GI-Edition

Lecture Notes in Informatics

Steffen Becker, Ivan Bogicevic,
Georg Herzwurm, Stefan Wagner (Hrsg.)

Software Engineering and Software Management 2019

18.–22. Februar 2019
Stuttgart

Proceedings

292

S. Becker, I. Bogicevic, G. Herzwurm, S. Wagner (Hrsg.):
Software Engineering and Software Management 2019

GESELLSCHAFT
FÜR INFORMATIK

Steffen Becker, Ivan Bogicevic,
Georg Herzwurm, Stefan Wagner (Hrsg.)

Software Engineering und Software Management 2019

18.-22. Februar 2019
Stuttgart, Deutschland

Gesellschaft für Informatik e.V. (GI)

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

Volume P-292

ISBN 978-3-88579-686-2
ISSN 1617-5468

Volume Editors
Prof. Dr.-Ing. Steffen Becker
Universität Stuttgart
Universitätsstraße 38, 70569 Stuttgart, Deutschland
steffen.becker@iste.uni-stuttgart.de

Dr. rer. nat. Ivan Bogicevic
Universität Stuttgart
Universitätsstraße 38, 70569 Stuttgart, Deutschland
ivan.bogicevic@iste.uni-stuttgart.de

Prof. Dr. rer. pol. habil. Georg Herzwurm
Universität Stuttgart
Keplerstraße 17, 70174 Stuttgart, Deutschland
georg.herzwurm@bwi.uni-stuttgart.de

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

Dissertations
Steffen Hölldobler, Technische Universität Dresden, Germany
Thematics
Andreas Oberweis, Karlsruher Institut für Technologie (KIT), Germany

printed by Köllen Druck+Verlag GmbH, Bonn

Vorwort

Herzlich willkommen zu den Tagungen Software Engineering (SE) des Fachbereichs Softwaretechnik der Gesellschaft für Informatik (GI) sowie Software Management (SWM) des GI-Fachausschusses WI-MAW in der Landeshauptstadt Stuttgart.

Die jährliche Tagung des Fachbereichs Softwaretechnik der GI hat sich als Plattform für den Austausch und die Zusammenarbeit in allen Bereichen der Softwaretechnik etabliert. Der Austausch erstreckt sich dabei sowohl auf neuste akademische Erkenntnisse als auch auf aktuelle industrielle Trends und Praktiken. Dies spiegelt sich auch in der Tatsache wider, dass sich die Tagung gleichermaßen an Softwareentwicklerinnen und -entwickler aus der Praxis als auch an Forscherinnen und Forscher aus dem akademischen Umfeld richtet.

Das Konferenzmotto im Jahr 2019, „Software und Umwelt“, ist angesichts des immer stärker erkennbaren Klimawandels wichtiger denn je. Insbesondere die lokale Stuttgarter Industrie mit Ihren Stärken im Automobilbau, dem Versicherungswesen und der Verwaltung sieht sich mit besonderen Herausforderungen konfrontiert, die diese Entwicklung mit sich bringt. Software ist ein, voraussichtlich wesentlicher, Bestandteil, um die anstehenden Herausforderungen zu meistern und auch weiterhin weltweit wettbewerbsfähige Produkte und Dienstleistungen anbieten zu können.

Die seit 1995 zum 13. Mal stattfindende Tagung des GI-Fachausschusses Fachausschuss „Management der Anwendungsentwicklung und -wartung“ (WI-MAW) diskutiert Aspekte des Software Management in Zeiten einer turbulenten Umwelt. Die Menschheit steht vor tiefgreifenden Veränderungen in den verschiedensten Bereichen wie Klimawandel, Globalisierung, Bevölkerungswachstum bzw. alternde Bevölkerung, populistische Bewegungen, Digitalisierung und ihre Auswirkungen auf Zusammenleben, Fortbewegung und Wirtschaft. Der Umgang mit diesen Veränderungen stellt Unternehmen, Regierungen und Bürger vor neue Herausforderungen und bietet gleichzeitig vielfältige Potentiale.

Die angenommenen Papiere widmen sich hierbei der nachhaltigen Mobilität und industriellen IoT-Plattformen. Dabei spielt Digitalisierung nicht nur für eine nachhaltige Umwelt eine bedeutsame Rolle, sondern auch für (software-) plattformbasierte Geschäftsmodelle und Ökosysteme, deren Entwicklung in einer turbulenten Umwelt agile Vorgehensweisen erfordern.

Die Software Engineering bietet wieder ein „Best-of“ der international in Fachzeitschriften und Konferenzen veröffentlichten Arbeiten deutschsprachiger Autoren. Sie enthält eine große Bandbreite an Themen, die beispielsweise in Software & Systems Modeling, der ACM/IEEE International Conference on Automated Software Engineering, der International Conference on Agile Software Development, dem Journal of Systems and Software oder dem International Symposium on Software Testing and Analysis veröffentlicht wurden. Neu in diesem Jahr ist, dass Papiere, die nicht als Vortrag berücksichtigt werden konnten, zu einem Poster eingeladen wurden. Damit konnten wir eine noch größere Themenvielfalt für die SE erreichen.

Zum zweiten Mal ergänzt der Special Track „Erklärbare Software“ das Programm der SE. Hier haben wir um Einreichungen speziell für die neue Herausforderung, Software und ihre Ergebnisse besser erklären zu können, gebeten. Dies führte zu drei spannenden Vorträgen wiederum basierend auf existierenden Artikel, durch die wir das Thema weiter diskutieren konnten.

Schließlich wurde die SE noch durch vier Workshops ergänzt, in denen weitere spezifischere Themen im kleineren Kreis intensiv diskutiert wurden:

- 6th Collaborative Workshop on Evolution and Maintenance of Long-Living Systems (EMLS 2019)
- 16th Workshop on Automotive Software Engineering (ASE 2019)
- 1st Workshop on Software Engineering for Avionics Systems (AvioSE 2019)
- 2nd Workshop on Innovative Software Engineering Education (ISEE 2019)

Wir danken allen, die zum Gelingen der Konferenz beigetragen haben. Insbesondere danken wir den Autoren, ohne deren Beiträge die Konferenz gar nicht möglich gewesen wäre, den Gutachtern für die reibungslose Erstellung ihrer Reviews, den Keynote-Speakern für ihre anregenden Vorträge, den Organisatoren der Workshops, den Teilnehmern auf unserem Panel, den Sponsoren, der Universität Stuttgart und der Stadt Stuttgart, der GI e.V. und dem Stuttgarter Informatikforum infos e.V. für die Übernahme eines signifikanten Teils der Organisation und nicht zuletzt allen fleißigen Helferinnen und Helfern vor Ort, ohne deren Engagement die Konferenz nicht reibungslos ablaufen könnte.

Wir freuen uns auf eine abwechslungsreiche Konferenz mit vielen Beiträgen aus allen Dimensionen der Softwaretechnik und hoffen auf viele spannende Anregungen und neue Ideen.

Stuttgart, im Februar 2019

Steffen Becker, Ivan Bogicevic, Georg Herzwurm, Stefan Wagner

Sponsoren

Wir danken den folgenden Unternehmen für die Unterstützung der Konferenz.

Platin-Sponsor: Novatec Consulting GmbH

Gold-Sponsoren

Airbus

adesso

Bronze-Sponsoren

itestra

iss innovative software services GmbH

Tagungsleitung

General Chair:	Steffen Becker, Universität Stuttgart
Program Chair:	Stefan Wagner, Universität Stuttgart
Workshop Chair:	André van Hoorn, Universität Stuttgart

Programmkomitee SE

Stefan Wagner	Universität Stuttgart (Vorsitz)
Eric Bodden	Universität Paderborn
Ruth Breu	Universität Innsbruck
Stephan Diehl	Universität Trier
Sabine Glesner	TU Berlin
Rainer Koschke	Universität Bremen
Anne Koziolek	KIT
Walid Maalej	Universität Hamburg
Barbara Paech	Universität Heidelberg
Ina Schieferdecker	Freie Universität Berlin
Matthias Tichy	Universität Ulm
Uwe Zdun	Universität Wien

Programmkomitee SWM

Georg Herzwurm	Universität Stuttgart (Vorsitz)
Katharina Peine	Universität Stuttgart (Vorsitz)
Urs Andelfinger	Hochschule Darmstadt
Guido Baltes	HTWG Hochschule Konstanz
Martin Bertram	Commerzbank AG
Jens Borchers	Borchers BfI
Hans Brandt-Pook	Fachhochschule Bielefeld
Birgit Demuth	Technische Universität Dresden
Martin Engstler	Hochschule der Medien, Stuttgart
Masud Fazal-Baqaie	Universität Paderborn
Eckhart Hanser	DHBW Lörrach
Andreas Helferich	highQ Professional Services GmbH
Jürgen Jähnert	bwcon GmbH
Stefan Jesse	Bechtle AG
Hans-Bernd Kittlaus	InnoTivum Consulting
Nane Kratzke	Technische Hochschule Lübeck
Marco Kuhrmann	Clausthal University of Technology
Jens Lachenmaier	Universität Stuttgart
Martin Mikusz	mm1 Consulting & Management
Jürgen Münch	Hochschule Reutlingen
Wolfram Pietsch	Fachhochschule Aachen

Inhaltsverzeichnis

Keynotes

Workshops

Hauptprogramm

Session 1: Computational Science

Session 2: Emergentes Verhalten

Session 3: Programmanalyse und Verifikation I und Funktionale Sicherheit

Session 4: Traceability, Performanz und Continuous SE

Session 8: Modelle und Anforderungen

Session 9: Microservices und Produktlinien

Session 10: Software Management I

Session 11: Cyber-physische Systeme

Session 12: Software Management II, Security und Technologietransfer

Session 13: Architektur und DSLs

Session 14: Programmanalyse und Verifikation II

Session 15: Erklärbare Software

Poster

Keynotes

Keynote: Prof. Dr. Olaf Kolditz (Umweltinformatik, Helmholzzentrum für Umweltforschung)

1 Workflow development for the open-source porous media simulator OpenGeoSys

OpenGeoSys (OGS) is a scientific open source project for the development of numerical methods for the simulation of thermo-hydro-mechanical-chemical (THMC) processes in porous and fractured media. OGS is implemented in C++, it is object-oriented with an focus on the numerical solution of coupled multi-field problems (multi-physics). Parallel versions of OGS are available relying on both MPI concepts. Application areas of OGS in energy geosciences span from resources to waste management (e.g., water resources, geothermal energy, CO2 sequestration, geological waste deposition and energy storage). OGS is developed by the OpenGeoSys Community[1] consisting of the core developer team at UFZs Environmental Informatics Department as well as users from national and international research facilities and universities. We employ a sophisticated approach to open-source software development enabling developers a rapid iteration on new implementations. The following aspects will be present in talk: OGS software engineering, continuous integration, quality assurance (GitHub, Jenkins Docker DSL). With the OGS setup large parts of the whole software engineering infrastructure and processes are formalized and defined via DSLs in version controllable code-repositories allowing for easy contributing to and peer-reviewing of the code, infrastructure and process levels breaking up the black-box behaviour of traditional testing setups. The user can obtain up-to-date quality-tested binaries, test data files and documentation inducing a rapid iteration between user feedback and developer implementation. A clear versioning scheme of both software and data as well as archived software binaries allow reproducibility of scientific results. Software versions can be cited with a digital object identifier (DOI).

Bilke L. et al. (2019): Development workflows of the open-source porous media simulator OpenGeoSys, EGU General Assembly 2019, Vienna, Session ESSI3.1 – Free and Open Source Software (FOSS) for Earth and Space Science Informatics, accepted talk/poster.

Bilke L. (2018): Open-source computational frameworks for the simulation of multi-physics processes in porous media, Transport in Porous Media, in revision.

[1] www.opengeosys.org

doi:10.18420/se2019-01

Fischer T. et al. (2015): GO2OGS 1.0: a versatile workflow to integrate complex geological information with fault data into numerical simulation models. Geosci. Model Dev. 8 (11), 3681 – 3694.

Kolditz O. et al. (2018): Environmental Information Systems: Paving the Path for Digitally Facilitated Water Management (Water 4.0), Engineering, submitted.

Kolditz O. et al. (2018): Workflows in Energy Geotechnics: Examples and Perspectives, 8. Int. Congress on Environmental Geotechnics, invited lecture, 28.10-01.11.2018, Hangzhou.

Naumov et al. (2018): OpenGeoSys-6, In: Thermo-Hydro-Mechanical-Chemical Processes in Fractured Porous Media: Modelling and Benchmarking, vol. 3: 271-282, ISBN: 978-3-319-68224-2, DOI: 10.1007/978-3-319-68225-9

Rink K. et al. (2018): Virtual geographic environments for water pollution control. Int. J. Digit. Earth 11 (4), 397 – 407.

2 Zur Person

Prof. Kolditz is the head of the Department of Environmental Informatics at the Helmholtz Center for Environmental Research (UFZ). He holds a Chair in Applied Environmental System Analysis at the Technische Universität in Dresden. His research interests are related to environmental fluid mechanics, numerical methods and software engineering with applications in geotechnics, hydrology and energy storage. Prof. Kolditz is the lead scientist of the OpenGeoSys project (www.opengeosys.org), an open source scientific software platform for the numerical simulation of thermo-hydro-mechanical-chemical processes in porous media, in use worldwide. He studied theoretical mechanics and applied mathematics at the University of Kharkov, got a PhD in natural sciences from the Academy of Science of the GDR (in 1990) and earned his habilitation in engineering sciences from Hannover University (in 1996), where he became group leader at the Institute of Fluid Mechanics. Until 2001 he was full professor for Geohydrology and Hydroinformatics at Tübingen University and director of the international Master course in Applied Environmental Geosciences. Since 2007 he is the speaker of the Helmholtz graduate school for environmental research HIGRADE. Prof. Kolditz is Editor-in-Chief of two international journals Geothermal Energy (open access) and Environmental Earth Sciences (ISI). Prof. Kolditz is the leading scientist of the Sino-German network initiative “Research Centre for Environmental Information Science-RCEIS” dedicated to the development of comprehensive data integration and knowledge platforms in China (funded by the Helmholtz Association, in cooperation with the Chinese Academy of Sciences, Institute for Geographical Sciences and Natural Resources Research) and of the joint priority project “Managing Water Resources in Urban Catchments - Chaohu” linked to the Mega-Water Project (funded by the German Federal Ministry of Education and Research, in cooperation with the Tongji University). Prof. Kolditz was awarded a professorship under the CAS President’s International Fellowship (PIFI) in 2015.

Keynote: Jonas Huthmacher (Senior Product Manager beim Lab1886, Daimler AG)

1 Lab1886 – der Inkubator der Daimler AG. Wie wir neue, digitale Geschäftsmodelle entwickeln an einem Beispiel aus der Praxis.

Zur Person:

Jonas spent most of his professional life abroad at Microsoft in the AI Research organization, in both sales and engineering teams. Now being with the Lab 1886, he is working as the project lead on building new software businesses for the DAG.

doi:10.18420/se2019-02

Keynote: Prof. Dr. Stefanie Betz (Sozioinformatik, Hochschule Furtwangen)

1 Sustainability Debt und Entscheidungen im Software Engineering

Nachhaltigkeit ist für die Gesellschaften unseres Planeten von grundlegender Bedeutung, ebenso wie Software Systeme immer mehr Teil der heutigen Gesellschaften werden. Daher gewinnt Nachhaltigkeit auch immer mehr an Relevanz im Software Engineering und es wurden erste Ansätze entwickelt, um Nachhaltigkeit bei dem Software System Design zu berücksichtigen. Dennoch bleibt es schwer die erst später eintreffenden Auswirkungen von Entscheidungen, die beim System Design getroffen werden, zu erkennen und zu bewerten. Um diese schwierige Aufgabe zu unterstützen, wird in der Keynote die Metapher „Sustainability Debt" vorgestellt. Die Metapher hilft bei der Identifikation, Dokumentation und Kommunikation von Nachhaltigkeitsfragen im Software Engineering. Sie baut auf der bestehenden Metapher des „Technical Debt" auf und erweitert diese um vier weitere Dimensionen der Nachhaltigkeit (individuell, sozial, ökologisch, ökonomisch). Neben der Bedeutung der Metapher Sustainability Debt und ihrer Verwendung im Software Engineering wird im Rahmen der Keynote auch darauf eingegangen wie im Software Engineering Entscheidungen getroffen werden. Da Entscheidungen im Rahmen des Sustainability Debts immer bedeuten einen Kompromiss zu schließen zwischen zeitnahen und entfernten Ergebnissen. Bei solchen intertemporalen Entscheidungen werden entfernte Ergebnisse oft als weniger bedeutsam bewertet als zeitnahe, was berücksichtigt werden muss, um eine entsprechende Entscheidungsunterstützung zur Verringerung des Sustainability Debts zu liefern.

2 Zur Person

Dr. Stefanie Betz ist Professorin für Sozio-Informatik im Fachbereich Informatik der Hochschule Furtwangen. Ihre Forschung konzentriert sich auf nachhaltiges Software Systems Engineering, insbesondere aus der Sicht des Requirements Engineering und des Business Prozess Engineering. Frau Betz promovierte 2013 in angewandter Informatik am Karlsruher Institut für Technologie (KIT). Ihre Forschungsarbeiten wurden auf bedeutenden Konferenzen und in renommierten Zeitschriften veröffentlicht, darunter die International Conference on Software Engineering (ICSE), die International Requirements Engineering Conference (RE) und das Journal of Software and Systems (JSS), IEEE Software, Empirical Software Engineering und Expert Systems. Sie ist unter anderem Mitglied der internationalen

doi:10.18420/se2019-03

ICT4S-Community, in der sie als Mitglied des Programmkomitees und als Reviewer für Konferenzen und Journals wie ICT4S, RE, JSS, Sustainability, Sustainable Computing, Informatics and Systems, und Empirical Software Engineering tätig ist.

Workshops

6th Collaborative Workshop on Evolution and Maintenance of Long-Living Systems

Robert Heinrich[1] Reiner Jung[2] Marco, Konersmann[3] Eric.Schmieders[4]

Die Digitalisierung ist eine zentrale gesellschaftliche und technologische Entwicklung. Sie wandelt nicht nur analoge Prozesse hin zu digitalen, sondern bedingt einen anderen Umgang mit Daten und Software. Beide erfahren häufige Änderungen der Nutzung. Dies führt zu ständigen Änderungen der Anforderungen und des technologischen Kontextes. Dadurch entstehen u. a. inkonsistente Anforderungsspezifikationen, Architekturerosion und SLA-Verletzungen. Dieses Problem ist vor allem in der industriellen Praxis relevant, in der ein solches System nicht nur eine initiale Entwicklung erfährt, sondern ständig weiterentwickelt werden muss.

Die EMLS-Workshopreihe setzt sich in den letzten Jahren mit den Herausforderungen beim Übergang zwischen den verschiedenen Software-Entwicklungsphasen auseinander. Für den Workshop sind hierbei Erfahrungen mit konkreten Technologien und Lösungsstrategien ebenso gefragt wie Problemstellungen und Evaluationsstrategien. Die EMLS-Workshops schaffen ein Forum, auf dem die Teilnehmer in kollaborativer Weise an gemeinsamen Themen arbeiten. Dabei streben wir einen Austausch zwischen Forschung und der Industrie an.

Ziel des Workshops ist es den Austausch zu den o.g. Themen zu fördern. Der Workshop bietet dazu ein Forum um Herausforderungen, Lösungsansätze und Erfahrungsberichte zu diskutieren. Dabei liegt der Fokus des Workshops auf intensiven Diskussionen, denen viel Zeit eingeräumt wird. Abschließend werden die akzeptierten Beiträge, Ergebnisse und eine Zusammenfassung veröffentlicht.

Der Workshop nutzt, wo möglich, die thematische Arbeit in Kleingruppen um den Austausch zwischen den Teilnehmern zu fördern. Zu Beginn jeder Session stellen die Autoren und Autorinnen ihre Themen in kurzen Impulsvorträgen vor. Anschließend werden zu diesen Themen Kleingruppen gebildet. Für jede Gruppe stellen die Organisatoren einen Moderator oder Moderatorin zur Leitung der Diskussion. Die Gruppen werden mit passenden Diskussionsmaterial, wie Flipchart und Diskussionskarten ausgestattet, welche zur Steuerung

[1] Karlsruher Institut für Technologie, Am Fasanengarten 5, 76131 Karlsruhe, robert.heinrich@kit.edu
[2] Christian-Albrechts-Universität zu Kiel, Christian-Albrechts-Platz 4, 24118 Kiel, reiner.jung@email.uni-kiel.de
[3] Universität Duisburg-Essen, Gerlingstraße 16, 45127 Essen, marco.konersmann@uni-due.de
[4] Landesamt für Zentrale Polizeiliche Dienste NRW, Schifferstraße 10, 47059 Duisburg

doi:10.18420/se2019-04

der Diskussion und zur Vorbereitung der Ergebnispräsentation genutzt werden können. Am Ende der Session werden die Ergebnisse der Diskussionen in den Kleingruppen zusammengetragen und anschließend im Plenum vorgestellt.

Um die Ergebnisse festzuhalten, nach außen hin sichtbar zu machen, und den Beteiligten eine Zusammenfassung aller behandelten Themen zu gewährleisten, werden die Beiträge und eine Zusammenfassung der Ergebnisse veröffentlicht. Zudem werden die erarbeiteten Präsentationen fotografiert und auf der Internetseite *emls.paluno.uni-due.de* bereitgehalten.

Der Workshop wird in diesem Jahr mit einer Keynote von Prof. Dr. Lars Grunske von der Humboldt-Universität zu Berlin eröffnet. Die Keynote mit dem Titel **Software engineering challenges for evolving data-intensive scientific software** befasst sich mit den Herausforderungen dieser speziellen Domäme im Rahmen der Evolution.

Im Anschluss wird der Beitrag **Clean Code: On the Use of Practices and Tools to Produce Maintainable Code for Long-Lived Software** von Björn Latte, Sören Henning und Maik Wojcieszak vorgestellt und diskutiert. Die Autoren und Autorinnen bringen mit ihrem Beitrag einen Erfahrungsbericht über den Einsatz verschiedener, kombinierter BestPractices im Software Engineering. Die Erfahrungen mit Clean Code, Code Reviews, Test-driven Development, statischer Code-Analyse, und Continuous Delivery tragen vor allem dazu bei, Evolution auf der Ebene des Programmcodes zu kontrollieren.

Langlebige Systeme sind oft mehrfach in Ihrem Lebenszyklus von Refactoring betroffen. Der Beitrag **A Reproduction Study of Refactoring Detection Tools** von Liang Tan und Christoph Bockisch beschreibt und evaluiert eine Übersicht von Werkzeugen zur Erkennung von Refactorings. Im Anschluss wird das Werkzeug *RefactoringMiner* eingehend untersucht und die Vor- und Nachteile herausgestellt.

Der Beitrag **Evolving a Use Case for Industry 4.0 Environments Towards Integration of Physical Access Control** von Stephan Seifermann und Maximilian Walter beschreibt ein Evolutionsszenario im Bereich der Industrie 4.0, bei dem defekte Teile zu Äderungen in der Zugangskontrolle für das Personal der Qualitätssicherung bedingt. Das Evolutionsszenario wird im Workshop im Detail besprochen und könnte in Zukunft als Basis für Benchmarks dienen. Der Workshop schließt mit einer Zusammenfassung der Beiträge und Diskussionen. Die Organisatoren danken allen Beitragenden und Teilnehmenden für den EMLS'19.

16. Workshop Automotive Software Engineering

Steffen Helke[1] Ina Schaefer[2] Andreas Vogelsang[3]

Wie seine Vorgänger setzt sich der 16. Workshop Automotive Software Engineering mit der Problematik der Softwareentwicklung im Automobilbereich und folglich mit dafür geeigneten Methoden, Techniken und Werkzeugen auseinander. Die Automobilsoftware spielt heutezutage mehr denn je mit zunehmend vernetzten Fahrzeugen und modernen Fahrerassistenzfunktionen einschließlich des vollautomatisierten Fahrens eine wichtige Rolle.

Dabei steigt die Systemkomplexität nicht nur stetig an, sondern auch strengere Anforderungen an Zuverlässigkeit, Sicherheit (Security und Safety) und Datenschutz (Privacy) müssen insbesondere erfüllt werden. Der Trend zur Vernetzung hat das Fahrzeug längst erreicht. Zudem bauen immer mehr Funktionen auf Sprachsteuerung auf, um eine Handybedienung während des Fahrens zu ermöglichen. Das Autofahren wird somit durch voranschreitende „digitale Kulturen" verändert: Menschen werden bald auf Dienste wie WhatsApp, Skype oder sogar Facebook vom Fahrzeug aus zugreifen können.

Der Austausch und die Diskussion darüber, wie aktuelle Herausforderungen im Automotive Software Engineering gemeistert werden können, sind Hauptziele des Workshops. Die thematische Ausrichtung bietet viele Anknüpfungspunkte zur Tagung Software Engineering (SE) des Fachbereichs Softwaretechnik. Der Workshop richtet sich gleichermaßen an Forscher, Entwickler und Anwender aus der Automobilindustrie sowie an Wissenschaftler aus Forschungsinstituten und Hochschulen, die im Gebiet Automotive Software Engineering arbeiten. Im Fokus stehen traditionell weniger theoretische, als vielmehr praxisnahe Arbeiten.

Es wurden für jeden eingereichten Beitrag zwei Gutachter festgelegt. Nach Auswertung der Gutachten konnten vier Papiere zur Veröffentlichung akzeptiert werden. Herzlichen Dank an alle Gutachter, die sich mit viel Engagement in den Begutachtungsprozess eingebracht haben.

Wie schon in den vergangenen Jahren wird das Workshop-Programm mit einer Keynote eröffnet. Wir bedanken uns bei Prof. Dr. Matthias Tichy (Institut für Softwaretechnik und Programmiersprachen, Universität Ulm), der spontan zusgesagt hat und zum Gelingen dieses Workshops entscheidend beiträgt. Zusätzlich gibt es in diesem Jahr noch drei eingeladene

[1] Brandenburgische Technische Universität Cottbus-Senftenberg, steffen.helke@b-tu.de
[2] Technische Universität Braunschweig, i.schaefer@tu-braunschweig.de
[3] Technische Universität Berlin, andreas.vogelsang@tu-berlin.de

doi:10.18420/se2019-05

Vorträge von Malte Mauritz (Fraunhofer ISST, Braunschweig), Benedikt Walter (Daimler AG, Stuttgart) und Philipp Hohl (Daimler AG, Stuttgart).

Programmkomitee

Dr. Christian Allmann	Audi AG
Prof. Dr. Marcel Baunach	Technische Universität Graz
Dr. Mirko Conrad	samoconsult GmbH
Dr. Heiko Dörr	Model Engineering Solution GmbH
Prof. Dr. Volker von Holt	Ostfalia Hochschule für angewandte Wissenschaften
Prof. Dr. Thomas Kropf	Robert Bosch GmbH
Dr. Thomas Noack	Individual Standard IVS GmbH
Prof. Dr. Dirk Nowotka	Universität Kiel
Prof. Dr. Jörn Schneider	Hochschule Trier
Dr. Thomas Sauer	Volkswagen AG

Organisation

Prof. Dr. Andreas Vogelsang	Technische Universität Berlin
Prof. Dr. Ina Schaefer	Technische Universität Braunschweig
Prof. Dr. Steffen Helke	Brandenburgische Technische Universität Cottbus-Senftenberg

Die Organisation erfolgte in enger Abstimmung mit der GI-Fachgruppe Automotive Software Engineering[4], die den ASE-Workshop seit vielen Jahren veranstaltet.

[4] http://fg-ase.gi.de/

1st Workshop on Avionics Systems and Software Engineering (AVIOSE'19)

Björn Annighöfer[1] Andreas Schweiger[2] Marina Reich[3]

Abstract: Companies are struggling with the complexity of digital avionics systems. Millions of man months are required for the development of digital airborne systems. Moreover, the complexity of functions, the number of vehicles, and systems continuously rises. There is a high demand for more efficient methods and tools of systems and software engineering. The AVIOSE workshop establishes a new forum for the exchange for the people working on simplifying, shortening, and maturing the creation of avionics systems.

Keywords: avionics; systems engineering; software engineering; formal methods; model-based; requirements; qualification; certification; simulation; processes

1 Introduction

Software development in the aerospace domain is driven by increasing complexity, new application potentials, and rising certification effort. Future applications demand for new software development methodologies, e.g. autonomous air transport and commercial UAVs and further enhancement of existing functionality. At the same time, there are issues in communication and navigation in airspace, multi-core processors or artificial intelligence. Many projects struggle with continuously rising effort required for the implementation, development, configuration, integration, and qualification of digital airborne systems, i. e. avionics.

Progress is made in various disciplines, as formal methods, process automation, simulation and regulations, but results often stay in their (academic) community. For instance, formal methods like CSP, FOCUS, π-Calculus, Hoare-Calculus, or Isabelle are around already for several years or even decades. However, they are still not widely adopted in industry.

Therefore, it is about time to create a central point of exchange for the engineers, computer scientists, and industry and all other involved parties with a holistic meeting on "Avionics Systems and Software Engineering". Invited are key players from the industrial and academic community to take part in a one-day workshop held in the context of the "Combined Conference Software Engineering & Management" in February 2019 in Stuttgart (SE'2019).

[1] University of Stuttgart, Institute of Aircraft Systems (ILS), Germany, bjoern.annighoefer@ils.uni-stuttgart.de
[2] Airbus Defence and Space GmbH, Manching, Germany, andreas.schweiger@airbus.com
[3] Airbus Defence and Space GmbH, University of Chemnitz, Manching, Germany, marina.reich@airbus.com

doi:10.18420/se2019-06

2 Workshop Goals

The core objectives of the workshop are: (1) Providing a forum for novel and innovative approaches and tools of systems and/or software engineering methods within the avionics domain. (2) Bring together all involved disciplines, e.g. computer scientists theories, engineering methods, and industrial processes. (3) Derive a common consensus on the most important challenges within the avionics sector within the next decade. (4) Discuss between participants on what way to proceed in order to solve the problems of the future. With these in mind, the topics of interest for the workshop are defined to be: *(A) Development technologies:* Requirements engineering, modeling languages and tools, transfer of modeling techniques to industrial application, verification via testing and formal methods, security & safety; *(B) Development methods:* Certification, agile development, interaction with other domains (e. g. physics, psychology); *(C) Product technologies:* Applications of artificial intelligence (including verification), autonomous systems; *(D) Additional challenges:* Reference architectures for hard- and software and interfaces between sub-systems, sensors and sensor fusion, Integrated Modular Avionics (IMA), obsolescence (management).

3 Talks and Contributions

The workshop is organized as one-day conference with contributions from researchers as specialists of their specific domain. Each author has the forum to present and discuss his results, such that important and novel ideas from the AVIOSE domain are shared in detail. On the high-level a motivation and general challenges are provided by invited long-term experts from the academic and industrial world as keynote speakers. In addition, it is envisaged to collect common topics, cross-relations, and lessons learned during the workshop; to evaluate and discuss those within a moderated podium discussion in order to find and define common results of the workshop.

In total six papers with topics from formal verification and testing, early validation as well as requirements process automation have been chosen for a presentation. Despite the SE being a German institution, the AVIOSE attracted authors of various nationalities. Most of them situated in Germany, but it got high European interest and includes a European contribution. The accepted publications are entitled: “MODCAP: A Platform for Cooperative Search and Rescue Missions”, “Model-Based Engineering for Avionics: Will Specification and Formal Verification e.g. based on Broy’s Streams become feasible?”, “Test Sequence Generation From Formally Verified SysML Models”, “Towards Computer-Aided Software Requirements Process”, “Tool Chain for Avionics Design, Development, Integration and Test”, “Using Runtime Monitoring to Enhance Offline Analysis”.

In addition, two keynotes by Prof. Dr.-Ing. Reinhard Reichel from the University of Stuttgart and by Franz Münz from Airbus Defence and Space are given.

4 Conclusion

Throughout the workshop, current industrial and academic challenges are gathered. All participants contribute to the identification of worthwhile problems and their ratings. Solutions, approaches, and methods are debated in plenary and panel discussion. The combination of industrial and academic participants increases awareness and information level on both sides.
Throughout the submissions we recognize the particular efforts and the resulting promising trends in model-based and model-driven development. Nevertheless, still a major of contributions focuses on adjustments, adaptations, extensions, and customization of the models to use cases in our domain and the development processes.
Though advertised in the call for papers, no contribution concerning new technologies such as AI has been received. Therefore, discussions are directed in this area during the workshop. The same holds true for certification endeavors. Exchanging experience and clarity in both fields are strongly recommended by the program committee and thus make up significant open topics for discussions during the workshop.

Acknowledgments

Many people contributed to the success of this workshop. First of all, we want to give thanks to the authors and presenters of the accepted papers and especially our keynote speakers, Prof. Dr.-Ing. Reinhard Reichel, University of Stuttgart and Franz Münz, Airbus Defence and Space GmbH.
Furthermore, we want to express our gratitude to the SE'2019 organizers for supporting our workshop. Finally, we are glad that these people (listed in alphabetic order) served the program committee, soliciting papers and writing peer reviews: Jun.-Prof. Björn Annighöfer (University of Stuttgart), Sven Bacher (Philotech Systementwicklung und Software GmbH), Ulrich Fräbel (Rolls-Royce Group), Prof. Dr. Ralf God (Technische Universität Hamburg), Prof. Dr. Lars Grunske (Humboldt-Universität zu Berlin), Prof. Dr. Eric Knauss (University of Gothenburg), Jürgen Krug (Diehl Aerospace GmbH), Alfred Lief (Airbus Defence and Space GmbH), Dr. Winfried Lohmiller (Airbus Defence and Space GmbH), Prof. Dr. Alexander Pretschner, (Technische Universität München), Dr. Stephan Rudolph (Northrop Grumman LITEF GmbH), Dr. Harald Rueß, (fortiss GmbH), Prof. Dr. Bernhard Rumpe (RWTH Aachen University), Dr. Andreas Schweiger (Airbus Defence and Space GmbH), Prof. Dr. Matthias Tichy (Universitaet Ulm), Prof. Dr. André Windisch (Airbus Defence and Space GmbH, Technische Universität Chemnitz).
With the engagement of contributors, program committee, and organization team, the expected interest in the topic is highlighted. The organization committee will leverage the acquaintance with the workshop for the future and to repeat the workshop as part of SE'2020.

2nd Workshop on Innovative Software Engineering Education

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

Abstract: 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. The workshop focuses on three main topics: (1) project courses with industry, (2) active learning in large courses, and (3) digital teaching and online courses.

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

1 Introduction

Software engineering instructors face more and more challenges due to the growing number of students. Motivating students to actively participate in a course is especially difficult in large classes. Even though practice-oriented and project-based training becomes increasingly important, such project courses in cooperation with industry 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 *2nd 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.

[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

doi:10.18420/se2019-07

The workshop provides an interactive forum with paper and poster presentations, and room for discussion. Authors give short talks about their contributions, which are followed by intensive discussions. Discussions are moderated by selected supporters, who prepare (critical) questions thus stimulating and guiding the discussion. The overall goal is 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 9 submissions covering different topics in the field of software engineering education of which 6 submissions (1 full paper, 4 short papers, 1 poster) have been accepted and selected for presentation. The accepted papers address topics such as tool-support for automating parts of the education thus reducing effort, e.g. the automatic assessment of text exercises, and tool support for face-to-face teaching. Other topics are the use of essence in a software development course and teaching wearable device development with a dedicated toolkit. Furthermore, code process metrics in programming education and interdisciplinary system courses to teach agile systems engineering are discussed.

Marcus Deininger (HFT Stuttgart) starts the workshop with his keynote on approaches and experiences in higher education for software project practice. He discusses the aims of computer science education, which only partially meet the requirements of real life software development. Following the keynote, the authors present their papers and posters briefly to initiate the discussion. All authors additionally present their paper as posters in a dedicated poster session, which allows for building small groups discussing topics of interest.

4 Conclusion

The contributions to the workshop highlight innovative approaches in software engineering education and emphasize that education is an important research topic. This motivates for additional workshops in the future.

Acknowledgements

Many people contributed to the success of this workshop. We want to thank the authors and presenters of the accepted papers and the keynote speaker Marcus Deininger. We express our gratitude to the SE'19 conference organizers for supporting our workshop. We are glad that the following people (listed in alphabetic order) served the program committee, soliciting papers and writing peer reviews: Jürgen Börstler, Bernd Brügge, Birgit Demuth, Marlo Häring, Regina Hebig, Michael Hilton, Carsten Kleiner, Jürgen Münch, Dirk Riehle, Andreas Seitz, and Swapneel Sheth. In addition, we like to thank the two reviewers Mariana Avezum and Nadine von Frankenberg for their support.

Hauptprogramm

Session 1: Computational Science

Software Engineering for Computational Science

Arne Johanson[1] Wilhelm Hasselbring[2]

Abstract: Despite the increasing importance of in silico experiments to the scientific discovery process, state-of-the-art software engineering practices are rarely adopted in computational science. To understand the underlying causes for this situation and to identify ways to improve it, we conducted a literature survey on software engineering practices in computational science. We identified recurring key characteristics of scientific software development that are the result of the nature of scientific challenges, the limitations of computers, and the cultural environment of scientific software development. Our findings allow us to point out shortcomings of existing approaches for bridging the gap between software engineering and computational science and to provide an outlook on promising research directions that could contribute to improving the current situation.

Keywords: Computational Science; Model-driven software engineering; Software architecture

Computational science (also scientific computing) involves the development of models and simulations to understand natural systems answering questions that neither theory nor experiment alone are equipped to answer [Rü18]. Computational science is a multidisciplinary field lying at the intersection of mathematics and statistics, computer science, and core disciplines of science. Despite the increasing importance of so-called in-silico experiments to the scientific discovery process, well-established software engineering practices are rarely adopted in computational science [JH18]. However, meanwhile the computational science community starts to appreciate that software engineering is central to any effort to increase computational science's software productivity [Rü18, page 737]. Among the methods and techniques that software engineering can offer to computational science are

- model-driven software engineering with domain specific languages [JH14; JH17; Jo16b; Jo17],
- modular software architectures [Ha18; HS17; Jo16a],
- specific requirements engineering techniques [Th09], and
- testing without test oracles [KB14].

Computational science requires maintainable, long-living software [Go15], enabled by domain-specific software engineering methods and techniques.

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

doi:10.18420/se2019-08

Literatur

[Go15] Goltz, U.; Reussner, R.; Goedicke, M.; Hasselbring, W.; Märtin, L.; Vogel-Heuser, B.: Design for future: managed software evolution. Computer Science – Research and Development 30/3, S. 321–331, Aug. 2015.

[Ha18] Hasselbring, W.: Software Architecture: Past, Present, Future. In: The Essence of Software Engineering. Springer, S. 169–184, 2018, URL: https://doi.org/10.1007/978-3-319-73897-0_10.

[HS17] Hasselbring, W.; Steinacker, G.: Microservice Architectures for Scalability, Agility and Reliability in E-Commerce. In: Proceedings 2017 IEEE International Conference on Software Architecture Workshops (ICSAW). S. 243–246, 2017.

[JH14] 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, S. 1–8, Aug. 2014, URL: https://doi.org/10.1145/2642803.2642820.

[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, S. 2206–2236, Aug. 2017, URL: http://rdcu.be/urXK.

[JH18] Johanson, A.; Hasselbring, W.: Software Engineering for Computational Science: Past, Present, Future. Computing in Science & Engineering 20/2, S. 90–109, März 2018, URL: https://doi.org/10.1109/MCSE.2018.021651343.

[Jo16a] Johanson, A.; Flögel, S.; Dullo, C.; Hasselbring, W.: OceanTEA: Exploring Ocean-Derived Climate Data Using Microservices. In: Proceedings of the Sixth International Workshop on Climate Informatics (CI 2016). S. 25–28, 2016.

[Jo16b] 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. Chapman und Hall/CRC, S. 175–200, 2016.

[Jo17] Johanson, A. N. et al.: 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.

[KB14] Kanewala, U.; Bieman, J. M.: Testing scientific software: A systematic literature review. Information and Software Technology 56/10, S. 1219–1232, 2014.

[Rü18] Rüde, U.; Willcox, K.; McInnes, L. C.; Sterck, H. D.: Research and education in computational science and engineering. Siam Review 60/3, S. 707–754, 2018.

[Th09] Thew, S.; Sutcliffe, A.; Procter, R.; De Bruijn, O.; McNaught, J.; Venters, C. C.; Buchan, I.: Requirements engineering for E-science: experiences in epidemiology. IEEE Software 26/1, S. 80–87, 2009.

Session 2: Emergentes Verhalten

S. Becker, I. Bogicevic, G. Herzwurm, S. Wagner (Hrsg.): SE/SWM 2019,
Lecture Notes in Informatics (LNI), Gesellschaft für Informatik, Bonn 2019 47

Validierung und Verifikation von emergentem Verhalten im Software Engineering

Ergebnisse eines Vergleichs unterschiedlicher Suchmethoden

Jennifer Brings[1] Marian Daun[1] Markus Kempe[1] Thorsten Weyer[1]

Abstract: Dieser Vortrag berichtet von dem Beitrag *On Different Search Methods for Systematic Literature Reviews and Maps: Experiences from a Literature Search on Validation and Verification of Emergent Behavior* [Br18], der auf der *22nd International Conference on Evaluation and Assessment in Software Engineering 2018* vorgestellt und in dem Konferenzband veröffentlicht wurde.

Keywords: Emergent Behavior; Verification; Systematic Literature Review

1 Einleitung

Die Verifikation emergenter Verhaltenseigenschaften stellt eine Herausforderung bei der Entwicklung kollaborierender cyber-physischer Systeme dar, da diese Systeme ein gemeinsames globales Verhalten im Zusammenspiel der unterschiedlichen Einzelsysteme generieren [Br17]. Im Rahmen einer systematischen Literaturrecherche wurde gezeigt, welche Ansätze in verwandten Software Engineering-relevanten Bereichen existieren. Die Ergebnisse der Untersuchung geben darüber hinaus Aufschluss über den Einfluss der Suchmethode auf die Ergebnisse systematischer Literaturrecherchen und Mapping Studies.

2 Verifikation emergenter Verhaltenseigenschaften

Die Untersuchung hat gezeigt, dass die Verifikation emergenter Verhaltenseigenschaften in vielen Subdisziplinen des Software Engineering als Herausforderung angesehen wird. Dementsprechend wurden in den verschiedenen Bereichen unterschiedliche Lösungen entwickelt. Trotzdem lässt sich erkennen, dass die verschiedenen Lösungsansätze grundlegend auf die gleichen Verifikationstechniken zurückgreifen. Am häufigsten finden Model-Checking und Simulation Anwendung, aber auch Laufzeitmonitoring, statische Analysen, Testen und Theoremprüfung (in absteigender Reihenfolge) werden vorgeschlagen. Auffällig an den Ergebnissen ist eine große Vielzahl an Veröffentlichungen, die keine

[1] Universität Duisburg-Essen, paluno - The Ruhr Institute for Software Technology, 45127 Essen, Deutschland
{jennifer.brings,marian.daun,thorsten.weyer}@paluno.uni-due.de

doi:10.18420/se2019-09

Verifikationstechnik, sondern Formalisierungen des Problemraums vorschlagen, um später den Einsatz von formalen Verifikationstechniken unterstützen zu können.

3 Einfluss der Suchmethode auf systematische Literaturrecherchen

Über die Zusammenfassung der bisherigen Arbeiten zur Verifikation emergenter Verhaltenseigenschaften hinausgehend gibt die Studie Aufschluss darüber, welche Auswirkungen die Suchmethode auf das Suchergebnis hat. Abbildung 1 illustriert den Studienaufbau.

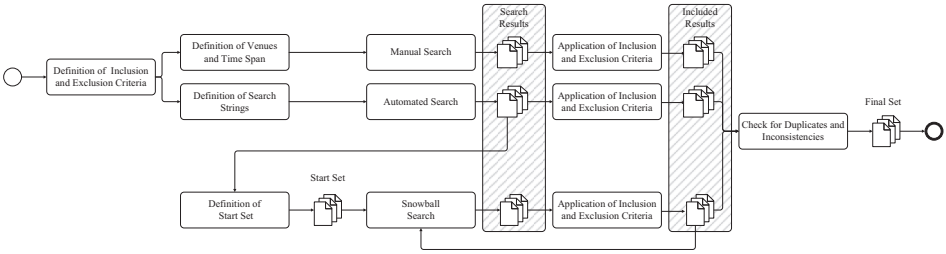


Abb. 1: Studienaufbau

Unsere Ergebnisse zeigen, dass Vorwärts-/Rückwärtssuchen und automatisierte Suchen ungefähr gleichermaßen effektiv sind und dass automatisierte Suchen die mit Abstand effizienteste Suchmethode sind. Jedoch hat sich auch herausgestellt, dass die verschiedenen Suchmethoden zu fast vollständig unterschiedlichen Sammlungen von Veröffentlichungen führen, was teilweise zu unterschiedlichen Schlussfolgerungen zum Stand der Wissenschaft bezüglich Valididerung und Verifikation von emergentem Verhalten führt. Folglich ist die Repräsentativität einer systematischen Literaturrecherche von den verwendeten Suchmethoden abhängig. Die Nutzung von nur einer Suchmethode kann bei systematischen Literaturrecherchen zu Verzerrungen der Ergebnisse und somit zu falschen Schlussfolgerungen führen. Dies erscheint insbesondere dann der Fall zu sein, wenn das Thema der Suche - wie im vorliegenden Fall - auf Resonanz in unterschiedlichen Disziplinen stößt.

Literaturverzeichnis

[Br17] Brings, J.: Verifying Cyber-Physical System Behavior in the Context of Cyber-Physical System-Networks. In: 2017 IEEE 25th International Requirements Engineering Conference (RE). S. 556–561, September 2017.

[Br18] Brings, Jennifer; Daun, Marian; Kempe, Markus; Weyer, Thorsten: On Different Search Methods for Systematic Literature Reviews and Maps: Experiences from a Literature Search on Validation and Verification of Emergent Behavior. In: Proceedings of the 22nd International Conference on Evaluation and Assessment in Software Engineering 2018 - EASE'18. ACM Press, Christchurch, New Zealand, S. 35–45, 2018.

Session 3: Programmanalyse und Verifikation I und Funktionale Sicherheit

S. Becker, I. Bogicevic, G. Herzwurm, S. Wagner (Hrsg.): SE/SWM 2019,
Lecture Notes in Informatics (LNI), Gesellschaft für Informatik, Bonn 2019 51

A Unified Lattice Model and Framework for Purity Analyses

Dominik Helm, Florian Kübler, Michael Eichberg, Michael Reif, Mira Mezini[1]

Abstract: This paper was presented in 2018 at *the 33rd ACM/IEEE International Conference on Automated Software Engineering* and proposes a framework for purity analyses. Analyzing methods in object-oriented programs whether they are side-effect free and also deterministic, i.e., *mathematically pure*, has been the target of extensive research. Identifying such methods helps to find code smells and security related issues, and helps analyses detecting concurrency bugs. *Pure* methods are further used for formal specifications and proving the *pureness* is necessary to ensure correct specifications. However, no common terminology exists which describes the purity of methods. Furthermore, some terms (e.g., *pure* or *side-effect free*) are used inconsistently. Further, all current approaches only report selected purity information making them only suitable for a smaller subset of the potential use cases. We present a fine-grained unified lattice model which puts the purity levels found in the literature into relation and which adds a new level that generalizes existing definitions. We have also implemented a scalable, modularized purity analysis which produces significantly more precise results for real-world programs than the best-performing related work. The analysis shows that all defined levels are found in real-world projects.

Keywords: Purity; Side-effects; Static Analysis; Lattice; Java.

1 Summary

Identifying side-effect free and also deterministic, i.e., *mathematically pure*, methods in object-oriented programs helps to improve subsequent analyses for finding bugs [Fi08]. Pure methods—written in programming languages such as Java—are also used by formal verification approaches as the foundation for the respective specifications [DL07]. In that case, it is necessary to prove a method's purity to ensure that the formal specifications are correct. Recent trends towards a more functional style of programming relying on pure methods, also demonstrate the overall relevance.

We present a fine-grained unified lattice model for specifying a method's purity. In the model, each of the 13 lattice elements has a well-defined semantics and is put into relation to the purity levels found in the literature. In addition to previously defined purity levels, the model is extended by the level *Contextual Purity* which generalizes the so-called *External Purity* [BF09] for methods that may modify their parameters but no static state. Being able to ignore specific operations in specific contexts [SCOD16], e.g., logging in business

[1] Technische Universität Darmstadt, FG Softwaretechnik, Germany
{helm,kuebler,eichberg,reif,mezini}@cs.tu-darmstadt.de

doi:10.18420/se2019-10

applications, is also supported and generalized to *Domain-specific Purity*. The proposed model is sufficiently detailed for all identified use cases. Furthermore, the purity levels (*External* and *Contextual* Purity) support purity analyses to rate methods as pure if the called methods have side-effects that are limited to the caller. Therefore, the lattice model is also a suitable target for modular purity analyses that reason about each method in isolation.

Additionally, we present a scalable, purity analysis, called OPIUM which is implemented using OPAL [Ei18], that produces more precise results for real-world code than the best-performing state-of-the-art tool ReIm [HM12]. OPIUM can, e.g., compute the purity for all ≈16000 methods of Batik—which requires computation of the purity of all library methods transitively used by Batik—in 103s on a modern computer which is on par with Batik while it analyzes Xalan in 104s, ≈25% faster than ReIm. OPIUM not only identifies more *side-effect free* methods that state-of-the-art tools including ReIm, but it also provides more fine-grained results by reporting *pure*, *externally* and *contextually pure* and *side-effect free* methods as well as *domain-specific* methods that adhere to these properties only in a given domain, e.g. excluding logging or exceptions. Using the analysis, we show that all defined levels are relevant when analyzing real-world projects such as those from XCorpus or the Java and Scala Development Kits. The analysis infers purity for individual methods in isolation. It relies on the results of several independent analyses, making it modularly composable with supporting analyses with different precision/performance trade-offs.

OPIUM is available at `www.opal-project.de/Opium.html`.

References

[BF09] Benton, William C; Fischer, Charles N: Mostly-functional behavior in Java programs. In: International Workshop on Verification, Model Checking, and Abstract Interpretation. Springer, pp. 29–43, 2009.

[DL07] Darvas, Ádám; Leino, K Rustan M: Practical reasoning about invocations and implementations of pure methods. In: FASE. volume 4422. Springer, pp. 336–351, 2007.

[Ei18] Eichberg, Michael; Kübler, F; Helm, D; Reif, M; Salvaneschi, G; Mezini, M: Lattice Based Modularization of Static Analyses. In: Companion Proceedings for the ISSTA/ECOOP 2018 Workshopss. ACM, pp. 111–116, 2018.

[Fi08] Finifter, Matthew; Mettler, Adrian; Sastry, Naveen; Wagner, David: Verifiable functional purity in Java. In: Proceedings of the 15th ACM conference on Computer and communications security. ACM, pp. 161–174, 2008.

[HM12] Huang, Wei; Milanova, Ana: ReImInfer: Method purity inference for Java. In: Proceedings of the ACM SIGSOFT 20th International Symposium on the Foundations of Software Engineering. ACM, p. 38, 2012.

[SCOD16] Stewart, Arran; Cardell-Oliver, Rachel; Davies, Rowan: Fine-grained classification of side-effect free methods in real-world Java code and applications to software security. In: Proceedings of the Australasian Computer Science Week Multiconference. ACM, p. 37, 2016.

S. Becker, I. Bogicevic, G. Herzwurm, S. Wagner (Hrsg.): SE/SWM 2019,
Lecture Notes in Informatics (LNI), Gesellschaft für Informatik, Bonn 2019 53

Inferring Visual Contracts from Java Programs

Abdullah Alshanqiti[1], Reiko Heckel[2], Timo Kehrer[3]

Abstract: In this work, we report about recent research results on "Inferring Visual Contracts from Java Programs", published in [1]. In this paper, we propose a dynamic approach to reverse engineering visual contracts from Java programs based on tracing the execution of Java operations. The resulting contracts give an accurate description of the observed object transformations, their effects and preconditions in terms of object structures, parameter and attribute values, and their generalised specification by universally quantified (multi) objects, patterns, and invariants. We explore potential uses in our evaluation, including in program understanding and testing, and we report on experimental results w.r.t. completeness (recall) and correctness (precision) of extracted contracts.

Keywords: Visual contracts, graph transformation, model extraction, dynamic analysis, reverse engineering, specification mining

Summary

Visual Contracts provide a precise high-level specification of the object graph transformations caused by invocations of operations on a component or service. They link static models (e.g., class diagrams describing object structures) and behavioural models (e.g., state machines specifying the order operations are invoked in) by capturing the preconditions and effects of operations on a system's objects.

Visual contracts differ from contracts embedded with code, such as JML in Java or Contracts in Eiffel, as well as from model-level contracts in OCL. They are *visual*, using UML notation to model complex patterns and transformations intuitively and concisely; *abstract*, providing a specification of object transformations at a high level of granularity to aid readability and scalability; *deep*, capturing the transformation of internal object structures besides input / output behaviour; and *executable*, based on graph transformation they support model-based oracle and test case generation, run-time monitoring, service specification and matching, state space analysis and verification. However, the detailed specification of internal data states and transformations, referred to as deep behavioural modelling, is an error-prone activity.

In this paper, we report on a line of research whose key idea is a dynamic approach to reverse engineering visual contracts from sequential Java programs based on tracing the execution of Java operations. The resulting contracts give accurate descriptions of the observed object

[1] Department of Computer Sciences, University of Leicester. a.m.alshanqiti@gmail.com
[2] Department of Computer Sciences, University of Leicester. reiko@mcs.le.ac.uk
[3] Institut für Informatik, Humboldt-Universität zu Berlin. timo.kehrer@informatik.hu-berlin.de

doi:10.18420/se2019-11

transformations, their effects and preconditions in terms of object structures, parameter and attribute values, and allow generalisation by *multi objects and patterns* and general invariants. The restriction to sequential Java is due to the need to associate each access to a unique operation invocation.

Given a Java application, the process starts by selecting the classes and operations within the scope of extraction and providing test cases for the relevant operations. We proceed by (A) observing the behaviour under these tests using AspectJ instrumentation and synthesising rule instances as pre/post snapshot graphs of individual invocations; (B) combining the instances into higher-level rules by abstracting from non-essential context; (C) generalising further by introducing multi objects and patterns; (D) deriving logical constraints and assignments over attribute and parameter values; and (E) identifying universally shared conditions and structures as invariants captured separately.

First solutions to variants of (A) and (B) were developed in our earlier work and have been summarized, consolidated and extended in [2, 3] which also presents a prototype tool and explores its potential uses in program understanding, testing and debugging. In this paper, we raise further the level of abstraction by supporting, in addition to previous work, the inference of multi patterns in (C), attribute assignments in (D) and universal context in (E), and exploiting subtyping throughout the process. We support the use of visual contracts as *deep oracles* by exporting contracts to the model transformation tool Henshin [4] and controlling the (otherwise non-deterministic) model execution by comparing the effect of each test invocation with the rules in the operation's contracts. This technology is used in a new evaluation of completeness (recall) and correctness (precision) of extracted contracts. Finally, we report on improved tool support, and we discuss two further application scenarios for our approach and tool in the field of model-based software engineering, namely the use of visual contracts for model-based (or visual) debugging as well as the automated learning of complex model editing operations by examples [5].

References

[1] Abdullah Alshanqiti, Reiko Heckel, and Timo Kehrer. "Inferring visual contracts from Java programs". In: *Automated Software Engineering* (2018), pp. 1–40.

[2] Abdullah Alshanqiti and Reiko Heckel. "Extracting Visual Contracts from Java Programs". In: *ASE'15*. IEEE. 2015, pp. 104–114.

[3] Abdullah Alshanqiti, Reiko Heckel, and Timo Kehrer. "Visual contract extractor: a tool for reverse engineering visual contracts using dynamic analysis". In: *ASE'16*. ACM. 2016, pp. 816–821.

[4] Daniel Strüber et al. "Henshin: A usability-focused framework for emf model transformation development". In: *ICGT'17*. Springer. 2017, pp. 196–208.

[5] Timo Kehrer, Abdullah Alshanqiti, and Reiko Heckel. "Automatic inference of rule-based specifications of complex in-place model transformations". In: *ICMT'17*. Springer. 2017, pp. 92–107.

Risk Mitigation Strategies in High Automation

Mario Gleirscher[1]

Abstract: The work underlying this presentation is titled "From Hazard Analysis to Hazard Mitigation Planning: The Automated Driving Case," accepted as a peer-reviewed full technical paper at the "NASA Formal Methods Symposium (NFM 2017)," published in April 2017.[2]

Keywords: high automation; autonomous systems; risk analysis; hazard mitigation; safe state; controller design; modelling; planning

Presentation Abstract

Autonomous machines—mobile, stationary, individual, collective, learning—need mechanisms to anticipate, predict, and mitigate dangerous situations. These mechanisms are required to be highly resilient, secure, and dependable. Increasing autonomy shifts risk ownership towards vendors, hence, raising the demand for such mechanisms to be valid and verified for a large variety of complex operational situations.

However, one of the main problems is that there is only little research on cross-disciplinary and compositional models supporting engineers with the transition from system-level hazard analysis and risk assessment to the design of controllers with risk mitigation strategies, and with the systematic refinement of abstractions of the controlled process towards the details relevant for controller design.

This presentation discusses an analysis and design method for high-level controllers for highly automated or autonomous machines with run-time mechanisms for risk monitoring and mitigation. This method aims at equipping engineers with a way to systematically examine a large number of risk scenarios and encode these scenarios in an action model. This model can then be given an operational semantics to make it suitable for optimal mitigation planning. In support of the planning layer of a machine or a system of machines, from this model it is possible to derive run-time risk monitors as well as controllers for effective handling of such scenarios at run-time.

[1] University of York, Computer Science, Deramore Lane, Heslington, York YO10 5GH, United Kingdom
mario.gleirscher@york.ac.uk

[2] See https://link.springer.com/chapter/10.1007/978-3-319-57288-8_23.

doi:10.18420/se2019-12

Towards these aims, the talk will discuss

(1) a framework for the analysis and specification of high-level controllers capable of run-time risk monitoring and mitigation,

(2) an incremental algorithm for the construction of high-level controller models from incremental inputs of risk analyses, and

(3) possible uses of this method and this algorithm in road vehicle and mobile robotics applications.

The model used in this framework is based on two main concepts, the controlled process and risk: The process model describes the action space of the considered process and the risk model contains information about which of these actions are associated with risk in certain scenarios, and which of these actions might be performed to mitigate such risk in these scenarios.

One of the main tasks of the (safety, systems, or requirements) engineer is to identify and model the process actions, the endangering actions, the mitigation actions, and the foreseeable causal relationships between these actions in each of the considered scenarios.

The method provides guidance for step-wise analysis of endangering actions, the identification of mitigation actions, and, based on these two steps, the modelling of risk mitigation strategies in high automation.

Furthermore, the method supports the composition of mitigation strategies as well as the composition of scenario-specific models. Moreover, the provided algorithm then calculates an operational semantics of the composed model. These semantics can then be used for various checking purposes, for example, consistency and causality checking.

The talk provides a snapshot of the current stage of research of the proposed method and explains the framework using typical examples of risk mitigation in road vehicles. It provides an outlook to the next steps. Further material for this presentation can be found in [Gl17] and [GK17].

References

[GK17] Gleirscher, Mario; Kugele, Stefan: From Hazard Analysis to Hazard Mitigation Planning: The Automated Driving Case. In (et al., C. Barrett, ed.): NASA Formal Methods (NFM) – 9th Int. Symp., Proceedings. volume 10227 of LNCS. Springer, Berlin/New York, 2017. https://doi.org/10.1007/978-3-319-57288-8_23.

[Gl17] Gleirscher, Mario: Run-Time Risk Mitigation in Automated Vehicles: A Model for Studying Preparatory Steps. In (Bulwahn, L.; Kamali, M.; Linker, S., eds): First iFM Workshop on Formal Verification of Autonomous Vehicles 2017 (FVAV 2017). EPTCS, 2017. https://doi.org/10.4204/eptcs.257.8.

Supporting Semi-Automatic Co-Evolution of Architecture and Fault Tree Models

Sinem Getir[1], André van Hoorn[2], Timo Kehrer[3], Yannic Noller[4], Matthias Tichy[5]

Abstract: In this work, we report about recent research results on "Supporting Semi-Automatic Co-Evolution of Architecture and Fault Tree Models", published in [Ge18]. During the whole life-cycle of software-intensive systems in safety-critical domains, system models must consistently co-evolve with quality evaluation models. However, performing the necessary synchronization steps is a cumbersome and often manual task prone to errors. To understand this problem in detail, we have analyzed the evolution of two representatives of system models and quality evaluation models, namely architecture and fault tree models, for a set of evolution scenarios of a factory automation system called Pick and Place Unit. We designed a set of intra- and inter-model transformation rules which fully cover the evolution scenarios of the case study and which offer the potential to semi-automate the co-evolution process. In particular, we validated these rules with respect to completeness and evaluated them by a comparison to typical visual editor operations. Our results show a significant reduction of the amount of required user interactions in order to realize the co-evolution.

Keywords: System architecture, fault trees, safety, model co-evolution, model transformation

Summary

A rigorous quality evaluation is among the key methods for the dependable engineering of software-intensive systems in safety-critical domains. To that end, model-based approaches have been proposed which use quality evaluation and system models to gain knowledge about the quality of a system. In model-based quality evaluation, the consistency of the involved models is of utmost importance. For example, the failures of an architectural component must be adequately considered in an associated fault tree model. While this consistency requirement can be reasonably met for a particular snapshot of a system, quality evaluation models typically become outdated when the system evolves, i.e., quality evaluation models and system models evolve in an inconsistent way. As a consequence, quality evaluation leads to highly improper results. Hence, loosely inter-related models such as architectural models and quality evaluation models like fault trees should consistently evolve in parallel, a phenomenon to which we refer to as (consistent) *model co-evolution*.

Since loosely inter-related models are typically changed in isolation of each other, one adequate approach to support developers is *model synchronization*, i.e., the task of adapting a model in response to changes in one of its inter-related counterparts in order to achieve consistent co-evolution. In general, however, achieving this kind of model co-evolution cannot

[1] Humboldt-Universität zu Berlin. getir@informatik.hu-berlin.de
[2] University of Stuttgart. van.hoorn@informatik.uni-stuttgart.de
[3] Humboldt-Universität zu Berlin. timo.kehrer@informatik.hu-berlin.de
[4] Humboldt-Universität zu Berlin. noller@informatik.hu-berlin.de
[5] University of Ulm. matthias.tichy@uni-ulm.de

doi:10.18420/se2019-13

be fully automated as usually assumed by existing approaches to model synchronization. At best, developers may be supported by *recommending* possible synchronization actions, as, e.g., in the model-based (co-)evolution framework known as CoWolf [Ge15]. To achieve consistent co-evolution, CoWolf follows a rule-based approach where incremental model transformations are used to recommend both intra- and inter-model change actions. However, since the adequacy of the recommendations strongly depends on the transformation rules being used by the tool, the evolution problem is shifted to the engineering of proper transformation rules.

We tackle this problem of engineering proper transformation rule sets for an important class of models in the context of model-based hazard analysis, namely system architecture models and fault tree models. We extend our previous work [Ge13] on the evolution of the so-called Pick and Place Unit (PPU) [LFV13], a case study from the automation engineering domain. To study co-evolution in terms of the PPU, we created consistent software architecture and fault tree models for all safety-relevant evolution scenarios. Thereupon, we conducted a thorough quantitative analysis of the evolution scenarios with respect to the co-evolution of the models, i.e., how changes in one model affect changes in the other model. We show that the models do not co-evolve in a systematic, automatable way and instead expertise of the developer is required to achieve co-evolution. Moreover, we developed a set of model transformation rules for 1) the independent evolution of software architecture and fault tree models and 2) the synchronization of one model based on changes in another model ensuring a correct co-evolution of both models. We show that the presented set of model transformations is *complete*, i.e., it supports performing all co-evolutions of the case study scenarios, and improves the *task efficiency* by reducing the amount of required model transformation applications to realize the co-evolution by, on average, 52% compared to visual editing operations and 85% compared to atomic model changes. Finally, we implemented these rules in the tool CoWolf to enable the co-evolution of fault trees and software architecture models.

References

[Ge13] Getir, S.; Van Hoorn, A.; Grunske, L.; Tichy, M.: Co-Evolution of Software Architecture and Fault Tree models: An Explorative Case Study on a Pick and Place Factory Automation System. In: NiM-ALP @ MoDELS'13. Pp. 32–40, 2013.

[Ge15] Getir, S.; Grunske, L.; Bernasko, C. K.; Käfer, V.; Sanwald, T.; Tichy, M.: CoWolf–A generic framework for multi-view co-evolution and evaluation of models. In: ICMT'15. Springer, pp. 34–40, 2015.

[Ge18] Getir, S.; Grunske, L.; van Hoorn, A.; Kehrer, T.; Noller, Y.; Tichy, M.: Supporting semi-automatic co-evolution of architecture and fault tree models. Journal of Systems and Software 142/, pp. 115–135, 2018.

[LFV13] Legat, C.; Folmer, J.; Vogel-Heuser, B.: Evolution in Industrial Plant Automation: A case study. In: IECON'13. IEEE, pp. 4386–4391, 2013.

Session 4: Traceability, Performanz und Continuous SE

Software Traceability in the Automotive Domain: Challenges and Solutions

Jan-Philipp Steghöfer, Salome Maro, and Miroslaw Staron[1]

Abstract: In the automotive domain, the development of all safety-critical systems has to comply with safety standards such as ISO 26262. These standards require established traceability, the ability to relate artifacts created during development of a system, to ensure resulting systems are well-tested and therefore safe. Our study [MSS18] contrasts general traceability challenges and solutions with those specific to the automotive domain, and investigates how they manifest in practice.
We combine a tertiary literature review to identify general challenges and solutions, a case study with an automotive supplier as validation for how challenges and solutions are experienced in practice, and a multi-vocal literature review to identify challenges and solutions specific to the automotive domain. We found 22 challenges and 16 unique solutions in the reviews. 17 challenges were identified in the case study; six remain unsolved. We discuss challenges and solutions from the perspectives of academia, tool vendors, consultants and users, and identify differences between scientific and "grey" literature. We discuss why challenges remain unsolved and propose solutions.
Our findings indicate that there is a significant overlap between general traceability challenges and those in the automotive domain but that they are experienced differently.

Keywords: Traceability, Automotive Software Engineering, Safety, ISO 26262, Automotive SPICE

In the automotive industry, traceability – the ability to relate artifacts created during the development of a system – is a necessity since safety standards require proof that safety requirements were specified, taken into account during the design and development, validated in test cases, and verified through safety analysis. In order to realize the benefits of traceability (and successfully argue their safety cases), software development companies need to establish a traceability strategy that is aligned with their goals. Defining and implementing a traceability strategy is not a trivial task, since it requires a good understanding of the artifacts to be traced as well as the ability to define meaningful links and to make sure the created links are useful.

The contribution of this paper is therefore to provide an exhaustive empirical evaluation of traceability challenges and solutions in the automotive domain that takes the specific characteristics of automotive software development into account. To achieve this, we conducted a tertiary literature review, a case study, and a multi-vocal literature review. This allows us to explore the traceability problem in the automotive domain from both the practical and the scientific perspective and provides insight into the challenges of traceability

[1] Chalmers | University of Gothenburg, Gothenburg, Sweden jan-philipp.steghofer@gu.se, salome.maro@gu.se, miroslaw.staron@gu.se

doi:10.18420/se2019-14

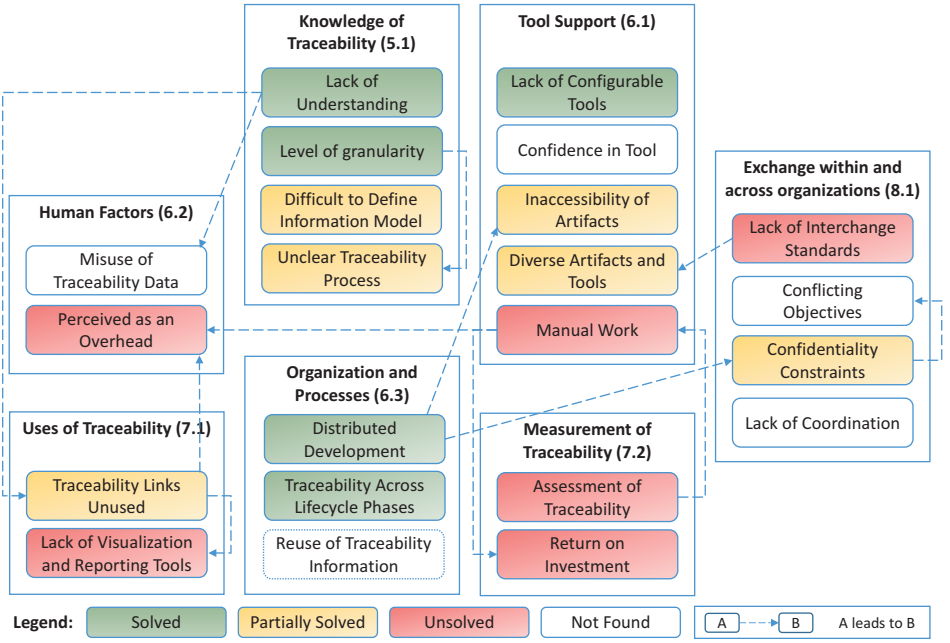


Fig. 1: Summary of Traceability Challenges. Challenges solved at the case company have a green background, partially solved challenges have a yellow background and unsolved challenges have a red background. Challenges that have no background color were only in the literature and not identified in the case study. Directed arrows indicate that one challenge leads to the presence of another one.

as they present themselves in practice as well as solution approaches proposed by academia, tool vendors, consultants, and the users of traceability themselves.

Our study shows that there is a significant overlap between general traceability challenges and solutions and those found in the automotive domain (Figure 1 provides an overview). It provides evidence that many solutions proposed in the literature are not applicable in the automotive domain due to its specific set of characteristics, such as system complexity, the safety-criticality of the developed systems, and the distributed development split between the OEMs and suppliers. While the tertiary review revealed challenges and solutions mostly from academia, the MLR was a richer data source that gave insight into practical problems. The case study validated our findings as most of the challenges were found there as well.

References

[MSS18] Maro, Salome; Steghöfer, Jan-Philipp; Staron, Miroslaw: Software Traceability in the Automotive Domain: Challenges and Solutions. In: Journal of Systems and Software Volume 141, July 2018, pp 85–110. https://doi.org/10.1016/j.jss.2018.03.060

S. Becker, I. Bogicevic, G. Herzwurm, S. Wagner (Hrsg.): SE/SWM 2019,
Lecture Notes in Informatics (LNI), Gesellschaft für Informatik, Bonn 2019 63

Traceability in the Wild: Automatically Augmenting Incomplete Trace Links

Michael Rath[1], Jacob Rendall[2], Jin L.C. Guo[3], Jane Cleland-Huang[2], Patrick Mäder[1]

Abstract: This paper was published at the 40th International Conference on Software Engineering (ICSE) in May 2018. The authors propose a novel approach to establish trace links among software development artifacts. In particular, it allows to automatically link commits made in a projects' version control systems (such as git) to respective issues in the projects' issue tracker (e. g. Atlassian Jira). Besides augmenting an existing code base with additional trace links, the approach enables active recommendation of issues to the developer while performing a new commit to the version control system. This simplifies the overall development workflow. The proposed method is based on state-of-the-art machine learning techniques and serves as a basic building block in establishing project wide traceability. It's feasibility, completeness, and usefulness was successfully evaluated through six empirical studies as well as one human study. The work was honored with an ACM SIGSOFT Distinguished Paper Award.

Keywords: Traceability, Link Recovery, Machine Learning, Open Source

[1] Technical University Ilmenau, Ilmenau, Germany {michael.rath,patrick.maeder}@tu-ilmenau.de
[2] University of Notre Dame, South Bend, USA {rendal1,JaneClelandHuang}@nd.edu
[3] McGill University, Montreal, Canada jguo@cs.mcgill.ca

doi:10.18420/se2019-15

Badger: Complexity Analysis with Fuzzing and Symbolic Execution

Yannic Noller[1], Rody Kersten[2], Corina S. Păsăreanu[3]

Abstract: In this work, we report on our recent research results on "Badger: Complexity Analysis with Fuzzing and Symbolic Execution" which was published in the proceedings of the 27th ACM SIGSOFT International Symposium on Software Testing and Analysis [NKP18]. Badger employs a hybrid software analysis technique that combines fuzzing and symbolic execution for finding performance bottlenecks in software. Our primary goal is to use Badger to discover vulnerabilities which are related to worst-case time or space complexity of an application. To this end, we use a cost-guided fuzzing approach, which produces inputs to increase the code coverage, but also to maximize a resource-related cost function, such as execution time or memory usage. We combine this fuzzing technique with a customized symbolic execution, which is also guided by heuristics that aim to increase the same cost. Experimental evaluation shows that this hybrid approach enables us to use the strengths of both techniques and overcome their individual weaknesses.

Keywords: Software Testing; Complexity Analysis; Fuzzing; Symbolic Execution

Summary

We explore here the application of fuzzing and symbolic execution to algorithmic complexity analysis, which can be applied to reason about programs, understand performance bottlenecks and find opportunities for compiler optimizations. Algorithmic complexity analysis can also reveal worst-case complexity vulnerabilities, which occur when the worst-case time or space complexity of an application is significantly higher than the average case. In such situations, an attacker can mount Denial-of-Service attacks by providing inputs that trigger the worst-case behavior.

Fuzzing is currently the state-of-the-art testing technique to discover security-related vulnerabilities in software. Companies like Microsoft [Mi16] and Google [OS17] are applying fuzzing techniques on a regular basis. Although fuzzing applies random mutation operations to produce inputs, and hence, a large fraction of them might be invalid, it still can be more effective in practice than more sophisticated testing techniques, such as symbolic execution based approaches, due to its low computation overhead. Most fuzzers are designed to generate inputs that increase code coverage, while for complexity analysis one is interested in generating inputs that trigger worst case execution behavior of the programs. Furthermore,

[1] Humboldt-Universität zu Berlin, Germany, yannic.noller@hu-berlin.de
[2] Synopsys, Inc., San Francisco, USA, rody@synopsys.com
[3] Carnegie Mellon University Silicon Valley, NASA Ames Research Center, Moffett Field, USA, corina.s.pasareanu@nasa.gov

doi:10.18420/se2019-16

fuzzers are known to be good at finding so called *shallow* bugs but they may fail to execute deep program paths [St16], i.e. paths that are guarded by specific conditions in the code. On the other hand, symbolic execution techniques are particularly well suited to find such cases, but usually are much more expensive in terms of computational resources required.

We therefore present BADGER: a framework that combines fuzzing and symbolic execution for automatically finding algorithmic complexity vulnerabilities in Java applications. It extends the KELINCI fuzzer with a worst-case analysis (aka KELINCIWCA) by adding a new heuristic to account for resource-usage *costs* of program executions. It further uses a modified version of SYMBOLIC PATHFINDER (SPF) to import inputs from the fuzzer, analyze them and generate new inputs that increase both coverage and execution cost. We modified SPF by adding a mixed concrete-symbolic execution mode, similar to concolic execution which allows us to *import* the inputs generated on the fuzzing side and quickly reconstruct the symbolic paths along the executions triggered by the concrete inputs. These symbolic paths are then organized in a *tree*, which is analyzed with the goal of generating new inputs that expand the tree. The analysis is guided by *heuristics* on the SPF side that favor new branches that increase resource-costs. The newly generated inputs are passed back to the fuzzing side. BADGER can use various cost models, such as number of conditions executed, actual execution time as well as user-defined costs that allow us to keep track of memory and disk usage as well as other resources of interest particular to an application.

Our evaluation of BADGER demonstrates that its performance and quality benefits over fuzzing and symbolic execution by themselves. We plan to explore more heuristics for worst-case analysis on both the fuzzing and symbolic execution side. Furthermore, we plan to extend our approach not only for complexity analysis, but also for a differential side-channel analysis of security-relevant applications.

References

[Mi16] Microsoft Security Risk Detection. `https://www.microsoft.com/en-us/security-risk-detection/`, 2016. Accessed: Dec 14, 2018.

[NKP18] Noller, Yannic; Kersten, Rody; Păsăreanu, Corina S.: Badger: Complexity Analysis with Fuzzing and Symbolic Execution. In: Proceedings of the 27th ACM SIGSOFT International Symposium on Software Testing and Analysis. ISSTA 2018, ACM, New York, NY, USA, pp. 322–332, 2018.

[OS17] OSS-Fuzz: Five months later, and rewarding projects. `https://testing.googleblog.com/2017/05/oss-fuzz-five-months-later-and.html`, 2017. Accessed: Dec 14, 2018.

[St16] Stephens, Nick; Grosen, John; Salls, Christopher; Dutcher, Andrew; Wang, Ruoyu; Corbetta, Jacopo; Shoshitaishvili, Yan; Kruegel, Christopher; Vigna, Giovanni: Driller: Augmenting Fuzzing Through Selective Symbolic Execution. In: 23nd Annual Network and Distributed System Security Symposium, NDSS 2016, San Diego, California, USA. 2016.

S. Becker, I. Bogicevic, G. Herzwurm, S. Wagner (Hrsg.): SE/SWM 2019,
Lecture Notes in Informatics (LNI), Gesellschaft für Informatik, Bonn 2019 67

The Eye of Continuous Software Engineering

Jan Ole Johanssen[1] Anja Kleebaum[2] Barbara Paech[3] Bernd Bruegge[4]

Abstract: We summarize the paper *Practitioners' Eye on Continuous Software Engineering: An Interview Study* [Jo18], which was presented at the 2018 edition of the *International Conference on Software and System Processes* (ICSSP) in Gothenburg, Sweden.

Keywords: Continuous Delivery; Continuous Integration; Continuous Software Engineering; Experience Report; Interview Study

1 Overview

Practitioners increasingly apply Continuous Software Engineering (CSE), i. e., combining activities such as continuous integration and deployment, to enable rapid software development [Bo14]. However, practitioners have different perceptions of CSE, which impedes its adoption in industry. Challenges in introducing CSE to real-world environments and improving activities as well as the lack of comparisons with other companies to identify risks and benefits hinder the adoption of CSE. We aim to support practitioners in establishing, assessing, and advancing CSE by providing an overview of current practices in industry.

We derived a list of CSE characteristics that distinguishes between *CSE categories* and *CSE elements*. This list served as the foundation for a semi-structured interview and was refined based on practitioners' answers: We completed the study with nine CSE categories: user, developer, business, development, operation, code, quality, software management, and knowledge. Each category encompasses two to four CSE elements; 33 elements in total.

We conducted the interview study with 24 practitioners from 17 companies. The practitioners were composed of developers, technical leaders, CSE specialists, project managers, and an executive director. All of them are primarily using, defining, or planning CSE. As outlined in Section 2, our research questions focused on practitioners' definition of CSE, elements perceived as most relevant, experiences, and future plans for CSE. We used a questionnaire and a guideline to run the interviews with the practitioners and audio-recorded their answers. We applied a two-step approach for the analysis of the transcripts. First, we allocated answers to research questions. Second, we coded text extracts with the CSE elements.

[1] Technical University of Munich, Munich, Germany, jan.johanssen@tum.de
[2] Heidelberg University, Heidelberg, Germany, kleebaum@informatik.uni-heidelberg.de
[3] Heidelberg University, Heidelberg, Germany, paech@informatik.uni-heidelberg.de
[4] Technical University of Munich, Munich, Germany, bruegge@in.tum.de

doi:10.18420/se2019-17

2 Results

How do practitioners define CSE? Practitioners primarily use CSE elements from the software management category, i. e., continuous integration of work, agile practices, and continuous deployment of releases, to define CSE. Overall, *tool*, *methodology*, *developer*, *life cycle*, and *product management* perspectives formed practitioners' definitions of CSE.

Which CSE elements are perceived as most relevant by practitioners? CSE elements from three categories were repeatedly mentioned by practitioners when being asked for relevance: quality, i. e., *automated tests*, user, i. e., *involved users and other stakeholders*, and developer, i. e., to *comply with a shared ruleset*. Notably, developers consider CSE elements from the *code* category as the base for CSE. The results for this research question can be summarized under *user commitment*, *team commitment*, and *automated loop*.

What are practitioners' experiences with CSE? We recorded 19 positive, 56 neutral, and 17 negative experiences with CSE elements. Categories in which positive experiences prevail indicate CSE elements that might serve as an entry point to CSE. Few positive mentions might signalize the low maturity of the respective CSE elements. In case practitioners are currently evaluating a CSE element, they might tend to provide neutral responses given their work-in-progress state. Multiple negative experiences can reveal CSE elements that are challenging to implement for practitioners. We provide experience reports for the CSE categories *developer*, *operation*, *software management*, *user*, and *quality*.

What are practitioners' future plans for CSE? Practitioners' reports are vague and mostly span over multiple CSE categories. They mention 19 CSE elements either only once, twice, or three times. With seven mentions, *automated tests* stood out from the other CSE elements. Practitioners' main strategies focus on *enhancement*, *expansion*, and *on-demand adaption*.

From the practitioners' answers, we conclude that some elements of CSE remain difficult to implement. Therefore, we created the *Eye of CSE* model. As depicted in [Jo18], the pupil of the eye is focused on CSE, while the CSE categories form the iris and thereby determine the perception of CSE. The CSE elements are arranged throughout the *sclera*, the white of the eye, and are loosely coupled with one category; their proximity to other elements indicates similarities and relationships to them. This structure helps to identify next steps for CSE.

Acknowledgments This work was supported by the DFG (German Research Foundation) under the Priority Programme SPP1593: Design For Future – Managed Software Evolution (CURES project).

References

[Bo14] Bosch, Jan: Continuous Software Engineering: An Introduction. Springer, 2014.

[Jo18] Johanssen, Jan Ole; Kleebaum, Anja; Paech, Barbara; Bruegge, Bernd: Practitioners' Eye on Continuous Software Engineering: An Interview Study. In: Proceedings of the 2018 Int. Conference on Software and System Process. ICSSP '18. ACM, pp. 41–50, 2018.

Session 5: Open and Inner Source

S. Becker, I. Bogicevic, G. Herzwurm, S. Wagner (Hrsg.): SE/SWM 2019,
Lecture Notes in Informatics (LNI), Gesellschaft für Informatik, Bonn 2019 71

Inner Source Definition, Benefits, and Challenges

Maximilian Capraro[1], Dirk Riehle[2]

Abstract: Inner source (IS) is the use of open source software development practices within an organization. The organization still develops proprietary software but internally opens up its development. The research area lacks a systematic assessment of known research work: No model exists that defines IS thoroughly. Various case studies provide insights into IS programs in a specific context but only few publications apply a broader perspective. To resolve this, we performed an extensive literature survey and analyzed 43 IS related publications. Using qualitative data analysis methods, we developed a model of elements that constitute IS. We present a classification framework for IS and apply it to lay out a map of known IS endeavors. Further, we present qualitative models summarizing the benefits and challenges of IS. This article is an extended abstract of [CR17].

Keywords: inner source; taxonomy; open collaboration; internal open source; software development methods; software engineering; software development efficiency; software development productivity

1 Contributions and Approach

The article presents an extensive literature survey. In detail, its contributions are:

- A theoretical model of elements that constitute inner source (IS).
- A classification framework for IS programs and projects.
- Qualitative models summarizing reported IS benefits and adoption challenges.

We followed a two phase research approach:

First, in the selection phase, we identified relevant IS literature using selected online databases. We searched with variety of phrases (including inner source, internal open source, ...) and read abstracts or full publications to decide on inclusion. This resulted in 43 publications.

Second, in the analysis phase, we analyzed and systematically arranged the literature. Using the inductive theory generation method [Th06], we identified key elements constituting IS as well as benefits and challenges of IS adoption.

[1] Computer Science Department, Friedrich-Alexander-Universität, Martensstraße 3, 91058 Erlangen, Germany
maximilian.capraro@fau.de
[2] Computer Science Department, Friedrich-Alexander-Universität, Martensstraße 3, 91058 Erlangen, Germany
dirk.riehle@fau.de

doi:10.18420/se2019-18

During analysis, we found contradicting elements of IS. For example, one organization internally opened all their source code, while another selected specific components. We used such contradicting observations and preexisting IS classifications to develop a classification framework of IS.

2 Results

Theoretical Model of IS Elements We found four key elements that constitute IS. An *(1) open environment* is created by opening up development artifacts, inviting external contributors, and establishing open communication. *(2) Shared cultural values* are internalized by individuals within the organization. Empowered by the open environment and shared cultural values, *(3) communities around software* form. *(4) Inner source development practices* are exercised by a project-specific or program-wide community.

Classification Framework for IS We found that IS programs differ on at least three dimensions; IS projects on at least two dimensions. For each of these dimensions, our framework lays out the classes we observed in literature. Figure 1 gives an overview of the framework.

Fig. 1: Classification framework for IS

Dimensions of IS **Programs**			Dimensions of IS **Projects**	
Prevalence	Degree of Self-Organization	Internal Economics	Governance	Objective
Universal	Free Task Choice & Free Component Choice	Local-Library	Single Organizational Unit	Exploration-Oriented
Selective	Assigned Tasks & Free Component Choice	Private-Market	Multiple Organizational Units	Utility-Oriented
Project-Specific	Assigned Tasks & Assigned Components		All Organizational Units	Service-Oriented

Benefits and Challenges Literature reported seven benefits of IS adoption (including increased development efficiency, higher code quality, and quicker development cycles). However, the large majority of surveyed literature neither validated the observed IS benefits nor discussed their generalizability. Literature reported eight challenges of IS adoption (including resistance from employees and diversity of the involved organizational units).

References

[CR17] Capraro, Maximilian; Riehle, Dirk: Inner source definition, benefits, and challenges. ACM Computing Surveys (CSUR), 49(4):67, 2017.

[Th06] Thomas, David R: A general inductive approach for analyzing qualitative evaluation data. American journal of Evaluation, 27(2):237–246, June 2006.

A Simple NLP-based Approach to Support Onboarding and Retention in Open Source Communities

Christoph Stanik[1] Lloyd Montgomery[2] Daniel Martens [3] Davide Fucci[4] Walid Maalej[5]

Abstract: This work has been presented at the 34th International Conference on Software Maintenance and Evolution (ICSME) in Madrid, Spain [**?**][6].

Successful open source communities are constantly looking for new members and helping them become active developers. A common approach for developer onboarding in open source projects is to let newcomers focus on relevant yet easy-to-solve issues to familiarize themselves with the code and the community. The goal of this research is twofold. First, we aim at automatically identifying issues that newcomers can resolve by analyzing the history of resolved issues by simply using the title and description of issues. Second, we aim at automatically identifying issues, that can be resolved by newcomers who later become active developers. We mined the issue trackers of three large open source projects and extracted natural language features from the title and description of resolved issues. In a series of experiments, we optimized and compared the accuracy of four supervised classifiers to address our research goals. Random Forest, achieved up to 91% precision (F1-score 72%) towards the first goal while for the second goal, Decision Tree achieved a precision of 92% (F1-score 91%). A qualitative evaluation gave insights on what information in the issue description is helpful for newcomers. Our approach can be used to automatically identify, label, and recommend issues for newcomers in open source software projects based only on the text of the issues.

Keywords: open source software; onboarding; task selection; newcomers; machine learning; natural language processing

1 Research Design

When developers decide to start contributing to an Open Source (OSS) project they become newcomers to that project. In large OSS projects, newcomers face thousands of open and unresolved issues they have to choose from. The goal of this work is to identify issues in OSS projects that newcomers can resolve–tackling the barrier of finding a way to start.

[1] UHH, Informatik, Vogt-Kölln-Str. 30, 22527 Hamburg, Germany stanik@informatik.uni-hamburg.de
[2] UHH, Informatik, Vogt-Kölln-Str. 30, 22527 Hamburg, Germany montgomery@informatik.uni-hamburg.de
[3] UHH, Informatik, Vogt-Kölln-Str. 30, 22527 Hamburg, Germany martens@informatik.uni-hamburg.de
[4] UHH, Informatik, Vogt-Kölln-Str. 30, 22527 Hamburg, Germany fucci@informatik.uni-hamburg.de
[5] UHH, Informatik, Vogt-Kölln-Str. 30, 22527 Hamburg, Germany maalej@informatik.uni-hamburg.de
[6] Open access link: `https://arxiv.org/abs/1806.02592`

doi:10.18420/se2019-19

Dataset. Our dataset contains issues of three large OSS projects, namely Qt, Eclipse, and LibreOffice. In total, we collected 225,000 issues from 2001 to 2017. For our research, we consider only the title and the description of the issues.

Machine Learning Experiments. In a series of machine learning experiments[7] we used natural language processing (NLP) to classify issues into "can be resolved by a newcomer" or "cannot be resolved by a newcomer". As for NLP text features, we used tf-idf, the sentiment of the text, as well as the number of words in the issues. In the experiments, we compared the performance of four classifiers including hyper-parameter tuning using grid search.

Qualitative Investigation. To better understand how mentors–experienced OSS contributor that help newcomers finding solvable issues–and newcomers themselves select issues to work on, we conducted eight semi-structured interviews and performed thematic analysis on a random sample of issues.

2 Results

Our results show that Random Forest can achieve, on average, a precision of 79% in identifying issues newcomers can resolve. We optimized the classification benchmarks towards precision as we think that it is more important to be sure that a newcomer can resolve an issue than covering all issues–which would introduce a lot of noise (high recall), and therefore includes more issues newcomers cannot resolve.

The qualitative analysis provides us with insights on how we might be able to improve our approach further. The interviews revealed that issues that contain, e.g., *code snippets* or *steps to reproduce* are more helpful for newcomers as this information hints towards where and how to start working in the code-base. The thematic analysis resulted in similar findings but further shows that experienced contributors not only work on more complex issues but also on issues newcomers can resolve.

[7] replication package: `https://mast.informatik.uni-hamburg.de/replication-packages/`

Session 6: Konfiguration

Propagating Configuration Decisions with Modal Implication Graphs

Sebastian Krieter[1,2], Thomas Thüm[3], Sandro Schulze[1], Reimar Schröter[1], Gunter Saake[1]

Abstract: This work was originally published as "Propagating Configuration Decisions with Modal Implication Graphs" at the 40th International Conference on Software Engineering 2018 [Kr18].

Keywords: Feature Modeling, Configuration, Software Product Lines, Configurable Systems

Extended Abstract

Modern software systems often comprise thousands of interdependent configuration options, also known as features. Due to the large number of features and their arbitrarily complex interdependencies, configuring and analyzing these highly-configurable systems poses real challenges to developers and users. To properly configure such systems, all dependencies between features must be considered, which requires a non-trivial configuration process that can become tedious and error-prone. Hence, for highly-configurable systems tool support becomes necessary to help users in their decision-making process.

Usually, configuring a system is an iterative and interactive process, in which users decide for one feature at a time, whether it is selected or deselected in the current configuration. An effective mechanism for supporting users during this configuration process is *decision propagation*, which calculates the implications of a user's decision immediately. For instance, the selection of a feature can lead to the mandatory selection or deselection of other features. By applying decision propagation, users are directly informed about the consequences of their decisions. Furthermore, users are not able to introduce conflicting decisions to the configuration.

In our paper [Kr18], we propose a data structure, the *modal implication graph (MIG)*, to improve the performance of current concepts for decision propagation. MIGs represent the feature dependencies of a system and provide efficient access to them via graph traversal algorithms. MIGs are an extensions of regular implication graphs, which can be used to represent binary relations between features (e.g., requires and excludes). In a MIG, we call

[1] University of Magdeburg, Germany
[2] Harz University of Applied Sciences Wernigerode, Germany
[3] TU Braunschweig, Germany

doi:10.18420/se2019-20

these binary relations *strong* edges. We introduce a new type of connection, called *weak* edges that indicate a non-binary relationship between three or more features. We show an example of a MIG in Fig. 1 on the right. On the left, we depict a corresponding feature diagram, which is commonly used to express dependencies between features.

In the worst case, a decision propagation algorithm has to examine every non-configured feature and determine whether its selection or deselection is implied by the user's most recent decision. We use MIGs to reduce the number of features that have to be examined during this process. By traversing through the graph, we can divide all non-configured features into three distinct groups, features that are not, strongly-, or weakly-connected to the current user decision. This increases the overall performance, as only the weakly-connected features must be inspected further. In our evaluation with 120 real-world systems from different domains, we confirm a significant increase in efficiency, when using a MIG for decision propagation.

Though, originally, decision propagation was proposed for an interactive configuration process, it can be applied to other domains as well, which in turn makes our approach also applicable to these domains. For instance, decision propagation can be used in calculating t-wise configuration samples for testing or benchmarking purposes [Al16]. Furthermore, MIGs can support other processes in addition to decision propagation. Due to the efficient access to dependencies between certain features, it can speed up other important analyses such as finding atomic sets, false-optional features, and redundant constraints.

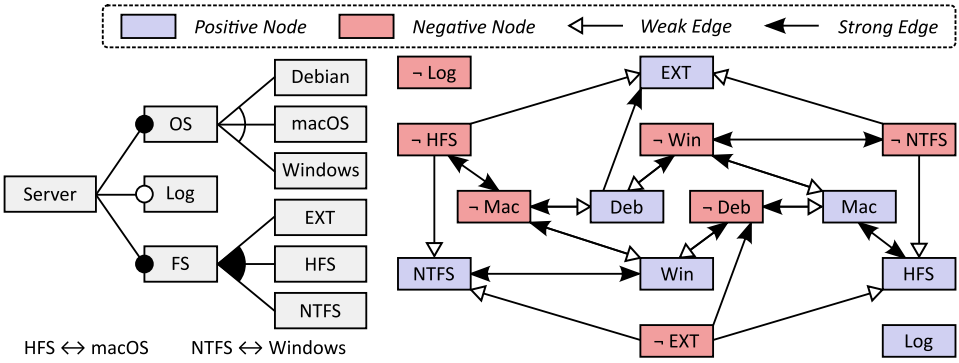


Fig. 1: Feature model (left) and its representation as modal implication graph (right).

References

[Al16] Al-Hajjaji, Mustafa; Krieter, Sebastian; Thüm, Thomas; Lochau, Malte; Saake, Gunter: IncLing: Efficient Product-Line Testing Using Incremental Pairwise Sampling. In: ACM SIGPLAN Notices. volume 52. ACM, pp. 144–155, 2016.

[Kr18] Krieter, Sebastian; Thüm, Thomas; Schulze, Sandro; Schröter, Reimar; Saake, Gunter: Propagating Configuration Decisions with Modal Implication Graphs. In: Proceedings of the 40th International Conference on Software Engineering. ICSE '18. ACM, pp. 898–909, 2018.

Tackling combinatorial explosion: a study of industrial needs and practices for analyzing highly configurable systems

Mukelabai Mukelabai[1], Damir Nešić[2], Salome Maro[1],
Thorsten Berger[1], and Jan-Philipp Steghöfer[1]

Abstract: Hundreds of dedicated analysis techniques for highly configurable systems have been conceived, many of them able to analyze properties for all possible system configurations. Unfortunately, it is largely unknown whether these techniques are adopted in practice, whether they address actual needs, or which strategies practitioners apply. We present a study [MNM+18] of analysis practices and needs in industry based on surveys and interviews. We confirm that properties considered in the literature (e.g., reliability) are relevant and that consistency between variability models and artifacts is critical, but that the majority of analyses for specifications of configuration options (a.k.a., variability model analysis) is not perceived as needed. We identified pragmatic analysis strategies, including practices to avoid the need for analysis. We discuss analyses that are missing and synthesize our insights into suggestions for future research.

Keywords: Highly configurable systems, software product lines, analysis

Engineering highly configurable systems such as software product lines is challenging due to variability—the number of configurations and system variants grows exponentially with the number of configuration options. Therefore, analysis techniques to validate *general properties* such as reliability or performance, *code properties* such as scattering or nesting, *variability model properties* such as absence of dead options, and *consistency properties* such as absence of dead code are necessary. Due to the need to analyse all possible variants, analysis techniques for highly configurable systems differ from traditional analyses. However, we lack empirical data whether the proposed analyses for highly configurable systems are adopted in practice, whether they address actual needs or find errors, or what analysis strategies are actually applied.

We present the results of a study which relied on a survey with 27 practitioners engineering highly configurable systems and follow-up interviews with 15 of them, covering 18 different companies from eight countries. Our study design relied on categorizing analysis techniques from the literature and identifying properties analyzed by them; we used these to elicit the need for and the severity of analyzing the properties. We also elicited industrial practices. Since it is intrinsically difficult to objectively understand the real practices and map them to the state of research, we triangulated results from the survey and interviews.

[1] Chalmers | Unviversity of Gothenburg, Sweden. mukelabai.mukelabai@gu.se, salome.maro@gu.se, thorsten.berger@gu.se, jan-philipp.steghofer@gu.se
[2] KTH, Stockholm, Sweden. damirn@kth.se

doi:10.18420/se2019-21

Our results show that common properties suggested in the literature are indeed relevant for highly configurable systems. However, most of the variability-model-related analysis properties are not seen as important by our practitioners. The proposed change-impact analyses are not seen as sufficient, because they are confined to the model and its configuration space, not providing holistic insights on impacts on implementation artifacts. Assuring consistencies between artifacts (especially variability model and source code) is considered highly critical, as well as identifying unwanted feature interactions.

We observed (as expected) testing as the dominant practice. Interestingly, the configuration sampling criteria that are necessary for testing primarily rely on experience. Hardly any systematic sampling or random sampling is used. Our results also suggest that the latter are not even applicable given the configuration spaces that would still leave too many irrelevant variants. Furthermore, hardly any formal method is used (apart from limited model checking). Besides testing, manual work, such as code reviews, is exercised, because often the variability models required for more sophisticated analyses do not exist or are not expressed in a form that can be used as an input. The lack of integrated tool chains is also a factor, since artifacts required for performing analyses are managed in different tools. Interestingly, the experience of the developers and rules, such as coding standards, but also engineering practices such as modularization of code, often alleviate the need for sophisticated analyses of the highly configurable system.

These findings suggest a number of research directions to support practical needs. Techniques to *avoid analyses* can help avert the substantial investment often required for more formal approaches. If analyses have to be conducted, they should be *lightweight* and account for the diversity of artifacts and tool chains. This also includes *unifying variability management and modeling concepts*. *Hybrid analysis* approaches that combine manual reviewing with variability-aware analyses allow practitioners to weed out false positives from static analysis approaches and allow focusing on relevant cases. The latter can also be achieved with *experience-based configuration sampling*. In terms of concrete analysis techniques, *automated test-case generation*, *test-case selection based on quality properties* as well as *traceability management* for failed tests can support practitioners in improving coverage and finding issues quickly. Traceability management can also support *change-impact analyses* that are not confined to the variability model. Finally, quicker and more flexible *consistency checking* and *feature-interaction analysis* for large and loosely-coupled systems will reduce the length of feedback cycles and support continuous integration and deployment.

References

[MNM+18] Mukelabai Mukelabai, Damir Nešić, Salome Maro, Thorsten Berger, Jan-Philipp Steghöfer: Tackling combinatorial explosion: a study of industrial needs and practices for analyzing highly configurable systems. In: Proceedings of the 33rd ACM/IEEE International Conference on Automated Software Engineering (ASE'18), September 2018, pp. 155–166, ACM. https://doi.org/10.1145/3238147.3238201

Session 7: Behavioural Software Engineering

Towards a Theory of Software Development Expertise

Sebastian Baltes[1] Stephan Diehl[2]

Abstract: Software development includes diverse tasks such as implementing new features, analyzing requirements, and fixing bugs. Being an expert in those tasks requires a certain set of skills, knowledge, and experience. Several studies investigated individual aspects of software development expertise, but what is missing is a comprehensive theory. We present a first conceptual theory of software development expertise that is grounded in data from a mixed-methods survey with 335 software developers and in literature on expertise and expert performance. Our theory currently focuses on programming, but already provides valuable insights for researchers, developers, and employers. The theory describes important properties of software development expertise and which factors foster or hinder its formation, including how developers' performance may decline over time. Moreover, our quantitative results show that developers' expertise self-assessments are context-dependent and that experience is not necessarily related to expertise.

Keywords: software engineering; expertise; theory; psychology

1 Introduction

An *expert* is, according to Merriam-Webster, someone "having or showing special skill or knowledge because of what [s/he has] been taught or what [s/he has] experienced". K. Anders Ericsson, a psychologist and expertise researcher, defines *expertise* as "the characteristics, skills, and knowledge that distinguish experts from novices and less experienced people". For some areas, such as playing chess, there exist representative tasks and objective criteria for identifying experts. In software development, however, it is more difficult to find objective measures for quantifying expert performance. Aside from programming, software development involves many other tasks such as requirements engineering, testing, and debugging, at which a software development expert is expected to be good.

In the past, researchers investigated certain aspects of software development expertise such as the influence of programming experience, desired attributes of software engineers, or the time it takes for developers to become "fluent" in software projects. However, there is currently no theory combining those individual aspects. Such a theory could help structuring existing knowledge about software development expertise in a concise and precise way and hence facilitate its communication.

[1] University of Trier, Software Engineering, 54296 Trier, Germany, research@sbaltes.com
[2] University of Trier, Software Engineering, 54296 Trier, Germany, diehl@uni-trier.de

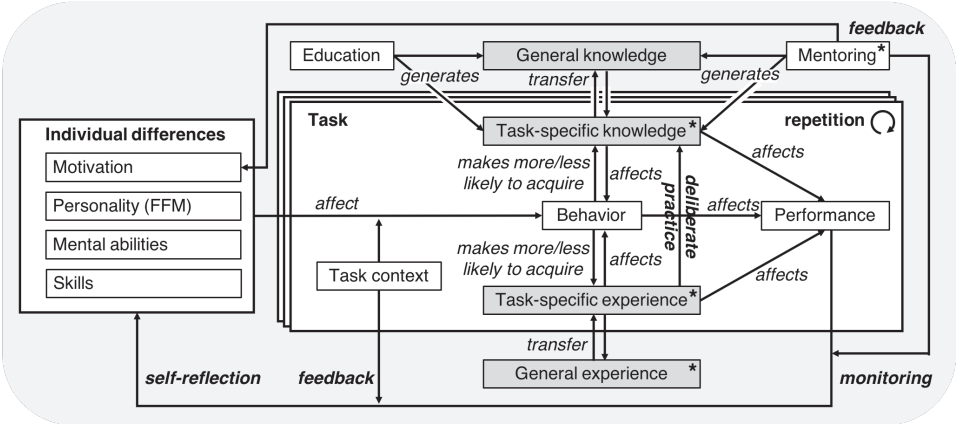
doi:10.18420/se2019-22

With our paper, we contribute a theory that describes *central properties* of software development expertise and *important factors* influencing its *formation*. The theory is grounded in data from a mixed-methods survey with 335 participants and in literature on expertise and expert performance. Our expertise model is task-specific, but includes the notion of transferable knowledge and experience from related fields or tasks. On a conceptual level, the theory focuses on factors influencing the formation of software development expertise over time.

The theory can help researchers, software developers as well as employers. *Researchers* can use it to design studies related to expertise and expert performance, and in particular to reflect on the complex relationship between experience and expertise, which is relevant for many self-report studies. *Software developers* can learn which properties are distinctive for experts in their field, which behaviors may lead to becoming a better software developer, and which contextual factors could affect expertise development. If they are already "senior", they can learn what other developers expect from good mentors or which effects age-related performance decline may have on them. Finally, *employers* can learn what typical reasons for demotivation among their employees are, hindering developers to improve, and how they can build a work environment supporting expertise development of their staff.

2 Conceptual Theory

The following figure shows the central concepts and relationships of our theory. More details on our research design and the resulting theory can be found in the full paper [BD18].



References

[BD18] Baltes, Sebastian; Diehl, Stephan: Towards a Theory of Software Development Expertise. In: 26th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering (ESEC/FSE 2018). ACM, pp. 187–200, 2018.

Understanding How Programmers Forget

Jacob Krüger[1,2], Jens Wiemann[2], Wolfram Fenske[2], Gunter Saake[2], Thomas Leich[1,3]

Abstract: This extended abstract is based on our paper "Do You Remember This Source Code?", published at the International Conference on Software Engineering 2018 [Kr18]. We summarize and discuss our results on programmers' memory and forgetting. To this end, we focus on reverse engineering of software, which was the primary context in which we conducted this work.

Keywords: Familiarity; forgetting; empirical study; maintenance; program comprehension

During software development, programmers rely on their knowledge about a system. More precisely, software familiarity comprises knowledge about the system's source code, design, architecture, and usage. Consequently, being familiar with a system is essential for various aspects of software development, such as task performance, program comprehension, bug fixing, maintenance, and re-engineering. Thus, researchers and practitioners consider familiarity as an important factor in techniques for cost modeling, expertise identification, and task assignments. Still, empirical analyses of software familiarity are sparse and these techniques usually rely on indirect heuristics or user estimates to consider forgetting.

Within our previous paper [Kr18], we report an empirical study with 60 open-source developers. We investigated whether an established forgetting model applies to software familiarity, analyzed three factors that may influence the memory, and computed a memory strength for our participants. In Figure 1, we display the familiarity of our participants, relating the times since their last commit to their subjective familiarity. Each circle represents a single developer and its size the number of commits the developer performed on the selected file. The solid blue line represents the average familiarity the developers stated, while the red dashed line represents the average only considering the data for single commits.

We can see that, besides the time, repeated commits to a file seem to significantly influence the familiarity of a developer. This is indicated by the peak of the average that appears at around 120 days after the last commit. In order to test this observation and other hypotheses, we performed two significance tests for four factors: The code size of the file; for which we tested that it does not have any impact, which we assumed based on the used forgetting curve; the number of repeated commits, the ratio of own code, and the tracking behavior of a developer. Our results indicate that *repetitions as well as the ratio of own code are significantly positively correlated to familiarity*. Furthermore, we computed the average

[1] Harz University of Applied Sciences Wernigerode; tleich@hs-harz.de
[2] Otto-von-Guericke-University Magdeburg; jkrueger@ovgu.de; wfenske@ovgu.de; saake@ovgu.de
[3] METOP GmbH Magdeburg

doi:10.18420/se2019-23

memory strengths to compare the used forgetting curve to our participants' responses. The dashed red line that we show in Figure 1 resembles this curve and matches with our results: We found that the curve is only applicable if a developer edited a file only once, as this mitigates the effects of repetitions and changes that others may perform. Thus, we conclude that *we require adapted forgetting curves for software development* that consider the identified factors that correlate to forgetting. These curves and factors can then be considered in software engineering techniques—based on further analyses of forgetting.

We conducted this research in the context of re-engineering software artifacts and estimating the efforts to perform this task. During re-engineering, software developers have to comprehend the code they aim to refactor and migrate. As program comprehension is one of the main and most costly activities in software development, the effort of re-engineering is heavily dependent on the remaining familiarity a developer may have. The question arises, whether it is more suitable to assign the original developer who implemented most code (ratio of own code) or the developer how did most (repetitions) or more recent (time) changes? Answering this and related questions is challenging and often based on educated guesses. In our research, we are concerned with understanding the impact of forgetting especially on re-engineering tasks, supporting decision making, improving cost estimations, and facilitating expert identification. Thus, our results help researchers to scope new techniques and practitioners in making reasonable decisions. We aim to conduct more studies to verify and extent our findings in future work, focusing on the impact of familiarity on development activities and utilizing implicit and explicit knowledge, for example, in version control systems, to support developers.

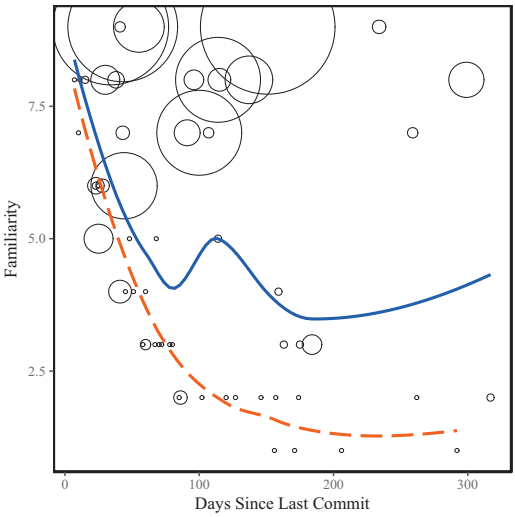


Fig. 1: Participants' familiarity related to the time since the last commit. The size of a circle represents the number of commits, the solid blue line average values, and the dotted red line the average of responses with only one commit.

Acknowledgments: Supported by DFG grants LE 3382/2-1 and SA 465/49-1.

References

[Kr18] Krüger, Jacob; Wiemann, Jens; Fenske, Wolfram; Saake, Gunter; Leich, Thomas: Do You Remember This Source Code? In: International Conference on Software Engineering. ICSE. ACM, pp. 764–775, 2018.

Positive Affect through Interactions in Meetings: The Role of Proactive and Supportive Statements

Kurt Schneider[1], Jil Klünder[1], Fabian Kortum[1], Lisa Handke[2], Julia Straube[2], Simone Kauffeld[2]

Abstract: Meetings often dominate software projects. Despite frequent dissatisfaction within meetings, many software engineers are not aware of the crucial importance about their behavior. This can set the tone for the entire project and influence the success. In a study based on 32 student development teams with 155 participants, we observed the participants' interactions during the first internal team meeting. Analyzing the observations showed that constructive remarks had a positive impact on positive group affect tone. However, this effect can only be observed when supportive utterances followed constructive remarks. In the article, we show a complete mediation of this statistically significant effect, e.g., about seemingly subtle interaction patterns that influence group affect tone. We also propose practical interventions on how software projects could benefit from supportive meeting behavior. This summary refers to the article *"Positive affect through interactions in meetings: The role of proactive and supportive statements"* [Sc18] published in the Journal of Systems & Software in 2018 (vol. 143).

Keywords: Positive group affect tone (PGAT), proactive and supportive statements, empirical study

1 Introduction

Most software projects require some sort of team work. In order to ensure an appropriate information exchange and the synchronization of all team members, meetings are part of most projects. The success of a meeting in the means of satisfied participants and desired meeting outcomes strongly depends on the interactions during the meetings. While proactive and structuring statements can support meeting success, destructive behavior endangers meeting success – and influences the further project. However, a lot of meeting participants are not aware of the relevance of their behavior. Interaction analysis can help to reveal appropriate and inappropriate behavior in meetings and to increase the participants' awareness of the relevance of their behavior [KLW12]. In particular, appropriate behavior such as proactive statements in meetings can support positive affect afterwards.

[1] Software Engineering Group, Leibniz Universität Hannover, Welfengarten 1, 30167 Hannover, Germany {kurt.schneider,jil.kluender,fabian.kortum}@inf.uni-hannover.de
[2] Department of Industrial, Organizational and Social Psychology, Technische Universität Braunschweig, Spielmannstr. 19, 38106 Braunschweig, Germany {l.handke,julia.straube,s.kauffeld}@tu-braunschweig.de

doi:10.18420/se2019-24

2 Study and Results

In a large-scale study, we analyzed the first team meeting of 32 student development teams. Each team consisted of 3 to 5 students, leading to a total number of 155 participants. Each of the meetings was analyzed using the established act4teams coding scheme [KLW12]. Applying this coding scheme leads to a fine-grained overview of the number of different types of statements, such as supporting, interrupting or complaining statements. In our study, we concentrated on supportive and proactive statements. Supportive statements are given by agreements to suggestions or ideas, e.g., "yes, this makes sense...". Proactive statements are mainly given by signalizing interest in change, e.g., "Let's assume we are successful...". Correlation analysis comparing the number of different statements with self-reported data on positive and negative affect leads to three core statements [Sc18]:

- The rate of proactive statements within a meeting does not significantly influence positive group affect after the meeting.
- Positive group affective tone is triggered by proactive statements via support.
- The probability of a supportive statement is significantly higher when a proactive statement has been made.

3 Conclusion

Our results show the importance of appropriate behavior in meetings and its influence on the success of a meeting. In particular, our results show that "a verbal and content-oriented utterance has a measurable impact on positive affect" [Sc18]. Consequently, not only direct and non-verbal stimuli influences positive affect, but also serious constructive statements. Furthermore, positive affect is supported by triggers and support from other meeting participants.

Literaturverzeichnis

[KLW12] Kauffeld, Simone; Lehmann-Willenbrock, Nale: Meetings matter: Effects of team meetings on team and organizational success. Small Group Research, 43(2):130–158, 2012.

[Sc18] Schneider, Kurt; Klünder, Jil; Kortum, Fabian; Handke, Lisa; Straube, Julia; Kauffeld, Simone: Positive affect through interactions in meetings: The role of proactive and supportive statements. Journal of Systems and Software, 143:59–70, 2018.

Session 8: Modelle und Anforderungen

S. Becker, I. Bogicevic, G. Herzwurm, S. Wagner (Hrsg.): SE/SWM 2019,
Lecture Notes in Informatics (LNI), Gesellschaft für Informatik, Bonn 2019 91

Model Transformation Languages under a Magnifying Glass: A Controlled Experiment with Xtend, ATL, and QVT

Regina Hebig[1] Christoph Seidl[2] Thorsten Berger[3] John Kook Pedersen[4]
Andrzej Wąsowski[5]

Abstract: In Model-Driven Software Development, models are processed automatically to support the creation, build, and execution of systems. A large variety of dedicated model-transformation languages exists, promising to efficiently realize the automated processing of models. To investigate the actual benefit of using such specialized languages, we performed a large-scale controlled experiment in which 78 subjects solved 231 individual tasks using three languages. The experiment sheds light on commonalities and differences between model transformation languages (ATL, QVT-O) and on benefits of using them in common development tasks (comprehension, change, and creation) against a modern general-purpose language (Xtend). The results of our experiment show no statistically significant benefit of using a dedicated transformation language over a modern general-purpose language. However, we were able to identify several aspects of transformation programming where domain-specific transformation languages do appear to help, including copying objects, context identification, and conditioning the computation on types.

Keywords: Model Transformation Languages; Experiment; Xtend; ATL; QVT

1 Introduction

In Model-Driven Software Development (MDSD), models are processed automatically to support creation, build and execution of systems. Transformations are, among others, used to compute views on models, to validate models, to refactor models as well as to interpret or otherwise execute models. We are specifically concerned with model-to-model (M2M) transformations, i.e., programs transforming instances of a source model to instances of a target model (structured data to structured data). Respective M2M transformation languages come with an implicit promise to be easier to use and more efficient for specifying transformations than general-purpose programming languages (GPLs). In this line of work, we investigated whether this promise holds.

[1] Chalmers | University of Gothenburg, Sweden, regina.hebig@cse.gu.se
[2] Technische Universität Braunschweig, Germany, c.seidl@tu-braunschweig.de
[3] Chalmers | University of Gothenburg, Sweden, thorsten.berger@chalmers.se
[4] IT University of Copenhagen, jkpe@itu.dk
[5] IT University of Copenhagen, wasowski@itu.dk

doi:10.18420/se2019-25

2 Method

We performed a pre-study in collaboration with a Danish industrial partner to investigate suitability of model transformation technology in the context of data aggregation. The results lead to the design of the subsequent experiment, where we had participants perform tasks to comprehend, change and create transformations for typical M2M scenarios. We selected ATL and QVT-O as M2M languages as well as Xtend as imperative GPL.

The experiment was performed in three runs at Chalmers | University of Gothenburg and Technische Universität Braunschweig. Each run of the experiment was performed in a lecture room where participants had enough space to work. Each participant had to use one of the languages, which we assigned to them arbitrarily beforehand, to perform the given tasks. We distributed task sheets in such a way that participants sitting next to each other would solve different tasks (i.e., different languages and/or different M2M scenario). The tasks had to be solved on paper, which eliminates the factor of familiarity with the programming environment. In total, we recruited 78 graduate students who participated in the experiment voluntarily.

3 Results

Analyzing solutions of 231 individual tasks, we found that:

- Handling multi-valued features (collections), recursion, and designing logical branching conditions are among the most difficult skills to master.
- Even well qualified subjects struggle to evolve transformations optimally, producing changes of widely diverse sizes.
- Implicit object creation/structure copying, support for type-driven computation and explicit computation context do appear to reduce the amount of errors.

Furthermore, the results of our experiment show no statistically significant benefit of using a dedicated M2M language over a modern GPL. However, we were able to identify several aspects of transformation programming where domain-specific transformation languages do appear to help, including copying objects, context identification, and conditioning the computation on types. Please refer to our paper for details on the experiment setup, conduction, results and conclusions [He18].

References

[He18] Hebig, Regina; Seidl, Christoph; Berger, Thorsten; Kook Pedersen, John; Wąsowski, Andrzej: Model Transformation Languages Under a Magnifying Glass - A Controlled Experiment with Xtend, ATL, and QVT. In: Proceedings of the 26th Symposium on the Foundations of Software Engineering (FSE). FSE'18, 2018.

How are Conceptual Models used in Industrial Software Development? A Descriptive Survey

Harald Störrle[1]

Original publication: Proc. 21st. Intl. Conf. Evaluation and Assessment in Sw. Eng. (EASE), 2017, pp. 518–534, E. Mendes, K. Petersen, S. Counsell (eds.), ACM, 2017.

There has been considerable controversy regarding the extent to which industrial software engineering benefits from conceptual modeling. Those in favor maintain that *"Model-based approaches [...] hold out the promise of significantly improving the productivity of software developers and the quality of the products they generate"*. They particularly promote using the Unified Modeling Language (UML) as the *"lingua franca of software engineering"* in the context of the Model Driven Engineering (MDE), and claim increased efficiency of software development, and better product quality. However, Mohagheghi and Dehlen famously asked *"Where is the proof?"* in their 2008 landmark paper. Even MDA-supporters had to admit that *"adoption of this approach has been surprisingly slow"*. Some scientists have outright denied that UML is used to any degree in industry [Pe13]. Where UML is used, Petre claims, it is used only informally, and *"if models end up merely as documentation, they are of limited value [...]."*.

In order to clarify this issue, we conducted a survey (see `http://tinyurl.com/MU-survey-2014` for the live survey, and `https://figshare.com/s/86d797cc3b816e972f1b` for anonymized results). We asked for specific usages of models and offered 16 predefined items with a five-point Likert scale. We also asked participants to describe further scenarios in prose, resulting in only one additional usage category ("Testing and Verification"). The results Fig. 1 indicate that the most popular usage scenarios by far (scenarios 1-3) center around communicative and cognitive processes: between 70 and 79% of all participants use models for such activities "often" or "always", while only between 4 and 8% do this "rarely" or "never". In contrast, about half the population in our survey used models "rarely" or "never" for generating code or a DSL, while only a quarter to a third of the population did this "often" or "always". Also, using models for domain- and requirements-oriented tasks (scenarios 4, 6, 9) is apparently more common than using models for technology-specific tasks (scenarios 5, 7, 10, 11). This is in line with the findings of [Pe13] who concludes that UML is mostly used as a "thought tool" and to facilitate communication with stakeholders. Usage scenario 2 ("Visualize an idea or concept") is the one that is used at least "sometimes" by most participants (96%).

[1] QAware GmbH, Aschauer Str. 32, 81549 München, Germany, HStorrle@acm.org

doi:10.18420/se2019-26

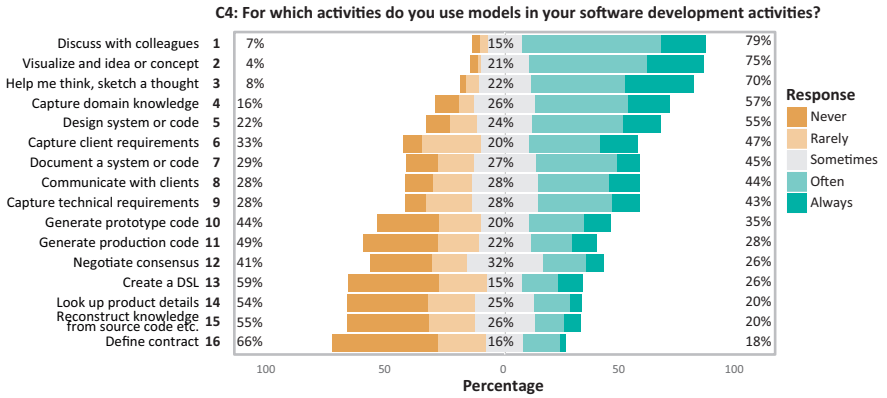


Fig. 1: Frequency of model usage scenarios sorted by decreasing frequency.

Based on our findings and prior experience, we distinguish three model usage scenarios: **Informal models** support communication and cognition, utilizing rich information implicit in the situational context; **Semi-formal models** support design and documentation activities; and **Fully formal models** are to be taken literal and binding, so as to allow the analysis of system properties, simulation, and generation of code and test cases. Observe that this grouping aligns with Fowler's distinction of models as sketches, blueprints, and programs [Fo03]. From this and other results of our study, we conclude that

- conceptual modeling languages like UML or BPMN are indeed used in industry, at least in some regions of the world.
- Models are used for a great variety of purposes by diverse stakeholders in the context software development, though Software Architects seem to benefit most from modeling.
- There are three distinct modes of modeling: for cognition and communication, for planning and documentation, and for code generation and contracts.
- Informal modeling clearly dominates in terms of frequency, followed by semi-formal modeling, but even formal modeling does occur in relevant quantities.

Our findings align with much of the previous research, but contrast with Petre's findings [Pe13]. It is remarkable, though, that the populations sampled by Petre and us have different regional distribution (Petre is unfortunately a little bit vague on this and many other methodological details of her study). This could indicate regional differences in software development practices, which are likely the result of more general cultural differences, and will possibly also impact other aspects of software development practice.

References

[Fo03] Fowler, Martin: UML Distilled. Addison-Wesley, 3rd edition, 2003.

[Pe13] Petre, Marian: UML in Practice. In: Proc. Intl. Conf. Software Engineering (ICSE). IEEE Press, pp. 722–731, 2013.

Linking Use Cases and Associated Requirements: On the Impact of Linking Variants on Reading Behavior

Oliver Karras[1] Alexandra Risch[1] Kurt Schneider[1]

Abstract: A wide variety of use case templates supports different linking variants. The main purpose of all linking options is to highlight the interrelationships between a use case and its associated requirements. Regardless of the linking, a reader needs to consider all materials together in order to achieve a high understanding of the overall content. Due to the efforts of creating and maintaining links, we investigated their impact on the reading behavior in terms of visual effort and intended way of interrelating both artifacts in an eye tracking study. Our findings show that all investigated linking variants cause comparable visual effort and share the most frequent reading pattern. In all cases, the use case and the requirements are read separated and successively. Nevertheless, we found significant differences in the reading behaviors between the linking variants. Only the most detailed linking variant significantly increases the number of attention switches between both artifacts which represents the required reading behavior of interrelating both artifacts. This summary refers to the paper "Interrelating Use Cases and Associated Requirements by Links – An Eye Tracking Study on the Impact of Different Linking Variants on the Reading Behavior" [KRS18] which was published as original research article in the proceedings of the 22*nd International Conference on Evaluation and Assessment in Software Engineering*.

Keywords: Linking; reading behavior; eye tracking; visual effort; attention switch

1 Introduction

Coleman [Co98] included a field for non-functional requirements in his proposed use case template. Based on this idea, further templates were invented to add any associated requirement to a use case. These templates provide different options to link to associated requirements. Based on literature, we identified three widely used linking options. Besides *no linking*, templates include either an *additional field* or *integrated links* in typical fields to list associated requirements. Links are mainly realized by labels that consist of identification numbers. These labels are one source for risky, dispersed changes of a use case. Due to the effort to create and maintain links, we investigated the following research question:

Research question:
Does the linking variant of a use case and its associated requirements influence the reading behavior in terms of visual effort and intended way of interrelating both artifacts?

[1] Leibniz Universität Hannover, Software Engineering Group, Welfengarten 1, 30167 Hannover, Germany
oliver.karras@inf.uni-hannover.de,alexandra.risch@se.uni-hannover.de,kurt.schneider@inf.uni-hannover.de

doi:10.18420/se2019-27

2 Study and Results

We conducted an eye tracking study in a between-subjects experiment with 3 groups each with 5 subjects to compare the three linking variants. Each linking variant was applied to the same use case and requirements. Based on the collected eye tracking data, we investigated the impact of the three linking variants on the reading behavior.

Visual Effort: We analyzed the visual effort by using three different metrics (*fixation count*, *fixation duration*, and *dwell time*). Based on all three metrics, there is no significant difference in the visual effort between the three linking variants. Therefore, the three linking variants cause comparable visual effort. Thus, adding links to a use case does not impact the visual effort of a reader.

Intended Way of Interrelating Both Artifacts: We analyzed the reading behavior of our subjects by applying sequential pattern mining on their scan-paths to identify *frequent reading patterns* and comparing their *attention switching frequencies*. According to our results, the main reading pattern of all three linking variants is the successive reading of the single artifacts (use case and requirements) one after the other. Only in case of *integrated links*, the joint consideration of the related materials, so-called intended way of interrelating both artifacts, occurs as a frequent reading pattern. This finding coincides with the analysis results of the attention switching frequencies. *Integrated links* significantly increase the number of attention switches between both artifacts. **The *integrated links* variant is the linking option that most increases a reader's efforts to interrelate both artifacts.**

3 Conclusion

The particular linking variant of a use case with its associated requirements has an impact on the reading behavior. Adding links to a use case does not increase the visual effort for a reader. However, the position of a link has an impact on how intensive a reader interrelates the connected artifacts. Our work indicates that all three linking variants do not impede the reading of the two artifacts for themselves. Nevertheless, the specific reading behavior of interrelating both artifacts is only supported by the detailed integration of links in a use case description. Based on our findings, we recommend preferring on the most detailed linking variant *integrated links*.

References

[Co98] Coleman, D.: A Use Case Template: Draft for Discussion. In: Use Case Template Guidelines. 1998.

[KRS18] Karras, O.; Risch, A.; Schneider, K.: Interrelating Use Cases and Associated Requirements by Links: An Eye Tracking Study on the Impact of Different Linking Variants on the Reading Behavior. In: Proceedings of the 22nd International Conference on Evaluation and Assessment in Software Engineering 2018. EASE'18, ACM, New York, NY, USA, 2018.

Session 9: Microservices und Produktlinien

TeaStore: A Micro-Service Reference Application for Benchmarking, Modeling and Resource Management Research

Jóakim v. Kistowski[1], Simon Eismann[1], Norbert Schmitt[1], André Bauer[1], Johannes Grohmann[1], Samuel Kounev[1]

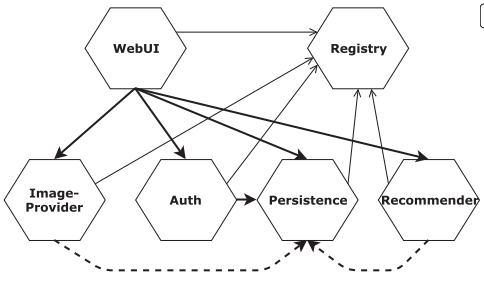
Moderne verteilte Microservice-Anwendungen haben komplexes Leistungsverhalten, da die konstituierenden Dienste unterschiedliche Leistungsmerkmale aufweisen und im Vergleich zu herkömmlichen Softwarekomponenten eine kurze Lebensdauer haben. Diese Anwendungen werden in der Regel in Container-Orchestrierungs-Frameworks wie Kubernetes, Docker Swarm oder Cloud Provider-spezifischen Lösungen eingesetzt. Diese Frameworks fügen eine weitere Komplexitätsstufe hinzu, indem sie die Service-Container automatisch skalieren, fehlerhafte Container regenerieren oder durch die Einführung von Monitoring- oder Service-Mesh-Sidecars.

Die aktuelle Forschung verwendet viele Analyse-, Modellierungs-, Optimierungs- und Managementansätze, die darauf abzielen, dieses anspruchsvolle Leistungsverhalten zu bewältigen [Ei18]. Die Evaluation, der Vergleich und die Bewertung der Ergebnisse einer solchen Forschung ist schwierig. Für die praktische Evaluation benötigen die Forscher eine Softwareanwendung, die sie als Referenz einsetzen können. Diese Referenzanwendung muss hinsichtlich des Leistungsverhaltens eine ausreichende Komplexität aufweisen, um Optimierungsmöglichkeiten zu gewährleisten. Zusätzlich sollte sie Open Source sein, für die Instrumentierung zur Verfügung stehen, reproduzierbare Ergebnisse liefern, und representativ für Anwendungen im Produktionseinsatz sein.

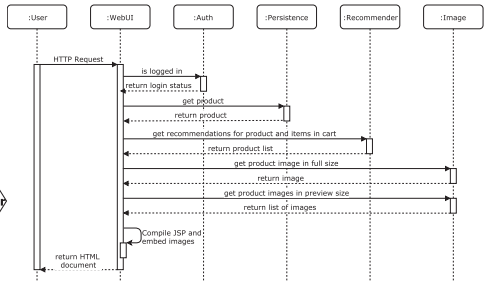
Produktivsoftware ist in der Regel proprietär und kann nicht für Experimente verwendet werden. Bestehende Test- und Referenzsoftware hingegen wird in der Regel explizit für die Auswertung eines einzelnen Beitrags erstellt, was den Vergleich erschwert. Andere bestehende und weit verbreitete Testsoftware bietet nicht die notwendigen Freiheitsgrade und wird oft manuell angepasst. Einige der am weitesten verbreiteten Test- und Referenzanwendungen, wie RUBiS oder Dell DVD Store, sind veraltet und daher nicht repräsentativ für moderne Anwendungen. Referenzanwendungen von Industrieanbietern, wie z.B. der Sock Shop von WeaveWorks, verwenden in der Regel einen modernen Technologie-Stack, sind aber so konzipiert, dass sie eine bestimmte Softwarelösung präsentieren und nicht das komplexe Leistungsprofil haben, dass die Evaluation aktueller Forschung erlauben würde.

[1] University of Würzburg, Deutschland, [Vorname].[Nachname]@uni-wuerzburg.de

doi:10.18420/se2019-28



(a) TeaStore Architektur



(b) Beispiel für den Controllfluss im TeaStore

Wir präsentieren TeaStore [Vo18], eine mikroservice-basierte Test- und Referenzanwendung, die von Forschern als Benchmarking-Framework genutzt werden kann. Der TeaStore besteht aus fünf verschiedenen Services, die jeweils über einzigartige Leistungsmerkmale und Bottlenecks verfügen. Abbildung 1a zeigt die einzelnen Services und die Kommunikation zwischen den Services. Abbildung 1b gibt ein Beispiel wie mithilfe dieser Services eine Anfrage beantwortet wird. Die Kommunikation zwischen den Services basiert auf REST-Aufrufen. Aufgrund der unterschiedlichen Ressourcenauslastung der Dienste stellt der TeaStore interessante Herausforderungen in den Bereichen Performance Modeling, Auto-Scaling, Resource Allocation und Energieeffizienz dar. Der TeaStore ist skalierbar und unterstützt lokale und verteilte Deployments. Darüber hinaus erlaubt die Architektur die Laufzeitskalierung, da Dienste und Service-Instanzen zur Laufzeit hinzugefügt, entfernt und repliziert werden können. Um die Analyse für Forscher zu vereinfachen bieten wir mit Kieker [VWH12] Monitoring instrumentierte Docker Container für alle TeaStore Services an.

Literatur

[Ei18] Eismann, S.; Walter, J.; von Kistowski, J.; Kounev, S.: Modeling of Parametric Dependencies for Performance Prediction of Component-based Software Systems at Run-time. In: 2018 IEEE International Conference on Software Architecture (ICSA). S. 135–144, 2018.

[Vo18] Von Kistowski, J.; Eismann, S.; Schmitt, N.; Bauer, A.; Grohmann, J.; Kounev, S.: TeaStore: A Micro-Service Reference Application for Benchmarking, Modeling and Resource Management Research. In: Proceedings of the 26th IEEE International Symposium on the Modelling, Analysis, and Simulation of Computer and Telecommunication Systems. MASCOTS '18, 2018.

[VWH12] Van Hoorn, A.; Waller, J.; Hasselbring, W.: Kieker: A framework for application performance monitoring and dynamic software analysis. In: Proceedings of the 3rd ACM/SPEC International Conference on Performance Engineering. S. 247–248, 2012.

Experience with Microservices for Legacy Software Modernization

Holger Knoche[1], Wilhelm Hasselbring[2]

Abstract: Microservices are known as an architecture for building scalable applications running in the cloud. However, they also promise high maintainability due to smaller code bases and strong component separation, making them an interesting option for software modernization. We present a migration process to decompose an existing application into microservices, and report on experiences from applying this process in an ongoing legacy modernization project.

Keywords: Microservices; Software Modernization

Microservices are an architectural style for software which currently receives a lot of attention in both industry and academia [Ha18]. In addition to scalability [Ha16], microservices may furthermore enable both agility and reliability [HS17]. As a consequence, many companies are currently considering whether microservices are a viable option for their software systems [KH19]. Microservices are also used for modular research software [Jo16].

A particularly interesting aspect of microservices is that several authors consider them as a viable means for modernizing existing, monolithic software applications [BHJ16]. For instance, the software visualization tool ExplorViz [FKH17; FRH15] has been restructured into microservices [ZKH18]. Furthermore, there are experience reports of several companies who have successfully replaced (parts of) their existing software by microservices, or are in the process of doing so.

Based on experience from an ongoing industrial legacy modernization project, we recommend an incremental approach for migration towards microservices consisting of five steps [KH18]. The project's main goals are to improve the maintainability of a legacy system by breaking it into strictly isolated components based on the bounded contexts of the underlying domain, and to incrementally migrate the system from Cobol to Java. Microservices are a suitable architecture for achieving these goals due to their organization around business capabilities, high evolvability, strong component separation, and focus on cross-platform interaction.

Our approach aims at isolating – and subsequently replacing – microservice candidates inside the monolith using well-defined, platform independent interfaces, while ensuring

[1] Kiel University, Software Engineering Group, Kiel, hkn@informatik.uni-kiel.de
[2] Kiel University, Software Engineering Group, Kiel, hasselbring@email.uni-kiel.de

doi:10.18420/se2019-29

that the functional requirements of the surrounding monolith are still met. As legacy modernization projects are both risky and costly and can take a long time to complete, the approach is designed to improve the structure of the monolith even before all steps are completed.

Literatur

[BHJ16] Balalaie, A.; Heydarnoori, A.; Jamshidi, P.: Microservices Architecture Enables DevOps: Migration to a Cloud-Native Architecture. IEEE Software 33/3, S. 42–52, 2016.

[FKH17] Fittkau, F.; Krause, A.; Hasselbring, W.: Software Landscape and Application Visualization for System Comprehension with ExplorViz. Information and Software Technology 87/, S. 259–277, Juli 2017.

[FRH15] Fittkau, F.; Roth, S.; Hasselbring, W.: ExplorViz: Visual Runtime Behavior Analysis of Enterprise Application Landscapes. In: 23rd European Conference on Information Systems (ECIS 2015). S. 1–13, Mai 2015.

[Ha16] Hasselbring, W.: Microservices for Scalability: Keynote Talk Abstract. In: Proceedings of the 7th ACM/SPEC International Conference on Performance Engineering (ICPE). ACM, S. 133–134, 2016.

[Ha18] Hasselbring, W.: Software Architecture: Past, Present, Future. In: The Essence of Software Engineering. Springer, S. 169–184, 2018, URL: https://doi.org/10.1007/978-3-319-73897-0_10.

[HS17] Hasselbring, W.; Steinacker, G.: Microservice Architectures for Scalability, Agility and Reliability in E-Commerce. In: Proceedings of the 2017 IEEE International Conference on Software Architecture Workshops (ICSAW). IEEE, S. 243–246, 2017.

[Jo16] Johanson, A.; Flögel, S.; Dullo, C.; Hasselbring, W.: OceanTEA: Exploring Ocean-Derived Climate Data Using Microservices. In: Proceedings of the Sixth International Workshop on Climate Informatics (CI 2016). S. 25–28, 2016.

[KH18] Knoche, H.; Hasselbring, W.: Using Microservices for Legacy Software Modernization. IEEE Software 35/3, S. 44–49, 2018.

[KH19] Knoche, H.; Hasselbring, W.: Drivers and Barriers for Microservice Adoption – A Survey among Professionals in Germany. Enterprise Modelling and Information Systems Architectures (EMISAJ) 14/, (in press), 2019.

[ZKH18] Zirkelbach, C.; Krause, A.; Hasselbring, W.: On the Modernization of ExplorViz towards a Microservice Architecture. In: Combined Proceedings of the Workshops of the German Software Engineering Conference 2018. Bd. Vol-2066, CEUR Workshop Proceedings, 2018, URL: http://ceur-ws.org/Vol-2066/.

Software-Produktlinien agilisieren: Ein Transformationsmodell für große Unternehmen

Jil Klünder[1], Philipp Hohl[2], Kurt Schneider[1]

Abstract: In der heutigen Zeit erfordert der Markt immer kürzer werdende Auslieferungszeiten von qualitativ hochwertiger Software. Dies geht mit kürzeren Entwicklungszyklen einher. Viele Ansätze zur Verbesserung des Softwareentwicklungsprozesses schlagen die Integration agiler Entwicklungsmethoden vor. Diese Integration ist jedoch nicht immer einfach. Große Unternehmen wie beispielsweise in der Automobilbranche treffen auf das Problem, agile Methoden in bereits existierende Softwareentwicklungsprozesse zu integrieren. Häufig basiert der Prozess auf dem Einsatz von Software-Produktlinien, die die Wiederverwendung von Software ermöglichen. Um eine Transformation des Softwareentwicklungsprozesses unter Erhaltung von Produktlinien zu unterstützten, wurde basierend auf einer Literaturstudie mit 164 Papern ein Transformationsmodell entwickelt. Diese Zusammenfassung basiert auf der Publikation "Becoming Agile While Preserving Software Product Lines: An Agile Transformation Model For Large Companies" [KHS18], die im Rahmen der *International Conference on the Software and Systems Process (ICSSP 2018)* veröffentlicht wurde.

Keywords: Transformationsmodell, Software-Produktlinien, Transformationsprozess

1 Einleitung

Um langfristig wettbewerbsfähig zu bleiben, streben viele Unternehmen eine agile Transformation ihrer Entwicklungsprozesse an. Dies geht meist mit einer Neugestaltung des Entwicklungsprozesses einher, um kontinuierlich lernen und sich an die Marktgegebenheiten anpassen zu können. Durch die Integration von agilen Methoden und Praktiken versprechen sich die Unternehmen zudem verkürzte Auslieferungszeiten, kürzere Entwicklungszyklen und die Möglichkeit, eine Aussage über die Qualität der Software in frühen Entwicklungsphasen abzugeben. Gerade in großen Unternehmen, deren Softwareentwicklungsprozess von einer Software-Produktlinie geprägt ist, stellt die agile Transformation eine Herausforderung dar. Vorteile, die aus der Verwendung einer Produktlinie resultieren, sollen bei einer agilen Transformation erhalten bleiben. Wenngleich es viele Transformationsmodelle gibt, unterstützt keines explizit die Erhaltung von Software-Produktlinien. Diese Publikation präsentiert ein Transformationsmodell, das funktionierende Ansätze aus der Literatur kombiniert und zusätzlich eine Erhaltung der Produktlinien unterstützt.

[1] Leibniz Universität Hannover, Software Engineering Group, Welfengarten 1, 30167 Hannover, Deutschland jil.kluender@inf.uni-hannover.de, kurt.schneider@inf.uni-hannover.de

[2] Daimler AG, Research and Development, Wilhelm Runge Straße 11, 89081 Ulm, Germany philipp.hohl@daimler.com

doi:10.18420/se2019-30

2 Transformationsmodell

Das Transformationsmodell basiert auf einer Literaturstudie [KHS18]. Aus 66 Publikationen, die Transformationsprozesse in großen Unternehmen beschreiben, wurden mögliche Ansätze, Best Practices und Erfahrungen extrahiert und kombiniert.
Für eine erfolgreiche Transformation müssen verschiedene Geschäftsbereiche des Unternehmens einbezogen werden: das Management, die Entwicklerteams, sowie die gesamte Organisation. Die wichtigsten Aufgaben des Managements sind die Definition der Unternehmensziele, die durch die Transformation erreicht werden sollen, das Herausarbeiten eines Projektplans, die Risikoevaluierung, die Budgetplanung und die Anpassung der hierarchischen Strukturen. Die Entwicklerteams sind dafür verantwortlich, sich das notwendige Wissen anzueignen, um adäquate Entscheidungen in Bezug auf den Entwicklungsprozess treffen zu können. Zudem können Trainings helfen, die agile Kultur zu etablieren. Gemeinsam mit dem Management müssen die Entwicklerteams den Entwicklungsprozess definieren und geeignete agile Methoden und Praktiken auswählen. Der Entwicklungsprozess stellt zudem oftmals Anforderungen an die Infrastruktur. Aufgaben und Verantwortlichkeiten sind ebenso wie die unterschiedlichen Rollen im Entwicklungsteam (neu) zu definieren. Bei der Transformation helfen Pilotstudien, regelmäßiges Feedback, Reviews und ein kontinuierliches Lernen. Auf diese Weise kann auch der kulturelle Wandel im Unternehmen angestrebt werden, der erforderlich ist, um die Vorteile agiler Entwicklungsmethoden zu nutzen [Ho18].

3 Zusammenfassung

Berücksichtigt ein Unternehmen die zuvor beschriebenen Aufgaben während der agilen Transformation, kann es agile Methoden und Praktiken in den Entwicklungsprozess integrieren, ohne auf die Vorteile der Software-Produktlinien zu verzichten. Um die Güte und die Anwendbarkeit des in [KHS18] präsentierten Modells zu evaluieren, sind jedoch weitere Studien vonnöten.

Literaturverzeichnis

[Ho18] Hohl, Philipp; Klünder, Jil; van Bennekum, Arie; Lockard, Ryan; Gifford, James; Münch, Jürgen; Stupperich, Michael; Schneider, Kurt: Back to the future: origins and directions of the “Agile Manifesto” – views of the originators. Journal of Software Engineering Research and Development, 6(1):15, Nov 2018.

[KHS18] Klünder, Jil; Hohl, Philipp; Schneider, Kurt: Becoming Agile While Preserving Software Product Lines: An Agile Transformation Model for Large Companies. In: Proceedings of the International Conference on Software and System Process. ICSSP ’18, ACM, New York, NY, USA, S. 1–10, 2018.

Identifying the Intensity of Variability Changes in Software Product Line Evolution

Christian Kröher[1], Lea Gerling[2], Klaus Schmid[3]

Abstract: This extended abstract summarizes the paper *Identifying the Intensity of Variability Changes in Software Product Line Evolution* [KGS18] published in the proceedings of the SPLC 2018 [BBB+18].

Keywords: Software product line evolution; evolution analysis; variability changes; intensity

A particular focus of Software Product Line (SPL) research is on understanding the evolution of variability information. This type of information realizes the configuration knowledge enabling the customization of artifacts for a specific product of a SPL, like the configuration options in the variability model or explicit references to these options in pre-processor statements. However, this research is typically based on a feature-perspective, which considers changes only in relation to a specific feature and, hence, abstracts from the implementation details of SPL evolution. As a consequence, the *frequency* with which developers generally change a specific *amount* of variability information in different types of artifacts independent from its relation to a specific feature is currently unknown. We believe that this also hides potential for improving current SPL analysis and verification approaches towards a better evolution support. In particular, the *intensity* (the frequency and amount) of changes to a particular type of artifact and information defines how often certain SPL analyses need to be redone in an evolutionary setting. Hence, we raise and answer the following research questions in this paper:

RQ1 How often do changes affect variability information in general?

RQ2 How often do changes affect variability information in code, build, and variability model artifacts?

RQ3 How many code, build, or variability model lines containing variability information are changed on average?

[1] University of Hildesheim, Institute of Computer Science, Universitätsplatz 1, 31141 Hildesheim, Germany
kroeher@sse.uni-hildesheim.de

[2] University of Hildesheim, Institute of Computer Science, Universitätsplatz 1, 31141 Hildesheim, Germany
gerling@sse.uni-hildesheim.de

[3] University of Hildesheim, Institute of Computer Science, Universitätsplatz 1, 31141 Hildesheim, Germany
schmid@sse.uni-hildesheim.de

doi:10.18420/se2019-31

In order to answer these questions, we first introduce a fine-grained and variability-centric evolution analysis approach, which provides the necessary details about the evolution of code, build, and variability model artifacts in C-preprocessor-based SPLs. Second, we present the results of applying this approach to the Linux kernel. These results cover more than 12 years of active development as documented by the Git repository[4] between 2005 and 2017. Finally, we discuss the results of this application in relation with the research questions, which reveals the following answers:

RQ1 Changes to variability information occur in about 11% of all commits from which only around 2% change this information exclusively. The majority of the commits (about 80%) exclusively changes information, which is not related to the configuration knowledge.

RQ2 Most of the changes to variability information affect code artifacts (7%) followed by variability model artifacts (5%) and build artifacts (3%).

RQ3 For code and build artifacts, only about 2-3% of the commits change 1-10 lines containing variability information, while for the variability model artifacts such changes occur only slightly more frequently (4%). Commits changing more lines at once occur even less frequently.

In summary, these results show that changes to variability information occur surprisingly infrequently and only affect small parts of the analyzed artifact types. On this basis, we further outline opportunities for improving approaches and tools to support SPL evolution. The main benefit of considering fine-grained information about the actual changes to variability information in the design of new approaches for evolutionary analysis is the reduction of analysis effort.

Acknowledgments: This work is partially supported by the Evoline project, funded by the DFG (German Research Foundation) under Priority Programme SPP 1593 and by the ITEA3 project REVaMP^2, funded by the BMBF (German Ministry of Research and Education) under grant 01IS16042H. Any opinions expressed herein are solely by the authors and not of the DFG or BMBF.

References

[BBB+18] T. Berger, P. Borba, G. Botterweck, T. Männistö, D. Benavides, S. Nadi, T. Kehrer, R. Rabiser, C. Elsner, M. Mukelabai, eds. *Proceedings of the 22nd International Systems and Software Product Line Conference*. Vol. 1. ACM, 2018.

[KGS18] C. Kröher, L. Gerling, K. Schmid. "Identifying the Intensity of Variability Changes in Software Product Line Evolution". In: *22nd International Systems and Software Product Line Conference*. ACM, 2018, pp. 54–64.

[4] https://git.kernel.org/pub/scm/linux/kernel/git/torvalds/linux.git

Session 10: Software Management I

Connected-Car-Services: eine Klassifikation der Plattformen für das vernetzte Automobil

Micha Bosler[1], Christopher Jud[2] und Georg Herzwurm[3]

Abstract: Plattformen stellen zunehmend in den unterschiedlichsten Industrien eine wichtige Komponente der Wertschöpfung dar. Das zeigt sich im Kontext der "Connected Cars" auch in der Automobilindustrie. Vernetzte Fahrzeuge mit Konnektivität-Funktionalitäten ermöglichen das Angebot neuartiger Dienste für Fahrer und Halter. Diesbezüglich etablieren Hersteller, Zulieferer sowie ehemals branchenfremde Unternehmen plattformbasierte Konstrukte, um die zugehörigen Connected-Car-Services zu realisieren. Basierend auf Erkenntnissen aus Experteninterviews werden die wichtigsten Typen der Connected-Car-Plattformen im Rahmen einer Klassifikation charakterisiert.

Keywords: Connected Cars · vernetzte Mobilität · Connected-Car-Plattformen · Connected-Car-Services · Plattformen für das vernetzte Automobil

1 Einleitung

Die zunehmende Digitalisierung von und das Angebot digitaler Services (softwarebasierter Erweiterungen, angeboten als so genannte Mehrwertdienste) für Produkte zeigt sich in einer zunehmenden Anzahl von Industrien. Das führt auch in der Automobilbranche zu einem grundlegenden Wandel hin zu vernetzten Fahrzeugen: So genannte Connected-Car-Plattformen werden von verschiedenen Akteuren angeboten und betrieben, neben den Automobilherstellern (den OEMs), sind dies u.a. auch Zulieferer. In Folge dessen entsteht eine heterogene, für Außenstehende undurchsichtige, Plattform-Landschaft.

Angesichts der komplexen Ausgangssituation untersucht und klassifiziert der vorliegende Beitrag Plattformen im Kontext des vernetzten Automobils. Basis bilden die Kriterien Analyseebene, Kundengruppe, Nutzenversprechen, Akteure, Grad der Offenheit und Eintrittsbarrieren für Komplementoren. Die konzipierte Klassifikation von Connected-Car-Plattformen strukturiert die bis dato unübersichtliche Branche anhand

[1] Universität Stuttgart, Institute for Business Administration, Dept. I, Keplerstraße 17, 70174 Stuttgart, micha.bosler@bwi.uni-stuttgart.de
[2] Giesecke+Devrient advance52 GmbH, Rosenheimer Staße 145e, 81671 München, mail@christopher-jud.de
[3] Universität Stuttgart, Institute for Business Administration, Dept. VIII, Keplerstraße 17, 70174 Stuttgart, georg.herzwurm@bwi.uni-stuttgart.de

doi:10.18420/se2019-32

generalisierter Idealtypen und erklärt die unterschiedlichen Konstrukte sowie deren Zusammenspiel im übergeordneten Ökosystem der vernetzten Automobile.

Der Beitrag schließt mit einer Zusammenfassung und Bewertung der erhobenen Erkenntnisse und diskutiert Implikationen für Theorie und Praxis.

2 Zusammenfassung der Ergebnisse

Der Trend hin zum vernetzten Automobil sorgt für eine hohe Dynamik in der Automobilbranche. OEMs, etablierte Zulieferer und ehemals branchenfremde Akteure unternehmen Anstrengungen, um plattformbasierte Services im Kontext der vernetzten Fahrzeuge zu etablieren. Das schlägt sich in einer heterogenen, stark fragmentierten Landschaft nieder. Mit der aufgestellten Klassifikation werden die vier wichtigsten Plattformtypen identifiziert und charakterisiert: Plattformen der Automobilhersteller, unterstützende IT-Plattformen, Smartphone Integration, CC Platform as a Service.

Das zentrale Nutzenversprechen der *Connected-Car-Plattformen* der *Automobilhersteller* zielt darauf ab, dem Käufer eines vernetzten Automobils zusätzliche Dienste zu offerieren. Den OEMs bietet sich die Chance, durch die zur Connected-Car-Plattformen gehörenden Dienste mittels kostenpflichtiger Zusatzausstattung weitere Erlöse zu erwirtschaften. Die *unterstützenden IT-Plattformen* wiederum stellen die Infrastruktur bereit, die erforderlich ist, um die Connected-Car-Services der OEMs anzubieten. Aus Sicht der Automobilhersteller ergeben sich im Idealfall finanzielle als auch fachliche Vorteile. Die Betreiber der unterstützenden IT-Plattformen erzielen durch den Vertrieb ihre Lösung an mehrere Kunden Skalenerträge. Die Lösungen zur *Smartphone-Integration in Connected Cars* verfolgen das Ziel, dem Fahrer die Nutzung seines Mobiltelefons über die zentrale Head-Unit im Fahrzeug zu ermöglichen. Trotz des substituierenden Charakters weisen die Smartphone-Integrations-Produkte von Apple und Google eine sehr gute Penetration und Akzeptanz (bezogen auf die Verfügbarkeit in Fahrzeugen) bei den OEMs auf. Im Unterschied zu den oben diskutierten IT-Plattformen, geht es bei *Connected Car Platform as Service* nicht um die reine Unterstützung der Abwicklung und Absicherung der im Hintergrund ablaufenden Prozesse einer Plattform, sondern um den Fremdbezug der eigentlichen, für den Endkunden sichtbaren, Services. Gerade für Volumenhersteller oder kleine OEMs ergibt sich die Gelegenheit, anstelle der Entwicklung einer eigenen Connected-Car-Plattform sämtliche Services über einen Partner zu beziehen. Für alle vier Typen werden im Beitrag Anwendungsbeispiele betrachtet.

Für die OEMs gilt, dass, um die Entwicklungen im Bereich der Connected-Car-Plattformen aktiv mitgestalten zu können und nicht Gefahr zu laufen, von den Connected-Car-Angeboten von u.a. Apple und Google substituiert zu werden, müssen

die eigenen Plattformen Kunden einen erkennbaren Mehrwert liefern. Diesbezüglich darf sich die Strategie nicht darauf beschränken, Dienste auf der Plattform anzubieten, die lediglich existierende (Infotainment-)Funktionalitäten des Smartphones imitieren. Stattdessen müssen die OEMs die Innovationsintensität steigern, sich bewusst differenzieren und Services entwickeln, die andere Akteure mit ihren Plattformen nicht abdecken können. Die Industrie trägt dabei stets die Verantwortung, im Sinne der Verkehrssicherheit, die Ablenkung des Fahrers durch Connected-Car-Dienste zu vermeiden.

Durch die zu erwartende Weiterentwicklung der Plattformen und der Einführung neuer Dienste wird die Komplexität der zugehörigen Prozessabwicklung zwangsläufig ansteigen. Dadurch ergeben sich weitere Herausforderungen im Hinblick auf die Absicherung der vernetzten Fahrzeuge gegenüber Hackern und Schadsoftware. Die wahrgenommene Sicherheit der Connected-Car-Services wird deren künftige Akzeptanz erheblich beeinflussen – schließlich besitzt der Schutz der Insassen im Automobilbereich die höchste Priorität.

Stakeholderanalyse in plattformbasierten Ökosystemen für industrielle IoT-Plattformen

Dimitri Petrik[1], Georg Herzwurm[2]

Abstract: Um die aktuellen Herausforderungen des globalen Wettbewerbs zu adressieren, gewinnen Plattformansätze im Maschinenbau zunehmend an Bedeutung und gelten als eine Lösung zur Erbringung wertschöpfender digitaler Services. Dabei finden offene Softwareplattformen zunehmend Verbreitung, die auch als industrielle Internet of Things (iIoT)-Plattformen bekannt sind. Offene iIoT-Plattformen bilden die Grundlage für flexible Wertschöpfungsnetzwerke und stellen die technologische Basis plattformbasierter Ökosysteme dar. Zur Etablierung der iIoT-Ökosystem und Generierung von Netzwerkeffekten, müssen Plattformanbieter die komplementären Drittanbieter und ihre Eigenschaften kennen. Der Artikel analysiert auf Basis qualitativer Experteninterviews in der Maschinenbaubranche unterschiedliche Typen komplementärer Stakeholder, ihre Eigenschaften und Kooperationsstrukturen in iIoT-Ökosystemen für den Anwendungsfall datengetriebener Instandhaltung.

Keywords: Industrial IoT, IoT-Plattform, IoT-Ökosysteme, Plattformökosysteme, Stakeholderanalyse.

1 Einleitung

Die vorliegende Zusammenfassung stellt erstmalig die unterschiedlichen Typen komplementärer Stakeholder zur Erstellung digitaler plattformbasierter Services in der Maschinenbaubranche vor. Zur Erzeugung der Netzwerkeffekte einer Plattform ist die Anzahl der komplementären Drittanbieter (welche durch modulare Erweiterungen den Mehrwert der Plattform für die Endkunden erhöhen) und die Anzahl der Endkunden (welche im Gegenzug die Attraktivität der Plattform für die komplementären Drittanbieter erhöhen) entscheidend. Somit gelten Plattformen als mehrseitige Märkte und bei ausreichend offen gestalteten Schnittstellen der Plattform, entsteht ein komplementäres Drittanbieternetzwerk, das als plattformbasiertes-Ökosystem bezeichnet werden kann. Im Bereich des iIoT können plattformbasierte Ökosysteme auch als iIoT-Ökosysteme bezeichnet werden und stellen die Untersuchungsdomäne des Beitrags dar. Die Klassifikation der komplementären Stakeholder basiert auf den Ergebnissen der qualitativ-induktiven Interviewstudie mit den Unternehmensvertretern aus der Maschinenbaubranche. Die Datenerhebung basiert auf 17 Interviews zur Umsetzung von Predictive Maintenance (PdM) als digitaler Service, das mit Hilfe von iIoT-Plattformen erbracht wird. Somit gilt jedes Interview als eine Einzelfallstudie. Die Auswertung der

[1] Graduate School of Excellence advanced Manufacturing Engineering (GSaME), Cluster B2, Nobelstr. 12, 70569 Stuttgart, Dimitri.petrik@gsame.uni-stuttgart.de
[2] Universität Stuttgart, Lehrstuhl für ABWL und Wirtschaftsinformatik II, Keplerstr. 17, 70174 Stuttgart, georg.herzwurm@bwi.uni-stuttgart.de

doi:10.18420/se2019-33

erhobenen qualitativen Daten (Transkription, Kodierung, Labeln) ermöglicht es für den Anwendungsfall von PdM die komplementären Stakeholder, sowie ihre Eigenschaften und ihre Kooperationsstrukturen in der Maschinenbaubranche (aus der Perspektive des iIoT-Plattformanbieters) zu identifizieren.

2 Komplementäre Stakeholdertypen in iIoT-Ökosystemen

Die Tabelle fasst die Stakeholdertypen im iIoT für PdM als Anwendungsfall zusammen:

Stakeholder	Komplemente	Kooperiert mit
Maschinenbauer	Aus verschiedenen Komponenten bestehende Werkzeugmaschine Prozess- und domänen-spezifische Expertise	Komponentenanbieter, Automatisierungsanbieter, Softwareunternehmen, Datenanalyseanbieter
Komponenten-anbieter	Komponenten zur Überwachung Prozess- und domänen-spezifische Expertise	Maschinenbauer, Automatisierungstechnikanbieter, Softwareunternehmen, Datenanalyseanbieter
Automatisierungs-technikanbieter	Komponenten für die Maschinen-steuerung und Kommunikations-module Brücke zwischen Prozess und Plattformanbindung	Maschinenbauer Komponentenanbieter, Automatisierungstechnikanbieter, Softwareunternehmen, Datenanalyseanbieter
Retrofitanbieter	Vernetzung der Werkzeugmaschine beim Endkunden (Maschinenbetreiber)	Automatisierungstechnikanbieter, Softwareunternehmen, Endkunde
Software-unternehmen	Softwaremodule und Applikationen Expertise in der Softwareentwicklung und Plattformanbindung	Maschinenbauer, Komponentenanbieter, Automatisierungsanbieter
Datenanalyse-anbieter	Identifikation relevanter Datenquellen Expertise in der Datenauswertung und Datenanalyse	Maschinenbauer, Komponentenanbieter, Automatisierungsanbieter

Tab. 1: Komplementäre Stakeholdertypen und ihre Kooperationsstrukturen

Die Stakeholderanalyse liefert sowohl aus der Perspektive der Plattformforschung, als auch aus der Perspektive des iIoT einen wissenschaftlichen Beitrag. Zukünftige Forschungsarbeiten rund um Plattformen können die Ergebnisse und Adaption der Plattformgovernance für das hochaktuelle Feld des iIoT nutzen. Überwiegend technischen Fragestellungen adressierende iIoT-Forschung profitiert hingegen vom Beitrag zu der bisher wenig erforschten domänenspezifischen Ökosystemorganisation im iIoT. Zum anderen können die Ergebnisse fokalen Plattformanbietern helfen, in iIoT-Ökosystemen als komplexen mehrseitigen Märkten, die Plattform an die Vielfalt der komplementären Stakeholder anzupassen und weitere Komplementoren zu gewinnen.

Software Management in turbulenter Umwelt – Warum IT-Projekte agil werden müssen!

Wandel der Vorgehensmodelle im Zeitalter der digitalen Transformation

Eckhart Hanser[1]

Abstract: Der folgende Artikel wurde als Keynote zum 25-jährigen Geburtstag der Fachgruppe Vorgehensmodelle für die betriebliche Anwendungsentwicklung (WI-VM) der Gesellschaft für Informatik e.V. an der Tagung PVM2018[2] in Düsseldorf gehalten. Er beschäftigt sich mit dem Wandel von Vorgehensmodellen in Software-Projekten aufgrund der beginnenden Digitalisierung um die Jahrtausendwende und beschreibt die Entstehung von agilen und hybriden Modellen als Reaktion auf sich schnell ändernde Projekt- und Produkt-Anforderungen im Zeitalter der digitalen Transformation von Geschäftsmodellen.

Keywords: Agile und hybride Vorgehensmodelle, Digitalisierung und digitale Transformation, Scrum, Kanban, MAP, Systemische Ansätze.

Seit der Jahrtausendwende ist ein radikaler Umbruch in IT-Projekten zu beobachten: Die damals einsetzende „Digitale Transformation“ ermöglicht bzw. erzeugt erhebliche Veränderungen der Wirtschaft und der Gesellschaft durch den Einsatz neuer digitaler Technologien. Neue IT-Produkte entstehen, wie auch neue Geschäftsmodelle[3]. Im Bereich der Software-Entwicklung und des Managements von IT-Systemen ergibt sich das Problem der „Moving Targets“ [Ha10]: Meist entsteht erst im Verlauf des Projekts die finale Vision des Produkts. Ausführliche Spezifikationen gibt es nur noch selten. Bisher dominierende, „dokumentenlastige“ Vorgehensmodelle wie das „Wasserfall-Modell“ oder das V-Modell sind für solche Projekte ungeeignet.

Die Digitale Transformation kann bestehende Geschäftsmodelle zerschlagen[4]. Somit stellt sich die Frage: Ist die digitale Transformation auch für Vorgehensmodelle disruptiv? Ersetzen deswegen neuartige agile Modelle[5] die „schwergewichtigen“ Modellfamilien? Analysiert man 2 bekannte agile Modelle, stellt man fest: Das agile Vorgehensmodell Scrum ist klar disruptiv! Die neue Rolle des Product Owners, die Team-Rolle, die aus mehreren Entwicklern besteht, und der ScrumMaster ersetzen

[1] DHBW Lörrach, Kompetenzzentrum für agile IT-Prozesse (KAP) im Studienzentrum IT-Management & Informatik (SZI) (www.dhbw-loerrach.de/kap.html)
[2] Die PVM2018 fand vom 15.-16. Oktober 2018 an der FOM - Hochschule für Oekonomie und Management in Düsseldorf statt (siehe auch www.pvm-tagung.de).
[3] z.B. Suchmaschinen wie Google, Webshops wie Amazon, Social Media-Plattformen wie Facebook u.a.
[4] wie man an den Problemen des stationären Handels durch die Existenz des Online-Handels erkennen kann
[5] die zeitgleich mit der Digitalisierung entstanden

doi:10.18420/se2019-34

zusammen die klassische Rolle des Projektleiters. Sie gibt es nicht mehr! Etwas „dezenter“ zeigt sich die Disruption bei Software-Kanban[6]. Kanban kennt keine Sprints, dafür aber einen *kontinuierlichen* Entwicklungsprozess. Für viele „Kanban-Tickets“[7] wird Aufwand und Entwicklungszeit nicht mehr geschätzt, sondern nur noch eine durchschnittliche Durchlaufzeit gemessen. Eine präzise Aufwandschätzung entfällt (meistens), der „Fluss“ der Tickets steht im Vordergrund. Somit zeigt sich, dass die Digitale Transformation auch für Vorgehensmodelle disruptiv ist. Es entstehen Umbrüche durch sich schnell ändernde Anforderungen an Produkt und Team. Im Zeitalter der Digitalisierung *müssen* Vorgehensmodelle agil werden!

Regulatorische Einschränkungen oder behördliche Anforderungen lösen die Unternehmen durch individuelle "hybride" Modelle als Mischungen aus agilen und traditionellen Ansätzen. In der großangelegten Studie „HELENA-Survey“ 2017/2018, durchgeführt durch Mitglieder des GI-Fachausschusses Management der Anwendungsentwicklung und -wartung[8] und seiner Fachgruppen, wird dieser Trend bestätigt. Selbst vorsichtig betrachtet, werden agile Vorgehensmodelle bereits heute in mehr als der Hälfte aller Software-Projekte eingesetzt [Ku17].

Bei sich immer schneller ändernden Projektbedingungen stellt sich die Frage, ob Vorgehensmodelle – auch agile und hybride – überhaupt noch zeitgemäß sind. Braucht es im Projekt nicht vielmehr die „optimal zusammengesetzten“ Teams mit ihrem eigenen „agilen Werkzeugkasten“, aus dem sie die Praktiken und Techniken holen, die im aktuellen Projekt geplant sind? Einen Fingerzeig in die richtige Richtung geben die Systemischen Ansätze, die aus der Organisationspsychologie abgeleitet sind. Der 2010 vom Autor erstmals veröffentliche MAP-Ansatz[9] ist kein Vorgehensmodell mehr, sondern ein agiles Meta-Modell, in dessen Mittelpunkt die optimale „Komposition“ des Teams aus 6 „Typen“ von Team-Mitgliedern steht. Im Zeitalter der Digitalen Transformation ist nicht mehr das gewählte Vorgehensmodell entscheidend, sondern die Zusammensetzung des Teams und seine Fähigkeit, auf disruptive Anforderungen effizient und schnell zu reagieren. Im individuellen Teamprozess werden „reine“ agile Vorgehensmodelle wie Scrum und Kanban zu anpassbaren agilen Bausteinen auf einem problemorientierten Lösungsweg mit dem Ziel, selbst bei sich schnell ändernden Anforderungen das für die Kunden optimale IT-Produkt zu entwickeln.

Literaturverzeichnis

[Ha10] Hanser, E.: Agile Prozesse: Von XP über Scrum bis MAP. Springer, Heidelberg, 2010.

[Ku17] Kuhrmann M. et.al.: Hybrid Software and Systems Development in Practice: Waterfall, Scrum, and Beyond. In Proceedings of 2017 International Conference on Software and Systems Process, Paris, 2017 (ICSSP’17)

[6] David J Anderson 2011 (www.djaa.com)
[7] Arbeitspakete im Kanban-Projekt („Sticky Notes“)
[8] der auch die Tagung SWM2019 ausrichtet
[9] Meta Agile Process Model [Ha10], Kompetenzzentrum für agile IT-Prozesse, DHBW Lörrach (www.dhbw-loerrach.de/kap.html)

Designing an App that Promotes Sustainable Mobility

Agile and user-centered development of an app and corresponding business model

Andreas Helferich[1], Katharina Peine[2]

1 Zusammenfassung

In diesem Beitrag, der auf einer bestehenden Veröffentlichung der Autoren beruht und diese erweitert (myQommute — An App as Sustainable Mobility Concept – Autoren: K. Peine, A. Helferich, Proceedings der 2018 IEEE International Conference on Engineering, Technology and Innovation (ICE/ITMC)), wird die Konzeption einer App beschrieben, welche darauf abzielt, durch Pendler verursachte Staus zu verringern. Da eine solche App nur Erfolg haben kann, wenn sie von einer breiten Masse an Nutzern verwendet wird, sind die User Experience sowie das hinter der App stehende Geschäftsmodell von großer Bedeutung und wurden im Projekt intensiv betrachtet. Das Paper beschreibt insb. die Vorgehensweise, also den Prozess der Entwicklung der App sowie des Geschäftsmodells. Zudem werden die Funktionen der App und die Abgrenzung zu bestehenden Angeboten am Markt aufgezeigt.

2 Innovationsgrad des Beitrages

Der Innovationsgrad des Beitrags ist zweiteilig: einerseits wird eine neuartige App beschrieben, welche ein ambitioniertes Ziel verfolgt und in dieser Form bislang nicht am Markt verfügbar ist – insbesondere nicht als Produkt-Service-Bündel mit den entsprechenden Zusatzleistungen. Andererseits wird unseres Wissens erstmalig beschrieben, wie die User Experience sowie das Geschäftsmodell für eine derartige App methodisch stringent entwickelt wurden. Basierend auf laufenden Projekten und erfolgreichen Produkten in verwandten Märkten wurde auf Basis des Effectuation-Ansatzes überlegt, wie mit bestehenden Ressourcen (insb. bereits bestehender Software und bestehenden Kundenbeziehungen) sowie durch bewährte Partnerschaften ein neuartiges Produkt-Dienstleistungsbündel geschaffen werden kann. Dieses sollte insb. größere Organisationen, die sich als umweltbewusst und mitarbeiterfreundlich positionieren möchten und ein Problem sowie Problembewusstsein im Bereich der (Pendel-)Mobilität ihrer Mitarbeiter haben, bedienen und sich von bestehenden Angeboten abheben.

[1] highQ Professional Services GmbH, Seyfferstraße 34, 70197 Stuttgart, a.helferich@highQ.de
[2] highQ Computerlösungen GmbH, Schwimmbadstraße 26, 79100 Freiburg, k.peine@highQ.de

doi:10.18420/se2019-35

Der auf dieser Basis entwickelte Business Model Canvas kann aufgrund der strategischen Relevanz der App für das Unternehmen im Beitrag nicht abgebildet werden, wird aber in Auszügen im Rahmen der Präsentation vorgestellt. Abbildung 1 zeigt einige der während des Prozesses entstandenen Mockups für Screenshots.

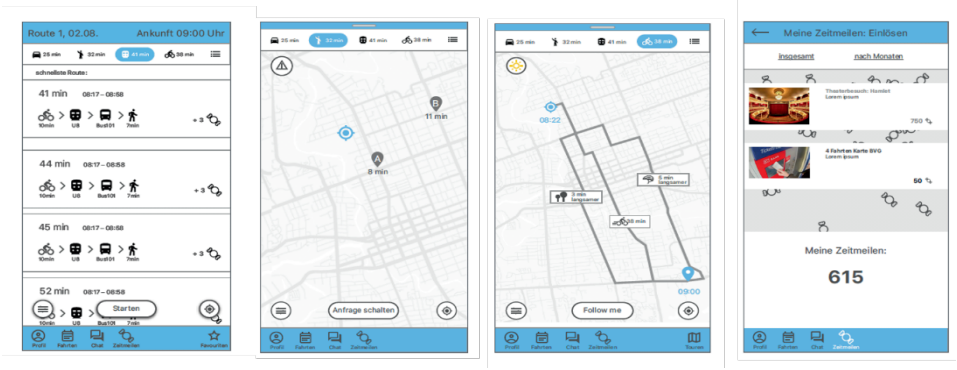


Abbildung 1: Mockups für Verbindungsauswahl, Mitfahrgelegenheiten und Incentivierung

3 Erkenntnisgewinn

Die Bedeutung von User Experience und Geschäftsmodell für die erstmalige, vor allem aber auch für die wiederholte Nutzung von Smartphone-Apps sind von hoher Relevanz für das Software-Produktmanagement. Auch beschreibt ein derartiger „Werkstattbericht“ zwar per se nur einen Einzelfall, die Vorgehensweise bietet allerdings Praktikern wie Wissenschaftlern Einblicke in ein hochaktuelles Feld. Praktiker können die Vorgehensweise mit ihrem eigenen Vorgehen vergleichen und an diesem ggf. Verbesserungen vornehmen, Wissenschaftler können anhand der Beschreibung mittels eines eher qualitativen Vorgehens erste mögliche Hypothesen generieren und weitere Fallstudien konzipieren.

4 Förderhinweis

Dieser Beitrag entstand im Rahmen des Forschungs- und Entwicklungsprojekt „Social Business:Digital - Digitale Soziale Netzwerke als Mittel zur Gestaltung attraktiver Arbeit“. Das Projekts wird im Rahmen des Programms „Zukunft der Arbeit" [FKZ 02L15A070, Laufzeit: 01.04.2017 bis 31.03.2020] vom Bundesministerium für Bildung und Forschung (BMBF) und dem Europäischen Sozialfonds (ESF) gefördert und vom Projektträger Karlsruhe (PTKA) betreut. Die Verantwortung für den Inhalt dieser Veröffentlichung liegt bei den Autoren. Die Autoren bedanken sich bei Fördermittelgeber, Projektträger und allen Projektpanrtnern für die Unterstützung.

Session 11: Cyber-physische Systeme

S. Becker, I. Bogicevic, G. Herzwurm, S. Wagner (Hrsg.): SE/SWM 2019,
Lecture Notes in Informatics (LNI), Gesellschaft für Informatik, Bonn 2019 121

Time-aware Test Execution Scheduling for Cyber-Physical Systems

Morten Mossige[1] Arnaud Gotlieb[2] Helge Spieker[2] Hein Meling[3] Mats Carlsson[4]

Abstract: The paper "Time-aware Test Execution Scheduling for Cyber-Physical Systems"first appeared in the application track of the 23rd International Conference on Principles and Practice of Constraint Programming (CP 2017).

Testing cyber-physical systems involves the execution of test cases on target-machines equipped with the latest release of a software control system. When testing industrial robots, it is common that the target machines need to share some common resources, e.g., costly hardware devices, and so there is a need to schedule test case execution on the target machines, accounting for these shared resources. With a large number of such tests executed on a regular basis, this scheduling becomes difficult to manage manually. In fact, with manual test execution planning and scheduling, some robots may remain unoccupied for long periods of time and some test cases may not be executed. We introduce TC-Sched, a time-aware method for automated test case execution scheduling. TC-Sched uses Constraint Programming to schedule tests to run on multiple machines constrained by the tests' access to shared resources, such as measurement or networking devices. We will further discuss challenges and requirements encountered when automating testing for industrial robots.

Keywords: Software Testing; Continuous Integration; Test Scheduling; Cyber-Physical Systems

Continuous integration (CI) aims to uncover defects in early stages of software development by frequently building, integrating, and testing software systems. When applied to the development of cyber-physical systems (CPS), which can simply be seen as communicating embedded software systems. The process may include running integration test cases involving real hardware components on different machines or machines equipped with specific devices. In the last decade, CI has been recognized as an effective process to improve software quality at reasonable costs [DMG07, St09, ERP14].

Different from traditional testing methods, running a test case in CI requires tight control over the *round-trip time*, that is, the time from when a source code change is committed until the success or failure of the build and test processes is reported back to the developer. Admittedly, the easiest way to minimize the round-trip time is simply to execute as many tests as possible in the shortest amount of time. But the achievable parallelism is limited by the availability of scarce global resources, such as a costly measurement instrument or

[1] ABB Robotics, Bryne, Norway & University of Stavanger, Stavanger, Norway morten.mossige@uis.no
[2] Simula Research Laboratory, Norway firstname@simula.no
[3] University of Stavanger, Stavanger, Norway hein.meling@uis.no
[4] RISE SICS, Kista, Sweden, mats.carlsson@ri.se

doi:10.18420/se2019-36

network device, and the compatible machines per test case, targeting different machine architecture and operating systems. These global resources are required in addition to the machine executing the test case and thereby require parallel adjustments of the schedule for multiple machines. Thus, computing an optimal test schedule with minimal round-trip time is a challenging optimization problem. Since different test cases have different execution times and may use different global resources that are locked during execution, finding an optimal schedule manually is mostly impossible. Nevertheless, manual scheduling still is state-of-the-practice in many industrial applications, besides simple heuristics. In general, successful approaches to scheduling use techniques from Constraint Programming (CP) and Operations Research (OR), additionally metaheuristics are able to provide good solutions to certain scheduling problems.

Informally, the optimal test scheduling problem (OTS) is to find an execution order and assignment of all test cases to machines. Each test case has to be executed once and no global resource can be used by two test cases at the same time. The objective is to minimize the overall test scheduling and test execution time. The assignment is constrained by the compatibility between test cases and machines, that is, each test case can only be executed on a subset of machines.

We introduce TC-Sched, a time-aware method to solve OTS. Using the Cumulatives [AB93, BC02] global constraint, we propose a cost-effective constraint optimization search technique. This method allows us to 1) automatically filter invalid test execution schedules, and 2) find among possible valid schedules, those that minimize the global test execution time (i.e., makespan). TC-Sched has been developed and deployed with ABB Robotics, Norway. An extensive experimental evaluation is conducted over test suites from industrial software systems, namely an integrated control system for industrial robots and a product line of video-conferencing systems. The primary goal in this paper is to demonstrate the scalability of the proposed approach for CI processes involving hundreds of test cases and tens of machines, which corresponds to a realistic development environment. Furthermore, we demonstrate the cost-effectiveness of integrating our approach within an actual CI process.

References

[AB93] Aggoun, A.; Beldiceanu, N.: Extending CHIP in Order to Solve Complex Scheduling and Placement Problems. Mathematical and Computer Modelling, 17(7):57–73, 1993.

[BC02] Beldiceanu, N.; Carlsson, M.: A New Multi-Resource Cumulatives Constraint With Negative Heights. In: CP 2002. volume 2470 of LNCS. Springer, pp. 63–79, 2002.

[DMG07] Duvall, P. M.; Matyas, S.; Glover, A.: Continuous Integration: Improving Software Quality and Reducing Risk. Pearson Education, 2007.

[ERP14] Elbaum, S.; Rothermel, G.; Penix, J.: Techniques for Improving Regression Testing in Continuous Integration Development Environments. In: FSE 2014. 2014.

[St09] Stolberg, S.: Enabling agile testing through continuous integration. In: AGILE 2009. IEEE, pp. 369–374, 2009.

Sichtenbasierte Kontextmodellierung für die Entwicklung kollaborativer cyber-physischer Systeme

Marian Daun[1], Bastian Tenbergen[2], Jennifer Brings[3], Patricia Aluko Obe[4]

Abstract: Dieser Vortrag berichtet von dem Beitrag *View-Centric Context Modeling to Foster the Engineering of Cyber-Physical System Networks* [Te18], der bei der *2018 IEEE International Conference on Software Architecture* vorgestellt und in dem Konferenzband veröffentlicht wurde.

Keywords: Cyber-physical systems; Context; View-based Architecture

1 Einleitung

Cyber-physische Systeme (CPS) schließen sich zur Laufzeit zu Systemverbünden zusammen, um einen Mehrwert zu erbringen, den das Einzelsystem nicht alleine erbringen kann. Die dynamische Natur dieser Systemverbünde, d.h. die Eigenschaft, ständig die Zusammensetzung des Systemverbunds zu ändern, führt dazu, dass sich auch der Kontext in einem ständigen Wandel befindet. Dies betrifft nicht nur den Kontext des einzelnen CPS, sondern auch den Kontext des Systemverbunds selbst. Folglich erschwert sich die Kontextbetrachtung zur Designzeit, da eine Vielzahl möglicher Systemverbundausprägungen Berücksichtigung finden muss. In dem Beitrag wird die Nutzung eines ISO/IEC/IEEE 42010-konformen Frameworks zur sichtenbezogenen Modellierung solcher dynamischer Systemverbünde bestehend aus CPS vorgeschlagen. Die Sichtenbildung erlaubt es u.a., den Fokus der Betrachtung auf eine einzelne Funktion zu legen. Hierbei können Funktionen spezifiziert und analysiert werden - unabhängig davon, von welchem konkreten System sie erbracht werden. Dies unterstützt die Entwicklung dahingehend, dass Situationen Berücksichtigung finden können, bei denen mehrere Systeme benötigt werden, um eine Funktionalität im Zusammenspiel des Systemverbunds zu erbringen. Fallstudienergebnisse verdeutlichen den Nutzen des Ansatzes.

[1] Universität Duisburg-Essen, paluno - The Ruhr Institute for Software Technology, 45217 Essen, Deutschland marian.daun@paluno.uni-due.de
[2] State University of New York, Oswego, New York, USA bastian.tenbergen@oswego.edu
[3] Universität Duisburg-Essen, paluno - The Ruhr Institute for Software Technology, 45217 Essen, Deutschland jennifer.brings@paluno.uni-due.de
[4] Universität Duisburg-Essen, paluno - The Ruhr Institute for Software Technology, 45217 Essen, Deutschland patricia.aluko-obe@paluno.uni-due.de

doi:10.18420/se2019-37

2 Dynamische Kontextsichten

Der Einsatz einer sichtenbasierten Betrachtung [Da16] des Kontexts, insbesondere die getrennte Beschreibung des operationellen Kontexts und des Wissenskontext [Da14], haben Vorteile bei der modellbasierten Spezifikation von CPS. Bei CPS, die in dynamischen Systemverbünden interagieren, ist ferner zu beachten, dass sich die Zugehörigkeit einzelner Systeme zu einem Systemverbund fortwährend ändern kann. Da die Funktionalität des Systemverbunds im Zusammenspiel der beteiligten CPS erbracht wird, muss eine Möglichkeit gegeben sein, die Funktionen des Systemverbunds vor dem Hintergrund der zum jeweiligen Zeitpunkt verfügbaren CPS zu beschreiben. Hierzu schlagen wir eine dynamische Sichtenbildung auf den Kontext vor, der jeweils auf eine spezifische Funktion des Systemverbunds fokussiert und hier die Auswirkungen von Änderungen im Systemverbund auf die Funktion spezifiziert.

Literaturverzeichnis

[Da14] Daun, Marian; Brings, Jennifer; Tenbergen, Bastian; Weyer, Thorsten: On the Model-based Documentation of Knowledge Sources in the Engineering of Embedded Systems. In: Gemeinsamer Tagungsband der Workshops der Tagung Software Engineering 2014, 25.-26. Februar 2014 in Kiel, Deutschland. S. 67–76, 2014.

[Da16] Daun, Marian; Tenbergen, Bastian; Brings, Jennifer; Weyer, Thorsten: SPES XT Context Modeling Framework. In: Advanced Model-Based Engineering of Embedded Systems, Extensions of the SPES 2020 Methodology, S. 43–57. 2016.

[Te18] Tenbergen, Bastian; Daun, Marian; Aluko Obe, Patricia; Brings, Jennifer: View-Centric Context Modeling to Foster the Engineering of Cyber-Physical System Networks. In: IEEE International Conference on Software Architecture, ICSA 2018, Seattle, WA, USA, April 30 - May 4, 2018. S. 206–216, 2018.

S. Becker, I. Bogicevic, G. Herzwurm, S. Wagner (Hrsg.): SE/SWM 2019,
Lecture Notes in Informatics (LNI), Gesellschaft für Informatik, Bonn 2019 125

A Framework for Self-adaptive Workflows in Cyber-physical Systems

Ronny Seiger[1] Steffen Huber[2] Peter Heisig[1] Uwe Aßmann[1]

Abstract: Workflows can be a useful means to formalize and enact processes among the sensors, actuators, smart objects and humans in Cyber-physical Systems (CPS). However, the dynamic nature of CPS and their resource constraint entities require the workflows and Workflow Management Systems (WfMSes) to be resilient and self-adaptive to deal with unanticipated situations and exceptions. We propose a generic framework based on the MAPE-K feedback loop to add self-management capabilities to WfMSes in the context of CPS. A case study in the smart home domain shows the general applicability of our framework for workflows and various WfMSes.

Keywords: Workflows; Cyber-physical Systems; Business Processes; Internet of Things

Motivation and Approach

The application of workflow and business process technologies to automate repetitive tasks and introduce high-level processes in the Internet of Things (IoT) and Cyber-physical Systems (CPS) has gained increasing attention over the years. With workflows, the data and control flow among sensors, actuators, embedded computers, smart objects, humans and other entities in CPS can be described and enacted on an abstract level facilitating simplified programming and flexible composition of existing functionality. However, due to the dynamic nature of CPS consisting of resource constraint devices, dynamically moving objects and physical entities that interact with the software controlled devices, formalized processes can only serve as basic structures for defining ordered sequences of activities.

We propose a framework and software component for adding the capability of self-adaptation based on the *MAPE-K* feedback loop to workflows and Worfklow Management Systems (WfMSes) [Se17]. The *Feedback Service* implements the MAPE-K loop and interacts with the WfMS (here: the *PROtEUS* WfMS for CPS [SHS16]) as well as sensors and actuators of CPS (cf. Fig. 1). Goals and objectives specify the expected outcome of the execution of a process step instance and error criteria referring to changes within specific context data. At runtime, the Feedback Service *Monitors* this data, *Analyzes* it with respect to the specified success and error criteria; *Plans* compensations in case of errors/exceptions; and *Executes* them [Se17]. Necessary data is contained in the *Knowledge Base*.

[1] Technische Universität Dresden, Software Technology, D-01062 Dresden, firstname.lastname@tu-dresden.de
[2] Technische Universität Dresden, Mechatronic Engineering, D-01062 Dresden, steffen.huber@tu-dresden.de

doi:10.18420/se2019-38

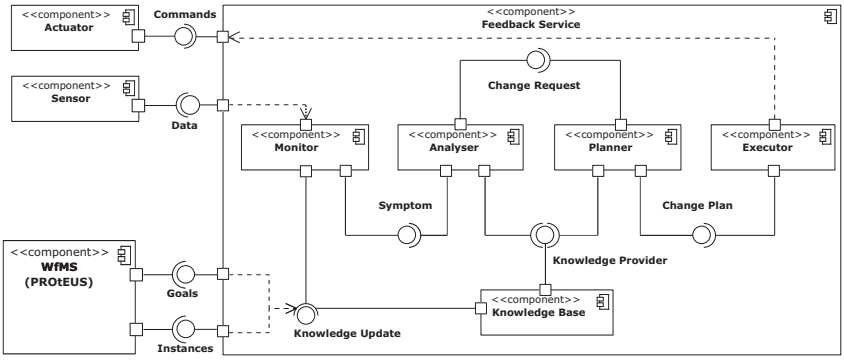


Fig. 1: Interactions of the Workflow Management System, Feedback Service and CPS Entities

To show the feasibility of our proof-of-concept implementation, we conducted a smart home case study. The example workflows include the automated assistance in case of a medical emergency [SHA18b], continuous light control [Se17], autonomous movement of service robots [SHA18b], and distributed execution of workflows on robots [SHA17].

The results show an increased success rate for error-prone processes when incorporating the MAPE-K feedback loop as part of the executions. The proposed framework for adding self-adaptation/-healing to workflows helps to increase the resilience of WfMSes as shown for the PROtEUS system [SHS16] but also as general retrofitting framework for other WfMSes [SHA18a]. Goals and objectives may refer to arbitrary performance or quality criteria and the strategies to find suitable compensations can be easily extended [Se17].

References

[Se17] Seiger, R.; Huber, S.; Heisig, P.; Aßmann, U.: Toward a framework for self-adaptive workflows in cyber-physical systems. Software & Systems Modeling/, 2017.

[SHA17] Seiger, R.; Herrmann, S.; Aßmann, U.: Self-healing for Distributed Workflows in the Internet of Things. In: IEEE International Conference on Software Architecture (ICSA) Workshops. 2017.

[SHA18a] Seiger, R.; Heisig, P.; Aßmann, U.: Retrofitting of Workflow Management Systems with Self-X Capabilities for Internet of Things. In: BP-Meet-IoT, Int. Conference on Business Process Management (BPM) Workshops. 2018.

[SHA18b] Seiger, R.; Huber, S.; Aßmann, U.: A Case Study for Workflow-based Automation in the Internet of Things. In: IEEE International Conference on Software Architecture (ICSA) Companion. 2018.

[SHS16] Seiger, R.; Huber, S.; Schlegel, T.: Toward an execution system for self-healing workflows in cyber-physical systems. Software & Systems Modeling/, 2016.

Session 12: Software Management II, Security und Technologietransfer

Checking Consistency and Completeness of Software Systems

Harry M. Sneed[1]

Abstract: The following paper presents the current state of the author's research on the subject of static software tracing, a research which dates back to the year 2000 when the author was requested to predict the costs of software maintenance projects. The goal of static tracing is to link the software artifacts produced in software development with the original requirements. These artifacts are linked by comparing the data names they use, i.e. their operands, with the nouns in the requirement text. Artifacts that have data names similar to those used in the requirement texts are considered to be implementations of those requirements.

Keywords: Requirement models, Design models, Code models, Test models, Model comparison, tracing requirements, linking software artifacts, consistency, completeness.

1 Static Checking of System Consistency

One of the goals of static analysis is to uncover consistency errors before a system is delivered. Of course the same errors may also be uncovered by extensive dynamic analysis. But dynamic analysis, i.e. testing, is expensive. It requires the setting up of a test environment and the provision of sufficient test data. In the case of large systems, the execution of several thousands of test cases may be necessary. Even then one cannot be sure that all the inconsistency errors will be found. Besides, there is another reason for checking consistency, namely to determine the degree of product completeness. A software product is only complete when all entities specified are also implemented. This can be ascertained by dynamic analysis if everything is tested but it can also be discovered by static analysis if the artifacts implemented at a lower level are matched against the artifacts specified at a higher level. [Chechik2001]

There are two schools of thought on how consistency should be judged

- the refinement school and
- the verification school

The refinement school requires only backward consistency checks since it assumes that the higher level system description is only a partial description of the lower levels, e.g.

[1] Technische Universität Dresden, Harry.Sneed@T-Online.de

doi:10.18420/se2019-39

the requirement specification only covers part of the design model and the code covers only a part of the design model. The point of step wise refinement is to add details as one progresses from one abstraction level to the next. The problem with this approach is that it will never be possible to verify the code against the specification since the specification is incomplete. Somewhere in the code there is a statement checking the control digit in a bank account number. It was not considered relevant enough to include in the design. Therefore if one is generating test cases from the design –model-based testing – there will be no test of this feature. Unless the tester interprets this as being relevant and adds a test case to cover it, it will never be tested. The test remains incomplete. The refinement approach leaves it to the developer and the tester to decide what is relevant and what is not. The only complete description of the system is the code. In testing one winds up testing the code against itself.

The verification school requires both backward and forward consistency checks since it assumes that the higher level system descriptions are at the same level of abstraction as the lowest level. What distinguishes them is the language. The highest level of description is the natural language followed by the design language and the programming language. There may also be a separate test language. Every last detail contained in the code, details like checking the control digit, appear in the design language. This makes it possible to verify the code against the design and the design against the requirement specification. The test can be fully automated without the need for human interpretation of what is relevant and what not, because the detailed rules are contained in the specification. Backward tracing becomes important. Therefore we must distinguish between entities declared at a lower level and those declared at a higher level. Lower level entities must not necessarily be present in the higher level artefact. This is a sign that the lower level artefact is not complete but it is not inconsistent. Higher level entities that are missing in the lower level artefacts show that the artefacts are not only inconsistent but that the lower level artefacts are incomplete. This is to be interpreted as an error [Lehn2013].

2 Matching Software Artifacts by Names

If development teams were to document the relationships between their artifacts from the very beginning, this matching would not be necessary. However, if that is not done the links between artifacts have to be established after the fact. Matching the names used in the artifacts is one way of achieving this. The basic hypothesis of the approach presented here is that artifacts which contain the same names are related. If a given requirement contains the noun “amount due” it is related to the class which uses the variable “AmountDue”. It can be assumed that this class implements that requirement. The more names that match the stronger that assumption becomes. If only one name matches the association is weak, but if two names match the association becomes stronger. If three names match one can be sure that the two artifacts are related.

Data names appear in all software artifacts. In the requirement texts they appear as nouns in the natural language sentences. In the design model they appear as method and attribute labels. In the code names appear as operands in procedural statements and as labels in data variable declarations. In the database schemas names appear as data field declarations and as keys for searching the data. In the test cases names appear as test data assignments and in test data conditions. Inconsistencies are located by comparing key entities in different artefacts. For instance the classes and variables defined in the design model should be found in the code and the use cases specified in the requirement document should be found in the design model.

3 The Name Matching Algorithm

The main problem in comparing models is that of matching the elements. Elements are identified by name and type. The starting point is the nouns in the requirement text or the change request [AnCD00]. They have to be recognized and collected as depicted in the following use case step description.

03; The <customer> should have the <possibility> of selecting <articles> to be ordered.

04; He must first enter his <customer_number>.

05; Then his <identity> and his <credibility> are checked.

Later the same or similar names are detected in the UML schema <ownedBehaviorxmi:type='uml:StateMachine' name='CustomerCredibility'>

As well as in the code:

```
public CustomerOrder(CustomerOrder cusOrder){
        this.OrderNumber = cusOrder.OrderNumber;
        this.Customer = cusOrder.Customer; }
```

If they can be matched then that particular entity is selected according to its type. Typical types are use cases, classes, methods, interfaces and attributes. Insofar as the types are uniform, i.e. the same types are used in the design as are used in the code and in the test, types can be easily matched. The type names may be a little different but they can be associated with one another. The problem is with the element names. They may differ. For instance, names of classes and attributes in the design model may be written in a different way in the code. Only if the code is generated from the design, can one be sure that they are the same. If the code is written by hand then it is almost for sure that they will not be the same. Thus an algorithm is needed for matching names according to rules.

The first rule of the algorithm is that only names of the same element type are compared. We only compare classes with classes and attributes with attributes.

The second rule is that plural nouns are converted to singular by leaving off the “s” or “es” at the end.

The third rule is that prefixes and suffixes are removed. We should only compare the word stem. Prefixes and suffixes are often used in the code to classify entities of a certain type. Suffixes serve the same purpose. To be able to math these adapted names with the original names the prefixes and suffixes have to be removed. We refer to this as “name stripping”.

The fourth rule is that concatenated names are split up into their surnames. For instance the name “ArticlePrice” is broken down into the two names: “Article” and “Price”. Each sub name is compared separately. If only one of the sub names matches to a name in the other table, the two elements are considered to be related. Experiments have demonstrated that a larger number of false positives may occur here, but it is better to have too many matches than it is to have too few. The human analyst can always delete the false positives.

The fifth rule is that names can be altered to match. If the matching tool recognizes that names almost match but not exactly, it can replace or delete certain characters to force a match. For instance if the word in the requirement document is “Record” and the name in the code “Rec”, the matching tool may only compare the first three characters. Here again there may be many false positives which the human analyst can delete.

Where ever at least one match occurs that entity is considered to be associated with the entity in the other table. At the end of the comparison the user will get a list of all entities for which no match occurred. These entities are then considered to be missing in the other model or document. Of course the human analyst will have to go thru the models to confirm that there entities are really missing, but the job of locating inconsistencies in the models is made much easier and quicker by using the automated comparison. It is also more thorough.

References

[Chechik2001] Chechik, M., Gannon, J.: Automatic Analysis of Consistency between Requirements and Designs. IEEE Transactions on Software Engineering, Vol. 27, Nr. 7, July 2001, p. 651

[Lehn13] Lehnert, S./Farooq, Q./Riebisch, M.: Rule-based Impact Analysis for heterogeneous Software Artifacts”, IEEE Proc.of CSMR2013, Genova, March 2013, p. 209

S. Becker, I. Bogicevic, G. Herzwurm, S. Wagner (Hrsg.): SE/SWM 2019,
Lecture Notes in Informatics (LNI), Gesellschaft für Informatik, Bonn 2019 133

How is security testing done in agile teams? A cross-case analysis of four software teams

Daniela Soares Cruzes [1], Michael Felderer[2], Tosin Daniel Oyetoyan[3], Matthias Gander[4], Irdin Pekaric[5]

Abstract: This summary refers to the paper 'How is security testing done in agile teams? A cross-case analysis of four software teams' [Cr17]. The paper was published as a full research paper in the proceedings of the 18th International Conference on Agile Software Development (XP 2017). It presents a multiple case study on how security testing is done in agile teams.

Keywords: Security testing; agile processes; agile testing; case study research; software testing; software processes; software security

1 Overview

In agile software development, there is a focus on the feature implementation and delivery of value to the customer. Therefore, non-functional aspects like security are often neglected. This holds for software constructions and even more for testing in agile teams. Security testing [Fe16a] can broadly be described as (1) the testing of security requirements that concerns confidentiality, integrity, availability, authentication, authorization, non-repudiation and (2) the testing of the software to validate how much it can withstand an attack. It is challenging for agile teams to systematically apply security testing in their development processes. There is in general a lack of systematic approaches and guidelines for agile security testing as well as of related empirical studies in real-world projects on agile security testing. The paper 'How is security testing done in agile teams? A cross-case analysis of four software teams' fills this gap and for the first time provides a multiple case study on security testing in agile projects based on four agile teams, two in Austria and two in Norway. We investigated how the security engineering process is managed/organized in agile teams, how security testing is performed in each testing phase, and how security testing techniques are generally used in the secure software development lifecycle. The main contribution of this paper is to deepen relevant knowledge and experience on the characterization of security testing in an agile context and to derive respective recommendations.

[1] SINTEF Digital, Trondheim, Norway danielac@sintef.no
[2] Universität Innsbruck, Innsbruck, Austria michael.felderer@uibk.ac.at
[3] SINTEF Digital, Trondheim, Norway tosin.oyetoyan@sintef.no
[4] Universität Innsbruck, Innsbruck, Austria matthias.gander@uibk.ac.at
[5] Universität Innsbruck, Innsbruck, Austria irdin.pekaric@uibk.ac.at

doi:10.18420/se2019-40

2 Results

The findings from investigating four agile teams show a lack of knowledge on security by agile teams in general, a large dependency on incidental penetration testers, and the ignorance of static testing of security. These are clear indicators that security testing is highly under-addressed and that more efforts should be invested for more proper security testing in agile teams. Although the study is based only on the insights of a limited amount of agile teams, we could derive recommendations for research and practice. Software engineering research can help to increase knowledge and application of security testing in several respects. First, knowledge can be increased by the development of suitable courses and guidelines based on empirical evidence showing which approaches work in which context. Then, with regard to model-based security testing [Fe16b], lightweight approaches are needed. Finally, also for penetration testing and security risk assessment suitable automation support and innovative techniques are required. In software development practice, there is a need to better use guidelines for secure coding and testing like from the OWASP. Within teams, there should be more systematic approaches of spreading knowledge in general and integrating static security analysis and penetration testing in particular. Furthermore, project owners should have more security awareness and take security issues into account when refining, prioritizing and validating the product backlog.

3 Conclusion

We summarized the paper 'How is security testing done in agile teams? A cross-case analysis of four software teams' [Cr17] that was published as a full research paper in the proceedings of the 18th International Conference on Agile Software Development (XP 2017). It presents a multiple case study on how security testing is done in agile teams. In the future, we plan to replicate this study and to develop and evaluate suitable security testing approaches to support the adoption of security testing in agile teams through action research studies with industry.

References

[Cr17] Cruzes, Daniela Soares; Felderer, Michael; Oyetoyan, Tosin Daniel; Gander, Matthias; Pekaric, Irdin: How is security testing done in agile teams? a cross-case analysis of four software teams. In: International Conference on Agile Software Development. Springer, pp. 201–216, 2017.

[Fe16a] Felderer, Michael; Büchler, Matthias; Johns, Martin; Brucker, Achim D; Breu, Ruth; Pretschner, Alexander: Security testing: A survey. In: Advances in Computers, volume 101, pp. 1–51. Elsevier, 2016.

[Fe16b] Felderer, Michael; Zech, Philipp; Breu, Ruth; Büchler, Matthias; Pretschner, Alexander: Model-based security testing: a taxonomy and systematic classification. Software Testing, Verification and Reliability, 26(2):119–148, 2016.

S. Becker, I. Bogicevic, G. Herzwurm, S. Wagner (Hrsg.): SE/SWM 2019,
Lecture Notes in Informatics (LNI), Gesellschaft für Informatik, Bonn 2019 135

Erfolgreicher Technologietransfer im Software Engineering

Transferansätze, Erfolgsfaktoren und Fallstricke

Marian Daun[1] Jennifer Brings[1] Kevin Keller[1] Sarah Brinckmann[1] Thorsten Weyer[1]

Abstract: Dieser Vortag berichtet von dem Beitrag *Approaches, success factors, and barriers for technology transfer in software engineering—Results of a systematic literature review* [Br18], der 2018 in der Fachzeitschrift *Journal of Software: Evolution and Process* veröffentlicht wurde.

Keywords: Software Engineering; Technologietransfer; Systematische Literaturrecherche

1 Einleitung

Technologietransfer zielt darauf ab, den Transfer von wissenschaftlichen Ergebnissen aus der Software Engineering-Forschung in die industrielle Praxis zu unterstützen. Im Rahmen einer systematischen Literaturrecherche wurde der aktuelle Stand der Forschung zum Thema Technologietransfer im Software Engineering analysiert. Die Ergebnisse zeigen die verschiedenen vorgeschlagenen Ansätze zum Technologietransfer auf und fassen die in der Literatur benannten Erfolgsfaktoren und Hindernisse für einen erfolgreichen Technologietransfer zusammen. Dies erlaubt es, Handlungsempfehlungen für einen erfolgreichen Technologietransfer abzuleiten. Die Ergebnisse zeigen u.a., dass eine Vielzahl von Ansätzen zum Transfer von Forschungsergebnissen des Software Engineering existiert. Von besonderer Bedeutung für einen erfolgreichen Technologietransfer im Software Engineering scheinen das Vorhandensein von empirischen Belegen für den Nutzen der Technologie und die Anpassbarkeit der Technologie zu sein. Als Hindernisse werden vor allem soziale und organisatorische Faktoren angesehen.

2 Ansätze zum Technologietransfer

Die meisten der im Rahmen der Studie identifizierten Ansätze zum Technologietransfer lassen sich in vier Ansatzkategorien zusammenfassen:

- *Kollaborationsansätze*. 33% der Lösungsansätze beschäftigen sich mit der Kollaboration zwischen Forschung und Industrie. Insbesondere werden Kollaborationsprojekte,

[1] Universität Duisburg-Essen, paluno - The Ruhr Institute for Software Technology, 45127 Essen, Deutschland
{marian.daun,jennifer.brings,kevin.keller,thorsten.weyer}@paluno.uni-due.de

doi:10.18420/se2019-41

Auftragsforschung und die Gründung von Instituten für den Technologietransfer vorgeschlagen.

- *Bildungsansätze*. 26% der Veröffentlichungen bieten als Lösungsansatz für den Technoligietransfer Lehr- und Lernansätze an. Hier sind zwei - im Wesentlichen gleich verbreitete - Richtungen zu unterscheiden. Zum einen werden Ansätze zum Training von Industriemitarbeitern vorgeschlagen, zum anderen existieren Ansätze, die einen langfristigen Technoligietransfer avisieren und hierzu eine Stärkung der universitären Lehre im Hinblick auf die Bedürfnisse der Industrie nahelegen.
- *Medien*. 21% der relevanten Veröffentlichungen unterbreiten den Vorschlag der Nutzung spezifischer Transfermedien. Hierzu zählen zunächst Guidelines und Informationsmedien wie Webseiten oder Repositorien.
- *Modell*. 14% der untersuchten Veröffentlichungen beschäftigen sich mit der Definition eines Modells, das die theoretischen Grundlagen des Technologietransfers zunächst besser erläutern soll, um diese Erkenntnisse langfristig nutzbar zu machen.

3 Erfolgsfaktoren und Fallstricke

Die Analyse der relevanten Veröffentlichungen - insbesondere der Erfahrungsberichte - zum Thema Technologietransfer erlauben die Identifikation von Erfolgsfaktoren, die einen gelungenen Technologietransfer unterstützen, und Fallstricken, die einen erfolgreichen Technologietransfer verhindern. In beiden Fällen ist die Gesamtheit des Technologietransfers zu betrachten. D.h., ein erfolgreicher Technologietransfer findet nur dann statt, wenn

- die *Technologie*, d.h. der zu transferrierende Ansatz, reif ist. D.h., u.a. die Technologie ist auf die Bedürfnisse der Industrie abgestimmt und ausreichend getestet und der Nutzen der Technolgie wurde nachgewiesen.
- der *Transferprozess* zielgerichtet ausgestaltet ist. D.h., die Kollaboration zwischen Industrie und Akademia ist klar geregelt. Die Industrie wirkt in der Definition des Problems und der Erstellung der Technoligie ausreichend mit, die Forschung steht bei der Einführung der Technologie im Unternehmen zur Verfügung.
- die *Organisation*, in die die Technologie transferiert werden soll, hierzu bereit ist. D.h., es gibt u.a. klare Verantwortlichkeiten, die Verantwortlichen sind auch mit ausreichend Verfügungsgewalt ausgestattet und die involvierten Mitarbeiter stehen dem Prozess positiv gegenüber.

Literaturverzeichnis

[Br18] Brings, Jennifer; Daun, Marian; Brinckmann, Sarah; Keller, Kevin; Weyer, Thorsten: Approaches, success factors, and barriers for technology transfer in software engineering - Results of a systematic literature review. Journal of Software: Evolution and Process, 30(11), 2018.

Session 13: Architektur und DSLs

Improving the Search for Architecture Knowledge in Online Developer Communities

Mohamed Soliman[1] Amr Rekaby Salama[2] Matthias Galster[3] Olaf Zimmermann[4]
Matthias Riebisch[5]

Abstract: When architecting a software system, software engineers search for architectural solutions (e.g. technologies), which fulfill certain requirements. Current approaches for architecture knowledge repositories facilitate learning about different architectural solutions. Nevertheless, the rapid and continuous increase of solution alternatives makes it challenging to manually acquire architecture knowledge and to ensure that this knowledge is up to date. Our goal in this paper is to improve how software engineers search for architecturally relevant information in online developer communities. We developed a new search approach for architecturally relevant information using Stack Overflow as an example of an online developer community. Our search approach differs from a conventional keyword-based search in that it considers semantic information of architecturally relevant concepts in Stack Overflow. We implemented the search approach as a web-based search engine. To show the effectiveness of the search approach compared to a conventional keyword-based search, we conducted an experiment with 16 practitioners. To ensure realism of the experiment, tasks given to practitioners are based on real scenarios identified in a separate interview study. The experiment showed that the new search approach significantly outperforms a conventional keyword-based search.

The full paper of this extended Abstract has been published in [So18].

Keywords: software architecture; architecture knowledge; search approach; online developer communities; Stack Overflow

Software engineers face recently unprecedented challenges to learn and keep up to date with new architectural solutions (e.g. technologies). Architecture knowledge (AK) repositories facilitate learning about architectural solutions. However, repositories require manually curating and maintaining AK, which becomes out of date quickly with the fast evolution of AK. Online developer communities (e.g. Stack Overflow) succeeded to persuade software engineers to share their knowledge about software issues, which encompass not only coding problems but also architectural issues [So16]. Nevertheless, the unstructured nature of text makes it arduous to search for architecturally relevant information in developer communities. Moreover, traditional keyword-based search engines return many irrelevant results when searching for architectural information [Go17].

[1] Universität Hamburg, SWK, Hamburg, Deutschland soliman@informatik.uni-hamburg.de
[2] Universität Hamburg, NATS, Hamburg, Deutschland salama@informatik.uni-hamburg.de
[3] University of Canterbury, Christchurch, New Zealand mgalster@ieee.org
[4] HSR Hochschule für Technik Rapperswil, Schweiz olaf.zimmermann@hsr.ch
[5] Universität Hamburg, SWK, Hamburg, Deutschland riebisch@informatik.uni-hamburg.de

doi:10.18420/se2019-42

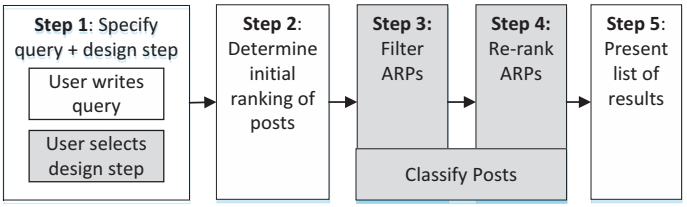


Abb. 1: Proposed search approach (gray boxes extend keyword-based search)

We propose a new search approach (see Abb. 1) to help finding architecturally relevant information on Stack Overflow. Steps 3 and 4 are our main contributions, which differentiate our enhanced search from a conventional keyword-based search. In step 3, we filter "architecture-relevant" posts from other types of posts. In step 4, we re-rank "architecture-relevant" posts based on their significance to support a certain architecture design step. To filter and re-rank posts in Stack Overflow, we developed a classification approach, which classify posts according to the type of discussed architectural solution and the purpose of a post [So16]. The classification approach combines two classifiers (Bag-of-Words and AK Ontology-based classification [SGR17]) using an ensemble learning. We conducted experiments using a corpus of Stack Overflow posts [So16] to evaluate the accuracy of classification. The average accuracy (F-Score) of the classification approach is 0.734. To compare the effectiveness of the proposed search approach to a conventional keyword-based search, we conducted an experiment with 16 practitioners. we asked participants to perform architecture design tasks and use both search engines to search for information that supports architectural design steps. Participants submitted a total of 422 queries. We calculated $Precision@k$ and $nDCG@k$ for both conventional and enhanced search approaches, with $@k_{1\rightarrow10}$. Our results show that our proposed enhanced search significantly improve the effectiveness of searching for architectural information in Stack Overflow.

Literatur

[Go17] Gorton, I.; Xu, R.; Yang, Y.; Liu, H.; Zheng, G.: Experiments in Curation: Towards Machine-Assisted Construction of Software Architecture Knowledge Bases. In: IEEE/IFIP ICSA 2017. S. 79–88, Apr. 2017.

[SGR17] Soliman, M.; Galster, M.; Riebisch, M.: Developing an Ontology for Architecture Knowledge from Developer Communities. In: IEEE/IFIP ICSA 2017. S. 89–92, Apr. 2017.

[So16] Soliman, M.; Galster, M.; Salama, A. R.; Riebisch, M.: Architectural Knowledge for Technology Decisions in Developer Communities: An Exploratory Study with StackOverflow. In: IEEE/IFIP WICSA 2016. S. 128–133, Apr. 2016.

[So18] Soliman, M.; Salama, A. R.; Galster, M.; Zimmermann, O.; Riebisch, M.: Improving the Search for Architecture Knowledge in Online Developer Communities. In: IEEE ICSA 2018. S. 186–18609, Apr. 2018.

S. Becker, I. Bogicevic, G. Herzwurm, S. Wagner (Hrsg.): SE/SWM 2019,
Lecture Notes in Informatics (LNI), Gesellschaft für Informatik, Bonn 2019 141

Developing and Evolving a DSL-based Approach for Runtime Monitoring of Systems of Systems

Rick Rabiser[1], Jürgen Thanhofer-Pilisch[1], Michael Vierhauser[2], Paul Grünbacher[3], Alexander Egyed[3]

Abstract: This is a summary of an article [Ra18] published in the Automated Software Engineering Journal in 2018 describing our experiences in developing and evolving a domain-specific language-based approach for runtime monitoring of systems of systems.

Keywords: Systems of systems; Requirements monitoring; Domain-specific languages; DSL evolution

1 Summary

Monitoring is needed to detect deviations from requirements in the context of complex software-intensive systems such as systems of systems [Ni15], because their behavior is only fully understandable during operation, when heterogeneous systems interact with each other and their environment. While many monitoring approaches exist [Ra17], they typically do not support dynamically defining and deploying diverse checks across multiple systems. In our article [Ra18], we report on our experiences of developing, applying, and evolving a domain-specific language (DSL)-based approach for monitoring a system of systems in the domain of industrial automation software. Specifically, we first developed a tool-supported approach to dynamically define and check constraints in systems of systems at runtime and later evolved this approach driven by requirements elicited in an industry collaboration project. In the article we describe how we evaluated the expressiveness and scalability of our DSL-based approach using an industrial system of systems and report the lessons we learned. For instance, while developing a DSL-based approach is a good solution to support industrial users, one must prepare the approach for evolution, e.g., support automated (re-)generation of tools and code after changes and automated testing.

[1] CDL MEVSS, Johannes Kepler University Linz, Altenberger Str. 69, 4040 Linz, Austria, rick.rabiser@jku.at
[2] University of Notre Dame, Computer Science and Engineering, Notre Dame, IN, USA, mvierhau@nd.edu
[3] ISSE, Johannes Kepler University Linz, Altenberger Str. 69, 4040 Linz, Austria, paul.gruenbacher@jku.at

doi:10.18420/se2019-43

Acknowledgements

The financial support by the Austrian Federal Ministry for Digital and Economic Affairs, the National Foundation for Research, Technology and Development, and Primetals Technologies is gratefully acknowledged.

References

[Ni15] Nielsen, Claus Ballegaard; Larsen, Peter Gorm; Fitzgerald, John; Woodcock, Jim; Peleska, Jan: Systems of Systems Engineering: Basic Concepts, Model-Based Techniques, and Research Directions. ACM Computing Surveys, 48(2):18:1–18:41, 2015.

[Ra17] Rabiser, Rick; Guinea, Sam; Vierhauser, Michael; Baresi, Luciano; Grünbacher, Paul: A Comparison Framework for Runtime Monitoring Approaches. Journal of Systems and Software, 125:309–321, 2017.

[Ra18] Rabiser, Rick; Thanhofer-Pilisch, Jürgen; Vierhauser, Michael; Grünbacher, Paul; Egyed, Alexander: Developing and evolving a DSL-based approach for runtime monitoring of systems of systems. Automated Software Engineering, 25(4):875–915, 2018.

S. Becker, I. Bogicevic, G. Herzwurm, S. Wagner (Hrsg.): SE/SWM 2019,
Lecture Notes in Informatics (LNI), Gesellschaft für Informatik, Bonn 2019 143

Using Language Workbenches and Domain-Specific Languages for Safety-critical Software Development

Markus Völter[1]

Abstract: In a 2018 article[2] in the journal on Software & Systems Modeling we[3] discussed the use of DSLs and language workbenches in the context of safety-critical software development. Language workbenches support the efficient creation, integration, and use of domain-specific languages. Typically, they execute models by code generation to programming language code. This can lead to increased productivity and higher quality. However, in safety-/mission-critical environments, generated code may not be considered trustworthy, because of the lack of trust in the generation mechanisms. This makes it harder to justify the use of language workbenches in such an environment. In the SOSYM paper, we demonstrate an approach to use such tools in critical environments. We argue that models created with domain-specific languages are easier to validate and that the additional risk resulting from the transformation to code can be mitigated by a suitably designed transformation and verification architecture. We validate the approach with an industrial case study from the healthcare domain. We also discuss the degree to which the approach is appropriate for critical software in space, automotive, and robotics systems.

Keywords: Domain-Specific Languages, Language Workbenches, Language Engineering, Safety-Critical Systems, Testing

1 Key Challenge

A language workbench (LWB) can be used to define domain-specific language (DSL) optimized for the application domain of a critical software component (CSC). Models are then used to describe one or more particular CSCs. The implementation of the CSC is automatically derived from the model. If the transformation to the executable code is correct, this leads to significant gains in productivity.

However, the approach can introduce additional failure modes because of errors in the LWB's generation framework or in the DSL-specific generators or interpreters.

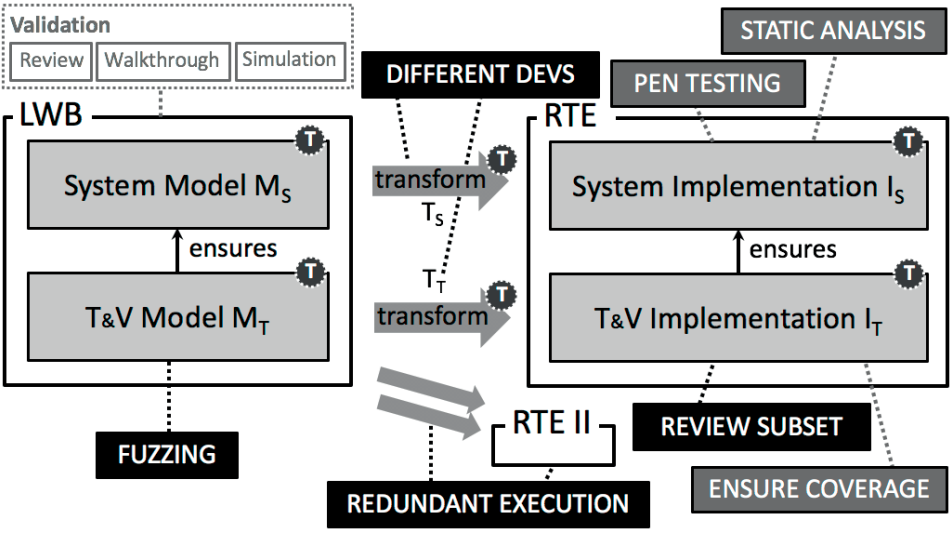
Correctness of the tool cannot be assumed because the LWBs and the DSLs cannot be argued to be qualified tools. So, how can a non-qualified LWB and custom-built DSLs be used in the development of critical systems?

[1] independent/itemis, Ötztaler Strasse 38, 70327 Stuttgart, Deutschland. voelter@acm.org

[2] https://link.springer.com/article/10.1007/s10270-018-0679-0

[3] coauthored by, Bernd Kolb, Klaus Birken , Federico Tomassetti, Patrick Alff, Laurent Wiart, Andreas Wortmann and Arne Nordmann

doi:10.18420/se2019-44



In effect, we are left with ensuring end-to-end correctness, from the model to the implementation; we treat the combination of model + DSL + generator + LWB as one "untrusted" black box. The challenge we address in this paper thus becomes:

How can a non-qualified LWB and custom-built DSLs be used in the development of critical systems, ensuring that the approach does not introduce faults into the CSC and continuing to exploit the benefits afforded by DSLs?

2 Solution

We express both the system and the tests or verification properties on model level and then translate both of them to the implementation and run them there. This way we *express and validate* the semantics on the convenient level of the model, but then *execute and verify* the semantics on the (ultimately relevant) implementation level.

We have performed a systematic analysis of the risks introduced by the DSL-based approach and use the techniques mentioned in the figure to address them.

3 Evaluation

We have evaluated the approach in a project in the healthcare industry, the paper contains a detailed case study. Voluntis developed a familiy of software medical devices. One successfully passed FDA pre-approval. Voluntis reported significant reductions in development and testing effort, as well as improvements in quality.

Session 14: Programmanalyse und Verifikation II

A Formal Framework for Incremental Model Slicing (Summary)

Gabriele Taentzer[1], Timo Kehrer[2], Christopher Pietsch[3], Udo Kelter[4]

Abstract: We report about a recently developed "Formal Framework for Incremental Model Slicing", published in [Ta18]. A model slice of a model is a submodel comprising a selected model part, called slicing criterion. In addition to classical use cases from the field of program understanding, model slicing is also motivated by specifying submodels of interest to be further processed more efficiently. Since slicing criteria are often modified during software development tasks, such slices often need to be updated. A slice update can be performed by creating the new slice from scratch or by incrementally updating the existing slice. We present a formal framework for defining model slicers that support incremental slice updates. This framework abstracts from the behavior of concrete slicers as well as from the concrete model modification approach. Incremental slice updates are shown to be equivalent to non-incremental ones. Furthermore, we present a framework instantiation based on the concept of edit scripts defining application sequences of model transformation rules, along with two two concrete model slicers implemented based on this instantiation.

Summary

Modeling frameworks such as the Eclipse Modeling Framework (EMF) do not scale beyond a few tens of thousands of model elements. Models of that size cannot even be edited in standard model editors. Thus, the *extraction of editable submodels from a larger model*, a specific form of model slicing, is the only viable solution to support an efficient yet independent editing of huge monolithic models [Pi17]. Adopting basic terminology from the field of program slicing as pioneered by Weiser [We81], a model slice is an interesting part of a model comprising a given slicing criterion, where it is up to a concrete slicing definition to specify which model parts are of interest.

Model slicing is faced with two challenging requirements which do not exist for traditional program slicers. First, the increasing importance and prevalence of domain-specific modeling languages (DSMLs) as well as a considerable number of different slicing specifications lead to a huge number of concrete slicers. Thus, methods for developing model slicers should abstract from a *slicer's concrete behavior*, and thus from a *concrete modeling language*, as far as possible. Second, slicing criteria are often modified during software development tasks since a slice created for an initial slicing criterion can turn out to be inappropriate.

[1] Philipps-Universität Marburg. taentzer@mathematik.uni-marburg.de
[2] Humboldt-Universität zu Berlin. timo.kehrer@informatik.hu-berlin.de
[3] Universität Siegen. cpietsch@informatik.uni-siegen.de
[4] Universität Siegen. kelter@informatik.uni-siegen.de

doi:10.18420/se2019-45

Rather than creating a new slice from scratch for a modified slicing criterion, slices must often be *updated incrementally*. This is indispensable for all use cases where slices are edited by developers since otherwise these slice edits would be blindly overwritten [Pi17]. In addition, incremental slice updating is a desirable feature when it is more efficient than creating the slice from scratch. To date, both requirements have been insufficiently addressed in the literature.

In this paper, we present a fundamental methodology for developing adaptable and incremental model slicers. To be independent of a concrete DSML and use cases, we restrict ourselves to static slicing in order to support both executable and non-executable models. Our framework is based on graph-based models and model modifications, and abstracts from the behavior of concrete slicers as well as from the concrete model modification approach. Within this framework, we show that incremental slice updates are equivalent to non-incremental ones. We present an instantiation of this formal framework where incremental model slicers are specified by model patches. Two concrete model slicers are implemented using EMF and the differencing and patching facilities of the model differencing framework SiLift [Si18].

The work presented in this paper has been conducted in terms of the research project MOCA [MO18], in which we develop fundamental methods, concepts and techniques supporting the version management of models. The extraction and updating of editable submodels supported through incremental model slicing is one of the key steps towards efficiently supporting the versioning of models based on a central repository and distributed workspaces. The repository hosts potentially huge models, while submodels thereof may be checked out into developer's workspaces in order to evolve certain parts of a model-based system.

Acknowledgment

This work was partially supported by the DFG (German Research Foundation) under the Priority Programme SPP1593: Design For Future - Managed Software Evolution.

References

[MO18] MOCA, http://www.dfg-spp1593.de/moca/index.html, 2018.

[Pi17] Pietsch, C.; Ohrndorf, M.; Kelter, U.; Kehrer, T.: Incrementally slicing editable submodels. In: Intl. Conf. on Automated Software Engineering. IEEE Press, 2017.

[Si18] SiLift, http://pi.informatik.uni-siegen.de/Projekte/SiLift, 2018.

[Ta18] Taentzer, G.; Kehrer, T.; Pietsch, C.; Kelter, U.: A Formal Framework for Incremental Model Slicing. In: International Conference on Fundamental Approaches to Software Engineering. Springer, pp. 3–20, 2018.

[We81] Weiser, M.: Program slicing. In: Intl. Conf. on Software Engineering. IEEE Press, 1981.

S. Becker, I. Bogicevic, G. Herzwurm, S. Wagner (Hrsg.): SE/SWM 2019,
Lecture Notes in Informatics (LNI), Gesellschaft für Informatik, Bonn 2019 151

Combining Verifiers in Conditional Model Checking via Reducers

Dirk Beyer[1] Marie-Christine Jakobs[2] Thomas Lemberger[3] Heike Wehrheim[4]

Abstract: Software verification received lots of attention in the past two decades. Nonetheless, it remains an extremely difficult problem. Some verification tasks cannot be solved automatically by any of today's verifiers. To still verify such tasks, one can combine the strengths of different verifiers. A promising approach to create combinations is conditional model checking (CMC). In CMC, the first verifier outputs a condition that describes the parts of the program state space that it successfully verified, and the next verifier uses that condition to steer its exploration towards the unverified state space. Despite the benefits of CMC, only few verifiers can handle conditions.

To overcome this problem, we propose an automatic plug-and-play extension for verifiers. Instead of modifying verifiers, we suggest to add a preprocessor: the reducer. The reducer takes the condition and the original program and computes a residual program that encodes the unverified state space in program code. We developed one such reducer and use it to integrate existing verifiers and test-case generators into the CMC process. Our experiments show that we can solve many additional verification tasks with this reducer-based construction.

Keywords: Conditional Model Checking, Testing, Software Verification, Sequential Combination

1 Overview

Automatic software verification received lots of attention in the past years: Many different approaches and verifiers were proposed, and all of them have different strengths, but also weaknesses. Thus, there still exist lots of programs that are in principle verifiable, but that no existing verifier can solve on its own. One solution to this problem is to combine the strength of different verifiers.

Conditional model checking (CMC) [Be12] is one promising combination approach. In CMC, the first verifier outputs a condition which summarizes the verifier's work, i.e., the state space that it successfully verified. If the first verifier fails to verify the complete state space, the condition and the program are passed to a second verifier, a so called conditional verifier. This conditional verifier uses that condition to restrict its verification

[1] LMU Munich, Institute of Informatics, Oettingenstraße 67, 80538 Munich, Germany
[2] LMU Munich, Institute of Informatics, Oettingenstraße 67, 80538 Munich, Germany
[3] LMU Munich, Institute of Informatics, Oettingenstraße 67, 80538 Munich, Germany
[4] Paderborn University, Department of Computer Science, Warburger Straße 100, 33098 Paderborn, Germany

doi:10.18420/se2019-46

to the unverified state space. Unfortunately, only few conditional verifiers exist and it is difficult to make a verifier understand and use conditions.

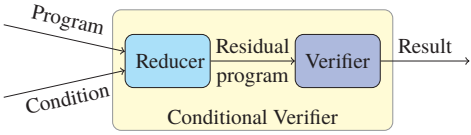


Fig. 1: Reducer-based conditional verifier template

Instead of adapting existing verifiers, we propose a reducer-based construction of conditional verifiers [Be18]. Our reducer-based construction uses the template shown in Fig. 1 to easily and automatically build a conditional verifier from an existing off-the-shelf verifier without changing the verifier itself. The idea is to add a preprocessor: the reducer. The reducer takes the condition and the original program to compute a *residual program* that encodes the unverified state space in a format that every verifier understands: program code.

In principle, many different reducers are possible. We implemented a reducer that uses a product construction: The program and the condition are converted to automata and the reduced product automaton is converted back to the residual program. We proved that the residual programs generated by this reducer neither miss any non-verified program path nor add program paths that do not occur in the original program. Hence, the reducer is reliable.

In our experiments, we instantiated the reducer-based construction template with the reducer implemented by us and six off-the-shelf verifiers; the best three verifiers from SV-COMP 2017[5] and three popular test-generation tools. Our evaluation on 5 687 verification tasks from a widely used software-verification benchmark[6] revealed that each of the created CMC combinations (1) solves additional tasks, (2) solves tasks that the corresponding sequential combination cannot solve, (3) solves tasks that were not solvable by any of the considered verifiers, and (4) solves tasks that only this specific CMC combination can solve. Summing up, we showed that reducer-based conditional model checking is effective, and to take full advantage of the approach one requires different conditional verifiers.

To solve difficult verification tasks, one needs to combine the strength of different verifiers, e.g., through CMC. The insight of our experiments is that we need different conditional verifiers for CMC to be effective and it is not worth the effort to modify existing verifiers to become conditional. Instead, our reducer-based construction of conditional verifiers allows us to easily derive k conditional verifiers from k arbitrary existing verifiers, without changing the implementation of the verifiers at all.

References

[Be12] Beyer, D.; Henzinger, T. A.; Keremoglu, M. E.; Wendler, P.: Conditional Model Checking: A Technique to Pass Information Between Verifiers. In: Proc. FSE. ACM, 2012.

[Be18] Beyer, D.; Jakobs, M.-C.; Lemberger, T.; Wehrheim, H.: Reducer-Based Construction of Conditional Verifiers. In: Proc. ICSE. ACM, pp. 1182–1193, 2018.

[5] https://sv-comp.sosy-lab.org/2017/

[6] https://github.com/sosy-lab/sv-benchmarks

Multi-Granular Conflict and Dependency Analysis in Software Engineering based on Graph Transformation (Summary)

Leen Lambers[1], Daniel Strüber[2], Gabriele Taentzer[3], Kristopher Born[4], Jevgenij Huebert[5]

We present a novel multi-granular static conflict and dependency analysis (CDA) of graph transformation as proposed in our ICSE 2018 (International Conference of Software Engineering) contribution [La18b].

Conflicts and dependencies are fundamental phenomena in software engineering. For example, when a software system is developed collaboratively, a change operation can facilitate or prohibit other change operations. In concurrent programming, conflicts may arise from data races when a thread writes to a memory location accessed by another thread. From unrecognized conflicts and dependencies, severe consequences may arise, ranging from productivity obstacles to fatal safety hazards. Therefore, there is a need for techniques to detect conflicts and dependencies automatically.

Graph transformation [Ro97] has been shown to be a versatile foundation for supporting conflict and dependency detection in software engineering, based on the following three principles: First, graphs are used for representing structures of interest, such as states of computation or versions of the system structure. Second, certain changes, such as state or structure modifications, are described using graph transformation rules. Third, the provided transformation specification is fed to the static *conflict and dependency analysis* (CDA) of graph transformations [Pl94, HKT02]: Given a set of transformation rules, all conflicts and dependencies arising from a given pair of rules are identified. A conflict arises, for example, if the first rule application deletes an element required by the second rule application. A key benefit of graph transformation is its mature formal foundation, which supports CDA techniques that are correct by design: all conflicts and dependencies can be detected.

Based on these principles, the CDA of graph transformations has enabled a large number of *use-cases in software engineering*, including analysis and design, model-driven engineering, and testing. For example, graph transformations can be used to model the execution behavior

[1] Universität Potsdam, Hasso-Plattner-Institut, Germany Leen.Lambers@hpi.de
[2] Chalmers University | University of Gothenburg, Sweden danstru@chalmers.se
[3] Philipps-Universität Marburg, Germany taentzer@mathematik.uni-marburg.de
[4] born@mathematik.uni-marburg.de
[5] huebert@mathematik.uni-marburg.de

doi:10.18420/se2019-47

of Java programs in terms of preconditions and effects on the object structure; identified conflicts and dependencies are then used for generating tests covering them. In model-based refactoring, graph transformations and CDA are used to find a suitable order of refactoring steps. In software product line engineering, feature interactions can be detected by specifying features as graph transformations and identifying conflicts and dependencies with CDA. We present a literature survey of 25 papers describing such use-cases and identify three key requirements for an improved CDA technique for software engineering: it shall be (i) *domain-independent* to be applicable to a large variety of software engineering domains, (ii) *usable* in the sense that it should display a reasonable amount of information to support understandability, and (iii) *efficient* when applied to software projects of realistic size.

To address these requirements, we present a novel static CDA technique for software engineering based on graph transformation. It builds on the notion of *granularity* of conflicts and dependencies introduced in [Bo17][6]. In particular, we provide an *efficient algorithm suite* for computing *binary, coarse-grained, and fine-grained* conflicts and dependencies: Binary granularity indicates the presence or absence of conflicts and dependencies, coarse focuses on root causes for conflicts and dependencies, and fine shows each conflict and dependency in full detail. In an experimental evaluation, our algorithm suite computes conflicts and dependencies rapidly. Finally, we present a *user study*, in which the participants found our coarse-grained results more understandable than the fine-grained ones reported in a state-of-the-art tool. In summary, we present a *multi-granular CDA technique based on graph transformation* achieving the same level of (i) domain-independence as the state of the art, while providing major (ii) understandability and (iii) performance improvements.

References

[Bo17] Born, Kristopher; Lambers, Leen; Strüber, Daniel; Taentzer, Gabriele: Granularity of Conflicts and Dependencies in Graph Transformation Systems. In: Graph Transformation, ICGT. pp. 125–141, 2017.

[HKT02] Heckel, Reiko; Küster, Jochen Malte; Taentzer, Gabriele: Confluence of Typed Attributed Graph Transformation Systems. In: ICGT. pp. 161–176, 2002.

[La18a] Lambers, Leen; Born, Kristopher; Kosiol, Jens; Strüber, Daniel; Taentzer, Gabriele: Granularity of conflicts and dependencies in graph transformation systems: A two-dimensional approach. Journal of Logical and Algebraic Methods in Programming, 103:105 – 129, 2018.

[La18b] Lambers, Leen; Strüber, Daniel; Taentzer, Gabriele; Born, Kristopher; Huebert, Jevgenij: Multi-granular Conflict and Dependency Analysis in Software Engineering Based on Graph Transformation. In: Proceedings of the 40th International Conference on Software Engineering. ICSE '18, ACM, New York, NY, USA, pp. 716–727, 2018.

[Pl94] Plump, Detlef: Critical Pairs in Term Graph Rewriting. In: Mathematical Foundations of Computer Science. volume 841, pp. 556–566, 1994.

[Ro97] Rozenberg, Grzegorz, ed. Handbook of Graph Grammars and Computing by Graph Transformations, Vol. 1: Foundations. World Scientific, 1997.

[6] The granularity notion has been refined in [La18a], partly based on the experiences gathered in [La18b].

Session 15: Erklärbare Software

S. Becker, I. Bogicevic, G. Herzwurm, S. Wagner (Hrsg.): SE/SWM 2019,
Lecture Notes in Informatics (LNI), Gesellschaft für Informatik, Bonn 2019 157

Testing Balancedness of ML Algorithms

Arnab Sharma[1], Heike Wehrheim[2]

Abstract: With the increased application of machine learning (ML) algorithms to decision-making processes, the question of *fairness* of such algorithms came into the focus. Fairness testing aims at checking whether a classifier as "learned" by an ML algorithm on some training data is biased in the sense of discriminating against some of the attributes (e.g. gender or age). Fairness testing thus targets the *prediction* phase in ML, not the learning phase.

In our approach, we investigate fairness for the *learning* phase. Our definition of fairness is based on the idea that the learner should treat all data in the training set equally, disregarding issues like names or orderings of features or orderings of data instances. We term this property *balanced data usage*. We have developed a (metamorphic) testing approach called TiLe for checking balanced data usage and report on some experiments of using TiLe to check classifiers from the `scikit-learn` library for balancedness.

1 Overview

In supervised machine learning, an ML algorithm is presented with a set of labelled training data, each consisting of a feature vector and a label (describing e.g. a class). From this, it learns a *predictive model* generalizing from the data as to later make predictions of labels for unseen data instances. A predictor is considered to be unfair if a change to the value of a feature in a data instance leads to a change in the prediction. Fairness testing [GYBM17] aims at checking such discrimination in ML predictors.

Complementary to that, we are interested in finding "unfairness" or "biasedness" in the learning phase. Basically, we aim at investigating whether all the data instances are used in the same way during training, and we formalize this using a number of metamorphic (mm) transformations [CCY98]: (a) permutation of training data instances (row permutation), (b) permutation of feature ordering (column permutation) and (c) shuffling of feature names. We consider an ML algorithm to be *balanced* when an application of mm-transformations on the training data does not change the predictive model learned.

For checking balancedness, we have implemented a framework called TiLe. TiLe contains a training data repository which consists of 4 artificial and 9 real-world datasets. Given an ML classifier as input, TiLe tests it for balancedness by training two predictors (per training data

[1] Universität Paderborn, Institut für Informatik, arnab.sharma@uni-paderborn.de
[2] Universität Paderborn, Institut für Informatik, wehrheim@upb.de

doi:10.18420/se2019-48

Tab. 1: Sensitivity to metamorphic transformations

Classifiers	*FN shuffling*	*Row perm.*	*Column perm.*
k-NN	✗	✓	✗
Decision Tree	✗	✓	✓
Naive Bayes	✗	✗	✗
SVM	✗	✓	✗
Neural Network	✗	✓	✓
Logistic Regression	✗	✓	✓
Random Forest	✗	✓	✓
Ada Boost	✗	✓	✓
Bagging Classifier	✗	✓	✓
Extra Trees	✗	✗	✓
Gradient Boosting	✗	✓	✓
Gaussian	✗	✗	✗
Stochastic Gradient	✗	✓	✗
Linear Regression	✗	✓	✓
Elastic Net	✗	✗	✓

and per mm-transformation): one with the original data and one with an mm-transformation applied on it. Afterwards, the predictors are checked for equality. Equality checking proceeds either by comparing the representations of the predictive models or by testing. A comparison of representations is only possible for some models (e.g. for support vector machines); often unequivalent representations define equal predictors. In a large number of cases we therefore employ testing, using different strategies for test case selection.

In our experiments, we used TiLe to check 13 classifiers from the `scikit-learn` library. Table 1 show the results of the experiments: none of the classifiers are sensitive to shuffling of feature names (i.e., feature names play no role for learning), but all are sensitive to either row or column permutation. Given that a software developer employing an off-the-shelf machine learning algorithm would expect it to learn some fixed predictive model when run on a given data set, this is a surprising result. To see whether our approach is also able to detect intentionally biased algorithms, we have furthermore applied it on 4 classifiers which are by design unbalanced (e.g. the ones of Zafar et al. [Za17]). All such unbalancedness was detected by TiLe.

References

[CCY98] Chen, T.Y.; Cheung, S.C.; Yiu, S.M.: Metamorphic testing: a new approach for generating next test cases. Technical report, Technical Report HKUST-CS98-01, 1998.

[GYBM17] Galhotra, S.; Y. Brun, Yuriy; Meliou, A.: Fairness testing: testing software for discrimination. In: ESEC/FSE. ACM, pp. 498–510, 2017.

[Za17] Zafar, M.B.; Valera, I.; Gomez-Rodriguez, M.; Gummadi, K.P.: Fairness Constraints: Mechanisms for Fair Classification. In: AISTATS. pp. 962–970, 2017.

Quality Assurance of Machine Learned Models by Integrating Domain Knowledge and Formal Verification

Rüdiger Ehlers[1] Jörg Grieser Christoph Knieke Andreas Rausch Mirco Schindler[2]

A large number of software systems nowadays employ artificial neural networks and other types of learned models from the area of machine learning. Such models are inferred from the available data and then used for classification and forecasting purposes.

While modern machine learning algorithms are highly scalable and able to deal with huge amounts of input data, they do not guarantee that the learned models capture the core aspects of the processes from which the data used for learning was acquired. Statistical approaches such as measuring the accuracy of a learned model on test data provide some insight into whether the model generalizes well, but the data set from which the model is learned is typically quite limited in size, which can impact its generalization capabilities and induces the need for quality assurance of learned models. Unlike in traditional software engineering, where code reviews and model-based testing are available as suitable quality assurance methods, learned models lack a structure that supports their manual review, and comprehensive specifications for exhaustive testing are seldom available.

So given that quality assurance is necessary, how can we interpret and explain the behavior of learned models? We argue in this talk that (1) the integration of *domain knowledge* into the learning process combined with (2) the employment of *formal verification* approaches to derive useful information about the learned model allows to explain the *behavior* of a model (as opposed to the model itself) to a large degree.

Let us discuss this idea by means of an example. Assume that a bank wants to predict whether a customer will default on a loan or not. There is a history of cases available in which for every customer, the number of years already working at the current employer is known in addition to the income per year. Additionally, it is known whether the customer defaulted, had only some delays paying the installments, or always paid the installments on time.

We can use a standard neural network architecture and a standard learning algorithm to compute a classifier from such a data set that maximizes the number of historical cases in which it is correctly predicted whether the customer defaulted or not (i.e., the credit worthiness). By doing so, we however ignore the possibility to integrate domain knowledge,

[1] Universität Bremen, Fachbereich für Mathematik & Informatik, Bibliotheksstraße 5, 28359 Bremen, ruediger.ehlers@uni-bremen.de
[2] Technische Universität Clausthal, Institut für Software and Systems Engineering, Arnold-Sommerfeld-Straße 1, 38678 Clausthal-Zellerfeld, Germany, <firstname>.<lastname>@tu-clausthal.de

doi:10.18420/se2019-49

namely that a higher yearly income should never decrease the credit worthiness. Hence, the classifier should be *monotone* in one of the input dimensions. For the number of years spent at the same employer, we may not have any insight in monotonicity, e.g., a very high number of years spent could decrease the probability of the employee finding work elsewhere in case his/her employer bankrupts. Hence, we only know that the learned model should be monotone in one dimension. Such a monotonicity requirement is relatively easy to integrate in modern *deep learning* toolkits, as we show in this talk.

Once we ensured that the learned model is partially monotone, it makes sense to ask questions such as "starting from which yearly income is a customer definitely classified in the best credit worthiness category when being with the same employer for the last 5-10 years" to gain insight into whether the learned classifier has reasonable behavior. While the question is also reasonable to ask without ensuring monotonicity, a monotone network is less likely to have outliers, i.e., small regions of the input space that are surrounded by a part of the input space that the network maps to a different classification. Such outliers may be small but make the answers to such queries less useful.

To obtain the answers to such questions on learned networks in applications that do not require frequent re-learning of the model, algorithms from the quickly growing area of *neural network verification* can be applied. They are currently limited to relatively small network sizes and hence have limited use for current computer vision applications. Yet, for many decision making applications, small networks often suffice and can already be analyzed by contemporary tools.

More challenging are systems that learn during operation. In this case, runtime monitoring is a promising approach. The runtime monitor verifies rules based on the domain knowledge, such as the existence of a pre-defined income threshold above which a customer is always seen as creditworthy. Due to the fact that the neural network can change its behavior almost arbitrarily, continuous verification is required. For a centralized system such as a loan determination system this is conceivable.

In case of restrictions such as limited computing resources, another approach can be followed. For instance, we can define a space of permitted network behavior such that verification is only triggered if an output leaves this space. In this way, we are able to verify behavioral properties and ensure that the network operates within reasonable limits.

We close the talk by giving an outlook on some additional challenges for quality assurance of systems involving learned models. For instance, in the currently running "*MaMMa : Maintained Mine and Machine*" project[3], we deal with the predictive maintenance problem for complex machines. Since maintenance is costly, the decision suggested by a learned model can only be followed if a reason is provided that is understandable by a machine engineer. As a consequence, we need to restrict the learning process to models that permit reason extraction.

[3] `https://mamma-project.eu`

Explaining Algorithmic Decisions with respect to Fairness

Qusai Ramadan[1] Amir Shayan Ahmadian[1] Jan Jürjens[1,2] Steffen Staab[1,3] Daniel Strüber[4]

Keywords: Software Fairness; Explainable Software; Model-Based Analysis; UML

1 Abstract

Decision-Making Software (D-MS) may exhibit biases against people on grounds of *protected characteristics* such as *gender* and *ethnicity*. Such undesirable behavior should not only be detected but also explained. To avoid complicated explanations and expensive fixes, fairness awareness has to be proactively embedded in the design phase of the system development. With *fairness by design*, system developers have to be supported with tools that detect and explain discriminations during the system architecture design [Ra18a].

Only avoiding protected characteristics in a D-MS does not prevent discrimination. Due to data correlations, other data may act as proxies for protected characteristics, thereby causing the so-called *discrimination by proxy* [GBM17]. There are two possible explanations for the correlations: **(1) Societal fact.** For instance, if females are more likely to have *long hair* more than males, then *long hair* can act as a proxy for the *gender*. **(2) Information flow.** The actual input of a D-MS may contain data that resulted from processing protected characteristics. For example, in an insurance company, it might be authorized to use the *gender* for identifying an *insurance tariff* but it might be not allowed to use *gender* for deciding about the *reimbursement factor*. However, if the *insurance tariff* is used as input to the reimbursement D-MS, a discrimination against *gender* have to be reported because the *insurance tariff* indirectly leaks a signal about the *gender* to the reimbursement D-MS.

Existing works only consider two approaches: *white-* (e.g., [Da17]) and *black-box* approaches (e.g., [GBM17]). While white box approaches can uncover discrimination, they cannot uncover discrimination in the above mentioned sense, as they did not consider possible information flow between system components. While black box approaches may solve this, they do not produce a witness to describe where and how a data flow can happen. Moreover, both approaches cannot be used in the early phase of the system design [Ra18a].

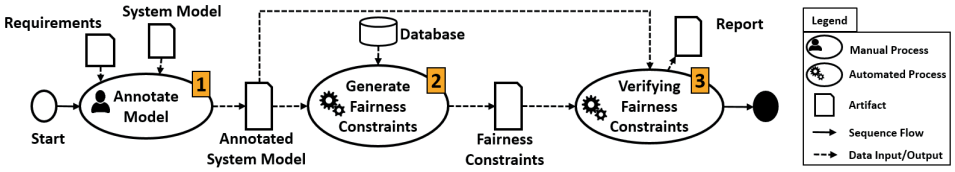
We aim to support developers with tools to reason about hidden flows for protected characteristics to a D-MS during the modeling of the system architecture. Detecting hidden information is a key challenge in security engineering [De76]. However, a model-based information flow analysis approach that supports fairness analysis is lacking [Ra18a]. We propose to develop a model-based discrimination analysis framework (see Fig. 1):

Input: (i) a *requirements document* containing fairness requirements. (ii) a model (UML) describing the structural and behavioral aspects of a system. (ii) a *database* of historical data. **Process:** (i) *Annotating a system model* with fairness requirements. For this, we plan to

[1] Universität Koblenz-Landau, Germany. Email: qramadan,ahmadian,juerjens,staab@uni-koblenz.de
[2] Fraunhofer Institute for Software and System Engineering, Germany
[3] University of Southampton, UK
[4] University of Gothenburg, Gothenburg, Sweden. Email:danstru@chalmers.se

doi:10.18420/se2019-50

Fig. 1: High-level overview of a model-based discrimination analysis framework.



extend the privacy UML profile in [Ah17b]. (iii) *Generating fairness constraints* provided as specifications, which have to be satisfied by a system model. In this step, proxies are also identified based on historical data from the system database. (iii) *Verifying the fairness constraints* of step (ii) against the system model (using a model checker). The last two tasks can be realized by extending CARiSMA, a tool that provides different security checks to support model-based security analysis [Ah17a]. **Output:** A witness reporting a violation. For instance, a sequence of actions that if executed will violate the fairness constraints.

Other open challenges. Information flow is not the only source for discriminations. Other sources are: First, a nonalignment between the organizational needs and the system models due to misunderstandings between expert stakeholders about fairness terminologies. This challenge can benefit from the model transformation technology, as proposed in our work in [Ra17]. Second, trade-offs between fairness and privacy requirements. A privacy requirement may disallow a D-MS from accessing protected characteristics. Although this requirement sounds as a fairness support, it prevents the D-MS from being able to uncover up-to-date proxies, as a proxy identification requires accessing to protected characteristics. For this, we plan to extend our work on conflicts detection [Ra18b].

Literatur

[Ah17a] Ahmadian, A. S.; Peldszus, S.; Ramadan, Q.; Jürjens, J.: Model-based privacy and security analysis with CARiSMA. In: Proceedings of the 2017 11th Joint Meeting on Foundations of Software Engineering. ACM, S. 989–993, 2017.

[Ah17b] Ahmadian, A. S.; Strüber, D.; Riediger, V.; Jürjens, J.: Model-based Privacy Analysis in Industrial Ecosystems. ECMFA, Springer/, 2017.

[Da17] Datta, A.; Fredrikson, M.; Ko, G.; Mardziel, P.; Sen, S.: Use Privacy in Data-Driven Systems. In: Proceedings of the ACM CCS Conference. 2017.

[De76] Denning, D. E.: A lattice model of secure information flow. Communications of the ACM 19/5, S. 236–243, 1976.

[GBM17] Galhotra, S.; Brun, Y.; Meliou, A.: Fairness testing: testing software for discrimination. In: Proceedings of the 2017 11th Joint Meeting on ESEC/FSE. ACM, S. 498–510, 2017.

[Ra17] Ramadan, Q.; Salnitri, M.; Strüber, D.; Jürjens, J.; Giorgini, P.: From Secure Business Process Modeling to Design-Level Security Verification. In: 20th ACM/IEEE MODELS 2017 International Conference. S. 123–133, 2017.

[Ra18a] Ramadan, Q.; Ahmadian, A. S.; Strüber, D.; Jürjens, J.; Staab, S.: Model-based discrimination analysis: a position paper. In: Proceedings of the International Workshop FairWare@ICSE 2018, Gothenburg, Sweden. 2018.

[Ra18b] Ramadan, Q.; Strüber, D.; Salnitri, M.; Riediger, V.; Jürjens, J.: Detecting Conflicts Between Data-Minimization and Security Requirements in Business Process Models. In: ECMFA 2018, Held as Part of STAF 2018, 2018, Proceedings. S. 179–198, 2018.

Poster

Understanding Parameters of Deductive Verification: An Empirical Investigation of KeY

Alexander Knüppel[1] Thomas Thüm[1] Carsten Immanuel Pardylla[1] Ina Schaefer[1]

Abstract: As formal verification of software systems is a complex task comprising many algorithms and heuristics, modern theorem provers offer numerous parameters that are to be selected by a user to control how a piece of software is verified. Evidently, the number of parameters even increases with each new release. One challenge is that default parameters are often insufficient to close proofs automatically and are not optimal in terms of verification effort. The verification phase becomes hardly accessible for non-experts, who typically must follow a time-consuming trial-and-error strategy to choose the right parameters even for trivial pieces of software. To aid users of deductive verification, we apply machine learning techniques to empirically investigate which parameters and combinations thereof impair or improve provability and verification effort. We exemplify our procedure on the deductive verification system KeY 2.6.1 and specified extracts of OpenJDK, and formulate 53 hypotheses of which only three have been rejected. We identified parameters that represent a trade-off between high provability and low verification effort, enabling the possibility to prioritize the selection of a parameter for either direction. Our insights give tool builders a better understanding of their control parameters and constitute a stepping stone towards automated deductive verification and better applicability of verification tools for non-experts.

Keywords: Deductive Verification, Design by Contract, Formal Methods, Theorem Proving, KeY, Control Parameters, Automated Reasoning

Overview

Software verification is vital for safety-critical and security-critical applications applied in industry. Although deductive verification is a promising static analysis technique that targets program verification directly on source code level, it has not yet found its way into industry due to issues with the scalability in specification and verification. Indeed, specifying large-scale software systems for efficient verification still demands high effort and expertise, which constitutes a problem for typical software developers who are not trained in proof theory.

Our long-term goal is to make deductive verification accessible for mainstream software developers. In this regard, one often overlooked hurdle that inexperienced users face is parameterization. While parameterization of formal method tools comes with the promise to ease the process of automatic verification, we exhibited that setting the right values for

[1] TU Braunschweig, Germany

doi:10.18420/se2019-51

the ever growing amount of parameters is challenging. Moreover, default parameters are often insufficient to close proofs automatically and are typically not optimal in terms of verification effort. Hence, in a recent publication, we empirically investigated the influence of parameters of KeY 2.6.1 [Ah16], a deductive verification system for Java source code.

The main results of our empirical study have been presented at the 9th International Conference on Interactive Theorem Proving (ITP'18) in Oxford, United Kingdom [Kn18b]. Based on our experience and observations combined with studying the online documentation of KeY, numerous publications, all tool tips in the KeY front-end, and the KeY book [Ah16], we formulated a total of 38 assumptions on how options in KeY improve or impair provability and verification effort. We derived a total of 53 statistical hypotheses and empirically measured the effect of different parameter configurations by employing significance tests and machine-learning techniques.

For our verification targets, we formally specified parts of OpenJDK's Collection API with JML [Kn18a]. By not only testing our hypotheses, but also by employing *SPL Conqueror* [Si12] for learning parameter-influence models, we were able to identify options of parameters that should be prioritized regarding their impact on verification effort. Moreover, we even identified parameters whose options represent a trade-off between provability and verification effort. Our insights provide valuable recommendations to users on which parameters to prioritize given a verification requirement. Moreover, tool builders can utilize our insights to improve on the user experience. For instance, implementing a recommendation system for parameters based on our investigation would help users to verify software more easily. Furthermore, KeY may hide insignificant parameters in specific verification scenarios or fine-tune parameters automatically during proofs. Although we focused on KeY, the problem of choosing sufficient parameters is not limited to deductive verification tools alone. Thus, our proposed approach for empirically studying the influence of parameters is applicable to arbitrary verification tools.

References

[Ah16] Ahrendt, Wolfgang; Beckert, Bernhard; Bubel, Richard; Hähnle, Reiner; Schmitt, Peter H; Ulbrich, Mattias: Deductive Software Verification–The KeY Book: From Theory to Practice. Springer, 2016.

[Kn18a] Knüppel, Alexander; I. Pardylla, Carsten; Thüm, Thomas; Schaefer, Ina: Experience Report on Formally Verifying Parts of OpenJDK's API with KeY. In: Proceedings of the Fourth Workshop on Formal Integrated Development Environment. Springer, 2018.

[Kn18b] Knüppel, Alexander; Thüm, Thomas; Pardylla, Carsten I.; Schaefer, Ina: Understanding Parameters of Deductive Verification: An Empirical Investigation of KeY. In: Proc. Int'l. Conf. Interactive Theorem Proving (ITP). Springer, 2018.

[Si12] Siegmund, Norbert; Rosenmüller, Marko; Kuhlemann, Martin; Kästner, Christian; Apel, Sven; Saake, Gunter: SPL Conqueror: Toward optimization of non-functional properties in software product lines. Software Quality Journal, 20(3-4):487–517, 2012.

Verifying Dynamic Architectures using Model Checking and Interactive Theorem Proving

Diego Marmsoler[1]

With the emergence of mobile and adaptive computing, dynamic architectures have become increasingly important. In such architectures, components can appear and disappear, and connections between their ports can change, both over time [Br14]. Thus, the state space of such architectures is changing dynamically, which makes their verification challenging.

To address this problem, we propose an approach based on *model checking* and *interactive theorem proving* (Fig. 1). To this end, verification is split into two parts: component types and component integration. The restricted state space of single component types makes them amenable to automatic verification techniques, such as model checking, and since their implementation changes frequently, they benefit most from the fast feedback provided by such techniques. The correctness of the integration of verified components, on the other hand, requires axiomatic reasoning and thus it is best done using interactive theorem proving. The additional effort induced by such techniques is justified by the robustness of verification results at the integration level: they remain valid as long as components fulfill the specification of their types.

To implement the approach, we developed FACTum: a framework for the axiomatic specification of dynamic architectures. In FACTum, data types are specified using *algebraic specification techniques*. Component types are then specified by means of *state machines* and associated *assertions* about their behavior in terms of first order LTL formulae. Finally, component integration is specified by means of *architectural assertions*: first order LTL formulae over component variables, using dedicated predicates to denote component activation and interconnection. *Architecture diagrams* complement these techniques with a graphical notation to specify interfaces and certain activation/connection constraints. A FACTum specification comes with a formal, denotational semantics in terms of sets of *architecture traces* [MG16a, MG16b]. To support the specification process, we implemented FACTum in Eclipse/EMF. FACTum Studio [MG18] supports the development of FACTum specifications with rigorous type checking mechanisms. Moreover, it allows to generate corresponding NuSMV models [MD17] and Isabelle/HOL theories [Ma18b] from a FACTum specification. To further support the interactive verification of component integration, we developed a calculus to reason about dynamic architectures [Ma17b, Ma17c] and implemented it in Isabelle/HOL [Ma17a, Ma18a].

[1] Technische Universität München, Fakultät für Informatik, Bolzmannstraße 3, Deutschland, diego.marmsoler@tum.de, https://marmsoler.com/

doi:10.18420/se2019-52

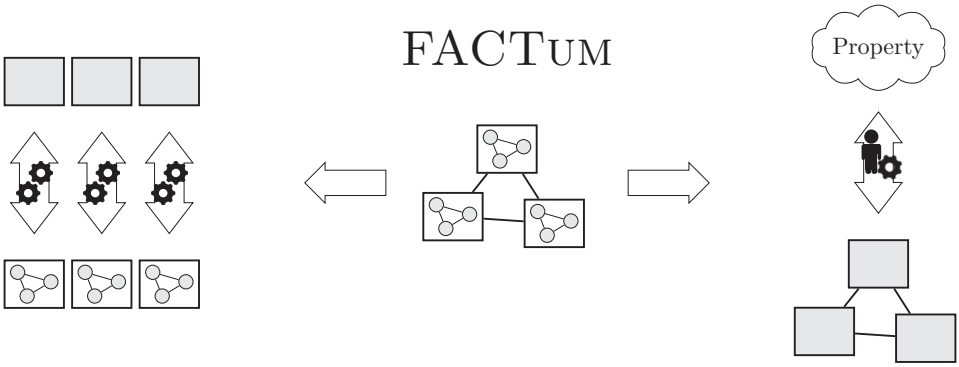


Abb. 1: Combined approach for the verification of dynamic architectures.

So far, we successfully applied the approach in four case studies [Ma18c]. First, we applied it to verify three well-known patterns for dynamic architectures: the *Singleton*, the *Publisher-Subscriber*, and the *Blackboard*. To demonstrate FACTUM's support for *hierarchical verification*, the Publisher-Subscriber pattern was modeled as an instance of the Singleton pattern and the Blackboard pattern as an instance of the Publisher-Subscriber. Thus, results obtained from the verification of lower-level patterns are automatically available to support the interactive verification of higher-level patterns. In another study, we applied the approach for the specification and verification of *Blockchain architectures*. The project consisted of roughly 3500 lines of Isabelle/HOL code, which demonstrates its feasibility for larger cases.

Keywords: Formal Methods, Dynamic Architectures, Interactive Theorem Proving, Model Checking, FACTUM

Literaturverzeichnis

[Br14] Broy, Manfred: A Model of Dynamic Systems. In (Bensalem, Saddek; Lakhneck, Yassine; Legay, Axel, Hrsg.): From Programs to Systems. The Systems Perspective in Computing, Jgg. 8415 in Lecture Notes in Computer Science, S. 39–53. Springer Berlin Heidelberg, 2014.

[Ma17a] Marmsoler, Diego: Dynamic Architectures. Archive of Formal Proofs, Juli 2017. `http://isa-afp.org/entries/DynamicArchitectures.html`, Formal proof development.

[Ma17b] Marmsoler, Diego: On the Semantics of Temporal Specifications of Component-Behavior for Dynamic Architectures. In: Eleventh International Symposium on Theoretical Aspects of Software Engineering. Springer, 2017.

[Ma17c] Marmsoler, Diego: Towards a Calculus for Dynamic Architectures. In (Hung, Dang Van; Kapur, Deepak, Hrsg.): Theoretical Aspects of Computing - ICTAC 2017 - 14th International Colloquium, Hanoi, Vietnam, October 23-27, 2017, Proceedings. Jgg. 10580 in Lecture Notes in Computer Science. Springer, S. 79–99, 2017.

[Ma18a] Marmsoler, Diego: A Framework for Interactive Verification of Architectural Design Patterns in Isabelle/HOL. In: The 20th International Conference on Formal Engineering Methods, ICFEM 2018, Proceedings. 2018.

[Ma18b] Marmsoler, Diego: Hierarchical Specication and Verication of Architecture Design Patterns. In: Fundamental Approaches to Software Engineering - 21th International Conference, FASE 2018, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2018, Thessaloniki, Greece, April 14-20, 2018, Proceedings. 2018.

[Ma18c] Marmsoler, Diego: A Theory of Architectural Design Patterns. Archive of Formal Proofs, März 2018. http://isa-afp.org/entries/Architectural_Design_Patterns.html, Formal proof development.

[MD17] Marmsoler, Diego; Degenhardt, Silvio: Verifying Patterns of Dynamic Architectures using Model Checking. In: Proceedings International Workshop on Formal Engineering approaches to Software Components and Architectures, FESCA@ETAPS 2017, Uppsala, Sweden, 22nd April 2017. S. 16–30, 2017.

[MG16a] Marmsoler, D.; Gleirscher, M.: On Activation, Connection, and Behavior in Dynamic Architectures. Scientific Annals of Computer Science, 26(2):187–248, 2016.

[MG16b] Marmsoler, Diego; Gleirscher, Mario: Specifying Properties of Dynamic Architectures using Configuration Traces. In: International Colloquium on Theoretical Aspects of Computing, S. 235–254. Springer, 2016.

[MG18] Marmsoler, Diego; Gidey, Habtom Kahsay: FACTUM Studio: A Tool for the Axiomatic Specification and Verification of Architectural Design Patterns. In: Formal Aspects of Component Software - FACS 2018 - 15th International Conference, Proceedings. 2018.

Metrics for Analyzing Variability and Its Implementation in Software Product Lines: A Systematic Literature Review

Sascha El-Sharkawy[1], Nozomi Yamagishi-Eichler[2], Klaus Schmid[3]

Abstract: This summary refers to the paper *Metrics for analyzing variability and its implementation in software product lines: A systematic literature review* [EYS19]. The paper was online first in 2018 and was finally published 2019 in the *Information and Software Technology (IST)* journal.

The use of metrics for assessing software products and their qualities is well established in traditional software engineering. However, such traditional metrics are typically not applicable to Software Product Line (SPL) engineering as they do not address variability management, a key part of product line engineering. Over time, various specialized product line metrics for SPLs have been described in literature, but no systematic description of these metrics and their characteristics is currently available.

This paper presents a systematic literature review, where we identify metrics explicitly designed for variability models, code artifacts, and metrics taking both kinds of artifacts into account. This captures the core of variability management for product lines. We discovered 42 relevant papers reporting 147 metrics intended to measure various aspects of variability models or code artifacts. We provide a categorization of these metrics and discuss problematic issues regarding the definition of the metrics. We also systematically assess the evaluation status of the metrics showing a current lack of high-quality evaluation in the field. Researchers and practitioners can benefit from the published catalog of variability-aware metrics and the assessment of their evaluation status.

Keywords: Software Product Lines; SPL; Metrics; Implementation; Systematic Literature Review

In software engineering, software metrics are an established approach to characterize properties of software [FB14]. However, such traditional metrics are typically not applicable to Software Product Line (SPL) engineering as they do not address variability management, a key part of product line engineering. Over time, various specialized product line metrics for SPLs have been described in literature, but no systematic description of these metrics and their characteristics is currently available.

In [EYS19], we present a systematic literature review to identify and characterize variability-aware metrics designed for the needs of SPLs. Our study aims at identifying existing metrics as a basis to draw qualitative conclusions on implementation properties of product lines. We include variability model metrics, because they are linked to all levels of product

[1] University of Hildesheim, Institute of Computer Science, Universitätsplatz 1, 31141 Hildesheim, Germany
elscha@sse.uni-hildesheim.de
[2] University of Hildesheim, Institute of Computer Science, Universitätsplatz 1, 31141 Hildesheim, Germany
[3] University of Hildesheim, Institute of Computer Science, Universitätsplatz 1, 31141 Hildesheim, Germany
schmid@sse.uni-hildesheim.de

doi:10.18420/se2019-53

line realization, including implementation. Thus, we focus only on variability model metrics and code metrics that take variation points into account to characterize product line implementation to answer the following research questions:

RQ1 Which metrics have been defined for variability models and implementation artifacts of SPLs?

RQ2 Which correlations between these measures and quality characteristics of product lines have been studied?

We discovered 42 peer-reviewed articles from the last decade to answer **RQ1**. We identified 57 variability model metrics, 34 annotation-based code metrics, 46 code metrics specific to composition-based implementation techniques, and 10 metrics integrating information from variability model and code artifacts. However, only 53 out of 147 metrics ($\approx 36\%$) have been evaluated in 14 out of 42 identified papers. This indicates that the product line community has only little knowledge regarding to what extend the existing metrics may be used to draw qualitative conclusions over the studied systems (**RQ2**).

The paper presents a description of the identified metrics, examples, and their intended purpose to provide practitioners and researchers with a catalog of the state of the art of variability-aware implementation metrics. Further, we discuss our key observations, which we made while surveying the literature. These are: There is only a *weak connection to established metrics* from traditional software engineering, the *weak use of related work* leads to redundant definitions of similar metrics, there are *ambiguous metric definitions* and *problematic evaluations* are used. Finally, we were surprised that there exist only a very *small number of metrics combining variability information* from multiple sources. We found only one empirical analysis evaluating such metrics combining the information from variability model and implementation artifacts, This study indicates a high usefulness of such a combined metric, encouraging further research along these lines.

Acknowledgments

This work is partially supported by the ITEA3 project REVaMP^2, funded by the BMBF (German Ministry of Research and Education) under grant 01IS16042H. Any opinions expressed herein are solely by the authors and not of the BMBF.

References

[EYS19] S. El-Sharkawy, N. Yamagishi-Eichler, K. Schmid. “Metrics for analyzing variability and its implementation in software product lines: A systematic literature review”. In: *Information and Software Technology* 106 (2019), pp. 1–30. ISSN: 0950-5849. DOI: 10.1016/j.infsof.2018.08.015.

[FB14] N. Fenton, J. Bieman. *Software metrics: a rigorous and practical approach*. CRC Press, 2014.

Integrating software quality models into risk-based testing

Harald Foidl[1], Michael Felderer[2]

Abstract: This summary refers to the paper 'Integrating software quality models into risk-based testing' [FF18]. The paper was published as an article in the Software Quality Journal. It shows for the first time how to integrate software quality models into risk-based testing.

Keywords: risk-based testing; software quality models; software testing; software risk management; software quality

1 Overview

Risk-based testing [FS14] is a frequently used testing approach which utilizes identified risks of a software system to provide decision support in all phases of the testing process. Risk assessment, which is a core activity of every risk-based testing process, is often done in an ad-hoc manual way. Software quality assessments based on quality models already describe the product-related risks of a whole software product and provide objective as well as automation-supported assessments. But so far, quality models have not been applied for risk assessment and risk-based testing in a systematic way. The paper 'Integrating software quality models into risk-based testing' fills this gap and investigates how the information and data of a quality assessment based on the established open quality model QuaMoCo [Wa12] can be integrated into risk-based testing. The paper presents two generic approaches how quality assessments based on quality models can be integrated into risk-based testing. In the first approach, a quality assessment based a quality model is conducted for each component (and the assessment results are then used to steer testing activities). In the second approach, metrics on the lowest level of the quality model hierarchy are directly used. The second approach implies one single quality assessment for a software product and further that the measured values of each metric and component are processed to an adequate probability factor for each component. Then, the concrete integration on the basis of the open quality model QuaMoCo and the second approach (because the metrics in the QuaMoCo quality model were calibrated by benchmarking whole software products) is performed in the paper. Finally, a case study is conducted based on five open source products.

[1] Universität Innsbruck, Institut für Informatik, Technikerstrasse 21a, 6020 Innsbruck, Austria harald.foidl@student.uibk.ac.at
[2] Universität Innsbruck, Institut für Informatik, Technikerstrasse 21a, 6020 Innsbruck, Austria michael.felderer@uibk.ac.at

doi:10.18420/se2019-54

2 Results

The case study of the developed integration approach based on five open source products showed that a risk-based testing strategy outperforms a lines of code-based testing strategy according to the number of classes which must be tested in order to find all defects. On average, all defects of the five analysed software products were found by testing 51.6% of all classes when a risk-based testing strategy was applied. In contrast, 63.8% of the classes had to be tested on average when a testing strategy based on the lines of code was applied. In addition, a significant positive relationship between the risk coefficient (impact factor assumed to be constant) and the associated number of defects of a class was found. Moreover, on average 80% of all defects of the five analysed software products were found by testing 30% of all classes when a risk-based testing strategy was applied.

3 Conclusion

We summarized the paper 'Integrating software quality models into risk-based testing' [FF18] that was published as an article in the Software Quality Journal. It shows for the first time how to integrate software quality models into risk-based testing. In the future, we plan to refine the integration of risk-based testing and quality models, to perform industrial case studies, and to improve tool support.

References

[FF18] Foidl, Harald; Felderer, Michael: Integrating software quality models into risk-based testing. Software Quality Journal, 26(2):809–847, 2018.

[FS14] Felderer, Michael; Schieferdecker, Ina: A taxonomy of risk-based testing. Software Quality Journal, 16(5):559–568, 2014.

[Wa12] Wagner, Stefan; Lochmann, Klaus; Heinemann, Lars; Kläs, Michael; Trendowicz, Adam; Plösch, Reinhold; Seidl, Andreas; Goeb, Andreas; Streit, Jonathan: The quamoco product quality modelling and assessment approach. In: Proceedings of the 34th international conference on software engineering. IEEE, pp. 1133–1142, 2012.

Comparison of the FMEA and STPA safety analysis methods–a case study

Sardar Muhammad Sulaman[1] Armin Beer[2] Michael Felderer[3] Martin Höst[4]

Abstract: This summary refers to the paper 'Comparison of the FMEA and STPA safety analysis methods–a case study' [Su17]. The paper was published as an article in the Software Quality Journal. It compares the Failure Mode and Effect Analysis (FMEA) and the System Theoretic Process Analysis (STPA) in an industrial case study.

Keywords: Hazard analysis; safety analysis; critical systems; risk management; failure mode and effect analysis; system theoretic process analysis; FMEA; STPA

1 Overview

As our society becomes more and more dependent on IT systems, failures of these systems can severely harm people and organizations. Diligently performing risk and hazard analysis helps to minimize the potential harm of IT system failures on individuals and the society and increases the probability of their undisturbed operation. Risk and hazard analysis is an important activity for the development and operation of critical software intensive systems, but the increased complexity and size puts additional requirements on the effectiveness of risk and hazard analysis methods. The paper presents a qualitative comparison of the two prominent hazard analysis methods Failure Mode and Effect Analysis (FMEA) [St03] and System Theoretic Process Analysis (STPA) [Le04] by applying the case study research methodology.

2 Results

To compare FMEA and STPA, both safety analysis methods been applied in a case study on the same forward collision avoidance system. Moreover, the analysis process of FMEA and STPA was also evaluated by applying qualitative criteria derived from the Technology Acceptance Model. It turned out that almost all types of hazards that were identified

[1] Lund University, Lund, Sweden sardar@cs.lth.se
[2] Beer Test Consulting, Baden, Austria armin.beer@bva.at
[3] Universität Innsbruck, Innsbruck, Austria michael.felderer@uibk.ac.at
[4] Lund University, Lund, Sweden martin.host@cs.lth.se

doi:10.18420/se2019-55

in the study were found by both methods. That is, both methods found hazards of type component interaction, software, component failure and system. With regard to component failure hazards, FMEA identified more component failure hazards than STPA. With regard to software hazards, STPA found more hazards than FMEA. With regard to component interaction hazards, STPA found some hazards, however, FMEA did not find any distinct hazards. Finally, with regard to system type error hazards, FMEA found slightly more hazards than STPA. Both FMEA and STPA consider system decomposition (FMEA decomposes and STPA considers whole system for analysis), identification of potential failures, their causes and effects, as well as definition of countermeasures. But STPA does not consider risk assessment in terms of risk priority number calculation and assignment of the application function to each subsystem. The methods have a different focus. FMEA especially takes the architecture and complexity of components into account, whereas STPA is stronger in finding causal factors of identified hazards. It can be concluded that, in this study, there was no hazard type that was not found by any of the methods. This means that it is not possible to point out any significant difference with respect to the identified hazard types. However, it can be observed that none of the methods in the study was effective enough to find all identified hazards, which means that they complemented each other well in that study.

3 Conclusion

We summarized the paper 'Comparison of the FMEA and STPA safety analysis methods–a case study' [Su17] that was published as an article in the Software Quality Journal. In the future, additional empirical studies (especially case studies and experiments) are needed in order to investigate differences, but also combinations of the methods and possible extensions of FMEA and STPA. In addition, safety has been defined as an important risk driver for testing, but the number of risk-based testing approaches taking safety analysis into account is limited. Comparing different safety analysis methods like FMEA and STPA with respect to test planning, design, execution and evaluation is another suggested topic for further research that could help to increase adoption of safety analysis methods for risk-based testing.

References

[Le04] Leveson, Nancy G: A systems-theoretic approach to safety in software-intensive systems. IEEE Transactions on Dependable and Secure computing, 1(1):66–86, 2004.

[St03] Stamatis, Dean H: Failure mode and effect analysis: FMEA from theory to execution. ASQ Quality Press, 2003.

[Su17] Sulaman, Sardar Muhammad; Beer, Armin; Felderer, Michael; Höst, Martin: Comparison of the FMEA and STPA safety analysis methods–a case study. Software Quality Journal, pp. 1–39, 2017. online first at https://doi.org/10.1007/s11219-017-9396-0.

S. Becker, I. Bogicevic, G. Herzwurm, S. Wagner (Hrsg.): SE/SWM 2019,
Lecture Notes in Informatics (LNI), Gesellschaft für Informatik, Bonn 2019 177

Architecture and Quality of Cloud Simulators

Zoltán Ádám Mann[1]

Abstract: Cloud simulators are complex programs that can simulate a cloud infrastructure and applications running on that infrastructure. Such simulators are often used to evaluate new algorithms for cloud resource management and software deployment optimization. However, the implementation of such algorithms in a cloud simulator is a challenging task that may lead to erosion of the architecture of the simulator, and even to faults in the implementation. Using appropriate abstractions, a clear separation of concerns can be achieved.

Keywords: Cloud computing; cloud simulator; architecture; software quality

1 Introduction

Modern cloud infrastructures offer the opportunity to deploy software components quickly in virtual machines that can be flexibly managed during operations. This allows for example automatic scaling of the resources used by the software, so that it can react automatically to workload changes, while other important metrics like energy consumption can be optimized [Ma15, BM15]. Cloud simulators are programs that can simulate a cloud infrastructure (e.g., servers, virtual machines) and applications running on that infrastructure. Such simulators are often used to test new algorithms for cloud resource management and to empirically compare different algorithms with each other [MS17]. The empirical evaluation of these algorithms is very important as their performance is typically not guaranteed theoretically. Simulation is the natural method to evaluate the performance of algorithms in many different system configurations. Cloud simulators have usually high complexity, must support extensibility, and allow experiments to be run efficiently. These expectations pose strict requirements on the architecture of cloud simulators.

2 Approach and results

In our original paper, the architecture of two cloud simulators (CloudSim and DISSECT-CF) is investigated, in particular regarding the extension of the simulators with new resource management algorithms and with experiments to evaluate the algorithms [Ma18]. Beside

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

doi:10.18420/se2019-56

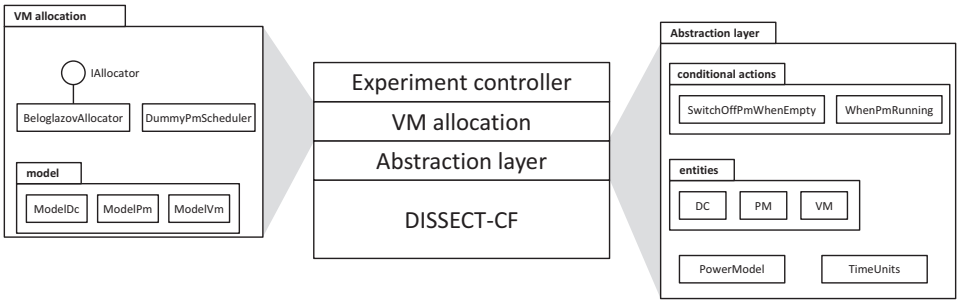


Fig. 1: Layered architecture featuring abstraction and decoupling

similarities and differences in the architecture of the two simulators, it is shown that a not sufficiently careful implementation of algorithms and experiments can lead to architecture erosion, which may result in unclear code and even faults. As a consequence, running the same algorithm in the two simulators with the same workload may lead to different results in metrics like overall energy consumption.

To avoid these problems, the paper shows how loose coupling and clearly defined interfaces between the simulator, the algorithms, and the experiments can be achieved by the introduction of additional abstractions (see Fig. 1). This way, a clear separation of concerns can be obtained, leading to code with improved quality. Furthermore, the paper demonstrates how the choice of simulator influences both the effort needed to implement an algorithm and the measured performance metrics. In particular, the API of CloudSim offered a higher level of abstraction than DISSECT-CF, but this could be bridged by extending DISSECT-CF with an appropriate abstraction layer. DISSECT-CF has a considerable advantage in execution time over CloudSim, even with the overhead added by the abstraction layer.

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

References

[BM15] Bartók, Dávid; Mann, Zoltán Ádám: A branch-and-bound approach to virtual machine placement. In: Proceedings of the 3rd HPI Cloud Symposium "Operating the Cloud". pp. 49–63, 2015.

[Ma15] 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.

[Ma18] Mann, Zoltán Ádám: Cloud simulators in the implementation and evaluation of virtual machine placement algorithms. Software: Practice and Experience, 48(7):1368–1389, 2018.

[MS17] Mann, Zoltán Adám; Szabó, Máté: Which is the best algorithm for virtual machine placement optimization? Concurrency and Computation: Practice and Experience, 29(10):e4083, 2017.

A framework for semi-automated co-evolution of security knowledge and system models (Summary)

Jens Bürger[1], Daniel Strüber[2], Stefan Gärtner[3], Thomas Ruhroth[4], Jan Jürjens[5,6], Kurt Schneider[7]

We present a summary of our article published in Elsevier's *Journal of Systems and Software* in 2018 [Bü18]. The presented approach has been developed in context of the SecVolution project, being part of the DFG SPP1593 *Design For Future*.

Security is an important and challenging quality aspect of software-intensive systems, and it becomes even more demanding in the case of long-living systems. Security issues do not necessarily arise from a flawed design, but can also manifest when the system fails to keep up with a changing environment, e.g., when a novel attack is discovered or a new law is passed. Thus, ongoing adaptations at system operation phase in response to security knowledge changes are inevitable.

We present a model-based framework for supporting the maintenance of security during the long-term evolution of a software system. It uses ontologies to manage the system-specific and the security knowledge. With model queries, graph transformation and differencing techniques, knowledge changes are analyzed and the system model is adapted.

We introduce the novel concept of *Security Maintenance Rules* to couple the evolution of security knowledge with co-evolutions of the system model.

To evaluate our technique, we used available community knowledge from various sources, including the Common Weakness Enumeration database (CWE). We demonstrate the framework by applying it to the *iTrust* system from the medical care domain and hence show the benefits of supporting co-evolution for maintaining security-critical systems.

The SecVolution approach is a holistic framework to deal with evolving knowledge in the environment of a software project. Existing approaches for secure software design fall

[1] University of Koblenz-Landau, Universitätsstraße 1, 56070 Koblenz, Germany buerger@uni-koblenz.de
[2] University of Koblenz-Landau, Universitätsstraße 1, 56070 Koblenz, Germany strueber@uni-koblenz.de
[3] adesso AG, Adessoplatz 1, 44269 Dortmund, Germany stefan.gaertner@adesso.de
[4] msg systems ag, Kruppstraße 82-100, 45145 Essen, Germany thomas.ruhroth@msg.group
[5] University of Koblenz-Landau, Universitätsstraße 1, 56070 Koblenz, Germany
[6] Fraunhofer ISST, Emil-Figge-Straße 91, 44227 Dortmund, Germany http://jan.jurjens.de
[7] Leibniz University Hannover, Welfengarten 1, 30167 Hannover, Germany kurt.schneider@inf.uni-hannover.de

doi:10.18420/se2019-57

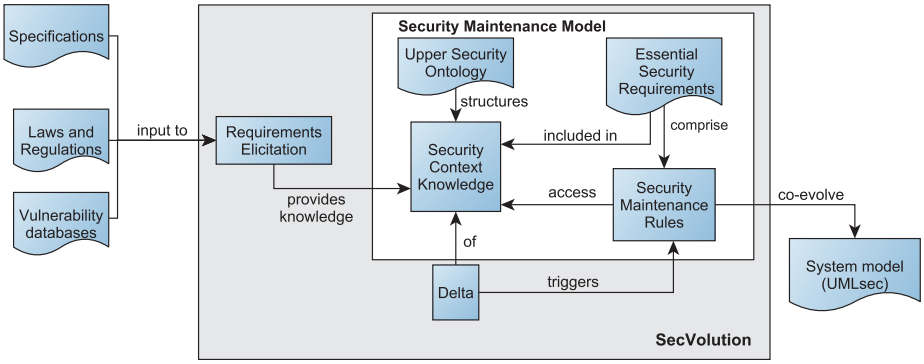


Fig. 1: Overview of the SecVolution approach

short on keeping up with environmental changes, thus providing only *one-shot* security. The overall goal is to restore security levels of an information system when changes in the environment put security at risk. At the beginning the software models are considered secure in accordance. As a triggering event the environmental knowledge or requirements concerning the software-project change are leveraged.

Figure 1 depicts an overview of the SecVolution approach in the publication's focus. Inputs are e.g. specification documents of various types. Laws and regulations provide knowledge about general security obligations. Vulnerability databases contain knowledge about security best practices and also known vulnerabilities in frameworks and algorithms, typically with appropriate mitigations.

Security-relevant knowledge is elicited and captured in an explicit representation called *Security Maintenance Model* (SMM). Security requirements that are defined on a coarse grained, *essential*, level, are used to define security requirements independent from their concrete technical realization. The security knowledge is based on a upper ontology for security notions we provide. Evolution of the security knowledge is captured as difference information which triggers execution of appropriate co-evolution actions, called *Security Maintenance Rules* (SMR). SecVolution focuses model-based software development and uses security-enriched UML models, built upon the UML security extension UMLsec. Thus, application of SecVolution leads to a co-evolved system model that is compliant to the evolved environment again.

References

[Bü18] Bürger, Jens; Strüber, Daniel; Gärtner, Stefan; Ruhroth, Thomas; Jürjens, Jan; Schneider, Kurt: A framework for semi-automated co-evolution of security knowledge and system models. Journal of Systems and Software, 139:142 – 160, 2018.

Serious Games for Software Refactoring

Thorsten Haendler[1], Gustaf Neumann[2]

Abstract:

This summary refers to the paper *Serious Refactoring Games* published as a full research paper in the proceedings of the 52nd Hawaii International Conference on System Sciences (HICSS 2019) [HN19].

Software design issues can severely impede software development and maintenance. Thus, it is important for the success of software projects that developers are aware of bad smells in code artifacts and improve their skills to reduce these issues via refactoring. However, software refactoring is a complex activity and involves multiple tasks and aspects. Therefore, imparting competences for identifying bad smells and refactoring code efficiently is challenging for software engineering education and training. The approaches proposed for teaching software refactoring in recent years mostly concentrate on small and artificial tasks and fall short in terms of higher level competences, such as analysis and evaluation. In this paper, we investigate the possibilities and challenges of designing serious games for software refactoring on real-world code artifacts. In particular, we propose a game design, where students can compete either against a predefined benchmark (technical debt) or against each other. In addition, we describe a lightweight architecture as the technical foundation for the game design that integrates pre-existing analysis tools such as test frameworks and software-quality analyzers. Finally, we provide an exemplary game scenario to illustrate the application of serious games in a learning setting.

Keywords: Serious Games; Refactoring; Software Engineering Education and Training.

Motivation and Overview

Software systems become increasingly complex while being maintained and extended over years. Issues in software design and architecture (such as bad smells) can negatively affect a systems' maintainability and extensibility. Removing these issues via refactoring is often neglected in practice due to the perceived difficulties, risks or lack of adequate tool support. Refactoring efficiently demands software developers competent and also motivated, which poses challenges for software engineering education and training. Existing approaches for training and teaching software refactoring fall short in addressing higher-level competences (e.g., analysis and evaluation), by mostly focusing on artificial tasks and small code examples.

[1] Institute for Information Systems and New Media, Vienna University of Economics and Business (WU), Austria
thorsten.haendler@wu.ac.at

[2] Institute for Information Systems and New Media, Vienna University of Economics and Business (WU), Austria
gustaf.neumann@wu.ac.at

doi:10.18420/se2019-58

Serious games represent a promising way to impart higher-level competences by simulating (or providing) real-world conditions, while including motivational (and fun) aspects.

In [HN19], we propose a game design and architecture for serious gaming in software refactoring applicable to real-world artifacts and under authentic conditions. For this purpose, we integrate development tools such as quality analyzers (measuring a system's technical debt as a score representing the person-hours to fix the debt items) as well as regression tests (ensuring the correct system behavior). The game design includes single-player and multi-player games modes (see Fig. 1), where players can either compete against a pre-defined benchmark (*technical debt score*) or against each other.

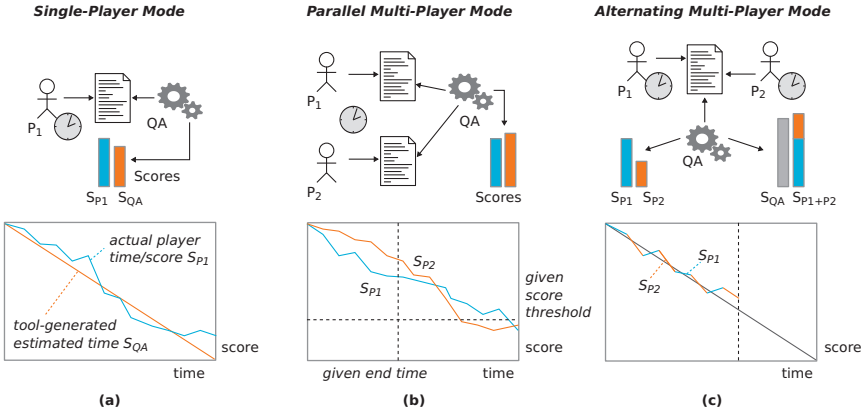


Fig. 1: Exemplary refactoring game modes with game constellations (top) and corresponding leader-boards with score-point progression (bottom) [HN19].

In addition, a lightweight game architecture is specified as a technical foundation for the game design. A game-play scenario illustrates the applicability. During game play, the players can demonstrate and consolidate the competences of identifying refactoring candidates and performing refactoring steps (*application*), analyzing code base and software design (*analysis*) as well as comparing refactoring options and strategies (*evaluation*).

The proposed approach is generic in the sense that it can be applied to any smell type (e.g., in source code, design/architecture, tests) provided that quality analyzers with corresponding smell metrics are available. We also believe that these refactoring games are not limited to training purposes, but could also be used for improving the quality of industry software. The code base can be analyzed and improved by a multitude of players, while each successful move is financially rewarded, similar to micro-task crowd-sourcing.

References

[HN19] Haendler, Thorsten; Neumann, Gustaf: Serious Refactoring Games. In: Proc. of the 52nd Hawaii International Conference on System Sciences (HICSS 2019), Software Engineering Education and Training Track, Hawaii, USA. 2019.

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

P-281 Hanno Langweg, Michael Meier, Bernhard C. Witt, Delphine Reinhardt (Hrsg.)
Sicherheit 2018
Sicherheit, Schutz und Zuverlässigkeit
25.–27. April 2018, Konstanz

P-282 Arslan Brömme, Christoph Busch, Antitza Dantcheva, Christian Rathgeb, Andreas Uhl (Eds.)
BIOSIG 2018
Proceedings of the 17th International Conference of the Biometrics Special Interest Group
26.–28. September 2018
Darmstadt, Germany

P-283 Paul Müller, Bernhard Neumair, Helmut Reiser, Gabi Dreo Rodosek (Hrsg.)
11. DFN-Forum Kommunikationstechnologien
27.–28. Juni 2018, Günzburg

P-284 Detlef Krömker, Ulrik Schroeder (Hrsg.)
DeLFI 2018 – Die 16. E-Learning Fachtagung Informatik
10.–12. September 2018, Frankfurt a. M.

P-285 Christian Czarnecki, Carsten Brockmann, Eldar Sultanow, Agnes Koschmider, Annika Selzer (Hrsg.)
Workshops der INFORMATIK 2018 - Architekturen, Prozesse, Sicherheit und Nachhaltigkeit
26.–27. September 2018, Berlin

P-286 Martin Mikusz, Alexander Volland, Martin Engstler, Masud Fazal-Baqaie, Eckhart Hanser, Oliver Linssen (Hrsg.)
Projektmanagement und Vorgehensmodelle 2018
Der Einfluss der Digitalisierung auf Projektmanagementmethoden und Entwicklungsprozesse
Düsseldorf 2018

P-287 A. Meyer-Aurich, M. Gandorfer, N. Barta, A. Gronauer, J. Kantelhardt, H. Floto (Hrsg.)
Informatik in der Land-, Forst- und Ernährungswirtschaft
Fokus: Digitalisierung für landwirtschaftliche Betriebe in kleinstrukturierten Regionen – ein Widerspruch in sich?
Referate der 39. GIL-Jahrestagung
18.–19. Februar 2019, Wien

P-291 Michael Räckers, Sebastian Halsbenning, Detlef Rätz, David Richter, Erich Schweighofer (Hrsg.)
Digitalisierung von Staat und Verwaltung
Gemeinsame Fachtagung Verwaltungsinformatik (FTVI) und Fachtagung Rechtsinformatik (FTRI) 2019
6.–7. März 2019 in Münster

P-292 Steffen Becker, Ivan Bogicevic,Georg Herzwurm, Stefan Wagner (Hrsg.)
Software Engineering and Software Management 2019
18.–22. Februar 2019 in Stuttgart

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 978-3-88579-686-2

This volume contains the contributions of the Software Engineering and Software Management (SE/SWM) 2019 conference held from 18.02. – 22.02.2019 in Stuttgart, Germany.
The SE/SWM proceedings contain extended abstracts from the scientific program, keynotes, and the workshop program.