuer Prägung in Wissenschaft, Studium und Wirtschaft. Informatik Spektrum Band 40, Heft 7, Dez. 2017.

[La17] Lauenroth, K.: Analog ist das neue Bio – Digital Design als neues Rollenideal. Business Technology 4.17, 2017.

[Sc96] Schulz-Schaeffer, I.: Software-Entwicklung zwischen Ingenieur- und Design-wissenschaft. Überzeugungskraft und nützliche Widersprüchlichkeiten von Software-Engineering und Software-Gestaltung. In: Hellige, H.D. (Hrsg.) Technikleitbilder auf dem Prüfstand, 1996.

[Se07] Selle, G.: Geschichte des Design in Deutschland. Campus 2007.

SE Session 4 - Testing

M. Tichy, E. Bodden, M. Kuhrmann, S. Wagner, J.-P. Steghöfer (Hrsg.): SE 2018,
Lecture Notes in Informatics (LNI), Gesellschaft für Informatik, Bonn 2018 75

Reinforcement Learning for Automatic Test Case Prioritization and Selection in Continuous Integration

Helge Spieker[1] Arnaud Gotlieb[1] Dusica Marijan[1] Morten Mossige[2]

Abstract: The paper appeared at the International Symposium on Software Testing and Analysis (ISSTA 2017) [Sp17]. It is part of a project on test case prioritization, selection, and execution in Continuous Integration (CI) (see also [Mo17]). Selecting the most promising test cases to detect bugs is hard if there are uncertainties on the impact of committed code changes or if traceability links between code and tests are not available. This paper introduces RETECS, a new method for automatically learning test case selection and prioritization in CI with the goal to minimize the round-trip time between code commits and developer feedback on failed test cases. RETECS uses reinforcement learning to select and prioritize test cases according to their duration, previous last execution and failure history. In a constantly changing environment, where new test cases are created and obsolete test cases are deleted, the RETECS method learns to prioritize test cases, which are most likely to fail, higher under the guidance of a reward function and by observing previous CI cycles.

Keywords: Software Testing; Machine Learning; Continuous Integration

1 Extended Abstract

Testing in CI requires tight control over the selection and prioritization of the most promising test cases. By most promising, we mean test cases that are prone to detect failures early in the process. In absence of traceability links between code and test cases and based on the hypothesis that test cases having failed in the past are more likely to fail in the future, *history-based test case prioritization* schedules these test cases first in new CI cycles. Testing in CI also means to control the time required to execute a complete cycle. As test case durations vary, not all tests can be executed and *test case selection* is required.

We argue that these two aspects of CI testing can hardly be solved by using only non-adaptive methods. First, the time allocated to test case selection and prioritization in CI is limited. So, time-effective methods shall be privileged over costly and complex prioritization algorithms. Second, history-based prioritization is not well adapted to changes in the execution environment. More precisely, it is frequent to see some test cases being removed from one cycle to another because they test an obsolete feature. At the same time, new test cases are introduced to test new or changed features. Additionally, some test cases are more

[1] Simula Research Laboratory, Lysaker, Norway, firstname@simula.no
[2] University of Stavanger and ABB Robotics, Norway, morten.mossige@uis.no

crucial in certain periods of time, because they test features on which customers focus the most, and then they loose their prevalence because the testing focus has changed. In brief, non-adaptive methods may not be able to spot changes in the importance of some test cases over others because they apply systematic prioritization algorithms.

This paper introduces RETECS, a lightweight test case selection and prioritization approach in CI based on reinforcement learning (RL). RL is well-tuned to design an adaptive method capable to learn from its experience of the execution environment. Our method has the goal to select and prioritize test cases which have been successfully used to detect faults in previous CI cycles, and to order them to execute the most promising ones first. Our method is able to adapt to situations where test cases are added to or deleted from a general repository, while progressively improving its efficiency. It can also adapt to situations where the testing priorities change because of different focus or execution platforms, indicated by changing failure indications. The time required for prioritization is negligible, which is an important aspect in CI. RETECS does not mine code repositories or change-history to compute a new test case schedule. At each cycle, after the tests have been executed, the knowledge to make decisions is updated from feedback provided by a reward function, the only component initially embedding domain knowledge.

The contributions of the paper are threefold: 1) The paper shows that history-based test case prioritization and selection can be approached as a reinforcement learning problem. By modeling the problem with notions such as states, actions, agents, policy, and reward functions, we demonstrate, that RL is suitable to automatically prioritize and select test cases; 2) Implementing an online RL method, without any previous training phase, into a Continuous Integration process is shown to be effective to learn how to prioritize test cases. Comparing two distinct representations (i.e., tableau and neural networks) and three distinct reward functions, our experimental results show that, without any prior knowledge and without any model of the environment, the RL approach is able to learn how to prioritize test cases better than other approaches. Remarkably, the number of cycles required to improve on other methods corresponds to less than 2-months of data, if there is only one CI cycle per day; 3) Experimental results have been computed on industrial data gathered over one year of Continuous Integration. Results show that the method is deployable in industrial settings.

References

[Mo17] Mossige, M.; Gotlieb, A.; Spieker, H.; Meling, H.; Carlsson, M.: Time-Aware Test Case Execution Scheduling for Cyber-Physical Systems. In: Proceedings of the 23rd International Conference on Principles and Practice of Constraint Programming. Vol. 10416. LNCS, Springer, pp. 387–404, 2017.

[Sp17] Spieker, H.; Gotlieb, A.; Marijan, D.; Mossige, M.: Reinforcement Learning for Automatic Test Case Prioritization and Selection in Continuous Integration. In: Proceedings of 26th International Symposium on Software Testing and Analysis (ISSTA'17). ACM, pp. 12–22, 2017.

M. Tichy, E. Bodden, M. Kuhrmann, S. Wagner, J.-P. Steghöfer (Hrsg.): SE 2018,
Lecture Notes in Informatics (LNI), Gesellschaft für Informatik, Bonn 2018 77

Code Defenders: Crowdsourcing Effective Tests and Subtle Mutants with a Mutation Testing Game

José Miguel Rojas[1] Thomas D. White[2] Benjamin S. Clegg[2] Gordon Fraser[3]

Abstract: Mutation testing aims to improve test suites by seeding artificial faults (mutants) that good tests should identify, and test generation tools can be used to automatically generate these tests. However, mutation tools tend to produce huge numbers of mutants, many of which are trivial, redundant, or semantically equivalent, and automated test generation tools are good at achieving code coverage, but are otherwise weak and have no clear purpose. We present an approach based on gamification and crowdsourcing to produce better tests and mutants: The Code Defenders game lets teams of players compete over a program, where attackers try to create subtle mutants, which defenders try to counter by writing tests. Our experiments suggest that writing tests as part of the game is more enjoyable, and that playing Code Defenders results in stronger test suites and mutants than those produced by automated tools.

Keywords: Gamification; crowdsourcing; mutation testing

1 Extended Abstract

Software needs to be thoroughly tested in order to remove bugs. To evaluate how thoroughly a program has been tested, the idea of mutation testing is to measure the number of seeded artificial bugs (mutants) a test suite can distinguish from the original program. Testers are guided to improve test suites by writing new tests that target previously undetected mutants. However, writing good tests is difficult and developers are often reluctant to do so, in particular since mutation tools tend to produce huge amounts of mutants, many of which are trivial, redundant, and sometimes even semantically equivalent to the original program. A commonly proposed solution is to generate tests automatically, but humans tend to write tests that are stronger, have a clear meaning, and are typically more readable.

To produce better mutants and tests, we use *gamification* and *crowdsourcing*: Gamification converts tasks to components of entertaining gameplay, where the competitive nature of humans motivates them to compete and excel. Crowdsourcing encodes and assigns a difficult problem to an undefined group of workers (a crowd), who provide their solutions back to the requester; the requester then derives the final solution from the solutions collected from

[1] Department of Informatics, University of Leicester, Leicester, UK, j.rojas@leicester.ac.uk
[2] Department of Computer Science, University of Sheffield, Sheffield, UK, {tdwhite1,bsclegg1}@sheffield.ac.uk
[3] Lehrstuhl für Software Engineering II, Universität Passau, Germany, gordon.fraser@uni-passau.de

the workers. We present CODE DEFENDERS [Ro17], a multi-player game in which players contribute to creating good tests and mutants.

In the CODE DEFENDERS game, players compete over a program under test: *Attackers* try to create subtle, hard to kill mutants by editing the source code of a Java class under test. They see all mutants in the game including their code diffs, and the code editor highlights the line coverage achieved so far. The highlighting reflects how often lines are covered; the more often a line is covered, the darker the highlighting is. *Defenders* try to create tests that can detect and counter these attacks. They see the source code of the class under test together with the locations of live and dead mutants. In their code editor, they are given a template to write a unit test for the CUT, and they also see previous tests as well as their coverage. Since mutants may be semantically equivalent, the gameplay integrates *duels*, where defenders can challenge attackers by requesting them to prove non-equivalence using a test. The game is driven by a point scoring system, where attackers score points whenever their mutants are covered but not detected by a test, and defenders score points whenever they succeed in detecting a mutant. Points are also at stake in the equivalence duels, where attackers can lose all points earned with a mutant if they do not succeed in showing non-equivalence.

To evaluate the gamification aspects of CODE DEFENDERS we conducted a controlled study, comparing it to traditional unit testing in terms of the objective performance and subjective perception of 41 participants. All participants of this experiment confirmed that playing the game is fun, and that writing tests as part of CODE DEFENDERS is more enjoyable than doing so outside the game. Code coverage and mutation scores are higher compared to tests written outside the game. To evaluate the crowdsourcing application, we studied 20 multi-player games played on open source classes. The tests resulting from these games are stronger than those generated by automated test generation tools, and the mutants resulting from the games are stronger than those created by state-of-the-art mutation analysis tools.

Besides the crowdsourcing application, CODE DEFENDERS is also naturally suited for educational purposes [RF16, CRF17]. For this, it is available as open-source project and freely available to play online at `http://www.code-defenders.org`.

References

[CRF17] Clegg, Benjamin S; Rojas, José Miguel; Fraser, Gordon: Teaching software testing concepts using a mutation testing game. In: Proceedings of the 39th International Conference on Software Engineering: Software Engineering and Education Track. IEEE Press, pp. 33–36, 2017.

[RF16] Rojas, José Miguel; Fraser, Gordon: Teaching Mutation Testing using Gamification. In: European Conference on Software Engineering Education (ECSEE). 2016.

[Ro17] Rojas, José Miguel; White, Thomas D; Clegg, Benjamin S; Fraser, Gordon: Code Defenders: Crowdsourcing effective tests and subtle mutants with a mutation testing game. In: Proceedings of the 39th International Conference on Software Engineering. IEEE Press, pp. 677–688, 2017.

M. Tichy, E. Bodden, M. Kuhrmann, S. Wagner, J.-P. Steghöfer (Hrsg.): SE 2018,
Lecture Notes in Informatics (LNI), Gesellschaft für Informatik, Bonn 2018 79

Quality Experience: A Grounded Theory of Successful Agile Projects Without Dedicated Testers

Lutz Prechelt[1], Holger Schmeisky[2], Franz Zieris[3]

This is an extended abstract of the paper with the same title [PSZ16] which was presented at the 38th International Conference on Software Engineering (2016).

Keywords: agile software development; quality assurance; industrial case study; grounded theory

1 Background, Context, and Research Method

Some agile development teams have team members or partner teams who are exclusively considered testers, whereas other teams do not even a separate tester *role*. Agile development methods emphasize personal communication and shun departments in favor of multidisciplinary teams. They have no uniform attitude towards an explicit tester role, let alone to the need for separate testing personnel: Scrum only talks about *Developers* [SS13], XP defines a *Tester* role (which is not necessarily to be assumed by non-developers) [BA04], and Kanban is agnostic on the issue of separate testers [An10].

In our exploratory, holistic, multiple-case study [Yi03], we wanted to find out how quality is assured in agile teams that do not employ separate testers and what the advantages and disadvantages of not employing separate testers are. We selected three agile teams (one with separate testers, two without) from two companies, all of which have a similar context: In-house development of large web portal that has millions of users, which are partially paying for particular services; any individual customer accounts for only a small part of the revenue stream; new versions of the software can (in principle) be deployed immediately.

We collected data through direct observation and semi-structured interviews with developers, product owners, scrum masters, testers, and higher-level managers in multiple rounds. We analyzed the transcripts using the Grounded Theory Methodology [SC90].

[1] Freie Universität Berlin, Institut für Informatik, Takustr. 9, 14195 Berlin, Deutschland prechelt@inf.fu-berlin.de
[2] Zalando SE, Berlin, Deutschland holger.schmeisky@zalando.de
[3] Freie Universität Berlin, Institut für Informatik, Takustr. 9, 14195 Berlin, Deutschland zieris@inf.fu-berlin.de

2 Results

All three agile teams were able to produce high-quality software – but the ways of working differ drastically along a dimension we call **Quality Experience**. A team with a strong Quality Experience (1) *feels fully responsible* for the general quality of their software, (2) has a *tight feedback loop* concerning this quality, and (3) *rapidly repairs deficiencies* when they occur. The two teams without separate testers have a strong Quality Experience, whereas the third one has a comparatively weak one.

There are several factors that influence the elements of the *Quality Experience* of a team, some of which relate directly to software engineering, others are sociological or psychological in nature: Having a *modularized architecture* that sufficiently decouples the work of one team from that of another is a fundamental precondition for having a strong Quality Experience. Given that precondition, management may then decide to hand over complete *control over deployment* to the development team. From here, it is only a small step to the developers both being *held responsible* and actually *feeling responsible*. *Automated tests* and *automated deployment* then facilitate *frequent deployments*. The tight feedback loop, which is (due to the lack of separate testers) characterized by *direct* (not intermediated), *quick* (available early on), and *realistic* (from a non-artificial setting) feedback, leads to a *high motivation* and ultimately *rapid repairs of defects*.

3 Conclusions

Agile teams without separate testers can achieve high quality through a strong Quality Experience. If the preconditions are fulfilled by the existing architecture, suitable domain, and willing management, separate testers would only get in the way of the developers learning what actually works from the end-users in the field.

References

[An10] Anderson, David J.: Kanban: Successful Evolutionary Change for Your Technology Business. Blue Hole Press, 2010.

[BA04] Beck, Kent; Andres, Cynthia: Extreme Programming Explained: Embrace Change, Second Edition. Addison-Wesley Professional, 2004.

[PSZ16] Prechelt, Lutz; Schmeisky, Holger; Zieris, Franz: Quality Experience: A Grounded Theory of Successful Agile Projects Without Dedicated Testers. In: Proc. 38th Int'l. Conf. on Software Engineering. ICSE '16, ACM, New York, NY, USA, pp. 1017–1027, 2016.

[SC90] Strauss, Anselm L.; Corbin, Juliet M.: Basics of Qualitative Research: Grounded Theory Procedures and Techniques. SAGE, 1990.

[SS13] Schwaber, Ken; Sutherland, Jeff: The Scrum Guide. Technical report, scrum.org, July 2013.

[Yi03] Yin, Robert K.: Case Study Research: Design and Methods. Sage, 2003.

SE Session 5 - Empirical 1

M. Tichy, E. Bodden, M. Kuhrmann, S. Wagner, J.-P. Steghöfer (Hrsg.): SE 2018,
Lecture Notes in Informatics (LNI), Gesellschaft für Informatik, Bonn 2018 83

Empirical Evaluation of a Domain-Specific Language for High-Performance Marine Ecosystem Simulation

Arne Johanson[1] Wilhelm Hasselbring[2]

Abstract: In this paper, we report on the empirical evaluation of domain-specific languages by evaluating the Sprat Ecosystem DSL for its effectiveness and efficiency.

Keywords: Domain-specific language; Model-driven software engineering; Empirical evaluation

1 Introduction

In science and research, we observe an increasing use of software. Often scientific experiments are conducted in virtual research environments with computer-based simulations, which are typically implemented by the scientists themselves. Traditionally, these research software engineers employ no or only few software engineering methods and techniques while developing large simulation software systems. Therefore, specific software support is required to efficiently enable such scientific work [JH18].

Domain-specific languages (DSL) are usually employed for model-driven software engineering in specific domains such as railway systems [Go12], software performance engineering [BH09; Vö16] and software modernization [Ho11].

Sprat is a DSL for specifying high-performance marine ecosystem simulation experiments [JH14a; JH14b; Jo16; Jo17]. We report on an online survey including controlled experiments to compare the correctness and the time spent of experts from the domain of ecosystem simulation in solving typical ecosystem simulation specification tasks with our DSL and with a GPL-based solution [JH17]. By measuring correctness scores and time spent on the assignments, we evaluate the effectiveness and efficiency of the Sprat Ecosystem DSL. The findings provide empirical evidence that the Sprat DSL can indeed increase the productivity of scientific software developers, who often have no formal software engineering training. Our results show that the domain experts achieve significantly higher accuracy and spend less time when using our DSL instead of the comparable GPL-based solution.

Our study extends the relatively scarce body of existing evaluation research on DSL to a scientific domain with users often not specifically trained in programming and software

[1] XING Marketing Solutions GmbH, Hamburg arj@informatik.uni-kiel.de
[2] Kiel University, Software Engineering Group, Kiel hasselbring@email.uni-kiel.de

engineering techniques. While the participants of many previous experiments are students, our subjects are professionals from the domain.

Literatur

[BH09] Boskovic, M.; Hasselbring, W.: Model Driven Performance Measurement and Assessment with MoDePeMART. In: Proc. MODELS 2009. Bd. 5795. Lecture Notes in Computer Science, Springer-Verlag, S. 62–76, Okt. 2009.

[Go12] Goerigk, W. et al.: Entwurf einer domänenspezifischen Sprache für elektronische Stellwerke. In: Software Engineering 2012. Bd. P-198. Lecture Notes in Informatics (LNI), GI, S. 119–130, März 2012.

[Ho11] van Hoorn, A. et al.: DynaMod Project: Dynamic Analysis for Model-Driven Software Modernization. In: Proceedings of the 1st International Workshop on Model-Driven Software Migration (MDSM 2011). Bd. 708, CEUR, 2011.

[JH14a] Johanson, A.; Hasselbring, W.: Hierarchical Combination of Internal and External Domain-Specific Languages for Scientific Computing. In: Proceedings of the 2014 European Conference on Software Architecture Workshops. ECSAW'14, ACM, 17:1–17:8, 2014.

[JH14b] Johanson, A.; Hasselbring, W.: Sprat: Hierarchies of Domain-Specific Languages for Marine Ecosystem Simulation Engineering. In: Proceedings TMS SpringSim'14. SCS, S. 187–192, 2014.

[JH17] Johanson, A.; Hasselbring, W.: Effectiveness and efficiency of a domain-specific language for high-performance marine ecosystem simulation: a controlled experiment. Empirical Software Engineering 22/4, http://rdcu.be/urXK, S. 2206–2236, Aug. 2017, URL: http://doi.org/10.1007/s10664-016-9483-z.

[JH18] Johanson, A.; Hasselbring, W.: Software Engineering for Computational Science: Past, Present, Future. IEEE Computing in Science & Engineering/, 2018.

[Jo16] Johanson, A.; Hasselbring, W.; Oschlies, A.; Worm, B.: Evaluating Hierarchical Domain-Specific Languages for Computational Science: Applying the Sprat Approach to a Marine Ecosystem Model. In: Software Engineering for Science. Taylor & Francis Group, CRC Press, S. 175–200, 2016.

[Jo17] Johanson, A. N.; Oschlies, A.; Hasselbring, W.; Worm, B.: Sprat: A spatially-explicit marine ecosystem model based on population balance equations. Ecological Modelling 349/, S. 11–25, 2017, URL: http://doi.org/10.1016/j.ecolmodel.2017.01.020.

[Vö16] Vögele, C.; van Hoorn, A.; Schulz, E.; Hasselbring, W.; Krcmar, H.: WESSBAS: extraction of probabilistic workload specifications for load testing and performance prediction—a model-driven approach for session-based application systems. Software & Systems Modeling/, S. 1–35, 2016, URL: http://doi.org/10.1007/s10270-016-0566-5.

Guidelines for Using Empirical Studies in Software Engineering Education

Fabian Fagerholm[1] , Marco Kuhrmann[2] , Jürgen Münch[3]

Abstract: Software engineering education is supposed to provide students with industry-relevant knowledge and skills. Educators must address issues beyond exercises and theories that can be directly rehearsed in small settings. A way to experience such effects and to increase the relevance of software engineering education is to apply empirical studies in teaching. In our article, we show how different types of empirical studies can be used for educational purposes in software engineering. We give examples illustrating how to utilize empirical studies, discuss challenges, and derive an initial guideline that supports teachers to include empirical studies in software engineering courses.

This summary refers to the paper *Guidelines for Using Empirical Studies in Software Engineering Education* [FKM17]. This paper was published in the PeerJ Computer Science journal.

Keywords: Software Engineering Education, Computer Science Curricula, Teaching Methods, Empirical Studies, Experimentation, Education, Guideline

1 Introduction

Providing relevant knowledge and skills is a continuous concern in software engineering education. Students must be exposed to realistic settings to understand why applying fundamental software engineering principles is necessary, why decisions should be grounded in evidence, and to learn to foresee long-term and delayed effects of certain behavior or decisions in software projects. Using empirical instruments is one approach to teach relevant software engineering knowledge and skills.

Problem Statement & Objective Since real-life software development routinely deals with large, software-intensive systems and is influenced by the manifold and complex effects of teamwork and distributed software development, software engineering education must enable students to understand such environments and to apply knowledge properly and effectively. However, restrictions in the academic curriculum and the complexity and criticality of real software products limit the level of realism that can be achieved in education. There is a

[1] University of Helsinki, Helsinki, Finland & Blekinge Institute of Technology, Karlskrona, Blekinge, Sweden, fabian.fagerholm@helsinki.fi
[2] Clausthal University of Technology, Institute for Applied Software Systems Engineering, 38640 Goslar, Germany, kuhrmann@acm.org
[3] Reutlingen University, Böblingen, Germany, Juergen.Muench@Reutlingen-University.de

lack of guidance on how to use empirical studies in software engineering education. In order to address this gap, our article [FKM17] provides an overview of different types of empirical studies, their suitability for use in education, as well as challenges with respect to their execution.

Contribution Based on our experience (e.g., [KM16, Fa17]) we show how different instruments, ranging from controlled experiments to qualitative studies, can be used for teaching purposes. We also consider overarching approaches that situate empirical studies in a larger context. We systematize purposes and challenges of the different study types, discuss the validity of the results that can be obtained in a teaching context, and create a link between teaching and research. The guidelines developed in this paper provide a systematic collection of purposes, learning goals, challenges, and validity constraints, and aims to support teachers in selecting proper study types for inclusion into their courses.

2 Results

We provide an initial assignment to the four major study types experiment, case study, simulation, and continuous experimentation. We derived a set of purposes and challenges relevant for selecting a particular study type. We identified a total of ten purposes, describing major learning goals associated with empirical studies in software engineering teaching that we consider important. Complementing the purposes, we identified eight challenges that should be taken into account when designing empirical studies for educational purposes. Taking the close relation to industry and relevance of the topics into account, we analyzed the different study types for validity constraints. Therefore, we derived four validity considerations associated with empirical studies.

3 Conclusion

The guideline presented in this paper represents a starting point based on reflection grounded in teaching practice. It can be seen as an initial systematization of empirical instruments from the educational perspective. Although the experiences from our cases show positive impacts, the learning outcomes of the application of empirical instruments should be further explored: beyond what is currently known, what do students learn by conducting empirical studies, and how do their learning outcomes differ from other approaches to software engineering education?

References

[Fa17] Fagerholm, Fabian; Guinea, Alejandro Sanchez; Mäenpää, Hanna; Münch, Jürgen: The RIGHT model for Continuous Experimentation. Journal of Systems and Software, 123:292–305, 2017.

[FKM17] Fagerholm, Fabian; Kuhrmann, Marco; Münch, Jürgen: Guidelines for Using Empirical Studies in Software Engineering Education. PeerJ Computer Science, 3:e131, 2017.

[KM16] Kuhrmann, Marco; Münch, Jürgen: When Teams Go Crazy: An Environment to Experience Group Dynamics in Software Project Management Courses. In: Proceedings of the 38th International Conference on Software Engineering Companion. ICSE '16, ACM, New York, NY, USA, pp. 412–421, 2016.

SE Session 6 - Modeling

M. Tichy, E. Bodden, M. Kuhrmann, S. Wagner, J.-P. Steghöfer (Hrsg.): SE 2018,
Lecture Notes in Informatics (LNI), Gesellschaft für Informatik, Bonn 2018 91

A framework for capturing, statistically modeling and analyzing the evolution of software models

Hamed Shariat Yazdi[1] Lefteris Angelis[2] Timo Kehrer[3] Udo Kelter[4]

Abstract: In this work, we report about a recently developed *framework for capturing, statistically modeling and analyzing the evolution of software models*, published in [Ya16]. The framework captures the changes between revisions of models in terms of both low-level (internal) and high-level (developer-visible) edit operations applied between revisions. Evolution is modeled statistically by using ARMA, GARCH and mixed ARMA-GARCH time series models. Forecasting and simulation aspects of these time series models are thoroughly assessed, and the suitability of the framework is shown by applying it to a large set of design models of real Java systems. A main motivation for, and application of, the resulting statistical models is to control the generation of realistic model histories which are intended to be used for testing model versioning tools. Further usages of the statistical models include various forecasting and simulation tasks.

Keywords: Model-Driven Engineering, Software model evolution analysis, Time series analysis Forecasting, Simulation

Summary

Model-Driven Engineering (MDE) strongly depends on comprehensive, high-quality tool support. MDE tool suites should offer not only basic services such as editing, checking and translation of models, but also support team collaboration and project management by services for model versioning and history analysis. This paper concentrates on the latter group of services, which includes services for comparing, merging, and patching models and for analyzing model histories, e.g. in order to assess the past development process or to forecast the future amount of changes. A common substantial problem in this context is the shortage or the complete lack of adequate test models for evaluating novel solutions. One way out of this problem is to generate realistic model histories, which leads back to the problem to understand how models evolve and to express the evolution, e.g. using statistical methods. However, less is known about how models of software systems evolve over time and which statistical model can describe their evolution.

This paper makes two main contributions: (1) A methodological framework for capturing, statistically modeling, forecasting and simulating the evolution of software models within a project. Such statistical models, forecasts and simulations can be used for a variety of purposes, including forecasting of the amount of changes in the next releases of the system and software cost/effort estimation methods [JS07]. (2) A test model generator which is able

[1] Computer Science Institute III, University of Bonn, Germany, shariat@cs.uni-bonn.de
[2] Department of Informatics, Aristotle University of Thessaloniki, Greece, lef@csd.auth.gr
[3] Department of Computer Science, Humboldt-University Berlin, Germany, timo.kehrer@informatik.hu-berlin.de
[4] Software Engineering Group, University of Siegen, Germany, kelter@informatik.uni-siegen.de

to produce artificial histories of model revisions which „behave“ as specified by a statistical model. These artificial model histories are intended to be used as benchmarks for various model versioning tools [PYK11; PYK12].

State-of-the-art approaches to understand the evolution of models of software systems are based on software metrics and similar static properties; the extent of the changes between revisions of a software system is expressed as differences of metrics values, and further statistical analyses are based on these differences (see, e.g. [VSN07]). Unfortunately, such approaches do not properly reflect the dynamic nature of changes. In contrast to this, our methodological framework assumes changes to be expressed structurally by means of edit operations which modify the abstract syntax graph (ASG) of a model. We support two levels of abstraction of edit operations, namely „low-level“ graph editing operations, which include creations and deletions of nodes and edges in an ASG, and „user-level“ edit operations, which are offered by modern model editors and which typically comprise many ASG changes. At both levels of abstraction, we were able to statistically model the evolution of design models of real world Java projects using two kinds of time series models: ARMA and mixed ARMA-GARCH models. We finally addressed the non-trivial problem of how to simulate the evolution in the artificially generated test models and how to generate more statistically realistic histories of test models. In this regard, we used our simulated time series sequences to configure our model generator. The simulated sequences were interpreted as sizes of model differences which were applied between subsequent revisions of models in the history. The inclined readers will find more information about other mathematical properties of models' evolution in [Ya13]. A more detailed work in presented in [Ya15].

References

[JS07] Jorgensen, M.; Shepperd, M.: A systematic review of software development cost estimation studies. IEEE Transactions on software engineering 33/1, 2007.

[PYK11] Pietsch, P.; Yazdi, H. S.; Kelter, U.: Generating realistic test models for model processing tools. In: IEEE/ACM Intl. Conf. on Automated Software Engineering. 2011.

[PYK12] Pietsch, P.; Yazdi, H. S.; Kelter, U.: Controlled Generation of Models with Defined Properties. In: Software Engineering. Pp. 95–106, 2012.

[VSN07] Vasa, R.; Schneider, J.-G.; Nierstrasz, O.: The inevitable stability of software change. In: IEEE Intl. Conf. on Software Maintenance. 2007.

[Ya13] Yazdi, H. S.; Pietsch, P.; Kehrer, T.; Kelter, U.: Statistical Analysis of Changes for Synthesizing Realistic Test Models. In: Software Engineering. 2013.

[Ya15] Yazdi, H. S.: Statistical Analysis and Simulation of Design Models Evolution, PhD thesis, University of Siegen, 2015.

[Ya16] Yazdi, H. S.; Angelis, L.; Kehrer, T.; Kelter, U.: A framework for capturing, statistically modeling and analyzing the evolution of software models. Journal of Systems and Software 118/, 2016.

M. Tichy, E. Bodden, M. Kuhrmann, S. Wagner, J.-P. Steghöfer (Hrsg.): SE 2018,
Lecture Notes in Informatics (LNI), Gesellschaft für Informatik, Bonn 2018 93

Reporting on a Survey on Approaches to Co-Evolution of Metamodels and Models

Regina Hebig[1] Djamel Eddine Khelladi[2] Reda Bendraou[3]

Abstract: Modeling languages, just as all software artifacts, evolve. This poses the risk that legacy models of a company get lost, when they become incompatible with the new language version. To address this risk, a multitude of approaches for metamodel-model co-evolution were proposed in the last 10 years. However, the high number of solutions makes it difficult for practitioners to choose an appropriate approach. In this paper, we present a survey on 31 approaches to support metamodel-model co-evolution. We introduce a taxonomy of solution techniques and classify the existing approaches. To support researchers, we discuss the state of the art, to better identify open issues. Furthermore, we use the results to provide a decision support for practitioners, who aim to adopt solutions from research.

Keywords: Survey; software engineering; metamodels; models; design notations; documentation

In the last decade, Model-Driven Engineering has proven to be effective in the development and maintenance of large scale and embedded systems [Hu11]. At the heart of this vision is the notion of Metamodel, a formal definition of the language that will be used to represent models of the application's domain. Raising the level of abstraction from code to models combined with the promise of better productivity and a shorter product development life-cycle, have led to the emergence of an impressive number of DSMLs (Domain Specific Modeling Languages). However, DSMLs do not come alone. Their productivity is mainly ensured by the ecosystem that surrounds them. This includes code generators, text and graphical editors, e.g. Xtext [Be08], and many other artifacts that constitute the development chain, spanning from modeling the problem domain to the code of the future system. While this development chain is effective and efficient, its Achilles heel remains the metamodel. One can see it as the central point of a dependency graph of artifacts and tools since any modification in the metamodel may impact all the other artifacts i.e. editors, code generators, transformation rules, consistency constraints, etc. [DRIP12].

Metamodels co-evolution has already been identified as a main issue in the literature [HBJ09], [GKP07], [CDRP09]. The evolution between two versions of the same metamodel may consist of hundreds of additions, deletions, and changes of elements. For example, during the evolution of UML Class Diagrams from UML version 1.5 to 2.0 a total of 238 language elements where added, deleted, or changed [Kh15]. For the evolution from GMF

[1] Chalmers | University of Gothenburg, Computer Science and Engineering Göteborg, Västra Götaland County, Sweden. hebig@chalmers.se
[2] Johannes Kepler University Linz, Austria djamel_eddine.khelladi@jku.at
[3] Sorbonne Universités, UPMC Univ Paris 06, UMR 7606, LIP6, F-75005, Paris, France reda.bendrao@lip6.fr

version 1.0 to 2.0, 136 changes have been identified by Herrmannsdoerfer et al. [HBJ09]. In the last decade, the literature introduced 31 approaches that focus exclusively on the co-evolution of models when the metamodel is evolved.

We provide an overview about results of a recent survey paper [HKB17] summarizing how these 31 approaches differ in the way they solve the involved tasks which are a) the identification of metamodel changes and b) the co-evolution of model instances. From the user's perspective these approaches may differ significantly. The trade-off between correctness and automation could be one way to compare them. Fully automated approaches make it easier to handle huge sets of models while it might not be possible to guarantee that the resulting models conform to the required outcome. The work that aims at helping companies and users in choosing and comparing this jungle of proposed approaches for metamodel-model co-evolution (MM-M co-evolution) according to their needs and constraints. It does so by providing a survey and a detailed catalog of existing co-evolution approaches and their technical properties. Furthermore, we want to provide researchers with a framework for comparing their works and enable them to identify open research needs.

References

[Be08] Behrens, Heiko; Clay, Michael; Efftinge, Sven; Eysholdt, Moritz; Friese, Peter; Köhnlein, Jan; Wannheden, Knut; Zarnekow, Sebastian: Xtext user guide. `https://eclipse.org/Xtext/documentation/1_0_1/xtext.pdf`, 2008.

[CDRP09] Cicchetti, Antonio; Di Ruscio, Davide; Pierantonio, Alfonso: Managing dependent changes in coupled evolution. In: Theory and Practice of Model Transformations, pp. 35–51. Springer, 2009.

[DRIP12] Di Ruscio, Davide; Iovino, Ludovico; Pierantonio, Alfonso: Evolutionary togetherness: how to manage coupled evolution in metamodeling ecosystems. In: Graph Transformations, pp. 20–37. Springer, 2012.

[GKP07] Gruschko, Boris; Kolovos, Dimitrios; Paige, Richard: Towards synchronizing models with evolving metamodels. In: Proceedings of the International Workshop on Model-Driven Software Evolution. 2007.

[HBJ09] Herrmannsdoerfer, Markus; Benz, Sebastian; Juergens, Elmar: COPE-automating coupled evolution of metamodels and models. In: ECOOP 2009–Object-Oriented Programming, pp. 52–76. Springer, 2009.

[HKB17] Hebig, Regina; Khelladi, Djamel Eddine; Bendraou, Reda: Approaches to co-evolution of metamodels and models: A survey. IEEE Transactions on Software Engineering, 43(5):396–414, 2017.

[Hu11] Hutchinson, John; Whittle, Jon; Rouncefield, Mark; Kristoffersen, Steinar: Empirical assessment of MDE in industry. In: Proceedings of the 33rd International Conference on Software Engineering. ACM, pp. 471–480, 2011.

[Kh15] Khelladi, Djamel Eddine; Hebig, Regina; Bendraou, Reda; Robin, Jacques; Gervais, Marie-Pierre: Detecting complex changes during metamodel evolution. In: Advanced Information Systems Engineering. Springer, pp. 263–278, 2015.

SE Session 7 - Empirical 2

M. Tichy, E. Bodden, M. Kuhrmann, S. Wagner, J.-P. Steghöfer (Hrsg.): SE 2018,
Lecture Notes in Informatics (LNI), Gesellschaft für Informatik, Bonn 2018 97

Component and Connector Views in Practice: An Experience Report (extended abstract)

Vincent Bertram[1] Shahar Maoz[2] Jan Oliver Ringert[3] Bernhard Rumpe[4] Michael von Wenckstern[5]

Abstract:

C&C views are a means for formal yet intuitive structural specification of C&C models. In [Be17] we report on our experience how C&C views and their verification help to address challenges of traceability and evolution in automotive industry. We analyzed the development process at Daimler AG and evaluated our C&C views verification tool on five Simulink models with more than 7700 subsystems in total and C&C views created for 183 textual requirements provided by Daimler AG. We describe our experience in detail and discuss a list of lessons learned, including, e.g., a missing abstraction concept in C&C models and C&C views that we have identified and added to the views language and tool, that engineers can create graphical C&C views quite easily, and how verification algorithms scale on real-size industry models. Furthermore, we report on the non-negligible technical effort needed to translate Simulink block diagrams to C&C models. We make all materials mentioned and used in our experience electronically available for inspection and further research.

Keywords: component and connector models, Simulink, architecture, industrial case study

C&C models, described using languages such as SysML, AADL, and related block diagram languages, are used extensively in software and systems engineering. Simulink/Stateflow [Ma16] are prevalent tools used in the automotive industry for model-based prototype implementation, simulation, and testing.

In recent work [MRR13; MRR14] we presented C&C views, as a means to formally and intuitively specify constraints on the structure of C&C models. The views allow engineers to specify constraints on hierarchy and connectivity, using partial examples, while crosscutting the implementation-oriented system/subsystem hierarchy of the target model.

In [Be17] we report on our experience in applying C&C views in practice, in an industrial, automotive setting, guided by the following questions:

[1] Daimler AG Group Research & MBC Development, Ulm, Germany, bertram@se-rwth.de
[2] School of Computer Science, Tel Aviv University, Israel, maoz@cs.tau.ac.il
[3] School of Computer Science, Tel Aviv University, Israel, ringert@post.tau.ac.il
[4] RWTH Aachen University, Aachen, Germany, rumpe@se-rwth.de
[5] RWTH Aachen University, Aachen, Germany, vonwenckstern@se-rwth.de

Q1 Which industrial contexts in automotive domain are relevant for C&C views and what challenges can the use of C&C views address?

Q2 Can domain experts create C&C views with reasonable effort and are they missing any language features?

Q3 Is C&C views verification applicable to automotive industry models and does it scale to deal with their size?

Q4 Are the verification outputs of use for the engineers?

Since the answer to **Q1** influences the experiment setup for the other questions, we decided to do a two-stage study. In the preliminary study, we interviewed industrial partners to investigate the automotive development processes and challenges of developers. Based on the findings of the preliminary study, we chose Daimler AG as an automotive partner and collected relevant documents and models for evaluation. We then executed the main study, to address questions **Q2** to **Q4**. We chose the automotive domain as representative for safety-critical, distributed control systems.

In the main case study, two domain experts created 50 C&C views based on 183 industrial textual requirements and design decisions of two automotive software systems: Advanced Driver Assistance Systems (ADAS), available in four different evolution versions, and Adaptive Light System (ALS). We devised a translation from Simulink block diagrams to C&C models to check the created C&C views using our existing verification tool [MRR14]. The translation from Simulink to C&C models involved non-negligible technical efforts. Finally, we presented the tool's generated witnesses, which demonstrate reasons for satisfaction or non-satisfaction, to the industrial partner who evaluated their usefulness with regard to two identified industrial challenges: traceability and evolution.

As part of our results, the industrial partner identified a missing abstraction concept in C&C views that we implemented. We found that given textual requirements, domain experts can create C&C views that highlight the implementation details of requirements in a Simulink model of hundreds of blocks with reasonable effort. We found that C&C views verification scales well for sizes of industrial models and average running times were below two seconds in all our experiments. Finally, C&C views helped the domain experts to discover several inconsistencies between requirements and their implementation.

We consider it an important contribution of our work that we have made **all artifacts we used and created available from** [www]. These materials include the four ADAS and the one ALS Simulink models (web export) by Daimler AG, their original requirements in German with an English translation, C&C views in textual and graphical representation, and all verification results. We encourage the reader to inspect these materials and use them for their own research.

References

[Be17] Bertram, V.; Maoz, S.; Ringert, J. O.; Rumpe, B.; von Wenckstern, M.: Component and Connector Views in Practice: An Experience Report. In: MODELS 2017. IEEE Computer Society, pp. 167–177, 2017, URL: https://doi.org/10.1109/MODELS.2017.29.

[Ma16] Mathworks: Simulink User's Guide, tech. rep. R2016b, MATLAB & SIMULINK, 2016, p. 4022.

[MRR13] Maoz, S.; Ringert, J. O.; Rumpe, B.: Synthesis of component and connector models from crosscutting structural views. In: ESEC/FSE'13. ACM, pp. 444–454, 2013, URL: http://doi.acm.org/10.1145/2491411.2491414.

[MRR14] Maoz, S.; Ringert, J. O.; Rumpe, B.: Verifying component and connector models against crosscutting structural views. In: ICSE '14. ACM, pp. 95–105, 2014, URL: http://doi.acm.org/10.1145/2568225.2568237.

[www] Supporting materials for our case study, URL: http://www.se-rwth.de/materials/cncviewscasestudy/.

Wo ist der Fehler und wie wird er behoben? Ein Experiment mit Softwareentwicklern.

Marcel Böhme[1], Ezekiel O. Soremekun[2], Sudipta Chattopadhyay[3], Emamurho Ugherughe[4], Andreas Zeller[5]

1 Zusammenfassung

Dieser Beitrag ist eine gekürzte, deutsche Version unseres Artikels "Where is the Bug and How is it Fixed? An Experiment with Practitioners" publiziert in dem Berichtsband des elften gemeinsamen Treffens der European Software Engineering Conference und des ACM SIGSOFT Symposium on the Foundations of Software Engineering [B7].

Seit mehr als zwanzig Jahren hat die Forschung eine Reihe von Ansätzen zur automatischen Lokalisierung, Erklärung und Behebung von Programmfehlern entwickelt. In letzter Zeit haben Forscher auch einige Benchmarks bereitgestellt, die zur empirischen Auswertung solcher Ansätze genutzt werden können. Mit Hilfe solcher Benchmarks können Wissenschaftler empirisch fundierte Behauptungen über die Effektivität solcher Werkzeuge und Techniken aufstellen. Zum Beispiel wird oft ein Fehlerlokalisierungswerkzeug als effektiver bewertet, welches einer Anweisung, die während der Fehlerbehebung geändert wurde, einen höheren Rang zuweist. Dabei wird angenommen, daß ein Entwickler dieselbe Anweisung als Fehlerursache identifizieren würde. Eine automatische Fehlerbehebung wird als effektiver bewertet, wenn daraufhin alle Regressionstests erfolgreich durchlaufen. Hier wird angenommen, daß ein Entwickler eine solche fehlerbehebende Programmänderung akzeptieren oder gar selbst durchführen würde. Leider ist der Debuggingprozess in der Realität nicht ganz so einfach, besonders nicht für den Menschen. Wir stellen einen anderen Benchmark zur Verfügung; einen Benchmark der eine Realitätskontrolle erlaubt.

In Anbetracht der Komplexität des Debuggingprozesses sollte man doch annehmen, daß neue Werkzeuge standardmäßig mithilfe von Nutzerstudien ausgewertet werden: Passt das Werkzeug in den Prozess? Ist es wertschaffend? Inwiefern? Dennoch ist nicht wirklich gut erforscht, wie Menschen eigentlich Fehler lokalisieren, erklären und beheben. Zwischen 1981 und 2010 haben Parnin und Orso gerade mal eine Handvoll von Artikeln identifiziert, die Ergebnisse von Nutzerstudien präsentiert haben. Keiner dieser Artikel involvierte sowohl richtige Entwickler als auch echte Programmfehler. Seit Ende 2010 haben auch wir gerade einmal drei Artikel gefunden, die beides involvierten.

[1] Monash University, Faculty of Information Technology, Melbourne, Australia marcel.boehme@acm.org
[2] Saarland University & CISPA, Saarbrücken, Germany soremekun@st.cs.uni-saarland.de
[3] Singapore University of Technology and Design, Singapore, Singapore sudipta_chattopadhyay@sutd.edu.sg
[4] SAP, Berlin, Germany emamurho@gmail.com
[5] Saarland University & CISPA, Saarbrücken, Germany zeller@st.cs.uni-saarland.de

In diesem Artikel versuchen wir nicht ein bestimmtes Werkszeug zu evaluieren. Stattdessen beleuchten wir den gesamten Debuggingprozess, wie er von Softwareentwicklern real durchgeführt wird. Wir untersuchen im Einzelnen wie lange es typischerweise dauert, wie schwierig die Fehlersuche und -behebung wahrgenommen wird und welche Strategien bei der Fehlersuche und -behebung angewendet werden. Für unseren Benchmark entnehmen wir die fehlerverursachenden Anweisungen, Erklärungen des Fehlerhergangs und die eigentlichen Fehlerbehebungen (d.h. Patches) welche *Entwickler* hervorbrachten. Wir nutzten 27 reale Fehler von CoREBench [BR14] die systematisch aus den 10.000 jüngsten Commits and den dazugehörigen Fehlerberichten extrahiert wurden. Wir beauftragten 12 Softwareingenieure aus 6 Ländern für jeden dieser Programmfehler i) die fehlerverursachenden Anweisungen zu lokalisieren, ii) den Fehlerhergang zu beschreiben und iii) den Fehler zu beheben.

Erkenntnisse. Nach unserem besten Wissen haben wir den ersten Beleg dafür gefunden, daß der Debuggingprozess überhaupt automatisiert werden kann. In unserem Experiment haben unterschiedliche Entwickler grundsätzlich dieselben Fehlerursachen lokalisiert und die gleiche Erklärung zu dem Fehlerhergang abgegeben. Wenn sich sogar unsere menschlichen Teilnehmer widersprächen, wie könnte eine Maschine dann die "richtige" Fehlerursache und -diagnose etablieren? Wir fanden auch heraus, daß viele der eingereichten Fehlerbehebungen (d.h. Patches) eigentlich inkorrekt waren: Obwohl 97% aller Patches plausibel waren, das heißt alle Regressionstests liefen erfolgreich durch, waren lediglich 63% wirklich korrekt, das heißt sie bestanden auch unseren Codereview. Das motiviert natürlich weitführende Forschung an der automatischen Fehlerbehebung. Drei von vier inkorrekten Patches sind inkorrekt, weil sie neue Fehler einführen oder den eigentlichen Fehler nicht vollständig beheben. Das motiviert Untersuchungen der automatischen Regressionstestgenerierung.

Benchmark. Da unsere Teilnehmer grundsätzlich bei den essentiellen Fehlermerkmalen übereinstimmen, kann man deren Artefakte als Grundwahrheit behandeln. Aus diesem Grund stellen wir die gewonnen Studiendaten als Benchmark mit dem Namen DbgBench bereit. DbgBench kann in kosteneffektiven Nutzerstudien genutzt werden um zu untersuchen wie sich die Zeit, Schwierigkeit und Patchkorrektheit mit dem neuen Werkzeug verbessert. DbgBench kann auch genutzt werden um ohne einer Nutzerstudie auszuwerten wie gut sich das neuartige Werkzeug gegen unsere menschlichen Teilnehmer misst wenn es um Aufgaben wie das Lokalisieren, Erklären und Beheben von Programmfehlern geht. Mehr Informationen zu DbgBench finden Sie unter folgender Adresse: `https://dbgbench.github.io`.

Literaturverzeichnis

[B7] Böhme, Marcel; Soremekun, Ezekiel O.; Chattopadhyay, Sudipta; Ugherughe, Emamurho; Zeller, Andreas: Where is the Bug and How is it Fixed? An Experiment with Practitioners. In: Proceedings of the 11th Joint meeting of the European Software Engineering Conference and the ACM SIGSOFT Symposium on the Foundations of Software Engineering. ESEC/FSE, S. 117–128, 2017.

[BR14] Böhme, Marcel; Roychoudhury, Abhik: CoREBench: Studying Complexity of Regression Errors. In: Proceedings of the 23rd ACM/SIGSOFT International Symposium on Software Testing and Analysis. ISSTA, S. 105–115, 2014.

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

Marco Kuhrmann[1] , Philipp Diebold[2] , Jürgen Münch[3] , Paolo Tell[4] , Vahid Garousi[5] , Michael Felderer[6] , Kitija Trektere[7] , Fergal McCaffery[7], Oliver Linssen[8] , Eckhart Hanser[9] , Christian R. Prause[10]

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 methods and flexible agile practices. Yet, the number of methods and practices is large and the actual process composition is often carried out in an ad-hoc manner. This 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.

This summary refers to the paper *Hybrid Software and System Development in Practice: Waterfall, Scrum, and Beyond* [Ku17]. This paper was published as full research paper in the proceedings of the International Conference on Software System Process.

Keywords: Agile Software Development, Software Process, Hybrid Development Approaches

1 Introduction

Software development is diverse, and companies have to adopt to new technologies and markets quickly. 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

Problem Statement & Objective In 2011, West et al. [We11] coined the term "Water-Scrum-Fall" and hypothesized that hybrid development methods will become the standard.

[1] Clausthal University of Technology, Germany, kuhrmann@acm.org
[2] Fraunhofer IESE, Germany, Philipp.Diebold@iese.fraunhofer.de
[3] Reutlingen University, Böblingen, Germany, Juergen.Muench@Reutlingen-University.de
[4] IT University Copenhagen, Denmark, pate@itu.dk
[5] Wageningen University, The Netherlands, vahid.garousi@wur.nl
[6] University of Innsbruck, Austria, Michael.Felderer@uibk.ac.at
[7] Dundalk Institute of Technology, Ireland, {kitija.trektere, fergal.mccaffery}@dkit.ie
[8] FOM University of Applied Sciences, Germany, oliver.linssen@fom.de
[9] DHBW Lörrach, Germany, hanser@dhbw-loerrach.de
[10] German Aerospace Center, Germany, Christian.Prause@dlr.de

A systematic review by Theocharis et al. [Th15] revealed a gap in literature: while research on agile software development is rich, traditional processes are widely ignored in recent research. 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.

Contribution The paper at hand presents results from the HELENA study. 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.

2 Results

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. A 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.

3 Conclusion & Future Work

Our paper [Ku17] documents findings from the first stage of the HELENA study. The data collection for HELENA's stage 2 has been finished. Based on the new data collected, findings from [Ku17] will be confirmed and new research questions will be answered.

References

[Ku17] Kuhrmann, Marco; Diebold, Philipp; Münch, Jürgen; Tell, Paolo; Garousi, Vahid; Felderer, Michael; Trektere, Kitija; McCaffery, Fergal; Prause, Christian R.; Hanser, Eckhart; Linssen, Oliver: Hybrid Software and System Development in Practice: Waterfall, Scrum, and Beyond. In: Proceedings of the International Conference on Software System Process. ICSSP, ACM, New York, NY, USA, pp. 30–39, July 2017.

[Th15] Theocharis, Georgios; Kuhrmann, Marco; Münch, Jürgen; Diebold, Philipp: Is Water-Scrum-Fall Reality? On the Use of Agile and Traditional Development Practices. In: International Conference on Product Focused Software Development and Process Improvement. volume 9459 of Lecture Notes in Computer Science, Springer, Cham, pp. 149–166, Dec 2015.

[We11] West, Dave; Gilpin, Mike; Grant, Tom; Anderson, Alissa: Water- Scrum-Fall Is The Reality Of Agile For Most Organizations Today. Technical report, Forrester Research Inc., July 2011.

SE Session 8 - Model evolution and transformation

M. Tichy, E. Bodden, M. Kuhrmann, S. Wagner, J.-P. Steghöfer (Hrsg.): SE 2018,
Lecture Notes in Informatics (LNI), Gesellschaft für Informatik, Bonn 2018 109

A systematic approach to constructing families of incremental topology control algorithms using graph transformation

Roland Kluge[1], Michael Stein[2], Gergely Varró[1], Andy Schürr[1], Matthias Hollick[3], Max Mühlhäuser[2]

Abstract: In this talk, we present results on integrating support for variability modeling into a correct-by-construction development methodology for topology control algorithms, as appeared online in the Software & Systems Modeling journal in 2017 [Kl17]. A topology control algorithm reduces the size of the visible neighborhood of a node in a wireless communication network. At the same time, it must fulfill important consistency properties to ensure a high quality of service. In previous work, we proposed a constructive, model-driven methodology for designing individual topology control algorithms based on declarative graph constraints and graph transformation rules; the resulting algorithms are guaranteed to preserve the specified properties. Even though many topology control algorithms share substantial (structural) parts, few works leverage these commonalities at design time. In this work, we generalize our proposed construction methodology by modeling variability points to support the construction of families of algorithms. We show the applicability of our approach by reengineering six existing topology control algorithms and developing e-kTC, a novel energy-efficient variant of the topology control algorithm kTC. Finally, we evaluate a subset of the algorithms using a novel integration of a wireless network simulator and a graph transformation tool.

Keywords: Graph transformation; Graph constraints; Static analysis; Model-driven engineering; Wireless networks; Network simulation

1 Summary

In the communication systems domain, wireless sensor networks (WSNs) [Sa05] are a highly active research area. WSNs serve, e. g., to monitor environmental conditions using small, battery-powered sensor nodes that cooperatively transmit data to a sink node. To improve important properties (e. g., the energy consumption), a topology control (TC) algorithm inactivates redundant communication links of a WSN [Sa05]. Constructing a TC

[1] TU Darmstadt, Real-Time Systems Lab, Merckstr. 25, 64283 Darmstadt, Germany, {roland.kluge,andy.schuerr}@es.tu-darmstadt.de, gervarro@gmail.com

[2] TU Darmstadt, Telecooperation Group, Hochschulstr, 10, 64289 Darmstadt, Germany, {stein,max}@tk.tu-darmstadt.de

[3] TU Darmstadt, Secure Mobile Networking Lab, TU Darmstadt, Mornewegstr. 32, 64293 Darmstadt, Germany, matthias.hollick@seemoo.tu-darmstadt.de

Acknowledgments: This work was supported by the DFG (German Research Foundation) as part of projects A1 and C1 within CRC 1053 – MAKI.

algorithm is a challenging task, which is carried out by highly skilled experts usually in two major steps. First, the TC algorithm is specified using a mathematically framework (e. g., graph theory), which allows us to prove the required formal properties. Then, the TC algorithm is implemented, typically in a general-purpose programming language (e. g., Java or C) for simulation or real-life evaluation purposes.

The traditional development process of TC algorithms suffers from (at least) two major shortcomings [Kl17, Sec. 1]: The first shortcoming is that a systematic mapping between the specification and the implementation is missing. This makes it difficult to verify that specification and implementation correspond to each other. To tackle this shortcoming, we proposed a model-driven development methodology that constructively integrates the required formal properties [Kl16]. We describe topologies as graph-based models, required properties as graph constraints, and possible operations of topology control algorithms as declarative graph transformation rules [HW95].

The second shortcoming is that—even though novel TC algorithms tend to build on former TC algorithms—the inherent commonalities and differences of TC algorithms are not specified systematically. This reduces reusability among and comparability of TC algorithms, especially w. r. t. the proven formal properties. In the presented article [Kl17], we generalize the constructive, model-driven methodology for designing TC algorithms using graph transformation [Kl16] to support the development of families of TC algorithms. More precisely, (i) to model commonalities and differences of TC algorithms, we extract common structural constraints and specify individual part of each TC algorithm as attribute constraints, thereby lifting all steps described in [Kl16] to algorithm families; (ii) to demonstrate that our approach is applicable, we specify six existing TC algorithms and e-kTC, a novel energy-aware TC algorithm; (iii) to support dynamic TC, we extend the constructive approach with a step that systematically derives context event handlers, which anticipate imminent constraint violations; (iv) we present a rapid evaluation environment, consisting of the graph transformation tool eMoflon and the network simulator Simonstrator.

References

[HW95] Heckel, R.; Wagner, A.: Ensuring Consistency of Conditional Graph Rewriting – A Constructive Approach. In: Proc. of the Joint COMPUGRAPH/SEMAGRAPH Workshop. volume 2 of ENTCS. Elsevier, pp. 118–126, 1995.

[Kl16] Kluge, R.; Stein, M.; Varró, G.; Schürr, A.; Hollick, M.; Mühlhäuser, M.: A systematic approach to constructing incremental topology control algorithms using graph transformation. Journal of Visual Languages & Computing (JVLC), 38:47–83, 2016.

[Kl17] Kluge, R.; Stein, M.; Varró, G.; Schürr, A.; Hollick, M.; Mühlhäuser, M.: A Systematic Approach to Constructing Families of Incremental Topology Control Algorithms Using Graph Transformation. Software & Systems Modeling (SoSyM), pp. 1–41, 2017.

[Sa05] Santi, P.: Topology Control in Wireless Ad Hoc and Sensor Networks. ACM Computing Surveys (CSUR), 37(2):164–194, 2005.

M. Tichy, E. Bodden, M. Kuhrmann, S. Wagner, J.-P. Steghöfer (Hrsg.): SE 2018,
Lecture Notes in Informatics (LNI), Gesellschaft für Informatik, Bonn 2018 111

Clone Detection for Rule-Based Model Transformation Languages

Daniel Strüber[1], Vlad Acreţoaie[2], Jennifer Plöger[3]

Keywords: model transformation; clone detection; quality assurance; Henshin; ATL

1 Abstract

We present our paper published in the Journal of Software and Systems Modeling [SAP17].

Model transformation is an enabling technology of Model-Driven Engineering, a software engineering paradigm in which models are routinely refined, translated, and optimized. To support the developers of a model transformation with advanced mechanisms for analysis, verification, and traceability, a large variety of dedicated model transformation languages has evolved [CH06]. Since many of these languages were originally developed in an academic setting, there is a lack of mature mechanisms for the reuse of existing transformations. As a result, the most frequently applied reuse mechanism remains cloning, i.e., copying and modifying existing transformations. Unfortunately, cloning presents severe maintainability challenges, which are well-known from the substantial body of research on *software clones* [Ko07]. Despite the availability of numerous clone detection techniques for programming and modeling languages, no specific ones have emerged for model transformation languages.

We present the first investigation of clone detection for model transformation languages. Our paper focuses on two main paradigms of model transformations that are related by a common denominator, the notion of rules: *graph-based* languages, in which transformations are expressed using graph rewriting rules, and *hybrid* languages, which combine declarative rule notions with imperative concepts. Specifically, we make the following contributions:

- A discussion of use-cases for clone detection for model transformation languages. Aiming to shed light on the indeed extensive scope of possible use-cases, we consider the refactoring of existing transformations towards suitable reuse mechanisms, quality

[1] Universität Koblenz-Landau, Universitätsstr. 1, 56070 Koblenz, Germany, strueber@uni-koblenz.de
[2] Frequentis Romania, Tetarom 1 Technology Park, Taietura Turcului 47, 400221 Cluj-Napoca, Romania, radu-vlad.acretoaie@frequentis.com
[3] Philipps-Universität Marburg, Hans-Meerwein-Str. 1, 35032 Marburg, Germany, jennifer.ploeger@gmx.de

assessment, performance optimizations, and the identification of transformation design patterns. Based on the use-cases, we elicit five key requirements for clone detection for graph-based and hybrid model transformations.

- A customization strategy of existing model clone detection techniques, allowing us to address the identified requirements. We customized eScan [Ng09], an a-priori-based technique, and ConQAT [De08], a heuristic one, to the graph-based language Henshin [Ar10, St17] and the hybrid language ATL [JK05].
- An extensive experimental evaluation. For the graph-based case, we considered three rule sets; the largest one included 1,404 rules. For the hybrid case, we considered the ATL zoon, including 2,566 rules in total. A key finding is that ConQAT's accuracy was nearly optimal when using the results of eScan as a ground truth.

In the established taxonomy of software clones [Ko07], our investigation focuses on *Type I* and *II* clones, i.e. identical fragments and almost-identical ones (except for naming) . Type I and II clones are typically created when model transformations are developed in a copy-and-paste manner. With our findings, we provide a first insight into the application of clone detection to model transformations languages. In the future, we aim to extend our work to Type III and IV clones and additional model transformation paradigms.

References

[Ar10] Arendt, Thorsten; Biermann, Enrico; Jurack, Stefan; Krause, Christian; Taentzer, Gabriele: Henshin: advanced concepts and tools for in-place EMF model transformations. In: MODELS, pp. 121–135. Springer, 2010.

[CH06] Czarnecki, Krzysztof; Helsen, Simon: Feature-based survey of model transformation approaches. IBM Systems Journal, 45(3):621–645, 2006.

[De08] Deissenboeck, Florian; Hummel, Benjamin; Jürgens, Elmar; Schätz, Bernhard; Wagner, Stefan; Girard, Jean-Francois; Teuchert, Stefan: Clone Detection in Automotive Model-Based Development. In: ICSE. ACM, pp. 603–612, 2008.

[JK05] Jouault, Frédéric; Kurtev, Ivan: Transforming Models with ATL. In: MODELS Workshops. Springer, pp. 128–138, 2005.

[Ko07] Koschke, Rainer: Survey of research on software clones. In: Dagstuhl Seminar 06301: Duplication, Redundancy, and Similarity in Software, p. 24. GI, 2007.

[Ng09] Nguyen, Hoan; Nguyen, Tung; Pham, Nam; Al-Kofahi, Jafar; Nguyen, Tien: Accurate and Efficient Structural Characteristic Feature Extraction for Clone Detection. In: FASE. Springer, pp. 440–455, 2009.

[SAP17] Strüber, Daniel; Acreţoaie, Vlad; Plöger, Jennifer: Model clone detection for rule-based model transformation languages. Software & Systems Modeling, Oct 2017.

[St17] Strüber, Daniel; Born, Kristopher; Gill, Kanwal Daud; Groner, Raffaela; Kehrer, Timo; Ohrndorf, Manuel; Tichy, Matthias: Henshin: A Usability-Focused Framework for EMF Model Transformation Development. In: ICGT. Springer, pp. 196–208, 2017.

M. Tichy, E. Bodden, M. Kuhrmann, S. Wagner, J.-P. Steghöfer (Hrsg.): SE 2018,
Lecture Notes in Informatics (LNI), Gesellschaft für Informatik, Bonn 2018 113

Modeling and Verification of Evolving Cyber-Physical Spaces

Christos Tsigkanos[1], Timo Kehrer[2], Carlo Ghezzi[3]

Abstract:

In this work, we report about recent research results on the Modeling and Verification of Evolving Cyber-Physical Spaces, published in [TKG17]. We increasingly live in cyber-physical spaces – spaces that are both physical and digital, and where the two aspects are intertwined. Such spaces are highly dynamic and typically undergo continuous change. Software engineering can have a profound impact in this domain, by defining suitable modeling and specification notations as well as supporting design-time formal verification. In this paper, we present a methodology and a technical framework which support modeling of evolving cyber-physical spaces and reasoning about their spatio-temporal properties. We utilize a discrete, graph-based formalism for modeling cyber-physical spaces as well as primitives of change, giving rise to a reactive system consisting of rewriting rules with both local and global application conditions. Formal reasoning facilities are implemented adopting logic-based specification of properties and according model checking procedures, in both spatial and temporal fragments. We evaluate our approach using a case study of a disaster scenario in a smart city.

Keywords: Cyber-Physical Spaces, Dependable Software-Intensive Systems, Safety and Reliability, Modelling and Specification, Formal Verification

Summary

Computing and communication capabilities are increasingly embedded into physical spaces thus blurring the boundary between computational and physical worlds; typically, this is the case in modern cyber-physical systems, like smart buildings or smart cities, hereafter called space-dependent systems. Conceptually, we consider such a composite environment as a cyber-physical space (CPSp), which consists of interrelated computational and physical entities. Like any other software-intensive system, a CPSp is not a static construct. Dynamic actions (e.g. performed by agents) generate continuous change, leading to the notion of an *evolving cyber-physical space*. Thus an evolving CPSp must face the manifold challenges of dynamism – change may affect requirements of the overall space-dependent system.

Formally modeling space and its change as well as reasoning about various properties of evolving space are crucial prerequisites for engineering dependable evolving CPSp. Our approach targets the critical system requirements phase, where a way to obtain formal assurances is highly sought. Elementary properties of an evolving spatial environment can be roughly classified into three kinds; (i) *spatial (local)*, referring to entities forming some structural pattern, (ii) *spatial (global)*, where entities are arbitrarily distributed in

[1] Dipartimento di Elettronica, Informazione e Bioingegneria, Politecnico di Milano. christos.tsigkanos@polimi.it
[2] Institut für Informatik, Humboldt-Universität zu Berlin. timo.kehrer@informatik.hu-berlin.de
[3] Dipartimento di Elettronica, Informazione e Bioingegneria, Politecnico di Milano. carlo.ghezzi@polimi.it

space, as well as (iii) *temporal*, expressing system behavior. A plethora of approaches are actively investigated by the community to support reasoning about properties of *one* of these kinds; graphs and graph pattern matching provide suitable methods to deal with local spatial properties, while model checking based on various forms of spatial and temporal logics provides a rigorous approach for the verification of global spatial and temporal properties. However, there is a lack of consideration of *all* of the above listed kinds of properties at the same time. This is a significant deficiency concerning engineering of dependable space-and-time-dependent systems, since properties of interest are often *complex spatio-temporal* properties. Informally, a complex spatio-temporal property refers to behavioral characteristics (temporal) of spatial relationships (spatial; global) of complex structures (spatial; local).

Software engineering (SE) can have a profound impact in engineering of space-and-time-dependent systems, by defining suitable modeling and specification notations as well as supporting design-time formal verification. The typical SE approach –provide a suitable model amenable for analysis and use it to validate a design– is applied to the domain of CPSp. The main contribution of this paper is a technical framework for integrating several fundamental techniques to support reasoning about complex spatio-temporal properties of a model of evolving space. Our modeling approach grounds on Bigraphs [Mi09], a fundamental theory for structures in ubiquitous computing. Local reconfigurations are expressed as rewriting rules called reaction rules, yielding a Bigraphical Reactive System (BRS). Reasoning facilities are implemented adopting logic-based specification of properties and according model checking procedures. Locally bounded spatial properties are expressed as bigraphical patterns, and bigraphical matching is used as a fundamental technique to locate points in space where such formulae hold. For checking of non-local spatial properties, we interpret a bigraphical model as a closure space [Ga03], paving the way for adopting an according spatial logic [Ci14]. Concerning checking of temporal properties, state transition models are obtained from a BRS. We restrict the combination of the above components such that reasoning complexity is manageable and expressiveness is not compromised. We demonstrate applicability using a disaster scenario in a smart city environment.

References

[Ci14] Ciancia, V.; Latella, D.; Loreti, M.; Massink, M.: Specifying and verifying properties of space. In: Theoretical Computer Science. Springer, pp. 222–235, 2014.

[Ga03] Galton, A.: A generalized topological view of motion in discrete space. Theoretical Computer Science 305/1, pp. 111–134, 2003.

[Mi09] Milner, R.: The Space and Motion of Communicating Agents. Cambridge University Press, 2009.

[TKG17] Tsigkanos, C.; Kehrer, T.; Ghezzi, C.: Modeling and verification of evolving cyber-physical spaces. In: Proc. of the 11th Joint Meeting on Foundations of Software Engineering, ESEC/FSE 2017. ACM, pp. 38–48, 2017.

SE Session 9 - Program analysis and failure prediction

M. Tichy, E. Bodden, M. Kuhrmann, S. Wagner, J.-P. Steghöfer (Hrsg.): SE 2018,
Lecture Notes in Informatics (LNI), Gesellschaft für Informatik, Bonn 2018 117

CodeMatch: Obfuscation Won't Conceal Your Repackaged App

Leonid Glanz, Sven Amann, Michael Eichberg, Michael Reif, and Mira Mezini[1]

Abstract: Popular mobile apps are regularly installed by millions of users. This fact attracts malicious actors to create altered, repackaged versions of those apps to steal the original owner's revenue or to trick users to infect their devices with malware. Detecting such repackaged apps is, therefore, necessary for a secure and viable app market but is challenging due to the use of code obfuscation and the widespread usage of libraries. Due to the recent fact, non-repackaged, legitimate apps often share a majority of their code base and are classified as repackaged by state-of-the-art detectors. We, therefore, propose a new library filtering approach that relies on code representations at five different abstraction levels to achieve resilience against code obfuscation. Additionally, we propose to use the most abstract representation in combination with fuzzy-hashing to detect repackaged apps. Our evaluation shows that the overall approach leads to a better detection rate up to 50%.

Keywords: library detection; repackage detection; obfuscation; code analysis

1 Overview

Since 2012 several techniques for repackage detection have been proposed and can be broadly classified as being code-agnostic, graph-based, user-interface-based, and code-signature-based. A challenge for all these repackage detectors is code transformation. Developers regularly minify and optimize their apps to increase performance. Additionally, they obfuscate their apps to protect their intellectual property. However, attackers also apply obfuscation to hide malicious code and to evade signature-based detectors, such as anti-virus software.

Current repackage detectors can only handle weak forms of obfuscation such as one-by-one renaming, but more sophisticated ones, are not supported, e.g., which change the defining package of classes. However, at least 20% [Gla17] of the apps found in Google Play Store [Ink17] use such techniques. The prevalent reuse of libraries in apps further inhibits the effectiveness of current detectors. Wang et al. [WGMC15] reported that more than 60% of the sub-packages in apps belong to libraries. Hence, separating the libraries from the app code is necessary. Otherwise, detectors wrongly flag apps as repackages which use similar libraries because they share a large code base. Another challenge for repackage detectors are apps generated by App Makers, e.g., apps-builder [Bui17]. In that case, the vast majority

[1] Technische Universität Darmstadt, Software Technology Group, Hochschulstraße 10, 64289 Darmstadt, Deutschland {glanz|amann|eichberg|reif|mezini}@cs.tu-darmstadt.de

of the code base will be the same, and the rest will still be very similar. Current approaches will falsely flag such apps as repackaged.
To address the challenges, we propose a library filter LibDetect and an app matcher CodeMatch, whereby the latter uses the former. LibDetect uses five hierarchically organized representations which enable an adequate precision/recall trade-off. If a library method is only weakly obfuscated, LibDetect will identify the method using a less abstract representation when compared to stronger obfuscated methods. After library filtering, our app matcher CodeMatch uses our most resilient representation as a foundation, sorts the output and performs fuzzy hashing [Kor06] on top of it to withstand various sophisticated obfuscation techniques.
For the evaluation, we built ground truths for library filtering and repackage detection by manually inspecting thousands of apps. Afterwards, we executed two library filters, and four repackage detectors to compare them with our approaches. Additionally, we executed CodeMatch with different library filters to evaluate its independent detection quality.

2 Conclusion

We presented an approach to detect repackaged apps, which relies on abstract code representations which are obfuscation-resilient. The approach consists of two steps (1) a new advanced library detection approach and (2) the fuzzy hashing of the app's code. Our evaluation shows that we can identify up to 50% more repackaged apps than the state-of-the-art. These results are due to LibDetect which correctly filters up to 70% more libraries than previous approaches.
Our implementation and all evaluation data is available at [Gla17].

References

[Bui17] A. Builder. *Apps Builder*. `http://www.apps-builder.com`. Lastaccessed: 01/11/2017. 2017 (cit. on p. 11).

[Gla17] L. Glanz. *CodeMatch Artifacts*. `http://www.st.informatik.tu-darmstadt.de/artifacts/codematch/`. Last Accessed: 06/30/2017. 2017 (cit. on pp. 11, 12).

[Ink17] G. Ink. *Google Play Store*. `https://play.google.com/store?hl=us`. Last Accessed: 12/04/2017. 2017 (cit. on p. 11).

[Kor06] J. Kornblum. "Identifying almost identical files using context triggered piecewise hashing". In: *Digital investigation* 3 (2006), pp. 91–97 (cit. on p. 12).

[WGMC15] H. Wang, Y. Guo, Z. Ma, and X. Chen. "Wukong: A scalable and accurate two-phase approach to android app clone detection". In: *Proceedings of the 2015 International Symposium on Software Testing and Analysis*. ACM. 2015, pp. 71–82 (cit. on p. 11).

M. Tichy, E. Bodden, M. Kuhrmann, S. Wagner, J.-P. Steghöfer (Hrsg.): SE 2018,
Lecture Notes in Informatics (LNI), Gesellschaft für Informatik, Bonn 2018 119

Call Graph Construction for Java Libraries

Michael Reif, Michael Eichberg, Mira Mezini[1]

Abstract: Today, every application uses software libraries. Yet, the research which targets the analysis of *libraries* – independent of any application – is scarce. This is unfortunate, because for library developers, e.g., those of the Java Development Kit (JDK), it is crucial that the library behaves as intended regardless of how it is used. In this paper, we discuss the construction of call graphs for libraries that abstract over all potential library usages. Unlike algorithms for applications, call-graph construction algorithms for libraries should distinguish between analyses w.r.t. potential exploitable vulnerabilities and those related to general software quality attributes. This distinction affects the decision about what constitutes the library-private implementation, which therefore, needs special treatment. Thus, building one call graph that satisfies all needs is not sensical.

Keywords: Call Graph Construction; Libraries; Java

1 Summary

Call graphs are a major building block of static analyses. They are, e.g., directly used to identify dead methods or act as foundation for more complex algorithms, such as solvers for data-flow problems, flow-sensitive points-to algorithms, or security-related analyses. Despite the the over-all presence of libraries in software development, a systematic discussion of constructing call graphs for libraries and their specific needs is missing.

Currently, the gold standard for constructing library call graphs is to use a standard algorithm [GC01], such as Class Hierarchy Analysis (CHA) or Variable-Type Analysis (VTA), and to consider all non-private methods as entry points. However, this ignores two properties that distinguish libraries from stand-alone applications. First, libraries are not closed worlds — they are extended by their users via inheritance. Second, libraries consist of classes and interfaces that define the public API and those which belong to the library private implementation, i.e., the part of the library that is only used internally and cannot be accessed by the libraries' users. Ignoring the first property leads to call graphs that miss important edges, ignoring the second property leads to call graphs with many spurious edges. Hence, for security focused analyses everything that could be extended should be treated as extensible; for software quality oriented analyses a fine-grained identification of the library private implementation – which takes generally accepted practices into account –

[1] Technische Universität Darmstadt, Softwaretechnik, Hochschulstraße 10, 64289 Darmstadt, Germany
{reif|eichberg|mezini}@cs.tu-darmstadt.de

should be performed. Consequently, we argue that call-graph construction algorithms for libraries must distinguish between two usage scenarios of the library.

In the first scenario, the library is assumed to be open, i.e., all non-private classes, fields, and methods can be accessed; non-final classes can be extended and non-final methods can be overridden. We use the term *open-package assumption* (OPA) to refer to this assumption and corresponding call graphs represent the *unrestricted usage scenarios* of the library. In the second scenario, only the code that belongs to a library's public API is used or gets extended. In Java, e.g., a library's classes and methods with package visibility do not belong to the public API. Additionally, all code that can only be reached via code that does not belong to the public API is also considered to belong to the library's implementation; irrespective of its visibility. We refer to this case as the *closed-package assumption* (CPA).

Under CPA, the public API reflects the usage interface that library designers intend to provide to users. CPA directly reflects the generally accepted practice: *Do not add code to the namespace of a 3rd party library*, which is already mandated by the first versions of the Java Language Specification[2]. Since then, libraries are generally developed based on this assumption[3], which represents the *intended usage scenarios* of the library.

We argue that *it is not possible to adequately address both scenarios by using the same call-graph algorithm*; any such algorithm would be either unnecessarily unsound or imprecise depending on the usage scenario. As a result, we propose and evaluate two call-graph algorithms for libraries w.r.t. OPA and CPA. Both algorithms: $\text{LibCHA}_{\text{OPA}}$ and $\text{LibCHA}_{\text{CPA}}$, build upon the CHA algorithm. The first algorithm ($\text{LibCHA}_{\text{OPA}}$) is sound under the open-package assumption, makes worst-case assumptions, and can be used to identify security (e.g. trusted method chaining attacks [Ko10]). However, the conservative algorithm may produce many spurious call graph edges, under CPA. This may lead to incorrect results — false positives and false negatives — when used for analyzing a library's implementation w.r.t. general software quality attributes.

We provide the implementation and all related data of our approach here:
Implemented within the OPAL project: `https://bitbucket.org/delors/opal`
Evaluation data via Docker: `https://hub.docker.com/r/mreif/fse2016/`.

References

[GC01] Grove, David; Chambers, Craig: A framework for call graph construction algorithms. ACM Transactions on Programming Languages and Systems (TOPLAS), 23(6):685–746, 2001.

[Ko10] Koivu, Sami: , Java Trusted Method Chaining (CVE-2010-0840/ZDI-10-056). `http://slightlyrandombrokenthoughts.blogspot.de/2010/04/java-trusted-method-chaining-cve-2010.html`, apr 2010.

[2]The part describing packages in which a developer is expected to put her code.

[3]99 of the top 100 most popular Java libraries on Maven central (`http://mvnrepository.com/popular`), as of Dec. 2015.

Architecture-Aware Online Failure Prediction for Distributed Software Systems

Teerat Pitakrat[1], Dušan Okanović[2], André van Hoorn[3], Lars Grunske[4]

Abstract: This extended abstract summarizes our article on architecture-aware online failure prediction, which has been published recently in the Journal on Software and Systems [Pi17].

Keywords: Online failure prediction; software architecture; distributed software systems

Today's software systems are complex. They comprise an immense number of distributed hardware and software components to deliver desired functionalities. Failures during production are inevitable despite successful approaches for quality assurance during software development. A failure in one component, e.g., a memory leak or slow response times, can create a chain of failures propagating to other components and the users [Av04]. Online failure prediction [SLM10] aims to foresee imminent failures by making predictions based on system parameters from monitoring data. Existing approaches employ prediction models that predict failures either for the whole system or for individual components without considering the software architecture.

We propose Hora, an architecture-aware online failure prediction approach, that combines failure prediction with architectural knowledge. The Hora approach is divided into three concurrent activities, as depicted in Fig. 1, which are component failure prediction, architectural dependency modeling, and failure propagation modeling and prediction.

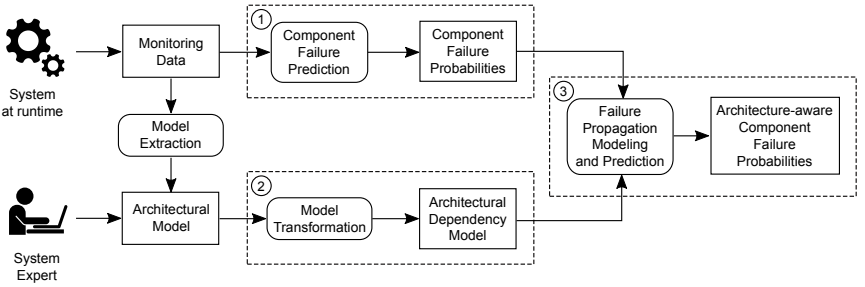


Fig. 1: Overview of the Hora approach

[1] University of Stuttgart, Institute of Software Technology, pitakrat@informatik.uni-stuttgart.de
[2] University of Stuttgart, Institute of Software Technology, okanovic@informatik.uni-stuttgart.de
[3] University of Stuttgart, Institute of Software Technology, van.hoorn@informatik.uni-stuttgart.de
[4] Humboldt University Berlin, Department of Computer Science, grunske@informatik.hu-berlin.de

The first activity of HORA is component failure prediction, denoted by (1). At runtime, the predictors receive monitoring data collected from application performance management (APM) tools [He17]. Each predictor is responsible for continuously predicting the failure of one component in the system. In the current implementation, we employ Kieker [vHWH12] to monitor method response times and resource utilization. The prediction technique used for predicting the failures is time series forecasting. Other prediction techniques, e.g., machine learning, can be employed if they can make predictions based on the monitoring data.

The second activity is architectural dependency modeling, denoted by (2). An Architectural Dependency Model (ADM) is a model that represents dependencies between components in the system. It indicates how components are connected and how likely a failure of one component will affect another. The ADM can be extracted from existing architectural models, e.g., SLAstic, or from monitoring data provided by APM tools, e.g., Kieker [vHWH12].

The third activity is failure propagation modeling and prediction, denoted by (3). The results of the first activity (component failure probabilities) and the ADM in the second activity are provided to a Failure Propagation Model (FPM). An FPM is a model that employs Bayesian network theory to model how a failure propagates through components until it reaches the system boundary and the users. The FPM continuously receives information from these two sources and periodically computes a system failure probability.

The evaluation of the HORA approach is performed on a microservice-based RSS feed reader application originally developed by Netflix. Three types of faults are injected into the system and HORA is applied to predict system failures. The results show that the prediction quality is improved when software architectural knowledge is considered explicitly.

References

[Av04] Avizienis, Algirdas; Laprie, Jean-Claude; Randell, Brian; Landwehr, Carl: Basic Concepts and Taxonomy of Dependable and Secure Computing. IEEE Transactions on Dependable and Secure Computing, 1(1):11–33, 2004.

[He17] Heger, Christoph; van Hoorn, André; Mann, Mario; Okanović, Dušan: Application Performance Management: State of the Art and Challenges for the Future. In: Proceedings of the 8th ACM/SPEC International Conference on Performance Engineering (ICPE). ACM, pp. 429–432, 2017.

[Pi17] Pitakrat, Teerat; Okanović, Dušan; van Hoorn, André; Grunske, Lars: Hora: Architecture-aware Online Failure Prediction. Journal of Systems and Software, 2017. In press. Online first: https://doi.org/10.1016/j.jss.2017.02.041.

[SLM10] Salfner, Felix; Lenk, Maren; Malek, Miroslaw: A Survey of Online Failure Prediction Methods. ACM Computing Surveys, 42(3):10:1–10:42, 2010.

[vHWH12] van Hoorn, André; Waller, Jan; Hasselbring, Wilhelm: Kieker: A Framework for Application Performance Monitoring and Dynamic Software Analysis. In: Proceedings of the 3rd ACM/SPEC International Conference on Performance Engineering (ICPE). ACM, pp. 247–248, 2012.

SE Session 10 - Software Process

Inner Source in Platform-based Product Engineering

Dirk Riehle[1], Maximilian Capraro[2], Detlef Kips[3], Lars Horn[4]

Abstract: Inner source is the use of open source best practices within an organization. As such, it is an approach to collaboration across intra-organizational boundaries for the creation of shared reusable assets. Prior project reports on inner source suggest improved code reuse and better knowledge sharing. Using a multiple-case case study research approach, we analyze the problems that three major software development organizations were facing in their product line engineering efforts. We find that a root cause, the separation of product units as profit centers from a platform organization as a cost center, leads to delayed deliveries, increased defect rates, and redundant software components. This article is a not-so-much extended abstract of [R+16].

Keywords: Inner source, product line engineering, product families, platform-based product engineering, open source, open collaboration, case study research.

1 Research Question and Approach

This article presents case study research on the situation of three major software development organizations which were trying to apply inner source to platform-based product engineering.

Our case study companies expected inner source to help them overcome problems with lack of resources, lack of pertinent skills, and unclear requirements. Yet, they had problems putting inner source into practice. To this end, this article addresses the following research questions:

- *RQ1: What are current problems in platform-based product engineering?*
- *RQ2: What benefits do organizations expect from adopting inner source?*
- *RQ3: What problems did they experience when adopting inner source?*

The research method employed is *multiple-case case study research*. Data gathering and analysis was performed using workshops, formal interviews, and materials review. The process was incremental with learnings being provided back to the case study

[1] Computer Science Department, Friedrich-Alexander University Erlangen-Nürnberg, 91058 Erlangen, Germany. E-mail: dirk.riehle@fau.de.
[2] Computer Science Department, Friedrich-Alexander University Erlangen-Nürnberg, 91058 Erlangen, Germany. E-mail: maximilian.capraro@fau.de.
[3] Develop Group, 91058 Erlangen, Germany. E-mail: kips@develop-group.de.
[4] E-solutions, 91058 Erlangen, Germany. E-mail: lars.horn@esolutions.de.

participants to receive validating feedback as to the theories being built ("member checking").

2 Contributions and Results

Similar to open source, which often evolved from a volunteer-based ("free-for-all") development process to a foundation-based ("managed") software development process, we find that for our case study organizations, inner source should move to a governed process beyond the definition given in the beginning of this section.

The theories we present are only as good as the hypotheses that they generate and that can be validated in future work. Such confirmatory research will also allow for generalized conclusions that are not possible from pure case study research.

H1 Resistance and misunderstandings (like expected lower code quality of inner source components) can be addressed by way of education and active participation in the practice of inner source software development.

H2 Psychological openness or resistance to inner source (i.e. desire or fear to work under quasi-public scrutiny) depends on manager and developer personalities and is not a function of organizational structure or process.

H3 As long as open source does not come natural to an organization, inner source will not come easy to it either. Until this has changed, an organization will need an explicitly governed inner source process.

H4 Inner source and open source draw on the same competencies of people and a person who is good at one is likely to be good at the other.

H5 While there is no doubt about the need of platform software and shared reusable assets, a platform development organization may not be needed any longer. It can be replaced by an inner source program.

The last hypothesis is an interesting though probably controversial hypothesis: If large companies can work together in an open source foundation to develop shared infrastructure components, why can't product units within an organization work together to create a platform of shared reusable assets without the need for a dedicated organizational unit that maintains this platform?

References

[R+16] Riehle, D., Capraro, M., Kips, D., & Horn, L. (2016). Inner source in platform-based product engineering. *IEEE Transactions on Software Engineering*, *42*(12), 1162-1177.

Observations on Knowledge Transfer of Professional Software Developers during Pair Programming

Franz Zieris[1], Lutz Prechelt[2]

This is an extended abstract of the paper with the same title [ZP16] which was presented at the 38th International Conference on Software Engineering (2016).

Keywords: pair programming; knowledge transfer; grounded theory

1 Background, Context, and Research Method

Pair programming (PP) in industrial settings is usually understood as either a productive practice in the XP-sense [Be99] or as a mentoring technique to bring new team members up to speed. It appears to be common sense that pair programmers are either good enough to perform a productive session (without any relevant knowledge transfer) or their skill levels are too far apart to be productive, so they retreat to a knowledge transfer session. But such a strict dichotomy does not capture the full reality of industrial software development: Knowledge transfer is a key ingredient of any pair programming session (*especially* between two experts), and even a knowledge-wise inferior developer can have big positive influence on the PP session's progression.

In contrast to just looking for an effect of pair programming on developers' knowledge levels, we want to understand the actual process of how knowledge transfer happens during pair programming. For this, we record in-vivo PP sessions of industrial software developers and analyze this material (consisting of screen capture, webcam, and audio) on an utterance-level granularity employing Grounded Theory Methodology [SC90].

2 Results

In a previous article [ZP14], we reported our initial findings on what we called "Knowledge Transfer Skill" in pair programming: Good pairs manage to (1) not pursue multiple knowledge needs at once, (2) recognize complex Topics and split them into separate Topics,

[1] Freie Universität Berlin, Institut für Informatik, Takustr. 9, 14195 Berlin, Deutschland zieris@inf.fu-berlin.de
[2] Freie Universität Berlin, Institut für Informatik, Takustr. 9, 14195 Berlin, Deutschland prechelt@inf.fu-berlin.de

(3) recognize complicated Topics and deal with them in stages, and (4) not lose sight of Topics. In our recent study [ZP16], we analyzed well over 400 knowledge transfer episodes from 13 PP sessions and were able to conceptualize more observations that are relevant for understanding knowledge transfer in pair programming:

- There is usually no pair member who is more knowledgeable in *all* relevant areas. Even in sessions dedicated to introducing a new team member, the senior developer learned something along the way.
- Occasionally, it is the knowledge-wise inferior developer who has the bigger positive impact on the session's progress, either through solving a problem (insight) or by avoiding a mistake (diligence).
- Existing knowledge is not just *"pulled"* from the more knowledgeable developer through asking questions; developers will also start explanations (*"push"*), even if her partner did not specifically ask for it. In PP, knowledge gaps can be dealt with even if the developer in need is not aware of them.
- When new understanding is being *"produced"* by one pair member alone (e.g., by reading source code), sinking into silence is a bad choice compared to uttering intermediate insights as they would allow the partner to both help and learn.
- When both developers cooperatively work on new understanding, one developer occasionally pulls ahead. The pair then needs to get close together again to fully use their complementary knowledge. This *resynchronization* requires some additional effort, which will pay off later.
- But even equally quick grasp of both developers does not guarantee common understanding: Producing new understanding as a pair always requires some additional synchronization. Otherwise the pair risks ending up on *parallel tracks* and taking avoidable detours in their process.

References

[Be99] Beck, Kent: Extreme Programming Explained: Embrace Change. Addison-Wesley Professional, 1999.

[SC90] Strauss, Anselm L.; Corbin, Juliet M.: Basics of Qualitative Research: Grounded Theory Procedures and Techniques. SAGE, 1990.

[ZP14] Zieris, Franz; Prechelt, Lutz: On Knowledge Transfer Skill in Pair Programming. In: Proc. 8th ACM/IEEE Int'l. Symposium on Empirical Software Engineering and Measurement. ESEM '14, ACM, New York, NY, USA, pp. 11:1–11:10, 2014.

[ZP16] Zieris, Franz; Prechelt, Lutz: Observations on Knowledge Transfer of Professional Software Developers During Pair Programming. In: Proc. 38th Int'l. Conf. on Software Engineering Companion. ICSE '16, ACM, New York, NY, USA, pp. 242–250, 2016.

LoCo CoCo: Automatically Constructing Coordination and Communication Networks from Model-Based Systems Engineering Data

Mazen Mohamad[1], Grischa Liebel[2], Eric Knauss[2]

Keywords: Systems Engineering; Communication; Coordination; Requirements Clarification; Empirical Software Engineering

1 Extended Abstract

Communication and coordination are essential activities in software and systems engineering [KS95]. In particular, communicating requirements [He07], and sharing the product and contextual knowledge required to understand requirements during analysis and development [Li16] are challenging.

The structure of all interacting individuals and groups in an organisation can be described and analysed as a social network. Social network analysis (SNA) and has successfully been used in software engineering to facilitate collaboration and relationships among individuals in software teams [Wo09].

Existing automated approaches for SNA, such as Codebook [BKZ10], focus on software development artefacts on a low level of abstraction, such as source code or bug requests. However, in large systems engineering projects, communication between different disciplines is required [Li16], taking place on a domain level independent of the source code.

We present LoCo CoCo, the Low-Cost Communication and Coordination approach, the result of a one-year design science research project at a large automotive original equipment manufacturer (OEM). LoCo CoCo automatically creates social networks from model-based systems engineering data by leveraging a structural meta model similar to standards like EAST-ADL [EA].

We identify people and their relations by extracting ownership and trace information from systems engineering data. The resulting networks are used as a supporting tool for enabling or improving communication and coordination. We evaluated LoCo CoCo analytically, by constructing social networks from real-life systems engineering data at the industrial

[1] Chalmers University of Technology, Gothenburg, Sweden mazenmhd@gmail.com
[2] Chalmers | University of Gothenburg, Department of Computer Science and Engineering, Gothenburg, Sweden. grischa@chalmers.se, knauss@chalmers.se

partner. Additionally, we collected empirical data from 15 interviews and 12 surveys with practitioners.

Our results indicate that LoCo CoCo helps addressing existing communication challenges by identifying important contacts across the organisation structure, thus facilitating communication of requirements in systems engineering. While we observed that the quality of social data in existing systems engineering tools, such as ownership data or information about who changed elements, is sometimes low, practitioners rated it as sufficient. Furthermore, visualising erroneous connections due to outdated social data can trigger practitioners to update the data. Finally, we elicited several ethical implications arising from the use of social data. These will have to be considered when using LoCo CoCo or similar approaches in industry.

Acknowledgments

We thank the case company and all participants in the original study for their great support and deep discussions. This work originates from a study conducted as a Master Thesis and was published previously at Information and Software Technology Journal [MLK17].

References

[BKZ10] Begel, Andrew; Khoo, Yit Phang; Zimmermann, Thomas: Codebook: Discovering and Exploiting Relationships in Software Repositories. In: Proceedings of the 32Nd ACM/IEEE International Conference on Software Engineering - Volume 1. pp. 125–134, 2010.

[EA] EAST-ADL Association: , EAST-ADL V2.1.12. http://www.east-adl.info/Specification/V2.1.12/html/index.html. last accessed Jun. 2015.

[He07] Herbsleb, J. D.: Global Software Engineering: The Future of Socio-technical Coordination. In: Future of Software Engineering, 2007. FOSE '07. pp. 188–198, 2007.

[KS95] Kraut, Robert E.; Streeter, Lynn A.: Coordination in Software Development. Commun. ACM, 38(3):69–81, 1995.

[Li16] Liebel, Grischa; Tichy, Matthias; Knauss, Eric; Ljungkrantz, Oscar; Stieglbauer, Gerald: Organisation and communication problems in automotive requirements engineering. Requirements Engineering, pp. 1–23, 2016.

[MLK17] Mohamad, Mazen; Liebel, Grischa; Knauss, Eric: LoCo CoCo: Automatically constructing coordination and communication networks from model-based systems engineering data. Information and Software Technology, 92(Supplement C):179–193, 2017.

[Wo09] Wolf, T.; Schröter, A.; Damian, D.; Panjer, L.D.; Nguyen, T. H.: Mining Task-Based Social Networks to Explore Collaboration in Software Teams. IEEE Software, 26(1):58–66, jan 2009.

SE Session 11 - Requirements and Traceability

Requirements Engineering Challenges in Large-Scale Agile System Development

Rashidah Kasauli[1], Grischa Liebel, Eric Knauss, Swathi Gopakumar, Benjamin Kanagwa[2]

Keywords: requirements engineering; large-scale agile; system engineering

Despite wide critic, agile approaches have significantly contributed to the way software is developed [Me14]and success stories have led to their application at large scale [DPL16] and in system development [BE15; PMG12], an environment characterized by long lead times [BE15] and stable, sequential engineering practices [PMG12]. In this environment, new challenges arise, especially with respect to managing requirements [SKV10].

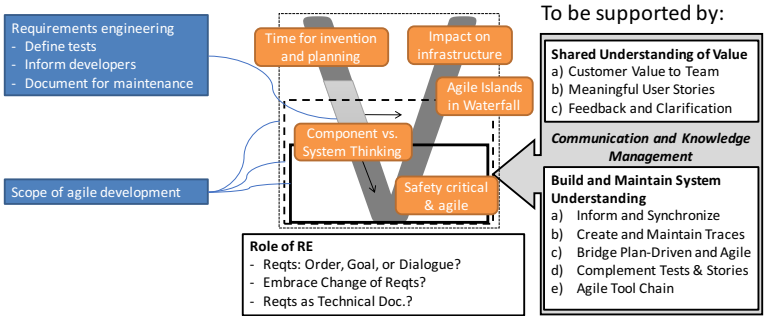


Fig. 1: Map of challenges with respect to scope of agile work in system development

We address the lack of empirical studies and report *RE related challenges of large-scale agile system development*. Through a multiple case study of four large-scale system development cases, based on 5 focus groups, 2 cross-company workshops and 20 semi-structured interviews, we present a catalogue of real-world RE challenges related to applying agile development in large-scale systems (Fig. 1). These challenges are effectively hindering a faster and more sustainable development of software. In order to yield their full benefits, agile practices and a holistic system requirements model must be better aligned (Tab. 1). Key challenges occur when there is an interaction, or a lack thereof, between system engineering domains and we believe that industry would benefit from new impulses from research in the area of Requirements Engineering for Large-Scale Agile System Development.

Acknowledgements: This work was supported by Software Center Project 27 on RE for

[1] Rashidah, Grischa, Eric and Swathi are with Chalmers | University of Gothenburg, Department of Computer Science and Engineering, Gothenburg, Sweden. Contact rashida@chalmers.se or knauss@chalmers.se
[2] Rashidah and Benjamin are with Makerere University, Kampala, Uganda. bkanagwa@cis.mak.ac.ug

Tab. 1: Conclusions of our research

1) Communication and knowledge management.	While related work implies that communication challenges are less prominent in agile RE [BWR11; In15], our challenges relate to communication and knowledge management. Both aspects are at the core of Agile and RE, indicating a need for research in these areas specifically for system development.
2) Two areas of requirements knowledge: User Value and System Understanding.	This is in line with traditional practices of user and system requirements, but not present in agile literature. Companies differ between doing RE in an agile way and doing RE to support agility. Our findings suggest that such support cannot be offered sufficiently by traditional, upfront RE, as indicated [He17; Me14].
3) The interplay of stakeholders from three domains: customer, development, and integration & testing.	Development embraces agility and dislikes traditional requirements and bulk updates, which requires better *synchronization between teams* and *establishing of an agile tool-chain*. Customer domain is concerned with breaking down customer-visible features in order to communicate *customer-value to team*, requiring support for *writing meaningful user stories* and for *bridging the gap between plan-driven and agile development*. Integration and testing domain is struggling to *create and maintain traces* and with the fact that *user stories and tests* are not sufficient to build and maintain sufficient system understanding.

Large-Scale Agile System Dev. and the SIDA BRIGHT project and was published previously at the International Requirements Engineering Conference [Ka17].

References

[BE15] Berger, C.; Eklund, U.: Expectations and Challenges from Scaling Agile in Mechatronics-Driven Companies – A Comparative Case Study. In: XP '15. 2015.

[BWR11] Bjarnason, E.; Wnuk, K.; Regnell, B.: A case study on benefits and side-effects of agile practices in large-scale requirements engineering. In: 1st Agile Reqts. Eng. WS. 2011.

[DPL16] Dikert, K.; Paasivaara, M.; Lassenius, C.: Challenges and Success Factors for Large-Scale Agile Transformations: A Systematic Literature Review. JSS/, 2016.

[He17] Heikkilä, V. T.; Paasivaara, M.; Lasssenius, C.; Damian, D.; Engblom, C.: Managing the requirements flow from strategy to release in large-scale agile development: a case study at Ericsson. Empirical Software Engineering/, pp. 1–45, 2017.

[In15] Inayat, I.; Salim, S. S.; Marczak, S.; Daneva, M.; Shamshirband, S.: A systematic literature review on agile requirements engineering practices and challenges. Computers in human behavior 51/, pp. 915–929, 2015.

[Ka17] Kasauli, R.; Liebel, G.; Knauss, E.; Gopakumar, S.; Kanagwa, B.: Requirements Engineering Challenges in Large-Scale Agile System Development. In: RE '17. Lisbon, Portugal, 2017.

[Me14] Meyer, B.: Agile! The Good, the Hype and the Ugly. Springer, 2014.

[PMG12] Pernstål, J.; Magazinius, A.; Gorschek, T.: A study investigating challenges in the interface between product development and manufacturing in the development of software-intensive automotive systems. SEKE 22/07, pp. 965–1004, 2012.

[SKV10] Savolainen, J.; Kuusela, J.; Vilavaara, A.: Transition to agile development: rediscovery of important requirements engineering practices. In: RE '10. Pp. 289–294, 2010.

Hazard Relation Diagrams: a diagrammatic representation to increase validation objectivity of requirements-based hazard mitigations

Bastian Tenbergen[1,2], Thorsten Weyer[2], Klaus Pohl[2]

Abstract: This talk is based on a paper published in the Requirements Engineering Journal in May 2017 [TWP17]. During the development of safety-critical systems, the development process must ensure that requirements, which are defined to mitigate a hazard, are adequate. Adequacy of such hazard-mitigating requirements (HMRs) means that the requirements may not oppose the system's operational purpose and must sufficiently avoid, reduce, or control, the occurrence of the conditions that trigger the hazard. However, information about the occurrence of the hazard's trigger conditions are a work product of hazard analyses during early stages of safety assessment, while HMRs are a work product of requirements engineering. Dependencies between HMRs and hazard analysis results are implicit and tacit. In consequence, there's a risk that during validation, inadequacy of HMRs regarding their ability to mitigate a hazard remains covert. The result may be that the system is assumed to be safe, but in fact may still cause injury or death. We introduced Hazard Relation Diagrams (HRDs) as a means to integrate and graphically visualize hazard analysis results with HMRs. Herein, we also provide insights into their empirical evaluation and show that HRDs increase objectivity in rationales containing adequacy judgments.

Principles and Visual Notation of Hazard Relation Diagrams

Hazard Relation Diagrams integrate HMRs with the hazard they are intended to mitigate in a single diagram [TWP15]. During validation, HRDs is reviewed individually and sequentially, thereby allowing for alternative mitigations to be validated with regard to each respective hazard. HMRs are depicted using modeling elements of UML activity diagrams. HRDs contain exactly one hazard and several mitigation partitions to support different multiplicities between hazards and HMRs. The dashed mitigation partitions surround the HMRs and can be distributed across "geometrically" distant areas within the same or several activity diagrams. HRDs contain the hazard's tree of trigger conditions, the conceived safety goal, and one Hazard Relation, which is an n-ary association relating the hazard, trigger conditions, safety goal, and mitigation partitions. An example is shown in Fig. 1 (activity labels are removed for legibility).

Empirical Evaluation Shows Increases in Review Objectivity

In two empirical experiments involving a total of 168 graduate and undergraduate students [TWP17], the hypothesis was investigated whether there is an impact on

[1] State University of New York at Oswego, NY, USA, bastian.tenbergen@oswego.edu
[2] University of Duisburg-Essen, paluno, Germany {thorsten.weyer, klaus.pohl}@paluno.uni-due.de

objectivity when using HRDs to validate the adequacy of HMRs (treatment condition) versus using activity diagrams (control condition). Participants were asked to review 10 hazard mitigations and judge if the hazard can still occur during operation and to justify their judgement in a written rationale. Rationales were categorized into those that mention "semantics" or "syntax" (i.e., diagram properties also found in activity diagrams, see also combined variable H1.a), or mention "mitigation," "trigger condition," or "safety goals" (i.e., properties specific to HRDs, see also combined variable H1.b). Fig. 2 shows the differences between treatment (black bars) and control (grey bars) conditions. Significant differences bear p-value, effect size, and achieved statistical power and show that using HRDs, judgments were more often based on objective information about the hazard, rather than on the diagram's meaning or style.

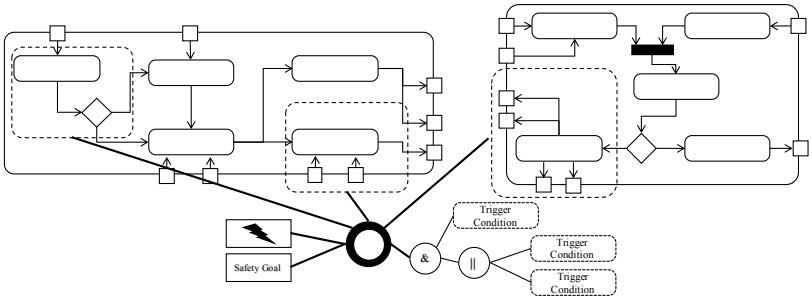


Fig. 1. Example of a HRD with three mitigation partitions surrounding HMRs in two activity diagrams.

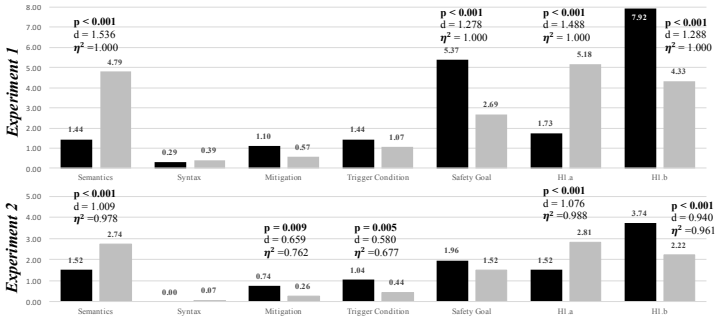


Fig. 2. Experimental Results show signitificantly more Judgments based on Hazard Analyses using HRDs.

References

[TWP15] B. Tenbergen, T. Weyer, and K. Pohl, "Supporting the validation of adequacy in requirements-based hazard mitigations," in Springer LNCS 9013, 2015, pp. 17-32.

[TWP17] B. Tenbergen, T. Weyer, and K. Pohl, "Hazard Relation Diagrams: A Diagrammatic Representation to Increase Validation Objectivity of Requirements-based Hazard Mitigations," in Requirements Eng J. DOI: 10.1007/s00766-017-0267-9.

M. Tichy, E. Bodden, M. Kuhrmann, S. Wagner, J.-P. Steghöfer (Hrsg.): SE 2018,
Lecture Notes in Informatics (LNI), Gesellschaft für Informatik, Bonn 2018 139

Experiences on Traceability and Consistency Checking across Engineering Tools in an Automation Solution Company

Andreas Demuth[1], Roland Kretschmer[2], Michael Tröls[2], Georgios Kanakis[2], Davy Maes[3], and Alexander Egyed[2]

Abstract: Engineers continuously adapt systems to changing requirements, which is particularly then a challenge when different engineering domains come together. Since engineers of different domains use quite distinct engineering tools, consistent change propagation is essential. This paper discusses experiences with a leading company in the area of production automation in maintaining the consistency between electrical models and the corresponding software controller when both are subject to continuous change. This is complicated by the fact that these engineer use different kinds of tools to capture and maintain models and code.

Keywords: Traceability, Consistency Checking, Experience Report, DesignSpace, Collaboration;

1 Introduction

The engineering of systems is unimaginable without software tools. Engineers use them to capture and analyze engineering problems; specify, implement, test, and maintain engineering solutions, and manage engineering processes. Yet, there is a gap between the capabilities of independently working engineers and the needs of a collaborative engineering team. The existing tool landscape emphasizes the former. Most engineering tools are single-user applications – often of excellent quality but limited in that they support the works of individual engineers and not that of a group of engineers. Herein lies one of the most fundamental problems of software and systems engineering. Engineers know well the engineering tools they use and the engineering knowledge they capture. Yet, engineers lack awareness of the many implications their work has on other engineers and/or other engineering domains. This is a problem because in today's engineering projects, companies continuously adapt their systems to changing customer or market requirements. This requires a flexible, iterative development process in which engineers build and update different parts of the system under construction concurrently.

This paper discusses such an experience in context of the construction of a conveyor belt system with Van Hoecke Automation. This system requires the collaboration among electrical engineers and software engineers. Not only do these engineers work on

[1] Dynatrace Austria GmbH, Freistaedter Strasse 313/2, 4040 Linz, Austria, andreas.demuth@jku.at
[2] Johannes Kepler Universität, ISSE, Altenbergerstraße 69, 4040 Linz, Austria, [firstName].[lastName]@jku.at
[3] Flanders Make, Oude Diestersebaan 133 Lommel, 3920 Belgium, davy.maes@flandersmake.be

different perspectives of that system but they use quite distinct tools to capture their engineering knowledge. As such, the electrical engineer captures electrical circuit diagrams in EPlan Electric P8 and the software engineering captures source code in Eclipse. Neither tool knows about the existence of the other. The key question that we asked was how these engineers could be made aware of the respective implications of their changes on each other's domain.

To address this problem, we developed the DesignSpace cloud [De15] to let engineers define relationships among arbitrary development artifacts. This way engineers can connect e.g. a motor element from the circuit diagram to its respective piece of code. Engineers define these relationships through explicit links in a wizard-style interface. These links (i.e., traceability [An01]) then provide the basis for consistency checking among the artifacts of these two tools. Changes engineers make in their tools are instantly synchronized with the DesignSpace and engineers receive instant error feedback if such changes violate defined consistency rules. For scalability, the DesignSpace cloud utilizes the Model/Analyzer approach to fast, incremental consistency checking [Eg11]. Further details about the case study are provided in an experience report [De16] we published previously at the International Conference on Software Maintenance and Evolution.

2 Acknowledgement

This research was supported by the Austrian Science Fund (FWF): P25289 N15, the Austrian COMET-K2 Programme of the Linz Center of Mechatronics (LCM), and a grant of the JKU Linz Institute ofTechnology (LIT)

3 References

[An01] Giuliano Antoniol, Bruno Caprile, Alessandra Potrich, and Paolo Tonella. Design-code traceability recovery: selecting the basic linkage properties. *Sci. Comput. Program.*, 40(2-3):213–234, 2001.

[De16] Andreas Demuth, Roland Kretschmer, Alexander Egyed, and Davy Maes. Introducing traceability and consistency checking for change impact analysis across engineering tools in an automation solution company: An experience report. In *International Conference on Software Maintenance and Evolution (ICSME), USA, 2016*.

[De15] Andreas Demuth, Markus Riedl-Ehrenleitner, Alexander Nöhrer, Peter Hehenberger, Klaus Zeman, and Alexander Egyed. Designspace: an infrastructure for multi-user/multi-tool engineering. In *Proceedings of the 30th Annual ACM Symposium on Applied Computing, Salamanca, Spain, April 13-17, 2015*, pages 1486–1491, 2015.

[Eg11] Alexander Egyed. Automatically detecting and tracking inconsistencies in software design models. *IEEE Transactions on Software Engineering*, 37(2):188–204, 2011.

SE Session 12 - Design

M. Tichy, E. Bodden, M. Kuhrmann, S. Wagner, J.-P. Steghöfer (Hrsg.): SE 2018,
Lecture Notes in Informatics (LNI), Gesellschaft für Informatik, Bonn 2018 143

On Continuous Detection of Design Flaws in Evolving Object-Oriented Programs using Incremental Multi-Pattern Matching

Sven Peldszus[1], Géza Kulcsár[2], Malte Lochau[2], Sandro Schulze[3]

Abstract: This work has been initially presented at the International Conference on Automated Software Engineering (ASE) 2016 [Pe16]. Design flaws in object-oriented programs may seriously corrupt code quality thus increasing the risk for introducing subtle errors. Most recent approaches identify design flaws in an ad-hoc manner, either focusing on software metrics, locally restricted code smells, or on coarse-grained architectural anti-patterns. In our work, we utilize an abstract program model capturing high-level object-oriented code entities, further augmented with qualitative and quantitative design-related information. Based on this model, we propose a comprehensive methodology for specifying object-oriented design flaws by means of compound rules integrating code metrics, code smells and anti-patterns in a modular way. This approach allows for efficient, automated design-flaw detection, by facilitating systematic information reuse among multiple detection rules as well as between subsequent detection runs on continuously evolving programs.

Summary

Object-oriented programming offers software developers rich concepts for structuring initial program designs. As software systems tend to become more and more long-living, their initial code bases have to be continuously maintained, improved and extended. In practice, corresponding evolution steps are frequently conducted in an ad-hoc manner. As a result, the initial program design may be prone to continuous erosion, eventually leading to structural decay whith negative side-effects such as increasing the risk for introducing subtle errors.

Object-oriented refactorings have been proposed as effective counter-measure against design flaws. In fact, a manual identification of problematic code structures to be removed by applying appropriate refactorings is tedious, error-prone, or even impossible for larger-scaled software projects. Various approaches have been recently proposed to assist and/or automate the identification of design flaws. The different attempts may be roughly categorized into three kinds of *symptoms*, potentially indicating object-oriented design flaws [Mo10].

- *Software metrics* assess quality problems in program designs by means of quantified measures on structural code entities (e.g. low cohesion of classes).

[1] University of Koblenz-Landau, Germany speldszus@uni-koblenz.de
[2] TU Darmstadt, Germany geza.kulcsar@es.tu-darmstadt.de, malte.lochau@es.tu-darmstadt.de
[3] OVGU Magdeburg, Germany sandro.schulze@iti.cs.uni-magdeburg.de

- *Code smells* qualify problematic, locally restricted code structures and anomalies in-the-small, at class- or member-level (e.g., large classes).
- *Anti-patterns* qualify architectural decay in-the-large, usually involving several classes spread over the entire program (e.g., God Classes) [Br98].

Based on this taxonomy, a precise and reliable identification of actual occurrences of design flaws requires arbitrary combinations of software metrics with adjustable thresholds, as well as code smells and anti-patterns into *compound detection rules* [Mo10]. However, most existing approaches lack a comprehensive formal foundation and a uniform, yet modular representation of such design-flaw detection rules. Instead, specifically tailored detection routines are applied for every design flaw individually, and being re-evaluated from scratch for every program version anew during software evolution.

In our work, we present a comprehensive methodology for specifying and automatically detecting design flaws in object-oriented programs. The approach utilizes a unified abstract program model comprising those high-level object-oriented code entities being relevant for a concise specification of well-known design flaws. Based on this model, compound design-flaw detection rules integrate software metrics, code smells and anti-patterns, and allow for arbitrary combinations thereof. The modular nature of the rule language allows for sharing similar symptoms among multiple rules. The corresponding pattern-matching routines derived from those rules incrementally augment the underlying abstract program model with qualitative and quantitative design-related information. This technique builds the basis for efficient design-flaw detection by systematically facilitating reuse of information among multiple detection rules, as well as between subsequent detection runs on continuously evolving programs.

Please note that our tool implementation as well as all experimental data sets are available on our GitHub site (`https://github.com/GRaViTY-Tool/`).

Acknowledgements

This work has been supported by the DFG in the Priority Programme SPP 1593: Design For Future – Managed Software Evolution (LO 2198/2-1, JU 2734/2-1).

Literaturverzeichnis

[Br98] Brown, William J.; Malveau, Raphael C.; McCormick, Hays W. “Skip”; Mowbray, Thomas J.: AntiPatterns: Refactoring Software, Architectures, and Projects in Crisis. Wiley, 1998.

[Mo10] Moha, Naouel; Guéhéneuc, Yann-Gaël; Duchien, Laurence; Le Meur, Anne-Francoise: DECOR: A Method for the Specification and Detection of Code and Design Smells. TSE, 36(1):20–36, 2010.

[Pe16] Peldszus, Sven; Kulcsár, Géza; Lochau, Malte; Schulze, Sandro: Continuous Detection of Design Flaws in Evolving Object-Oriented Programs using Incremental Multi-pattern Matching. In: ASE. 2016.

M. Tichy, E. Bodden, M. Kuhrmann, S. Wagner, J.-P. Steghöfer (Hrsg.): SE 2018,
Lecture Notes in Informatics (LNI), Gesellschaft für Informatik, Bonn 2018 145

Symbolic Execution for Realizability-Checking of Scenario-based Specifications[1][2]

Joel Greenyer[3], Timo Gutjahr[4]

Abstract: Scenario-based specification with the Scenario Modeling Language (SML) is an intuitive approach for formally specifying the behavior of reactive systems. SML is close to how humans conceive and communicate requirements, yet SML is executable and simulation and formal realizability checking can find specification flaws early. The realizability checking complexity is, however, exponential in the number of scenarios and variables. Therefore algorithms relying on explicit-state exploration do not scale and, especially when specifications have message parameters and variables over large domains, fail to unfold their potential. In this paper, we present a technique for the symbolic execution of SML specifications that interprets integer message parameters and variables symbolically. It can be used for symbolic realizability checking and interactive symbolic simulation. We implemented the technique in ScenarioTools. Evaluation shows drastic performance improvements over the explicit-state approach for a range of examples. Moreover, symbolic checking produces more concise counter examples, which eases the comprehension of specification flaws.

Keywords: Reactive Systems; Scenario-Based Modeling; Realizability Checking; Symbolic Execution

Many software-intensive systems, especially cyber-physical systems, consist of reactive components that interact with each other and the environment in order to realize complex and often safety-critical functionality. Scenario-based modeling with *Live Sequence Charts* (LSCs) [HM03], and a textual variant, the *Scenario Modeling Language* (SML), is an intuitive, yet formal approach for specifying the behavior of such systems. SML extends LSCs with concepts for specifying environment assumptions and dynamic topologies [Gr17].

LSC/SML specifications are executable via the *play-out* algorithm [HM03], and can be analyzed via simulation. Violations encountered during simulation runs hint at possible specification flaws. Simulation alone, however, cannot prove the absence of flaws. For this purpose, there exist methods for proving the *realizability* of LSC/SML specifications [BH05, MS12], i.e., checking whether there exist an implementation of the specification or not, due to contradicting requirements or other inconsistencies.

[1] This is an extended abstract of [GG17]

[2] Funded by the German Israeli Foundation for Research and Development (GIF), grant No. 1258, and the Deutsche Forschungsgemeinschaft (DFG), project EffiSynth.

[3] Leibniz Universität Hannover, Software Engineering Group, Welfengarten 1, 30167 Hannover, Germany, greenyer@inf.uni-hannover.de

[4] Zühlke Engineering GmbH, Podbielskistraße 333, 30659 Hannover, Germany Timo.Gutjahr@zuehlke.com

These approaches, however, usually do not scale well, since they rely on an exploration of the state space induced by the specification, which can be exponential in the number of scenarios and variables in the specification. Approaches relying on BDD-based algorithms [MS12] only support basic scenario-based language concepts, since the mapping of elaborate scenario language concepts to lower-level formalisms and tools is very difficult.

We therefore investigated how to extend the existing play-out algorithms with *symbolic execution* [Ki76]. Symbolic execution, known from program analysis, executes a program with symbolic values as inputs; it can deduce for which constraints on these inputs it is possible to arrive at a particular part of a program that is of interest, e.g., the violation of an assertion. In the case of symbolic play-out, environment event parameters and initial component state attributes are interpreted symbolically. The particular challenge, compared to the symbolic execution of sequential programs, is twofold: (1) during a symbolic play-out execution, inputs occur repeatedly, and (2) in an execution state, multiple scenarios can be active and formulate conditions over message parameter and variable values. One step in symbolic play-out could thus result in branching into as many paths as required to cover all possible combinations of branchings implied by each active scenario.

We formalized symbolic play-out, and implemented it within ScenarioTools, which we combined with Z3 for constraint solving. We experimented with different strategies to match symbolic execution states, which is required for systematically exploring execution paths. The evaluation shows that symbolic play-out can drastically improve realizability checking performance for specifications with message parameters and variables over large domains.

References

[BH05] Bontemps, Yves; Heymans, Patrick: From Live Sequence Charts to State Machines and Back: A Guided Tour. IEEE Transactions on Software Engineering, 31(12):999–1014, 2005.

[GG17] Greenyer, Joel; Gutjahr, Timo: Symbolic Execution for Realizability-Checking of Scenario-Based Specifications. In: 2017 ACM/IEEE 20th International Conference on Model Driven Engineering Languages and Systems (MODELS). volume 00, pp. 312–322, Sept. 2017.

[Gr17] Greenyer, Joel; Gritzner, Daniel; Gutjahr, Timo; König, Florian; Glade, Nils; Marron, Assaf; Katz, Guy: ScenarioTools – A tool suite for the scenario-based modeling and analysis of reactive systems. Science of Computer Programming, 149(Supplement C):15 – 27, 2017. Special Issue on MODELS'16.

[HM03] Harel, D.; Marelly, R.: Come, Let's Play: Scenario-Based Programming Using LSCs and the Play-Engine. Springer, 2003.

[Ki76] King, James C.: Symbolic execution and program testing. Communications of the ACM, 19(7):385–394, 1976.

[MS12] Maoz, Shahar; Sa'ar, Yaniv: Assume-Guarantee Scenarios: Semantics and Synthesis. In (France, R. B.; Kazmeier, J.; Breu, R.; Atkinson, C., eds): Model Driven Engineering Languages and Systems - 15th International Conference, MODELS 2012. Proceedings. volume 7590 of Lecture Notes in Computer Science. Springer, pp. 335–351, 2012.

Modularity and architecture of PLC-based software for automated production Systems: An analysis in industrial companies

Birgit Vogel-Heuser [1], Juliane Fischer, Stefan Feldmann, Sebastian Ulewicz, Susanne Rösch, Safa Bougouffa

Abstract: Adaptive and flexible production systems require modular and reusable software, especially considering their long-term life cycle of up to 50 years. We introduce a benchmark process – so-called SWMAT4aPS – to measure software maturity for industrial control software of automated production systems.

Keywords: Automated Production Systems, Maturity, Modularity, Control Software

1 Introduction to SWMAT4aPS benchmark process

Automated production systems (aPS) are long living systems that are exposed to changes over decades. Their complexity including automation hardware as well as system functionality realized by software is increasing. The ability to adapt flexibly to changing requirements by replacing or expanding cross-disciplinary modules, tracing of changes and management of software variants and versions are a prerequisite for intelligent, self-organizing Industry 4.0-compliant aPS. To evaluate whether industrial control software is qualified for Industry 4.0, the benchmark process SWMAT4aPS (Software Maturity for aPS) was developed. SWMAT4aPS consists of two elements, a self-assessment questionnaire and a detailed expert analysis for selected industrial companies. The core of the approach consists of four steps, which are performed in an experimentation and reporting phases. The first step is to conduct a survey with the developed questionnaire, which contains 45 questions grouped into three maturity categories. Maturity in design, maturity in test/quality assurance and maturity in start-up/operation/maintenance. In the second step the questionnaires' results of 16 German world-leading companies in machine and plant manufacturing are analysed (cp. Fig 1). The third step is the expert analysis in which four selected companies' software architecture, code structure and the workflow were analysed. We prove the validity of SWMAT4aPS by comparing the results of the questionnaire with the results of the expert analysis. The best companies reached overall maturity values of 0.86 %. We identified a huge variation of maturity levels in most of the companies, some have high values in design, others in start-up and operation. Refer to the full paper [VF17] for a complete discussion of results.

[1] Institute of Automation and Information Systems, Technische Universität München, Boltzmannstr. 15, 85748 Garching near Munich, Germany, email: {vogel-heuser, juliane.fischer, stefan.feldmann, sebastian.ulewicz, susanne.roesch,safa.bougouffa}@tum.de

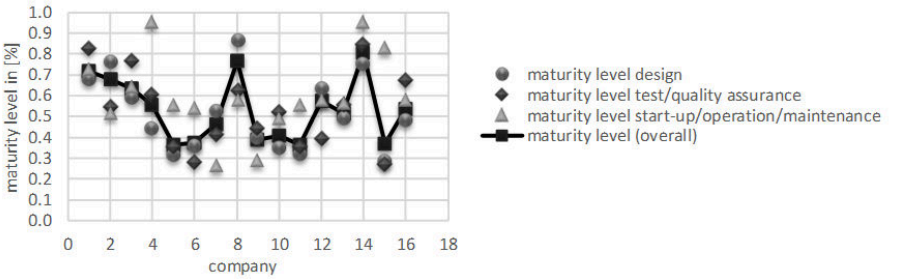


Fig. 1: Overview of maturity levels of companies 1 to 16 from first questionnaire showing the huge variance in the individual company regarding the three different maturity values

To further elaborate the differences between the wide range of machine and plant manufactures, to reveal modular dependency effects and to get deeper insight into the use of software module a second questionnaire was conducted with more than 68 companies. Results of design level of the industrial control software regarding factors for cross-disciplinary modularization are depicted (cp. Fig 2). The detailed analysis confirmed that the SWMAT4aPS approach delivers valid results and gives a first overview of the state of the art in software engineering in the respective company compared to others of the same branch. In a third questionnaire with more than 70 companies we strengthened the aspects of electrical engineering and included technical debt aspects and their reasons and impact.

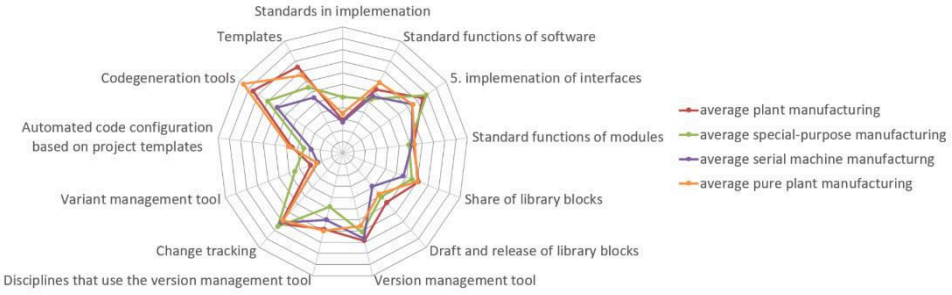


Fig. 2: Maturity level design of control software of 68 companies in machine and plant manufacturing from second questionnaire

2 References

[VF17] Vogel-Heuser, Birgit; Fischer, Juliane; Feldmann, Stefan; Ulewicz, Sebastian; Rösch, Susanne: Modularity and Architecture of PLC-based Software for Automated Production Systems: An analysis in industrial companies. The Journal of Systems & Software 2017, Elsevier, doi:10.101016/j.jss.2017.05.051.

SE Session 13 - Program editing and comprehension

M. Tichy, E. Bodden, M. Kuhrmann, S. Wagner, J.-P. Steghöfer (Hrsg.): SE 2018,
Lecture Notes in Informatics (LNI), Gesellschaft für Informatik, Bonn 2018 151

Neural Efficiency of Top-Down Program Comprehension

Norman Peitek[1] Janet Siegmund[2] Chris Parnin[3] Sven Apel[2] Johannes C. Hofmeister[2]
Christian Kästner[4] Andrew Begel[5] Anja Bethmann[1] André Brechmann[1]

Abstract: We observed program comprehension with functional magnetic resonance imaging (fMRI) and found a difference in neural efficiency between top-down and bottom-up comprehension, but failed to find a significant effect from beacons. Furthermore, we were able to replicate the results of a previous fMRI study, thereby strengthening the role of fMRI as measurement technique to observe program comprehension and other related cognitive processes.

Keywords: functional magnetic resonance imaging; program comprehension; neural efficiency

Program comprehension is an important cognitive process, because programmers spend most of their time understanding code [LVD06]. An efficient way to understand source code is top-down comprehension, where *beacons* and *plans* guide programmers to the relevant information [Br83]. When code lacks beacons and plans, or when programmers lack the experience to recognize them, a slow and tedious statement-by-statement process is necessary, which is called bottom-up comprehension [Pe87].

Understanding how programmers comprehend code is inherently difficult, because we cannot directly observe internal cognitive processes. In cognitive neuroscience, functional magnetic resonance imaging (fMRI) is being used to better understand such elusive cognitive processes. In our line of work, we use fMRI to infer neural processes involved in program comprehension based on observed brain activation of programmers in order to evaluate the often decades old models of program comprehension.

In our previous fMRI study on bottom-up comprehension, we found activation in brain areas related to working memory, divided attention, problem solving, and language processing [Si14]. In this follow-up study, we adapted the previously used material to isolate specific neural processes related to top-down comprehension [Si17].[6]

First, we could replicate the results of our first study on bottom-up comprehension. In the original study, we found activation in Brodmann areas (BAs) 6, 21, 40, 44, and 47

[1] Leibniz Institute for Neurobiology, Magdeburg, Germany, firstname.lastname@lin-magdeburg.de
[2] University of Passau, Passau, Germany, {janet.siegmund,apel,johannes.hofmeister}@uni-passau.de
[3] NC State University, Raleigh, North Carolina, USA, cjparnin@ncsu.edu
[4] Carnegie Mellon University, Pittsburgh, Pennsylvania, USA, kaestner@cs.cmu.edu
[5] Microsoft Research, Redmond, Washington, USA, andrew.begel@microsoft.com
[6] More information and material is available on the project's Web site `https://github.com/brains-on-code/paper-esec-fse-2017/`.

within the brain's left hemisphere, i.e. the speech hemisphere. We found part of these areas again (i.e., BAs 21, 40, 44), indicating the suitability of fMRI for measuring program comprehension. However, not all areas were significantly activated, which could be due to individual anatomical differences between the participant groups or the reduced statistical power of the current experiment.

Second, we found that top-down comprehension did not result in stronger activation than bottom-up comprehension, which at first sight is in contrast to program-comprehension models. However, we found a difference in the activation strength, such that the activation is significantly lower for top-down comprehension than for bottom-up comprehension. Thus, top-down comprehension has a higher neural efficiency than bottom-up comprehension.

Third, we could not find an effect of beacons on top-down comprehension, such that it did not matter whether beacons were in the source code or not. Based on the participants' comments and the data, we believe this effect is too small to be captured with the applied study framework.

In a nutshell, our results indicate that fMRI is a useful approach to better understand the cognitive processes of program comprehension. However, with this replication, we also found potential weaknesses of the experimental design, which we are currently addressing. Specifically, we are combining eye tracking with fMRI to help us map what participants are seeing to what the brain is doing.

References

[Br83] Brooks, R.: Towards a Theory of the Comprehension of Computer Programs. Int. J. Man-Machine Studies 18/6, pp. 543–554, 1983.

[LVD06] LaToza, T. D.; Venolia, G.; DeLine, R.: Maintaining Mental Models: A Study of Developer Work Habits. In: Proc. Int. Conf. Software Engineering (ICSE). ACM, Shanghai, China, pp. 492–501, 2006, ISBN: 1-59593-375-1.

[Pe87] Pennington, N.: Stimulus Structures and Mental Representations in Expert Comprehension of Computer Programs. Cognitive Psychologys 19/3, pp. 295–341, 1987.

[Si14] Siegmund, J.; Kästner, C.; Apel, S.; Parnin, C.; Bethmann, A.; Leich, T.; Saake, G.; Brechmann, A.: Understanding Understanding Source Code with Functional Magnetic Resonance Imaging. In: Proc. Int. Conf. Software Engineering (ICSE). IEEE, pp. 378–389, 2014.

[Si17] Siegmund, J.; Peitek, N.; Parnin, C.; Apel, S.; Hofmeister, J.; Kästner, C.; Begel, A.; Bethmann, A.; Brechmann, A.: Measuring Neural Efficiency of Program Comprehension. In: Proceedings of the 2017 11th Joint Meeting on Foundations of Software Engineering. ESEC/FSE 2017, ACM, Paderborn, Germany, pp. 140–150, 2017, ISBN: 978-1-4503-5105-8, URL: http://doi.acm.org/10.1145/3106237.3106268.

M. Tichy, E. Bodden, M. Kuhrmann, S. Wagner, J.-P. Steghöfer (Hrsg.): SE 2018,
Lecture Notes in Informatics (LNI), Gesellschaft für Informatik, Bonn 2018 153

Efficiency of Projectional Editing (Extended Abstract)

Thorsten Berger[1], Markus Voelter[2], Hans Peter Jensen[3], Taweesap Dangprasert[3], Janet Siegmund[4]

Abstract: Projectional editors are editors where a user's editing actions directly change the abstract syntax tree without using a parser. For programming, they promise essentially unrestricted language composition and flexible notations. For instance, graphical and textual domain-specific languages can be easily embedded into source code, avoiding intricate parser integration. Yet, despite these benefits, programming still mainly relies on editing textual code, where projectional editors imply a very different experience, often seen as the main adoption challenge. We describe an experiment [Be16] on the efficiency of code editing in a projectional editor conducted with industrial and student developers.

Keywords: projectional editing; language workbench; experiment

Projectional editor describes a type of editor where users work on a projection of a program's abstract syntax tree (AST) and directly change it with their editing gestures. This concept is different from parser-based editing, where users change the concrete syntax, and a parser then constructs the AST. Projectional editing, also known as structured editing or syntax-directed editing, is not a new idea; early references go back to the 1980s and include the Incremental Programming Environment, GANDALF, and the Synthesizer Generator. More contemporary incarnations are Intentional Programming, the Whole Platform, Más, Onion, and Jetbrains Meta Programming System (MPS). Most projectional editors are used in language workbenches—tools for developing and composing languages.

Projectional editors have two main advantages resulting from the absence of parsing. First, they support notations that cannot easily be parsed, such as tables, diagrams or mathematical formulas—each of which can be mixed with the others and with textual notations. Second, they support various ways of language composition, typically including modular language extension as well as embedding unrelated languages into a host language. Projectional editors can deal with mixed-language code while retaining awareness of the code structure (avoiding syntactic ambiguities), which is much harder to achieve with parser-based tools.

These benefits come at a cost. Even though, projectional editors support a wide range of non-textual notations, a significant share of any program, such as expressions and statements, will be expressed textually. For textual notations, however, projectional editors imply a very different—typically perceived as worse—editing experience compared to

[1] Chalmers | University of Gothenburg, Sweden, thorsten.berger@chalmers.se
[2] independent / itemis, Germany, voelter@acm.org
[3] ITU Copenhagen, Denmark, {hpgurre|taweesap}@gmail.com
[4] University of Passau, Germany, janet.siegmund@uni-passau.de

textual (parser-based) editors, often seen as the main challenge prohibiting their widespread adoption. The early projectional editors from the 1980s did very little to address this issue, ultimately limiting their adoption. Contemporary tools, such as MPS, have significantly improved usability, but inherited the bad reputation.

In our main publication [Be16], we present an experiment of code-editing activities in a projectional editor, conducted with 19 graduate computer-science students and industrial developers. We investigate the effects of projectional editing on editing efficiency, editing strategies, and types and frequencies of errors made—each of which we also compare to conventional, parser-based editing. We design the experiment based on a survey [Vo14] we conducted with industrial developers familiar with projectional editing. The instrument for our experiment is JetBrains MPS, since (i) it is the most widely used projectional editor today, (ii) it improved significantly over the tools from the 1980s, and (iii) it is open-source software, which fosters the replicability of our results.

Our results show that efficiency with projectional editing can be quickly achieved for basic code-editing activities. More experience does not lead to significantly better results. In contrast, advanced editing (e.g., larger code modifications or refactorings) requires significantly more experience and understanding of the underlying concepts (in particular, the AST structure). Then, however, experienced developers can outperform beginners with a projectional editor and even the participants using the parser-based editor. The projectional editor also fosters fewer errors (mistakes) and different editing strategies (e.g., increased use of operations that work well on ASTs).

Based on our results, we conceived techniques to further improve the editing experience. We presented Grammar Cells [Vo16], which support creating consistent language-editing experiences, with a focus on supporting the editing of very hierarchical program constructs, such as expressions. Grammar cells create a consistent editing experience that increasingly resembles linear (textual) editing, further reducing the need for understanding the underlying AST. In future work we plan to systematically study the benefits of arbitrary language composition (i.e., language embedding) and flexible notations (graphical and textual).

Literatur

[Be16] Berger, T.; Voelter, M.; Jensen, H. P.; Dangprasert, T.; Siegmund, J.: Efficiency of Projectional Editing: A Controlled Experiment. In: 24th ACM SIGSOFT International Symposium on the Foundations of Software Engineering (FSE). 2016.

[Vo14] Voelter, M.; Siegmund, J.; Berger, T.; Kolb, B.: Towards User-Friendly Projectional Editors. In: 7th International Conference on Software Language Engineering (SLE). 2014.

[Vo16] Voelter, M.; Szabo, T.; Lisson, S.; Kolb, B.; Erdweg, S.; Berger, T.: Efficient Development of Consistent Projectional Editors using Grammar Cells. In: 9th International Conference on Software Language Engineering (SLE). 2016.

M. Tichy, E. Bodden, M. Kuhrmann, S. Wagner, J.-P. Steghöfer (Hrsg.): SE 2018,
Lecture Notes in Informatics (LNI), Gesellschaft für Informatik, Bonn 2018 155

Projectional Editing of Software Product Lines (Extended Abstract)

Benjamin Behringer[1], Jochen Palz[2], Thorsten Berger[3]

Abstract: A software product line is a portfolio of software variants. To implement its common and variable features, a variety of techniques emerged, representing the feature implementation either as annotated code or as dedicated feature modules. While each technique provides distinct advantages, developers need to choose one feature representation and then adhere to it when the product line evolves. In this extended abstract, we describe PEoPL, an approach [BPB17] that combines the advantages of different feature representations. Developers can engineer features using the most-suited representation, switch it seamlessly, and even use different representations (editable views) in parallel.

Keywords: projectional editing; software product lines; variability mechanisms

A software product line (SPL) is a portfolio of system variants engineered in an application domain. It implements common and variable features using implementation techniques called *variability mechanisms*. Many such mechanisms have emerged, typically classified into annotative (e.g., preprocessors) and modular (e.g., feature modules) mechanisms. Each represents a feature's artifacts differently, with distinct advantages and disadvantages. For instance, annotative mechanisms are easy to apply, but annotations clutter source code, obscure control flows, and force developers to work on all variants in parallel. They also lead to scattered and tangled feature implementations, challenging program comprehension, maintenance, and evolution. In contrast, feature modules ease the latter by realizing features modularly in cohesive units. Unfortunately, creating feature modules imposes substantial engineering overhead, while their interaction with other features is more difficult to comprehend, both hindering their adoption in practice.

Although these representations are complementary, existing SPL engineering approaches typically focus on one representation. Most importantly, developers need to choose one representation per feature and to adhere to it for maintenance and evolution. While refactorings were proposed for migrating between annotative and modular representations, they do not allow to quickly switch the representation during evolution and maintenance. Ideally, developers could always select the best-suited representation.

[1] htw saar, Germany, benjamin.behringer@htwsaar.de
[2] htw saar, Germany, jochen.palz@htwsaar.de
[3] Chalmers | University of Gothenburg, Sweden, thorsten.berger@chalmers.se

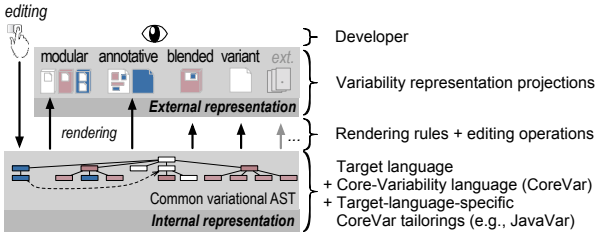


Abb. 1: PEoPL separates internal and external variability representations

In our main publication [BPB17], we present PEoPL, an approach that combines the benefits of annotative and modular variability mechanisms. PEoPL allows developers to quickly switch representations and even use different representations side-by-side. The core idea of PEoPL is to establish an internal representation of the SPL and separate it from the external representations that developers use. Fig. 1 illustrates the approach. Internally, PEoPL persists an abstract syntax tree (AST) adhering to a programming language composed with our languages CoreVar (to internally represent feature artifacts) and a programming-language-specific tailoring of CoreVar (to realize well-formedness constraints). We currently provide such tailorings for Java, C, and fault trees. Externally, the AST is rendered into editable projections used by developers. According to the projectional-editing paradigm, the editing gestures of developers directly change the AST, without any involvement of parsing [Be16]. The AST is rendered into concrete syntax using projections. We currently provide projections showing artifacts as *textual annotations* (e.g., #ifdef), *visual annotations* (colored bars), *feature modules*, *annotations blended into modules*, *fade-in modules* (i.e., show external code within a module), *individual variants* (i.e., hide non-selected features), and *reused code snippets* (e.g., to support cross-cutting aspects) [Be17]. We realize PEoPL's concepts in a full-fledged IDE based on JetBrains Meta Programming System (MPS).

We evaluate PEoPL by adopting and implementing eight Java-based SPLs, such as the Berkeley DB (70kLOC, 42 features, 218 classes). Our evaluation shows the feasibility of the approach, especially that internal and external feature representations can be separated, and that it scales. A preliminary user study confirms the benefits of our external representations and of seamlessly switching them. Controlled experiments are subject to our future work.

Literaturverzeichnis

[Be16] Berger, Thorsten; Völter, Markus; Jensen, Hans Peter; Dangprasert, Taweesap; Siegmund, Janet: Efficiency of Projectional Editing: A Controlled Experiment. In: 24th ACM SIGSOFT International Symposium on the Foundations of Software Engineering (FSE). 2016.

[Be17] Behringer, Benjamin: Projectional Editing of Software Product Lines—The PEoPL Approach. Dissertation, University of Luxembourg, 2017.

[BPB17] Behringer, Benjamin; Palz, Jochen; Berger, Thorsten: PEoPL: Projectional Editing of Software Product Lines. In: 30th International Conference on Software Engineering (ICSE). 2017.

SE Session 14 - Software Architecture

Automated Analysis of the Co-evolution of Software Systems and Business Processes

Kiana Busch[1] Robert Heinrich[1] Axel Busch[1] Ralf Reussner[1]

Abstract: Software systems are an essential part of business processes. As business processes and the corresponding software systems mutually affect each other, they co-evolve during their life cycle. Thus, to adequately predict the impact of a change, their mutual dependencies have to be considered. However, existing approaches to change propagation analysis consider one domain in isolation and neglect the mutual dependencies between the domains. In this paper, we propose the Karlsruhe Architectural Maintainability Prediction for Business Processes (KAMP4BP) to analyze the change propagation in business processes and the corresponding software systems.

This is an extended abstract of the paper *Architecture-based Change Impact Analysis in Information Systems and Business Processes* published in *ICSA'17* proceedings [Ro17].

Keywords: Software System; Business Process; Evolution; Change Impact

1 Karlsruhe Architectural Maintainability Prediction for Business Processes (KAMP4BP)

Software systems are increasingly used in organisations to support their business processes. The business process can be considered as a set of actor steps (i.e., performed totally by humans involved in the business process) and system steps (i.e., performed totally by the software system) [He15]. Thus, there are mutual dependencies between business processes and the corresponding software systems, which result in co-evolution of both domains. Therefore, we cannot consider only one domain in isolation when analyzing the effects of a change. However, the mutual dependencies between both domains are one of the main challenges during the change propagation analysis, as a change in one of the domains can propagate to the other domain.

KAMP4BP [Ro17] calculates the change propagation in the business process or the corresponding software system based on an initial change request. It is a scenario-based approach with focus on mutual dependencies between both domains. Fig. 1 gives an overview of KAMP4BP: i) In the preparation phase, the architecture of the software system and the corresponding business process design have to be modeled. In order to model the software architecture, the software architect uses the Palladio Component Model (PCM) [Re16]

[1] Karlsruhe Institute of Technology, {kiana.busch, robert.heinrich, axel.busch, ralf.reussner}@kit.edu

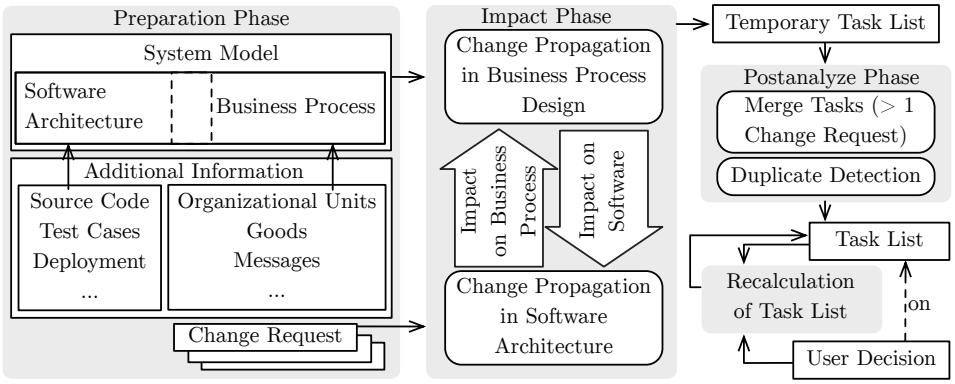


Fig. 1: Overview of the approach [Ro17]

which is a metamodel describing the software architecture with focus on software quality attributes. The business process designer models the business process using the extensions of the usage profile of PCM metamodels which allows modelling the business process as a set of actor steps and system steps [He15]. Further, the model of the software architecture and the business process design can be annotated with additional information such as test cases in software systems or organizational units in business processes. Annotating further information allows more realistic change impact analysis, as KAMP4BP can calculate the change impact on these artifacts as well. Additionally, the initial change requests have to be marked using the corresponding metamodels. ii) In the impact phase, KAMP4BP automatically calculates the change propagation in business processes, software systems, and between business processes and software systems. To this end, the analysis is based on predefined change propagation rules. The output of this step is a temporary task list. Each task references a changing element in the model of the software architecture or the business process design. iii) If more than one initial change request were marked in the preparation phase, KAMP4BP merges the corresponding task lists of the change requests and eliminates the duplicates. Further, the user of KAMP4BP may need to manually exclude some tasks referring to model elements that cannot be affected by the change. As changing these model elements could lead to further changes in other model elements, KAMP4BP considers the users' decisions and recalculates the task list.

References

[He15] Heinrich, Robert et al.: Integrating business process simulation and information system simulation for performance prediction. SoSyM Journal, pp. 1–21, 2015.

[Re16] Reussner, Ralf et al.: Modeling and Simulating Software Architectures – The Palladio Approach. MIT Press, 2016.

[Ro17] Rostami, K. et al.: Architecture-Based Change Impact Analysis in Information Systems and Business Processes. In: IEEE ICSA. pp. 179–188, 2017.

M. Tichy, E. Bodden, M. Kuhrmann, S. Wagner, J.-P. Steghöfer (Hrsg.): SE 2018,
Lecture Notes in Informatics (LNI), Gesellschaft für Informatik, Bonn 2018 161

Using Architecture Knowledge to Improve Automated Software Architecture Design Space Exploration

Axel Busch[1] Anne Koziolek[1]

Abstract: During the development of modern software systems software architects make more and more trade-off decisions between many quality attributes. Quality attributes such as performance, reliability, but also others such as security and usability are important aspects in software projects. While the former are comparatively easy to quantify, this is for the latter more difficult or costly. However, existing approaches usually consider either only quantified or only qualitatively modelled quality attributes. With existing approaches, it is not possible to model, analyse, and optimize a subset of quantified attributes and another subset of qualitatively modeled quality attributes together. In this paper, we extend PerOpteryx, an approach to optimize software architectures by meta models and mechanisms for the simultaneous analysis and optimization of quantitatively and qualitatively modelled quality attributes. This makes it possible to include even estimated values for individual quality dimensions in systematic optimization processes.

This is an extended abstract of the paper *Considering Not-quantified Quality Attributes in an Automated Design Space Exploration* published in *QoSA'16* proceedings [BK16].

Keywords: Architecture knowledge; Design Decision; Architecture Trade-off; Automated

1 Analysing Quantified and Not-Quantified Quality Attributes

In modern design processes for software systems, often trade-off decisions have to be made between a variety of quality attributes such as performance, security or reliability. Design decisions regarding the software architecture thereby play a critical role. The quality of the software architecture has proved to be one of the decisive factors influencing the overall quality of the software system [Ko12].

Different approaches, such as Palladio and PerOpteryx, have been shown as promising approaches to predict [Re16] and optimize [Ma10] quality attributes based on a software architecture design. However, these and other approaches can only analyse either quantified or purely qualitatively modeled quality attributes that represents the informal architecture knowledge of the software architect. Approaches that enable the analysis of qualitatively modeled quality attributes are often completely or at least partially automatic processes. However, approaches that only require quantified quality attributes are often not applicable due to the high costs of the quantification.

[1] Karlsruhe Institute of Technology, {axel.busch, anne.koziolek}@kit.edu

Our approach combines both approaches and allows the analysis of qualitative modeled attributes together with quantified analysis functions. Quality attributes can thus always be analysed and optimized quantitatively whenever a quantitative objective function is available. If no function is available or practicable, values can be modeled qualitatively. This enables software architects to decide individually for each quality attribute whether a complex quantification step should be carried out or whether a qualitative assessment would be sufficient. Although estimated values are usually less accurate than measured values or values obtained by quantitative objective functions, they still model important architecture reasoning of the software architect. Furthermore, architecture reasoning is often the only resource when quantitative methods are not available or too costly to perform.

In order to implement our approach, we first extend the Quality Modeling Language (QML) used in PerOpteryx. QML allows the specification of quality attributes, quality dimensions and quality requirements. We extend QML so that values can be modeled on different scale levels, so that we can model ordinal relations, nominal relations and values within a certain range of values as possible manifestations of a dimension. In addition, we extend the meta model of PerOpteryx with elements for the annotation of software components with quality attributes. This allows the annotation of (estimated or measured) quality values on components for arbitrary quality attributes. Finally, we extend the exploration mechanism of PerOpteryx so that not-quantified quality attributes can be optimized together with already existing quantified quality attributes. For this purpose, we consider the quality values of individual components as separate quality dimensions. If an order is defined within a dimension on all possible values, this dimension can be used as objective.

For our evaluation, we consider two case studies. The results of our evaluation show how our approach can be used to include the architecture reasoning of the software architect in a simple manner. One of the results demonstrate how the software architect can analyse the impact of an architecture decision on other quality attributes with low overhead. The evaluation demonstrates that our approach can also be used to analyse the effects of architecture decisions on quantitatively determined quality attributes, even if the architecture decision actually targets at improving a not-quantified quality attribute of the system.

References

[BK16] Busch, Axel; Koziolek, Anne: Considering Not-quantified Quality Attributes in an Automated Design Space Exploration. In: Proceedings of the 12th International ACM SIGSOFT Conference on the Quality of Software Architectures. QoSA'16. IEEE, pp. 50–59, 2016.

[Ko12] Koziolek, Anne: Architecture-Driven Quality Requirements Prioritization. In: TwinPeaks. IEEE CS, 2012.

[Ma10] Martens, Anne; Koziolek, Heiko; Becker, Steffen; Reussner, Ralf H.: Automatically Improve SW Models for Perform., Reliab. and Cost Using Gen. Algor. WOSP/SIPEW ICPE, 2010.

[Re16] Reussner, Ralf H.; Becker, Steffen; Happe, Jens; Heinrich, Robert; Koziolek, Anne; Koziolek, Heiko; Kramer, Max; Krogmann, Klaus: Modeling and Simulating Software Architectures: The Palladio Approach. The MIT Press, 2016.

SWM Session 1 - Wissenschaftliches Programm

M. Tichy, E. Bodden, M. Kuhrmann, S. Wagner, J.-P. Steghöfer (Hrsg.): SE 2018,
Lecture Notes in Informatics (LNI), Gesellschaft für Informatik, Bonn 2018 165

Agile Software Quality Function Deployment

Sixten Schockert[1,2] und Georg Herzwurm[1]

Abstract: Dieser Beitrag stellt das Agile Software Quality Function Deployment (QFD) als Methode des agilen Requirements Engineering (RE) vor. Es basiert auf 27 Gestaltungsanforderungen, die aus den Prinzipien und Werten der agilen Softwareentwicklung sowie dem praktischen Umgang mit Anforderungen in agilen Entwicklungsmodellen abgeleitet sind. Das Agile Software QFD ist gekennzeichnet durch die konsequente Ausrichtung an den wichtigsten Stakeholderbedürfnissen, der Suche nach alternativen und besseren Lösungen sowie der engen Zusammenarbeit mit Kunden/Nutzern. Agiles Software QFD ist damit Ausdruck eines am Business Value orientierten, gestaltenden Requirements Engineering.

Diese Zusammenfassung bezieht sich auf den Beitrag „Agile Software Quality Function Deployment“ [SH17], der im Rahmen des 23rd International Symposium on Quality Function Deployment (ISQFD'17) in Tokio präsentiert wurde.

Keywords: Agile Requirements Engineering, Software QFD, User Stories, Priorisierung

1 Agile Software QFD als Methode des agilen RE

User Stories repräsentieren das wesentliche Artefakt der Kommunikation von Anforderungen in einer agilen Entwicklung. Und unabhängig davon, ob sie sich als präzise Anforderungen für die Entwickler eignen, auf Basis der User Stories im Product Backlog wird entschieden, was in der nächsten Iteration umgesetzt wird und was nicht. Von daher muss ein Agiles Requirements Engineering (RE) Wege aufzeigen, gute User Stories zu finden, zu entwerfen und die gemäß Business Value vielversprechendsten für die Implementierung in der nächsten Iteration auszuwählen. Das ist entscheidend für eine nicht nur effiziente, sondern auch effektive agile Entwicklung, die an den wichtigsten Anforderungen ansetzt und nicht „nur“ plausible User Stories zügig umsetzt.

Dieser Beitrag stellt dazu das Agile Software QFD [Sc17] vor. Es repräsentiert eine Adaption auf das agile Entwicklungsumfeld und damit evolutionäre Weiterentwicklung des sog. Software QFD, einer Variante der Qualitätsmethode Quality Function Deployment (QFD) zur Entwicklung von Softwareprodukten. Vor allem zur Fokussierung der Entwicklungsaktivitäten in den frühen Phasen der Softwareentwicklung hat sich der Einsatz von Software QFD bewährt. Und genau diese Bewertung und Priorisierung von Anforderungen bzw. User Stories ist zwingender Bestandteil jeder agilen Softwareentwicklung in Inkrementen.

[1] Universität Stuttgart, Lehrstuhl für ABWL und Wirtschaftsinformatik II (Unternehmenssoftware), Keplerstraße 17, D-70174 Stuttgart, {schockert, herzwurm} @wius.bwi.uni-stuttgart.de
[2] Fachhochschule für Ökonomie und Management (FOM), Rotebühlstraße 121, D-70178 Stuttgart

Die QFD-Haupttätigkeiten der Gewinnung, Abstimmung und Umsetzung von Anforderungen betten sich beim Agilen Software QFD nahtlos in den agilen Iterationszyklus und die 3C's der User Stories ein. So gibt es in jeder Iteration eine Pflege des Backlogs im Sinne der Gewinnung von Anforderungen (Card), darauf aufbauend eine Abstimmung bzgl. der umzusetzenden Anforderungen in der nächsten Iteration im ersten Planungsmeeting (Conversation) und eine mit dem zweiten Planungsmeeting startende Umsetzung von Anforderungen (Confirmation).

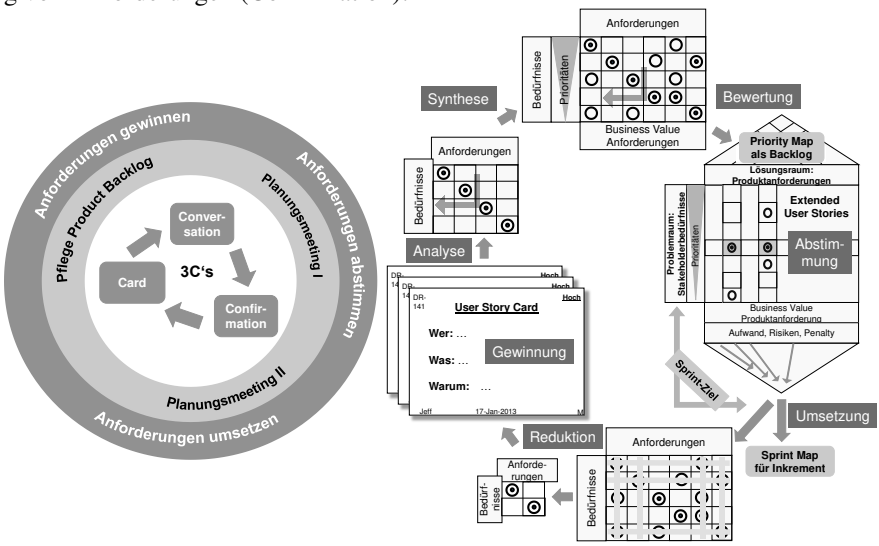


Abbildung 1: Iterationszyklus des Agilen Software QFD [Sc17]

Zur praktischen Umsetzung des zyklischen Ablaufs besitzt das Agile Software QFD besondere methodische Merkmale wie die inkrementell wachsende und reduzierende Priorisierungsmatrix und die Priority Map. Letztere repräsentiert den Product Backlog und dabei sowohl die Spaltung (Analyse) als auch die Verbindung (Synthese) von Bedürfnissen der Stakeholder mit möglichen Lösungen in Extended User Stories. Dabei werden die Anforderungen konsequent auf Basis der gemäß Business Value wichtigsten Bedürfnisse bewertet und für das nächste Inkrement zur Erreichung des Sprint-Ziels ausgewählt (Sprint Map).

Literaturverzeichnis

[Sc17] Schockert, S.: Agiles Software Quality Function Deployment. Lohmar – Köln, Eul Verlag, Dissertation Universität Stuttgart, 2017.

[SH17] Schockert, S., Herzwurm, G.: Agile Software Quality Function Deployment. In: JUSE & ICQFD (Hrsg.): Proceedings of the 23rd International Symposium on Quality Function Deployment (ISQFD'17), Tokio 7./8.9.2017, S. 134-148, 2017.

M. Tichy, E. Bodden, M. Kuhrmann, S. Wagner, J.-P. Steghöfer (Hrsg.): SE 2018,
Lecture Notes in Informatics (LNI), Gesellschaft für Informatik, Bonn 2018 167

Das Business Model House of Quality

Bewertung plattformbasierter Geschäftsmodelle mit Quality Function Deployment

Felix Schönhofen[1], Sixten Schockert[1] und Georg Herzwurm[1]

Abstract: Aufgrund ständiger Veränderungen der Märkte stehen Unternehmen in der IT-Branche vor der dauernden Herausforderung, das eigene Geschäftsmodell an die sich ändernden Gegebenheiten anzupassen um den langfristigen Erfolg zu sichern. Aktuelle Trends wie die Sharing Economy und die wachsende Bedeutung der Plattformökonomie erhöhen zusätzlich die Komplexität der Problemstellung. Voraussetzung für eine erfolgreiche Anpassung ist eine umfassende Bewertung der Leistungsfähigkeit des eigenen Geschäftsmodells. Dieser Beitrag zeigt eine Möglichkeit, wie eine solche Bewertung mithilfe von Quality Function Deployment möglich ist.

Diese Zusammenfassung bezieht sich auf den Beitrag „Das Business Model House of Quality – Bewertung plattformbasierter Geschäftsmodelle mit Quality Function Deployment [ScS17], der im Rahmen der 13. Internationalen Tagung Wirtschaftsinformatik (WI 2017) präsentiert wurde.

Keywords: Business Model, Platform Ecosystem, Quality Function Deployment

1 Motivation

Im Zuge aktueller Trends wie der Digitalisierung und der Sharing Economy haben mehrseitige Plattformen an Bedeutung gewonnen. Das Wertversprechen liegt hier in der reinen Vermittlung von Sach- und Dienstleistungen [AG14]. Diese plattformbasierten Geschäftsmodelle erfordern die Berücksichtigung einer oft großen Zahl an unterschiedlichen Komplementoren, die ihre Leistungen auf der Plattform des Anbieters einbringen. Diese stellen neben den Endkunden eine weitere wichtige Stakeholdergruppe dar, deren spezifischen Bedürfnisse bei der (Weiter-)entwicklung des Geschäftsmodells beachtet werden müssen. In diesem Beitrag soll, basierend auf der Qualitätsmethode Quality Function Deployment (QFD), ein Vorgehensmodell zur Bewertung plattformbasierter Geschäftsmodelle vorgestellt werden. QFD eignet sich dafür in besonderem Maße, da es überall dort anwendbar ist, wo in einem Team Lösungen zu Problemen gesucht werden. Hierzu trennt QFD analytisch konsequent zwischen den Bedürfnissen der Stakeholder einerseits und konkreten Anforderungen an eine mögliche Lösung andererseits [Sc17, ScS17].

[1] Universität Stuttgart, Lehrstuhl für ABWL und Wirtschaftsinformatik II, 70174 Stuttgart,
{schoenhofen, schockert, herzwurm}@wius.bwi.uni-stuttgart.de

2 Vorgehensmodell

Im Rahmen von Workshops werden, basierend auf den Einträgen einer im Team ausgefüllten Business Model Canvas (BMC) [OP11] Bedürfnisse der verschiedenen Stakeholder identifiziert und anschließend priorisiert. Die Geschäftsmodellmerkmale aus der BMC werden dann anhand der priorisierten Stakeholderbedürfnisse in einer Matrix auf ihre Fähigkeit diese Bedürfnisse zu erfüllen, hin bewertet. Dadurch kann das Verbesserungspotential bzgl. der unterschiedlichen Stakeholder der Plattform identifiziert werden.

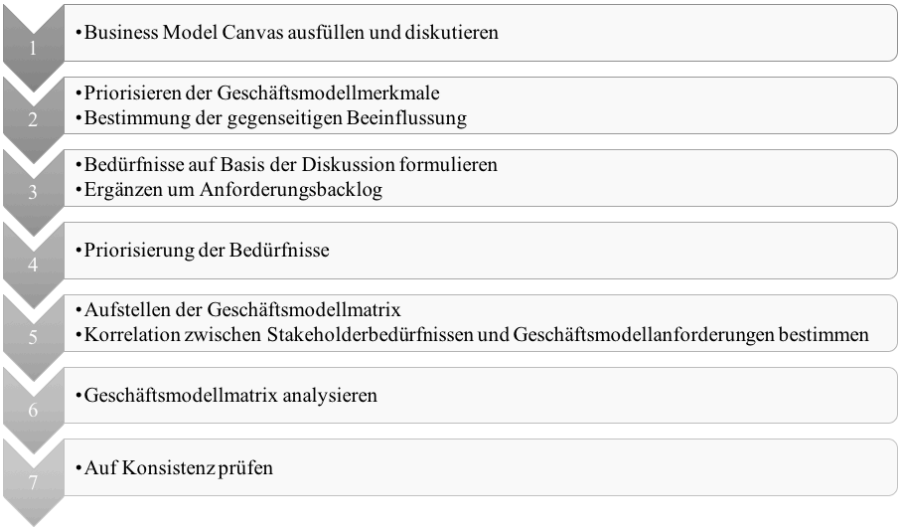


Abbildung 1: Vorgehensmodell zur Bewertung plattformbasierter Geschäftsmodelle [ScS17]

Literaturverzeichnis

[Ga14] Gawer, A.: Bridging differing Perspectives on technological Platforms: Toward an integrative framework. In: Research policy, vol. 43, pp. 1239-1249. Elsevier, Oxford 2014.

[Sc17] Schockert, S.: Agiles Software Quality Function Deployment. Eul, Lohmar und Köln 2017.

[ScS17] Schönhofen, F., Schockert, S.: Das Business Model House of Quality: Bewertung plattformbasierter Geschäftsmodelle mit Quality Function Deployment. In: Proceedings der 13. Internationalen Tagung Wirtschaftsinformatik (WI 2017), pp. 1477-1488. St. Gallen, 2017.

[OP11] Osterwalder, A., Pigneur, Y.: Business Model Generation – Ein Handbuch für Spieleveränderer und Herausforderer. Campus, 2011.

M. Tichy, E. Bodden, M. Kuhrmann, S. Wagner, J.-P. Steghöfer (Hrsg.): SE 2018,
Lecture Notes in Informatics (LNI), Gesellschaft für Informatik, Bonn 2018 169

The QDAcity-RE Method for Structural Domain Modeling Using Qualitative Data Analysis

Andreas Kaufmann[1], Dirk Riehle[2]

Abstract: The creation of domain models from qualitative input relies heavily on experience. An uncodified ad-hoc modeling process is still common and leads to poor documentation of the analysis. In this article we present a new method for domain analysis based on qualitative data analysis (QDA). The method helps identify inconsistencies, ensures a high degree of completeness, and inherently provides traceability from analysis results back to stakeholder input. These traces do not have to be documented after the fact. We evaluate our approach using four exploratory studies.

Keywords: domain modeling; domain model; requirements engineering; requirements elicitation; qualitative data analysis

The full paper of this extended Abstract has been published in [KR17]

1 Motivation

The quality of a requirements specification mainly depends on the experience of the analyst and his or her understanding of the problem domain. To establish a good understanding of the problem domain, the analyst may create a domain model as part of his or her analysis.

Domain models must correctly represent the reality of the domain and be easy to understand from the stakeholder's perspective.

Qualitative exploratory research faces similar challenges. The area under study is often highly complex and the gathered data is frequently unstructured, inconsistent, and incomplete. In scientific research, these challenges are addressed by using methods for *qualitative data analysis (QDA)*. QDA methods focus on extracting the relevant information from qualitative data, interpreting the data, and abstracting from it. QDA is employed in theory building research to study a wide range of phenomena through the gathering and interpretation of qualitative data. The process ensures thorough documentation of the analysis process.

We equate the process of theory building to the domain analysis process, and propose a method for domain modeling, called QDAcity-RE, based on principles of QDA.

[1] Friedrich-Alexander University Erlangen-Nürnberg, Open Source Research Group, Martenstr. 3, 91059 Erlangen, Germany andreas.kaufmann@fau.de
[2] Friedrich-Alexander University Erlangen-Nürnberg, Open Source Research Group, Martenstr. 3, 91059 Erlangen, Germany dirk@riehle.org

2 Method

In our method, requirements engineers sample stakeholders, interview them, correlate other materials, and perform QDA of the materials to derive a so-called *code system*. The code system is then extended to derive the relevant requirements engineering results.

A model of our method is outlined in figure 1.

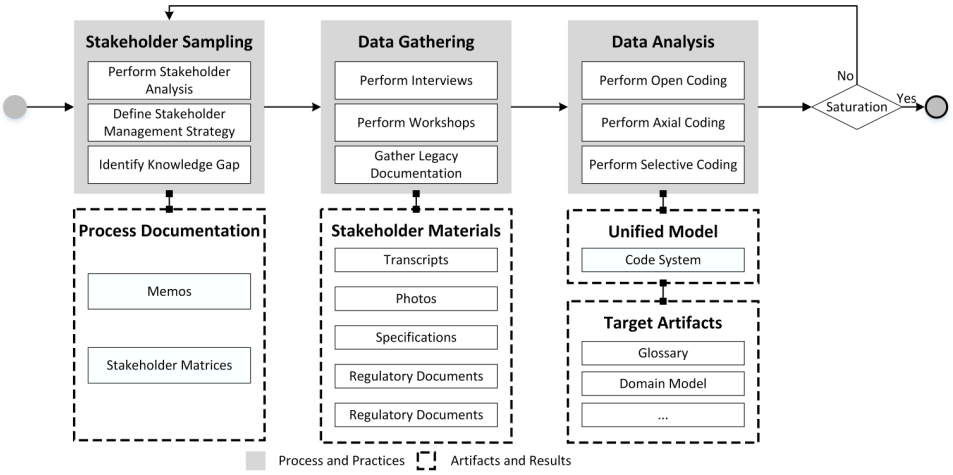


Fig. 1: The QDAcity-RE Process for Structural Domain Modeling

3 Evaluation

We evaluated our method in four studies in the fields of human resource development, medical imaging diagnostics, railway systems and qualitative research.

As data gathering technique we relied mostly on expert interviews, which we triangulated with existing documentation, workshop artifacts and norms and regulations.

We also evlaluated the derivation of different artifact types such as feature models, conceptual models, domain specific languages and software requirements specifications from our unified model, the code sytem.

References

[KR17] Kaufmann, A.; Riehle, D.: The QDAcity-RE method for structural domain modeling using qualitative data analysis. Requirements Engineering/, Nov. 2017, ISSN: 1432-010X, URL: https://doi.org/10.1007/s00766-017-0284-8.

SWM Session 2 - Fallstudien in der Industrie

M. Tichy, E. Bodden, M. Kuhrmann, S. Wagner, J.-P. Steghöfer (Hrsg.): SE 2018,
Lecture Notes in Informatics (LNI), Gesellschaft für Informatik, Bonn 2018 173

Innovationsschub – Erfahrungen am Fallbeispiel IT Products

Andreas Rösel[1]

Abstract: Das IT Umfeld und die Herausforderungen haben sich verändert. Die Erwartungshaltung der Nutzer hat sich verändert. Das zeigt sich in Bereichen wie Nutzerfreundlichkeit, sowie Verfügbarkeit der Anwendung auf mobilen Geräten. Dies ist für die typische innerbetriebliche Software eine große Herausforderung, die zu meistern einen Kultur"shift" erfordert. Einen Schub hin zu einer neuen, einer stärkeren Innovationskultur. Das drückt sich schlussendlich in einer höheren Geschwindigkeit von Innovationszyklen aus und ist oft mit ein einer feineren Granularität von Liefereinheiten verbunden. Wie kommen aber die vielen möglichen Elemente wie agile Methoden, Design-Thinking, Intrapreneurship und so weiter in einem Unternehmen tatsächlich zur Umsetzung und wie spielen sie zusammen? Diese Aspekte werden in einem aktuellen Fallbeispiel aus einem IT Unternehmen betrachtet. Dabei beleuchten wir insbesondere die Verbindung von Abläufen und firmeninterner Innovationskultur, einer Herausforderung der sich viele Unternehmen heute gegenübersehen.

Keywords: Business Canvases; Agile Methoden; Innovationsmethoden; Startup-Kultur; Intrapreneurship; Change Management

1 Introduction

In den letzten Jahren hat sich das IT Umfeld weiterentwickelt und verändert. Cloud Computing und das Internet-der-Dinge begegnen uns in den Medien, in der Arbeitswelt und im persönlichen Alltag. Diese und weitere Themen, wie Computing Everywhere wurden als IT-Topthemen für 2015 genannt [Ga14]. Gerade durch die Verfügbarkeit und Mobilität von Rechnern in Form von Smartphones sowie Tablets und die tägliche Anbindung an das Internet haben sich die Erfahrungen der Endnutzer geändert [st17] und damit auch die Erwartungen, die sie an IT Produkte und Dienstleistungen haben. Dieser Erwartungsdruck muss unserer Erfahrung nach zu rapiden Veränderung auf der IT Anbieterseite führen, den wir hier als Innovationsschub betrachten. Im Folgenden beleuchten wir anhand einiger Beispiele wie sich dies für die Bereitstellung von IT Produkten innerhalb eines Unternehmens auswirkt. In unserem Kontext werden Business Systeme, sowie Endgeräte und Anwendungen für mehrere zehntausend Endanwender im Unternehmen bereitgestellt. Hunderte von Entwicklern sind eingebunden, um Standard IT Produkte für die weltweite Nutzung einzuführen, zu betreiben und nach Nutzen und Priorisierung spezifisch anzupassen. Da das Unternehmen selbst Software-Produkte erstellt, kommt der internen IT dabei eine Vorreiterrolle als Erstanwender

[1] SAP SE, IT Process Office, Dietmar-Hopp-Allee 16, 69190 Walldorf, Germany andreas.roesel@sap.com

zu mit hohen Erwartungen an Innovationsbereitschaft. Das beinhaltet Innovationen zur Vereinfachung der internen Geschäftsprozesse mit den jeweiligen Geschäftsbereichen sowie Ko-innovation mit dem Entwicklungsbereich. Dies drückt sich unter anderem in einer höheren Geschwindigkeit von Innovationszyklen aus und ist oft mit einer feineren Granularität von Liefereinheiten verbunden.

Als nächstes wollen wir untersuchen welche Prinzipien und methodischen Ansätze als Grundlage einer Innovationskultur im hier betrachteten Unternehmen[2] genutzt werden.

2 Prinzipien und Methodische Ansätze einer Innovationskultur

Wir sammeln und kategorisieren welche Prinzipien und methodischen Ansätze als Grundlage einer Innovationskultur[3] im Unternehmen genutzt werden.

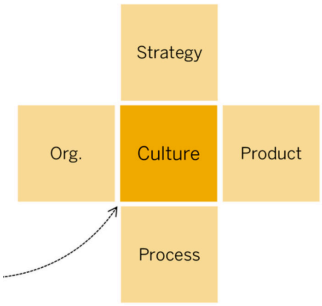


Abb. 1: Einordnung der Innovationskultur in 5 Dimensionen der Vereinfachung/Innovation

Für jeden Punkt gibt es eine Beschreibung, eine Zuordnung zu den Dimensionen (Strategie, Produkt, Prozess), eine Einordnung, ob der Punkt auf Unternehmensebene und/oder im IT-Bereich (Dimension Organisation) eingesetzt wird, sowie Hinweise zur Nutzung.

2.1 Verhaltensprinzipien – *How we run*

Die Verhaltensprinzipien "How we run" wurden im Unternehmen in 2015 eingeführt, um die Unternehmenskultur für die strategischen Ziele Innovation und Vereinfachung greifbar zu machen (Verknüpfung zur Dimension Strategie). Die sechs Prinzipien wurden mit Beteiligung der Mitarbeiter erarbeitet und sind für alle Bereiche im Unternehmen gültig: *Keep the promise, Tell it like it is, Stay curious, Build bridges – not silos, embrace differences*.

[2] Mit dem Begriff *Unternehmen* ohne weitere Qualifizierung beziehen wir uns im Weiteren immer auf das für diese Fallstudie betrachtete Unternehmen.

[3] Die 5 Dimensionen der Vereinfachung/Innovation sind aus der Dokumentation im Mitarbeiter Portal des Unternehmens unter der Rubrik "Strategy/Simplify" entnommen.

How-We-Run Prinzip	Fragestellung
Keep the Promise/ *Halte das Versprechen*	Halte ich mich an die Zusage dem Kunden Mehrwert zu liefern?
Tell it like it is/ Sage es wie es ist	Kann ich meine Sichtweise vereinfachen? Habe ich eine Lösung angeboten und nicht nur meine Meinung?
Stay curious/ *Bleibe neugierig*	Kann ich die positiven Aspekte in dieser Veränderung sehen? Fühle ich mich sicher etwas Neues/Anderes auszuprobieren?
Build bridges, not silos/ *Baue Brücken, keine Silos*	Kann ich durch einen Kollegen außerhalb meines Kernteams oder meines Bereichs eine neue Perspektive bekommen? Baue ich in Richtung der bestmöglichen Lösung?
Embrace differences/ *Nehme Unterschiede an*	Vertraue ich darauf, dass die anderen Teilnehmer auch mit besten Intensionen arbeiten?

Tab. 1: Werte / How-We-Run Prinzipien und Beispielfragen

Wie diese Prinzipien genutzt werden zeigen wir am Beispiel von Entscheidungsfindungen. Wenn es unterschiedliche Wahlmöglichkeiten gibt, die jeweils vernünftig erscheinen, kann der Bezug auf die gemeinsamen Werte zu einer gemeinsam getragenen Entscheidung helfen. Hier für die How-we-run Prinzipien einige Fragen, die dabei helfen können.

Im IT-Bereich wird in Gesprächen, Teambesprechungen und Entscheidungsfindungen nicht immer, aber immer wieder auf die How-we-Run Prinzipien verwiesen. Damit wird ersichtlich, dass sie ein Baustein der Innovationskultur geworden sind.

2.2 Endanwender im Fokus

Um beim Kunden und am Markt erfolgreich zu sein muss die Innovation ein Kundenproblem lösen [Sp]. Die Ausrichtung auf den Endkunden ist in dem hier betrachteten Unternehmen ein unternehmensweites Prinzip. Es ist im IT-Bereich jedoch besonders ausgeprägt. Dies zeigt sich in der Darstellung der internen IT-Wertschöpfungskette im Portal, dort ist der Endkunde prominent am Anfang und am Ende zu sehen. (Verknüpfung zur Dimension Ablauf/Prozess). Es wird deutlich durch die immer wieder betonten Aussagen des CIO an IT-Mitarbeiter/innen sowie interne Kunden: “In allem was wir tun sind wir auf den Endanwender ausgerichtet”. Es zeigt sich auch in der erweiterten Investition in das Endnutzererfahrung-Design-Zentrum, das IT-Projekte dabei unterstützt die Endnutzerinteraktion schon während der Entwicklung systematischer zu verbessern (Verknüpfung zur Dimension Produkt).

2.3 Ideenmanagement

Das unternehmensweite Ideenmanagement hat durch den verstärkten Innovationsfokus weitere Anforderungen bekommen: Transparenz über die unterschiedlichen Ideenpfade, klarere Zuordnung und schnellere Bearbeitung. Die möglichen Eingangspfade und Schnittstellen

zu Innovationevents wurden dokumentiert und im Mitarbeiterportal veröffentlicht. Weiter wurde die Zuordnung für die Bearbeitung der Ideen über die Zuordnung zu Prozessen im Unternehmens-Ablaufmodell (Process Map) systematisiert und damit eine klarere Verantwortung für Bewertung durch Experten erreicht (Verknüpfung zur Dimension Prozess). Für Ideen, die den IT Bereich betreffen, gibt es eine regelmäßige Abstimmung zwischen Ideenmanagement und dem IT Process Office, um Entscheidungen über Grenzfälle in der Zuordnung und mögliche Beschleunigung von Langläufern zu treffen.

2.4 Agile Entwicklung

Der Einsatz von Agilen Entwicklungsmethoden hat sich im Unternehmen über einen Zeitraum von mehr als zehn Jahren etabliert [Rö16]. Insbesondere im Produktentwicklungsbereich ist er das Standardvorgehen. Im IT-Bereich dagegen waren die Agilen Entwicklungsmethoden bis vor wenigen Jahren in einzelnen Bereichen wie Web-Entwicklung oder IT-Produkte für den Schulungsbereich etabliert. In anderen Bereichen, wie der Entwicklung für Business-Kernsysteme, wurde im Projektmodus entwickelt, also nicht mit stabilen Produktteams und oft mit einem Lieferzyklus von drei Monaten. Um den notwendigen Schub hin zu einer neuen, einer stärkeren Innovationskraft mit häufigeren Lieferungen zu erreichen wurde im IT-Bereich die flächendeckende Umstellung auf die Agile-Entwicklung mit einem Agile-Transition Projekt umgesetzt. Das Ergebnis ist eine organisatorische Umstellung auf mehr als 70 stabile Produktteams (Verknüpfung zur Dimension Organisation) mit jeweils verantwortlichem Produkteigner. Weitere Aspekte der Transition waren methodische Trainings für die umgestellte Methodik, einschließlich Komponenten wie den Rollen in Scrum-Teams, Nutzung und Priorisierung von Backlogs, schnelleren Iterationen mit kleineren Liefereinheiten etc. Die Umstellung der IT-Entwicklungsprozesse wurde und wird parallel als Aktualisierungen der IT-Prozesslandkarte dokumentiert und kommuniziert (Verknüpfung zur Dimension Prozesse). In Abschnitt 3.2 gehen wir etwas näher auf die Umstellungen im “Änderung und Freigabe”-Prozess ein.

2.5 Freiräume für Innovation

Im hier betrachteten Unternehmen gibt es mehrere Ansätze, um Freiräume zu schaffen und Innovation zu fördern.

- Fellowships bieten Mitarbeitern/innen die Möglichkeit für einen Zeitraum von bis zu sechs Monaten aus der aktuellen Rolle auszusteigen und in einem anderen Bereich oder einer anderen Rolle zu arbeiten. Die Maßnahme ist durch die zentrale Personalabteilung gesteuert und damit nicht nur abhängig von Abteilungsentscheidungen (Verknüpfung zur Dimension *Organisation*). Im IT-Bereich wird diese Möglichkeit gerne genutzt, einerseits um Mitarbeiter/innen mit neuer Sichtweise in aktuelle

Themen einzubinden andererseits nutzen Mitarbeiter/innen aus IT diese Möglichkeit, um sich selbst in andere Bereiche, neue Themen oder Rollen einzubringen.

- Intrapreneur-/Binnenunternehmer-förderung im Unternehmen ermutigt unternehmerisches Denken, die Weiterentwicklung eigener Ideen bis hin zum Spin-off [DE03]. Im IT-Bereich zeigt sich die Förderung in der Unterstützung vom Management, eigene Ideen zu treiben, auszuprobieren und damit Endkundenrückmeldung einzuholen. Da gilt es zu zeigen, dass die Idee innovativ ist, ein Problem löst und umsetzbar ist. Dies wird auch als "Show-Case" sichtbar in Hackathons, bei denen für die am höchsten bewerteten Vorschläge die jeweiligen Teams für einige Tage miteinander wetteifern und dann ihre erarbeiteten Lösungen im Plenum mit Endbenutzerrückmeldung vorstellen. Für alle IT-Mitarbeiter/innen zeigt sich die Förderung des unternehmerischen Handels durch die strategische Entscheidung, die Entscheidungsfreiheit jedes Mitarbeiters / jeder Mitarbeiterin zu ermutigen und zu honorieren[4]. Wenn "Empowerment" (siehe auch Beispiel in Abschnitt 3.2) und "Right to fail"/Recht-auf-Fehler gelebt werden, dann wird es leichter sich den Freiraum zu nehmen etwas auszuprobieren, das auch mal schiefgehen darf. Weil dieser Baustein einen Sichtwechsel/ "Mindshift" bedeutet, ordnen wir ihn mit Verknüpfung zur Dimension Strategie ein.
- DesignThinking-Räume wurden in den letzten Jahren in den Gebäuden im Unternehmen eingerichtet, um das kreative und innovative Arbeiten zu fördern (Verknüpfung zur Dimension Prozess). Diese Räume sind anders ausgestattet als traditionelle Besprechungsräume. Stattdessen findet man Hocker, Stehtische, Material für Ideensammlung, Interaktive Bildschirme und beschreibbare Wände. Im IT-Bereich, in dem Besprechungsräume oft ausgebucht sind, wird die Buchung von DesignThinking-Räumen besonders überwacht, um die Priorisierung der Nutzung für kreatives Arbeiten sicherzustellen.

2.6 Weitere Ansätze

Wir haben einige Prinzipien und methodische Ansätze betrachtet, die im Unternehmen und im IT-Bereich im Besonderen als Bausteine einer Innovationskultur genutzt werden. Anhand der Dimensionen Innovationskultur, Strategie, Organisation, Prozess, Produkt konnten wir einige Verknüpfungen aufzeigen. Weiter lässt sich aus den Hinweisen der Nutzung im IT Bereich die eine und andere Abhängigkeit zwischen den Bausteinen ablesen. Zu den oben betrachteten Ansätzen gibt es weitere, auf die wir hier nicht eingehen können, dazu gehören zum Beispiel Innovationsnetzwerke [FHM16], die von mehr oder weniger regelmäßigem Austausch bis zu intensiven Gemeinschafts-entwicklungen reichen können. Ein weiterer Baustein ist die Design-Thinking Methode, auf die an anderer Stelle eingegangen wird [Rö16].

[4] Der Ansatz des Intrapreneurship und die zusätzlichen Freiräume können auch negative Effekte haben, darauf wird z.B. in [Ge02] hingewiesen.

3 Beispiele

Wir zeigen an Hand von zwei Beispielen wie sich der Innovationsschub im IT-Bereich darstellt. Ein Tag im Leben eines Mitarbeiters/einer Mitarbeiterin zeigt die Endkundensicht. Anhand des Tagesablaufs eines typischen Mitarbeiters / einer typischen Mitarbeiterin erleben wir mit, wie diese mit innovativen Produkten aus dem IT-Bereich in Berührung kommen und damit produktiver arbeiten können. Das zweite Beispiel zeigt die Prozesssicht für den "IT Änderung und Freigabezyklus" und betrachtet die Auswirkungen der Innovationskultur auf diesen Prozess.

3.1 Ein Tag im Leben eines Mitarbeiters

Der Tag einer Mitarbeiterin beginnt zum Beispiel mit einer Besprechung im Nachbarstandort. Sie nutzt die Shuttle-App, um zu prüfen, wo der Verbindungsbus gerade fährt, und ob sie ihn noch erreicht. Im Bus, oder auch sonst, unterstützt die EasyConnect-App Besprechungen online oder in Person. Das FioriLaunchPad bietet eine einheitliche Schnittstelle zu Urlaubsanträgen, verschiedenen Ticket-Systemen und anderen Anwendungen. Die EasyShopping-App erleichtert die Bestellung von IT Bedarf, für die am häufigsten gesuchten Teile mit nur einem Klick. Besprechungen werden in immer mehr Räumen durch das Microsoft Surface Hub mit Skype und interaktiver Whiteboardfunktion unterstützt. Die Concur-App ermöglicht papierlose Geschäftsreisen von der Planung über die Abrechnung bis zur Auszahlung der Reisekosten.

3.2 IT Änderungs- und Freigabe Zyklus

In dem Prozess IT-Änderung-und-Freigabe wird die Freigabe von IT-Systemänderungen gegen verpflichtende und empfohlene Kriterien geprüft (Sicherheit, Architektur, Datensicherheit, Einhaltung von Standards, u.a.). Mit der Einführung der agilen Methodik mussten in diesem Prozess wesentliche Umstellungen vorgenommen werden, um die Innovationsanforderungen für hohe Frequenz von Lieferzyklen zu erreichen, bei gleichzeitiger Reduzierung des zeitlichen administrativen Aufwands pro Lieferung (Mehrwert: Vereinfachung). Waren für Geschäftsprozesse auf IT-Kernsystemen früher einmal im Monat Freigaben ins Produktivsystem möglich, so sind jetzt zusätzlich wöchentliche Freigaben möglich. Zudem laufen die Wartungsfreigaben, die früher als Sonderfälle behandelt wurden, auch über den neuen einheitlichen Freigabeprozess. Das ermöglicht die Nutzung derselben Werkzeugunterstützung und führt zu mehr Transparenz zwischen den Anforderungen für agile Innovation und stabilen Systemen. Abbildung 2 zeigt den IT Änderung- und Freigabeablauf im Kontext der "IT Process Map".

Schon auf dieser Übersichtsebene wird der Wandel sichtbar von der Betonung der sequentiellen Qualitätsprüfungen zu der Betonung des Agilen-Entwicklungszyklus (Baustein Agile

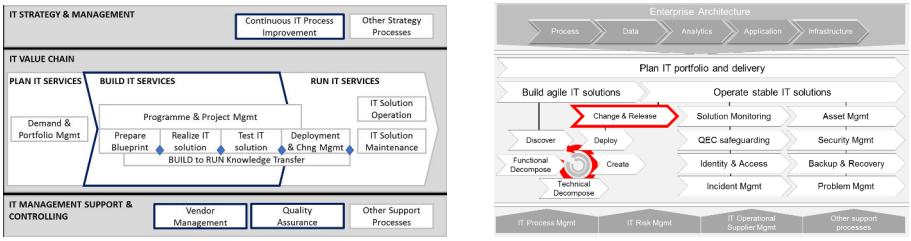


Abb. 2: Prozesssicht vor und nach der “Agile Transition”

Methoden) mit dem Prozess IT-Änderung-und-Freigabe als Übergang in die Produktivlandschaft. Ein wichtiger Aspekt dieser Umstellung ist auch der Baustein Intrapreneurship/ Empowerment. Der jeweils Verantwortliche für eine Lieferung ins Produktivsystem ist für alle Aspekte dieser Lieferung verantwortlich. Es gibt Werkzeuge, Prozesse und Experten die sie/ihn unterstützen, aber es gibt keine großen “Quality-Gate”-Sitzungen mehr in denen früher von der IT-Qualitätsabteilung für jedes Projekt/Lieferung die Abnahme gemacht wurde. Die Umstellung bekommt inzwischen sehr gute Rückmeldungen von den internen Kunden, insbesondere für die schnelleren Lieferzyklen, und von IT-Mitarbeitern vor allem für die gute Werkzeugunterstützung (Release-App) und die Auswahl an wöchentlichen Lieferterminen.

4 Herausforderungen, Miss- und Erfolge

Wir haben Aspekte firmeninterner Innovationskultur an aktuellen Beispielen in einem IT Unternehmen betrachtet. An einigen Stellen ist die Stärkung der Innovation gelungen und zeigt sich durch die Innovationszyklen von IT-Lieferungen, die schneller und häufiger sind, sowie in einem Schub an IT-Produkten, die dem Endanwender den Arbeitsalltag erleichtern. Wir habe dabei einige wesentliche Bausteine und Dimensionen der Innovationskultur betrachtet und hoffen, dass das Aufzeigen der Verknüpfungen anhand von tatsächlichen Umsetzungen dem Leser hilft, zu reflektieren und eigene Erkenntnisse abzuleiten[5].

Als eine Herausforderung für das Zusammenwirken und Nutzung der Bausteine für eine effektive Innovationskultur wollen wir die Größe des Unternehmens betrachten. Einerseits kann die Innovationskultur in einem größeren Unternehmen von gewissen Vorteilen profitieren. Durch die “Economy of scale” ist es generell leichter, dediziertes Personal für Innovationsthemen über Bereichsgrenzen hinweg bereitzustellen. Somit lassen sich z.B. ein unternehmensweites Ideenmanagement, Trainings für Themen wie Design-Thinking und Innovationsnetzwerke leichter etablieren. Andererseits ist in einem kleineren Unternehmen typischerweise die Abhängigkeit von anderen Abteilungen und die Entscheidungslatenz geringer. Mit weniger Hierarchien und Schnittstellen liegen auch die Entscheidungsebene und die Umsetzung näher beieinander und können damit grundsätzlich spezifischer sein.

[5] Wir verweisen den interessierten Leser auf [Sc15] für weitere Ausführung zum Thema Innovationskultur.

Ein Beispiel: Es ist wahrscheinlicher, in einem großen Unternehmen eine oder mehrere Vernetzungen mit Hochschulen vorzufinden als in einem kleineren Unternehmen. Andererseits sagt dies nichts über die Qualität und den Innovationsbeitrag aus. So kann eine intime Innovationszusammenarbeit aus einem kleinen Unternehmen gegebenenfalls von Anfang an konkreter sein und in kürzerer Zeit zu Innovationen führen.

Konkrete Beobachtungen aus dem hier beschriebenen Kontext zeigen, dass sich für den IT-Bereich einige Innovationsvorteile aus dem Gesamtunternehmen nutzen lassen, z.B. durch gemeinsame Prinzipien, Trainings für Innovationsmethodik und Impulse aus dem Austausch in Innovationsnetzwerken sowie Innovationsevents. Andererseits zeigt sich auch, dass man den Bereich der eigenen Einflussnahme als Unternehmen im Unternehmen verstehen und steuern muss, um bereichsspezifische Innovationen auch priorisieren und treiben zu können. Sich auf die unternehmensweiten Innovationsansätze zu verlassen führt nicht zum Erfolg. Hier kann der "Intrapreneurship"-Ansatz helfen Misserfolge und Enttäuschungen zu reduzieren.

Literaturverzeichnis

[DE03] Draeger-Ernst, Anne: Vitalisierendes Intrapreneurship: Gestaltungskonzept und Fallstudie. Hampp, 2003.

[FHM16] Fieseler, Christian; Hoffmann, Christian Pieter; Meckel, Miriam: Eine Kultur der Innovation: Die Bedeutung von Innovationsnetzwerken. In (Hoffmann, Christian Pieter; Lennerts, Silke; Schmitz, Christian; Stölzle, Wolfgang; Uebernickel, Falk, Hrsg.): Business Innovation: Das St. Galler Modell. Springer Fachmedien Wiesbaden, Wiesbaden, S. 313–337, 2016.

[Ga14] Gartner: , Gartner Identifies the Top 10 Strategic Technology Trends for 2015. Online: https://www.gartner.com/newsroom/id/2867917, Oktober 2014.

[Ge02] Gerlmaier, Anja: Neue Selbstständigkeit in der Informationsgesellschaft. Dissertation, Institut für Psychologie, Universität Dortmund, August 2002.

[Rö16] Rösel, Andreas: Are We Ready for Disruptive Improvement? In (Kuhrmann, Marco; Münch, Jürgen; Richardson, Ita; Rausch, Andreas; Zhang, He, Hrsg.): Managing Software Process Evolution: Traditional, Agile and Beyond – How to Handle Process Change. Springer International Publishing, Cham, S. 77–91, 2016.

[Sc15] Schmitt, Markus: Innovationskultur – Grundlage einer zukunftsfähigen Arbeitskultur. In (Widuckel, Werner; De Molina, Karl; Ringlstetter, Max J.; Frey, Dieter, Hrsg.): Arbeitskultur 2020: Herausforderungen und Best Practices der Arbeitswelt der Zukunft. Springer Fachmedien Wiesbaden, Wiesbaden, S. 73–88, 2015.

[Sp] Springer Gabler Verlag: , Stichwort: Innovation. Online: http://wirtschaftslexikon.gabler.de/Archiv/54588/innovation-v11.html.

[st17] statista: Das Statistik-Portal: , Entwicklung der durchschnittlichen täglichen Nutzungsdauer des Internets in Deutschland in den Jahren 2000 bis 2017 (in Minuten). Online: https://de.statista.com/statistik/daten/studie/1388/umfrage/taegliche-nutzung-des-internets-in-minuten/, 2017.

M. Tichy, E. Bodden, M. Kuhrmann, S. Wagner, J.-P. Steghöfer (Hrsg.): SE 2018,
Lecture Notes in Informatics (LNI), Gesellschaft für Informatik, Bonn 2018 181

Impact of Hypothesis-Driven Development on Effectiveness, Quality, and Efficiency in Innovation Projects

Sebastian Klepper[1] Bernd Bruegge[2]

Abstract: We evaluate the impact of hypothesis-driven development in innovation projects dealing with complex problems, focusing on decision making processes instead of experimentation methods. Our findings from multiple qualitative case studies show that this type of empirical research used for decision support can enhance effectiveness and quality without negatively impacting efficiency.

Keywords: Hypothesis-driven development; Decision support; Continuous innovation; Continuous experimentation; Project success factors

1 Introduction

The latest Annual State of Agile Report[3] indicates that there are multiple success factors for innovation projects which drive organizations to adopt agile methodologies in the hopes of maximizing their overall productivity: Organizations want the ability to react to change while delivering the right scope with great speed, maximize business value as well as product quality and user satisfaction. In short, they expect agile projects to strike the perfect balance between all possible constraints. Interestingly, while almost all respondents are able to realize improvements in efficiency, about a quarter of them fail to achieve reduced project risk, improved quality, or better business/IT alignment. A consequential trend seems to be to revert to planning to better control risk and results. Popular activities include story mapping, iteration and release planning. Even road-mapping and portfolio planning have entered the picture, possibly related to increasingly large organizations adopting agile methodologies.[4]

This indicates a desire to gain more control of project performance and outcome and begs the question if there are better ways to achieve this than re-introducing planning activities. Our hypothesis is that instead of through planning we can improve project performance or success through better decision making processes and an empirical approach to problem solving. Related case studies in this area have studies the introduction of continuous experimentation

[1] Technische Universität München, Munich, Germany sebastian.klepper@tum.de
[2] Technische Universität München, Munich, Germany bruegge@in.tum.de
[3] VersionOne Inc.: 11th Annual State of Agile Report, 2017, URL: https://explore.versionone.com/state-of-agile/versionone-11th-annual-state-of-agile-report-2
[4] also see VersionOne Inc.: 10th Annual State of Agile Report, 2016, URL: https://explore.versionone.com/state-of-agile/versionone-10th-annual-state-of-agile-report-2

in large organizations and discovered that, besides the challenge of experimenting in production, there is a fundamental conflict with the established way of working and thinking. [RM15; Ya17] "Company philosophy or culture at odds with core agile values" is also reported as the primary challenge when adopting and scaling agile processes.[3] We therefore focus on relevant decision making processes which form the basis for agile exploration as well as reactivity in the development processes. [see FS17; Ri11]

To judge impact on project success we must first define how to measure it. There have been many attempts at describing project constraints and success factors. The most prominent example is the "iron triangle" of scope, time, and cost with quality as an emergent property of the result. Over time, other factors have been described such as system quality, service quality, user satisfaction, learning effects, impact on stakeholders, organizational benefits. [At99; DM03] Bloomberg [Bl13] proposes an extension to the historic iron triangle in order to transform it to agility, quality, time. Ralph; Kelly [RK14] try to summarize the plethora of individual factors in a "software engineering success framework": Projects produce artifacts which perform in markets; projects exhibit efficiency, artifacts exhibit quality, and markets exhibit performance. Following these updated models, we differentiate *effectiveness*, *quality*, and *efficiency* as dimensions of project success.

2 Hypothesis-driven development

Our methodology of choice to improve decision making processes is hypothesis-driven development (HDD) – essentially the application of the scientific method to software product development. This school of thought found its way into software engineering by way of the "lean startup" movement popularized by Ries [Ri11] and further refined in a collaborative note with Eisenmann et al. [ERD11] that provides guidelines to formulating, evaluating and acting on hypotheses. A *hypothesis* in this context is any assumption critical to project success that is falsifiable, i.e., it can be validated by means of an experimental method.

This approach has gained traction in the world of empirical software engineering and multiple models have been proposed, either high-level process models or methods for specific types of experimentation. They focus on either initial exploration or continuous development and use hypotheses correspondingly to capture assumptions about business strategy, feature value, or feature refinement. Necessary data is collected using techniques like customer feedback, prototypes, or analytics data from live experiments. [Bo13; Bo14; EÞ16; Fa17; OB15]

It is important to differentiate between the mechanics of experimentation and the higher-level decision making process. At the core of our investigation is decision support, meaning different types of decisions such as triage, prioritization, selection, or scheduling. [AW05; HKP15, see] These decision problems are continuously identified and supporting empirical evidence is produced to address uncertainty and corresponding risk.

3 Case studies

To investigate impact of hypothesis-driven development on effectiveness, quality, and efficiency, projects applied the methodology over the course of several weeks. In total, we analyzed 10 projects with a total of 69 developers for a total of 120 weeks, amounting to 3,920 person days of evaluating hypothesis-driven development. Fig. 1 and 2 show the distribution of developer count and evaluation period across projects.

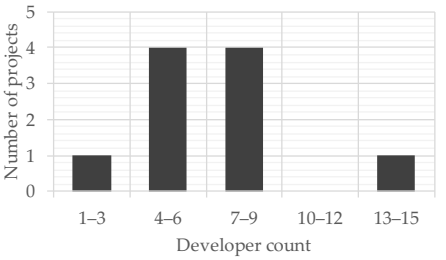


Fig. 1: Distribution of developer count

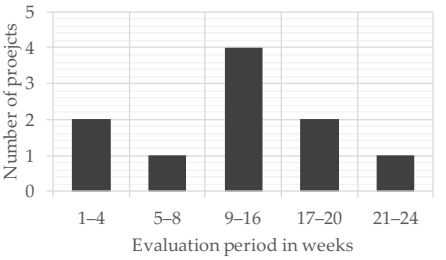


Fig. 2: Distribution of evaluation period

Subsequently projects qualitatively evaluated success criteria using personal opinion surveys in combination with data from issue trackers. [see also SSS08, pp. 35-62, 63-92] We also selected suitable projects with similar characteristics within each organizations to serve as a control group. These control projects also used agile methodologies but did not apply HDD and reportedly suffered from the aforementioned lack of quality in decision making.

To help projects apply hypothesis-driven development quickly and consistently we provided them with relevant literature as well as a custom issue type in JIRA[5]. This also allowed collection of structured data about experiments in correlation with regular development tasks.

4 Findings

Collecting and consolidating data and feedback from all cases studies allowed us to study differenct aspects of how hypothesis-driven development impacts project success. We subsequently present our findings grouped by the success dimensions defined initially.

4.1 Impact on effectiveness

Our first and foremost research question is: **Does hypothesis-driven development allow teams to solve problems more effectively?** We reason that frequently validating critical

[5] `https://www.atlassian.com/software/jira`

assumptions enables teams to make *better* decisions *earlier* in time. This should reduce risk posed by uncertainty since misconceptions are recognized faster and less time is wasted following a wrong path, presumably leading to better problem/solution fit as well as faster discovery of the right solutions.

A qualitative survey amongst project stakeholders captured feedback about solution fit, covering two important aspects:

1. How good does the solution match the problem?
2. How good is the solution compared to other available solutions?

In addition to "absolute" solution fit we are also measuring the aspect of innovation (finding new and better solutions). Fig. 3 and 4 show these ratings on a scale of 1 (excellent) to 6 (deficient). The majority of stakeholders rated both solution fit and innovativeness 1 (excellent) or 2 (good) with a respective mean of 1.85 and 1.75, indicating that projects indeed achieved a good problem/solution fit.

However, there is a noticeable second peak at 4 (adequate) in both dimensions. Further investigation revealed that these ratings stem from advanced users with undiscovered special needs. Their below-average rating conveys that the feature set is not "satisfactory" (3) yet.

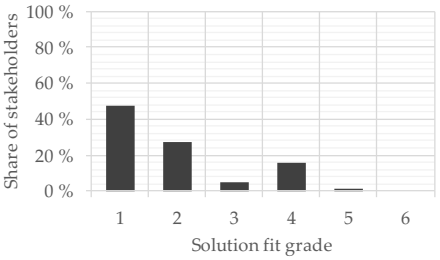


Fig. 3: Distribution of rating of solution fit

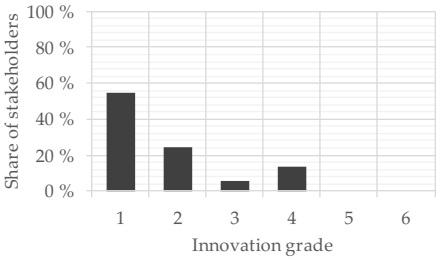


Fig. 4: Distribution of rating of innovativeness

Anecdotal evidence additionally suggests that testing of critical assumptions led yielded the following benefits to project members:

- Structured overview of facts and areas of uncertainty,
- conscious evaluation of decision components such as priority, effort, or alternative,
- easier decisions once comfortable level of confidence was reached,
- faster discovery of mistakes and less changes or pivots,
- implicit generation and documentation of rationale of these decisions.

On the other hand, critical respondents mentioned that empirical evidence is often trumped by other factors such as urgency, taste, or politics, raising the question whether additional research effort is justified. Sect. 4.3 addresses this concern about efficiency.

4.2 Impact on quality

In addition to higher effectiveness, we hypothesize that better decision making results in a solution of high quality. Timely recognition of wrong assumptions should result in fewer pivots, changes, or reverts and therefore more time being spent on proper execution and refinement. Our research question is: **Does hypothesis-driven development enable teams to also produce a high-quality solution?**

Using a qualitative survey, we collected feedback from different stakeholders of the projects, ranging from users to managers to technical experts. We asked for a rating of specific qualities such as ease of use, intuitiveness, performance, and robustness on a scale of 1 (excellent) to 6 (deficient). Due to the wide range of stakeholder type and quality factors we grouped feedback into two categories: Fig. 6 shows rating of technical quality based on performance, stability, and maintainability; Fig. 5 shows rating of usability based on visualization, intuitiveness, and ease of use.

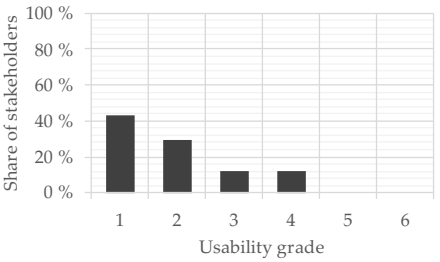


Fig. 5: Distribution of rating of user satisfaction

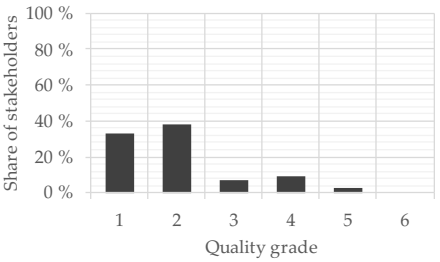


Fig. 6: Distribution of rating of technical quality

With the distribution skewed towards 1 (excellent) and 2 (good), projects achieved a mean rating of 1.81 on technical quality and 1.85 on usability. We consider this an above-average result, which is also backed by stakeholder feedback and the results of the Annual State of Agile Report where 25 % of respondents reported no improvement regarding software quality and 36 % regarding maintainability.[3]

This notably is an absolute and qualitative measurement of subjective criteria and not the quantitative result of a controlled experiment. However, anecdotal evidence from project stakeholders suggests that, compared to other projects in their respective organization, this approach indeed results in an improved use of time and therefore attention to non-functional requirements such as usability and sustainability.

4.3 Impact on efficiency

A serious concern was that HDD could negatively impact efficiency since it involves additional effort for designing, conducting, and evaluating experiments. As a counter-

argument, this additional effort could also be outweighed by time-savings through of increased effectiveness.

To investigate this, projects reported both cycle time and lead time before, during, and after the evaluation period. We differentiate between work items regarding functionality (e.g., user stories) and experiments themselves. To control for external influences or fluctuations load, we collected the same data from control projects. The collected data give an indication that efficiency is not negatively affected, in particular regarding the following questions:

Does HDD slow down development of functionality? Comparison of evaluation periods with non-evaluation periods both within the same projects and with control projects showed no significant shift in cycle time of other work items.

Does HDD slow down start of development of functionality? Similar analysis of lead times showed no significant differences in how much time work items were spending in the backlog before being picked up for development.

Does cycle time of experiments differ from other work items? Average cycle time for experiments was the same as for other work items. Interestingly, similar to outliers in development time some experiments took longer than expected to collect the required amount of data.

Fig. 7 shows a particularly demonstrative example of a project with stable cycle time over 28 weeks, unaffected by an 18-week evaluation period of hypothesis-driven development.

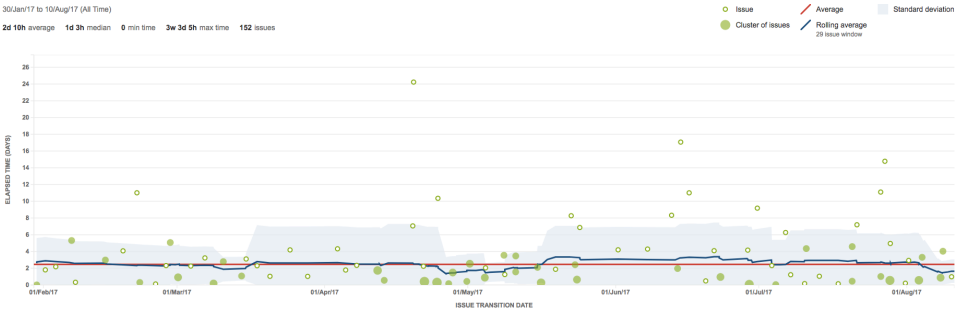


Fig. 7: JIRA control chart showing stable cycle time with and without HDD

5 Limitations

We presented the result of multiple independent experiments that were primarily qualitative in nature and not strictly controlled. However, initially having to dissect the aspect of "performance" or "success" into separate dimensions is an indicator for interplay of multiple influencing factors. In industry projects these are hard to isolate but we made an effort to

conduct evaluation during undisturbed periods and selected additional projects as a control group.

In our surveys and data analysis we tried to control for cross-influences. The line between effectiveness and quality is particularly blurry since both poorly executed features and the wrong feature set presumably both cause low satisfaction. Stakeholder surveys also are subjective and might suffer from selection bias as well as confirmation bias. Additionally, quality metrics are missing mantainability concerns like code-quality and test coverage.

Evaluation time per project was limited, ranging from two weeks to six months. This might skew results due to project members either not being familiar enough with the techniques yet or certain types of problems not having occurred yet. Longer evaluation periods and multiple evaluation points might alleviate this effect.

6 Conclusion

We studied how the emerging trend of empirical research in software engineering impacts project success. As both a contrast and supplement to related case studies, we focused on hypothesis-driven decision support instead of methods for experimentation. Our findings indicate that hypothesis-driven development enhances the quality of decisions and helps projects deal with uncertainty while not losing the ability to react to change. In particular, problem solving is enhanced by allowing to recognize misconceptions earlier and discover the right solutions faster. This leads to better problem/solution fit, overall quality of software, and usability since less time is spent on correcting mistakes. The same effect also appears to reduce negative impact on efficiency – the primary concern we faced when introducing the methodology. For more reliable and generalizable insights into the benefits and drawbacks of hypothesis-driven development, we pursue long-term quantitative evaluation.

References

[At99] Atkinson, R.: Project management: cost, time and quality, two best guesses and a phenomenon, its time to accept other success criteria. International Journal of Project Management 17/6, pp. 337–342, 1999.

[AW05] Aurum, A.; Wohlin, C.: Engineering and Managing Software Requirements. Springer, 2005.

[Bl13] Bloomberg, J.: The Agile Architecture Revolution: How Cloud Computing, Rest-Based SOA, and Mobile Computing are Changing Enterprise IT. Wiley, 2013.

[Bo13] Bosch, J.; Olsson, H. H.; Björk, J.; Ljungblad, J.: The early stage software startup development model: A framework for operationalizing lean principles in software startups. In: Lecture Notes in Business Information Processing. Vol. 167, pp. 1–15, 2013.

[Bo14] Bosch, J.: Continuous Software Engineering. Springer, 2014.

[DM03] Delone, W. H.; McLean, E. R.: The DeLone and McLean Model of Information Systems Success: A Ten-Year Update. Journal of Management Information Systems 19/4, pp. 9–30, 2003.

[ERD11] Eisenmann, T.; Ries, E.; Dillard, S.: Hypothesis-Driven Entrepreneurship: The Lean Startup. Harvard Business School Background Note 812-095 44/December, pp. 1–23, 2011.

[EÞ16] Ekström, M.; Þorvaldsson, Í. D.: Predicting and Analyzing Feature Value when R&D is an Experiment System, PhD thesis, University of Gothenburg, 2016.

[Fa17] Fagerholm, F.; Sanchez Guinea, A.; Mäenpää, H.; Münch, J.: The RIGHT model for Continuous Experimentation. Journal of Systems and Software 123/, pp. 292–305, 2017.

[FS17] Fitzgerald, B.; Stol, K. J.: Continuous software engineering: A roadmap and agenda. Journal of Systems and Software 123/, pp. 176–189, 2017.

[HKP15] Hesse, T.-M.; Kücherer, C.; Paech, B.: Experiences with Supporting the Distributed Responsibility for Requirements through Decision Documentation. Softwaretechnik-Trends 35/1, 2015.

[OB15] Olsson, H. H.; Bosch, J.: Towards continuous customer validation: A conceptual model for combining qualitative customer feedback with quantitative customer observation. In: Lecture Notes in Business Information Processing. Vol. 210, pp. 154–166, 2015.

[Ri11] Ries, E.: The Lean Startup: How Today's Entrepreneurs Use Continuous Innovation to Create Radically Successful Businesses. Crown Business, 2011.

[RK14] Ralph, P.; Kelly, P.: The dimensions of software engineering success. In: Proceedings of the 36th International Conference on Software Engineering - ICSE 2014. Pp. 24–35, 2014.

[RM15] Rissanen, O.; Münch, J.: Continuous Experimentation in the B2B Domain: A Case Study. In: Proceedings - 2nd International Workshop on Rapid Continuous Software Engineering, RCoSE 2015. Pp. 12–18, 2015.

[SSS08] Shull, F.; Singer, J.; Sjøberg, D. I.: Guide to advanced empirical software engineering. Springer London, 2008.

[Ya17] Yaman, S. G.; Munezero, M.; Münch, J.; Fagerholm, F.; Syd, O.; Aaltola, M.; Palmu, C.; Männistö, T.: Introducing continuous experimentation in large software-intensive product and service organisations. Journal of Systems and Software/, 2017.

SWM Session 3 - Neue Ideen

M. Tichy, E. Bodden, M. Kuhrmann, S. Wagner, J.-P. Steghöfer (Hrsg.): SE 2018,
Lecture Notes in Informatics (LNI), Gesellschaft für Informatik, Bonn 2018 191

Continuous Innovation and Experimentation in Complex Problem Domains: Problem Solving and Decision Support as a Starting Point for a Unified Process Framework

Sebastian Klepper[1], Christian Grimm[2], Bernd Bruegge[3]

Abstract: Continuous software engineering enables experimentation and empirical research in complex problem domains. Existing process models describe different approaches for exploration, innovation, and refinement. We propose a new approach based on problem solving techniques and focused on decision support to serve as the starting point for a unified process framework.

Keywords: Continuous innovation; Continuous experimentation; Continuous software engineering; Hypothesis-driven development; Complex problem solving; Decision support; Agile development

1 Introduction

Historically, software engineering aimed to produce complete, consistent, and correct solutions. Agile methodologies improved on completeness and consistency through incremental and iterative development, respectively. Providing the correct solution to a problem appears to remain an issue (to this day[4]), especially considering cost-effectiveness. Lack of domain knowledge was identified as a root cause early on, causing a shift towards value-orientation and knowledge management techniques. [AW05, pp. 309-326, 427-452, 453-476]

New technologies and techniques utilized in continuous software engineering (CSE) reduce time and effort required for applying empirical research and the scientific method. Multiple models have emerged for using feedback, data, experiments, and more to generate domain knowledge for both exploration and refinement. [Bo12; Bo14; FS17]

In this paper we review existing models in dimensions to discover commonalities and differences as well as gaps. We subsequently present a complementary approach that combines existing concepts with new ideas to serve as the starting point for our vision of a unified process framework that is comprehensive but still easy to learn, apply, and tailor.

[1] Technische Universität München, Munich, Germany sebastian.klepper@tum.de
[2] Technische Universität München, Munich, Germany christian.grimm@tum.de
[3] Technische Universität München, Munich, Germany bruegge@in.tum.de
[4] see VersionOne Inc.: 11th Annual State of Agile Report, 2017, URL: https://explore.versionone.com/state-of-agile/versionone-11th-annual-state-of-agile-report-2

2 Related work

We reviewed recently published process models that fit the definition of continuous innovation and experimentation. We differentiate between models focused on innovation and value prediction (HDE [ERD11], IES [Bo12], ESSSDM [BLB13], EVAP [FOB15]) and models focused on experimentation and continuity (HYPEX [Bo14, pp. 155–163], QCD [OB15], DVOCE [EÞ16], RIGHT [Fa17]). From this analysis as well as from the literature presented initially, we deem the following dimensions essential for a unified model of this scientific approach to continuous software engineering:

Continuity Execute activities asynchronously and in parallel, but at least iteratively.
Opportunism Learn from unexpected insights and seize opportunities for value creation.
Experimentation Test hypotheses, measure effect of changes, compare alternatives.
Prediction Validate critical assumptions before investing significant time and resources.
Innovation Discover market conditions, customer needs, potential features, or technologies.
Refinement Improve and optimize existing solutions, e.g., performance, quality, utilization.
Decision-making Support decisions, e.g., regarding priorities, feature set, or technology.
Research methodology Use research method and data source appropriate for the situation.

Models are rated on how comprehensively they describe these dimensions: scarcely (0), rudimentary (1), or detailed (2). Fig. 1 and 2 show our rating for both groups of models, illustrating the difference in scope, focus, and depth. We want to draw special attention to the need for integration of decision and execution processes — an integral part of CSE dubbed "BizDev" by Fitzgerald; Stol [FS17].

Fig. 1: Rating of process models focused on innovation and value prediction

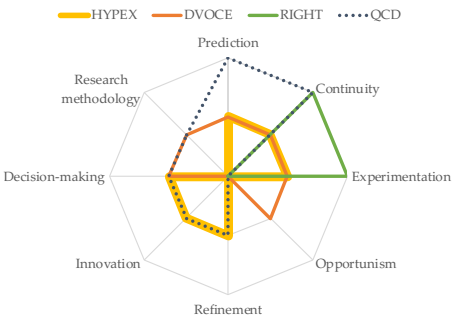


Fig. 2: Rating of process models focused on experimentation and continuity

3 Proposed workflow

Since our goal is to unify existing approaches, the review results suggest starting with an overarching problem solving process. This process should focus on decision-making,

research, and prediction; later it should integrate with other dimensions. We draw inspiration from approaches to solving "wicked" problems with no clear definition or solution, replacing issues and arguments with problems and insights. [cf. Ku70] Our workflow recursively moves through the problem space by analyzing and breaking down problems, looking for solutions, and conducting research for decision support in the face of uncertainty.

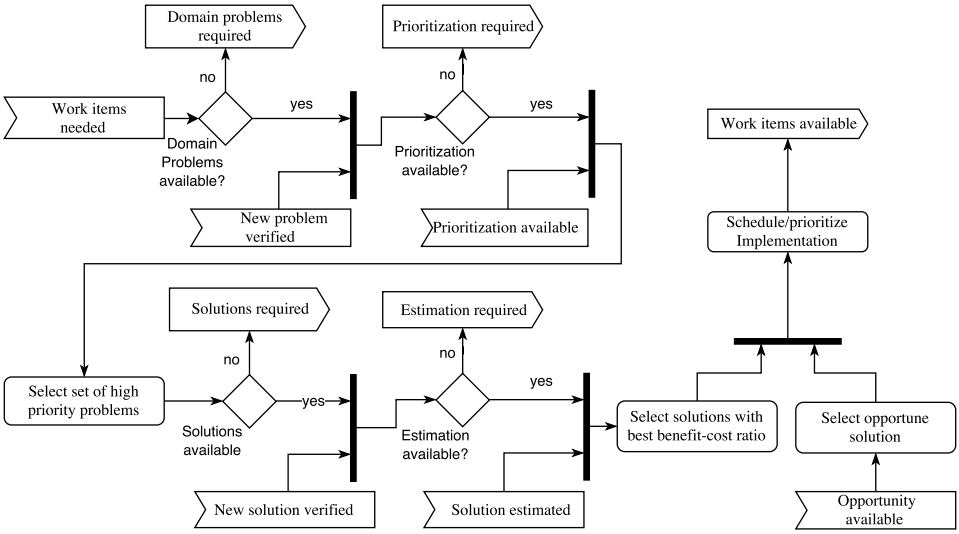


Fig. 3: Workflow for continuous problem solving

Fig. 3 models this problem solving process as a flow of activities triggered by events. While their higher-level order is goal-oriented, individual parts can be invoked at any time and subprocesses run asynchronously, ensuring continuity. These subprocesses (omitted for brevity) find and validate problems; break them down; find, validate, and estimate solutions; handle stakeholder input; and react to unexpected opportunities. We model these activities as prioritization, selection, and classification problems. [see AW05, pp. 267-286]

These types of decisions are supported by invoking the empirical research process illustrated in Fig. 4 to answer questions or validate hypotheses. Research activities are also executed asynchronously, allowing integration with experimentation models like QCD or RIGHT, and available results are handled in the subprocesses accordingly.

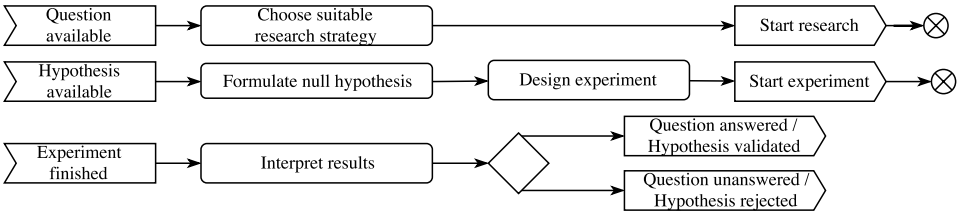


Fig. 4: Workflow for empirical and scientific research

4 Conclusion

Our vision is a comprehensive and flexible process framework for continuous innovation and experimentation. After reviewing existing models we designed a complementary workflow for "continuous problem solving" and plan to leverage existing approaches for subprocesses like experimentation, value prediction, prioritization, or selection.

At the current stage our workflow primarily aims to optimize the benefit-cost ratio but is also able to react to opportunities like technological advancements. Future work includes refining the design of subprocesses, integration with other models, and thorough validation.

References

[AW05] Aurum, A.; Wohlin, C.: Engineering and Managing Software Requirements. Springer, 2005.

[BLB13] Björk, J.; Ljungblad, J.; Bosch, J.: Lean Product Development in Early Stage Startups. In: Proceedings of the From Start-ups to SaaS Conglomerate: Life Cycles of Software Products Workshop (IW-LCSP). Pp. 19–32, 2013.

[Bo12] Bosch, J.: Building products as innovation experiment systems. In: Lecture Notes in Business Information Processing. Vol. 114 LNBIP, pp. 27–39, 2012.

[Bo14] Bosch, J.: Continuous Software Engineering. Springer, 2014.

[ERD11] Eisenmann, T.; Ries, E.; Dillard, S.: Hypothesis-Driven Entrepreneurship: The Lean Startup. Harvard Business School Background Note 812-095 44/, pp. 1–23, 2011.

[EÞ16] Ekström, M.; Þorvaldsson, Í. D.: Predicting and Analyzing Feature Value when R&D is an Experiment System, PhD thesis, University of Gothenburg, 2016.

[Fa17] Fagerholm, F.; Sanchez Guinea, A.; Mäenpää, H.; Münch, J.: The RIGHT model for Continuous Experimentation. Journal of Systems and Software 123/, pp. 292–305, 2017.

[FOB15] Fabijan, A.; Olsson, H. H.; Bosch, J.: Early value argumentation and prediction: An iterative approach to quantifying feature value. In: Lecture Notes in Computer Science. Vol. 9459, pp. 16–23, 2015.

[FS17] Fitzgerald, B.; Stol, K. J.: Continuous software engineering: A roadmap and agenda. Journal of Systems and Software 123/, pp. 176–189, 2017.

[Ku70] Kunz, W.; Rittel, H. W. J.; Messrs, W.; Dehlinger, H.; Protzen, T. M.; J., J.: Issues as elements of information systems, tech. rep., 1970.

[OB15] Olsson, H. H.; Bosch, J.: Towards continuous customer validation: A conceptual model for combining qualitative customer feedback with quantitative customer observation. In: Lecture Notes in Business Information Processing. Vol. 210, pp. 154–166, 2015.

M. Tichy, E. Bodden, M. Kuhrmann, S. Wagner, J.-P. Steghöfer (Hrsg.): SE 2018,
Lecture Notes in Informatics (LNI), Gesellschaft für Informatik, Bonn 2018 195

Herausforderungen für das IT-Produktmanagement durch externe Plattformen

Christopher Jud[1] und Georg Herzwurm[2]

1 Einleitung

Um Kunden Softwareprodukte zur Verfügung zu stellen, müssen diese von Unternehmen auf externen Plattformen angeboten werden [PTR15]. Diese Unternehmen werden dadurch zu Akteuren, die Softwareprodukte basierend auf einer Plattform entwickeln und vertreiben, die Plattform aber nicht selbst betreiben. Solche Akteure werden in der Literatur als Komplementoren bezeichnet. Auf Komplementoren wirken in der Entwicklung und dem Vertrieb von Softwareprodukten verschiedene Einflussfaktoren, welche sich aus den Plattformen ergeben. Als Beispiel kann hier die technische Realisierung von Softwareprodukten genannt werden oder Vorgaben für den Vertrieb über Absatzkanäle der Plattform wie z.B. AppStores.

Werden z.B. Vorgaben der Plattform nicht eingehalten, kann es passieren, dass Softwareprodukte nicht zugelassen und von Kunden nicht bezogen werden können. Die aus den Einflussfaktoren resultierende Komplexität für einen Komplementor wird erhöht, wenn mehrere Plattformen adressiert werden. Für viele Komplementoren ist die Bedienung mehrerer Plattformen (in der Literatur „multi-homing“ genannt [RT03]) daher eher Notwendigkeit als Wahl, wenn man sich nicht von relevanten Kundengruppen abschneiden möchte. Beispiele hierfür finden sich im Bereich mobiler Betriebssysteme wie Apple iOS und Google Android sowie bei Desktopbetriebssystemen Apple macOS, Linux oder Microsoft Windows.

Im Rahmen eines Forschungsprojekts wurden durch Fallstudien und Experteninterviews in Unternehmen sowie durch eine ergänzende Dokumentenanalyse verschiedene Einflussfaktoren von Plattformen auf komplementäre Softwareprodukte erhoben. Dabei beschränkt sich das Projekt auf Standardprodukte wie Apps, die von einer großen Menge von Kunden bezogen und nur durch vorgesehene Parameter angepasst werden können [HP09, YHL10]. Produktmanagement (PM) wurde dabei als eine Möglichkeit identifiziert, den Einfluss von Plattformen zu begegnen. Bisherige Ansätze des PM werden allerdings Anforderungen von Komplementoren auf Plattformen nicht oder nur bedingt gerecht. Erkenntnisse aus dem Forschungsprojekt zeigen, dass das PM um eine Plattformmanagement-Komponente ergänzt werden sollte. Hier werden Anforderungen

[1] Universität Stuttgart, BWI, Abt. VIII (Lehrstuhl für ABWL und Wirtschaftsinformatik II), Keplerstr. 17, 70174 Stuttgart, jud@wius.bwi.uni-stuttgart.de
[2] Universität Stuttgart, BWI, Abt. VIII (Lehrstuhl für ABWL und Wirtschaftsinformatik II), Keplerstr. 17, 70174 Stuttgart, herzwurm@wius.bwi.uni-stuttgart.de

oder Vorgaben externer Plattformen analysiert sowie Auswirkungen auf die verschiedenen Phasen bewertet, unter anderem dem Roadmapping. Eine wichtige Aufgabe ist dabei, Aktivitäten auf mehreren Plattformen zu koordinieren. Im Rahmen dieses Vortrags wird das Plattformmanagement als Teil des PM eingeführt sowie Einsatzmöglichkeiten und Anwendungsfälle diskutiert. Mit der Ergänzung durch das Plattformmanagement werden Komplementoren in der Praxis unterstützt, die Komplexität durch Plattformen sowie deren Einfluss auf Softwareprodukte zu begegnen.

2 Motivation

Der Einfluss von Plattformen wird in der Literatur aus verschiedenen Blickwinkeln diskutiert u.a. von [BHT15]. So wurde u.a. die Offenheit von Plattformen [BHT15] oder AppStores als Distributionskanal [JB13] diskutiert. Aus Sicht der Forschung lassen sich dabei verschiedene Herausforderungen für Komplementoren abgrenzen. Tabelle 1 gibt einen Überblick über unterschiedliche exemplarische Einflussfaktoren von Plattformen auf die Entwicklung sowie den Vertrieb von Softwareprodukten.

Tabelle 2: Kategorien und Beispiele von Einflussfaktoren von Plattformen

Einflussfaktoren	Beispiele
Governance-Strukturen	Zertifizierungsprozesse und Leitlinien zur Plattform
Technologien	Programmiersprachen, Datenbanken, proprietäre und OpenSource-Technologien, verbreitete oder spezifische Technologien
Ressourcen	SDK, IDE, Entwicklungs-Leitfäden, Trainings, APIs und Schnittstellen um auf Funktionalitäten von Plattformen zuzugreifen
Plattformbetreiber	Opportunistisches Verhalten des Plattformbetreibers, Unterstützung der Komplementoren, Entwicklung eigener Softwareprodukte auf der Plattform
Plattform-Ökosystem	Andere Komplementoren und Softwareprodukte, Kunden der Plattform, Potentiale von Erträgen mit der Plattform
Entwicklung der Plattform	Updates der Plattform, Änderungen in der Strategie des Plattformbetreibers, Änderungen in der Architektur der Plattform

Insbesondere aus Sicht des PM wird die Berücksichtigung sowie das Management von Plattformen bisher noch sehr zurückhaltend betrachtet. PM-Ansätze wie [HP09, KC09, FRI12, WBN06] adressieren insbesondere Fragestellungen im Bereich der Entwicklung von Softwareprodukten [WBN06] sowie darüber hinaus den Lebenszyklus [HP09, KC09]. Bei einem PM, insbesondere für Softwareprodukte auf externen Plattformen, ist es wichtig, betriebswirtschaftliche und technische Blickwinkel zu integrieren [HP09], da beide Bereiche Entwicklung und Vertrieb von Softwareprodukten auf Plattformen stark beeinflussen und nicht isoliert voneinander betrachtet werden können.

Die Einflussfaktoren einzelner Plattformen verstärken sich, wenn mehrere Plattformen adressiert werden. Es entsteht eine Komplexität, welche sich aus denen sich entwickelnden Plattformen sowie den unterschiedlichen Anforderungen an Komplementoren und Softwareprodukte resultieren. Dieser Umstand muss berücksichtigt sowie in Entwicklung und Management von Softwareprodukten integriert werden. Hierzu gehören einerseits organisatorische, andererseits technologische Vorbereitungen.

3 Plattformmanagement

Einen Ansatz, um die Komplexität und den Einfluss für Komplementoren zu adressieren, bietet ein dezidiertes Plattformmanagement innerhalb des Produktmanagements. In diesem Bereich werden Aufgaben gebündelt, wenn Softwareprodukte für externe Plattformen entwickelt und über diese vertrieben werden sollen. Zusätzlich werden Aufgaben vorgesehen, die Komplementoren betreffen, welche Softwareprodukte auf mehreren Plattformen anbieten. Möglichkeiten einer plattform-übergreifenden Entwicklung sollten vorgesehen werden, um Komplexität und Aufwand für den Komplementor zu reduzieren sowie Skaleneffekte bei u.a. der Entwicklung von Softwareprodukten nutzen zu können. Hierbei sind allerdings die Anforderungen an die Architektur des Softwareprodukts und Aufwände für individuelle Adaption und Portierung von Softwareprodukten für Plattformen zu bewerten. Weiterhin müssen Faktoren berücksichtigt werden, die unabhängig von Plattformen in z.B. Branchen vorhanden sind, von Plattformen aber durchgesetzt werden. Hierzu gehören u.a. gesetzliche Vorgaben, die eingehalten werden müssen.

Bisher fehlen Ansätze für ein Plattformmanagement als Teil des PM. Ein erster Einblick auf die Forschungsergebnisse wird in diesem Vortrag gegeben und Ansätze diskutiert.

Literaturverzeichnis

[BHT15] Benlian, A.; Hilkert, D.;Hess, T.: How open is this platform? The meaning and measurement of platform openness from the complementors' perspective, In: Journal of Information Technology 30 (3), pp. 209-228, 2015.

[FRI12] Fricker, S.: Software Product Management, In: Software for People, pp. 53-81. Springer Berlin Heidelberg, 2012.

[HP09] Herzwurm, G.; Pietsch, W.: Management von IT-Produkten, dpunkt-Verl., Bonn, 2009.

[JB13] Jansen, S.; Bloemendal, E.: Dening App Stores: The Role of Curated Marketplaces in Software Ecosystems, In: International Conference of Software Business, pp. 195-206.Springer, Berlin, Heidelberg, 2013

[KC09] Kittlaus, H-B.; Clough, P.: Software product management and pricing: Key success factors for software organizations. Springer Science & Business Media, 2009.

[PTR15] Porch, C.; Timbrell, G.; Rosemann, M.: Platforms: A Systematic Review Of The Literature Using Algorithmic Historiography. In: Proceedings of the 25th European Conference on Information Systems (ECIS) Research Paper, Paper 143, 2015.

[RT03] Rochet, J. C., & Tirole, J.: Platform competition in two-sided markets. In: Journal of the european economic association, 1(4), pp. 990-1029, 2003.

[WBN06] Van De Weerd, I.; Brinkkemper, S.; Nieuwenhuis, R.; Versendaal, J. Bijlsma, L.: A reference framework for software product management. In: Utrecht University, 2006.

[YHL10] Yoo, Y.; Henfridsson, O.; Lyytinen, K.: The New Organizing Logic of Digital Innovation: An Agenda for Information Systems Research, In: Information Systems Research 21 (4), pp. 724–735, 2010.

agile Produktentwicklung bei Software-Spin-Offs an der Universität

Katrin Kahle[1], Alexander Götze[2]

Abstract: Am Beispiel von Software-Spin-Offs der TU Dresden stellen wir vor, wie agile Methoden genutzt werden können, um den gesamten Vorgründungsprozess zu beschleunigen. Genauer gesagt geht es darum, unter begrenzter Zeit mit begrenzten Mitteln so schnell wie möglich ein Minimal Viable Product zu entwickeln, dessen erstes Featureset echten Nutzen für die vielversprechendste Kundengruppe liefert. Um dieses Ziel zu erreichen, müssen unzählige Entscheidungen in vielfältigen Themen wie Marktexploration, Produktkonzeption, Produktentwicklung, Geschäftsmodellentwicklung, Kundenentwicklung oder Finanzmitteleinwerbung von drei bis vier Teammitgliedern in kurzer Zeit getroffen werden. Bei diesen Entscheidungen wird das Team vom Softwareinkubationsprogramm DeltaHochDrei der TU Dresden bei Aufstellung und Priorisierung des Backlogs und des Sprint Plannings mit wirtschaftlichem, technologischem und unternehmerischem Hintergrundwissen unterstützt. Es werden typische Entscheidungssituationen und bisherige Erfahrungen mit der Anwendung agiler Managementprinzipien vorgestellt.

Keywords: Spin-Off, University, Interdisciplinary, Scrum, Lean, Agile, Business-Modell, Value Proposition Canvas, Jobs to get Done-Modell, SPIN-Modell, Softwaredevelopment, Design Thinking, Prototyping

1 Probleme von Spin-Offs aus der TU Dresden

Vom Gründungszentrum dresden|exists[3] der TU Dresden werden im Inkubationsprogramm DeltaHochDrei Gründerteams mit softwarebasierten Produkten begleitet. Die Mehrzahl der an der Universität entstandenen Gründungsideen unterscheidet ein Merkmal maßgeblich von allen anderen Gründungsideen, die wir insgesamt vorfinden:

Die Idee selbst beruht auf einer technologischen Innovation, die während eines Forschungsprojektes, einer Dissertation, Diplom- oder Masterarbeit untersucht wurde. Für die Ausarbeitung der Gründungsidee werden Probleme aus der Realwirtschaft gesucht, die mittels der innovativen Technologie gelöst werden können. Zielmärkte können dabei sogar in diversen Branchen liegen, zu denen häufig gar kein Kontaktnetzwerk besteht. Es wird also nach dem passenden Zielmarkt, einem attraktiven Produktdesign und Geschäftsmodell gesucht.

[1] Gründungszentrum dresden|exists der Technischen Universität Dresden, katrin.kahle@tu-dresden.de
[2] Gründungszentrum dresden|exists der Technischen Universität Dresden, alexander.goetze@tu-dresden.de
[3] dresden|exists begleitet seit mehr als 15 Jahren Startups an der TU Dresden. Seit März 2016 wird mit DeltaHochDrei ein besonderes Inkubationsprogramm für softtwarebasierte Gründungen angeboten.

Nascent Entrepreneurs außerhalb der Universität hingegen kennen ein Problem in einem bestimmten Zielmarkt und bauen eine genau passende Lösung für dieses Problem. Hier wird nach einer modernen Technologie für ein attraktives Produkt und Geschäftsmodell gesucht.

Spin-Offs basierend auf innovativen Technologien benötigen also extra Zeit für Marktexploration, Marktentwicklung und häufig den Aufbau zumindest eines kleinen Netzwerks in verschiedene Branchen, für das erste Produkt den richtigen Market-Fit zu finden.

Diese Teams sind häufig drei bis vier Personen stark und besitzen gute Sofwareentwicklungskompetenzen. Zum größten Teil gibt es noch keine Berufserfahrung außerhalb der Wissenschaft und generell wenig Erfahrung im Aufbau von Netzwerken in die Wirtschaft, Vertrieb, (Projekt-)Management und Software-Produktentwicklung.

Zum Aufbau eines marktfähigen Produktes und eines Unternehmens werden jedoch diese Kompetenzen auch schon ganz am Anfang eines Gründungsvorhabens benötigt. In Sachsen stehen durch eine Kombination aus dem EXIST-Gründerstipendium[4] und dem SAB Technologiegründerstipendium[5] maximal zwei Jahre als Produktentwicklungszeit zur Verfügung, um eine technologiebasierte Gründung mithilfe von Fördermitteln auf eigene Füße zu stellen – sprich danach entweder aus eigenen Umsätzen zu leben oder eine Risikokapitalfinanzierung einzuwerben respektive Beteiligungspartner zu finden.

Das kleine Team steht also simultan vor vier komplexen bis chaotischen Herausforderungen [ST96]:

1. Findung eines passenden Zielmarktes,
2. Entwicklung eines Produktes basierend auf einer Technologie aus der Forschung,
3. Aufbau von Wissen und Fähigkeiten in Marketing, Vertrieb, Produktentwicklung, Management, Finanzierung etc. und
4. Aufbau eines produktiven Teams und Unternehmens.

2 Methoden

Um die Zeit bis zu einem marktfähigen Produkt für Universitäts-Spin-Offs zu minimieren, arbeiten wir mit einem breiten Baukasten von aufeinander abgestimmten Methoden aus

[4] https://www.exist.de/DE/Programm/Exist-Gruenderstipendium/inhalt.html
[5] https://www.sab.sachsen.de/privatpersonen/f%C3%B6rderprogramme/technologiegr%C3%BCnderstipendium.jsp

den Bereichen Lean Startup[6], Design Thinking[7], agile Softwareentwicklung[8] und des agilen Managements[9].

Das tragende Skelett ist ein an Scrum[10] angelehnter Managementprozess für sehr kleinen Teams mit heterogenen Aufgaben. Wichtige Aspekte sind dabei

1. simultanes Innehaben der Scrum-Rollen Product Owner, Entwicklungsteam, Scrummaster und Management[11],
2. der Aufbau eines Backlogs für Entwicklungsaufgaben ebenso wie für alle Aufgaben des Unternehmensaufbaus und
3. die gemeinsame Priorisierung von Aufgaben aus der Softwareentwicklung und des Unternehmensaufbaus.

Darüber hinaus werden die Teams von erfahrenen Gründungsberatern mit Berufserfahrung in Softwarestartups und agilem Management unterstützt, um Entscheidungssituation deutlicher zu erkennen und die Traktion von getroffenen Entscheidungen zu erhöhen genauso wie die Geschwindigkeit der Entscheidungsfindung.

Die immense Transitionsleistung, die Forscher und Studierenden hin zur Gründerpersönlichkeit leisten müssen wird durch strukturiertes Lernen in der Gruppe mit anderen Software-Spin-Offs maßgeblich unterstützt.

Weiterhin werden die Spin-Offs in verschiedene Toolboxes, wie Design Thinking und Prototyping eingeführt. Mit Techniken wie Design Thinking als Prozess zur Förderung kreativer Ideen in den verschiedenen Gründungsphasen soll möglichst viel kreatives Potential bei allen Beteiligten des Innovationsvorhabens freigesetzt werden, um die systematisch komplexen Probleme zu lösen. In den verschiedenen Phasen des Design Thinking: Verstehen, Beobachten, Synthese, Ideen Finden, Prototypen entwickeln, Testen, Verfeinern können die Spin-Offs schnell ihre Hypothesen praxisnah validieren.

Unterstützend dazu wird durch die Möglichkeiten der Prototyping-Techniken gezeigt, wie man in den verschiedenen Phasen der Produktentwicklung mit wenig Aufwand Kundenfeedback einholen kann. Angefangen vom explorativen Prototypen hin zum funktionalen Prototypen als On- oder auch Offline-Version.

---

[6] Eric Ries – The Lean Startup; Steve Blank – 4 Steps of Epiphany; Ash Maurya – Running Lean; Etienne Garbulgi – Lean B2B; Croll und Yoskovitz – Lean Analytics; Osterwalder – Business Model Canvas

[7] http://www.designkit.org/methods

[8] Schwaber, Ken; Beedle, Mike: Agile Software Development with Scrum, Prentice Hall, 2002.

[9] Boris Gloger – Agiles Projektmanagement

[10] Agiles Manifest – http://agilemanifesto.org/

[11] Dieser Scrum-Prozess für kleine Gründerteams wurde insbesondere mit dem Startup washabich.de modelliert.

3 Ergebnisse

Ohne den Erfolg eines Gründungsvorhabens vorwegzunehmen wurden deutliche Effekte bei den Teams im Vergleich zu begleiteten Teams außerhalb des Programms beobachtet[12]. Im Programm

- entwickeln Teams eine ausgeprägte Feedback- und Reviewkultur[13],
- entwickeln Teams einen Managementprozess um bewusst und strukturiert Entscheidungen zu treffen[14],
- lernen typische Managementaufgaben zu erkennen,
- haben zu einem früheren Zeitpunkt mit potentiellen Kunden gesprochen,
- haben Teams im ersten Jahr mit deutlich mehr potentiellen Kunden und Feedbackgebern gesprochen,
- haben Teams eine höhere Anzahl von Prototypen getestet und
- bei vielen Fällen konnten Zeiträume zur ersten Finanzierung um massiv verkürzt werden[15].

Bibliography

[ST96] Stacey, R.: Complexity and Creativity in Organizations, Berrett-Koehler Publishers, San Francisco, pp. 72–106, 1996.

[12] Hierzu wird an einer Studie gearbeitet. Die Ergebnisse beruhen auf den Beobachtungen aus mehr als 15 Jahren Begleitung von Spin-Offs aus der Universität.

[13] Insbesondere die Meetingstruktur in Scrum führt auch bei kleinen Gründerteams zu einer Optimierung – Sprint Planning, Daily Scrum, Sprint Review und Sprintretrospektive.

[14] In unserer Scrum-Anlehnung wird dieser Prozess in einem gemeinsamen Backlog-Grooming angesiedelt

[15] Kahle/Brade (2015) https://openaccess.tu-dresden.de/ojs/index.php/ddtr/article/view/242 und https://subs.emis.de/LNI/Proceedings/Proceedings239/470.pdf

Erklärbare Software 1 - Understandable Verification

M. Tichy, E. Bodden, M. Kuhrmann, S. Wagner, J.-P. Steghöfer (Hrsg.): SE 2018,
Lecture Notes in Informatics (LNI), Gesellschaft für Informatik, Bonn 2018 205

Explainable Static Analysis

Eric Bodden[1] Lisa Nguyen Quang Do[2]

Abstract: Static code analysis is an important tool that aids in the early detection of programming errors, e.g. functional flaws, performance bottlenecks or security vulnerabilities. Past research in static analysis has mainly focused on the precise and efficient detection of programming mistakes, allowing new analyses to return more accurate results in a shorter time. However, end-user experience or static analysis tools in industry shows high abandonment rates. Previous work has discovered that current analysis tools are ill-adapted to meet the needs of their users, taking a long time to yield results and causing warnings to be frequently misinterpreted. This can quickly make the overall benefit of static analyses deteriorate.

In this work, we argue for the need of developing a line of research on aiding users of static analysis tools, e.g., code developers, to better understand the findings reported by those tools. We outline how we plan to address this problem space by a novel line of research that ultimately seeks to change static analysis tools from being tools for static analysis experts to tools that can be mastered by general code developers. To achieve this goal, we plan to develop novel techniques for formulating, inspecting and debugging static analyses and the rule sets they validate programs against.

Keywords: static analysis; debugging; visualization; program understanding

1 State of the Art

Historically, static code analysis tools have always faced the problem of unsatisfied users [Jo13, XWM14, CB16, Wi15]. In order to provide complete results, static analysis algorithms are known to over-approximate, returning all potential errors to the end-user, and sometimes overwhelming them with a "wall of bugs" [Jo13]. To lower the number of false positives, analyses are made more complex, but they take longer to run: minutes to hours to days. This forces developers to wait for a long time before they receive feedback on their code. Additionally, analysis tools often present warnings without explaining why they are returned, making it difficult for the code developer to distinguish false positives from genuine warnings. As researchers from the static-analysis tool vendor Coverity [Sy] (now owned by Synopsis) pointed out in 2010 [Be10], the lack of easy-to-understand warning messages is a primary cause of user dissatisfaction. Another one is the insufficient integration of the analysis tool with the developer's workflow [CB16].

As static analysis is generally an undecidable problem, all static-analysis tools must approximate their computations, thus unavoidably reporting false positives. In practice,

[1] Heinz Nixdorf Institute, Paderborn University & Fraunhofer IEM eric.bodden@upb.de
[2] Fraunhofer IEM lisa.nguyen@iem.fraunhofer.de

no tool can be precise enough to be false-positive free, which means that ultimately, the developer is the final judge in deciding whether a warning is true or false. Situations have been observed where complex warnings explained poorly by the tool misled the developer into thinking they were false positives [Be10]. Moreover, the notion of a false positive can by highly subjective, as even a true warning might not be relevant to certain developers. Therefore, there is a dire need for tools that efficiently help developers in determining which warnings are relevant to them.

2 Challenges

When trying to address the problems presented above, one faces the following challenges:

- To allow developers to better understand warnings, an analysis tool must provide richer information about *why* the analysis thinks that a certain program part is erroneous. Such information is not readily available. In fact, the static analysis tool must compute it while it conducts the analysis or as part of a post-processing module that runs after the analysis finishes. However, if no care is taken, such additional computations can significantly increase the time and memory consumption of the static analysis.
- Warnings must be presented to the developer in a way that is easy to understand. Little research has been performed on assessing which representations aid and don't aid code developers, and what elements should be part of an ideal representation [SBMH17].
- A method known to help developers assess a warning is the presentation of a witness, i.e., a real execution trace triggering the programming error [Le14]. But how can one generate witnesses also for incomplete code, and what are the program interfaces in such a case?

3 Required Research

In past work, we have investigated different means of adapting static analysis to the developers' needs [Ng17], and different ways of presenting the analysis results in an intuitive, user-friendly manner [Ng]. Through user studies, we have demonstrated that integrating developer information into the analysis and presenting results in a more transparent way helps developers fix errors significantly faster and provide them with a better overall experience of static analysis. While those first results are promising, there is still a lot to research to be conducted, especially in the areas of visualizations and responsiveness.

The envisioned line of research should aim at producing novel and improved means to present static analysis warnings to developers, ideally allowing them to assess a problematic situation quickly and correctly in all cases. To address the above problems, one must bring together expertise from the areas of static and dynamic program analysis and human-computer interaction. User studies would need to be an essential part of a work plan to address the problems mentioned above.

4 Possible solution outline

Concretely we propose to address the challenges outlined above by moving from a bare code-analysis technology to a developer-assistant system that will collaborate interactively with the developer to reach a joint understanding of the vulnerability situation at hand. As one of its design principles, such an assistant could include elements of gamification. For instance, the assistant could present to the developer potential vulnerabilities in a prioritized way, showing first those that are directly relevant to the developer, e.g. because they are close to the code they are currently editing, and who are also known to be easy to fix, taking into account the bug-fixing knowledge the particular developer is known to have acquired over the past. Then, when known to master well vulnerabilities at one level, the assistant could move the developer to the next level, displaying vulnerabilities e.g. of a different kind, or which are deemed slightly harder to judge. With knowledge of the developer's workflow and team setup, the assistant could even make use of the knowledge of the developer's team colleagues, e.g. by automatically including them into the judgment of vulnerabilities that the developer at hand is known not yet to be able to handle. This would allow the assistant to help one both derive the correct judgement (vulnerability or not) but would also aid the transfer of bug-fixing knowledge within the team.

As a second important element, however, we see dedicated new views and corresponding interactions that such an assistant should allow developers to make use of. In the past, we have seen a positive analysis developer response to a prototype that we built and which displays side by side the analyzed code with statements highlighted that contribute to a vulnerability, the implementation of the analysis but also a graph representation showing how the analysis executes over the given program part [Ng]. The user interface also allows novel interactions, e.g. setting special breakpoints that halt not the analyzed program but the static analysis when it processes the statement of the analyzed program that carries the breakpoint. In user experiments, we were able to show that such an approach eases the developers of static analyses in debugging the analyses that they write. The same views will likely be too complex to understand for developers who have no experience in static analysis. Yet, at the same time, we do believe that novel views are needed to explain developers *why* a static analysis arrives at a particular judgement, i.e., why it thinks that a certain vulnerability actually exists. Such novel views could also, for instance, display information about code locations at which the program analysis is uncertain, or might even ask dedicated, simple questions to the developer to help judge a vulnerability situation [DDA12], e.g.: "Is this untrusted input known to be sanitized elsewhere?" Those views, would then enable the developer to judge the vulnerability situation with much greater confidence, and in turn boost their ability to correctly judge future situations.

References

[Be10] Bessey, Al; Block, Ken; Chelf, Ben; Chou, Andy; Fulton, Bryan; Hallem, Seth; Henri-Gros, Charles; Kamsky, Asya; McPeak, Scott; Engler, Dawson: A Few Billion Lines of Code Later: Using Static Analysis to Find Bugs in the Real World. Commun. ACM, 53(2):66–75, February 2010.

[CB16] Christakis, Maria; Bird, Christian: What Developers Want and Need from Program Analysis: An Empirical Study. In: Proceedings of the 31st IEEE/ACM International Conference on Automated Software Engineering. ASE 2016, ACM, New York, NY, USA, pp. 332–343, 2016.

[DDA12] Dillig, Isil; Dillig, Thomas; Aiken, Alex: Automated Error Diagnosis Using Abductive Inference. SIGPLAN Not., 47(6):181–192, June 2012.

[Jo13] Johnson, Brittany; Song, Yoonki; Murphy-Hill, Emerson R.; Bowdidge, Robert W.: Why don't software developers use static analysis tools to find bugs? In: International Conference on Software Engineering (ICSE). pp. 672–681, 2013.

[Le14] Lerch, Johannes; Hermann, Ben; Bodden, Eric; Mezini, Mira: FlowTwist: Efficient Context-sensitive Inside-out Taint Analysis for Large Codebases. In: Proceedings of the 22Nd ACM SIGSOFT International Symposium on Foundations of Software Engineering. FSE 2014, ACM, New York, NY, USA, pp. 98–108, 2014.

[Ng] Nguyen Quang Do, Lisa; Krüger, Stefan; Hill, Patrick; Ali, Karim; Bodden, Eric: VisuFlow. https://blogs.uni-paderborn.de/sse/tools/visuflow-debugging-static-analysis/.

[Ng17] Nguyen Quang Do, Lisa; Ali, Karim; Livshits, Benjamin; Bodden, Eric; Smith, Justin; Murphy-Hill, Emerson: Just-in-time Static Analysis. In: Proceedings of the 26th ACM SIGSOFT International Symposium on Software Testing and Analysis. ISSTA 2017, ACM, New York, NY, USA, pp. 307–317, 2017.

[SBMH17] Smith, J.; Brown, C.; Murphy-Hill, E.: Flower: Navigating Program Flow in the IDE. In: Proceedings of the IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC'17). October 2017.

[Sy] Synopsys: Coverity. http://www.coverity.com/.

[Wi15] Witschey, Jim; Zielinska, Olga; Welk, Allaire; Murphy-Hill, Emerson; Mayhorn, Chris; Zimmermann, Thomas: Quantifying Developers' Adoption of Security Tools. In: Foundations of Software Engineering (FSE). pp. 260–271, 2015.

[XWM14] Xiao, Shundan; Witschey, Jim; Murphy-Hill, Emerson R.: Social influences on secure development tool adoption: why security tools spread. In: Computer Supported Cooperative Work & Social Computing (CSCW). pp. 1095–1106, 2014.

M. Tichy, E. Bodden, M. Kuhrmann, S. Wagner, J.-P. Steghöfer (Hrsg.): SE 2018,
Lecture Notes in Informatics (LNI), Gesellschaft für Informatik, Bonn 2018 209

Ein Ansatz zur nachvollziehbaren Verifikation medizinisch-cyber-physikalischer Systeme

Julia Padberg[1], Alexander Schlaefer[2] und Sibylle Schupp[3]

Abstract: Medizinische cyberphysikalische Systeme erfordern einerseits die Adaption an patientenindividuelle Parameter während einer Behandlung und andererseits den Nachweis eines sicheren Systemverhaltens. Wir schlagen vor, Nachweisbarkeit mittels Online Model-Checking und Nachvollziehbarkeit durch Anwendung von regelbasierten Transformationen zu verbinden.

Keywords: Softwaretechnik, Verifikation, Medizintechnik, MCPS, Patient-in-the-Loop, Echtzeit.

1 Motivation und Problemstellung

Die fortschreitende Digitalisierung im Gesundheitswesen betrifft auch Entwicklung und Einsatz technischer Assistenzsysteme, die medizinisches Personal in Diagnostik und Therapie unterstützen. Diese Systeme erreichen eine Komplexität, die für Ärzte und Ärztinnen nicht mehr nachvollziehbar ist, beispielsweise wenn adaptive, lernende Methoden zum Einsatz kommen. Zum Beispiel ändern sich Atemmuster während einer bewegungskompensierten Strahlentherapie [AR16]. Insbesondere wo Software aktiv Einfluss nimmt, wie etwa Kontrollsoftware bei "Patient-in-the-Loop"-Systemen, ist es wünschenswert, eine Methodik zu entwickeln, die Patientensicherheit nachweisbar garantiert und gleichzeitig für Entwickler und Anwender nachvollziehbar ist.

2 Lösungsansatz

Einen Ansatz zur Verifikation des Systemverhaltens in Echtzeit bietet das Online Model-Checking (OMC), das auch für unvollständig modellierte und adaptive Systeme Abweichungen vom erwarteten Verhalten nachweisen kann. Im Kontext medizintechnischer Systeme ist dadurch beispielweise der Abbruch der Behandlung möglich, bevor Schäden verursacht werden [AR16]. Adaption an das Patientenverhalten ist ebenfalls möglich, typischerweise aber nicht explizit gegeben und kann somit von klinischen Anwendern nicht nachvollzogen werden. Regelbasierte Transformation [PS16] andererseits kann den Adaptionsschritt des OMC so – und in einer Sprache - explizieren, dass er für das Fachpersonal nachvollziehbar wird.

[1] HAW Hamburg, Department Informatik, Berliner Tor 7, 20099 Hamburg, julia.padberg@haw-hamburg.de
[2] TUHH, Institut für Medizintechnische Systeme, 21073 Hamburg, schlaefer@tuhh.de
[3] TUHH, Institut für Softwaresysteme, 21073 Hamburg, schupp@tuhh.de

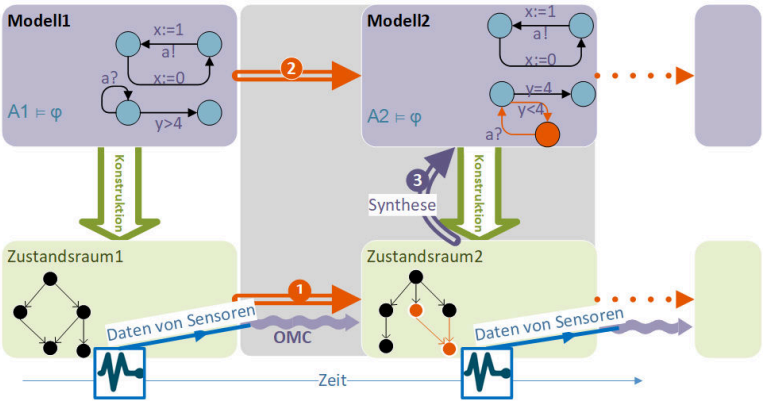


Abb. 1: Methodische Basis.

Abb.1 stellt das Ineinandergreifen von Verifikation und regelbasierter Transformation dar. Online Model-Checking verwendet ein partielles Modell (Modell1, links) und vergleicht die entlang der Zeitachse erfassten Daten mit dem daraus erzeugten Zustandsraum. Der grau hinterlegte Teil veranschaulicht den neuen Ansatz einer Adaption des Modells (Modell2, rechts), der den Adaptionsschritt des OMC durch die Anwendung von Regeln ergänzt (① in Abb. 1). Diese Regeln transformieren den Zustandsraum entsprechend der erhaltenen Daten und repräsentieren so die Änderungen, die sich durch die aktuellen Sensorwerte ergeben. Die Regeln zur Transformation des Zustandsraumes sind koordiniert mit denen der Modelltransformation (② in Abb. 1). Um eine Überanpassung der Modelle zu vermeiden, werden diese Modelltransformationen, die das spezifische Domänenwissen über medizinisch plausible Modelle abstrahieren, festgelegt. Die Synthese (③ in Abb. 1) wird unterstützt durch Techniken des Process-Mining einerseits und die koordinierten Regeln andererseits. Die Verifikation selbst wird abgeschlossen mit einer Prädiktion, die bestimmt, wie lange Sicherheitsgarantien gegeben werden können.

In OMC werden Eigenschaften des Gesamtsystems als wiederholte Nachweise über Teilsysteme des fortlaufend angepassten Gesamtsystems erbracht. Die explizite Modelltransformation erlaubt Simulationen, die dem Fachpersonal das Nachvollziehen der Verifikation ermöglichen.

Literaturverzeichnis

[AR16] Antoni,S.-T.; Rinast,J; Ma,X; Schupp S; Schlaefer, A.: Online model checking for monitoring surrogate-based respiratory motion tracking in radiation therapy. Int. Journal of Computer Assisted Radiology and Surgery, Bd. 11, pp. 2085-2096, 2016.

[PS16] Padberg J; Schulz, A.: Model Checking Reconfigurable Petri Nets with Maude, in Graph Transformation, 9th Int. Conf. on, LNCS, Bd. 9761, Springer, pp. 54-70, 2016.

Shifting Quality Assurance of Machine Learning Algorithms to Live Systems

Florian Auer[1] Michael Felderer[1]

The behavioral adaptability of machine learning algorithms makes the assurance of their quality challenging (e.g. previous inputs affect future processing). The complex problems tackled by these algorithms complicate the assurance techniques. Testing, for example, suffers from the situation that in most cases only costly or even no test oracles are available [Ba15]. Solutions presented in literature require additional effort in testing. For instance, the use of pseudo-oracles [MKA06] that require an additional, independent implementation or metamorphic testing [MSK09] that involves the manual finding of meaningful metamorphic properties and execution of additional test cases.

A fundamental weakness of existing solutions is the assumption that test environments sufficiently mimic the later application in order to allow quality assurance. However, given the data dependent behavior of these algorithms, only limited reasoning about their later performance is possible. Thus, meaningful quality assurance is not possible with test environments exclusive. A shift from the traditional testing environment to the live system is needed. Hence, costly test environments for simulation are replaced with available live systems that constantly execute the algorithm – in some cases even supervised by humans.

In order to observe an algorithm in live systems, its implementation has to be prepared for monitoring and the collection of the measured data has to be organized. Similar requirements are found for the evaluation of experimental features. *Live experimentation* is the controlled deployment of experimental features (like a new learning algorithm) in live systems by instrumenting the program, collecting runtime data and analyzing the affect [Fa14].

This kind of experimentation serves as a vehicle to enable quality assurance in live systems. The observation of the algorithm at runtime allows to evaluate quality attributes like functionality (e.g. unsatisfying results), reliability (e.g. frequency of failures), usability (e.g. user feedback) or efficiency (e.g. execution time). Furthermore, information about the input and output data can be collected. Given the behavioral dependency of these algorithms on data, its analysis can reveal valuable information about the data (e.g. value space, frequency of values) and about the related behavior of the algorithm (e.g. clusters of inaccurate output, learning behavior). This results into a deeper understanding of the algorithms and serves as an initial step towards improved explainability of machine learning algorithms.

Thus, it seems that live experimentation is suitable for quality assurance. Nevertheless, it requires further research on how to find the appropriate metrics, instrument the implementation, deploy the implementation (e.g. it is not desirable to deliver experimental/insufficient tested algorithms to all customers) and analyze the results. In addition, the immature state of

[1] University of Innsbruck, Quality Engineering, Innsbruck, Austria, firstname.lastname@uibk.ac.at

practice for live experimentation requires further research on a taxonomy of experimentation types along with patterns that represent best practices on the implementation of experiments. This would support initial decision making and give practitioners a clear guidance [Fa14].

Many of the traditional testing techniques are still applicable in live systems. For example, regression testing (e.g. between versions), invariant checking (e.g. in combination with automatic invariant detection [Ba15]), manual testing (e.g. feedback from users) or metamorphic testing (e.g. execution of input variants in background [MSK09]). However, the implications of their application in a live system are not well understood (e.g. performance impact). Thus, an evaluation of testing techniques for the use in live systems is necessary.

With the transition to the live system, testing is conducted implicitly by the users of the system. As a consequence, the test data is actual data from the users. This requires a different data management as in traditional testing environments. For instance, data may be confidential or business-critical. Another aspect that requires attention is user satisfaction in the presence of continuous change and less mature algorithms. Furthermore, ethical aspects have to be considered (e.g. wrong credit score assessment). As similar challenges are faced in crowdtesting, existing approaches for crowdtesting may be adaptable.

Finally, this approach to quality assurance of machine learning algorithms is not always applicable. For instance, safety-critical software has to provide reliable services. Deploying insufficient tested algorithms to such systems is dangerous and may lead to serious damage. In contrast, other applications (like suggestion services) can provide wrong results without serious consequences. Thus, research about techniques to support application cases that require correct results is necessary to broaden the applicability of this approach. Another challenge that requires further research is that not always a definition for correct output exists (e.g. output is approximation). As a result, the collected feedback may be inconsistent.

In order to provide an environment that allows further conclusions about this approach, the next step would be a software quality assurance analysis platform that allows to plan, prepare, conduct and analyze systematic deployments of instrumented algorithms.

References

[Ba15] Barr, Earl T; Harman, Mark; McMinn, Phil; Shahbaz, Muzammil; Yoo, Shin: The oracle problem in software testing: A survey. IEEE transactions on software engineering, 41(5):507–525, 2015.

[Fa14] Fagerholm, Fabian; Guinea, Alejandro Sanchez; Mäenpää, Hanna; Münch, Jürgen: Building blocks for continuous experimentation. In: Proceedings of the 1st International Workshop on Rapid Continuous Software Engineering. ACM, pp. 26–35, 2014.

[MKA06] Murphy, Christian; Kaiser, Gail; Arias, Marta: A Framework for Quality Assurance of Machine Learning Applications. -, 2006.

[MSK09] Murphy, Christian; Shen, Kuang; Kaiser, Gail: Automatic system testing of programs without test oracles. In: Proceedings of the eighteenth international symposium on Software testing and analysis. ACM, pp. 189–200, 2009.

Erklärbare Software 2 - Understandable Decisions

M. Tichy, E. Bodden, M. Kuhrmann, S. Wagner, J.-P. Steghöfer (Hrsg.): SE 2018,
Lecture Notes in Informatics (LNI), Gesellschaft für Informatik, Bonn 2018 215

Comprehensible Decisions in Complex Self-Adaptive Systems

Verena Klös[1], Thomas Göthel[2], Sabine Glesner[3]

Abstract: To cope with uncertain and statically unforeseen environment behaviour of complex systems, self-adaptivity has gained wide acceptance. While adaptation decisions are required to be close to optimal decisions, they at the same time should be efficient, comprehensible, and reusable. To achieve this, we have developed an engineering and analysis approach for self-learning self-adaptive systems based on our notion of timed adaptation rules. Through continuous evaluation and learning, inaccurate rules can be improved and new rules can be learned at run-time to cope with changing environments and system goals. A separate verification phase enables us to provide offline and online guarantees of evolving adaptation logics based on human-comprehensible formal models. Our approach, which incorporates the precise retracing of previous adaptation decisions, enables the understanding of the contexts in which certain adaptation decisions have been made, and assessing whether they have gained their expected effect in time within the system. This comprehensibility of complex decisions in self-adaptive systems enables the precise understanding and reuse of adaptation logics and provides trust in autonomous decision making.

Keywords: Self-Adaptivity; Adaptation Rules; Self-Learning; Comprehensible Adaptation Decisions

Self-adaptivity realised through *MAPE-K* feedback loops [IB04] is a prominent approach to cope with changing environment behaviour and system goals in complex dynamically evolving systems. Such self-adaptive systems separate a managed system from an adaptation layer that continuously *M*onitors the managed system and the environment behaviour, *A*nalyses the need for adaptations, *P*lans suitable adaptation strategies, and *E*xecutes the adaptation plan within the managed system. These MAPE components share a common *K*nowledge base, which they continuously update. While this reference architecture provides abstract means to self-adaptivity, it neither provides concrete knowledge models nor makes adaptation logics itself subject to dynamic evolution.

In [KGG15], we present our approach for self-learning self-adaptive systems, which imposes a structured knowledge base consisting of system and environment parameters (K_{Sys}, K_{Env}), a hierarchical goal model (K_{Goal}), and an adaptation logic (K_{Adapt}) based on timed adaptation rules of the following form: $r : g$ **&** $c_1; c_2; \ldots; c_n \longrightarrow$ *effect* **after** *time*. A rule r is applicable if its guard g is satisfied. If it is chosen by the planner, its commands $c_1, ..., c_n$ are applied to the managed system. After a certain amount of *time*, an *effect* is assumed to be observable. We provide a further (meta-)adaptation layer that consists of an *evaluation* of previously applied adaptation rules, *learning* of improved and new

[1] TU Berlin, Softw. and Embedded Syst. Eng., Ernst-Reuter-Platz 7, 10587 Berlin, verena.kloes@tu-berlin.de
[2] TU Berlin, Softw. and Embedded Syst. Eng., Ernst-Reuter-Platz 7, 10587 Berlin, thomas.goethel@tu-berlin.de
[3] TU Berlin, Softw. and Embedded Syst. Eng., Ernst-Reuter-Platz 7, 10587 Berlin, sabine.glesner@tu-berlin.de

adaptations rules, and *verification* of the current adaptation logic. For analysis, planning, and learning, we employ distance functions, which we automatically extract from a goal model K_{Goal} [Kl17]. They capture the "distance" from a system context to the system goals.

In the literature, especially analysis and planning for self-adaptive systems have been realised using diverse methods. While techniques such as online planning enable the calculation of optimal plans for a current system context, the underlying decision-making is hard to comprehend. This dramatically limits reusability of adaptation logics. In contrast, timed adaptation rules make adaptation decisions explicit and comprehensible due to their explicit application condition and timed expected effect. At run-time, we employ history information of system and environment parameters together with history information of applied adaptation rules to retrace past adaptation decisions and record whether their expected effect was achieved and in which system contexts. We use this information to evaluate the accuracy of adaptation rules and to dynamically improve the adaptation logic through learning-based mechanisms. As a result, we achieve an updated adaptation logic that consists of more accurate, yet comprehensible, timed adaptation rules. For our verification approach, we extract formal timed automata models from a (possibly partial) system implementation in SystemC [KGG16]. At run-time, we update them using actual environment and system parameters as well as current adaptation rules. This enables us to apply model checking to verify general properties, such as stability of the (updated) adaptation logic. As our formal models are human-comprehensible due to structure-preserving transformations, we not only employ them for offline and online verification, but also for analysing and understanding system and adaptation behaviour throughout the entire system lifecycle.

In summary, our approach for self-learning self-adaptive systems integrates a) comprehensible timed adaptation rules, b) precise retracing of previous adaptation decisions and accuracy evaluation w.r.t. their effects, c) learning-based optimisation of adaptation logics, and d) their analysis based on human-comprehensible models. As a result, we achieve comprehensibility and reusability of complex, dynamically evolving adaptation logics and, thus, provide a basis for trust in autonomous decision-making.

References

[IB04] IBM Corp.: An architectural blueprint for autonomic computing. IBM Corp., 2004.

[KGG15] Klös, Verena; Göthel, Thomas; Glesner, Sabine: Adaptive Knowledge Bases in Self-Adaptive System Design. In: 41st Euromicro Conference on Software Engineering and Advanced Applications (SEAA). IEEE, pp. 472–478, 2015.

[KGG16] Klös, Verena; Göthel, Thomas; Glesner, Sabine: Formal Models for Analysing Dynamic Adaptation Behaviour in Real-Time Systems. In: 3rd Workshop on Quality Assurance for Self-adaptive, Self-organising Systems (QA4SASO). IEEE, pp. 106–111, 2016.

[Kl17] Klös, Verena; Göthel, Thomas; Lohr, Adrian; Glesner, Sabine: Runtime Management and Quantitative Evaluation of Changing System Goals. In: 43rd Euromicro Conference on Software Engineering and Advanced Applications (SEAA). IEEE, pp. 226–233, 2017.

M. Tichy, E. Bodden, M. Kuhrmann, S. Wagner, J.-P. Steghöfer (Hrsg.): SE 2018,
Lecture Notes in Informatics (LNI), Gesellschaft für Informatik, Bonn 2018 217

Erklärungen (nur) bei Bedarf

Kurt Schneider[1]

1 Was muss erklärt werden?

Software ist kompliziert. Es ist schwierig, zu verstehen, wie sie funktioniert und wie sie bestimmte Resultate erzielt oder wie sie zu einer automatisierten Entscheidung gelangt. In diesem Beitrag wird die These vertreten, dass es dennoch nicht sinnvoll ist, möglichst viele Informationen für mögliche spätere Erklärungen zu sammeln oder zu generieren, denn ein großer Teil davon wird nie benötigt. Erklärbarkeit mag ein wichtiger Softwarequalitätsaspekt sein; sie ist aber dennoch eine „Sekundärtugend". In sie sollte man nur Aufwand und Zeit investieren, wenn der absehbare Nutzen dies auch rechtfertigt. Es ist wichtig, zunächst den angestrebten Nutzen genau zu verstehen.

Die erste Frage bei der Gestaltung verbesserter Erklärbarkeit lautet daher: Auf welche Arten von Erklärungen soll die Software vorbereitet sein? Denn es gibt verschiedene Typen von Fragen, zu denen man eine Erklärung sucht: (a) Wie wurde das Resultat aus einer Regel (einem Algorithmus, einem Neuronalen Netzwerk usw.) abgeleitet? (b) Wie lautet die dort umgesetzte Regel anschaulich und verständlich? (c) Wieso wurde gerade diese Regel gewählt? Wer eine Antwort auf (b) oder (c) sucht, dem ist mit einem detaillierten Trace eines komplexen Algorithmus (a) nicht gedient.

2 Analogie: Wissen aus Erfahrungen und Wissen für Erklärungen

Vor einigen Jahren hat Basili die Experience Factory vorgeschlagen. Unternehmen versuchten, systematisch aus ihren eigenen Erfahrungen zu lernen. Erfahrungsnutzung und zukünftige Nutzung von Wissen in Erklärungen haben einiges gemeinsam: In Basilis Experience Factory sammelte eine Gruppe in Unternehmen Messdaten und quantitative Erfahrungen, prüfte die Anwendbarkeit auf neue Projekten und lieferte übertragbare Erfahrungen. Zunächst versuchte man, mit Werkzeugen das Wissen aus Erfahrungen effizient zu verwalten und zugänglich zu machen. Das war aber ein Irrweg.

Die eigentliche Herausforderung lag nämlich darin, solche Erfahrungen zu identifizieren und zu erfassen, *die später auch wirklich gebraucht wurden*. Es gibt einen Lebenszyklus für Erfahrungen, an dem bemerkenswert ist, dass Erfahrungsträger erst einmal erkennen müssen, welche ihrer Erfahrungen für andere wichtig sein könnten. Übertragen auf die erklärbare Software sind wohl Mechanismen nötig, um komplizierte Abläufe zu verfolgen (tracen); es könnte aber viel wichtiger sein, diese detaillierten Abläufe zu verdichten

[1] Leibniz Universität Hannover, FG Software Engineering, Welfengarten 1, 30167 Hannover
kurt.schneider@inf.uni-hannover.de

und nach ihrem Sinn zu hinterfragen. Das erfordert vielleicht nicht so detailliertes Tracing, dafür aber komplementäre Begründungen (Rationale) und Anforderungen. Sie sind für Fragen der obigen Typen (b) und (c) unverzichtbar. Beobachtungen, Traces und Wissen können zudem – wie Erfahrungen - nicht „roh“ genutzt werden, sondern müssen zunächst geeignet aufbereitet werden, bevor sie anderswo und von anderen Personen genutzt werden können. Man darf sich nicht darauf beschränken, das zu sammeln, was l zugänglich ist, wie beispielsweise Traces – sondern das, was tatsächlich gebraucht wird. Das hängt auch von der Aufgabe dessen ab, der die Erklärung sucht.

3 Vision: Pragmatischer Ansatz - mit Grenzen

Ein *idealistischer Ansatz* könnte darin bestehen, alle Veränderungen am Code zu verfolgen und seinen Ablauf durch Instrumentierung zu tracen Ich plädiere jedoch stattdessen für einen *pragmatischen Ansatz*, in dem Aufwand gezielt reduziert und erst dann aufgewendet wird, wenn der Bedarf entsteht. Beispielsweise würde man also generierbare Erklärungen erst dann erzeugen, wenn sie abgerufen werden. Hier kann man von der systematischen Erfahrungsnutzung lernen und unorthodoxe vorgehen: Zunächst gibt es keine Erklärung; wer sich zuerst fragt, wieso hier ein bestimmter Algorithmus eingesetzt wird, muss seine Erkenntnisse nachher dokumentieren. Denn in diesem Fall war die Information ja schon einmal nützlich, und es lohnt sich, sie in diesem Kontext für künftige Fälle bereitzuhalten. Wird sie dagegen nie gebraucht, muss man auch keine Erklärung vorhalten. Es wäre unverhältnismäßig aufwändig, jede denkbare Erklärung schon bei der Entwicklung abzufragen oder abzuleiten, solange man noch gar nicht weiß, welcher Typ von Erklärung (z.B. a, b, c), für wen und für welchen Zweck gebraucht werden wird.

Dabei zeigen sich jedoch die Grenzen auch des pragmatischen Ansatzes: Oft stecken die gewünschten Informationen gar nicht in der Software und ihrem Verhalten selbst. Es handelt sich vielmehr um eine Kombination aus Anforderung, Entwicklungsprozess und Entscheidungen, die nur zusammen mit dem Algorithmus verständlich sind. Algorithmen werden im Verlauf von Änderungen in Anforderungen und Umfeld schrittweise angepasst, Fragen nach Sinn und Begründung antworten typischerweise auf Fragen der Typen (b) oder (c). Eines der größten Probleme für die Erklärbarkeit dürfte darin liegen, genau an dieses Hintergrundwissen heranzukommen, ohne dabei in eine übertriebene und teure Sammelwut abzugleiten. Man muss Hintergrundwissen „retten“, bevor die Wissensträger es vergessen oder nicht mehr im Unternehmen arbeiten. Man muss neue Mechanismen entwickeln, um das Wissen aus dem System von Mensch und Software zu retten. Es lässt sich sicher nicht nur aus der Software allein ableiten.

Ein Rezept für eine perfekte Balance gibt es noch nicht. In diesem Vortrag werden exemplarisch einige praktische Vorschläge gemacht, wie man mit Hintergrundwissen umgehen und es (erst) zum geeigneten Zeitpunkt erheben kann. Dazu gehört die effiziente Wissenserschließung aus Prototypen oder die Verwendung von Visionsvideos zur anschaulichen Aufbereitung von Sinn und Zweck von Anforderungen.

Generating Explanations for Algorithmic Decisions of Usage-Based Insurances using Natural Language Generation

Daniel Braun[1], Florian Matthes[2]

Abstract: Usage-based insurances are becoming more and more popular, especially for cars. We present an approach to use Natural Language Generation (NLG) in order to explain customers which aspects of their behaviour influenced the assessment which is performed by these insurances.

Keywords: Explainable AI; Natural Language Generation; Telematics

1 Usage-based insurances

So-called telematic insurances use different sensors installed in a car to track the individual driving style of the driver. Instead of calculating insurance premiums based on statistical risk groups, insurance companies can use these data to create individual risk profiles and calculate insurance premiums accordingly. While at the moment, most of the usage-based insurances are car insurances, it is expected that in the future other areas will follow, e.g. health insurances with premiums calculated based on data from wearables and fitness trackers. In the UK, usage-based car insurances are more popular and it is expected that nearly 40% of all car insurances will be telematic by 2020. [Ro13] In Germany, however, customers are still wary of usage-based insurances. Nevertheless, these insurances offer customers who belong to a high-risk group, like young male drivers, the opportunity to save money. We believe that explaining to customers how their individual risk is assessed by algorithms may help to increase acceptance for the assessment. Moreover, the customer could benefit from his own data, by getting advice on how to adapt behaviour in order to decrease insurance premiums.

Some insurances, like AXA Drivesave, already give feedback to their customers in form of scores between 0 and 100 in categories like “Smoothness” and “Calmness”. However, they give no explanation whatsoever how these scores are calculated. Not surprisingly, Braun et al. found out that this form of feedback is not perceived as helpful by drivers and they would prefer textual feedback. [BRS15] Moreover, in order to not only report, but (positively)

[1] Technical University of Munich, Department of Informatics, Boltzmannstraße 3, 8574 Garching, Germany
daniel.braun@tum.de
[2] Technical University of Munich, Department of Informatics, Boltzmannstraße 3, 8574 Garching, Germany
matthes@tum.de

influence behaviour, textual feedback is more useful and its psychology is well studied. [HT07]

2 Prototype

We have built a prototype (cf. [BRS15]) that follows the architecture for Data-2-Text systems described by Reiter [Re07], which consists of four stages: Signal Analysis, Data Interpretation, Document Planning and Microplanning and Realisation. For the Signal Analysis stage, we tried to mimic the algorithms used by insurance companies to create risk profiles for drivers. While their exact metrics are secret, we used the available information [Hä14] to get as close as possible. During Data Interpretation, we try to cluster incidents which lead to a reduced score (i.e. higher premiums) by features that can be understood by the customer and also give advice on how to change the behaviour in order to increase the score in the future. Figure 1 shows an example output text from the prototype. In a study we conducted, six drivers used the prototype over the course of one month. At the end of the study, all six drivers had reduced the number of speeding incidents per kilometre and for out of six drivers also reduced the number of acceleration incidents per kilometre and hence improved their driving score.

Driving Report 19 - 25 January

You drove **390 miles in 10 hours and 50 minutes** during the last week. You reduced the number of speeding incidents per mile by **more than 10 %**, well done!

You **accelerated or braked harshly 645 times**, mostly on **highways** and on **roads with 20 mph speed limit**. **Five times you drove more than 30 mph too fast**: On Castle Road, on Kirkton Road, on North Deeside Road and twice on A92. Going 30 mph slower could shorten your braking distance by 108 yards.

You also **speeded on 175** other occasions, 7 times on **roads with 20 mph speed limit** and 12 times **on weekends on roads with 30 mph speed limit**.

Fig. 1: Generated text

References

[BRS15] Braun, Daniel; Reiter, Ehud; Siddharthan, Advaith: Creating Textual Driver Feedback from Telemetric Data. In: Proceedings of the 15th European Workshop on NLG. pp. 156–165, 2015.

[Hä14] Händel, Peter; Skog, Isaac; Wahlström, Johan; Bonawiede, Farid; Welch, Richard; Ohlsson, Jens; Ohlsson, Martin: Insurance telematics: Opportunities and challenges with the smartphone solution. Intelligent Transportation Systems Magazine, 6(4):57–70, 2014.

[HT07] Hattie, John; Timperley, Helen: The power of feedback. Review of educational research, 77(1):81–112, 2007.

[Re07] Reiter, Ehud: An Architecture for Data-to-Text Systems. In: Proceedings of the 11th European Workshop on NLG. Saarbrücken, Germany, pp. 97–104, June 2007.

[Ro13] Rose, Stuart: Telematics: How Big Data Is Transforming the Auto Insurance Industry. Technical report, 2013.

Erklärbare Software 3 - Understandable Software

"What Does My Classifier Learn?" A Visual Approach to Understanding Natural Language Text Classifiers

Jonas Paul Winkler[1], Andreas Vogelsang[2]

Abstract: Neural Networks have been utilized to solve various tasks such as image recognition, text classification, and machine translation and have achieved exceptional results in many of these tasks. However, understanding the inner workings of neural networks and explaining why a certain output is produced are no trivial tasks. Especially when dealing with text classification problems, an approach to explain network decisions may greatly increase the acceptance of neural network supported tools. We have developed an approach to visualize reasons why a classification outcome is produced by convolutional neural networks by tracing back decisions made by the network. The approach is applied to various text classification problems, including our own requirements engineering related classification problem. We argue that by providing these explanations in neural network supported tools, users will use such tools with more confidence and also may allow the tool to do certain tasks automatically.

Keywords: visual feedback, neural networks, artificial intelligence, machine learning, natural language processing, explanations, requirements engineering

1 Introduction

Artificial Neural Networks have become powerful tools for performing a wide variety of tasks such as image classification, text classification, and speech recognition. Recently, convolutional neural networks that were almost exclusively used for image processing tasks were also adapted to solve natural language classification tasks [Ki14]. However, neural networks usually do not explain why certain decisions are made. Especially when integrating neural networks into tools, users may not understand certain network decisions and consequently do not use the tool.

In [WV17], we proposed a technique to trace back decisions made by convolutional neural networks for text classification. We use this technique to create visual explanations by highlighting most important words in an input sentence. In this paper, we present a brief overview of the technique and its applications.

[1] Technische Universität Berlin, jonas.winkler@tu-berlin.de
[2] Technische Universität Berlin, andreas.vogelsang@tu-berlin.de

2 Tracing Back Network Decisions

Convolutional Neural Networks for text classification as described in [Ki14] classify examples by applying a set of filters to a sentence matrix (i.e., a sentence transformed into a numerical representation using word embeddings) and associating individual filters with output classes using fully connected layers. At each step, itermediate values are calculated, representing whether learned features have been detected or not. High intermediate values indicate that a feature was detected, and thus imply that a certain input part was important for deciding towards or against classifying an input example as a certain output class. We utilize these phenomena to calculate *Document Influence Matrices*. These matrices indicate which input elements were important for deciding the class of an input example.

Tab. 1: Examples

positive	both a successful adaptation and an enjoyable film in its own right .
negative	just a bunch of good actors flailing around in a caper that's neither original nor terribly funny .

Visual representations may be created based on these matrices. An example is provided in Tab. 1. A convolutional neural network has been trained on a dataset containing positive and negative movie reviews. Our tracing technique shows why these examples have been classified as positive and negative. Example 1 contains word groups („successful“, „an enjoyable film“, „its own“) usually accociated with positive sentiment, whereas example 2 contains word sequences with negative sentiment („neither ... nor terribly funny“).

3 Applications & Conclusions

At our industry partner, requirements documents are created to document the expected behavior of automotive systems. These documents contain both requirements and non-requirements (i.e., explanations, examples, references). A strict separation between these two classes is required. Requirements engineers usually classify the contents of a document manually. We have built a tool, incorporating the approach presented above to assist RE experts in this task.

We assume that by providing explanations, users may better understand the decisions made by the tool and consequently perform the given task more effectively and efficiently.

References

[Ki14] Kim, Yoon: Convolutional Neural Networks for Sentence Classification. In: Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP). pp. 1746–1751, 2014.

[WV17] Winkler, Jonas P.; Vogelsang, Andreas: "What Does My Classifier Learn?" A Visual Approach to Understanding Natural Language Text Classifiers. In: Proceedings of the 22nd International Conference on Natural Language & Information Systems. NLDB, 2017.

M. Tichy, E. Bodden, M. Kuhrmann, S. Wagner, J.-P. Steghöfer (Hrsg.): SE 2018,
Lecture Notes in Informatics (LNI), Gesellschaft für Informatik, Bonn 2018 225

UI-Tracer: A Lightweight Approach to Help Developers Tracing User Interface Elements to Source Code

Regina Hebig[1]

Abstract: The ability to understand software systems is crucial to identify hidden threats or maintain software systems over many years. Still software comprehension activities take up around 58% of software development time. While most approaches support the comprehension of a software system's code perspective, its connection to the user perspective is barely explored. We present UI-Tracer, a lightweight support for tracing user interface elements to source code using the version history of a system. The evaluation on two open source systems shows that the approach can cover all UI elements that have been changed or added within the accessible part of the version history. Furthermore, the median numbers of files flagged as potentially responsible for a UI element is 8 and 3 for the two studied systems. Thus, UI-Tracer provides an easy starting ground for developers to identify files relevant for future UI changes.

Keywords: Software Comprehension; UI Tracing

1 Introduction

One of the key ideas behind open source systems is that they enable users and experts to inspect the source code, making it more difficult to build in hidden threats. Despite the openness it is, however, not trivial to read, comprehend or explain source code. Even experienced software developers have to invest time if they want to understand a new system. Also systems with long live-spans need to be understood over the years over and over again by new developers and maintainers. While there are approaches that can be used to easier understand a software system, until today, software comprehension activities take up around 58% of software developers' time [Xi17, Fj83, Ti11]. Also, developers who know a system have little tool support, when explaining the source code to new developers.

Related Work Most existing works towards software comprehension can be split into code-level and architecture level approaches. Thus, there are coding conventions, e.g. Oracle's Java Coding Conventions[2], code annotations, as well as approaches for stepwise execution/simulation, e.g. [DL00] and debugging, e.g. [ZL96]. Some approaches aim at automatically recovering trace links in source code [An02]. Furthermore, there is the

[1] Chalmers | University of Gothenburg Universität, Software Engineering Division, Hörsegången 11, 412 96 Göteborg, Sweden hebig@chalmers.se

[2] Oracle's Java Coding Conventions`http://www.oracle.com/technetwork/java/index-135089.html`

research area of code visualizations, that focuses on illustrating metrics about building blocks, such as number of lines of code or complexity, as well as relations between these building blocks. Examples are code cities [WL08] and circular bundle views [TBD12]. Other approaches aim at reverse engineering models from code [Ch08], generation of documentations, e.g. with Doxygen[3] and approaches to summarize software in natural language [Bi13]. Nearly all of these techniques remain at representing the source code perspective on the system. The resulting visualizations, simulations, models, documentation, and explanations, reflect the structure of the code and its terminology, such as class names. However, it can be argued that each software system has another fundamental perspective: the user perspective. So far there is no approach that provides a bridge between user and source code perspective.

UI-Tracer Research has shown that professional developers who have to comprehend a new piece of code try to connect knowledge about the user interface with knowledge about the code [Ro12]. However, so far there is no approach that supports developers in making these connections. On the other hand, it might take even experienced developers some effort to reproduce and document these traces, when explaining source code to novices.

In this paper, we ask the following questions: How can an automated approach help users to identify source code that has impact on a user interface element of interest? We present the UI-Tracer, a lightweight approach that analyzes the UI of a system throughout its version history, and identifies files that were changed together with observed changes in the UI. This way the UI-Tracer can provide developers who want to learn about the code connected to an UI element, with a starting set of files. While not necessarily all of the files are relevant for the UI element, the value lies in the limitation of choices, making it easier to find the right files. We evaluate the approach based on two open source systems with regards to the questions a) whether all UI elements can be covered with the approach and b) how small, i.e. precise, are the sets of identified files. The UI-Tracer approach is presented in Section 2. In Section 3, we evaluate the approach and discuss the results. Finally, in Section 4 we conclude and discuss future work.

2 The UI-Tracer

The basic idea of the UI-Tracer[4] approach is that the following two questions have the same answers: “What code is responsible for a UI element?” and “What code causes changes of a UI element when changed?”. Thus, when tracing UI elements to code, those classes are of interest that can change a UI element. Furthermore, change happens during the development of a system, where each element is at one point or another introduced.

Therefore, this approach uses the granularity of commit to a project for the tracing. Analyzing how UI and source code change commit by commit. The three technical pillars of this

[3] Doxygen `http://www.stack.nl/~dimitri/doxygen/`
[4] The UI-Tracer prototype can be found here `https://github.com/rhebig/UI-Tracer`

approach are: online open source repositories, e.g. GitHub, technologies for automatic compilation, e.g. ANT, and user interface scripting languages and image comparison techniques, such as Sikuli [YCM09]. In the following subsections it will be explained how the UI-Tracer works and how it can be adapted to analyzing new systems.

2.1 Approach and Prototype

Figure 1 provides an overview of the approach, which consists of two halves. A class diagram of the UI-Tracer can be seen in Figure 2.

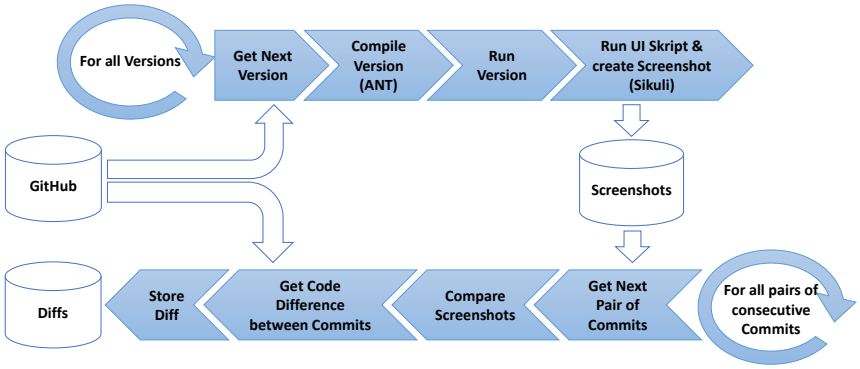


Fig. 1: Overview about the approach

In the ***first half***, the repository of the system to be analyzed is cloned and then commit by commit reverted back to former versions (performed by the class *CloneRemoteRepository* shown in Figure 2). For each commit/version, the UI-Tracer compiles the software to create an executable, e.g. a jar (performed by the class *Builder* shown in Figure 2). This is done at the moment for Java using Ant. However, since, the build process is the only step the approach that is specific to the programming language. In future, the prototype can be extended at this point to allow for other technologies, such as Gradle or Maven, and with it further programming languages, e.g. C++ or python.

After the executable is created, the UI-Tracer (class *BuildAndScreenshotCoordinator* shown in Figure 2) uses a process builder to execute the compiled system. Once the system is running the UI-Tracer executes some random mouse movement. This is done to make sure that random UI effects, as they sometimes occur with just started systems, disappear. Then a small standard routine is executed to enlarge the systems window to full screen. This is done to ensure that the collected images are comparable. Afterward the first screen-shot is done.

In the next step a customized Sikuli script is executed. This script is typically created based on the most current version of the system and can for example click and hover over all main menu items to make sure that the sub-menu items are shown as well. During that execution extra screen-shots are taken and stored. It turned out to be a crucial step to terminate the running system after making all relevant screen-shots. Otherwise the machine will be

overloaded quickly and slow down dramatically. Therefore, the UI-Tracer kills the started process before downloading a new version of the system from Github.

In the ***second half*** the class *ComparingScreens* (Figure 2) iterates over all pairs of consecutive commits. Note that it is possible that single versions cannot be compiled. One possible reason for that is that a commit in the repository introduced faulty or breaking code, which was corrected in a later commit. To compensate for that two commits are considered consecutive, if screen-shots could successfully be obtained for both commits and if there are no commits in between them for which screen-shots could be applied.

For each pair, the initial screen-shots are subject to an automated image comparison. Similarly, for each screen-shot made during the execution of the customized Sikuli script a comparison between the two commits is performed. If a pair of screen-shots is marked as different, by the automated comparison, the UI-Tracer will start to use GitHub to retrieve the code difference between the involved commits (performed by class *CodeDiffTracer* shown in Figure 2). In doing so, the UI-Tracer considers all changes of all commits that took place in between the considered pair of commits (to cover for cases where some commits could not be successfully compiled). Finally, the ids of those commits are stored together with references to the two screen-shots and the list of files in the repository that have been modified, added, or deleted in between the two commits.

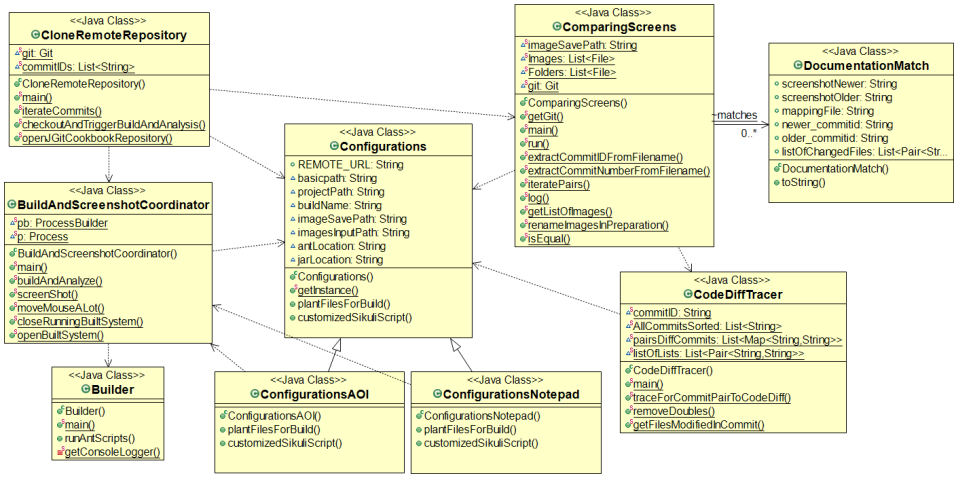


Fig. 2: Class Diagram of the UI-Tracer

2.2 Configuring the Tool to Analyze a new Software System

At the moment the UI-Tracer uses a primitive configuration class to enable its application to new systems. For a new system to be studied a new class can be created that inherits from the class *Configurations* (as shown in Figure 2 for the two example systems AOI and Notepad). Then the *getInstance()* method in class *Configurations* should be changed to return the instance of the newly created class.

Plant Files for Build While some systems already provide an automated Ant script, others do not. Therefore, it might be necessary to plant the Ant script and used libraries, e.g. jar files, into the system. Depending on the system, very old version might have slight differences that need to be addressed in the Ant script, e.g. a differently named main class. A typical implementation of the plantFilesForBuild(Integer buildCount) method is shown in Listing 1. Here another Ant script is planted when the commit to be built is 80 commits older than the current version (or more). The Ant script and relevant libraries are thereby stored in a folder *FilesToPlant* of which all content is copied to the system to be compiled.

```
public void plantFilesForBuild(Integer buildCount) {
        File srcDir;
        if(buildCount<80)
                srcDir = new File(basicpath + "FilesToPlant");
        else
                srcDir = new File(basicpath + "FilesToPlant_2");
        File destDir = new File(projectPath);
        try {
                FileUtils.copyDirectory(srcDir, destDir);
        }catch(Exception e) {System.out.println(e);}
}
```

List. 1: Typical implementation of plantFilesForBuild

Customized Sikuli Script The customized sikuli script can be added here. Listing 2 shows an example of such a script with a typical building block. The parameters *screenRect*, *commitNumber*, and *commitID* are there to ensure that the screen-shot captures the relevant region of the screen and that it is saved in association to the current commit. These parameters are only used when the *BuildAndScreenshotCoordinator* is called to take a screen-shot and can otherwise be ignored. When calling the latter a fourth parameter should be used to define a subcategory of screen-shots, so that the automated comparison can later on compare the right screen-shot with each other. In the example, the subcategory is set to *DropDown* by defining the folder "DropDown\\" as target for saving the screen-shot.

```
public void customizedSikuliScript(Rectangle screenRect, Integer
    commitNumber, String commitID) {
        Screen s = new Screen();
        try {
                s.click(imagesInputPath+"DropDown.png");
                TimeUnit.SECONDS.sleep(1);
                BuildAndScreenshotCoordinator.screenShot(screenRect,
                    commitNumber, commitID, "DropDown\\");
                s.type(Key.ESC);
        }catch(Exception e) {System.out.println(e);}
}
```

List. 2: Typical building block of the customized sikuli script

Tab. 1: Paths to be set

Variable	
REMOTE_URL	the url of the repository that contains the system that should be analyzed
basicpath	a basis path to the working directory
projectPath	(extension to basicpath) the path where the system will be cloned to
buildName	the name of the build
imageSavePath	(extension to basicpath) the path to the folder where screen-shots will be stored
imagesInputPath	(extension to basicpath) the path to the folder where images are stored that are used in the customized sikuli script
antLocation	(extension to projectpath) the path to the xml file specifying the ant script
jarLocation	(extension to projectpath) the path to the executable that will be created when running Ant

Fig. 3: Image of the DropDown to be clicked in listing 2

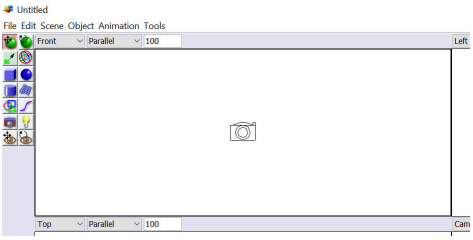
Another important aspect is that Sikuli allows users to define the target of a click or hover action by providing an image of the region or button. By matching the image to the screen, Sikuli is robust against factors, such as absolute positioning of user interface elements. Figure 3 shows the image used in the example. Other actions that are useful are sleeping actions to give the program time to react to an action, e.g. after a click, and escape actions to go back to the starting screen, e.g. by pressing the ESC key programmatically. This way the script can also be used to navigate between multiple views of a UI.

Paths and URLs Finally, the constructor of the new subclass of *Configurations* should set all relevant paths and URLs, as listed in Table 1.

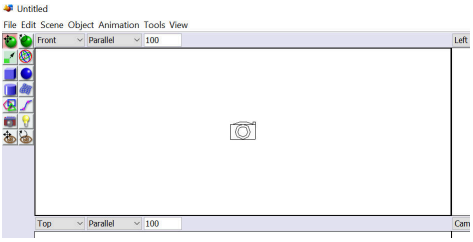
2.3 Example Output and Envisioned Use

An example for the output can be seen in Figures 4 and 5. The figures show parts of screen-shots that UI-Tracer made for the Art Of Illusion (AOI) software, compiled 77 commits before the current version and 78 commit before the current version. Note, that screen-shots from the initial screen show that a new menu item was added in commit 77 that has not yet been there after commit 78: *View*. Furthermore, the Figure shows the screen-shots done after clicking/hovering over the buttons *Scene* and *View*. Observe that several sub-menu items from the menu item *Scene* in commit 78 are moved to the new menu item *View*. UI-Tracer reports that only one file was modified in commit 77: "src/artofillusion/LayoutWindow.java". Thus, a user of UI-Tracer can conclude that the definition of menu items and sub-menu items is done in that file.

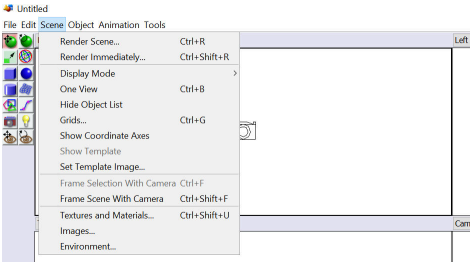
The current version of the UI-Tracer can be used by developers who are interested in learning about an open source system's implementation, e.g. in order to make future contributions. The images can be used as an entry point to identify what files are responsible for certain UI elements.



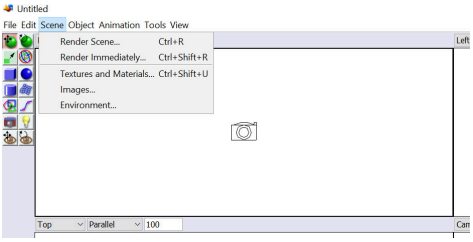
(a) AOI 78 commits before current version, initial screen



(b) AOI 77 commits before current version, initial screen



(c) AOI 78 commits before current version, clicking *Scene* menu



(d) AOI 77 commits before current version, clicking *Scene* menu

Fig. 4: Example of automatically detected difference, between screen-shots of AOI consecutive commits 77 and 78 (i.e. 77 and 78 commits before the current version).

Another possible use case is to apply the UI-Tracer incrementally during development that follows the continuous integration principle. This could be done by hooking it into the repository and automatically executing it with every new commit. In both cases the resulting images (e.g as in Figure 4), can be integrated into the documentation of the system, to give readers a quick and intuitive impression about what a class, such as *LayoutWindow*, is responsible for.

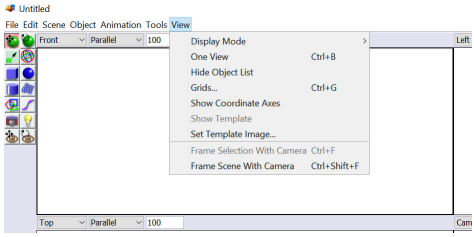


Fig. 5: Continuation of example of automatically detected difference: AOI 77 commits before current version, clicking *View* menu

3 Evaluation

We evaluate the two following questions:
(a) Does the approach cover all UI elements?
(b) How small, and with it useful, are the sets of identified files?
The following subsections describe the used case systems for the evaluations, the method used for data collection, the results of the evaluation as well as a discussion of these results.

3.1 Used Systems

To evaluate the approach it was applied to two open source system: PHNotepad[5], which is a simple text editor written in Java, and Art of Illusion[6], a program to create 3D models.

PHNotepad is a java project with 8 contributors. The fork[7] that was analyzed was made on the 11th of October 2017 and had 85 commits on GitHub. The oldest commit has been submitted in May 2012 and included an initial small version of the system with 15 files. AOI is a java project as well and has 4 contributors. UI-Tracer was used to analyze a fork of AOI[8] made on the 12th of September 2017, when the project had 526 commits on github. The project was developed already before it came to GitHub. Therefore, AOI had already reached version 2.4.1 when it was initially committed to Github in March 2007. The last stable release from December 2016 is version 3.0.3.

3.2 Method

UI-Tracer was configured for each of the two systems and run over night. For AOI, no ant and jar files needed to be planted for the most recent 537 commits as the developers use Ant themselves. However, for older commits 15 jars and an ant file needed to be provided. Here it was possible to reuse the ant file that is used for the most recent versions. Furthermore, the Ant file needed to be changed slightly for commits older than the last 417 commits, as at that commit the main class and method were renamed. The customized sikuli script was written, so that all main menu items are clicked and the sub-menu items become visible in the respective screen-shots

Also PHNotepad comes with an ant script. However, to ensure that the system compiles properly, a new ant script was planted for each commit. Altogether, three different ant scripts are planted, as in the early version of the system no image files were used and the main class was renamed at some point. Again, the customized sikuli script was written, so that all main menu items are clicked to show the sub-menu items. Furthermore, the sikuli script clicks through all the tool icons at the top row, to make eventual pup-up windows visible.

To make sure that the tool provided correct results, two manual checks were performed. The first manual check made sure that the comparison of the screen-shots worked correctly, i.e. that no difference was missed and no identical screen-shots were identified as different. The second manual check was performed to ensure that the tracing of code diffs for a pair of commits worked correctly.

To get a baseline of UI elements for which changes should be identify by the UI-Tracer, we manually compared the initial commit of each system with its most current version. In this

[5] PHNotepad `https://github.com/pH-7/Simple-Java-Text-Editor.git`
[6] AOI `https://github.com/ArtOfIllusion/ArtOfIllusion`
[7] PHNotepad fork used in evaluation `https://github.com/rhebig/Simple-Java-Text-Editor`
[8] AOI fork used in the evaluation `https://github.com/rhebig/ArtOfIllusion`

Tab. 2: General statistics about the evaluation runs.

	AOI	**PHNotepad**
Duration of analysis run	3h 15 min.	1h 25 min.
Number of commits in Repository	526	85
Number of commits with successful screen-shots	320	78
Number of commits without successful screen-shots	206	7

comparison we considered UI elements that changed or were added to the initial screens of both tools, as well as changed/added UI elements that are visible when executing the actions encoded in the customizable sikuli scripts, e.g. sub-menus and pop-up windows, when clicking and hovering over menu items and buttons. For each of those UI elements, we then searched the results of the UI-Tracer run to see whether the tool identified at least one UI change, e.g. appearance or change of position, for that UI element. For example, the UI-changes observed in Figure 4 concern 10 UI-elements: the menu item "View", which was added and the 9 sub-menu items that were moved from "Scene" to "View".

3.3 Results

The analysis of the 526 AOI commits took around 3 hours and 15 minutes, while the analysis of PHNotepad, which has only 85 commits, was done within around 1 hour and 25 minutes. Table 2 shows some general statistics about the analysis of both systems. For 320 of the AOI commits is was possible to successfully derive screen-shots of the running system (around 60% of the commits). For PHNotepad the amount was with 78 commits much higher (around 91%).

(a) Does the approach cover all UI elements? As baseline, between the initial commit and the current version, 36 UI elements of AOI changed and 32 UI elements of PHNotepad changed. These changes include among other additions of top menu items, e.g. "View" in AOI, additions and renames of sub-menu items, additions and removal of buttons and icons, rearrangements of information fields within the UI, moving of sub-menu items amongst top menu items and changing texts. For AOI the UI-Tracer identified 18 pairs of commits in between which the UI changed, including 51 UI changes that affected 39 different UI elements. For PHNotepad is were 22 pairs of commits, including 77 UI changes that affected 33 different UI elements (see Table 3).

First of all, these identified changes covered all of the 36 (AOI) + 32 (PHNotepad) UI elements for which changes were expected. Note this success-rate does not suffer from the fact that no screen-shots could be obtained for many of the commits. In theory, just comparing the initial and current commit is sufficient to get this coverage.

However, by comparing commits in between, the UI-Tracer identified 3 (AOI) + 1 (PH-Notepad) additional changes. These are changes that are not visible when only comparing the first and last version of a program. The reason for that is that changes can cover each

Tab. 3: Comparison of User Interfaces

	AOI	**PHNotepad**
Number of commit pairs associated to identified UI changes	18	22
Number of UI changes identified by UI-Tracer	51	77
UI elements with identified changes	39	33
UI elements expected to change (initial commit vs. current version)	36	32
Coverage of expected changes by identified changes	**100%**	**100%**
UI elements with changes identified in screen-shots of initial screens	16	14
Number of changes visible in screen-shots of initial screens	20	22
Number of redundant change reports (indication in multiple screen-shots)	121	181

other over time, e.g. such as the addition and later renaming of a UI element. Some changes are even done and re-done after a while, leading to information about additional UI elements.

Finally, 20 (AOI) and 22 (PHNotepad) of the changes, affecting 16 and 14 UI elements, could be identified without the customized sikuli script. Furthermore, the tool often reported multiple times on the same UI change, if the affected elements were visible on the initial screen and the customized screen-shots. We got 121 (AOI) and 181 (PHNotepad) of such redundantly reported changes.

(b) How small, and with it useful, are the sets of identified files? While it is not necessary that all commits are successfully analyzed to get a high coverage of UI elements, it influences the precision of the set of identified files. The reason is that missing information about commits in between two commits which show UI differences creates an uncertainty about which commit introduced the observed changes. Consequently, the files changed in those commits need to be integrated to the set of returned candidates.

As summarized in Table 4 AOI consists of around 713 files (including code, image, and some files) and PHNotepad consists of 32 files. For AOI, where only 60% of the commits were covered, the average number of files identified as relevant for an observed UI change is 21 (around 2.9% of all AOI files). Note that there are three UI changes that belong to the same pair of commits: 204 and 265 commits before the current version. Due to this large gap between the two successful commits, UI-Tracer had to associate 109 files with these three changes. Without these outliers, the average number of identified files for AOI is 15.5. That the average is skewed by outliers is also reflected by the low median of 8 files (1.1% of all AOI files). For PHNotepad, the number of files identified as relevant for an observed UI change is with in average 3.96 files (median 3) quite low (around 12% of all PHNotepad files). Figure 6 summarizes these results.

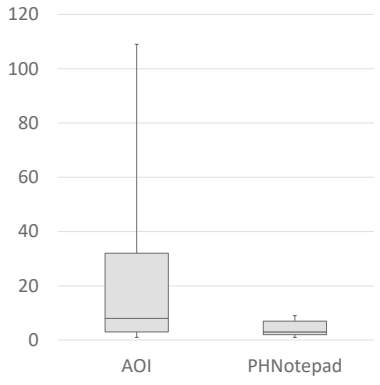


Fig. 6: Files identified by UI-Tracer per found UI changes

Tab. 4: Code Tracing per Observed UI Change

	AOI	**PHNotepad**
absolute number of files in the system	713	32
per Observed UI Change:		
median number of files identified by UI-Tracer	**8**	**3**
average number of files identified by UI-Tracer	21 (15.5 without outliers)	3.96
minimum number of files identified by UI-Tracer	1	1
maximum number of files identified by UI-Tracer	109	9
Number of UI Changes:		
... identified by UI-Tracer	51	77
... with > 2 files identified by UI-Tracer	39	48
... with > 10 files identified by UI-Tracer	23	0

The minimum number of files identified for a UI change is in both cases 1, the maximum numbers are 109 (AOI) and 9 (PHNotepad). Of the 51 identified UI changes in AOI, 12 were associated to 2 or less files and 28 to 10 or less files. Of the 77 PHNotepad UI changes, 29 were associate with 2 or less files and none was associated with more than 10 files.

3.4 Discussion

The results show that it is possible to cover all changes in UI elements that happen within the observed period of time. For those UI elements that are introduced or changed, the approach will associate relevant source code files. Many systems, however, are not open source from the very beginning, such as AOI. Thus, the approach can only cover all elements of the user interface, when applied by someone who has access to the full version history.

Furthermore, the precision of the identified files is dependent on the number of commits that can successfully be compiled and the number of files committed per commit. Both aspects are highly dependent on the project culture.

However, bringing the number of candidate files to look at down to median 3 to 8 files, especially in large systems, such as AOI, is already very useful for users who want to identify responsible parts of the source code. Despite the current limitations the results can be considered as very promising, as they open the path towards a completely new approach of software documentation and comprehension support. The UI-Tracer shows that it is possible to automatically trace UI elements to source code in a lightweight and simple way. This will allow novices to approach new systems by directly connecting the code to the user interface.

4 Conclusion and Future Work

The paper introduced a lightweight approach to support software comprehension, by helping users to trace changes in user interfaces to files in the code base. The evaluation on the two open source systems Art of Illusion and PHNotepad showed that the approach can cover

all UI elements that have been changed or added within the accessible part of the version history. With median 3 to 8 files we think that the approach can be a useful support for users who aim at changing or extending a for them unknown system. Furthermore, we think that the UI-Tracer is an important first step towards integrating the user interface perspective with approaches for comprehending source code.

In future work, we aim at further exploring the potentials of the UI-Tracer. The presented approach is in theory independent of the programming language of the system to be analyzed. In future work we plan to adapt UI-Tracer to work with additional build systems, e.g. gradle or Maven, and apply it to systems with different languages. Furthermore, our next step is to explore whether we can achieve a better precision, by automatically planting changes into the source code of a system and observing the changes to the user interface. Another direction for future work is the use of data. For example, AOI is a graphics tool. It will be interesting to see what happens if the customized Sikuli script would not just press buttons but actually use the tool to create graphics.

References

[An02] Antoniol, Giuliano; Canfora, Gerardo; Casazza, Gerardo; De Lucia, Andrea; Merlo, Ettore: Recovering traceability links between code and documentation. IEEE transactions on software engineering, 28(10):970–983, 2002.

[Bi13] Binkley, Dave; Lawrie, Dawn; Hill, Emily; Burge, Janet; Harris, Ian; Hebig, Regina; Keszocze, Oliver; Reed, Karl; Slankas, John: Task-driven software summarization. In: Software Maintenance (ICSM), 2013. IEEE, pp. 432–435, 2013.

[Ch08] Chouambe, Landry; Klatt, Benjamin; Krogmann, Klaus: Reverse engineering software-models of component-based systems. In: Software Maintenance and Reengineering, 2008. CSMR 2008. IEEE, pp. 93–102, 2008.

[DL00] Drappa, Anke; Ludewig, Jochen: Simulation in software engineering training. In: Software Engineering, 2000. IEEE, pp. 199–208, 2000.

[Fj83] Fjeldstad, Richard K: Application program maintenance study: Report to our respondents. Proceedings GUIDE 48, 1983, 1983.

[Ro12] Roehm, Tobias; Tiarks, Rebecca; Koschke, Rainer; Maalej, Walid: How do professional developers comprehend software? In: Proceedings of the 34th International Conference on Software Engineering. IEEE Press, pp. 255–265, 2012.

[TBD12] Trümper, Jonas; Beck, Martin; Döllner, Jürgen: A visual analysis approach to support perfective software maintenance. In: Information Visualisation (IV), 2012. IEEE, pp. 308–315, 2012.

[Ti11] Tiarks, Rebecca: What maintenance programmers really do: An observational study. In: Proceedings of the Workshop Software Reengineering (WSR). pp. 36–37, 2011.

[WL08] Wettel, Richard; Lanza, Michele: Codecity: 3d visualization of large-scale software. In: Companion of the 30th international conference on Software engineering. ACM, pp. 921–922, 2008.

[Xi17] Xia, Xin; Bao, Lingfeng; Lo, David; Xing, Zhenchang; Hassan, Ahmed E; Li, Shanping: Measuring Program Comprehension: A Large-Scale Field Study with Professionals. IEEE Transactions on Software Engineering, 2017.

[YCM09] Yeh, Tom; Chang, Tsung-Hsiang; Miller, Robert C: Sikuli: using GUI screenshots for search and automation. In: Proceedings of the 22nd annual ACM symposium on User interface software and technology. ACM, pp. 183–192, 2009.

[ZL96] Zeller, Andreas; Lütkehaus, Dorothea: DDD - a free graphical front-end for UNIX debuggers. ACM Sigplan Notices, 31(1):22–27, 1996.

M. Tichy, E. Bodden, M. Kuhrmann, S. Wagner, J.-P. Steghöfer (Hrsg.): SE 2018,
Lecture Notes in Informatics (LNI), Gesellschaft für Informatik, Bonn 2018 237

Smart Contract based API usage tracking on the Ethereum Blockchain

Patrick Holl[1], Elena Scepankova[2], Florian Matthes[3]

Abstract: API service providers usually charge their customers based on internally kept usage protocols. The whole process is highly intransparent for consumers because they are dependent on the providers' honesty. Using smart contracts on the Ethereum blockchain to log API usage creates an immutable and trustless single source of truth between consumers and providers. Leveraging the blockchain makes the whole process of API usage logging more transparent and comprehensible.

Keywords: Smart Contract; Ethereum; Blockchain; Logging; API; Usage

1 Motivation

Cloud computing and Software as a service (SaaS) is a fast-growing market. One segment of cloud computing includes provisioning of service providers' APIs. Those APIs are often used by companies or individuals who want to implement certain functionalities without developing the existing services on their own. Nowadays, there are APIs for almost everything, from general purpose machine learning to use case tailored APIs. In today's API service provider based economy, subscriptions and usage-based payment models are the defacto standard [Ma07]. Usually, consumers pay certain fees to get access to specific resources. Mostly, those resources are limited by the number of requests or the amount of transmitted data. To keep track of the consumed resources, providers log all requests associated with respective API keys. However, this process is highly intransparent for the consumer. There is currently no way to verify usage records in a trustless manner. Service consumers are required to trust the log files kept by the API providers without having the possibility to verify them. This is especially a problem when internally kept usage logs at the consumer side do not match with the ones provided by the service providers.

2 Idea & Prototype

To make API usage tracking more transparent, we are currently working on a system that leverages smart contracts on the Ethereum blockchain to store usage logs. The use of an

[1] Technical University of Munich (TUM), Department of Informatics (sebis chair), Boltzmannstraße 3, 85748 Garching bei München, Germany patrick.holl@tum.de
[2] TUM, Department of Informatics (sebis chair) elena.scepankova@tum.de
[3] TUM, Department of Informatics (sebis chair) matthes@tum.de

immutable public and decentralized data structure like the Ethereum blockchain establishes a trustless environment between API providers and consumers. The service providers identify each request via a message that is signed with the private key of a consumer. Afterwards, the request is logged via the smart contract and written to the blockchain. Providers and consumers are always synced and share the same state since the data is publicly visible and immutable.
Authentication on conventional API requests is usually done via a specific API key and a secret. To prevent manipulation, we require a wallet signature on each request. Only those consumers who hold the private key to a certain wallet (i.e. consumers who pay for the API service) are authenticated. The smart contract enforces that only signed messages can be added to the log entries of a consumer. This prevents the API provider (or any other party) from adding requests that never occurred to the usage logs. This whole process is illustrated in Fig 1.

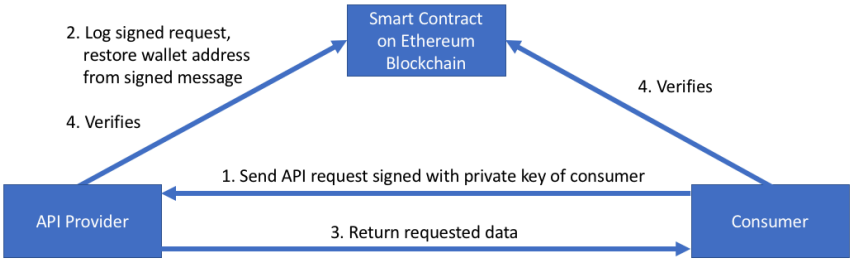


Abb. 1: Lifecycle of an API request

We are currently implementing a prototypical system which offers the stated functionality and characteristics. Primarily, small API providers profit from the proposed solution since they don't have to build their own API logging infrastructure. The smart contract represents a shared state of usage data between the API provider and the consumer.

3 Current limitations

A major reason for using Ethereum is that it's one of the most established blockchains which currently supports smart contracts [Wo14]. However, there are several challenges that are related to Ethereum. First, the transaction fees are rather high, at the time of writing the costs of a single transaction[4] are around 0.0008 ETH which is ~ 0.24$. These costs make the logging inefficient in high transaction scenarios. An alternative decentralized data structure to overcome the mentioned drawbacks could be the Tangle proposed by the IOTA foundation [Po16]. However, at the time of writing, IOTA does not support smart contracts which makes it impossible to build a trustless system for the proposed use case.

[4] In Ethereum, transaction costs and block times depend on several factors. All numbers are estimated averages on the current time of writing. Source: The Ethereum blockchain itself and https://etherscan.io (accessed on 10/10/2017)

Literaturverzeichnis

[Ma07] Ma, Dan: The business model ofßoftware-as-a-service". In: Services Computing, 2007. SCC 2007. IEEE International Conference on. IEEE, S. 701–702, 2007.

[Po16] Popov, Serguei: The tangle. IOTA, 2016.

[Wo14] Wood, Gavin: Ethereum: A secure decentralised generalised transaction ledger. Ethereum Project Yellow Paper, 151, 2014.

Autorenverzeichnis

GI-Edition Lecture Notes in Informatics

P-1 Gregor Engels, Andreas Oberweis, Albert Zündorf (Hrsg.): Modellierung 2001.

P-2 Mikhail Godlevsky, Heinrich C. Mayr (Hrsg.): Information Systems Technology and its Applications, ISTA'2001.

P-3 Ana M. Moreno, Reind P. van de Riet (Hrsg.): Applications of Natural Lan-guage to Information Systems, NLDB'2001.

P-4 H. Wörn, J. Mühling, C. Vahl, H.-P. Meinzer (Hrsg.): Rechner- und sensorgestützte Chirurgie; Workshop des SFB 414.

P-5 Andy Schürr (Hg.): OMER – Object-Oriented Modeling of Embedded Real-Time Systems.

P-6 Hans-Jürgen Appelrath, Rolf Beyer, Uwe Marquardt, Heinrich C. Mayr, Claudia Steinberger (Hrsg.): Unternehmen Hochschule, UH'2001.

P-7 Andy Evans, Robert France, Ana Moreira, Bernhard Rumpe (Hrsg.): Practical UML-Based Rigorous Development Methods – Countering or Integrating the extremists, pUML'2001.

P-8 Reinhard Keil-Slawik, Johannes Magenheim (Hrsg.): Informatikunterricht und Medienbildung, INFOS'2001.

P-9 Jan von Knop, Wilhelm Haverkamp (Hrsg.): Innovative Anwendungen in Kommunikationsnetzen, 15. DFN Arbeitstagung.

P-10 Mirjam Minor, Steffen Staab (Hrsg.): 1st German Workshop on Experience Management: Sharing Experiences about the Sharing Experience.

P-11 Michael Weber, Frank Kargl (Hrsg.): Mobile Ad-Hoc Netzwerke, WMAN 2002.

P-12 Martin Glinz, Günther Müller-Luschnat (Hrsg.): Modellierung 2002.

P-13 Jan von Knop, Peter Schirmbacher and Viljan Mahni_ (Hrsg.): The Changing Universities – The Role of Technology.

P-14 Robert Tolksdorf, Rainer Eckstein (Hrsg.): XML-Technologien für das Semantic Web – XSW 2002.

P-15 Hans-Bernd Bludau, Andreas Koop (Hrsg.): Mobile Computing in Medicine.

P-16 J. Felix Hampe, Gerhard Schwabe (Hrsg.): Mobile and Collaborative Business 2002.

P-17 Jan von Knop, Wilhelm Haverkamp (Hrsg.): Zukunft der Netze –Die Verletzbarkeit meistern, 16. DFN Arbeitstagung.

P-18 Elmar J. Sinz, Markus Plaha (Hrsg.): Modellierung betrieblicher Informationssysteme – MobIS 2002.

P-19 Sigrid Schubert, Bernd Reusch, Norbert Jesse (Hrsg.): Informatik bewegt – Informatik 2002 – 32. Jahrestagung der Gesellschaft für Informatik e.V. (GI) 30.Sept.-3. Okt. 2002 in Dortmund.

P-20 Sigrid Schubert, Bernd Reusch, Norbert Jesse (Hrsg.): Informatik bewegt – Informatik 2002 – 32. Jahrestagung der Gesellschaft für Informatik e.V. (GI) 30.Sept.-3. Okt. 2002 in Dortmund (Ergänzungsband).

P-21 Jörg Desel, Mathias Weske (Hrsg.): Promise 2002: Prozessorientierte Methoden und Werkzeuge für die Entwicklung von Informationssystemen.

P-22 Sigrid Schubert, Johannes Magenheim, Peter Hubwieser, Torsten Brinda (Hrsg.): Forschungsbeiträge zur "Didaktik der Informatik" – Theorie, Praxis, Evaluation.

P-23 Thorsten Spitta, Jens Borchers, Harry M. Sneed (Hrsg.): Software Management 2002 – Fortschritt durch Beständigkeit

P-24 Rainer Eckstein, Robert Tolksdorf (Hrsg.): XMIDX 2003 – XML-Technologien für Middleware – Middleware für XML-Anwendungen

P-25 Key Pousttchi, Klaus Turowski (Hrsg.): Mobile Commerce – Anwendungen und Perspektiven – 3. Workshop Mobile Commerce, Universität Augsburg, 04.02.2003

P-26 Gerhard Weikum, Harald Schöning, Erhard Rahm (Hrsg.): BTW 2003: Datenbanksysteme für Business, Technologie und Web

P-27 Michael Kroll, Hans-Gerd Lipinski, Kay Melzer (Hrsg.): Mobiles Computing in der Medizin

P-28 Ulrich Reimer, Andreas Abecker, Steffen Staab, Gerd Stumme (Hrsg.): WM 2003: Professionelles Wissensmanagement – Er-fahrungen und Visionen

P-29 Antje Düsterhöft, Bernhard Thalheim (Eds.): NLDB'2003: Natural Language Processing and Information Systems

P-30 Mikhail Godlevsky, Stephen Liddle, Heinrich C. Mayr (Eds.): Information Systems Technology and its Applications

P-31 Arslan Brömme, Christoph Busch (Eds.): BIOSIG 2003: Biometrics and Electronic Signatures

P-32 Peter Hubwieser (Hrsg.): Informatische Fachkonzepte im Unterricht – INFOS 2003

P-33 Andreas Geyer-Schulz, Alfred Taudes (Hrsg.): Informationswirtschaft: Ein Sektor mit Zukunft

P-34 Klaus Dittrich, Wolfgang König, Andreas Oberweis, Kai Rannenberg, Wolfgang Wahlster (Hrsg.): Informatik 2003 – Innovative Informatikanwendungen (Band 1)

P-35 Klaus Dittrich, Wolfgang König, Andreas Oberweis, Kai Rannenberg, Wolfgang Wahlster (Hrsg.): Informatik 2003 – Innovative Informatikanwendungen (Band 2)

P-36 Rüdiger Grimm, Hubert B. Keller, Kai Rannenberg (Hrsg.): Informatik 2003 – Mit Sicherheit Informatik

P-37 Arndt Bode, Jörg Desel, Sabine Rathmayer, Martin Wessner (Hrsg.): DeLFI 2003: e-Learning Fachtagung Informatik

P-38 E.J. Sinz, M. Plaha, P. Neckel (Hrsg.): Modellierung betrieblicher Informationssysteme – MobIS 2003

P-39 Jens Nedon, Sandra Frings, Oliver Göbel (Hrsg.): IT-Incident Management & IT-Forensics – IMF 2003

P-40 Michael Rebstock (Hrsg.): Modellierung betrieblicher Informationssysteme – MobIS 2004

P-41 Uwe Brinkschulte, Jürgen Becker, Dietmar Fey, Karl-Erwin Großpietsch, Christian Hochberger, Erik Maehle, Thomas Runkler (Edts.): ARCS 2004 – Organic and Pervasive Computing

P-42 Key Pousttchi, Klaus Turowski (Hrsg.): Mobile Economy – Transaktionen und Prozesse, Anwendungen und Dienste

P-43 Birgitta König-Ries, Michael Klein, Philipp Obreiter (Hrsg.): Persistance, Scalability, Transactions – Database Mechanisms for Mobile Applications

P-44 Jan von Knop, Wilhelm Haverkamp, Eike Jessen (Hrsg.): Security, E-Learning. E-Services

P-45 Bernhard Rumpe, Wofgang Hesse (Hrsg.): Modellierung 2004

P-46 Ulrich Flegel, Michael Meier (Hrsg.): Detection of Intrusions of Malware & Vulnerability Assessment

P-47 Alexander Prosser, Robert Krimmer (Hrsg.): Electronic Voting in Europe – Technology, Law, Politics and Society

P-48 Anatoly Doroshenko, Terry Halpin, Stephen W. Liddle, Heinrich C. Mayr (Hrsg.): Information Systems Technology and its Applications

P-49 G. Schiefer, P. Wagner, M. Morgenstern, U. Rickert (Hrsg.): Integration und Datensicherheit – Anforderungen, Konflikte und Perspektiven

P-50 Peter Dadam, Manfred Reichert (Hrsg.): INFORMATIK 2004 – Informatik verbindet (Band 1) Beiträge der 34. Jahrestagung der Gesellschaft für Informatik e.V. (GI), 20.-24. September 2004 in Ulm

P-51 Peter Dadam, Manfred Reichert (Hrsg.): INFORMATIK 2004 – Informatik verbindet (Band 2) Beiträge der 34. Jahrestagung der Gesellschaft für Informatik e.V. (GI), 20.-24. September 2004 in Ulm

P-52 Gregor Engels, Silke Seehusen (Hrsg.): DELFI 2004 – Tagungsband der 2. e-Learning Fachtagung Informatik

P-53 Robert Giegerich, Jens Stoye (Hrsg.): German Conference on Bioinformatics – GCB 2004

P-54 Jens Borchers, Ralf Kneuper (Hrsg.): Softwaremanagement 2004 – Outsourcing und Integration

P-55 Jan von Knop, Wilhelm Haverkamp, Eike Jessen (Hrsg.): E-Science und Grid Adhoc-Netze Medienintegration

P-56 Fernand Feltz, Andreas Oberweis, Benoit Otjacques (Hrsg.): EMISA 2004 – Informationssysteme im E-Business und E-Government

P-57 Klaus Turowski (Hrsg.): Architekturen, Komponenten, Anwendungen

P-58 Sami Beydeda, Volker Gruhn, Johannes Mayer, Ralf Reussner, Franz Schweiggert (Hrsg.): Testing of Component-Based Systems and Software Quality

P-59 J. Felix Hampe, Franz Lehner, Key Pousttchi, Kai Ranneberg, Klaus Turowski (Hrsg.): Mobile Business – Processes, Platforms, Payments

P-60 Steffen Friedrich (Hrsg.): Unterrichtskonzepte für inforrmatische Bildung

P-61 Paul Müller, Reinhard Gotzhein, Jens B. Schmitt (Hrsg.): Kommunikation in verteilten Systemen

P-62 Federrath, Hannes (Hrsg.): „Sicherheit 2005“ – Sicherheit – Schutz und Zuverlässigkeit

P-63 Roland Kaschek, Heinrich C. Mayr, Stephen Liddle (Hrsg.): Information Systems – Technology and ist Applications

P-64 Peter Liggesmeyer, Klaus Pohl, Michael Goedicke (Hrsg.): Software Engineering 2005

P-65 Gottfried Vossen, Frank Leymann, Peter Lockemann, Wolffried Stucky (Hrsg.): Datenbanksysteme in Business, Technologie und Web

P-66 Jörg M. Haake, Ulrike Lucke, Djamshid Tavangarian (Hrsg.): DeLFI 2005: 3. deutsche e-Learning Fachtagung Informatik

P-67 Armin B. Cremers, Rainer Manthey, Peter Martini, Volker Steinhage (Hrsg.): INFORMATIK 2005 – Informatik LIVE (Band 1)

P-68 Armin B. Cremers, Rainer Manthey, Peter Martini, Volker Steinhage (Hrsg.): INFORMATIK 2005 – Informatik LIVE (Band 2)

P-69 Robert Hirschfeld, Ryszard Kowalcyk, Andreas Polze, Matthias Weske (Hrsg.): NODe 2005, GSEM 2005

P-70 Klaus Turowski, Johannes-Maria Zaha (Hrsg.): Component-oriented Enterprise Application (COAE 2005)

P-71 Andrew Torda, Stefan Kurz, Matthias Rarey (Hrsg.): German Conference on Bioinformatics 2005

P-72 Klaus P. Jantke, Klaus-Peter Fähnrich, Wolfgang S. Wittig (Hrsg.): Marktplatz Internet: Von e-Learning bis e-Payment

P-73 Jan von Knop, Wilhelm Haverkamp, Eike Jessen (Hrsg.): “Heute schon das Morgen sehen“

P-74 Christopher Wolf, Stefan Lucks, Po-Wah Yau (Hrsg.): WEWoRC 2005 – Western European Workshop on Research in Cryptology

P-75 Jörg Desel, Ulrich Frank (Hrsg.): Enterprise Modelling and Information Systems Architecture

P-76 Thomas Kirste, Birgitta König-Riess, Key Pousttchi, Klaus Turowski (Hrsg.): Mobile Informationssysteme – Potentiale, Hindernisse, Einsatz

P-77 Jana Dittmann (Hrsg.): SICHERHEIT 2006

P-78 K.-O. Wenkel, P. Wagner, M. Morgenstern, K. Luzi, P. Eisermann (Hrsg.): Land- und Ernährungswirtschaft im Wandel

P-79 Bettina Biel, Matthias Book, Volker Gruhn (Hrsg.): Softwareengineering 2006

P-80 Mareike Schoop, Christian Huemer, Michael Rebstock, Martin Bichler (Hrsg.): Service-Oriented Electronic Commerce

P-81 Wolfgang Karl, Jürgen Becker, Karl-Erwin Großpietsch, Christian Hochberger, Erik Maehle (Hrsg.): ARCS´06

P-82 Heinrich C. Mayr, Ruth Breu (Hrsg.): Modellierung 2006

P-83 Daniel Huson, Oliver Kohlbacher, Andrei Lupas, Kay Nieselt and Andreas Zell (eds.): German Conference on Bioinformatics

P-84 Dimitris Karagiannis, Heinrich C. Mayr, (Hrsg.): Information Systems Technology and its Applications

P-85 Witold Abramowicz, Heinrich C. Mayr, (Hrsg.): Business Information Systems

P-86 Robert Krimmer (Ed.): Electronic Voting 2006

P-87 Max Mühlhäuser, Guido Rößling, Ralf Steinmetz (Hrsg.): DELFI 2006: 4. e-Learning Fachtagung Informatik

P-88 Robert Hirschfeld, Andreas Polze, Ryszard Kowalczyk (Hrsg.): NODe 2006, GSEM 2006

P-90 Joachim Schelp, Robert Winter, Ulrich Frank, Bodo Rieger, Klaus Turowski (Hrsg.): Integration, Informationslogistik und Architektur

P-91 Henrik Stormer, Andreas Meier, Michael Schumacher (Eds.): European Conference on eHealth 2006

P-92 Fernand Feltz, Benoît Otjacques, Andreas Oberweis, Nicolas Poussing (Eds.): AIM 2006

P-93 Christian Hochberger, Rüdiger Liskowsky (Eds.): INFORMATIK 2006 – Informatik für Menschen, Band 1

P-94 Christian Hochberger, Rüdiger Liskowsky (Eds.): INFORMATIK 2006 – Informatik für Menschen, Band 2

P-95 Matthias Weske, Markus Nüttgens (Eds.): EMISA 2005: Methoden, Konzepte und Technologien für die Entwicklung von dienstbasierten Informationssystemen

P-96 Saartje Brockmans, Jürgen Jung, York Sure (Eds.): Meta-Modelling and Ontologies

P-97 Oliver Göbel, Dirk Schadt, Sandra Frings, Hardo Hase, Detlef Günther, Jens Nedon (Eds.): IT-Incident Mangament & IT-Forensics – IMF 2006

P-98 Hans Brandt-Pook, Werner Simonsmeier und Thorsten Spitta (Hrsg.): Beratung in der Softwareentwicklung – Modelle, Methoden, Best Practices

P-99 Andreas Schwill, Carsten Schulte, Marco Thomas (Hrsg.): Didaktik der Informatik

P-100 Peter Forbrig, Günter Siegel, Markus Schneider (Hrsg.): HDI 2006: Hochschuldidaktik der Informatik

P-101 Stefan Böttinger, Ludwig Theuvsen, Susanne Rank, Marlies Morgenstern (Hrsg.): Agrarinformatik im Spannungsfeld zwischen Regionalisierung und globalen Wertschöpfungsketten

P-102 Otto Spaniol (Eds.): Mobile Services and Personalized Environments

P-103 Alfons Kemper, Harald Schöning, Thomas Rose, Matthias Jarke, Thomas Seidl, Christoph Quix, Christoph Brochhaus (Hrsg.): Datenbanksysteme in Business, Technologie und Web (BTW 2007)

P-104 Birgitta König-Ries, Franz Lehner, Rainer Malaka, Can Türker (Hrsg.) MMS 2007: Mobilität und mobile Informationssysteme

P-105 Wolf-Gideon Bleek, Jörg Raasch, Heinz Züllighoven (Hrsg.) Software Engineering 2007

P-106 Wolf-Gideon Bleek, Henning Schwentner, Heinz Züllighoven (Hrsg.) Software Engineering 2007 – Beiträge zu den Workshops

P-107 Heinrich C. Mayr, Dimitris Karagiannis (eds.) Information Systems Technology and its Applications

P-108 Arslan Brömme, Christoph Busch, Detlef Hühnlein (eds.) BIOSIG 2007: Biometrics and Electronic Signatures

P-109 Rainer Koschke, Otthein Herzog, Karl-Heinz Rödiger, Marc Ronthaler (Hrsg.) INFORMATIK 2007 Informatik trifft Logistik Band 1

P-110 Rainer Koschke, Otthein Herzog, Karl-Heinz Rödiger, Marc Ronthaler (Hrsg.) INFORMATIK 2007 Informatik trifft Logistik Band 2

P-111 Christian Eibl, Johannes Magenheim, Sigrid Schubert, Martin Wessner (Hrsg.) DeLFI 2007: 5. e-Learning Fachtagung Informatik

P-112 Sigrid Schubert (Hrsg.) Didaktik der Informatik in Theorie und Praxis

P-113 Sören Auer, Christian Bizer, Claudia Müller, Anna V. Zhdanova (Eds.) The Social Semantic Web 2007 Proceedings of the 1st Conference on Social Semantic Web (CSSW)

P-114 Sandra Frings, Oliver Göbel, Detlef Günther, Hardo G. Hase, Jens Nedon, Dirk Schadt, Arslan Brömme (Eds.) IMF2007 IT-incident management & IT-forensics Proceedings of the 3rd International Conference on IT-Incident Management & IT-Forensics

P-115 Claudia Falter, Alexander Schliep, Joachim Selbig, Martin Vingron and Dirk Walther (Eds.) German conference on bioinformatics GCB 2007

P-116 Witold Abramowicz, Leszek Maciszek (Eds.) Business Process and Services Computing 1st International Working Conference on Business Process and Services Computing BPSC 2007

P-117 Ryszard Kowalczyk (Ed.) Grid service engineering and manegement The 4th International Conference on Grid Service Engineering and Management GSEM 2007

P-118 Andreas Hein, Wilfried Thoben, Hans-Jürgen Appelrath, Peter Jensch (Eds.) European Conference on ehealth 2007

P-119 Manfred Reichert, Stefan Strecker, Klaus Turowski (Eds.) Enterprise Modelling and Information Systems Architectures Concepts and Applications

P-120 Adam Pawlak, Kurt Sandkuhl, Wojciech Cholewa, Leandro Soares Indrusiak (Eds.) Coordination of Collaborative Engineering - State of the Art and Future Challenges

P-121 Korbinian Herrmann, Bernd Bruegge (Hrsg.) Software Engineering 2008 Fachtagung des GI-Fachbereichs Softwaretechnik

P-122 Walid Maalej, Bernd Bruegge (Hrsg.) Software Engineering 2008 - Workshopband Fachtagung des GI-Fachbereichs Softwaretechnik

P-123 Michael H. Breitner, Martin Breunig, Elgar Fleisch, Ley Pousttchi, Klaus Turowski (Hrsg.)
Mobile und Ubiquitäre Informationssysteme – Technologien, Prozesse, Marktfähigkeit
Proceedings zur 3. Konferenz Mobile und Ubiquitäre Informationssysteme (MMS 2008)

P-124 Wolfgang E. Nagel, Rolf Hoffmann, Andreas Koch (Eds.)
9th Workshop on Parallel Systems and Algorithms (PASA)
Workshop of the GI/ITG Speciel Interest Groups PARS and PARVA

P-125 Rolf A.E. Müller, Hans-H. Sundermeier, Ludwig Theuvsen, Stephanie Schütze, Marlies Morgenstern (Hrsg.)
Unternehmens-IT:
Führungsinstrument oder Verwaltungsbürde
Referate der 28. GIL Jahrestagung

P-126 Rainer Gimnich, Uwe Kaiser, Jochen Quante, Andreas Winter (Hrsg.)
10th Workshop Software Reengineering (WSR 2008)

P-127 Thomas Kühne, Wolfgang Reisig, Friedrich Steimann (Hrsg.)
Modellierung 2008

P-128 Ammar Alkassar, Jörg Siekmann (Hrsg.)
Sicherheit 2008
Sicherheit, Schutz und Zuverlässigkeit
Beiträge der 4. Jahrestagung des Fachbereichs Sicherheit der Gesellschaft für Informatik e.V. (GI)
2.-4. April 2008
Saarbrücken, Germany

P-129 Wolfgang Hesse, Andreas Oberweis (Eds.)
Sigsand-Europe 2008
Proceedings of the Third AIS SIGSAND European Symposium on Analysis, Design, Use and Societal Impact of Information Systems

P-130 Paul Müller, Bernhard Neumair, Gabi Dreo Rodosek (Hrsg.)
1. DFN-Forum Kommunikations-technologien Beiträge der Fachtagung

P-131 Robert Krimmer, Rüdiger Grimm (Eds.)
3rd International Conference on Electronic Voting 2008
Co-organized by Council of Europe, Gesellschaft für Informatik and E-Voting. CC

P-132 Silke Seehusen, Ulrike Lucke, Stefan Fischer (Hrsg.)
DeLFI 2008:
Die 6. e-Learning Fachtagung Informatik

P-133 Heinz-Gerd Hegering, Axel Lehmann, Hans Jürgen Ohlbach, Christian Scheideler (Hrsg.)
INFORMATIK 2008
Beherrschbare Systeme – dank Informatik
Band 1

P-134 Heinz-Gerd Hegering, Axel Lehmann, Hans Jürgen Ohlbach, Christian Scheideler (Hrsg.)
INFORMATIK 2008
Beherrschbare Systeme – dank Informatik
Band 2

P-135 Torsten Brinda, Michael Fothe, Peter Hubwieser, Kirsten Schlüter (Hrsg.)
Didaktik der Informatik –
Aktuelle Forschungsergebnisse

P-136 Andreas Beyer, Michael Schroeder (Eds.)
German Conference on Bioinformatics GCB 2008

P-137 Arslan Brömme, Christoph Busch, Detlef Hühnlein (Eds.)
BIOSIG 2008: Biometrics and Electronic Signatures

P-138 Barbara Dinter, Robert Winter, Peter Chamoni, Norbert Gronau, Klaus Turowski (Hrsg.)
Synergien durch Integration und Informationslogistik
Proceedings zur DW2008

P-139 Georg Herzwurm, Martin Mikusz (Hrsg.)
Industrialisierung des Software-Managements
Fachtagung des GI-Fachausschusses Management der Anwendungsentwicklung und -wartung im Fachbereich Wirtschaftsinformatik

P-140 Oliver Göbel, Sandra Frings, Detlef Günther, Jens Nedon, Dirk Schadt (Eds.)
IMF 2008 - IT Incident Management & IT Forensics

P-141 Peter Loos, Markus Nüttgens, Klaus Turowski, Dirk Werth (Hrsg.)
Modellierung betrieblicher Informations-systeme (MobIS 2008)
Modellierung zwischen SOA und Compliance Management

P-142 R. Bill, P. Korduan, L. Theuvsen, M. Morgenstern (Hrsg.)
Anforderungen an die Agrarinformatik durch Globalisierung und Klimaveränderung

P-143 Peter Liggesmeyer, Gregor Engels, Jürgen Münch, Jörg Dörr, Norman Riegel (Hrsg.)
Software Engineering 2009
Fachtagung des GI-Fachbereichs Softwaretechnik

P-144 Johann-Christoph Freytag, Thomas Ruf, Wolfgang Lehner, Gottfried Vossen (Hrsg.)
Datenbanksysteme in Business, Technologie und Web (BTW)

P-145 Knut Hinkelmann, Holger Wache (Eds.)
WM2009: 5th Conference on Professional Knowledge Management

P-146 Markus Bick, Martin Breunig, Hagen Höpfner (Hrsg.)
Mobile und Ubiquitäre Informationssysteme – Entwicklung, Implementierung und Anwendung
4. Konferenz Mobile und Ubiquitäre Informationssysteme (MMS 2009)

P-147 Witold Abramowicz, Leszek Maciaszek, Ryszard Kowalczyk, Andreas Speck (Eds.)
Business Process, Services Computing and Intelligent Service Management
BPSC 2009 · ISM 2009 · YRW-MBP 2009

P-148 Christian Erfurth, Gerald Eichler, Volkmar Schau (Eds.)
9th International Conference on Innovative Internet Community Systems
I^{2}CS 2009

P-149 Paul Müller, Bernhard Neumair, Gabi Dreo Rodosek (Hrsg.)
2. DFN-Forum Kommunikationstechnologien
Beiträge der Fachtagung

P-150 Jürgen Münch, Peter Liggesmeyer (Hrsg.)
Software Engineering
2009 - Workshopband

P-151 Armin Heinzl, Peter Dadam, Stefan Kirn, Peter Lockemann (Eds.)
PRIMIUM
Process Innovation for Enterprise Software

P-152 Jan Mendling, Stefanie Rinderle-Ma, Werner Esswein (Eds.)
Enterprise Modelling and Information Systems Architectures
Proceedings of the 3rd Int'l Workshop EMISA 2009

P-153 Andreas Schwill, Nicolas Apostolopoulos (Hrsg.)
Lernen im Digitalen Zeitalter
DeLFI 2009 – Die 7. E-Learning Fachtagung Informatik

P-154 Stefan Fischer, Erik Maehle Rüdiger Reischuk (Hrsg.)
INFORMATIK 2009
Im Focus das Leben

P-155 Arslan Brömme, Christoph Busch, Detlef Hühnlein (Eds.)
BIOSIG 2009:
Biometrics and Electronic Signatures
Proceedings of the Special Interest Group on Biometrics and Electronic Signatures

P-156 Bernhard Koerber (Hrsg.)
Zukunft braucht Herkunft
25 Jahre »INFOS – Informatik und Schule«

P-157 Ivo Grosse, Steffen Neumann, Stefan Posch, Falk Schreiber, Peter Stadler (Eds.)
German Conference on Bioinformatics 2009

P-158 W. Claupein, L. Theuvsen, A. Kämpf, M. Morgenstern (Hrsg.)
Precision Agriculture Reloaded – Informationsgestützte Landwirtschaft

P-159 Gregor Engels, Markus Luckey, Wilhelm Schäfer (Hrsg.)
Software Engineering 2010

P-160 Gregor Engels, Markus Luckey, Alexander Pretschner, Ralf Reussner (Hrsg.)
Software Engineering 2010 – Workshopband
(inkl. Doktorandensymposium)

P-161 Gregor Engels, Dimitris Karagiannis Heinrich C. Mayr (Hrsg.)
Modellierung 2010

P-162 Maria A. Wimmer, Uwe Brinkhoff, Siegfried Kaiser, Dagmar Lück-Schneider, Erich Schweighofer, Andreas Wiebe (Hrsg.)
Vernetzte IT für einen effektiven Staat
Gemeinsame Fachtagung Verwaltungsinformatik (FTVI) und Fachtagung Rechtsinformatik (FTRI) 2010

P-163 Markus Bick, Stefan Eulgem, Elgar Fleisch, J. Felix Hampe, Birgitta König-Ries, Franz Lehner, Key Pousttchi, Kai Rannenberg (Hrsg.)
Mobile und Ubiquitäre Informationssysteme
Technologien, Anwendungen und Dienste zur Unterstützung von mobiler Kollaboration

P-164 Arslan Brömme, Christoph Busch (Eds.)
BIOSIG 2010: Biometrics and Electronic Signatures Proceedings of the Special Interest Group on Biometrics and Electronic Signatures

P-165 Gerald Eichler, Peter Kropf, Ulrike Lechner, Phayung Meesad, Herwig Unger (Eds.)
10th International Conference on Innovative Internet Community Systems (I^2CS) – Jubilee Edition 2010 –

P-166 Paul Müller, Bernhard Neumair, Gabi Dreo Rodosek (Hrsg.)
3. DFN-Forum Kommunikationstechnologien
Beiträge der Fachtagung

P-167 Robert Krimmer, Rüdiger Grimm (Eds.)
4th International Conference on Electronic Voting 2010
co-organized by the Council of Europe, Gesellschaft für Informatik and E-Voting.CC

P-168 Ira Diethelm, Christina Dörge, Claudia Hildebrandt, Carsten Schulte (Hrsg.)
Didaktik der Informatik
Möglichkeiten empirischer Forschungsmethoden und Perspektiven der Fachdidaktik

P-169 Michael Kerres, Nadine Ojstersek Ulrik Schroeder, Ulrich Hoppe (Hrsg.)
DeLFI 2010 - 8. Tagung der Fachgruppe E-Learning der Gesellschaft für Informatik e.V.

P-170 Felix C. Freiling (Hrsg.)
Sicherheit 2010
Sicherheit, Schutz und Zuverlässigkeit

P-171 Werner Esswein, Klaus Turowski, Martin Juhrisch (Hrsg.)
Modellierung betrieblicher Informationssysteme (MobIS 2010)
Modellgestütztes Management

P-172 Stefan Klink, Agnes Koschmider Marco Mevius, Andreas Oberweis (Hrsg.)
EMISA 2010
Einflussfaktoren auf die Entwicklung flexibler, integrierter Informationssysteme
Beiträge des Workshops der GI-Fachgruppe EMISA (Entwicklungsmethoden für Informationssysteme und deren Anwendung)

P-173 Dietmar Schomburg, Andreas Grote (Eds.)
German Conference on Bioinformatics 2010

P-174 Arslan Brömme, Torsten Eymann, Detlef Hühnlein, Heiko Roßnagel, Paul Schmücker (Hrsg.)
perspeGKtive 2010
Workshop „Innovative und sichere Informationstechnologie für das Gesundheitswesen von morgen“

P-175 Klaus-Peter Fähnrich, Bogdan Franczyk (Hrsg.)
INFORMATIK 2010
Service Science – Neue Perspektiven für die Informatik
Band 1

P-176 Klaus-Peter Fähnrich, Bogdan Franczyk (Hrsg.)
INFORMATIK 2010
Service Science – Neue Perspektiven für die Informatik
Band 2

P-177 Witold Abramowicz, Rainer Alt, Klaus-Peter Fähnrich, Bogdan Franczyk, Leszek A. Maciaszek (Eds.)
INFORMATIK 2010
Business Process and Service Science – Proceedings of ISSS and BPSC

P-178 Wolfram Pietsch, Benedikt Krams (Hrsg.)
Vom Projekt zum Produkt
Fachtagung des GI-Fachausschusses Management der Anwendungsentwicklung und -wartung im Fachbereich Wirtschafts-informatik (WI-MAW), Aachen, 2010

P-179 Stefan Gruner, Bernhard Rumpe (Eds.)
FM+AM`2010
Second International Workshop on Formal Methods and Agile Methods

P-180 Theo Härder, Wolfgang Lehner, Bernhard Mitschang, Harald Schöning, Holger Schwarz (Hrsg.)
Datenbanksysteme für Business, Technologie und Web (BTW)
14. Fachtagung des GI-Fachbereichs „Datenbanken und Informationssysteme“ (DBIS)

P-181 Michael Clasen, Otto Schätzel, Brigitte Theuvsen (Hrsg.)
Qualität und Effizienz durch informationsgestützte Landwirtschaft, Fokus: Moderne Weinwirtschaft

P-182 Ronald Maier (Hrsg.)
6th Conference on Professional Knowledge Management
From Knowledge to Action

P-183 Ralf Reussner, Matthias Grund, Andreas Oberweis, Walter Tichy (Hrsg.)
Software Engineering 2011
Fachtagung des GI-Fachbereichs Softwaretechnik

P-184 Ralf Reussner, Alexander Pretschner, Stefan Jähnichen (Hrsg.)
Software Engineering 2011
Workshopband
(inkl. Doktorandensymposium)

P-185 Hagen Höpfner, Günther Specht, Thomas Ritz, Christian Bunse (Hrsg.)
MMS 2011: Mobile und ubiquitäre Informationssysteme Proceedings zur 6. Konferenz Mobile und Ubiquitäre Informationssysteme (MMS 2011)

P-186 Gerald Eichler, Axel Küpper, Volkmar Schau, Hacène Fouchal, Herwig Unger (Eds.)
11th International Conference on Innovative Internet Community Systems (I^2CS)

P-187 Paul Müller, Bernhard Neumair, Gabi Dreo Rodosek (Hrsg.)
4. DFN-Forum Kommunikations-technologien, Beiträge der Fachtagung 20. Juni bis 21. Juni 2011 Bonn

P-188 Holger Rohland, Andrea Kienle, Steffen Friedrich (Hrsg.)
DeLFI 2011 – Die 9. e-Learning Fachtagung Informatik der Gesellschaft für Informatik e.V. 5.–8. September 2011, Dresden

P-189 Thomas, Marco (Hrsg.)
Informatik in Bildung und Beruf
INFOS 2011
14. GI-Fachtagung Informatik und Schule

P-190 Markus Nüttgens, Oliver Thomas, Barbara Weber (Eds.)
Enterprise Modelling and Information Systems Architectures (EMISA 2011)

P-191 Arslan Brömme, Christoph Busch (Eds.)
BIOSIG 2011
International Conference of the Biometrics Special Interest Group

P-192 Hans-Ulrich Heiß, Peter Pepper, Holger Schlingloff, Jörg Schneider (Hrsg.)
INFORMATIK 2011
Informatik schafft Communities

P-193 Wolfgang Lehner, Gunther Piller (Hrsg.)
IMDM 2011

P-194 M. Clasen, G. Fröhlich, H. Bernhardt, K. Hildebrand, B. Theuvsen (Hrsg.)
Informationstechnologie für eine nachhaltige Landbewirtschaftung
Fokus Forstwirtschaft

P-195 Neeraj Suri, Michael Waidner (Hrsg.)
Sicherheit 2012
Sicherheit, Schutz und Zuverlässigkeit
Beiträge der 6. Jahrestagung des Fachbereichs Sicherheit der Gesellschaft für Informatik e.V. (GI)

P-196 Arslan Brömme, Christoph Busch (Eds.)
BIOSIG 2012
Proceedings of the 11th International Conference of the Biometrics Special Interest Group

P-197 Jörn von Lucke, Christian P. Geiger, Siegfried Kaiser, Erich Schweighofer, Maria A. Wimmer (Hrsg.)
Auf dem Weg zu einer offenen, smarten und vernetzten Verwaltungskultur
Gemeinsame Fachtagung Verwaltungsinformatik (FTVI) und Fachtagung Rechtsinformatik (FTRI) 2012

P-198 Stefan Jähnichen, Axel Küpper, Sahin Albayrak (Hrsg.)
Software Engineering 2012
Fachtagung des GI-Fachbereichs Softwaretechnik

P-199 Stefan Jähnichen, Bernhard Rumpe, Holger Schlingloff (Hrsg.)
Software Engineering 2012
Workshopband

P-200 Gero Mühl, Jan Richling, Andreas Herkersdorf (Hrsg.)
ARCS 2012 Workshops

P-201 Elmar J. Sinz Andy Schürr (Hrsg.)
Modellierung 2012

P-202 Andrea Back, Markus Bick, Martin Breunig, Key Pousttchi, Frédéric Thiesse (Hrsg.)
MMS 2012:Mobile und Ubiquitäre Informationssysteme

P-203 Paul Müller, Bernhard Neumair, Helmut Reiser, Gabi Dreo Rodosek (Hrsg.)
5. DFN-Forum Kommunikations-technologien
Beiträge der Fachtagung

P-204 Gerald Eichler, Leendert W. M. Wienhofen, Anders Kofod-Petersen, Herwig Unger (Eds.)
12th International Conference on Innovative Internet Community Systems (I2CS 2012)

P-205 Manuel J. Kripp, Melanie Volkamer, Rüdiger Grimm (Eds.)
5th International Conference on Electronic Voting 2012 (EVOTE2012)
Co-organized by the Council of Europe, Gesellschaft für Informatik and E-Voting.CC

P-206 Stefanie Rinderle-Ma, Mathias Weske (Hrsg.)
EMISA 2012
Der Mensch im Zentrum der Modellierung

P-207 Jörg Desel, Jörg M. Haake, Christian Spannagel (Hrsg.)
DeLFI 2012: Die 10. e-Learning Fachtagung Informatik der Gesellschaft für Informatik e.V.
24.–26. September 2012

P-208 Ursula Goltz, Marcus Magnor, Hans-Jürgen Appelrath, Herbert Matthies, Wolf-Tilo Balke, Lars Wolf (Hrsg.) INFORMATIK 2012

P-209 Hans Brandt-Pook, André Fleer, Thorsten Spitta, Malte Wattenberg (Hrsg.) Nachhaltiges Software Management

P-210 Erhard Plödereder, Peter Dencker, Herbert Klenk, Hubert B. Keller, Silke Spitzer (Hrsg.) Automotive – Safety & Security 2012 Sicherheit und Zuverlässigkeit für automobile Informationstechnik

P-211 M. Clasen, K. C. Kersebaum, A. Meyer-Aurich, B. Theuvsen (Hrsg.) Massendatenmanagement in der Agrar- und Ernährungswirtschaft Erhebung - Verarbeitung - Nutzung Referate der 33. GIL-Jahrestagung 20. – 21. Februar 2013, Potsdam

P-212 Arslan Brömme, Christoph Busch (Eds.) BIOSIG 2013 Proceedings of the 12th International Conference of the Biometrics Special Interest Group 04.–06. September 2013 Darmstadt, Germany

P-213 Stefan Kowalewski, Bernhard Rumpe (Hrsg.) Software Engineering 2013 Fachtagung des GI-Fachbereichs Softwaretechnik

P-214 Volker Markl, Gunter Saake, Kai-Uwe Sattler, Gregor Hackenbroich, Bernhard Mit schang, Theo Härder, Veit Köppen (Hrsg.) Datenbanksysteme für Business, Technologie und Web (BTW) 2013 13. – 15. März 2013, Magdeburg

P-215 Stefan Wagner, Horst Lichter (Hrsg.) Software Engineering 2013 Workshopband (inkl. Doktorandensymposium) 26. Februar – 1. März 2013, Aachen

P-216 Gunter Saake, Andreas Henrich, Wolfgang Lehner, Thomas Neumann, Veit Köppen (Hrsg.) Datenbanksysteme für Business, Technologie und Web (BTW) 2013 – Workshopband 11. – 12. März 2013, Magdeburg

P-217 Paul Müller, Bernhard Neumair, Helmut Reiser, Gabi Dreo Rodosek (Hrsg.) 6. DFN-Forum Kommunikations-technologien Beiträge der Fachtagung 03.–04. Juni 2013, Erlangen

P-218 Andreas Breiter, Christoph Rensing (Hrsg.) DeLFI 2013: Die 11 e-Learning Fachtagung Informatik der Gesellschaft für Informatik e.V. (GI) 8. – 11. September 2013, Bremen

P-219 Norbert Breier, Peer Stechert, Thomas Wilke (Hrsg.) Informatik erweitert Horizonte INFOS 2013 15. GI-Fachtagung Informatik und Schule 26. – 28. September 2013

P-220 Matthias Horbach (Hrsg.) INFORMATIK 2013 Informatik angepasst an Mensch, Organisation und Umwelt 16. – 20. September 2013, Koblenz

P-221 Maria A. Wimmer, Marijn Janssen, Ann Macintosh, Hans Jochen Scholl, Efthimios Tambouris (Eds.) Electronic Government and Electronic Participation Joint Proceedings of Ongoing Research of IFIP EGOV and IFIP ePart 2013 16. – 19. September 2013, Koblenz

P-222 Reinhard Jung, Manfred Reichert (Eds.) Enterprise Modelling and Information Systems Architectures (EMISA 2013) St. Gallen, Switzerland September 5. – 6. 2013

P-223 Detlef Hühnlein, Heiko Roßnagel (Hrsg.) Open Identity Summit 2013 10. – 11. September 2013 Kloster Banz, Germany

P-224 Eckhart Hanser, Martin Mikusz, Masud Fazal-Baqaie (Hrsg.) Vorgehensmodelle 2013 Vorgehensmodelle – Anspruch und Wirklichkeit 20. Tagung der Fachgruppe Vorgehensmodelle im Fachgebiet Wirtschaftsinformatik (WI-VM) der Gesellschaft für Informatik e.V. Lörrach, 2013

P-225 Hans-Georg Fill, Dimitris Karagiannis, Ulrich Reimer (Hrsg.) Modellierung 2014 19. – 21. März 2014, Wien

P-226 M. Clasen, M. Hamer, S. Lehnert, B. Petersen, B. Theuvsen (Hrsg.) IT-Standards in der Agrar- und Ernährungswirtschaft Fokus: Risiko- und Krisenmanagement Referate der 34. GIL-Jahrestagung 24. – 25. Februar 2014, Bonn

P-227 Wilhelm Hasselbring,
Nils Christian Ehmke (Hrsg.)
Software Engineering 2014
Fachtagung des GI-Fachbereichs Softwaretechnik
25. – 28. Februar 2014
Kiel, Deutschland

P-228 Stefan Katzenbeisser, Volkmar Lotz, Edgar Weippl (Hrsg.)
Sicherheit 2014
Sicherheit, Schutz und Zuverlässigkeit
Beiträge der 7. Jahrestagung des Fachbereichs Sicherheit der Gesellschaft für Informatik e.V. (GI)
19. – 21. März 2014, Wien

P-229 Dagmar Lück-Schneider, Thomas Gordon, Siegfried Kaiser, Jörn von Lucke,Erich Schweighofer, Maria A.Wimmer, Martin G. Löhe (Hrsg.)
Gemeinsam Electronic Government ziel(gruppen)gerecht gestalten und organisieren
Gemeinsame Fachtagung Verwaltungsinformatik (FTVI) und Fachtagung Rechtsinformatik (FTRI)
2014, 20.-21. März 2014 in Berlin

P-230 Arslan Brömme, Christoph Busch (Eds.)
BIOSIG 2014
Proceedings of the 13th International Conference of the Biometrics Special Interest Group
10. – 12. September 2014 in Darmstadt, Germany

P-231 Paul Müller, Bernhard Neumair, Helmut Reiser, Gabi Dreo Rodosek (Hrsg.)
7. DFN-Forum Kommunikationstechnologien
16. – 17. Juni 2014
Fulda

P-232 E. Plödereder, L. Grunske, E. Schneider, D. Ull (Hrsg.)
INFORMATIK 2014
Big Data – Komplexität meistern
22. – 26. September 2014
Stuttgart

P-233 Stephan Trahasch, Rolf Plötzner, Gerhard Schneider, Claudia Gayer, Daniel Sassiat, Nicole Wöhrle (Hrsg.)
DeLFI 2014 – Die 12. e-Learning Fachtagung Informatik
der Gesellschaft für Informatik e.V.
15. – 17. September 2014
Freiburg

P-234 Fernand Feltz, Bela Mutschler, Benoît Otjacques (Eds.)
Enterprise Modelling and Information Systems Architectures
(EMISA 2014)
Luxembourg, September 25-26, 2014

P-235 Robert Giegerich, Ralf Hofestädt, Tim W. Nattkemper (Eds.)
German Conference on Bioinformatics 2014
September 28 – October 1
Bielefeld, Germany

P-236 Martin Engstler, Eckhart Hanser, Martin Mikusz, Georg Herzwurm (Hrsg.)
Projektmanagement und Vorgehensmodelle 2014
Soziale Aspekte und Standardisierung
Gemeinsame Tagung der Fachgruppen Projektmanagement (WI-PM) und Vorgehensmodelle (WI-VM) im Fachgebiet Wirtschaftsinformatik der Gesellschaft für Informatik e.V., Stuttgart 2014

P-237 Detlef Hühnlein, Heiko Roßnagel (Hrsg.)
Open Identity Summit 2014
4.–6. November 2014
Stuttgart, Germany

P-238 Arno Ruckelshausen, Hans-Peter Schwarz, Brigitte Theuvsen (Hrsg.)
Informatik in der Land-, Forst- und Ernährungswirtschaft
Referate der 35. GIL-Jahrestagung
23. – 24. Februar 2015, Geisenheim

P-239 Uwe Aßmann, Birgit Demuth, Thorsten Spitta, Georg Püschel, Ronny Kaiser (Hrsg.)
Software Engineering & Management 2015
17.-20. März 2015, Dresden

P-240 Herbert Klenk, Hubert B. Keller, Erhard Plödereder, Peter Dencker (Hrsg.)
Automotive – Safety & Security 2015
Sicherheit und Zuverlässigkeit für automobile Informationstechnik
21.–22. April 2015, Stuttgart

P-241 Thomas Seidl, Norbert Ritter, Harald Schöning, Kai-Uwe Sattler, Theo Härder, Steffen Friedrich, Wolfram Wingerath (Hrsg.)
Datenbanksysteme für Business, Technologie und Web (BTW 2015)
04. – 06. März 2015, Hamburg

P-242 Norbert Ritter, Andreas Henrich, Wolfgang Lehner, Andreas Thor, Steffen Friedrich, Wolfram Wingerath (Hrsg.)
Datenbanksysteme für Business, Technologie und Web (BTW 2015) – Workshopband
02. – 03. März 2015, Hamburg

P-243 Paul Müller, Bernhard Neumair, Helmut Reiser, Gabi Dreo Rodosek (Hrsg.)
8. DFN-Forum Kommunikationstechnologien
06.–09. Juni 2015, Lübeck

P-244 Alfred Zimmermann, Alexander Rossmann (Eds.)
Digital Enterprise Computing (DEC 2015)
Böblingen, Germany June 25-26, 2015

P-245 Arslan Brömme, Christoph Busch , Christian Rathgeb, Andreas Uhl (Eds.)
BIOSIG 2015
Proceedings of the 14th International Conference of the Biometrics Special Interest Group
09.–11. September 2015
Darmstadt, Germany

P-246 Douglas W. Cunningham, Petra Hofstedt, Klaus Meer, Ingo Schmitt (Hrsg.)
INFORMATIK 2015
28.9.-2.10. 2015, Cottbus

P-247 Hans Pongratz, Reinhard Keil (Hrsg.)
DeLFI 2015 – Die 13. E-Learning Fachtagung Informatik der Gesellschaft für Informatik e.V. (GI)
1.–4. September 2015
München

P-248 Jens Kolb, Henrik Leopold, Jan Mendling (Eds.)
Enterprise Modelling and Information Systems Architectures
Proceedings of the 6th Int. Workshop on Enterprise Modelling and Information Systems Architectures, Innsbruck, Austria
September 3-4, 2015

P-249 Jens Gallenbacher (Hrsg.)
Informatik allgemeinbildend begreifen
INFOS 2015 16. GI-Fachtagung Informatik und Schule
20.–23. September 2015

P-250 Martin Engstler, Masud Fazal-Baqaie, Eckhart Hanser, Martin Mikusz, Alexander Volland (Hrsg.)
Projektmanagement und Vorgehensmodelle 2015
Hybride Projektstrukturen erfolgreich umsetzen
Gemeinsame Tagung der Fachgruppen Projektmanagement (WI-PM) und Vorgehensmodelle (WI-VM) im Fachgebiet Wirtschaftsinformatik der Gesellschaft für Informatik e.V., Elmshorn 2015

P-251 Detlef Hühnlein, Heiko Roßnagel, Raik Kuhlisch, Jan Ziesing (Eds.)
Open Identity Summit 2015
10.–11. November 2015
Berlin, Germany

P-252 Jens Knoop, Uwe Zdun (Hrsg.)
Software Engineering 2016
Fachtagung des GI-Fachbereichs Softwaretechnik
23.–26. Februar 2016, Wien

P-253 A. Ruckelshausen, A. Meyer-Aurich, T. Rath, G. Recke, B. Theuvsen (Hrsg.)
Informatik in der Land-, Forst- und Ernährungswirtschaft
Fokus: Intelligente Systeme – Stand der Technik und neue Möglichkeiten
Referate der 36. GIL-Jahrestagung
22.-23. Februar 2016, Osnabrück

P-254 Andreas Oberweis, Ralf Reussner (Hrsg.)
Modellierung 2016
2.–4. März 2016, Karlsruhe

P-255 Stefanie Betz, Ulrich Reimer (Hrsg.)
Modellierung 2016 Workshopband
2.–4. März 2016, Karlsruhe

P-256 Michael Meier, Delphine Reinhardt, Steffen Wendzel (Hrsg.)
Sicherheit 2016
Sicherheit, Schutz und Zuverlässigkeit
Beiträge der 8. Jahrestagung des Fachbereichs Sicherheit der Gesellschaft für Informatik e.V. (GI)
5.–7. April 2016, Bonn

P-257 Paul Müller, Bernhard Neumair, Helmut Reiser, Gabi Dreo Rodosek (Hrsg.)
9. DFN-Forum Kommunikationstechnologien
31. Mai – 01. Juni 2016, Rostock

P-258 Dieter Hertweck, Christian Decker (Eds.)
Digital Enterprise Computing (DEC 2016)
14.–15. Juni 2016, Böblingen

P-259 Heinrich C. Mayr, Martin Pinzger (Hrsg.)
INFORMATIK 2016
26.–30. September 2016, Klagenfurt

P-260 Arslan Brömme, Christoph Busch, Christian Rathgeb, Andreas Uhl (Eds.)
BIOSIG 2016
Proceedings of the 15th International Conference of the Biometrics Special Interest Group
21.–23. September 2016, Darmstadt

P-261 Detlef Rätz, Michael Breidung, Dagmar Lück-Schneider, Siegfried Kaiser, Erich Schweighofer (Hrsg.)
Digitale Transformation: Methoden, Kompetenzen und Technologien für die Verwaltung
Gemeinsame Fachtagung Verwaltungsinformatik (FTVI) und Fachtagung Rechtsinformatik (FTRI) 2016
22.–23. September 2016, Dresden

P-262 Ulrike Lucke, Andreas Schwill, Raphael Zender (Hrsg.)
DeLFI 2016 – Die 14. E-Learning Fachtagung Informatik
der Gesellschaft für Informatik e.V. (GI)
11.–14. September 2016, Potsdam

P-263 Martin Engstler, Masud Fazal-Baqaie, Eckhart Hanser, Oliver Linssen, Martin Mikusz, Alexander Volland (Hrsg.)
Projektmanagement und Vorgehensmodelle 2016
Arbeiten in hybriden Projekten: Das Sowohl-als-auch von Stabilität und Dynamik
Gemeinsame Tagung der Fachgruppen Projektmanagement (WI-PM) und Vorgehensmodelle (WI-VM) im Fachgebiet Wirtschaftsinformatik
der Gesellschaft für Informatik e.V., Paderborn 2016

P-264 Detlef Hühnlein, Heiko Roßnagel, Christian H. Schunck, Maurizio Talamo (Eds.)
Open Identity Summit 2016
der Gesellschaft für Informatik e.V. (GI)
13.–14. October 2016, Rome, Italy

P-265 Bernhard Mitschang, Daniela Nicklas,Frank Leymann, Harald Schöning, Melanie Herschel, Jens Teubner, Theo Härder, Oliver Kopp, Matthias Wieland (Hrsg.)
Datenbanksysteme für Business, Technologie und Web (BTW 2017)
6.–10. März 2017, Stuttgart

P-266 Bernhard Mitschang, Norbert Ritter, Holger Schwarz, Meike Klettke, Andreas Thor, Oliver Kopp, Matthias Wieland (Hrsg.)
Datenbanksysteme für Business, Technologie und Web (BTW 2017)
Workshopband
6.–7. März 2017, Stuttgart

P-267 Jan Jürjens, Kurt Schneider (Hrsg.)
Software Engineering 2017
21.–24. Februar 2017, Hannover

P-268 A. Ruckelshausen, A. Meyer-Aurich, W. Lentz, B. Theuvsen (Hrsg.)
Informatik in der Land-, Forst- und Ernährungswirtschaft
Fokus: Digitale Transformation – Wege in eine zukunftsfähige Landwirtschaft
Referate der 37. GIL-Jahrestagung
06.–07. März 2017, Dresden

P-269 Peter Dencker, Herbert Klenk, Hubert Keller, Erhard Plödereder (Hrsg.)
Automotive – Safety & Security 2017
30.–31. Mai 2017, Stuttgart

P-270 Arslan Brömme, Christoph Busch, Antitza Dantcheva, Christian Rathgeb, Andreas Uhl (Eds.)
BIOSIG 2017
20.–22. September 2017, Darmstadt

P-271 Paul Müller, Bernhard Neumair, Helmut Reiser, Gabi Dreo Rodosek (Hrsg.)
10. DFN-Forum Kommunikations-technologien
30. – 31. Mai 2017, Berlin

P-272 Alexander Rossmann, Alfred Zimmermann (eds.)
Digital Enterprise Computing (DEC 2017)
11.–12. Juli 2017, Böblingen

P-273 Christoph Igel, Carsten Ullrich, Martin Wessner (Hrsg.)
BILDUNGSRÄUME
DeLFI 2017
Die 15. e-Learning Fachtagung Informatik der Gesellschaft für Informatik e.V. (GI)
5. bis 8. September 2017, Chemnitz

P-274 Ira Diethelm (Hrsg.)
Informatische Bildung zum Verstehen und Gestalten der digitalen Welt
13.–15. September 2017, Oldenburg

P-275 Maximilian Eibl, Martin Gaedke (Hrsg.)
INFORMATIK 2017
25.–29. September 2017, Chemnitz

P276 Alexander Volland, Martin Engstler, Masud Fazal-Baqaie, Eckhart Hanser, Oliver Linssen, Martin Mikusz (Hrsg.)
Projektmanagement und Vorgehensmodelle 2017
Die Spannung zwischen dem Prozess und den Menschen im Projekt
Gemeinsame Tagung der Fachgruppen Projektmanagement und Vorgehensmodelle im Fachgebiet Wirtschaftsinformatik der Gesellschaft für Informatik e.V. in Kooperation mit der Fachgruppe IT-Projektmanagement der GPM e.V.,
Darmstadt 2017

P-277 Lothar Fritsch, Heiko Roßnagel, Detlef Hühnlein (Hrsg.)
Open Identity Summit 2017
5.–6. October 2017, Karlstad, Sweden

P-278 Arno Ruckelshausen, Andreas Meyer-Aurich, Karsten Borchard, Constanze Hofacker, Jens-Peter Loy, Rolf Schwerdtfeger, Hans-Hennig Sundermeier, Helga Floto, Brigitte Theuvsen (Hrsg.)
Informatik in der Land-, Forst- und Ernährungswirtschaft
Referate der 38. GIL-Jahrestagung
26.–27. Februar 2018, Kiel

P-279 Matthias Tichy, Eric Bodden, Marco Kuhrmann, Stefan Wagner, Jan-Philipp Steghöfer (Hrsg.)
Software Engineering und Software Management 2018
5.–9. März 2018, Ulm

P-280 Ina Schaefer, Dimitris Karagiannis, Andreas Vogelsang, Daniel Méndez, Christoph Seidl (Hrsg.)
Modellierung 2018
21.–23. Februar 2018, Braunschweig

The titles can be purchased at:

Köllen Druck + Verlag GmbH
Ernst-Robert-Curtius-Str. 14 · D-53117 Bonn
Fax: +49 (0)228/9898222
E-Mail: druckverlag@koellen.de

Gesellschaft für Informatik e.V. (GI)

publishes this series in order to make available to a broad public recent findings in informatics (i.e. computer science and information systems), to document conferences that are organized in cooperation with GI and to publish the annual GI Award dissertation.

Broken down into
• seminars
• proceedings
• dissertations
• thematics
current topics are dealt with from the vantage point of research and development, teaching and further training in theory and practice. The Editorial Committee uses an intensive review process in order to ensure high quality contributions.

The volumes are published in German or English.

Information: http://www.gi.de/service/publikationen/lni/

ISSN 1617-5468
ISBN 978-3-88579-673-2

This volume contains the contributions of the Software Engineering and Software Management (SE/SWM) 2018 conference held from 05.03 – 09.03.2018 in Ulm, Germany.

The SE/SWM proceedings contain extended abstracts from the scientific program, the industrial program, keynotes, and the workshop program