GESELLSCHAFT
FÜR INFORMATIK

Michael Felderer, Wilhelm Hasselbring,
Rick Rabiser, Reiner Jung (Hrsg.)

Software Engineering 2020

Fachtagung des GI-Fachbereichs Softwaretechnik

24. – 28. Februar 2020
Innsbruck, Austria

Gesellschaft für Informatik e.V. (GI)

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

Volume P-300

ISBN 978-3-88579-694-7
ISSN 1617-5468

Volume Editors
Prof. Dr. Michael Felderer
Department of Computer Science
University of Innsbruck
Technikerstraße 21a, 6020 Innsbruck, Austria
michael.felderer@uibk.ac.at
Prof. Dr. Wilhelm (Willi) Hasselbring
Software Engineering Research Group
Department of Computer Science
Kiel University
Christian-Albrechts-Platz 4, 24118 Kiel, Germany
hasselbring@email.uni-kiel.de
Prof. Dr. Rick Rabiser
LIT CPS Lab
Johannes Kepler University Linz
Altenberger Str. 69, 4040 Linz, Austria
rick.rabiser@jku.at
Dr.-Ing. Reiner Jung
Software Engineering Research Group
Department of Computer Science
Kiel University
Christian-Albrechts-Platz 4, 24118 Kiel, Germany
reiner.jung@email.uni-kiel.de

printed by Köllen Druck+Verlag GmbH, Bonn

Vorwort

Herzlich willkommen zur Tagung Software Engineering 2020 (SE 20) des Fachbereichs Softwaretechnik der Gesellschaft für Informatik (GI). 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. Die Tagung richtet sich sowohl an Softwareentwickler und Softwareentwicklerinnen aus der Praxis, als auch an Forscherinnen und Forscher aus dem akademischen Umfeld. Software ist der wesentliche Bestandteil um aktuelle Herausforderungen in Wirtschaft und Gesellschaft zu meistern und auch weiterhin weltweit wettbewerbsfähige Produkte und Dienstleistungen anbieten zu können.

Die SE 20 bietet im wissenschaftlichen Hauptprogramm ein „Best-Of " der international in Fachzeitschriften und Konferenzen veröffentlichten Arbeiten deutschsprachiger Autoren. Sie umfasst eine große Bandbreite an Themen, die beispielsweise in der International Conference on Software Engineering (ICSE), den IEEE Transactions on Software Engineering (TSE), den ACM Transactions of Software Engineering and Methodology (TOSEM) und der ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering (ESEC/FSE) und vielen weiteren facheinschlägigen Fachzeitschriften und Konferenzen veröffentlicht wurden.

Wie bereits bei der SE 19 wurden eingereichte Papiere, die nicht als Vortrag berücksichtigt werden konnten, zu einem Poster eingeladen um eine noch größere Themenvielfalt für die SE zu erreichen.

Die SE 20 bietet neben dem wissenschaftlichen Hauptprogramm auch noch folgende weitere Tracks:

Forschungsmethoden im Software Engineering. Ziel ist es Forschende unterschiedlicher fachlicher Ausrichtungen, die sich insbesondere den Forschungsmethoden im Software Engineering verschrieben haben, zusammenzuführen, um den wissenschaftlichen Diskurs und den Erfahrungsaustausch anzuregen. Mit Blick auf das Programm der SE 20 ist dies jedenfalls gelungen.

SE FIT 20 – Software Engineering: Forschung – Innovation –Transfer. Ziel ist es, Akteure aus Praxis und Wissenschaft zusammenzubringen und so den Informations- und Erfahrungsaustausch zum Wissens- und Technologietransfer im Software Engineering zu forcieren. Auch dies ist ein wichtiger Aspekt des Software Engineering und wird durch die SE 20 abgedeckt.

Schließlich wird die SE 20 noch durch fünf im Rahmen der SE organisierte Events (vier Workshops und eine Konferenz) ergänzt, in denen weitere Themen im kleineren Kreis intensiv diskutiert werden:

SEUH 20 – Software Engineering im Unterricht der Hochschulen

ASE'20 – 17th Workshop on Automotive Software Engineering

AvioSE'20 – 2nd Workshop on Avionics Systems and Software Engineering

EMLS'20 – 7th Collaborative Workshop on Evolution and Maintenance of Long-Living Systems EMLS'20

AESP'20 – Anforderungsmanagement in Enterprise Systems Projekten

Wir danken allen, die zum Gelingen der Konferenz beigetragen haben, insbesondere den Autoren und den Gutachtern, den Keynote-Speakern, den Organisatoren der Workshops und Tracks, den Teilnehmern, den Sponsoren (Fabasoft, Dynatrace, Automated Software Testing, QAware, Siemens, ARZ, CQSE und msg) und Unterstützern (Land Tirol, Stadt Innsbruck, Universität Innsbruck im Allgemeinen und Forschungsgruppe Quality Engineering am Institut für Informatik im Speziellen, GI e.V. und OCG), den Medienpartnern (dpunkt.verlag und Sigs Datacom) und nicht zuletzt allen Helferinnen und Helfern vor Ort.

Innsbruck, im Februar 2020

Michael Felderer

Sponsoren

Platin Sponsor

Fabasoft AG
Honauerstraße 4
4020 Linz

Webseite
https://www.fabasoft.com

Gold Sponsoren

Dynatrace Austria GmbH
Am Fünfundzwanziger Turm 20
4020 Linz

Webseite
https://www.dynatrace.de

Automated Software Testing GmbH
Gewerberpark Urfahr 8
4040 Linz

Webseite
https://www.automated-software-testing.com

QAware GmbH München
Aschauer Straße 32
81549 München

Webseite
https://www.qaware.de

Silber Sponsor

Siemens AG Österreich
Siemensstraße 90
1210 Wien

Webseite
https://new.siemens.com

Bronze Sponsoren

ARZ GmbH
Tschamlerstraße 2
6020 Innsbruck

Webseite
https://www.arz.at

CQSE GmbH
Centa-Hafenbrädl-Straße 59
81249 München

Webseite
https://www.cqse.eu

msg systems ag
Robert-Bürkle-Straße 1
85737 Ismaning

Webseite
https://www.msg.group

Tagungsleitung

Gesamtleitung:	Michael Felderer, University of Innsbruck
Leitung des Programmkomitees:	Wilhelm Hasselbring, Kiel University Rick Rabiser, Johannes Kepler University Linz
Technologietransfer:	Stefan Sauer, Software Innovation Campus Paderborn Rudolf Ramler, Software Competence Center Hagenberg
SE Forschungsmethoden:	Sven Apel, Saarland University Daniel Mendez, Blekinge Institute of Technology
Workshops:	Regina Hebig, Chalmers University of Technology Robert Heinrich, Karlsruhe Institute of Technology
Lokale Organisation:	Ilona Zaremba, University of Innsbruck
Proceedings:	Reiner Jung, Kiel University

Programmkomitee

Sven Apel	Universität des Saarlandes
Stefan Biffl	TU Wien
Ruth Breu	Universität Innsbruck
Alexander Felfernig	Universität Graz
Martin Glinz	Universität Zürich
Lars Grunske	Humboldt-Universität zu Berlin
Anne Koziolek	KIT
Daniel Méndez	Blekinge Institute of Technology, Sweden, and fortiss GmbH, Germany
Barbara Paech	Universität Heidelberg
Martin Pinzger	Alpen-Adria-Universität Klagenfurt
Klaus Pohl	Universität Duisburg-Essen
Gunter Saake	Universität Magdeburg
Ina Schaefer	TU Braunschweig
Klaus Schmid	Universität Hildesheim
André van Hoorn	Universität Stuttgart
Manuel Wimmer	Johannes Kepler Universität Linz
Uwe Zdun	Universität Wien

Inhaltsverzeichnis

Keynotes

SE Prozesse

Domänen-spezifische Softwareentwicklung

Software Architektur, Design und Model-to-Code Mapping

Produktlinien und Variabilität

Agile Development

Feature-Modellierung

Wartung und Evolution 1

Software Intelligence und Enterprise Cloud

Models, MDE und MDD

Testing 1

Cloud, Edge und Microservices

Security

Wartung und Evolution 2

Testing 2

Performance und Benchmarking

Forschungsmethoden im Software Engineering

Software Engineering: Forschung — Innovation — Transfer

Software Engineering im Unterricht der Hochschulen

Workshops

Autorenverzeichnis

Keynotes

The R System: From Open Source to Open Science - An Insider's View

Achim Zeileis[1]

R (<https://www.R-project.org/>) is an open-source source software system and programming language that — along with Python — is among the most popular general data science tools. While R's popularity outside academia has also been growing rapidly, it is probably fair to say that the R user/developer community has been very actively involved in the broader open science movement. This encompasses various facets of openness, including among others:

- Classical open-source software written in and for R.
- Research software engineering aiming at developing, maintaining, and teaching software for research.
- Open-access publishing including dynamic documents generated from Markdown and/or LaTeX combined with R code.
- Reproducible research practices, sharing not only papers but also the underlying data, code, and (intermediate) results.
- Authoring and maintaining code, documentation, manuscripts, and empirical analyses using open version control tools.

Here, we trace how R evolved from the GNU system for statistical computing and graphics in the mid-1990s to today's broad and diverse ecosystem, providing many tools for open science including: literate programming, document preparation, package management, pipelines, or workflows. Already in the late 1990s and early 2000s initiatives within the R community emphasized the importance of (statistical) software being the potential principal outcome of scientific research, rather than a by-product of methodological or empirical work. This necessitated establishing suitable communication channels in conferences (like the *Directions in Statistical Computing* or *useR!* conferences) and journals (like the *Journal of Statistical Software* or *The R Journal*) along with a common understanding of good practices, review and evaluation criteria, standards for citations, etc.

[1] Universität Innsbruck, Austria Achim.Zeileis@uibk.ac.at

doi:10.18420/SE2020_01

For illustration, we highlight which quality criteria are important for successful publications in the Journal of Statistical Software (JSS). JSS fosters comprehensive open-source implementations of broad classes of statistical models which can be easily embedded into standard workflows. Strategies that typically help accomplishing this goal include:

- Adopting a design for the computational tools that reflects the features of the underlying conceptual methods.
- Employing an object-oriented programming paradigm with standard generic functions and methods.
- Reusing standard data structures and object classes.
- Providing modular building blocks.

Finally, some comparisons to other new publication and collaboration initiatives related to the R community are provided (e.g., *rOpenSci* or the *Journal of Open Source Software*) as well as an outlook on R's future.

Vom Wiegen allein wird die Sau nicht fett - Erfahrungen aus einem Jahrzehnt Qualitätsanalyse in Forschung und Praxis

Elmar Jürgens[1]

Ich habe vor 12 Jahren als Doktorand zum ersten Mal eigene statische Qualitätsanalysen in der Praxis durchgeführt. Mich hat damals fasziniert, welche Einblicke mir die Ergebnisse in die Entwicklungsprozesse und Probleme in erfolgreichen Organisationen erlaubten.

Oft haben sie mir jeodch gezeigt, dass die Vorstellung, die ich als Forscher von der Praxis hatte, bestenfalls unvollständig war. Als wir beispielsweise Clone Detection bei einem unserer ersten Industriepartner eingesetzt haben, haben wir hunderttausende Zeilen Code gefunden, die zwischen mehreren Abteilungen in dreifacher Kopie gepflegt wurden. Und bei denen nachweislich kritische Fehler nur teilweise behoben waren. Unser Vorschlag, hierfür gemeinsame Komponenten einzuführen, rief jedoch zu unserer Überraschung wenig Begeisterung hervor: Die drei Kopien waren entstanden, weil die Organisation vorher mit der Pflege dieser Funktionalität in einer gemeinsamen Komponenten gescheitert war. Was dieser Partner wirklich brauchte, waren funktionierende Ansätze für Management von Duplikaten, nicht für die Erkennung. Hierzu gab es aber wenig nützliche Forschungsarbeiten. Heute bekomme ich diese Einblicke öfter. Unsere Firma beschäftigt inzwischen 37 Mitarbeiter, von denen die Hälfte in Software Engineering promoviert hat. Wir alle arbeiten ausschließlich an der Entwicklung oder dem Einsatz von Qualitätsanalysen. Aus unseren Forschungsprototypen ist inzwischen ein kommerzielles Analysewerkzeug geworden, das eine Vielzahl der Analyseansätze aus unserer Community implementiert. Es wird von Entwicklern und Testern weltweit täglich eingesetzt, von KMUs bis hin zu DAX-Konzernen, von Behörden bis zu Unternehmen im Silicon Valley. Auch 12 Jahre später fasziniert mich, welche Ergebnisse und Überraschungen unsere Analysen in der Praxis zu Tage fördern.

Und immer noch zwingen mich die dabei gewonnenen Einblicke regelmäßig, mein Bild von „der Praxis“ weiterzuentwickeln. Beispielsweise setzen wir bei vielen Kunden inzwischen flächendeckend dynamische Analysen auch in Produktionsumgebungen ein, um die Ausführung von Code durch Tester und Anwender aufzuzeichnen. Häufig fällt dabei auf, das auch in strukturierten Testprozessen große Code-Mengen ungetestet in Produktion gelangen würden. Oder das große Teile der Funktionalität seit Jahren von keinem einzigen Anwender genutzt wurden, und vermutlich gelöscht werden könnten. Diese Analysen sind konzeptionell relativ einfach, bringen aber oft die für die Anwender deutlich spannenderen Analysen, als viele technisch bedeutend aufwändigere, statische Analysen.

[1] CCQSE GmbH, Centa-Hafenbrädl-Straße 59, 81249 München, Germany juergens@cqse.eu

doi:10.18420/SE2020_02

In dieser Keynote möchte ich diese und weitere zentralen Erfahrungen und Überraschungen teilen.

Digitalisierung technischer Innovationen - Eine besondere Chance für das Software Engineering oder sollte man lieber die Annahme seines Konferenzpapiers optimieren?

Ralf Reussner[1]

In jeder technischen Innovation steckt heutzutage Software drin. Meist sogar an ganz entscheidender Stelle. Dementsprechend trägt Software immer stärker zur Wertschöpfung durch technische Systeme bei. Es bieten sich daher gerade auch methodisch Chancen des Software Engineering. Informatik-Methoden helfen bei Herausforderungen des Systems Engineering von der Sichtenkonsistenz über Modellanalyse bis hin zur Handhabung von Versionen und Varianten. Damit kann Software Engineering auch methodisch eine zentrale Rolle bei der Systementwicklung einnehmen. Oder wir können stattdessen visionslos nur die Annahme des nächsten Konferenzbeitrags optimieren. Das hilft scheinbar bei der Optimierung der eigenen Laufbahn, vergibt aber die Chance der Weiterentwickkung unserer Disziplin. Der Vortrag beleuchtet technisch die Chancen, die sich gegenwärtig für eine Neupositionierung des Software Engineerings ergeben wie auch aktuelle gegenläufige Trends im Publikationsgebahren unserer Disziplin.

[1] Karlsruhe Institute of Technology, Germany ralf.reussner@kit.edu

doi:10.18420/SE2020_03

SE Prozesse

What are Hybrid Development Methods Made Of? An Evidence-based Characterization

Paolo Tell[1], Jil Klünder[2], Steffen Küpper[3], David Raffo[4], Stephen G. MacDonell[5], Jürgen Münch[6], Dietmar Pfahl[7], Oliver Linssen[8], Marco Kuhrmann[9]

Abstract: Regardless of company size or industry sector, a majority of project teams and companies use customized processes that combine different development methods—so-called hybrid development methods. Even though such hybrid development methods are highly individualized, a common understanding of how to systematically construct synergetic practices is missing. Based on 1,467 data points from a large-scale online survey among practitioners, we study the current state of practice in process use to answer the question: What are hybrid development methods made of? Our findings reveal that only eight methods and few practices build the core of modern software development. This small set allows for statistically constructing hybrid development methods.

This summary refers to the paper *What are Hybrid Development Methods Made Of? An Evidence-based Characterization* [Te19]. This paper was published as full research paper in the proceedings of the International Conference on Software System Process.

Keywords: Software Development; Software Process; Hybrid Methods; Survey Research

1 Introduction

Software development is diverse, and companies have to adopt to new technologies and markets quickly. Hence, software engineers seek suitable development methods. According to Klünder et al. [Kl19], 78.5% of practitioners combine and evolve their processes over time, e.g., to improve product quality and to keep flexibility to react to change.

Problem Statement & Objective An understanding of what a *hybrid development* method is composed of is missing, e.g., which combinations of frameworks, methods, and practices for software and system development help practitioners implement a suitable process.

[1] IT University Copenhagen, Denmark, pate@itu.dk
[2] Leibniz University Hannover, Germany, jil.kluender@inf.uni-hannover.de
[3] Clausthal University of Technology, Germany, steffen.kuepper@tu-clausthal.de
[4] Portland State University, USA, raffod@pdx.edu
[5] Auckland University of Technology, New Zealand, smacdone@aut.ac.nz
[6] Reutlingen University, Böblingen, Germany, Juergen.Muench@Reutlingen-University.de
[7] University of Tartu, Estonia, dietmar.pfahl@ut.ee
[8] FOM University of Applied Sciences, Germany, oliver.linssen@fom.de
[9] University of Passau, Germany, kuhrmann@acm.org

doi:10.18420/SE2020_04

Contribution Based on a large-scale international online survey, we analyze 1,467 data points that provide information about the combined use of 60 frameworks, methods, and practices. Our findings indicate that using hybrid development methods *is* the norm in companies of all sizes and across all industry sectors. We identify eight base methods providing the basis for devising hybrid methods, and we statistically compute sets of practices used to embody the base methods.

2 Results

An analysis of 1,467 data points revealed that using different frameworks, methods and practices in combination as hybrid methods is the norm across companies of all sizes and industry sectors. We identified eight base methods and few practices only that find agreement among study participants. For the study participants that explicitly stated to use processes in combination, we could identify 27 base methods and method combinations that, together with three practices forming three pairs, build the basis to devise hybrid methods. We also found that the sets of practices have limited dependencies to the methods. We therefore argue that practices are the building blocks for devising hybrid methods. We also note this core set of practices along with the complementary sets of practices identified in [Te19] are common to all development methodologies. Since they are so widely deployed, we observe that development organizations see these practices as essential activities enabling them to deliver good software to their customers.

3 Conclusion & Future Work

Our paper [Te19] documents findings from the second stage of the HELENA study. Based on the data collected in HELENA stage 2, among other things, we study different influence factors for method construction and we work towards the development of a statistical construction procedure for hybrid methods.

References

[Kl19] Klünder, Jil; Hebig, Regina; Tell, Paolo; Kuhrmann, Marco; Nakatumba-Nabende, Joyce; Heldal, Rogardt; Krusche, Stephan; Fazal-Baqaie, Masud; Felderer, Michael; Bocco, Marcela Fabiana Genero; Küpper, Steffen; Licorish, Sherlock A.; Lopez, Gustavo; McCaffery, Fergal; Top, Özdem Özcan; Prause, Christian R.; Prikladnicki, Rafael; Tüzün, Eray; Pfahl, Dietmar; Schneider, Kurt; MacDonell, Stephen G.: Catching up with Method and Process Practice: An Industry-Informed Baseline for Researchers. In: Proceedings of International Conference on Software Engineering. SEIP. IEEE, pp. 255–264, May 2019.

[Te19] Tell, Paolo; Klünder, Jil; Küpper, Steffen; Raffo, David; MacDonell, Stephen G.; Münch, Jürgen; Pfahl, Dietmar; Linssen, Oliver; Kuhrmann, Marco: What are hybrid development methods made of? An evidence-based characterization. In: Proceedings of the International Conference on Software and System Processes. ICSSP. IEEE, pp. 105–114, May 2019.

What´s Hot in Product Roadmapping?

Key Practices and Success Factors

Jürgen Münch[1] Stefan Trieflinger[2] Dominic Lang[3]

Abstract: Context: Organizations are increasingly challenged by dynamic and technical market environments. Traditional product roadmapping practices such as detailed and fixed long-term planning typically fail in such environments. Therefore, companies are actively seeking ways to improve their product roadmapping approach. **Goal:** This paper aims at identifying problems and challenges with respect to product roadmapping. In addition, it aims at understanding how companies succeed in improving their roadmapping practices in their respective company contexts. **Method:** We conducted semi-structured expert interviews with 15 experts from 13 German companies and conducted a thematic data analysis. **Results:** The analysis showed that a significant number of companies is still struggling with traditional feature-based product-roadmapping and opinion-based prioritization of features. The most promising areas for improvement are stating the outcomes a company is trying to achieve and making them part of the roadmap, sharing or co-developing the roadmap with stakeholders, and establishing discovery activities.

This summary refers to the paper *What´s Hot in Product Roadmapping? Key Practices and Success Factors* [MTL19c]. This paper was published as full research paper in the Proceedings of the International Conference on Product-Focused Software Process Improvement (PROFES).

Keywords: Product management, Product roadmap, Roadmapping, Product discovery, Innovation

1 Problem Statement and Objective

In general, the purpose of a roadmap is to provide essential understanding, proximity, direction and some degree of certainty regarding the planning of a journey [KS01]. Currently, the product roadmaps of many organizations cover long time horizons and concrete products, features or services together with precise release dates. This approach works well in market environments that are predictable, stable and reliable. However, through increasing market dynamics, rapidly evolving technologies and shifting user expectations, coupled with the adoption of lean and agile practices, it becomes almost impossible to predict which products, features or services will satisfy the needs of the customers and the organization. Thus, companies are increasingly struggling with their ability to create reliable product roadmaps. It seems that the traditional process of product roadmap creation does

[1] Reutlingen University, Germany, juergen.muench@reutlingen-university.de
[2] Reutlingen University, Germany, stefan.trieflinger@reutlingen-university.de
[3] Robert Bosch GmbH, Germany, dominic.lang2@bosch.com

doi:10.18420/SE2020_05

not fulfil its purpose anymore [MTL19b]. The goal of our research is to identify currently applied practices, challenges and success factors with respect to product roadmapping. It also aims at understanding how companies succeed in improving their roadmapping practices in their respective company contexts.

2 Results

Our study revealed that those companies that have already implemented fairly modern product roadmapping practices treat different timeframes differently with respect to the detailing level and the type of items in the roadmap. Companies that show a high level of successful product roadmapping are able to change or update their roadmaps in a way that stakeholders trust these changes and can reliably use the roadmap for their tasks. Frequent ad hoc adjustments usually occur in organizations where products or features are planned in detail over a long-time horizon. This typically leads to a decrease in reliability and trustworthiness of the roadmap. Reliability and trust can be seen as indispensable for the acceptance and successful usage of a product roadmap. Many companies have already a good understanding of success factors for roadmapping processes in dynamic environments, but they are currently struggling with overcoming key challenges. The findings of this study have been used by the authors to develop the so-called DEEP product roadmap maturity model with which practitioners can assess their current product roadmapping practices and identify potentials for an effective improvement of their product roadmapping practices [MTL19a].

References

[KS01] Kostoff, R. N.; Schaller, R. R.: Science and technology roadmaps. IEEE Transactions on Engineering Management 48/2, pp. 132–143, May 2001.

[MTL19a] Münch, J.; Trieflinger, S.; Lang, D.: DEEP: The Product Roadmap Maturity Model. In: Proc. of the 2nd ACM SIGSOFT Int. Workshop on Software-Intensive Business: Start-ups, Platforms, and Ecosystems. IWSiB 2019, ACM, Tallinn, pp. 19–24, 2019.

[MTL19b] Münch, J.; Trieflinger, S.; Lang, D.: The Product Roadmap Maturity Model DEEP: Validation of a Method for Assessing the Product Roadmap Capabilities of Organizations. In (Hyrynsalmi, S.; Suoranta, M.; Nguyen-Duc, A.; Tyrväinen, P.; Abrahamsson, P., eds.): Software Business. Springer International Publishing, Cham, pp. 97–113, 2019.

[MTL19c] Münch, J.; Trieflinger, S.; Lang, D.: What's Hot in Product Roadmapping? Key Practices and Success Factors. In (Franch, X.; Männistö, T.; Martínez-Fernández, S., eds.): Product-Focused Software Process Improvement. Springer International Publishing, Cham, pp. 401–416, 2019.

User Feedback Practices in Continuous Software Engineering

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

Abstract: We summarize the paper *How do Practitioners Capture and Utilize User Feedback during Continuous Software Engineering?* [Jo19], which was presented at the 2019 edition of the IEEE International Requirements Engineering Conference (RE) in Jeju Island, South Korea.

Keywords: User Feedback; Usage Monitoring; Usage Data; Practitioners; Continuous Software Engineering; Tool Support; Experience Report; Interview Study

1 Overview

Continuous software engineering (CSE) is a process for software evolution that enables new capabilities for developers to gain insights. For instance, with the help of continuous delivery, CSE promotes user feedback on the latest changes to a software increment.

Reports on practices of how practitioners capture and utilize user feedback in industry are sparse. As a result, the interaction with users in CSE environments is less understood and there is an identified research gap for mechanisms that make use of feedback. For that reason, we strive to investigate current user feedback practices with the goal to identify improvements and benefits during requirements engineering.

In 2017, we conducted a semi-structured interview study with 24 practitioners from 17 companies. The group of interviewed practitioners consisted of multiple developers, technical leaders, CSE specialists, project managers, and an executive director. We relied on a questionnaire supported by a guideline to ensure comparability between the interviews. The interviews' transcripts served as the starting point for a two-step analysis, in which we allocated answers to research questions (RQs) and then applied fine-grained codings.

As outlined in Section 2, we created three RQs that address general user feedback considerations as well as its capture and utilization. We further divided them into nine sub-RQs. On the basis of the interviews' results, we derived five recommendations which we summarize in Section 3 with the goal to improve continuous user feedback capture and utilization.

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

doi:10.18420/SE2020_06

2 Results

Which user feedback do practitioners consider for CSE? All practitioners rely on explicit user feedback during CSE. None relies solely on implicit user feedback, but almost half of them use it to support explicit user feedback. Many practitioners report to relate user feedback to both the application itself as well as to specific features. However, they lack a systematic approach which results in less actionable insights derived from the user feedback.

How do practitioners capture user feedback in CSE? Most of the practitioners do not continuously capture user feedback; instead, they rely on either event-based or periodic capture processes, such as the release of a new version or on a monthly basis. More than half of the practitioners employ tool support to capture user feedback and predominantly rely on standard software. At the same time, many practitioners noted to rely on manual capture approaches. Feedback capture from external sources dominates internal sources. Technical limitations hinder practitioners from capturing the context of user feedback.

How do practitioners utilize user feedback in CSE? Practitioners utilization of user feedback spans across planning activities, support activities such as bug fixing, as well as the improvement of existing features. Practitioners barely exploit changes in the user feedback over time as they already struggle to cope with basic user feedback utilization. While only a few of the practitioners combine different user feedback types following a pragmatic approach, many other practitioners would welcome this possibility as well.

3 Recommendations

We derived five recommendations to achieve continuous user feedback capture and utilization to benefit requirements engineering during CSE. First, internal sources, such as team members or colleagues from other teams, should be systematically approached, as they provide a rich source of user feedback. Second, existing tool support should be adapted and extended to automate user feedback processing with the goal to ensure requirements elicitation in a timely manner. Third, a lightweight concept for creating reference points should be established in order to relate user feedback to individual requirements. Fourth, the utilization of user feedback for requirements validation should be increased to improve existing and explore new requirements. Fifth, the communication between developers and users should be enabled to enhance the quality of requirements.

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

Bibliography

[Jo19] Johanssen, Jan Ole; Kleebaum, Anja; Bruegge, Bernd; Paech, Barbara: How do Practitioners Capture and Utilize User Feedback during Continuous Software Engineering? In: Proceedings of the 27th IEEE International Requirements Engineering Conference. RE, September 2019.

Domänen-spezifische Softwareentwicklung

Reusability in Artificial Neural Networks: An Empirical Study

Javad Ghofrani[1], Ehsan Kozegar[2], Arezoo Bozorgmehr[3], Mohammad Divband Soorati[4]

Abstract: This manuscript summarizes our empirical study on reuse in the context of Artificial Neural Networks (ANNs) extensively described in [Gh19b] as a paper with the same title, presented in 2nd International Workshop on Experiences and Empirical Studies on Software Reuse (WEESR 2019).

Keywords: Systematic reuse; Artificial Neural Networks; Reusability; Survey; Empirical study

Summary

Artificial Neural Networks (ANNs), especially deep neural networks, are machine learning methods that have fast growing trends in research and practice since 2014. Solutions based on ANNs have shown successful outcomes where conventional software systems reach their limits—e.g. natural language processing and image recognition. Although ANNs are computer programs, important concepts of classical software development methods—especially reuse—are not effectively considered in the development of these software solutions. In our paper [Gh19b], we investigated the state-of-the-practice around reusability in the context of ANNs through an online survey. We selected the participants of our survey from academics and industrial staffs who were involved in the development of ANNs. Through this study we try to create a connection between software engineering and artificial intelligence communities. We proposed three research questions:
RQ1) How are neural networks already reused?
RQ2) What types of neural networks are commonly reused and why?
RQ3) What are the main challenges related to the reuse of ANNs?
We conducted an online survey (from April 8th, 2019 to May 15th, 2019) for answering these research questions. Our survey consists of five main parts. First part filters the non-relevant participants. In the second part, we collect information about "Most significant experience with ANNs". In the third part, our questionnaire extracts details of participants' opinions and experiences in reusing ANNs. The fourth part is about demographic data of the participants.

[1] HTW Dresden, Faculty of Mathematic/Informatics, Friedrich-List-Platz 1, 01069 Dresden," Germany javad.ghofrani@gmail.com
[2] University of East Guilan, Department of computer science, Guilan, Iran ehsan.kozegar@uniguilan.ac.ir
[3] General Practice and family medicine University Hospital Bonn, Bonn, Germany arezoo.bozorgmehr@ukbonn.de
[4] School of Electronics and Computer Science, University of Southampton, UK m.divband-soorati@soton.ac.uk

doi:10.18420/SE2020_07

The last part collects the personal feedback about the survey. Our questionnaire and the results are available online [Gh19a]. Complete surveys (114 from 229 responds) that passed the filter questions (in the first part of the questionnaire) were selected. 112 completed surveys were analyzed and the responses were used to answer our research questions (RQ1-RQ3).

Majority of participants had two to five years experience in working with ANNs. Most of them came from Iran, Germany, China, USA, and India and were from domain of Engineering and Computer Sciences. Most of the participants state that they use ad-hoc methods for managing projects instead of classical development processes.

The results of the survey showed that reuse is considered as an essential element in all main levels of ANNs' structure. Although, reuse happens around network structures (code) and data-sets more than parameters (network weights). The participants have different perceptions about reusability. Majority of participants define reuse as "modifying an existing module and adapt it for self use cases" towards "Creating a general neural network with the aim of using in various context". As shown in Figure 1, Deep Convolutional Neural Networks (CNNs) were reused at most while Deep Belief Networks and Capsule NNs were 'never' reused. Public repositories were the most used resource for reusing the artifacts of ANN Development.

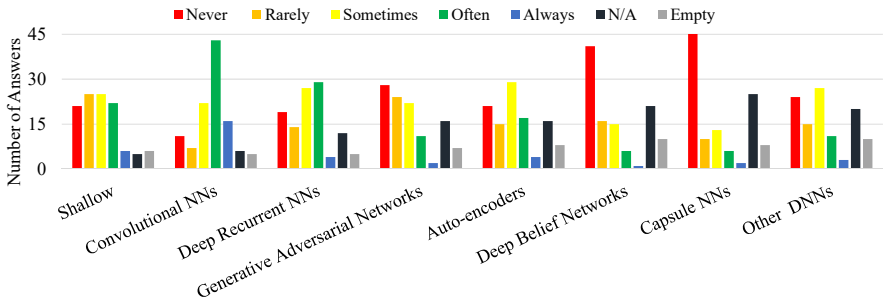


Fig. 1: Frequency of reuse for common ANNs

Based on the answers, reuse happens when the domain is known and the most significant challenge in reusing ANNs is that "poorly documented modules make them complicated to reuse".

Bibliography

[Gh19a] Ghofrani, Javad; Kozegar, Ehsan; Bozorgmehr, Arezoo; Divband Soorati, Mohammad: , Results of survey about reusability in Artificial neural networks, Jun 2019.

[Gh19b] Ghofrani, Javad; Kozegar, Ehsan; Bozorgmehr, Arezoo; Divband Soorati, Mohammad: Reusability in artificial neural networks: an empirical study. In: Proceedings of the 23rd International Systems and Software Product Line Conference-Volume B. ACM, p. 77, 2019.

Review-Modelle zur Unterstützung in der funktionszentrierten Entwicklung eingebetteter Systeme

Marian Daun[1], Thorsten Weyer[2], Klaus Pohl[3]

Abstract: Dieser Vortrag berichtet von dem Beitrag *Improving manual reviews in function-centered engineering of embedded systems using a dedicated review model* [DWP19], der in der Fachzeitschrift *Software and Systems Modeling* veröffentlicht wurde. Im Rahmen des Beitrags wurde ein dezidiertes Review-Modell vorgeschlagen, um die Validierung der interaktionsbasierten Verhaltensanforderungen und des funktionalen Entwurfs in einem gemeinsamen Qualitätssicherungsschritt zu unterstützen. Neben der automatisierten Erzeugung des Review-Modells wurde mithilfe kontrollierter Experimente die Vorteilhaftigkeit des vorgeschlagenen Review-Modells nachgewiesen.

Keywords: Reviews; Eingebettete Systeme; Requirements Engineering; Manuelle Validierung

1 Einleitung und Problemstellung

In der modellbasierten Entwicklung eingebetteter Systeme werden Modelle häufig durch Inaugenscheinnahme qualitätsgesichert, um sicherzustellen, dass das zu entwickelnde System die Erwartungen der diversen Stakeholder erfüllt. Hierzu werden in der Regel manuelle Reviews oder Inspektionssitzungen durchgeführt. Änderungen der Stakeholderintentionen führen zu einer Überarbeitung der Entwicklungsartefakte, der Anforderungen und des Entwurfs. In der industriellen Praxis werden Änderungen der Stakeholderintentionen häufig nicht unmittelbar aufgedeckt und in alle Entwicklungsartefakte eingearbeitet [DHW14]. Hierdurch entstehen Inkonsistenzen zwischen verschiedenen Entwicklungsartefakten, die im Laufe der Zeit nicht einfach aufzulösen sind, da es schwer nachzuvollziehen ist, welches der inkonsistenten Entwicklungsartefakte den aktuell korrekten Stand zeigt und welches veraltet ist, sofern nicht beide veraltet sind. Daher wird ein dezitiertes Review-Modell vorgeschlagen, das durch Modelltransformationen automatisiert erzeugt werden kann und den Entwicklern Inkonsistenzen zwischen verschiedenen Artefakten aufbereitet anzeigt.

2 Review-Modell

Um die manuelle Qualitätssicherung zu unterstützen, wird ein Review-Modell erzeugt, das verschiedene Ursprungsartefakte zusammenfasst und Unterschiede hervorhebt. Für

[1] Universität Duisburg-Essen, paluno - The Ruhr Institute for Software Technology, marian.daun@uni-due.de
[2] Universität Koblenz-Landau, Institut für Softwaretechnik, thorstenweyer@uni-koblenz.de
[3] Universität Duisburg-Essen, paluno - The Ruhr Institute for Software Technology, klaus.pohl@uni-due.de

doi:10.18420/SE2020_08

die funktionszentrierte Entwicklung eingebetteter Systeme wurde dies für die Behavioral Requirements und das Functional Design gezeigt. Zur Unterstützung des Reviewers werden drei unterschiedliche Diagrammtypen unterschieden:

- *Refinement Diagrams* zeigen Sachverhalte, die in beiden Artefakten konsistent dargestellt sind. Da in der Regel das Functional Design eine konkretere Spezifikation des Systems darstellt als die Behavioral Requirements, werden hierbei auch Verfeinerungsbeziehungen automatisiert identifiziert und berücksichtigt.
- *Unrefinable Diagrams* zeigen Sachverhalte, die nur in den Behavioral Requirements enthalten sind und nicht im Functional Design berücksichtigt wurden.
- *Unspecified Diagrams* zeigen Sachverhalte, die im Functional Design enthalten sind, aber nicht auf die Behavioral Requirements zurückgeführt werden können.

3 Evaluation

Zur Evaluation wurden kontrollierte Experimente entworfen [Da15, Da19]. Mit diesen konnte die Vorteilhaftigkeit des Review-Modells für die Qualitätssicherung nachgewiesen werden. Insbesondere zeigten sich eine verbesserte Effektivität und Effizienz sowie eine sicherere Entscheidungsfindung. Daneben wurde das Review-Modell von den Teilnehmern mehrheitlich als vorteilhaft angesehen. Darüber hinaus erlaubt der Vergleich mehrerer Experimente, Aussagen über die Wirkung des Review-Modells zu treffen. Es liegt nahe, dass sowohl die gewählte Notation des Review-Modells als auch die Synthese zweier Artefakte in einem einzelnen Modell einen positiven Effekt auf den Review haben.

Literaturverzeichnis

[Da15] Daun, Marian; Salmon, Andrea; Weyer, Thorsten; Pohl, Klaus: The Impact of Students' Skills and Experiences on Empirical Results: A Controlled Experiment with Undergraduate and Graduate Students. In: Proceedings of the 19th International Conference on Evaluation and Assessment in Software Engineering. EASE '15, ACM, New York, NY, USA, S. 29:1–29:6, 2015.

[Da19] Daun, M.; Brings, J.; Krajinski, L.; Weyer, T.: On the Benefits of using Dedicated Models in Validation Processes for Behavioral Specifications. In: 2019 IEEE/ACM International Conference on Software and System Processes (ICSSP). S. 44–53, May 2019.

[DHW14] Daun, M.; Höfflinger, J.; Weyer, T.: Function-centered engineering of embedded systems: Evaluating industry needs and possible solutions. In: 2014 9th International Conference on Evaluation of Novel Approaches to Software Engineering (ENASE). S. 1–9, April 2014.

[DWP19] Daun, Marian; Weyer, Thorsten; Pohl, Klaus: Improving manual reviews in function-centered engineering of embedded systems using a dedicated review model. Software and Systems Modeling, 18(6):3421–3459, 2019.

Enactment of Adaptation in Data Stream Processing with Latency Implications—A Systematic Literature Review

Cui Qin[1], Holger Eichelberger[2], Klaus Schmid[3]

Abstract: This summary refers to the paper *Enactment of adaptation in data stream processing with latency implications – A systematic literature review* [QES19]. This paper is a journal paper published in *Information and Software Technology (IST)* in July 2019.

Runtime adaptation in stream processing plays a significant role in supporting the optimization of data processing tasks. In recent years, runtime adaptation, particularly its enactment, has received significant interest in scientific literature. However, so far no categorization of the enactment approaches for runtime adaptation in stream processing has been established.

This paper presents a systematic literature review (SLR), where we identify and characterize different approaches towards the enactment of runtime adaptation in stream processing with a main focus on latency as quality dimension. We discovered 75 relevant papers out of 244 papers from the search. We identified 17 different enactment categories and developed a taxonomy to characterize all possible enactment approaches. We extracted the realization techniques of each identified enactment approach and classified them into categories. Furthermore, we identified 9 categories of processing problems, 6 adaptation goals, 9 evaluation metrics and 12 evaluation parameters from the identified enactment approaches. The research interest on enactment approaches has significantly increased in recent years. The most commonly applied enactment approaches are parameter adaptation to tune parameters or settings of the processing, load balancing used to re-distribute workloads, and processing scaling to dynamically scale up and down the processing.

Keywords: Stream processing; Big Data; runtime adaptation; enactment; latency; systematic literature review

In recent years, we witnessed increasing attention to data-intensive applications [CJ09; Kl17] such as stock trading, cyber-physical systems, social media, etc. Such applications aim at continuously providing analysis results to end-users, some even, with rather strict time constraints. In the continuous processing of data streams, the processing situations may vary over time. For example, the volume or velocity of streams can change drastically. To cope with the dynamic characteristics of data streams as well as the varying processing environment, runtime adaptation of the processing becomes critical. There are different

[1] University of Hildesheim, Institute of Computer Science, Software Systems Engineering, Universitätsplatz 1, 31141 Hildesheim, Germany qin@sse.uni-hildesheim.de
[2] University of Hildesheim, Institute of Computer Science, Software Systems Engineering, Universitätsplatz 1, 31141 Hildesheim, Germany eichelberger@sse.uni-hildesheim.de
[3] University of Hildesheim, Institute of Computer Science, Software Systems Engineering, Universitätsplatz 1, 31141 Hildesheim, Germany eichelberger@sse.uni-hildesheim.de

doi:10.18420/SE2020_09

ways of adapting the processing behavior in literature. However, so far no categorization for runtime adaptation in stream processing has been established.

In [QES19], we present a systematic literature review to identify and characterize different approaches for the enactment of runtime adaptation in stream processing with a main focus on latency as quality dimension. Our study aims to identify the existing enactment approaches for runtime adaptation and categorizing them into a taxonomy. Moreover, we analyze each enactment approach by extracting their realization techniques, their addressed problems, goals as well as their evaluation focus. We target the following main research questions:

RQ1. In studies analyzing adaptation in data stream processing from a latency perspective, which enactment approaches exist?

RQ2. What techniques are used to realize which enactment?

We discovered 244 relevant papers out of which 75 were categorized as relevant. As the result of **RQ1**, we identified 17 categories of enactment approaches and provided a taxonomy classifying them from the perspective of resource, data, and processing. We summarized the frequency of the papers in which the enactment approach is proposed or applied. We identified that the most commonly applied enactment approaches are Parameter Adaptation to tune parameters or settings of the processing, Load Balancing used to re-distribute workloads, and Processing Scaling to dynamically scale up and down the processing. Furthermore, for each enactment approach we identified the underlying realization techniques to gain insight into the diverse realizations of adaptation in different contexts for answering **RQ2**. We categorized the enactment approaches into 9 categories of addressed processing problems, 6 adaptation goals, 9 evaluation metrics and 12 evaluation parameters to obtain an overview of the adaptation focuses from different perspectives.

Literatur

[CJ09] Chakravarthy, S.; Jiang, Q.: Stream data processing: a quality of service perspective: modeling, scheduling, load shedding, and complex event processing. Springer Science & Business Media, 2009.

[Kl17] Kleppmann, M.: Designing Data-Intensive Applications: The Big Ideas Behind Reliable, Scalable, and Maintainable Systems. O'Reilly Media, Inc., 2017.

[QES19] Qin, C.; Eichelberger, H.; Schmid, K.: Enactment of Adaptation in Data Stream Processing with Latency Implications—A Systematic Literature Review. Information and Software Technology 111/, S. 1–21, 2019, ISSN: 0950-5849, URL: https://www.sciencedirect.com/science/article/abs/pii/S0950584919300539.

Software Architektur, Design und Model-to-Code Mapping

Architektur-basierte Analyse von Änderungsausbreitung in Software-intensiven Systemen

Robert Heinrich[1], Sandro Koch[2], Suhyun Cha[3], Kiana Busch[4], Ralf Reussner[5], Birgit Vogel-Heuser[6]

Software ist ein wesentlicher Bestandteil unseres täglichen Lebens. Mobilität, Energie, Wirtschaft, Produktion und Infrastruktur hängen stark von Software ab, die allerdings nicht immer von hoher Qualität ist. Kritische Probleme, wie Effizienzeinbrüche oder hohe Wartungsaufwände, können durch schlechte Softwarequalität verursacht werden. Beispiele sind vielfältig in der Presse zu finden. Die Qualität software-intensiver Systeme muss nicht nur bei der initialen Entwicklung sichergestellt werden, sondern insbesondere aufrecht erhalten werden bei wiederholten Änderungen des Systems über einen langen Zeitraum hinweg, was als Systemevolution bezeichnet wird. Qualitätseigenschaften hängen stark von Entwurfsentscheidungen bzgl. der Architektur eines Systems ab [Re16]. Um eine hohe Qualität bei der Systemevolution zu gewährleisten, sind Forschung und Praxis an Ansätzen interessiert, mit denen verschiedene Entwurfsalternativen modelliert und analysiert werden können.

Mit diesen Ansätzen können Auswirkungen von Änderungsszenarien auf die Qualitätseigenschaften eines Systems vor deren Umsetzung vorhergesagt werden. Dies wirft interessante Forschungsfragen danach auf, ob Architektur eine geeignete Abstraktion zur Qualitätsanalyse in software-intensiven Systemen ist und wie Sprachen zur Modellierung dieser Systeme und deren Qualitätseigenschaften sowie dazugehörige Analysetechniken so entworfen werden können, dass sie erweiterbar und wiederverwendbar sind, um Evolution zu unterstützen.

Aufbauend auf dem Ansatz "Karlsruhe Architecture Maintainability Prediction" (KAMP) [Ro15] entwickeln wir den Ansatz KAMP4aPS [He18] zur Analyse von Änderungsausbreitung in automatisierten Produktionssystemen [Vo15]. Es werden Modellierungs- und Analysetechniken vorgestellt, die zur Untersuchung von Wartungsaufwänden für elektronische, mechanische und Software-Bestandteile eines automatisierten Produktionssystems auf Architekturebene geeignet sind. Die verschiedenen Ausprägungen von KAMP für Softwaresysteme [Ro15], Geschäftsprozesse [Ro17], PLC-Software [Bu18] und automatisierte Produktionssystemen [He18] werden zu einer Domänen-übergreifenden Methodologie zur

[1] Karlsruher Institut für Technologie (KIT) robert.heinrich@kit.edu
[2] Karlsruher Institut für Technologie (KIT) sandro.koch@kit.edu
[3] Technische Universität München suhyun.cha@tum.de
[4] Karlsruher Institut für Technologie (KIT) kiana.busch@kit.edu
[5] Karlsruher Institut für Technologie (KIT) reussner@kit.edu
[6] Technische Universität München vogel-heuser@tum.de

doi:10.18420/SE2020_10

Änderungsausbreitungsanalyse verallgemeinert [HBK18]. Darüber hinaus werden Modularisierungskonzepte für Sprachen (definiert durch Metamodelle) und Analysetechniken vorgestellt. Diese führen zur ersten Referenzarchitektur für Metamodelle zur Qualitätsmodellierung und -analyse [HSR19].

Literaturverzeichnis

[Bu18] Busch, Kiana; Rätz, Jannis; Koch, Sandro; Heinrich, Robert; Reussner, Ralf; Cha, Suhyun; Seitz, Matthias; Vogel-Heuser, Birgit: A Model-Based Approach to Calculate Maintainability Task Lists of PLC Programs for Factory Automation. In: IECON 2018 - 44th Annual Conference of the IEEE Industrial Electronics Society. IEEE, S. 2949–2954, 2018.

[HBK18] Heinrich, Robert; Busch, Kiana; Koch, Sandro: A Methodology for Domain-spanning Change Impact Analysis. In: 2018 44th Euromicro Conference on Software Engineering and Advanced Applications. IEEE, S. 326–330, 2018.

[He18] Heinrich, Robert; Koch, Sandro; Cha, Suhyun; Busch, Kiana; Reussner, Ralf; Vogel-Heuser, Birgit: Architecture-based change impact analysis in cross-disciplinary automated production systems. Journal of Systems and Software, 146:167 – 185, 2018.

[HSR19] Heinrich, Robert; Strittmatter, Misha; Reussner, Ralf Heinrich: A Layered Reference Architecture for Metamodels to Tailor Quality Modeling and Analysis. IEEE Transactions on Software Engineering, 2019.

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

[Ro15] Rostami, Kiana; Stammel, Johannes; Heinrich, Robert; Reussner, Ralf: Architecture-based Assessment and Planning of Change Requests. In: Proceedings of the 11th International ACM SIGSOFT Conference on Quality of Software Architectures. ACM, S. 21–30, 2015.

[Ro17] Rostami, Kiana; Heinrich, Robert; Busch, Axel; Reussner, Ralf: Architecture-based Change Impact Analysis in Information Systems and Business Processes. In: 2017 IEEE International Conference on Software Architecture. IEEE, S. 179–188, 2017.

[Vo15] Vogel-Heuser, Birgit; Fay, Alexander; Schaefer, Ina; Tichy, Matthias: Evolution of software in automated production systems: Challenges and research directions. Journal of Systems and Software, 110:54 – 84, 2015.

Datenzentrische Softwarearchitekturen zur Analyse von Vertraulichkeit[1]

Stephan Seifermann[2], Robert Heinrich[2], Ralf Reussner[2]

Keywords: Softwarearchitektur; Datenflussanalyse; Vertraulichkeit

1 Übersicht

Die Definition und Umsetzung von Sicherheitsanforderungen ist für alle Arten von Anwendungen essentiell. Insbesondere die Wahrung von Vertraulichkeit ist von hoher Bedeutung, da ansonsten Reputationsverlust oder hohe Strafen drohen. Mit Inkrafttreten der europäischen Datenschutzgrundverordnung sind die rechtlichen Rahmenbedingungen zur Wahrung der Vertraulichkeit sogar noch einmal verschärft worden.

Die Einhaltung von Anforderungen zu überprüfen ist nicht trivial und gerade in komplexen Softwaresystemen nicht mehr ohne unterstützende Analysewerkzeuge möglich. Im Allgemeinen gilt, dass verletzte Anforderungen so früh wie möglich identifiziert werden müssen, um teuere Nachbesserungen in späteren Entwicklungsphasen zu vermeiden.

In frühen Entwicklungsstadien können Entwurfszeitanalysen genutzt werden. Dabei ist zwischen kontrollfluss- und datenflussorientierten Analysen zu unterscheiden. Die als Bedingungen an Daten formulierten Anforderungen können in datenflussorientierten Analysen genutzt werden, um in der gleichen Terminologie Analyseziele zu formulieren. Bestehende Ansätze integrieren sich jedoch nicht gut in existierende architekturelle Beschreibungssprachen (ADLs), sind eingeschränkt in ihrer Ausdrucksmächtigkeit oder erfordern detaillierte Beschreibungen auf dem Niveau von Quelltext.

In unserem Ansatz [SHR19] haben wir daher Datenflussmodellierung in die ADL Palladio [Re16] integriert. Über Propagation von Datencharakteristiken und Abgleich dieser Charakteristiken mit denen von Verarbeitungsschritten können Analysen formuliert werden. Mehr dazu findet sich in Abschnitt 2. Die Nutzung von Geschäftsprozessen [PSH18] und grobgranularen Datenflussdiagrammen [SWE19] zur Erzeugung und Parametrisierung unserer Modelle wurde initial untersucht. Eine Evaluierung mittels zweier Fallstudien zeigte, dass eine gute Präzision mittels der Analyse erreicht werden kann.

[1] Diese Arbeit wurde gefördert durch das deutsche Bundesministerium für Bildung und Forschung unter dem Förderkennzeichen 01IS17106A (Trust 4.0).

[2] Karlsruher Institut für Technologie (KIT), IPD Reussner, Am Fasenengarten 5, 76131 Karlsruhe, Deutschland, firstname.lastname@kit.edu

doi:10.18420/SE2020_11

2 Ansatz

Der Ansatz unterteilt sich in einen Modellierungs- und einen Analyseteil.

Modellierungsansatz Der Modellierungansatz erweitert die Palladio-ADL um Annotationen an Rechenknoten und bestehenden kontrollflussorientierten Verarbeitungsoperationen. Rechenknoten können mit Charakteristiken, wie der geographischen Lokation, annotiert werden. Kontrollflussorientierte Verarbeitungsoperationen können mit einer Menge von Datenverarbeitungsoperationen annotiert werden. Datenverarbeitungsoperationen berechnen Charakterstiken von ausgehenden Daten basierend auf den Charakteristiken von eingehenden Daten, sowie der Semantik der Operation. Einige Datenverarbeitungsoperationen sind vordefiniert, es ist jedoch auch möglich, eigene Operationen zu definieren. Der Verknüpfung von Operationen ist durch die Datenabhängigkeiten und Aufrufe aus dem kontrollflussorientierten Palladio-Modell gegeben. Charakteristiken können grundsätzlich frei definiert werden, sodass der Einsatz in verschiedenen Anwendungsdomänen möglich ist.

Analyseansatz Der Ansatz zur Analyse basiert auf der Propagation von Datencharakterstiken und einem Vergleich dieser Charakteristiken mit Vorgaben oder Charakterstiken von Ressourcen. Die Propagation von Charakterstiken erfolgt durch Auswertung der für die Datenverarbeitungsoperation definierten Berechnungsvorschrift und der Anwendung der berechneten Charakteristiken auf die Ausgabedaten. Dabei ist entscheidend, welche Charakteristiken die eingehenden Daten haben. Da es grundsätzlich mehrere mögliche Sequenzen von Datenverarbeitungsoperationen und initialen Datencharakterstiken geben kann, wird während der Analyse jeden mögliche Sequenz berücksichtigt. Eine konkrete Fragestellung für eine Analyse ist durch Formulierung einer Anforderung gegeben. Dabei werden berechnete Datencharakteristiken mit einer fixen Charakteristik oder einer Ressourcencharakterstik verglichen. Das Ergebnis der Analyse ist dann eine Verletzung der Anforderung inklusive der dazu führenden Sequenz von Verarbeitungsoperationen.

Literatur

[PSH18] Pilipchuk, R.; Seifermann, S.; Heinrich, R.: Aligning Business Process Access Control Policies with Enterprise Architecture. In: Proceedings of the Central European Cybersecurity Conference 2018. CECC'18, ACM, 2018.

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

[SHR19] Seifermann, S.; Heinrich, R.; Reussner, R. H.: Data-Driven Software Architecture for Analyzing Confidentiality. In: IEEE International Conference on Software Architecture, ICSA 2019. IEEE, S. 1–10, 2019.

[SWE19] Seifermann, S.; Werle, D.; Ebada, M.: Mapping Data Flow Models to the Palladio Component Model. Softwaretechnik-Trends/, 2019.

Measuring Object-Oriented Design Principles: The Results of Focus Group-Based Research

Johannes Bräuer[1] Reinhold Plösch[2] Matthias Saft[3] Christian Körner[4]

Abstract: This work was published in the **Journal of Systems and Software**, Volume 140, **June 2018**, Pages 74-90, **doi.org/10.1016/j.jss.2018.03.002**. Object-oriented design principles are fundamental concepts that foster the development of software-intensive systems with a focus on good design quality. The aim of this paper is to examine the relationship between design best practices and 10 selected design principles. This should provide evidence whether the key design aspects of the design principles are covered. We conducted focus group research with six focus groups and 31 participants in total. Each group discussed five design principles and assessed the coverage by using the Delphi method. The result reveals the impact of each design best practice to the design principle and shows that the main design aspects of the design principles are covered by our approach.

Keywords: design best practices; design rules; design principles; software design quality

1 Introduction and Research Design

In previous work [Pl16a], [Pl16b] and [Br17] we have gained evidence on the importance of design knowledge-carrying concepts (design best practices and design principles) and the findings met our expectations. However, the important and remaining question is whether the prosed design best practices for a specific design principle cover the essential aspects of the principle or just touch on some minor design aspects. To answer this general question, we derived following research questions: *RQ1:* How important are design best practices concerning their assigned design principle? *RQ2:* Are there additional design best practices for operationalizing a design principle? *RQ3:* To what extent can the design knowledge of a design principle be grasped by the associated design best practices?

To answer the research questions, participants need to understand the concept and characteristics of each design principle. Further, they may have questions about some aspects that need to be clarified. With a solid understanding of the design principle, it is possible to judge whether design best practices are related to a design principle. According to these circumstances, we decided to conduct focus group research. The entire design of the research method and its combination with Delphi as a data collection method is aligned with the guidance given by [KBL08].

[1] Johannes Kepler University Linz, Linz, Austria johannes.braeuer@gmail.com
[2] Johannes Kepler University Linz, Linz, Austria, reinhold.ploesch@jku.at
[3] Siemens AG, Corporate Technology, Munich, Germany, matthias.saft@siemens.com
[4] Siemens AG, Corporate Technology, Munich, Germany, christian.koerner@siemens.com

doi:10.18420/SE2020_12

2 Results

For *RQ1* we assessed 10 design principles and their design best practices. The ten principles are: Single responsibility principle (SRP), Information hiding principle (IHI), Don't repeat yourself (DRY), Open closed principle (OCP), Acyclic dependency principle (ADP), Interface segregation principle(ISP), Favor composition over inheritance (FCOI), Command query separation (CQS), Common closure principle (CCP), and Program to an interface, not an implementation (PINI). Except for OCP, the research reveals that the other nine principles have at least one design best practice, which has high or very high importance. In other words, those design best practices assessed as very high or high meet the design aspect of the principle. This provides a first justification that practices express certain parts of the associated design principles. In the context of answering *RQ2*, the particiapants of our study suggested additional 34 design best practices for our 10 design principles. Some of these suggestions, e.g. the design best practice *UseMeaningfulVariableNames* cannot be implemented with classical static code analysis. For *RQ3* the participants of the study rated the completeness of the design best pracitces. Table 1 gives an overview of the results.

Tab. 1: Average completeness of the design best practices

DP	Mean	Std. Dev. in Points	DP	Mean	Std. Dev. in Points
IHI	66	14	DRY	65	14
SRP	58	17	ADP	74	16
FCOI	83	13	OCP	65	14
CCP	71	18	ISP	57	20
CQS	76	19	PINI	81	19

The assessment of the obtained completeness of each design principle shows that most design aspects are covered by design best practices. In other words, none of the principles is missing a central element.

Bibliography

[Br17] Bräuer, Johannes; Plösch, Reinhold; Saft, Matthias; Körner, Christian: A Survey on the Importance of Object-oriented Design Best Practices. In: 43rd Euromicro Conference on Software Engineering and Advanced Applications (SEAA). IEEE, Vienna, Austria, pp. 27–34, 2017.

[KBL08] Kontio, Jyrki; Bragge, Johanna; Lehtola, Laura: The Focus Group Method as an Empirical Tool in Software Engineering. In (Shull, Forrest; Singer, Janice; Sjöberg, Dag I. K., eds): Guide to Advanced Empirical Software Engineering, pp. 93–116. Springer London, 2008.

[Pl16a] Plösch, Reinhold; Bräuer, Johannes; Körner, Christian; Saft, Matthias: Measuring, Assessing and Improving Software Quality based on Object-Oriented Design Principles. Open Computer Science, 6(1):187–207, 2016.

[Pl16b] Plösch, Reinhold; Bräuer, Johannes; Körner, Christian; Saft, Matthias: MUSE - Framework for Measuring Object-Oriented Design. Journal of Object Technology, 15(4):2:1–29, 2016.

Secure Data-Flow Compliance Checks between Models and Code based on Automated Mappings (Summary)

Sven Peldszus[1] Katja Tuma[2] Daniel Strüber[2] Jan Jürjens[1,3] Riccardo Scandariato[2]

Abstract: We present our paper published at the 2019 edition of the International Conference on Model Driven Engineering Languages and Systems (MODELS) [Pe19]. During the development of security-critical software, the system implementation must capture the security properties postulated by the architectural design. To iteratively guide the developer in discovering such compliance violations we introduce *automated mappings*. These mappings are created by searching for correspondences between a design-level model (Security Data Flow Diagram) and an implementation-level model (Program Model). We limit the search space by considering name similarities between model elements and code elements as well as by the use of heuristic rules for matching data-flow structures. The automated mappings support the designer in an early discovery of implementation absence, convergence, and divergence with respect to the planned software design as well as the discovery of secure data-flow compliance violations. We provide a publicly available implementation of the approach and its evaluation on five open source Java projects.

1 Introduction

Security threats to software systems are a growing concern in many organizations. Therefore, one needs to consider security early in the design phase, when little is known about the system. Before any new functionality is released, it must be checked that every security assumption made in any of the phases is met. The state of the art for this check in practice are manual code reviews by security experts. Since such reviews are expensive and error-prone, they are only performed on selected code parts, leaving a large leeway for security threats.

In the context of software architecture design, threat analysis techniques aim to identify security threats to software systems and to plan countermeasures to mitigate them. Yet, empirical evidence shows that existing threat analysis techniques can be manually labor intensive and lack in automation. Furthermore, design-level models are seldom kept in sync with the implementation, potentially resulting in architectural erosion and technical debt. Threat analysis is often performed on *Data Flow Diagrams* (DFD), an informal representation of the software architecture. To support the detection of problematic information flows, in our rearlier work we introduced *SecDFD*, an extension of the DFD notation supporting the specification of security-relevant information [TBS19]. However, the outcomes of such

[1] University of Koblenz-Landau, Koblenz, Germany, EMail: speldszus@uni-koblenz.de, juerjens@uni-koblenz.de
[2] University of Gothenburg and Chalmers University of Technology, Gothenburg, Sweden, EMail: katja.tuma@cse.gu.se, danstru@chalmers.se, riccardo.scandariato@cse.gu.se
[3] Fraunhofer Institute for Software and Systems Engineering, Dortmund, Germany

doi:10.18420/SE2020_13

detection are of limited value if the implementation does not comply with the security properties described in the DFD model.

Our work aims to support the discovery of secure data-flow compliance violations between the designed and the implemented system. We present a technique that automatically establishes mappings between a design-level model enriched with security-relevant information (SecDFD) and an implementation-level model (*Program Model* [Pe15]). These mappings can be used to discover compliance violations of secure data-flow properties: First, the designed data flow is captured in the SecDFD model and afterwards the actual data flow is obtained from implementation-level data-flow analysis tools. These tools typically require sophisticated meta-data (e.g. an explicit tagging of security-critical data) as input, which can be obtained from our mappings. Finally, our mappings also support the designer in an early discovery of implementation absence, convergence, and divergence with respect to the planned software design and its security properties. We make the following contributions:

(i) We present an automated technique for establishing mappings between SecDFDs and program models, thereby supporting the discovery of secure data-flow compliance violations. The key idea of our technique is twofold. First, we define a mapping between SecDFD and program-model element types, constraining how elements can be mapped to each other. Second, we combine similarity-based matching of element names with structural heuristics (based on data-flow properties) to automatically derive suggested mappings between the SecDFD and the program model.

(ii) We present an incremental methodology, in which the user is involved to successively discover new mappings and eventually derive an adequate mapping.

(iii) We present our implementation of the approach as a publicly available Eclipse plugin and the evaluation of its accuracy on five open source Java projects.

Our tool implementation as well as all experimental data sets are available on our GitHub site (`https://github.com/SvenPeldszus/GRaViTY-SecDFD-Mapping`).

References

[Pe15] Peldszus, S.; Kulcsár, G.; Lochau, M.; Schulze, S.: Incremental Co-Evolution of Java Programs based on Bidirectional Graph Transformation. In: PPPJ. 2015.

[Pe19] Peldszus, S.; Tuma, K.; Strüber, D.; Jürjens, J.; Scandariato, R.: Secure Data-Flow Compliance Checks between Models and Code based on Automated Mappings. In: MODELS. 2019.

[TBS19] Tuma, K.; Balliu, M.; Scandariato, R.: Flaws in Flows: Unveiling Design Flaws via Information Flow Analysis. In: ICSA. 2019.

Produktlinien und Variabilität

On Automated N-way Program Merging for Facilitating Family-based Analyses of Variant-rich Software

Dennis Reuling[1] Udo Kelter[1] Johannes Bürdek[2] Malte Lochau[2]

Abstract: In this work, we report about research results initially published in ACM Transactions on Software Engineering and Methodology (TOSEM), volume 28 Issue 3, 2019 [Re19]. Nowadays software comes in many different, yet similar variants, often derived from common code via clone-and-own. Family-based-analysis strategies show promising potentials for improving efficiency of quality assurance for variant-rich programs, as compared to variant-by-variant approaches. Unfortunately, these strategies require one superimposed program representation containing all program variants in a syntactically well-formed, semantically sound, and variant-preserving manner, which is hard to obtain manually in practice. In this talk, we present our methodology SiMPOSE for generating superimpositions of program variants to facilitate family-based analyses of variant-rich software. We utilize a novel N-way model-merging methodology for control-flow automaton (CFA) representations of C programs, an abstraction used by many recent software-analysis tools. To cope with the complexity of N-way merging, we use similarity-propagation to reduce the number of N-way matches and enable incremental merging of arbitrary subsets of variants. We apply our SiMPOSE tool to realistic C programs and investigate applicability and efficiency/effectiveness trade-offs of family-based program analyses. Our results reveal efficiency improvements by a factor of up to 2.6 for unit-test generation and 2.4 for model-checking under stable effectiveness, as compared to variant-by-variant.

Keywords: Program Merging, Model Matching, Variability Encoding, Quality Assurance

1 Summary

Software-product-line engineering is a comprehensive methodology to cope with the inherent complexity of variant-rich software, by making explicit common and variable parts in a family of program variants. Product-line engineering facilitates systematic *reuse* of code fragments among variants to increase productivity and quality of software development, as compared to variant-by-variant or clone-and-own approaches. Novel techniques for also lifting quality-assurance techniques (e.g., unit testing and model-checking) to variant-rich software pursue so-called *family-based* analysis strategies [Th14]. The goal is to analyze all program variants in a single run, instead of considering every variant separately one after the other. However, these techniques often require a (virtual) integration of all program variants into one *superimposed* representation satisfying several requirements: it has to be a)

[1] University of Siegen, Software Engineering Group, Siegen, Germany, dreuling@informatik.uni-siegen.de, kelter@informatik.uni-siegen.de
[2] TU Darmstadt, Real-time Systems Lab, Darmstadt, Germany, malte.lochau@es.tu-darmstadt.de, johannes.buerdek@es.tu-darmstadt.de

doi:10.18420/SE2020_14

syntactically well-formed, b) semantically sound (e.g., its functionality corresponds to the union of functionality of all variants), c) variant-preserving (e.g., meta-data for tracing back to program variants) and d) sufficiently succinct (e.g., all parts shared among variants are identified and integrated to maximize reuse). Most recent approaches either apply N-way merging to superimpose N design models instead of programs, or perform purely syntactic matching on locally restricted program fragments. The latter yields inherently imprecise (i.e., non-succinct or even unsound) merges as well as merge conflicts that are not automatically resolvable. Conversely, succinct N-way merges are, in general, not efficiently computable due to the combinatorial explosion of the number of possible matches [RC13]. Finally, existing approaches are either not variant-preserving, or utilize *compile-time variability* like #ifdef directives which is often incompatible with family-based analyses tools.

Our novel methodology SIMPOSE allows for automatically superimpose N program variants for enabling family-based analysis. SIMPOSE comprises a novel N-way model merging algorithm for control-flow automata (CFA) representations of programs. To cope with the complexity of N-way comparison, matching and merging of path-based program models like CFA, our approach uses principles of *similarity propagation* for a controllable trade-off between precision and computational effort. To meet requirements a)–d), we semantically embed variability-information as conditional CFA patterns into the merged CFA. SIMPOSE further enables *compositional* merging such that N-way program merging can be decomposed into incremental merging steps. Our SIMPOSE tool integrates our novel technique with a tool for family-based program analysis using the C model-checker CPACHECKER. Our experiments show that SIMPOSE outperforms state-of-the-art algorithms GNU Diffutils (DIFF) in terms of precision and N-way model merging (NWM) [RC13] in terms of scalability thus constituting, on average, the best efficiency/effectiveness trade-off between efficiency and effectiveness. The results further reveal efficiency improvements by a factor of up to 2.6 for unit-test generation and 2.4 for model-checking under stable effectiveness, as compared to variant-by-variant approaches.

Acknowledgments. This work was partially supported by the DFG (German Research Foundation) within the CoMoVa project (grant nr 330452222). This work was funded by the Hessian LOEWE initiative within the Software Factory 4.0 project.

Bibliography

[RC13] Rubin, J.; Chechik, M.: N-way Model Merging. In: Proceedings of the 2013 Joint Meeting on Foundations of Software Engineering. 2013.

[Re19] Reuling, Dennis; Kelter, Udo; Bürdek, Johannes; Lochau, Malte: Automated N-way Program Merging for Facilitating Family-based Analyses of Variant-rich Software. ACM Trans. Softw. Eng. Methodol., 28(3):13:1–13:59, July 2019.

[Th14] Thüm, T.; Apel, S.; Kästner, C.; Schaefer, I.and Saake, G.: A Classification and Survey of Analysis Strategies for Software Product Lines. ACM Comput. Surv., 2014.

Intention-Based Integration of Software Variants

Max Lillack[1] Stefan Stanciulescu[2] Wilhelm Hedman[3] Thorsten Berger[1] Andrzej Wasowski[4]

Abstract: This abstract summarizes our paper with the same title published at the 41st International Conference on Software Engineering (ICSE) 2019 [Li19].

Keywords: software variants; software re-engineering; software integration; software product lines

Cloning is a simple way to create new variants of a system. While cheap at first, it increases maintenance cost in the long term. Eventually, the cloned variants need to be integrated into a configurable platform. Such an integration is challenging: it involves merging the usual code improvements between the variants, and also integrating the variable code (features) into the platform. As such, variant integration differs from traditional software merging, which does not produce or organize configurable code, but creates a single system that cannot be configured into variants. In practice, variant integration requires fine-grained code edits, performed in an exploratory manner, in multiple iterations. Unfortunately, little tool support exists for integrating cloned variants.

In this work, we show that fine-grained code edits needed for integration can be alleviated by a small set of integration *intentions*—domain-specific actions declared over code snippets controlling the integration. Developers can interactively explore the integration space by declaring (or revoking) intentions on code elements. We contribute the intentions (e.g., 'keep functionality' or 'keep as a configurable feature') and the IDE tool INCLINE, which implements the intentions and five editable views that visualize the integration process and allow declaring intentions producing a configurable integrated platform. In a series of experiments, we evaluated the completeness of the proposed intentions, the correctness and performance of INCLINE, and the benefits of using intentions for variant integration. The experiments show that INCLINE can handle complex integration tasks, that views help to navigate the code, and that it consistently reduces mistakes made by developers during variant integration.

[1] Leipzig University, Germany
[2] ABB Corporate Research Center, Switzerland
[3] Chalmers | University of Gothenburg, Sweden
[4] IT University of Copenhagen, Denmark

doi:10.18420/SE2020_15

References

[Li19] Lillack, Max; Stanciulescu, Stefan; Hedman, Wilhelm; Berger, Thorsten; Wasowski, Andrzej: Intention-Based Integration of Software Variants. In: 41st International Conference on Software Engineering (ICSE). 2019.

Trace-Based Propagation of Variability Annotations[1]

Bernhard Westfechtel[2], Sandra Greiner[3]

Abstract: This contribution presents a mechanism to extend single- to multi-variant model transformations based on traces created during the transformation. The approach tackles a problem typically occurring in model-driven software product line engineering. Models are the key artifacts of such product lines and annotated with variability annotations in the case an annotative approach towards product line engineering is followed. Although model transformations are well-developed by now and a key facility when developing model-driven software, they are not capable to handle the variability annotations of product lines. Consequently, they transform modeled artifacts of a product line but ignore their annotations. We propose to propagate variability annotations a posteriori, using the traces of transformation execution. This approach is generic and may be applied to heterogeneous tools.

Keywords: Model transformation, software product line, annotative variability

1 Background

Model-Driven software product line engineering (MDPLE) combines two disciplines, *model-driven software engineering* (MDSE) and *software product line engineering* (SPLE), both aiming at increasing the level of productivity when developing software-intensive systems. On the one hand, SPLE follows the paradigms of organized reuse and variability [PBvdL05]. In annotative approaches [Ap09] software artifacts carry presence conditions, we refer to as *annotations* below, which are boolean expressions defining the visibility of an artifact in a product. Typically, feature models capture the common and varying parts of the product line. A feature configuration, providing each feature with a selection state, allows to derive customized products. On the other hand, MDSE aims at increasing the level of abstraction and automation by using models instead of source code artifacts. Model transformations are the common means to convert one model representation into another, e.g., to generate source code in a model-to-text transformation. In MDPLE, model transformations may be required in all development phases. In particular, they may become necessary when the product line is designed, for instance, with class diagrams which should be turned into source code (or its modeled representation). However, state-of-the-art model transformations cannot handle annotations attached to model elements out-of-the-box. Consequently, the product line engineer needs to annotate the target representation manually, which contradicts the aim of automation and increased productivity in MDPLE.

[1] This paper is an extended abstract of [WG18].

[2] Universität Bayreuth, Lehrstuhl für Angewandte Informatik, Universitätsstraße 30, 95440 Bayreuth, Germany
Bernhard.Westfechtel@uni-bayreuth.de

[3] Universität Bayreuth, Lehrstuhl für Angewandte Informatik, Universitätsstraße 30, 95440 Bayreuth, Germany
Sandra1.Greiner@uni-bayreuth.de

doi:10.18420/SE2020_16

2 Contribution

In order to address the problem outlined above, we propose automatic *trace-based propagation of variability annotations* [WG18]: The trace of a single-variant model transformation is used to propagate annotations from source to target model elements. This a posteriori approach is defined generically, i.e., independent of the transformation language as well as the tool environment. As a consequence, it may be integrated in heterogeneous MDPLE tool suites. We formally prove its correctness based on the commutativity criterion. It postulates that the two branches transform-filter (1) and filter-transform (2) should commute: For each feature configuration, transforming a multi-variant model and filtering the multi-variant target model afterwards (branch 1) should result in the same model as on branch 2, where the multi-variant source model is filtered first, followed by transforming the source model by the reused single-variant model transformation.

3 Results and Outlook

The commutativity proof assumes that single-variant model transformations conform to a computational model with certain properties and traces provide sufficient information to propagate annotations correctly. Current model transformation languages and tools partially violate these assumptions. Since we intend to support heterogeneous tool suites, we have not addressed the development of a new transformation language and tool guaranteeing commutativity. Rather, we are designing extensions to trace-based propagation that address violations of the assumptions underlying the commutativity proof. For example, in [GW19] we address incomplete traces, resulting in annotations of target model elements that are initially incomplete. Various algorithms are proposed to make these annotations complete.

Bibliography

[Ap09] Apel, Sven; Janda, Florian; Trujillo, Salvador; Kästner, Christian: Model Superimposition in Software Product Lines. In: Theory and Practice of Model Transformations, Second International Conference, ICMT 2009, Zurich, Switzerland, June 29-30, 2009. Proceedings. pp. 4–19, 2009.

[GW19] Greiner, Sandra; Westfechtel, Bernhard: On Determining Variability Annotations In Partially Annotated Models. In: Proceedings of the 13th International Workshop on Variability Modelling of Software-Intensive Systems, VAMOS 2019, Leuven, Belgium, February 06-08, 2019. pp. 17:1–17:10, 2019.

[PBvdL05] Pohl, Klaus; Böckle, Günter; van der Linden, Frank: Software Product Line Engineering: Foundations, Principles and Techniques. Springer, Berlin, Germany, 2005.

[WG18] Westfechtel, Bernhard; Greiner, Sandra: From Single- to Multi-Variant Model Transformations: Trace-Based Propagation of Variability Annotations. In: Proceedings of the 21th ACM/IEEE International Conference on Model Driven Engineering Languages and Systems, MODELS 2018, Copenhagen, Denmark, October 14-19, 2018. pp. 46–56, 2018.

Distance-Based Sampling of Software Configuration Spaces

Christian Kaltenecker[1] Alexander Grebhahn[2] Norbert Siegmund[3] Jianmei Guo[4] Sven Apel[5]

Abstract: Configurable software systems provide configuration options to adjust and optimize their functional and non-functional properties. However, to obtain accurate performance predictions, a *representative* sample set of configurations is required. Different sampling strategies have been proposed, which come with different advantages and disadvantages. In our experiments, we found that most sampling strategies do not achieve a good coverage of the configuration space with respect to covering relevant performance values. That is, they miss important configurations with distinct performance behavior. Based on this observation, we devise a new sampling strategy that is based on a distance metric and a probability distribution to spread the configurations of the sample set across the configuration space. To demonstrate the merits of distance-based sampling, we compare it to state-of-the-art sampling strategies on 10 real-world configurable software systems. Our results show that distance-based sampling leads to more accurate performance models for medium to large sample sets.

Keywords: Distance-Based Sampling; Configuration Sampling; Configurable Systems; Performance Modeling

1 Summary

Modern software systems can be configured by users to adapt them to specific devices, operating systems, and requirements. Configuration options often have a significant influence on non-functional properties, such as performance or energy consumption. Despite the benefits of configurability, identifying the performance-optimal configuration for a given setting is often a non-trivial task, due to the sheer size of configuration spaces [Xu15] and potential interactions among configuration options [KKB08, Ko18]. To identify the performance-optimal configuration of a configuration space, one can measure the performance of every valid configuration of the software system in a brute-force manner, which usually does not scale.

To avoid measuring all configurations, machine-learning techniques, such as multiple linear regression [Si12, Si15] and classification and regression trees [Gu13, Na17, Na18], have been used to learn a performance model based on a set of valid configurations,

[1] Saarland University, Germany
[2] adesso SE, Germany
[3] University of Weimar, Germany
[4] Alibaba Group, China
[5] Saarland University, Germany

doi:10.18420/SE2020_17

called the *sample set*. A performance model allows us to predict the performance of a configuration, and it can be used by an optimizer to determine the performance-optimal configuration [He15, Na18]. To create an accurate performance model, the sample set must be well-chosen, which is a non-trivial task especially when no domain knowledge is available. In essence, selecting a small, valid, and representative sample set is key to efficiency and accuracy of performance prediction, as performance measurements are usually costly in practice [Gu18].

Several sampling strategies have been proposed, which differ in their methods of selecting the sample set: (1) at random [GD06, Ch14], (2) by using an off-the-shelf constraint solver [He15], or (3) by aiming at a certain coverage criterion (e.g., selecting each configuration option, at least once) [Le08, JHF12, Ma13]. Naturally, all sampling strategies come with advantages and disadvantages. The main idea is often to cover the configuration space such that one obtains a *representative* sample set, which, ideally includes both influential configuration options and interactions among options relevant to performance, so that accurate performance models can be learned.

In general, a uniform coverage of the configuration space is desirable to obtain a representative sample set when no prior knowledge is available, since it tends to be unbiased when covering the configuration space. However, it is far from trivial to ensure unbiased uniformity if there are non-trivial constraints among configuration options. To achieve the goal in a light-weight way, we propose a new sampling strategy, called *distance-based sampling*, that addresses the shortcomings of existing strategies. The key idea behind distance-based sampling is to produce a sample set that covers the configuration space as uniformly as possible (or following another given probability distribution). To this end, distance-based sampling relies on a *distance metric* and assigns each configuration a *distance value*. It further relies on a *discrete probability distribution* to select configurations according to their distance values from the configuration space. It differs from other sampling strategies in that (1) it spreads the selected configurations across the configuration space according to a given probability distribution and is able to resemble the performance distribution of the whole population, (2) it does not require an analysis on the whole population, and (3) it uses a constraint solver for efficiency while avoiding locally-clustered sample sets [Ka19].

Our results demonstrate that distance-based sampling, when used in combination with a diversity optimization, leads to significantly lower error rates than state-of-the-art strategies, especially for larger sample sizes (t=2, t=3), and the predictions are more stable than solver-based sampling with respect to multiple runs using different random seeds. Our results demonstrate that, based on a distance metric and a probability distribution, we can effectively sample diverse configurations across the configuration space and without the need for a whole-population analysis, which makes random sampling unfeasible for highly configurable software systems. This work provides a new view on sampling based on probability distributions and paves the way for further research in this area. For instance, using other metrics or distributions could lead more accurate predictions or improve the prediction robustness [Ka19].

Bibliography

[Ch14] Chakraborty, Supratik; Fremont, Daniel J; Meel, Kuldeep S; Seshia, Sanjit A; Vardi, Moshe Y: Distribution-Aware Sampling and Weighted Model Counting for SAT. In: Proceedings of the Association for the Advancement of Artificial Intelligence (AAAI). AAAI Press, pp. 1722–1730, 2014.

[GD06] Gogate, Vibhav; Dechter, Rina: A New Algorithm for Sampling CSP Solutions Uniformly at Random. In: Proceedings of the International Conference on Principles and Practice of Constraint Programming (CP). Springer, pp. 711–715, 2006.

[Gu13] Guo, J.; Czarnecki, K.; Apel, S.; Siegmund, N.; Wasowski, A.: Variability-Aware Performance Prediction: A Statistical Learning Approach. In: Proceedings of the International Conference on Automated Software Engineering (ASE). IEEE, pp. 301–311, 2013.

[Gu18] Guo, Jianmei; Yang, Dingyu; Siegmund, Norbert; Apel, Sven; Sarkar, Atrisha; Valov, Pavel; Czarnecki, Krzysztof; Wasowski, Andrzej; Yu, Huiqun: Data-Efficient Performance Learning for Configurable Systems. Empirical Software Engineering, 23(3):1826–1867, 2018.

[He15] Henard, Christopher; Papadakis, Mike; Harman, Mark; Le Traon, Yves: Combining Multi-Objective Search and Constraint Solving for Configuring Large Software Product Lines. In: Proceedings of the International Conference on Software Engineering (ICSE). IEEE, pp. 517–528, 2015.

[JHF12] Johansen, Martin Fagereng; Haugen, Øystein; Fleurey, Franck: An Algorithm for Generating T-Wise Covering Arrays from Large Feature Models. In: Proceedings of the International Software Product Line Conference (SPLC). ACM, pp. 46–55, 2012.

[Ka19] Kaltenecker, Christian; Grebhahn, Alexander; Siegmund, Norbert; Guo, Jianmei; Apel, Sven: Distance-Based Sampling of Software Configuration Spaces. In: Proceedings of the International Conference on Software Engineering (ICSE). pp. 1084–1094, 2019.

[KKB08] Kim, Chang Hwan Peter; Kästner, Christian; Batory, Don S: On the Modularity of Feature Interactions. In: Proceedings of the International Conference on Generative Programming and Component Engineering (GPCE). ACM, pp. 23–34, 2008.

[Ko18] Kolesnikov, Sergiy; Siegmund, Norbert; Kästner, Christian; Grebhahn, Alexander; Apel, Sven: Tradeoffs in Modeling Performance of Highly-Configurable Software Systems. Software and Systems Modeling, 2018. Online first: `http://rdcu.be/GzLq`.

[Le08] Lei, Yu; Kacker, Raghu; Kuhn, D Richard; Okun, Vadim; Lawrence, James: IPOG/IPOG-D: Efficient Test Generation for Multi-Way Combinatorial Testing. Software Testing, Verification and Reliability, 18(3):125–148, 2008.

[Ma13] Marijan, Dusica; Gotlieb, Arnaud; Sen, Sagar; Hervieu, Aymeric: Practical Pairwise Testing for Software Product Lines. In: Proceedings of the International Software Product Line Conference (SPLC). ACM, pp. 227–235, 2013.

[Na17] Nair, Vivek; Menzies, Tim; Siegmund, Norbert; Apel, Sven: Using Bad Learners to Find Good Configurations. In: Proceedings of the Joint Meeting of the European Software Engineering Conference and the ACM SIGSOFT Symposium on the Foundations of Software Engineering (ESEC/FSE). ACM, pp. 257–267, 2017.

[Na18] Nair, Vivek; Yu, Zhe; Menzies, Tim; Siegmund, Norbert; Apel, Sven: Finding Faster Configurations using FLASH. IEEE Transactions on Software Engineering, 2018. Online first: https://arxiv.org/abs/1801.02175/.

[Si12] Siegmund, Norbert; Kolesnikov, Sergiy S; Kästner, Christian; Apel, Sven; Batory, Don S; Rosenmüller, Marko; Saake, Gunter: Predicting Performance via Automated Feature-Interaction Detection. In: Proceedings of the International Conference on Software Engineering (ICSE). IEEE, pp. 167–177, 2012.

[Si15] Siegmund, Norbert; Grebhahn, Alexander; Apel, Sven; Kästner, Christian: Performance-Influence Models for Highly Configurable Systems. In: Proceedings of the Joint Meeting of the European Software Engineering Conference and the ACM SIGSOFT Symposium on the Foundations of Software Engineering (ESEC/FSE). ACM, pp. 284–294, 2015.

[Xu15] Xu, Tianyin; Jin, Long; Fan, Xuepeng; Zhou, Yuanyuan; Pasupathy, Shankar; Talwadker, Rukma: Hey, You Have Given Me Too Many Knobs!: Understanding and Dealing with Over-Designed Configuration in System Software. In: Proceedings of the Joint Meeting of the European Software Engineering Conference and the ACM SIGSOFT Symposium on the Foundations of Software Engineering (ESEC/FSE). ACM, pp. 307–319, 2015.

Agile Development

Behavior-Driven Dynamics in Agile Development: The Effect of Fast Feedback on Teams

Fabian Kortum[1], Jil Klünder[1], Kurt Schneider[1]

Abstract: This paper with the title *"Behavior-Driven Dynamics in Agile Development: The Effect of Fast Feedback on Teams"* was published as full paper [KKS19] in the proceedings of the International Conference on Software and System Processes (ICSSP) in 2019.

Agile software development teams strive for fast and continuous feedback. Both the quality of the resulting software and the performance of the team require feedback. The performance of development teams is often addressed in retrospectives, which are not only part of the SCRUM framework, but also in various customized development processes. Reflecting on incidents during the last sprint helps the team to increase its performances, expressed by, e.g., efficiency and productivity. However, it is not only essential to identify volatile sprint performances, but also to characterize the root causes. The main reasons for low performance are often not visible, in particular when they are related to social-driven team behavior, such as communication structures, mood, or satisfaction. In this paper, we analyze whether automated team feedback about retrospective sprint-behavior can help the team to increase performances by additional awareness about the dynamic effects over time. In a comparative case study with 15 software projects and a total of 130 undergraduate students, we investigated the sustainable impact of feedback on human aspects. Our results indicate that automated feedback positively affects team performances – and customer satisfaction.

Keywords: Team dynamics; Agile; Retrospectives; Pro-active feedback; Information transparency

1 Introduction

The performance of agile software development teams mainly bases on effective processes, a culture of team balance, trust as well as constructive feedback. The study describes an approach that supports proactive feedback realized in JIRA involving the development team's behavioral dynamics during sprints. Investigations of human factors are actively in need of understanding the development and organizational team performances better. Awareness of potential dependencies with a focus on productivity and social-driven team factors is essential. While progress metrics in agile projects are simple to trace, e.g., by using issue tracking software like JIRA, the human factors are often difficult to capture and interpret. In an iterative knowledge discovery process, system-aided feedback support in ongoing projects is realized as an integrated plugin-solution for JIRA. The team-specific feedback

[1] Leibniz University Hannover, Software Engineering Group, Welfengarten 1, 30167 Hannover, Germany, (fabian.kortum,jil.kluender,kurt.schneider)@inf.uni-hannover.de

doi:10.18420/SE2020_18

comes along with visualization graphs about interdependencies and textual implications according to team dynamics and development performances during sprints.

2 Methodology

The plugin was evaluated in a case study with 15 student software projects over four consecutive Sprint iterations. For comparability, some teams could actively access the *ProDynamics* features. We investigated whether the system-aided feedback involving behavior dynamics and progress metrics showed positive or negative effects for the organizational structures and development performances over time. *ProDynamics* integrates the elicitation of human factors through a self-adapting survey that grants feedback implications after each Sprint. Weekly gathered information about communications, meetings, and emotional metrics, together with standard development metrics, were automatically analyzed, visualized, textually implicated, and made available within the projects. Customers and Scrum master provided additional feedback about their perceived team and development performances. Teams without access only received insights from Sprint reports in JIRA.

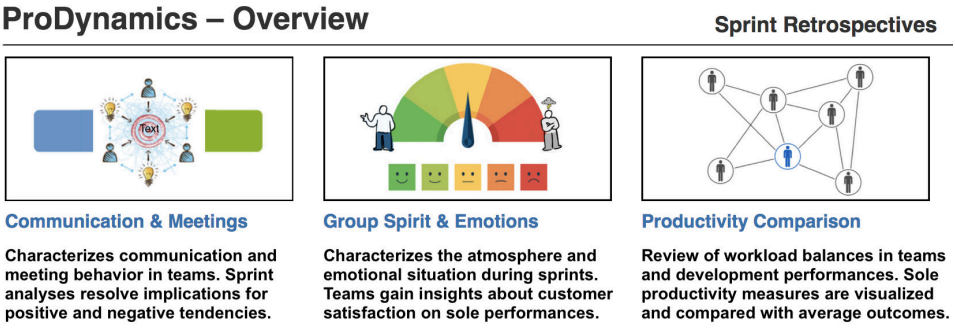


Fig. 1: *The ProDynamics JIRA-Plugin:* Enables Sprint Feedback on the Human Factors in Teams

3 Research Results

The study results are statistically significant and revealed that the groups with access to the additionally provided *ProDynamics* feedback had a consecutive increase in the development performances over time. Meanwhile, teams without access tend for more error-gaps when estimating Sprints. The study shows that 68% of 130 developers believe that team feedback takes a crucial position in agile software developments, while 59% of the *ProDynamics* participants noticed a supportive effect in Sprints. Therefore, the plugin simplifies complex data elicitation and activates awareness in teams also for human factors in software projects.

References

[KKS19] Kortum, Fabian; Klünder, Jil; Schneider, Kurt: Behavior-driven Dynamics in Agile Development: The Effect of Fast Feedback on Teams. In: Proceedings of the International Conference on Software and System Processes. ICSSP '19, IEEE Press, Piscataway, NJ, USA, pp. 34–43, 2019.

How has SPI changed in times of agile development? Results from a multi-method study

Steffen Küpper[1], Dietmar Pfahl[2], Kristjan Jürisoo[3], Philipp Diebold[4], Jürgen Münch[5], Marco Kuhrmann[6]

Abstract: The emergence of agile methods and practices has not only changed the development processes but might also have affected how companies conduct software process improvement (SPI). Through a set of complementary studies, we aim to understand how SPI has changed in times of agile software development. Specifically, we aim (1) to identify and characterize the set of publications that connect elements of agility to SPI, (2) to explore to which extent agile methods/practices have been used in the context of SPI, and (3) to understand whether the topics addressed in the literature are relevant and useful for industry professionals. To study these questions, we conducted an in-depth analysis of the literature identified in a previous mapping study, an interview study, and an analysis of the responses given by industry professionals to SPI-related questions stemming from an independently conducted survey study.

This summary refers to the paper *How has SPI changed in times of agile development? Results from a multi-method study* [Kü19]. This paper was published as research article (empirical) in the *Journal of Software: Evolution and Process*.

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

1 Introduction

Software development processes have adopted many elements from agile methods and practices. That is, processes have become more agile and more hybrid. This brings up the question whether *software process improvement* (SPI) has become more agile as well. Do companies today still use standardized SPI frameworks like CMMI? Is SPI conducted in the form of projects with defined start and end dates, or is it a continuously performed activity that adapts to short-term goals and needs? Is SPI itself undergoing a change towards adopting an agile flavor, represented by short improvement cycles, adaptive improvement goals and flexible improvement actions?

[1] Clausthal University of Technology, Germany, steffen.kuepper@tu-clausthal.de
[2] University of Tartu, Estonia, dietmar.pfahl@ut.ee
[3] University of Tartu, Estonia, k.jyrisoo@gmail.com
[4] Bagilstein GmbH, Germany, kontakt@bagilstein.de
[5] Reutlingen University, Böblingen, Germany, Juergen.Muench@Reutlingen-University.de
[6] University of Passau, Germany, kuhrmann@acm.org

doi:10.18420/SE2020_19

Problem Statement & Objective To answer the questions above, it is important to understand what experience exists regarding SPI in the context of agile and hybrid development processes. We aim to study the current state of the art and practice. We aim to better understand how agile methods/practices are used in the context of planning and conducting SPI.

Contribution The paper at hand presents contributes findings from a multitude-method research. Based on a systematic mapping study, we selected 55 publications for a systematic literature review. The complement the secondary studies, we conducted an expert interview with seven companies in Estonia and backed up the findings with data obtained in the first stage of the HELENA study [Ku17].

2 Results

We identified 55 publications that focus on both SPI and agility of which 48 present and discuss how agile methods/practices are used to steer SPI initiatives. We found that the two most frequently mentioned agile methods in the context of SPI are Scrum and Extreme Programming (XP), while the most frequently mentioned agile practices are integrate often, test-first, daily meeting, pair programming, retrospective, on-site customer, and product backlog. Furthermore, we found that a majority of the interviewed and surveyed industry professionals see SPI as a continuous activity. They agree with the agile SPI literature that agile methods and practices play an important role in SPI activities but that the importance given to specific agile methods and practices does not always coincide with the frequency with which these methods and practices are mentioned in the literature.

3 Conclusion & Future Work

Our paper [Kü19] indicates that the trend towards agile SPI is visible both in the research literature and in surveys and interviews with industry professionals. Differences between these data sources exist regarding the prominence of specific agile development methods and practices in the context of SPI and the focus on implicit and continuous SPI versus explicit and strategic SPI.

References

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

[Kü19] Küpper, Steffen; Pfahl, Dietmar; Jürisoo, Kristjan; Diebold, Philipp; Münch, Jürgen; Kuhrmann, Marco: How has SPI changed in times of agile development? Results from a multi-method study. Journal of Software: Evolution and Process, 31(11):1–28, 2019.

Attitudes, Beliefs, and Development Data Concerning Agile Software Development Practices

Christoph Matthies [1], Johannes Huegle[2], Tobias Dürschmid [3], Ralf Teusner [4]

Abstract: This work is a summary of research previously published at the 41st International Conference on Software Engineering: Software Engineering Education and Training in 2019. The perceptions and attitudes of developers impact how software projects are run and which development practices are employed in development teams. Recent Agile methodologies have taken this into account, focusing on collaboration and a shared team culture. In this research, we investigate the perceptions of Agile development practices and their usage in Scrum software development teams. Although perceptions collected through surveys of 42 participating students did not evolve significantly over time, our analyses show that the Scrum role significantly impacted participants' views of employed development practices. We find that using the version control system according to Agile ideas was consistently rated most related to the values of the Agile Manifesto. Furthermore, we investigate how common software development artifacts can be used to gain insights into team behavior and present the development data measurements we employed. We show that we can reliably detect well-defined Agile practices, such Test-Driven Development, in this data and that usage of these practices coincided with participants' self-assessments.

Keywords: Agile software development, software engineering

1 Research Context

As Software Engineering is an activity conducted by humans, developers' perceptions, beliefs, and attitudes towards software engineering practices significantly impact the development process.In terms of modern software engineering, Agile software development methodologies, such as Scrum, have highlighted the importance of people, collaboration and teamwork. Scrum, currently the most popular Agile software development methodology employed in industry ,has been described as a process framework that is designed to manage work on complex products .

Software developers attitudes towards certain Agile development practices stem mainly from applying these practices in software projects . It is these attitudes towards Agile practice application and developers' perceptions of them, that we aim to study. While

[1] Hasso Plattner Institute, University of Potsdam, christoph.matthies@hpi.de
[2] Hasso Plattner Institute, University of Potsdam, johannes.huegle@hpi.de
[3] Carnegie Mellon University, Pittsburgh, duerschmid@cmu.edu
[4] Hasso Plattner Institute, University of Potsdam, ralf.teusner@hpi.de

doi:10.18420/SE2020_20

human factors are integral to the software development process, the main goal remains to produce a product that serves the intended purpose. To this end, a large variety of primary artifacts, such as executable code and documentation as well as secondary, supportive development artifacts, such as commits in a version control repository or user stories containing requirements, are created. All the data produced by software engineers on a daily basis, as part of regular development activities, is empirical evidence of the way that work is performed and represents a valuable source of actionable information . Combining and contrasting the perceptions of developers with the empirical evidence gathered from project data can yield insights into development teams not possible by relying on only a single one of these aspects.

2 Method

In the context of an undergraduate collaborative software engineering course employing Agile methodologies, which has become standard practice in universities , we study the usage and application of Agile practices by developers and other Scrum roles based on the software project data they create and the related perceptions of these practices. We developed a set of survey claims concerning Agile practice application to collect these assessments and present the results of the survey. Furthermore, we developed a set of six development data measures based on non-intrusively collected software development artifacts, which allow insights into team behaviors.

3 Results

We show that the concepts of Collective Code Ownership, usage of the version control system in line with Agile ideas, and not working at the last minute, correlated with high self-assessments of Agile value application. We show that measurements regarding Test-Driven-Development and last minute work correlate with corresponding self-assessments. These findings highlight areas where assumptions of project team work were validated as well as those areas where perceptions of Agile practices and measurements diverged. These represent opportunities for further investigation. In line with related research , we consider translating development practices into workable definitions and measures as one of the biggest challenges and opportunities. By sharing our development data measurements and their background in detail we hope to take another step towards this goal.

Feature-Modellierung

Principles of Feature Modeling

Damir Nešić[1] Jacob Krüger[2] Ștefan Stănciulescu[3] Thorsten Berger[4]

Abstract: This abstract summarizes our paper with the homonymous title published at the Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering (ESEC/FSE) 2019 [Ne19].

Keywords: feature models; modeling principles; software product lines

A software product line is a family of system variants in an application domain, typically engineered as an integrated software platform that allows to reuse and configure each system variant based on a built-in variability mechanism. As the *features* (and their dependencies) of all system variants are part of a single software platform, product lines are inherently complex. To document and manage this complexity, feature models have become one of the most successful notations. They help organizations to keep an understanding of the platform, and support scoping, planning, development, variant derivation, configuration, and maintenance activities. Still, feature models are challenging to build and maintain: While feature models have been researched for three decades, resulting in various feature-modeling notations as well as automated analysis and configuration techniques, a generic set of modeling principles for building and evolving feature models is still missing. Actually, it is not even certain whether feature models could be engineered based on recurrent principles.

To tackle this problem, we analyzed feature-modeling practices that we triangulated from two different sources. First, we conducted a systematic literature review, based on which we included 31 papers that describe such practices. Second, we interviewed ten industrial practitioners on how they and their organizations build and evolve feature models. As a result, we elicited 34 principles by reading through nearly 190 instances of practices that we identified in the papers and interview transcripts. These principles cover eight phases of feature modeling: planning and preparation (6), training (3), information sources (1), model organization (5), modeling (11), dependencies (2), quality assurance (3), as well as model maintenance and evolution (3). Grounded in empirical evidence, these principles provide practical, context-specific advice on how to perform feature modeling, describe what information sources to consider, and highlight common characteristics of feature models. The principles are a first step towards our overarching goal of defining a feature-modeling process and help practitioners as well as researchers to understand best practices.

[1] KTH Royal Institute of Technology, Stockholm
[2] Otto-von-Guericke-University, Magdeburg
[3] ABB Corporate Research, Baden-Dättwil
[4] Chalmers | University of Gothenburg

doi:10.18420/SE2020_21

While analyzing and discussing the principles, we obtained the following insights:

- Most papers report on experiences of *extracting* a product line from existing systems. Therefore, the practices and the principles identified mostly relate to bottom-up feature modeling.
- Some of our principles are applicable only in specific scenarios (e.g., depending on the purpose of the feature model) or represent alternative solutions to the same issue (e.g., splitting a feature model or providing views on it).
- Applying any principle comes with pros, cons, and dependencies to other principles. For example, using a single feature model facilitates its governance and avoids interface models. However, this will also result in a higher depth of the feature model and more complex cross-tree constraints, which challenges maintenance activities.
- Planning and training activities are rarely reported in our sources, and mainly in interviews with tool-vendors or in industrial case studies. So, while feature modeling is considered intuitive, there are apparent efforts for training to support an organization in defining the scope and purpose of its feature model.
- The principles we identified are based on papers comprising ten different feature-model notations employed in 14 domains. Still, none of the principles requires to rely on a specific notation or is applicable only to a specific domain, indicating that the principles can be applied by any organization.

The principles and insights we describe in our paper are highly valuable for practitioners to build and evolve feature models, and for researchers to design supportive techniques. Currently, we ordered the principles in the most reasonable sequence of steps for creating a feature model, which does not represent an actual modeling process. For instance, some principles are alternatives, depend on each other, can be optional or depend on an organization's context. So, as future work, we aim to further evaluate the principles considering their applicability, to synthesize an executable modeling process, and to validate this process with our industrial partners.

Bibliography

[Ne19] Nešić, Damir; Krüger, Jacob; Stănciulescu, Ștefan; Berger, Thorsten: Principles of Feature Modeling. In: Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering. ESEC/FSE. ACM, pp. 62–73, 2019.

Effects of Explicit Feature Traceability on Program Comprehension

Jacob Krüger[1] Gül Çalıklı[2] Thorsten Berger[2] Thomas Leich[3] Gunter Saake[1]

Abstract: This abstract is based on our paper with the homonymous title published at the Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering (ESEC/FSE) 2019 [Kr19].

Keywords: program comprehension; feature traceability; software maintenance; separation of concerns

Software developers spend a substantial amount of their effort in software development and maintenance on program comprehension. While numerous artifacts can support program comprehension, for instance, documentations or models, developers mainly focus on understanding the actual source code. In this regard, a particular challenge for developers is to locate the features of their system in the source code, which is needed to perform any update, bug fix or extension on any feature. So, it is crucial to improve the comprehensibility of the source code itself, with various studies showing that even small improvements can have strong effects. One particular technique for improving program comprehension are explicit feature traces in the source code to facilitate feature location. Mainly, feature traces are instantiated using one or both of the following concrete techniques:

- *Annotations* represent a virtual separation of features, marking the begin and end of code that belongs to a feature within a single code base.
- *Decomposition* refers to physically separating features, for example, by implementing features in separate classes or feature modules that are separated from the base code.

While both techniques have been investigated in several studies, which highlight different pros and cons of either technique, the actual effect of feature traces on program comprehension remains unclear and requires more detailed evidence. In particular, most studies solely compare between both techniques, but do not compare them to source code without traces—often assuming that either technique is beneficial to use anyway.

In our paper, we tackled this problem with an empirical study, comprising an experimental and a survey part, in which we compared annotations, decomposition, and pure object-oriented code of the same system. To improve the motivation and gain more reliable insights,

[1] Otto-von-Guericke-University, Magdeburg
[2] Chalmers | University of Gothenburg
[3] Harz University of Applied Sciences & METOP GmbH, Wernigerode & Magdeburg

doi:10.18420/SE2020_22

we personally invited 144 professional developers from various organizations and countries to participate in our online study. We obtained results for 49 of these software developers, analyzing their (1) effectiveness in correctly solving three program-comprehension and three bug-localization tasks, (2) efficiency in terms of time needed to perform these tasks, and (3) subjective perception of explicit feature traces.

Overall, we observed the following effects and perceptions of explicit feature traces:

- Annotations did significantly improve the comprehension of feature interactions compared to object-oriented source code for two tasks. Our participants' responses suggest that annotations allowed them to focus on the correct code parts and identify interactions more easily, conflicting some common believes that annotations hamper program comprehension.
- Decomposition did significantly hamper bug localization compared to object-oriented code for one task. While researchers often argue that physically separated features are easier to comprehend, our participants could not identify the correct interaction between a feature and the base code that caused the bug.
- Concerning the survey responses of our participants, we found that explicit feature traces (i) extend program-comprehension strategies, (ii) do not impair program comprehension, and (iii) are perceived as positive. So, we argue that explicit feature traces, and especially annotations, are a simple, yet effective, technique to improve program comprehension. However, there are open challenges, concerning the granularity of feature traces and their maintenance.

None of the three source-code versions resulted in significant changes on task completion time. Our results confirm most works that compare between annotations and decomposition, yielding no significant differences. However, we also showed that it is problematic to not compare against the baseline of source code without any traces, for which we observed conflicting results compared to common believes in software-engineering research. In summary, our results indicate that lightweight feature traces, such as annotations, provide immediate benefits to developers while developing and maintaining software. Moreover, lightweight annotations do not require extensive training or tooling, making it simpler to introduce them compared to heavyweight traceability tools (e.g., DOORS). Our future work will focus on further analyzing especially the effect of feature annotations, and on developing tooling to support their introduction, maintenance, and analysis.

Bibliography

[Kr19] Krüger, Jacob; Çalıklı, Gül; Berger, Thorsten; Leich, Thomas; Saake, Gunter: Effects of Explicit Feature Traceability on Program Comprehension. In: Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering. ESEC/FSE. ACM, pp. 338–349, 2019.

A Study of Feature Scattering in the Linux Kernel

Leonardo Passos[1], Rodrigo Queiroz[1], Mukelabai Mukelabai[2], Thorsten Berger[2], Sven Apel[3], Krzysztof Czarnecki[1], Jesus Alejandro Padilla[1]

Abstract: This abstract summarizes our paper with the same title accepted (November 2018) for the journal IEEE Transactions on Software Engineering (TSE) [Pa18]. The paper is a journal extension of a previous conference paper [Pa15], which won a best paper award.

Keywords: feature scattering; software modularity; longitudinal study; Linux kernel

Feature code is often scattered across a software system. Scattering is not necessarily bad if used with care, as witnessed by systems with highly scattered features that evolved successfully. Feature scattering, often realized with a pre-processor, circumvents limitations of programming languages and software architectures. Unfortunately, little is known about the principles governing scattering in large and long-living software systems.

We present a longitudinal study of feature scattering in the Linux kernel, complemented by a survey with 74, and interviews with nine Linux kernel developers. We analyzed almost eight years of the kernel's history, focusing on its largest subsystem: device drivers. We learned that the ratio of scattered features remained nearly constant and that most features were introduced without scattering. Yet, scattering easily crosses subsystem boundaries, and highly scattered outliers exist. Scattering often addresses a performance-maintenance tradeoff (alleviating complicated APIs), hardware design limitations, and avoids code duplication. While developers do not consciously enforce scattering limits, they actually improve the system design and refactor code, thereby mitigating pre-processor idiosyncrasies or reducing its use.

References

[Pa15] Passos, Leonardo; Padilla, Jesus; Berger, Thorsten; Apel, Sven; Czarnecki, Krzysztof; Valente, Marco Tulio: Feature Scattering in the Large: A Longitudinal Study of Linux Kernel Device Drivers. In: 14th International Conference on Modularity (MODULARITY). 2015.

[Pa18] Passos, Leonardo; Queiroz, Rodrigo; Mukelabai, Mukelabai; Berger, Thorsten; Apel, Sven; Czarnecki, Krzysztof; Padilla, Jesus: A Study of Feature Scattering in the Linux Kernel. IEEE Transactions on Software Engineering, 2018. Preprint.

[1] University of Waterloo, Canada
[2] Chalmers | University of Gothenburg, Sweden
[3] Saarland University, Germany

doi:10.18420/SE2020_23

Wartung und Evolution 1

Generating Accurate and Compact Edit Scripts using Tree Differencing

Veit Frick, Thomas Grassauer, Martin Pinzger[1], Fabian Beck[2]

Abstract: For analyzing changes in source code, edit scripts are used to describe the differences between two versions of a file. These scripts consist of a list of actions that, applied to the source file, result in the new version of the file. In contrast to line-based source code differencing, tree-based approaches such as GumTree, MTDIFF, or ChangeDistiller extract changes by comparing the abstract syntax trees (AST) of two versions of a source file. One benefit of tree-based approaches is their ability to capture moved (sub)trees in the AST. Our approach, the Iterative Java Matcher (IJM), builds upon GumTree and aims at generating more accurate and compact edit scripts that capture the developer's intent. This is achieved by improving the quality of the generated move and update actions, which are the main source of inaccurate actions generated by previous approaches. To evaluate our approach, we conducted a study with 11 external experts and manually analyzed the accuracy of 2400 randomly selected edit actions. Comparing IJM to GumTree and MTDIFF, the results show that IJM provides better accuracy for move and update actions and is more beneficial to understanding the changes.

Keywords: Change Extraction; Tree Differencing; Abstract Syntax Trees; Software Evolution

Edit scripts describe the differences between two versions of a source code file. Such scripts consist of a list of actions that, when applied to a given source code file, correctly transfers it from one version of that file to another. While edit scripts gnerated by line-based differencing algorithms are quickly generated and provide an overview, they are coarse grained and do not consider syntax information. Approaches such as GumTree [Fa14] and MTDIFF [DP16], compare the ASTs parsed from the source code files instead. This allows the algorithms to refine the granularity down to the level of single AST nodes.

Existing state-of-the-art approaches generate edit scripts that are correct in the sense of transforming one AST into another. However, in a manual investigation, we found that those edit scripts can consist of actions, especially *move* and *update* actions, that can be classified as inaccurate. Consider an edit script where every node of the original AST is deleted and every node of the new AST is inserted. This edit script always correctly transfers the original AST into the new AST, even if both ASTs are exactly the same. The actions of such an edit script would be correct but not accurate. We propose the following definition of an accurate edit action: An accurate action has to fulfill all of the following three criteria: The action has to be comprehensible, helpful, and the most simple solution. We found that over 55% of GumTree's and over 81% of MTDIFF's generated move and update actions are inaccurate

[1] Alpen-Adria-Universität Klagenfurt, SERG, vorname.nachname@aau.at
[2] University of Duisburg-Essen, fabian.beck@paluno.uni-due.de

doi:10.18420/SE2020_24

according to our definition given above. We argue that such a high misclassification rate significantly impacts the understanding of code changes, in particular of moved and updated code. Our approach is based on the GumTree approach and aims at improving the *matching* phase. We use the existing implementation of the algorithm by Chawathe et al. [Ch96] to create the final edit scripts. For improving the matching of AST nodes and subtrees, we add three matching strategies:

Partial Matching: We restrict the scope for the matching to selected parts of the source code, each represented by a subtree in the AST. We assume that most of the changes happen within such a subtree and only few changes happen between them. For instance, the source code of a method is more likely to be changed and moved within the same method. The partial matching approach is therefore to divide the AST into smaller parts that are individually matched. IJM is built as a combination of different matchers.

Merged Name Nodes: Name nodes are children of various other nodes like method or type declarations containing the name of their parent node as value. In order to prevent these nodes from beeing matched with other name nodes, belonging to different parents, IJM merges the name node together with their parent into one atomic node. This reduces mismatched name nodes and shortens the AST.

Name Aware Matching: GumTree, in its bottom-up phase, only uses the node type to determine whether or not two nodes can be matched. IJM addresses this by adding name-awareness to the bottom-up phase of GumTree. This is realized by considering the similarity of the names of the nodes in addition to their node types.

We evaluated and compared the accuracy and helpfulness of IJM to the state of the art approaches GumTree and MTDIFF. A study with 2400 randomly selected edits shows a higher accuracy for move and update actions without increasing the misclassification rate for insert and delete actions. A study with 11 independent external experts showed that they found more helpful for understanding the changes in a revision. Furthermore, an evaluation on 10 Java open source projects shows no increase in edit script size and runtime.

References

[Ch96] Chawathe, Sudarshan S.; Rajaraman, Anand; Garcia-Molina, Hector; Widom, Jennifer: Change Detection in Hierarchically Structured Information. In: Proceedings of the ACM SIGMOD International Conference on Management of Data. SIGMOD '96, ACM, New York, NY, USA, pp. 493–504, 1996.

[DP16] Dotzler, G.; Philippsen, M.: Move-optimized source code tree differencing. In: Proceedings of the 31st IEEE/ACM International Conference on Automated Software Engineering. ASE '16, pp. 660–671, Sept 2016.

[Fa14] Falleri, Jean-Rémy; Morandat, Floréal; Blanc, Xavier; Martinez, Matias; Monperrus, Martin: Fine-grained and Accurate Source Code Differencing. In: Proceedings of the 29th ACM/IEEE International Conference on Automated Software Engineering. ASE '14, ACM, New York, NY, USA, pp. 313–324, 2014.

The List is the Process: Reliable Pre-Integration Tracking of Commits on Mailing Lists

Ralf Ramsauer[1], Daniel Lohmann[2], Wolfgang Mauerer[3]

Abstract: International Conference on Software Engineering 2019, Technical Track. Artifact evaluation: *available* and *evaluated*. Code+Data available as reproducible OSS.

Research on software evolution often focuses on mining changes in software repositories, but omits their pre-integration history.

Our work allows for tracking this invisible evolution of software changes on mailing lists (used by many low-level system components) by connecting all early revisions of changes to their final version in repositories. Since only updates to fragments (i.e., patches) are available, identifying semantically similar changes is a non-trivial task that we solve language-independently. We evaluate our method on high-profile OSS projects like the Linux kernel, and validate its high accuracy using an elaborately created ground truth.

Our approach can quantify properties of OSS development processes, which is an essential requirement for using OSS in reliable or safety-critical industrial products. We can also quantitatively determine if an open development process effectively aligns with given formal process requirements.

We will report on integration of our results in workflows of the Linux Foundation, one of the largest non-profit technology consortia for open source, and joint work with the automotive industry to deploy our results in practise.

This work was supported by Siemens AG, Corporate Research, the iDev40 project and the German Research Council (DFG) under grant no. LO 1719/3-1.The iDev40 project has received funding from the ECSEL Joint Undertaking (JU) under grant no. 783163. The JU receives support from the European Union Horizon 2020 research and innovation program. It is co-funded by the consortium members, grants from Austria, Germany, Belgium, Italy, Spain and Romania.

[1] Ostbayerische Technische Hochschule Regensburg, ralf.ramsauer@oth-regensburg.de
[2] Leibniz Universität Hannover, lohmann@sra.uni-hannover.de
[3] Ostbayerische Technische Hochschule Regensburg, wolfgang.mauerer@oth-regensburg.de

doi:10.18420/SE2020_25

On Controlling the Attack Surface of Object-Oriented Refactorings

Sebastian Ruland[1], Géza Kulcsár[1], Erhan Leblebici[1], Sven Peldszus[2], Malte Lochau[1]

Abstract: The results of this work have originally been published in [Ru18]. Refactorings constitute an effective means to improve quality and maintainability of evolving object-oriented programs. Search-based techniques have shown promising results in finding near-optimal sequences of behavior-preserving program transformations that (1) maximize code-quality metrics and (2) minimize the number of code changes. However, the impact of refactorings on non-functional properties like security has received little attention so far. To this end, we propose, as a further objective, to minimize the *attack surface* of object-oriented programs (i.e., to maximize strictness of declared accessibility of class members). Minimizing the attack surface naturally competes with applicability of established refactorings like *MoveMethod*, frequently used for improving code quality in terms of coupling/cohesion measures. Our tool implementation is based on an EMF meta-model for JAVA-like programs and utilizes *MOMoT*, a search-based model-transformation and optimization framework. Our experimental results gained from applying different accessibility-control strategies to a collection of real-world JAVA programs show the impact of attack surface minimization on design-improving refactorings. We further compare the results to those of existing refactoring tools.

Keywords: Object-Oriented Refactorings; Search-based Refactorings; Attack Surface; Model Transformation

1 Summary

One major challenge in designing object-oriented programs is to solve the *Class-Responsibility-Assignment (CRA)* problem, by identifying candidates for classes and assigning related *responsibilities* (i.e., program data and operations) to them. In modern software development practices, especially in agile development, these responsibilities frequently change over time. Therefore, program-evolution phases are accompanied by refactorings (i.e., behavior-preserving code transformations) to update CRA decisions. Hence, refactorings help to maintain or even improve code quality over time [Fo00]. In this regard, the different, and often contradicting, objectives of refactorings are (1) to optimize code-quality metrics (e.g., coupling and cohesion or anti-pattern avoidance), and (2) to preserve the initial program design as much as possible, by minimizing the number of changes executed by the refactoring [Ca16]. A manual search for refactorings constituting

[1] TU Darmstadt, Real-Time Systems Lab, Magdalenenstr. 4, 64289 Darmstadt, Germany, sebastian.ruland@es.tu-darmstadt.de, malte.lochau@es.tu-darmstadt.de
[2] University of Koblenz-Landau, Institute for Software Technology, Universitätsstraße 1, 56070 Koblenz, Germany, speldszus@uni-koblenz.de

doi:10.18420/SE2020_26

a reasonable trade-off between both objectives is often impractical in case of real-world programs, as each refactoring consists of sequences of interdependent code transformations accompanied by complex applicability constraints. Due to the large search space and the multitude of contradicting objectives, search-based optimization techniques constitute a promising approach to find applicable refactorings. There exists recent work considering various semi-automated approaches for specific refactorings, where validity of possible refactorings is mostly concerned with purely functional behavior preservation [Ou16].

In contrast, our proposed methodology, called GOBLIN, further takes into account *attack-surface metrics* as well-known indicator for program security. In particular, the *attack surface* of a program comprises all conventional ways of entering a software by users/attackers. Therefore, a large attack surface increases the danger of vulnerability exploitation. Hence, we consider minimization of the attack surface (i.e., granting least privileges to class members) as an additional *non-functional* optimization objective during refactoring. To this end, GOBLIN utilizes search-based optimization techniques aiming to find (near-)optimal sequences of refactorings for JAVA-like programs, where we additionally take accessibility constraints into account. We apply already established refactoring operations, namely *move-method* refactorings, and optimize different code-quality metrics as well as the impact on the attack surface. Thus, we have to strengthen or weaken accessibility declarations on demand to ensure correctness of accessibility modifiers after refactoring. As further objectives for code optimization, we consider *elimination of design flaws* (i.e., by optimizing coupling and cohesion measures and avoiding anti-patterns such as the *The Blob*)and *preservation of the original program design* (i.e., minimizing the number of changes). We evaluated our approach by applying GOBLIN to a collection of real-world programs. Besides showing the applicability of our tool, the evaluation results further provide in-depth insights into the interplay between traditional code-quality metrics and attack-surface metrics.

Acknowledgments. This work was funded by the Hessian LOEWE initiative within the Software-Factory 4.0 project.

Bibliography

[Ca16] Candela, Ivan; Bavota, Gabriele; Russo, Barbara; Oliveto, Rocco: Using Cohesion and Coupling for Software Remodularization: Is It Enough? ACM Trans. Softw. Eng. Methodol., 25(3):24:1–24:28, June 2016.

[Fo00] Fowler, Robert: Refactoring: Improving the Design of Existing Code. Addison-Wesley, 2000.

[Ou16] Ouni, Ali; Kessentini, Marouane; Sahraoui, Houari A.; Inoue, Katsuro; Deb, Kalyanmoy: Multi-Criteria Code Refactoring Using Search-Based Software Engineering: An Industrial Case Study. ACM Trans. Softw. Eng. Methodol., 25(3):23:1–23:53, 2016.

[Ru18] Ruland, Sebastian; Kulcsár, Géza; Leblebici, Erhan; Peldszus, Sven; Lochau, Malte: Controlling the Attack Surface of Object-Oriented Refactorings. In: Fundamental Approaches to Software Engineering. Springer International Publishing, Cham, pp. 38–55, 2018.

Software Intelligence und Enterprise Cloud

Methodological Principles for Reproducible Performance Evaluation in Cloud Computing

Alessandro V. Papadopoulos[1] Laurens Versluis[2] André Bauer[3] Nikolas Herbst[3] Jóakim von Kistowski[3] Ahmed Ali-Eldin[4] Cristina L. Abad[5] José Nelson Amaral[6] Petr Tůma[7] Alexandru Iosup[2]

Abstract: In this work, we report on our recent article on "Methodological Principles for Reproducible Performance Evaluation in Cloud Computing" published in the IEEE Transactions on Software Engineering (TSE)[Pa19]. This publication is a result of a joint effort[8] coordinated by the Cloud Research Working Group of SPEC (Standard Performance Evaluation Corporation).

Keywords: IEEE Keywords: Cloud computing, Performance evaluation, Benchmark testing, Systematics, Computer performance, Software engineering

Author Keywords: Experimental evaluation, observation study, experimentation

Summary

The rapid adoption and the diversification of cloud computing technology exacerbate the importance of a sound experimental methodology for this domain. It has been a decade since the first commercial cloud has opened for a general public (Amazon AWS, in 2007). Cloud computing systems are now in much demand, and command significant industrial and scientific interest. Although scientific progress and industry growth depend on using sound principles for measuring and reporting cloud-system performance, this process remains complex.

This work investigates how to measure and report performance in the cloud, and how well the cloud research community is already doing it. There are several difficulties, including controlling the time-accuracy trade-off, deciding which data to collect and to report, and using the right summarization statistics in the report.

[1] Mälardalen University, Sweden, Email: alessandro.papadopoulos@mdh.se
[2] Vrije Universiteit Amsterdam, The Netherlands
[3] University of Würzburg, Germany
[4] Umeå University, Sweden
[5] Escuela Superior Politecnica del Litoral, Ecuador
[6] University of Alberta, Canada
[7] Charles University, Czech Republic
[8] https://research.spec.org/news/single-view/article/technical-report-on-reproducible-performance-evaluation-in-cloud-computing-published.html

doi:10.18420/SE2020_27

We propose a set of eight important methodological principles that combine best-practices from nearby fields with concepts applicable only to clouds, and with new ideas about the time-accuracy trade-off.

We show how these principles are applicable using a practical use-case experiment. To this end, we analyze the ability of the newly released SPEC Cloud IaaS benchmark to follow the principles, and showcase real-world experimental studies in common cloud environments that meet the principles.

Last, we report on a systematic literature review including top conferences and journals in the field, from 2012 to 2017, analyzing if the practice of reporting cloud performance measurements follows the proposed eight principles. Worryingly, this systematic survey and the subsequent two-round human reviews, reveal that few of the published studies follow the eight experimental principles. We conclude that, although these important principles are simple and basic, the cloud community is yet to adopt them broadly to deliver sound measurement of cloud environments.

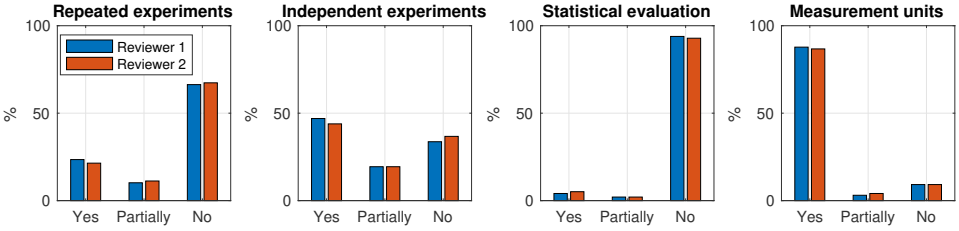


Fig. 1: Presence of selected principles in the cloud performance evaluation.

Bibliography

[Pa19] Papadopoulos, A. V.; Versluis, L.; Bauer, A.; Herbst, N.; Von Kistowski, J.; Ali-eldin, A.; Abad, C.; Amaral, J. N.; Tůma, P.; Iosup, A.: Methodological Principles for Reproducible Performance Evaluation in Cloud Computing. IEEE Transactions on Software Engineering, pp. 1–1, 2019.

Models, MDE und MDD

Strategies and Best Practices for Model-based Systems Engineering Adoption in Embedded Systems Industry

Tiago Amorim[1], Andreas Vogelsang[2], Florian Pudlitz[3], Peter Gersing[4], Jan Philipps[5]

Model-based Systems Engineering (MBSE) is the formalized application of modeling to support system requirements, design, analysis, verification and validation activities. Specialized tools automate much of the non-creative work (which translates to gains in productivity and quality) and generate code based on the models. MBSE foster artifact reuse, improves product quality, and shortens time to market [Bö14]. Despite the aforementioned benefits, adopting MBSE is a complex task, especially for larger and established companies [Vo17]. Process changes are required in all system life-cycle phases as well as a shift in the development paradigm (i.e., abstract thinking) and application of new tools. Projects are not likely to meet their cost and delivery target when adoption is carried out poorly. Our goal was to find out what was tried, what worked, what did not work, how the problems were solved, what can be recommended, and what should be avoided when adopting MBSE in organizations that develop embedded systems. For this purpose, we devised an inductive-deductive research approach in a triangulation fashion. We conducted 14 semi-structured interviews with experts from embedded systems organizations. From these interviews, we extracted 18 best practices fitted for tackling MBSE adoption challenges. Sequentially, we validated and prioritized the best practices with the help of an on-line questionnaire which was answered by MBSE practitioners. Our findings provide input for planning MBSE adoption based on the knowledge of practitioners that went through the experience of implementing MBSE in already established embedded systems development organizations.

Results: The best practices are listed in the following: **BP01:** The organization should start adopting MBSE with new projects. **BP02:** The pilot project should create real value for the organization (i.e., no didactic project). **BP03:** The pilot project should have enough budget and time allocated to bear the overhead of adoption. **BP04:** No translation of old artifacts, exception is when considering reusable artifacts. **BP05:** Start small in terms of project and team size in order to acquire some experience. **BP06:** Tools with open interfaces and homogeneous work-flow are preferred. **BP07:** All engineers should have access to the tools. **BP08:** Tool acquisition is very costly therefore should be thoroughly planned. **BP09:** Have

[1] Technische Universität Berlin, Fakultät Elektrotechnik und Informatik, Germany buarquedeamorim@tu-berlin.de
[2] Technische Universität Berlin, Fakultät Elektrotechnik und Informatik, Germany andreas.vogelsang@tu-berlin.de
[3] Technische Universität Berlin, Fakultät Elektrotechnik und Informatik, Germany florian.pudlitz@tu-berlin.de
[4] GPP Communication GmbH & Co. KG, Munich, Germany p.gersing@gppag.de
[5] foqee GmbH, Munich, Germany philipps@foqee.de

doi:10.18420/SE2020_28

the new MBSE processes well documented so you better understand what tool you will need. **BP10:** All engineers should get, at least, basic training in MBSE. **BP11:** Using examples that are familiar to the domain of the organization eases the understanding. Model some existing artifacts for using as examples. **BP12:** Many strategies can be used to build knowledge of an organization, the context should be taken into consideration. **BP13:**There should be a planned form of later evaluation to fill eventual gaps. **BP14:** Make the advantages of MBSE clear. **BP15:** Have technically prepared people to support your engineers (i.e., not sales personnel). **BP16:** Bring everyone to adoption (i.e., avoid creating castes). **BP17:** If you have good engineers let them do the work for you, it is cheaper and they will engage more (i.e., empowering). **BP18:** Management should unify all employees towards adoption.

Most important best practices: We used a questionnaire to discover which best practices (BP) are considered the most important. The BPs are discussed in the following under this light: **BP14** Engineers are less likely to withstand the hurdles of adoption if they cannot perceive its benefits [Vo17]. **BP05** Through experimenting, managers can understand which tools, languages, and styles are best fitted for the organization and its respective domains and current processes. **BP02** By working into something that will be used in production setting, the employees have to learn and employ MBSE. If it is something that is just for learning, there are no consequences if the project is incomplete or not well done thus making the learning also incomplete. It also gives room for procrastination (i.e., learning later when it is necessary). **BP03** Learning-curve costs as well as the delivery dates should be planned accordingly. Time-critical projects should be avoided. Engineers will drop the MBSE techniques in favor of already established development methods in order to achieve celerity gains and meet deadlines. **BP10** Although not all engineers require deep modeling skills, everyone should be able to, at least, read and understand the models.

Conclusion: The complexity of MBSE and its pervasiveness creates challenges that could jeopardize its implementation in an organization. In the long run, benefits outweigh the costs and hurdles, however, it is necessary to identify success factors and share best practices enabling efficient and effective MBSE adoption in industry. As future work, we could investigate the best practices that received many disagreement votes to understand whether context plays a role in such phenomenon.

Literaturverzeichnis

[Bö14] Böhm, Wolfgang; Junker, Maximilian; Vogelsang, Andreas; Teufl, Sabine; Pinger, Ralf; Rahn, Karsten: A Formal Systems Engineering Approach in Practice: An Experience Report. In: Proceedings of the 1st International Workshop on Software Engineering Research and Industrial Practices. S. 34–41, 2014.

[Vo17] Vogelsang, Andreas; Amorim, Tiago; Pudlitz, Florian; Gersing, Peter; Philipps, Jan: Should I Stay or Should I Go? On Forces that Drive and Prevent MBSE Adoption in the Embedded Systems Industry. In: PROFES 2017, Innsbruck, Austria, November 29–December 1, 2017, Proceedings. S. 182–198, 2017.

Managing Inter-Model Inconsistencies in Model-based Systems Engineering: Application in Automated Production Systems Engineering

Stefan Feldmann[1], Konstantin Kernschmidt[1], Manuel Wimmer[2], Birgit Vogel-Heuser[1]

Abstract: This work summarizes our paper [Fe19] originally published in the Journal of Systems and Software in 2019 about a model-based inconsistency management approach.

Keywords: Automated production systems, Model-based systems engineering, Inconsistency management

The use of model-based approaches [BCW17] is a crucial and competitive factor for successful engineering processes in the automated production systems domain, and hence an emerging practice in industry. To represent the specific views of the system under design, e.g., requirements engineering, system specifications, software design or system analyses, as well as to deal with the multitude of involved disciplines, e.g., mechanical, electrical, and software engineering, a variety of different models is created within a project. In addition, to address their specific viewpoints, stakeholders make use of various, heterogeneous modelling languages and tools [Br10].

Although all models represent different aspects of the same system under investigation, dependencies between these models are inevitable [Vo15]. As a consequence, inconsistencies between models are likely to occur and have to be carefully considered to guarantee a high quality of the final system design. In particular, the necessity to continuously diagnose and handle inconsistencies between models arises.

To tackle this challenge, we propose a comprehensive approach to specify, diagnose, and handle inconsistencies in model-based systems engineering. To explicitly capture the dependencies and consistency rules that must hold between the different engineering models, a dedicated graphical modelling language is proposed. By means of this language, stakeholders can specify, diagnose, and handle inconsistencies in the accompanying inconsistency management framework.

[1] Chair of Automation and Information Systems, Technical University of Munich, Boltzmannstr. 15, 85748 Garching near Munich, Germany {feldmann,kernschmidt,vogel-heuser}@ais.mw.tum.de
[2] Institute of Business Informatics - Software Engineering, JKU Linz, Altenbergerstr. 69, 4040 Linz, Austria manuel.wimmer@jku.at

doi:10.18420/SE2020_29

With the help of the proposed approach we enable stakeholders to:

- explicitly specify and elaborate n-to-m links between the different, but overlapping engineering models,
- graphically specify consistency rules between models including different mathematical mapping properties and function types,
- continuously diagnose potentially occurring inconsistencies, and
- systematically handle the life-cycle of these inconsistencies through either ignoring, tolerating or resolving them.

The approach is implemented based on the Eclipse Modeling Framework (EMF). The resulting prototypical tool support allows for automatically generating executable consistency rules from the graphically expressed consistency patterns and for automatically creating inconsistency management repositories. The prototypical tool support has been evaluated based on a case study about a production system demonstrator project [Vo14] as well as a user experiment. Our findings indicate that the approach is expressive enough to capture typical dependencies and consistency rules in the automated production system domain and that it requires less effort compared to manually developing and maintaining inter-model inconsistency management solutions. The findings gained from the case study based evaluation are underpinned through the user experiment.

Literatur

[BCW17] Brambilla, M.; Cabot, J.; Wimmer, M.: Model-Driven Software Engineering in Practice. Morgan & Claypool, 2017.

[Br10] Broy, M.; Feilkas, M.; Herrmannsdoerfer, M.; Merenda, S.; Ratiu, D.: Seamless Model-Based Development: From Isolated Tools to Integrated Model Engineering Environments. Proceedings of the IEEE 98/4, S. 526–545, 2010.

[Fe19] Feldmann, S.; Kernschmidt, K.; Wimmer, M.; Vogel-Heuser, B.: Managing inter-model inconsistencies in model-based systems engineering: Application in automated production systems engineering. Journal of Systems and Software 153/, S. 105–134, 2019.

[Vo14] Vogel-Heuser, B.; Legat, C.; Folmer, J.; Feldmann, S.: Researching Evolution in Industrial Plant Automation: Scenarios and Documentation of the Pick and Place Unit, Techn. Ber. TUM-AIS-TR-01-14-02, TU Munich, 2014, URL: https://mediatum.ub.tum.de/node?id=1208973.

[Vo15] Vogel-Heuser, B.; Fay, A.; Schaefer, I.; Tichy, M.: Evolution of Software in Automated Production Systems: Challenges and Research Directions. Journal of Systems and Software 110/, S. 54–84, 2015.

Searching for Optimal Models: Comparing Two Encoding Approaches (Summary)

Stefan John [1], Alexandru Burdusel[2], Robert Bill[3], Daniel Strüber[4], Gabriele Taentzer [5], Steffen Zschaler[6], Manuel Wimmer[7]

Abstract: This work summarizes our paper originally published in The Journal of Object Technology in the course of the International Conference on Model Transformations 2019 [Jo19].

Keywords: Model-driven Engineering; Search-based Software Engineering; Optimization; Encoding; Comparative evaluation

Many software engineering problems give rise to a tremendous space of possible solutions that differ in various qualities, such as their performance, resource efficiency, and understandability. To find optimal solutions, *search-based software engineering (SBSE)* seeks to formulate these problems as optimization problems and applies metaheuristic search techniques, to efficiently explore the solution space. Model-driven engineering (MDE) is a paradigm that aims to raise the level of abstraction in a broad range of application domains by the use of models, which are continuously refined and transformed.

Research combining SBSE and MDE under the umbrella term *search-based model-driven engineering* (SBMDE) has become increasingly popular. One particular line of research in SBMDE, which we call *model-driven optimization* (MDO), aims to reduce the level of expertise required by users of SBSE techniques[8]. In MDO, models are used to specify optimization problems and transformation rules are used to explore the search space. Thus, rather than becoming involved in the intricacies of the used optimization technology, users interact with a domain-specific formulation of their problem. They can rely on the familiar modeling and model transformation tools to inspect solutions and specify change operations.

Recently, a variety of MDO frameworks has emerged with their key distinction being the way in which solutions are encoded [ZM16]: The *model-based encoding* approach represents solutions as models. In the *rule-based encoding* approach, a solution is a sequence of rule

[1] Philipps-Universität Marburg, Marburg, Germany, stefan.john@uni-marburg.de
[2] King's College London, London, United Kingdom, alexandru.burdusel@kcl.ac.uk
[3] TU Wien | Austrian Center for Digital Production, Wien, Austria, bill@big.tuwien.ac.at
[4] Chalmers University | University of Gothenburg, Gothenburg, Sweden, danstru@chalmers.se
[5] Philipps-Universität Marburg, Marburg, Germany, taentzer@informatik.uni-marburg.de
[6] King's College London, London, United Kingdom, szschaler@acm.org
[7] CDL-MINT, Johannes Kepler University, Linz, Austria, manuel.wimmer@jku.at
[8] As opposed to applying SBSE techniques to solve MDE problems.

doi:10.18420/SE2020_30

calls in the context of a given input model. While both encodings have been applied to different use cases, no study has yet compared them systematically. Hence, we evaluate both approaches on a common set of optimization problems, investigating their impact on optimization performance. To that end, we rely on two state-of-the-art MDO frameworks (MOMoT [Bi17] and MDEOptimiser [BZS18]) that differ in the encoding approach used, but otherwise share the same technological basis: EMF[9]. (as the modeling platform), Henshin [Ar10; St17] (as model transformation language), and the MOEA evolutionary search framework[10]. Additionally, we discuss the differences, strengths, and weaknesses of both encoding approaches laying the foundation for a knowledgeable choice of when to use which encoding. Consequently, the main contributions of the paper are as follows:

1. A *qualitative comparison* between the model-based and the rule-based encoding in MDO frameworks, based on a systematic study of their features.
2. A *quantitative comparison* of both encodings with their implementations in MOMoT and MDEOptimiser, based on their performance (regarding solution quality and execution time) in a set of three diverse use cases.
3. *Insights into the applicability* of both encoding approaches; their strengths and weaknesses. We study whether the differences can be attributed to the different encoding approaches.

References

[Ar10] Arendt, T.; Biermann, E.; Jurack, S.; Krause, C.; Taentzer, G.: Henshin: Advanced concepts and tools for in-place EMF model transformations. In: Int. Conference on Model Driven Engineering Languages and Systems. Pp. 121–135, 2010.

[Bi17] Bill, R.; Fleck, M.; Troya, J.; Mayerhofer, T.; Wimmer, M.: A local and global tour on MOMoT. Software & Systems Modeling/, pp. 1–30, 2017.

[BZS18] Burdusel, A.; Zschaler, S.; Strüber, D.: MDEOptimiser: A search based model engineering tool. In: Int. Conference on Model Driven Engineering Languages and Systems. Pp. 12–16, 2018.

[Jo19] John, S.; Burdusel, A.; Bill, R.; Strüber, D.; Taentzer, G.; Zschaler, S.; Wimmer, M.: Searching for optimal models: Comparing two encoding approaches. Journal of Object Technology 18/3, 2019.

[St17] Strüber, D.; Born, K.; Gill, K. D.; Groner, R.; Kehrer, T.; Ohrndorf, M.; Tichy, M.: Henshin: A usability-focused framework for EMF model transformation development. In: Int. Conference on Graph Transformation. Pp. 196–208, 2017.

[9] https://www.eclipse.org/modeling/emf/ (last visited: December 2019)
[10] http://moeaframework.org (last visited: December 2019)

[ZM16] Zschaler, S.; Mandow, L.: Towards model-based optimisation: Using domain knowledge explicitly. In: Workshop on Model-Driven Engineering, Logic and Optimization. Pp. 317–329, 2016.

Testing 1

Cooperative Test-Case Generation with Verifiers

Dirk Beyer[1], Marie-Christine Jakobs[2]

Abstract: Software testing is widely applied in software quality assurance. Often, test suites fulfilling a certain coverage measure must be constructed. Manually constructing them is laborious. However, numerous automatic test-generation approaches exist. Due to various strengths and weaknesses of individual approaches, hybrid approaches, which combine different approaches, construct test suites that achieve higher coverage values than test suites generated by individual approaches.

We propose the hybrid test-generation approach CoVeriTest. CoVeriTest is flexible, cooperative, and based on verification technology. It iteratively executes a sequence of verifiers that may exchange analysis information between each other and output a test case whenever they reach a test goal. The verifiers, their individual time limits, and which analysis information is exchanged between them is configurable. We experimented with different CoVeriTest configurations. The best configuration participated in the 1st International Competition on Software Testing (Test-Comp'19) and won the third place. This proves the value of our CoVeriTest approach.

Keywords: Test-case generation; Software testing; Test coverage; Conditional model checking; Cooperative verification; Model checking

1 Overview

Often, testing is an integral part of the software-development process, in which it is used to evaluate the software quality. Thereby, coverage criteria are applied to assess the adequacy of a generated test suite. Manually constructing adequate test suites is typically too laborious. Hence, automatic approaches are used. However, different strengths and weaknesses of existing approaches make it necessary to combine different approaches to achieve good coverage values. Inspired by abstraction-driven concolic testing [DGH16] and recent advances of software verifiers, we therefore propose the flexible, cooperative hybrid test-generation approach CoVeriTest [BJ19].

CoVeriTest is based on verifiers, which construct a test case whenever they reach a test goal (e. g., a branch). Figure 1 shows the workflow of CoVeriTest. CoVeriTest iteratively executes a sequence of n verifiers. In each iteration, verifier i is run for t_i time units and adds its generated test cases to the global test suite. Before a verifier is run, the `init` procedure sets up its context, e. g., the open test goals, the program, and the information from other verifiers. At the end of a verification run, the verifier provides all analysis information in form of the

[1] LMU Munich, Institute of Informatics, Oettingenstraße 67, 80538 Munich, Germany
[2] TU Darmstadt, Department of Computer Science, Hochschulstraße 10, 64289 Darmstadt, Germany
This is a summary of an article that is published in Proc. FASE'19 [BJ19].

doi:10.18420/SE2020_31

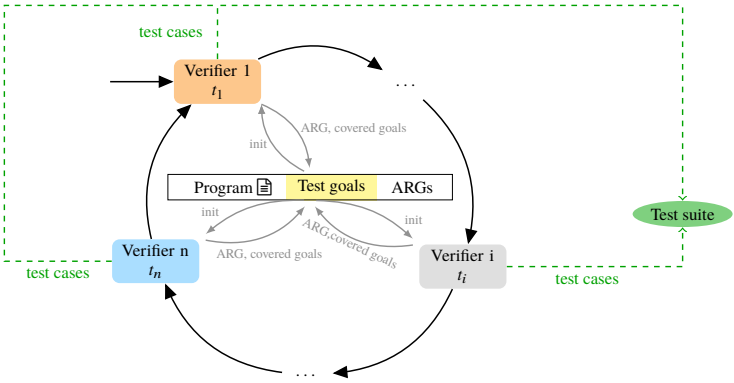


Fig. 1: Workflow of the CoVeriTest approach

reached state space (ARG). Additionally, it reports the goals it covered and that are no longer considered. The number n of verifiers as well as the verifiers i and their time limits t_i can be configured. Also, the cooperation, i.e., the information exchange between verification runs, can be configured. Next to no cooperation, CoVeriTest supports (1) cooperation between different runs of the same verifier, e. g., reuse the abstraction level or continue the exploration of the state space (ARG), (2) cooperation between different verifiers, e. g., ignore already explored paths, and (3) combinations of the previous cooperation types.

For our experiments, CoVeriTest combines value and predicate analysis[3], uses either time unit pair (10 s, 10 s), (50 s, 50 s), (100 s, 100 s), (250 s, 250 s), (80 s, 20 s), or (20 s, 80 s) and one of the different cooperation settings mentioned above. Our evaluation with the programs from the software verification benchmark[4] revealed that the instance using value and predicate analysis with time units (20 s, 80 s) and that lets the analyses continue their exploration works best. Also, this CoVeriTest configuration often achieves higher coverage values than (1) either value or predicate analysis alone, (2) their parallel combination, and (3) their sequential combination (20 % value analysis followed by 80 % predicate analysis). Furthermore, CoVeriTest's 3rd place in Test-Comp'19 [Be19] confirms its value.

Bibliography

[Be19] Beyer, D.: International Competition on Software Testing (Test-Comp). In: Proc. TACAS. LNCS 11429. Springer, pp. 167–175, 2019.

[BJ19] Beyer, D.; Jakobs, M.-C.: CoVeriTest: Cooperative Verifier-Based Testing. In: Proc. FASE. LNCS 11424. Springer, pp. 389–408, 2019.

[DGH16] Daca, P.; Gupta, A.; Henzinger, T. A.: Abstraction-Driven Concolic Testing. In: Proc. VMCAI. LNCS 9583. Springer, pp. 328–347, 2016.

[3] Meanwhile we also experimented with combinations of bounded model checking and symbolic execution.
[4] https://github.com/sosy-lab/sv-benchmarks

Investigating Next Steps in Static API-Misuse Detection

Sven Amann[1] Hoan Anh Nguyen[2] Sarah Nadi[3] Tien N. Nguyen[4] Mira Mezini[5]

Keywords: API Misuse, Constraint Mining, Machine Learning, Bug Detection

Incorrect usages of an Application Programming Interface (API), or *API misuses*, are violations of (implicit) *usage constraints* of the API. An example of a usage constraint is having to check that `hasNext()` returns `true` before calling `next()` on an `Iterator`, in order to avoid a `NoSuchElementException` at runtime. API misuse is a prevalent cause of software bugs, crashes, and vulnerabilities [Na16, Am16].

To mitigate API misuse, researchers have proposed several *API-misuse detectors* [WZ11, MM13, Ng15]. These detectors analyze *API usages*, i.e., code snippets that use a given API. The detectors commonly mine *usage patterns*, i.e., equivalent API usages that occur frequently, and then report deviations from these patterns as potential misuses. Unfortunately, the reported precision of such detectors is typically low and a recent study [Am18] showed that their recall is also very low. Thus, we need better detectors to address the still-prevalent problem of API misuse [Le16, Ac16].

Previous work identified individual as well as common strengths and weaknesses of existing detectors [Am18] in an empirical study using the open-source benchmark MuBench [MU17]. In this paper, we investigate whether addressing the reported weaknesses indeed leads to better performance in practice. Therefore, we design a new misuse detector, MuDetect. MuDetect encodes API usages as API-Usage Graphs (AUGs), a comprehensive usage representation that captures different types of API misuses. MuDetect employs a greedy, frequent-subgraph-mining algorithm to mine patterns and a specialized graph-matching strategy to identify pattern violations. Both components consider code semantics to improve the overall detection capabilities. On top, MuDetect uses an empirically optimized ranking strategy to effectively rank true positives. While previous detectors mostly target a per-project setting [Am18], MuDetect also works in a cross-project setting, where it mines thousands of usage examples from third-party projects.

We assess the precision and recall of MuDetect and show that it outperforms the four state-of-the-art detectors evaluated in prior work [Am18]. In our evaluation, we extended MuBench

[1] CQSE GmbH, Centa-Hafenbrädl-Straße 59, 81249 München, Germany, amann@cqse.eu
[2] Amazon Research, hoan@iastate.edu
[3] University of Alberta, 4-41 Athabasca Hall, Canada, nadi@ualberta.ca
[4] The University of Texas at Dallas, 800 W. Campbell Road, ECSS 4.229 Richardson, TX 75080-3021, USA, tien.n.nguyen@utdallas.edu
[5] Technische Universität Darmstadt, Hochschulstraße 10, 64289 Darmstadt, mezini@cs.tu-darmstadt.de

doi:10.18420/SE2020_32

by 107 real-world misuses identified in a recent study on run-time verification [Le16]—more than doubling its size—to ensure that our design decisions generalize. We show that, in a setting with perfect training data, MuDetect achieves a recall of 72.5%, which is 20.3% higher than the next best detector and over 50% higher than the other detectors. In the typical per-project setting, MuDetect achieves recall of 20.9%, which is 10.2% better than the second-best detector, and precision of 21.9%, which is 13.1% better than the second-best detector. In a cross-project setting, MuDetect's recall and precision again improve significantly to 42.2% and 33.0%, respectively. Throughout the experiments, MuDetect identified 27 previously unknown misuses, which we reported in eight pull requests (PRs). To date, three of the PRs got accepted, demonstrating that MuDetect identifies actual issues in current software projects.

We publish our MuBench extension, MuDetect's implementation, and all experiment data, tooling, and results [Ar19].

Bibliography

[Ac16] Acar, Yasemin; Backes, Michael; Fahl, Sascha; Kim, Doowon; Mazurek, Michelle L.; Stransky, Christian: You Get Where You're Looking For. The Impact of Information Sources on Code Security. In: Proceedings of the 37th IEEE Symposium on Security and Privacy. IEEE Computer Society Press, 2016.

[Am16] Amann, Sven; Nadi, Sarah; Nguyen, Hoan A.; Nguyen, Tien N.; Mezini, Mira: MUBench: A Benchmark for API-Misuse Detectors. In: Proceedings of the 13th Working Conference on Mining Software Repositories. MSR '16. ACM Press, 2016.

[Am18] Amann, Sven; Nguyen, Hoan A.; Nadi, Sarah; Nguyen, Tien N.; Mezini, Mira: A Systematic Evaluation of Static API-Misuse Detectors. IEEE Transactions on Software Engineering, 2018.

[Ar19] Artifacts: , http://www.st.informatik.tu-darmstadt.de/artifacts/mudetect/, 2019.

[Le16] Legunsen, Owolabi; Hassan, Wajih Ul; Xu, Xinyue; Roşu, Grigore; Marinov, Darko: How Good Are the Specs? A Study of the Bug-finding Effectiveness of Existing Java API Specifications. In: Proceedings of the 31st IEEE/ACM International Conference on Automated Software Engineering. ASE '16. ACM Press, pp. 602–613, 2016.

[MM13] Monperrus, Martin; Mezini, Mira: Detecting Missing Method Calls as Violations of the Majority Rule. ACM Transactions on Software Engineering and Methodology, 22(1):1–25, 2013.

[MU17] MUBench: , https://github.com/stg-tud/MUBench/, 2017.

[Na16] Nadi, Sarah; Krüger, Stefan; Mezini, Mira; Bodden, Eric: "Jumping Through Hoops": Why do Developers Struggle with Cryptography APIs? In: Proceedings of the 38th International Conference on Software Engineering. ICSE'16. ACM Press, 2016.

[Ng15] Nguyen, T. T.; Pham, H. V.; Vu, P. M.; Nguyen, T. T.: Recommending API Usages for Mobile Apps with Hidden Markov Model. In: Proceedings of the 30th ACM/IEEE International Conference on Automated Software Engineering. ASE '15. IEEE Computer Society Press, pp. 795–800, 2015.

[WZ11] Wasylkowski, Andrzej; Zeller, Andreas: Mining Temporal Specifications from Object Usage. Automated Software Engineering, 18(3-4):263–292, 2011.

GuideGen: An Approach for Keeping Requirements and Acceptance Tests Aligned

Sofija Hotomski[1], Martin Glinz[2]

Abstract: This contribution was originally published as: Sofija Hotomski, Martin Glinz (2019). GuideGen: An Approach for Keeping Requirements and Acceptance Tests Aligned via Automatically Generated Guidance. *Information and Software Technology* 110:17–38.

Keywords: requirements; acceptance tests; document alignment; software engineering tools; software evolution

1 Context and Motivation

When software-based systems evolve, their requirements change. The changes in requirements affect the associated acceptance tests, which should be adapted accordingly. In practice, however, requirements and their acceptance tests are not always kept up-to-date nor aligned [HBCG16]. Such inconsistencies may introduce software quality problems, unintended costs and project delays [Bj14].

In order to keep evolving requirements and their acceptance tests aligned, we have developed an approach called GuideGen. GuideGen automatically generates guidance in natural language about how to adapt the impacted acceptance tests when their requirements change [HG19].

2 Method

We have implemented GuideGen as a prototype tool [HG18a] and evaluated it in two studies. In the first one, we assessed the correctness, completeness, understandability and relevance of the generated guidance using three data sets from industry [HBCG18]. In the second study, we investigated the applicability and usefulness of the approach and the tool with 23 practitioners from ten companies [HG18b]. When a requirement having more than one associated acceptance test is changed, GuideGen currently generates guidance for all of them

[1] ASMIQ AG, Zurich, Switzerland, sofija.hotomaski@gmail.com (work was done while the author was with the University of Zurich)
[2] Department of Informatics, University of Zurich, Switzerland, glinz@ifi.uzh.ch

doi:10.18420/SE2020_33

together. As a first step towards overcoming this limitation, we experimentally assessed how well existing methods for change impact analysis can identify the tests actually impacted by the changes in a requirement [HG19].

3 Results

In the first study, we found that GuideGen produced correct guidance in about 67 to 89 percent of all changes. Our approach performed better for agile requirements than for traditional ones. The results of the second study show that GuideGen is perceived to be useful, but that the practitioners would prefer a GuideGen plug-in for commercial tools instead of a standalone tool. Further, in our experiment about identifying affected tests with existing change impact analysis methods, we could correctly identify the affected acceptance tests for 63% to 91% of the changes in the requirements.

4 Conclusion

Our approach facilitates the alignment of acceptance tests with the actual requirements and can improve the communication between requirements engineers and testers.

References

[Bj14] Bjarnason, Elizabeth; Runeson, Per; Borg, Markus; Unterkalmsteiner, Michael; Engström, Emelie; Regnell, Björn; Sabaliauskaite, Giedre; Loconsole, Annabella; Gorschek, Tony; Feldt, Robert: Challenges and practices in aligning requirements with verification and validation: a case study of six companies. Empirical Software Engineering, 19(6):1809–1855, 2014.

[HBCG16] Hotomski, Sofija; Ben Charrada, Eya; Glinz, Martin: An Exploratory Study on Handling Requirements and Acceptance Test Documentation in Industry. In: 24th IEEE International Requirements Engineering Conference (RE'16). IEEE, pp. 116–129, 2016.

[HBCG18] Hotomski, Sofija; Ben Charrada, Eya; Glinz, Martin: Keeping Evolving Requirements and Acceptance Tests aligned with Automatically Generated Guidance. In: 24th International Working Conference on Requirements Engineering: Foundation for Software Quality (REFSQ 2018). Springer, pp. 247–264, 2018.

[HG18a] Hotomski, Sofija; Glinz, Martin: GuideGen: a tool for keeping requirements and acceptance tests aligned. In: Proceedings of the 40th International Conference on Software Engineering (ICSE 2018), Companion volume. ACM, pp. 49–52, 2018.

[HG18b] Hotomski, Sofija; Glinz, Martin: A Qualitative Study on Using GuideGen to Keep Requirements and Acceptance Tests Aligned. In: 26th IEEE International Requirements Engineering Conference (RE'18). 2018.

[HG19] Hotomski, Sofija; Glinz, Martin: GuideGen: An Approach for Keeping Requirements and Acceptance Tests Aligned via Automatically Generated Guidance. Information and Software Technology, 110:17–38, 2019.

Cloud, Edge und Microservices

Microservices and Containers – Architectural Patterns for Cloud and Edge

Claus Pahl[1], Pooyan Jamshidi[2], Olaf Zimmermann[3]

Abstract: Software architecture research needs to address the specific needs and constraints of specific deployment contexts. We propose an architectural style for cloud-deployed software referring to principles and patterns. Patterns map abstract principles to development and deployment platform solution templates Together, principles and patterns link common software architecture concepts, such as services, adaptivity or models at runtime, to deployment specifics such as virtualisation and controller-based feedback loops. The results of this broader framework shall be discussed in the context of recent trends such as microservices.

Keywords: Software Architecture, Architectural Style, Patterns, Cloud, Edge, Microservices, Container.

1 Architectural Principles for Cloud Software

Software architecture should support the specific needs and constraints of specific deployment contexts. An architectural style for cloud-deployed software can provide this support based on the following principles [PJZ18]: (1) service-orientation: services provided using layering, modularity, and loose coupling. (2) virtualization: services for shared resources and portable containers. (3) uncertainty: distribution, heterogeneity, and multi-user involvement cause uncertainty. (4) adaptivity: operation and management support allow for dynamic adaptation.

Patterns map principles to development and deployment platform solution templates: (1) microservices: flexible composition with independent, self-managed containers and cloud-native services. (2) models at runtime: allows aspects of uncertainty to be addressed dynamically. (3) controller-based feedback loop: allows controllers to adapt to and manage change. Together, principles and patterns link common software architecture concepts, such as services, adaptivity or models at runtime, to deployment specifics such as virtualisation and controller-based feedback loops.

The results of this broader framework shall be discussed in the context of open challenges such as self-adaptive microservices (see [Ja18, Pa17, Me19]), but also deployments not only in cloud, but also related edge and IoT settings [Le19].

[1] Software and Systems Engineering, Free University of Bozen-Bolzano, Bozen, Italy
[2] Computer Science and Engineering, University of South Carolina, Columbia, USA
[3] Institute for Software, Hochschule für Technik Rapperswil, Rapperswil, Switzerland

doi:10.18420/SE2020_34

2 Self-Adaptive Microservices

A self-adaptive system dynamically adapts its behaviour to preserve or enhance quality attributes in uncertain operating conditions. Here, the development of microservice applications as such self-adaptive systems is still a challenge. In practice, e.g., the Kubernetes container orchestration platform facilitates to deploy and manage microservice applications, natively only supports autoscaling (automatically change the number of instances of a service) and self-healing (automatically restart failed service instances. Microservice quality attributes can be improved by using planning techniques (determine the best adaptation strategy for each microservice), machine learning (learn new adaptation strategies from past adaptation results), reasoning under uncertainty (cope with noisy monitoring data), and multi-objective optimization (cater for multiple, possibly conflicting requirements). However, an important observation is that independent and frequent deployments, the need for a high degree of automation in a DevOps context, and complex run-time architectures make microservices ideal candidates to be deployed as self-adaptive systems.

3 Containerised Edge and IoT

Containers and microservices are lightweight approaches for architecting and deploying software that are particularly needed for an edge computing model, which aims to provide low-cost local clusters at the outer edge of the cloud, possibly composed of IoT devices themselves. We can demonstrate that fully containerised microservice architectures for data streaming platforms can be deployed even on small, e.g., single-board device clusters. They can serve IoT use cases for which the data volume is not too high. Benefits include flexible container deployment and (at least) sufficient performance and scalability despite device limitations. Challenges still exist in the dynamic load-driven distribution management of the containers in clusters and the adaptivity problem already discussed. Autoscaling is feasible, which we implemented for a serverless architecture using a fuzzy controller.

Literaturverzeichnis

[Ja18] Jamshidi, P.; Pahl, C.; Mendonca, N. C.; Lewis, J.; Tilkov, S.: Microservices: The Journey So Far and Challenges Ahead. IEEE Software, 35(3):24–35, 2018.

[Le19] von Leon, David; Miori, Lorenzo; Sanin, Julian; El Ioini, Nabil; Helmer, Sven; Pahl, Claus: A Lightweight Container Middleware for Edge Cloud Architectures. In: Fog and Edge Computing. John Wiley and Sons, Ltd, Kapitel 7, S. 145–170, 2019.

[Me19] Mendonca, N. C.; Jamshidi, P.; Garlan, D.; Pahl, C.: Developing Self-Adaptive Microservice Systems: Challenges and Directions. IEEE Software, 2019.

[Pa17] Pautasso, C.; Zimmermann, O.; Amundsen, M.; Lewis, J.; Josuttis, N.: Microservices in Practice, Part 1: Reality Check and Service Design. IEEE Software, 34(1):91–98, 2017.

[PJZ18] Pahl, Claus; Jamshidi, Pooyan; Zimmermann, Olaf: Architectural Principles for Cloud Software. ACM Trans. Internet Technol., 18(2):17:1–17:23, Februar 2018.

Optimized Application Deployment using Fog and Cloud Computing Environments

Zoltán Ádám Mann[1] Andreas Metzger[2] Johannes Prade[2] Robert Seidl[3]

Abstract: Fog computing uses geographically distributed fog nodes that can supply nearby end devices with low-latency access to cloud-like compute resources. If the load of a fog node exceeds its capacity, some non-latency-critical application components may be offloaded to the cloud. Using commercial cloud offerings for such offloading incurs financial costs. Optimally deciding which application components to keep in the fog node and which ones to offload to the cloud is a difficult combinatorial problem. We introduce an optimization algorithm that (i) guarantees that the deployment always satisfies capacity constraints and affinity requirements, (ii) achieves near-optimal cloud usage costs, and (iii) is fast enough to be run online. The practical use of the algorithm is illustrated by applying it to optimizing the applications in a mobile factory.

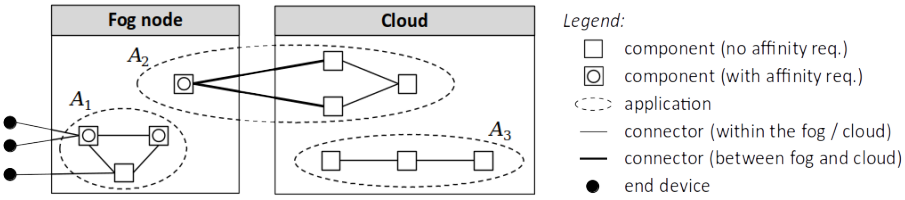
Keywords: Fog Computing, cost-optimized software deployment, resource optimization.

1 Introduction

Fog and edge computing provide compute resources that are frequently used in industrial environments. An interesting example is the concept of small mobile factories, or Factory in a Box, FiaB [19]. The limited compute resources in the FiaB, which represent a fog node, host latency-critical application components that are relevant to production processes and must be placed in the FiaB. Also software components processing sensitive data have to be placed in the FiaB for data protection reasons [MMP19]. For software components that are not critical in terms of latency or data protection (e.g., related to maintenance), there is no such affinity constraint. Such components can be placed either also in the FiaB, or remotely in a cloud, as shown in the below figure. The placement of these components is a complex optimization problem [Ma19a].

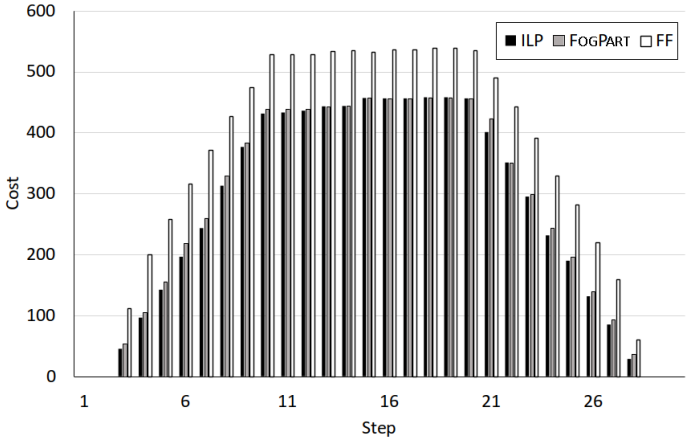
[1] Universität Duisburg-Essen, Essen, Germany
[2] Nokia, Munich, Germany
[3] Nokia Bell Labs, Munich, Germany

doi:10.18420/SE2020_35



2 Approach and Result

To determine a cost-effective deployment for distributed applications, taking into account the components' affinity requirements, the fog node's capacity, and the financial costs of using the cloud, we devised an algorithm called FogPart [Ma19b]. To apply the algorithm to the FiaB case study, we defined the software functions needed for the FiaB, the communication needs of these functions and their resource requirements. With this input, we ran the algorithm to determine an optimized placement for the FiaB applications. We also compared FogPart to an exact algorithm based on integer linear programming (ILP) and a heuristic using first-fit (FF) in a set of controlled experiments. As the following figure shows, FogPart leads to costs that are very close to the ILP results, and significantly lower than those of the FF heuristic.



The experiments also showed that FogPart is very fast. For example, it took less than 300ms on a commodity computer to optimize the deployment of 450 components. Thus, FogPart can be used to re-optimize the placement of components at run time.

References

[19] Figure 4® Additive Manufacturing at Nokia in an Industry 4.0 Ecosystem, https://youtu.be/4kSE8WaeVTc, 2019.

[Ma19a] Mann, Z.: Optimization Problems in Fog and Edge Computing. In. Pp. 103–121, Jan. 2019.

[Ma19b] Mann, Z. Á.; Metzger, A.; Prade, J.; Seidl, R.: Optimized Application Deployment in the Fog. In: 17th International Conference on Service-Oriented Computing (ICSOC). Springer International Publishing, pp. 283–298, 2019.

[MMP19] Mann, Z. Á.; Metzger, A.; Pohl, K.: Situativer Datenschutz im Fog-Computing. Informatik Spektrum 42/4, pp. 236–243, Aug. 2019.

Security

Reproducing Taint-Analysis Results with ReproDroid

Felix Pauck[1], Eric Bodden[2], Heike Wehrheim[3]

Abstract:

More and more Android taint-analysis tools appear each year. Any paper proposing such a tool typically comes with an in-depth evaluation of its supported features, accuracy and ability to be applied on real-world apps. Although the authors spent a lot of effort to come up with these evaluations, comparability is often hindered since the description of their experimental targets is usually limited.

To conduct a comparable, automatic and unbiased evaluation of different analysis tools, we propose the framework ReproDroid. The framework enables us to precisely declare our evaluation targets, in consequence we refine three well-known benchmarks: DroidBench, ICC-Bench and DIALDroid-Bench. Furthermore, we instantiate this framework for six prominent taint-analysis tools, namely Amandroid, DIALDroid, DidFail, DroidSafe, FlowDroid and IccTA. Finally, we use these instances to automatically check whether different promises commonly made in the associated proposing papers are kept.

Keywords: Android Taint Analysis; Tools; Benchmarks; Empirical Studies; Reproducibility

1 The ReproDroid Study

Fig. 1: Overview of ReproDroid

Figure 1 depicts the conceptual idea behind our Android benchmark reproduction framework ReproDroid, which helps to (i) precisely specify the ground-truth of benchmarks and allows to (ii) automatically execute and (iii) evaluate benchmarks. First, the app-set representing a certain benchmark must be loaded into ReproDroid. In a semi-automatic manner the ground-truth can then be specified for the associated apps. Effectively a list of precisely defined sources, sinks and expected flows between them is generated this way. A *refined* benchmark, for which such a list exists, can be stored and reused anytime without having to redo this first refinement step. Once a refined benchmark becomes available in ReproDroid an arbitrary analysis tool can automatically be executed for each benchmark case. In order to do so, only

[1] Paderborn University, Warburger Str. 100, 33098 Paderborn, Germany fpauck@mail.uni-paderborn.de
[2] Paderborn University, Warburger Str. 100, 33098 Paderborn, Germany eric.bodden@uni-paderborn.de
[3] Paderborn University, Warburger Str. 100, 33098 Paderborn, Germany wehrheim@uni-paderborn.de

doi:10.18420/SE2020_36

ReproDroid's configuration needs to be adapted to suit the designated analysis tool. In the end, results produced this way are collected and automatically compared against the beforehand specified ground-truth.

In our evaluation we took a look at three different categories of promises given in the proposing papers considering Amandroid [We14], DIALDroid [Bo17], DidFail [Kl14], DroidSafe [Go15], FlowDroid [Ar14] and IccTA [Li15]. The first type of evaluated promises (*feature-promises*) considers the features and sensitivities which are claimed to be supported by these six tools. Second, *accuracy-promises* are checked by attempting to reproduce the F-measure values reported (+/- 0.2) for certain benchmarks or subsets. The authors of the papers associated with all six tools promise that their tool is real world ready. Thus, lastly we checked these *real-world-promises*. Table 2 shows the number of promises given and how many could not be confirmed per category – a verbose version of these results is available in [PBW18]. As we can see, only a minority of promises cannot be confirmed. Nonetheless, the accuracy-promises are not strictly kept and in particular the real-world-promises are not fully kept by any tool.

Promises	Given / Unconfirmed	
Feature	64	5
Accuracy	12	6
Real-World	6	6

Fig. 2: Evaluation results

References

[Ar14] Arzt, Steven; Rasthofer, Siegfried; Fritz, Christian; Bodden, Eric; Bartel, Alexandre; Klein, Jacques; Traon, Yves Le; Octeau, Damien; McDaniel, Patrick D.: FlowDroid: precise context, flow, field, object-sensitive and lifecycle-aware taint analysis for Android apps. In: Proceedings of PLDI, 2014. ACM, pp. 259–269, 2014.

[Bo17] Bosu, Amiangshu; Liu, Fang; Yao, Danfeng (Daphne); Wang, Gang: Collusive Data Leak and More: Large-scale Threat Analysis of Inter-app Communications. In: Proceedings of AsiaCCS, 2017. ACM, pp. 71–85, 2017.

[Go15] Gordon, Michael I.; Kim, Deokhwan; Perkins, Jeff H.; Gilham, Limei; Nguyen, Nguyen; Rinard, Martin C.: Information Flow Analysis of Android Applications in DroidSafe. In: Proceedings of the 22nd NDSS, 2015. The Internet Society, 2015.

[Kl14] Klieber, William; Flynn, Lori; Bhosale, Amar; Jia, Limin; Bauer, Lujo: Android taint flow analysis for app sets. In: Proceedings of the 3rd SOAP, 2014. ACM, pp. 5:1–5:6, 2014.

[Li15] Li, Li; Bartel, Alexandre; Bissyandé, Tegawendé F.; Klein, Jacques; Traon, Yves Le; Arzt, Steven; Rasthofer, Siegfried; Bodden, Eric; Octeau, Damien; McDaniel, Patrick D.: IccTA: Detecting Inter-Component Privacy Leaks in Android Apps. In: Proceedings of the 37th ICSE, 2015. IEEE Computer Society, pp. 280–291, 2015.

[PBW18] Pauck, Felix; Bodden, Eric; Wehrheim, Heike: Do Android taint analysis tools keep their promises? In: Proceedings of the 26th ESEC/FSE, 2018. ACM, pp. 331–341, 2018.

[We14] Wei, Fengguo; Roy, Sankardas; Ou, Xinming; Robby: Amandroid: A Precise and General Inter-component Data Flow Analysis Framework for Security Vetting of Android Apps. In: Proceedings of CCS, 2014. ACM, pp. 1329–1341, 2014.

DifFuzz: Differential Fuzzing for Side-Channel Analysis

Shirin Nilizadeh[1] Yannic Noller[2] Corina S. Păsăreanu[3]

Abstract: This summary is based on our research results on "DifFuzz: Differential Fuzzing for Side-Channel Analysis" which was published in the proceedings of the 41st International Conference on Software Engineering [NNP19]. Side-channel analysis aims to investigate the risk that a potential attacker can infer any secret information through observations of the system, such as the execution time or the memory consumption. Side-channel vulnerabilities therefore represent security risks that can cause serious damage and need to be identified and repaired. DifFuzz applies differential fuzzing to identify inputs that trigger such vulnerabilities. Our fuzzing approach analyzes multiple program executions, which vary in their secret information, and uses resource-guided heuristics to identify inputs that maximize the observable cost difference between these executions. Our evaluation shows that such a dynamic analysis approach can find the same side-channel vulnerabilities as state-of-the-art static analysis techniques, and even more vulnerabilities since it does not rely on models for its analysis. Additionally, the advantage of DifFuzz compared to other techniques is not only that it can generate inputs that show a vulnerability, but that the resulting cost difference can also be used to estimate the severity of an identified vulnerability. This enables the comparing of repaired versions of an application.

Keywords: Vulnerability Detection; Side-Channel Analysis; Dynamic Analysis; Fuzzing

Summary

Side-channel vulnerabilities might cause the leakage of sensitive information by observing non-functional characteristics of the program execution, such as the execution time, memory usage, response size, network traffic or power consumption. Popular side-channel attacks like Meltdown and Spectre [Th18] highlight the increased need for tools and techniques that can effectively discover side channels before they are exploited by a malicious user in the field. However, it is difficult to reason about side-channel vulnerabilities as they involve analyzing correlations between resource usage over multiple program paths.

Our approach is similar to the well-known method of *self-composition* [BDR04], which is used to check that no matter what the secret is, the program yields the same output. If that is the case, the program is said to satisfy *non-interference*, meaning that the program leaks *no* information; otherwise, the program is vulnerable. Although satisfying non-interference is a sound guarantee for a system to be secure, this requirement is too strict for the side-channel analysis of most realistic programs. Particularly for timing channels, small differences in computations may be imperceptible to an attacker and can thus not be exploited in practice. This problem was observed before [An17, CFD17] and was formalized as checking

[1] University of Texas at Arlington, USA, shirin.nilizadeh@uta.edu
[2] Humboldt-Universität zu Berlin, Germany, yannic.noller@hu-berlin.de
[3] Carnegie Mellon University Silicon Valley, NASA Ames Research Center, USA, corina.s.pasareanu@nasa.gov

doi:10.18420/SE2020_37

ϵ-bounded non-interference: not only programs with zero interference can be accepted as secure, but also programs where the difference between observations is too small (below a threshold ϵ) to be exploitable in practice.

DifFuzz uses a form of differential fuzzing, in which it analyzes the program with the *same* public inputs but with *different* secret values, and computes the *difference* in the side channel measurements observed over the two executions. Therefore, it *guides* the exploration to find inputs that maximize the cost difference between two program executions, for which only the secret values are different:

$$\underset{pub,sec_1,sec_2}{\text{maximize: }} \delta = |c(P[\![pub, sec_1]\!]) - c(P[\![pub, sec_2]\!])| \tag{1}$$

If the difference is large, then it means that the program has a side-channel vulnerability, which should be remedied by the developer.

We implemented DifFuzz on top of AFL [Za14], a state-of-the-art, security-oriented grey-box fuzzer. Our evaluation showed that DifFuzz can keep up with existing approaches such as Blazer [An17] and Themis [CFD17], two state-of-the-art analysis tools for finding side channels in Java programs. Both tools perform static analysis and can in principle guarantee absence of side channels, but may also give false alarms due to underlying over-approximation. In contrast DifFuzz performs a dynamic analysis, but it can not prove absence of vulnerabilities. Furthermore, DifFuzz found new vulnerabilities in popular open-source Java applications such as Apache FtpServer. In the future, we plan to explore automated repair methods to eliminate the vulnerabilities discovered with DifFuzz.

Bibliography

[An17] Antonopoulos, Timos; Gazzillo, Paul; Hicks, Michael; Koskinen, Eric; Terauchi, Tachio; Wei, Shiyi: Decomposition instead of self-composition for proving the absence of timing channels. In: Proceedings of the 38th ACM SIGPLAN Conference on Programming Language Design and Implementation, PLDI 2017, Barcelona, Spain, June 18-23, 2017. pp. 362–375, 2017.

[BDR04] Barthe, G.; D'Argenio, P. R.; Rezk, T.: Secure information flow by self-composition. In: Proceedings. 17th IEEE Computer Security Foundations Workshop, 2004. pp. 100–114, June 2004.

[CFD17] Chen, Jia; Feng, Yu; Dillig, Isil: Precise Detection of Side-Channel Vulnerabilities using Quantitative Cartesian Hoare Logic. In: Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, CCS 2017, Dallas, TX, USA, October 30 - November 03, 2017. pp. 875–890, 2017.

[NNP19] Nilizadeh, Shirin; Noller, Yannic; Păsăreanu, Corina S.: DifFuzz: Differential Fuzzing for Side-channel Analysis. In: Proceedings of the 41st International Conference on Software Engineering. ICSE '19, IEEE Press, Piscataway, NJ, USA, pp. 176–187, 2019.

[Th18] The Meltdown Attack. `https://meltdownattack.com/`, Accessed: 2019-12-03.

[Za14] American Fuzzy Lop (AFL). `http://lcamtuf.coredump.cx/afl/`, Accessed: 2019-12-03.

Wartung und Evolution 2

Maintaining Consistency across Engineering Artifacts

Alexander Egyed[1], Klaus Zeman[1], Peter Hehenberger[2], Andreas Demuth[3], Larissa Cardoso Zimmermann[1], Roland Kretschmer[1]

Abstract: We summarize the paper Maintaining Consistency across Engineering Artifacts, published at IEEE Computer 51(2), pp. 28-35, 2018. Detecting inconsistencies across multi-domain and multi-tool artifacts is an important and critical task. Inconsistencies may lead to project failures, cost, and schedule overrun - especially when identified incorrectly or late. The paper we summarize explores a technology for consistency checking that is able to automatically and continuously detect inconsistencies - both among knowledge within and across engineering tools.

Keywords: Model-Driven Engineering; Consistency Checking; Cloud Engineering

1 Heading

Software and systems engineering has become a complex and multidisciplinary process. Engineers capture engineering knowledge within their tools – artifacts such as requirements, hardware components, computations, and use cases. Although these artifacts are syntactically and semantically diverse, they are nonetheless interdependent. Hence, artifact inconsistencies may happen not only within individual engineering tools (intratool inconsistency) but also across multiple tools (intertool inconsistency) [Eg11]. The paper presents the Model/Analyzer approach, an automated intratool consistency checker. Its most important capability is providing inconsistency feedback instantly when engineers change artifacts [Eg11]. Engineers thus use the Model/Analyzer to understand the impact of any changes they are making immediately and, if they choose to do so, fix inconsistencies right away. The DesignSpace cloud [De15] provides a common representation and allows its artifacts to be linked. By integrating the Model/Analyzer with the DesignSpace cloud, we demonstrate multi-tool, multi-engineer consistency checking. Tool adapters synchronize the artifacts from tools to the common representation via transformation steps. Within the cloud, the Model/Analyzer then finds all elements needed for consistency checking and communicates the inconsistencies back to the engineers.

References

[De15] Demuth, Andreas; Riedl-Ehrenleitner, Markus; Nöhrer, Alexander; Hehenberger, Peter; Zeman, Klaus; Egyed, Alexander: DesignSpace - An Infrastructure for Multi-User/Multi-Tool Engineering. In: SAC. 2015.

[Eg11] Egyed, A.: Automatically Detecting and Tracking Inconsistencies in Software Design Models. IEEE Transactions on Software Engineering, 37(2):188–204, March 2011.

[1] Johannes Kepler University, alexander.egyed@jku.at, klaus.zeman@jku.at, larissa.cardoso_zimmermann@jku.at, roland.kretschmer@jku.at
[2] University of Applied Sciences Upper Austria
[3] Dynatrace

doi:10.18420/SE2020_38

Comparing Multiple MATLAB/Simulink Models Using Static Connectivity Matrix Analysis

Alexander Schlie[1] Sandro Schulze[2] Ina Schaefer[3]

Abstract: Model-based languages such as MATLAB/Simulink are crucial for the development of embedded software systems. To adapt to changing requirements, engineers commonly copy and modify existing systems to create new variants. Denoted clone-and-own, this straightforward reuse strategy entails severe maintenance and consistency issues as redundant and similar assets proliferate. Software product lines can be a remedy but require all existing variants to be compared prior to their actual migration. However, current work mostly revolves around comparing only two systems and those approaches coping with more are not applicable to embedded software systems such as MATLAB/Simulink. We bridge this gap and propose Static Connectivity Matrix Analysis (SCMA), a novel comparison procedure that evaluates multiple MATLAB/Simulink model variants at once. We transfer models into matrix form and identify all similar structures between them, even with parts being completely relocated during clone-and-own. Moreover, we allow engineers to tailor results and to focus on any arbitrary variant subset, enabling individual reasoning prior to migration. We provide a feasibility study from the automotive domain, showing our matrix representation to be suitable and SCMA to be fast and precise.

Keywords: MATLAB/Simulink; clone-and-own; software product lines; variability; descriptors

Overview

Industrial domains such as automotive, rail and avionic facilitate on embedded systems to develop safety-critical functionality. With reliability and maintainability being pivotal factors, the paradigm of *model-driven engineering (MDE)* prevails in such fields, with languages such as MATLAB/Simulink utilizing function-block based designs. Overall system development remains a challenging and time-intensive task, and to address changing requirements, engineers commonly resort to ad-hoc reuse of existing systems by copying and subsequently modifying them. Denoted clone-and-own, maintenance and evolution of the emerging system portfolio are impaired, as redundancies proliferate and knowledge about commonalities and differences between system variants is rendered void. *Software product lines (SPLs)* can be a remedy, facilitating strategic reuse and promoting maintainability. Unfortunately, the necessity to transitioning towards an SPL practice only becomes evident in the aftermath, then posing an enormous challenge to practitioners. In the context of MDE however, current approaches addressing SPL migration strategies mostly revolve around an incremental comparison of only two related systems. We argue that such restricted evaluation is insufficient for strategically migrating a portfolio towards an SPL, but rather consider a comprehensive assessment of all variants to be indispensable.

[1] TU Braunschweig, Braunschweig, Germany a.schlie@tu-braunschweig.de
[2] Otto-von-Guericke-University in Magdeburg, Magdeburg, Germany sandro.schulze@iti.cs.uni-magdeburg.de
[3] TU Braunschweig, Braunschweig, Germany i.schaefer@tu-braunschweig.de

doi:10.18420/SE2020_39

To compare multiple MATLAB/Simulink model at once, hence addressing aforementioned problems, we propose *Static Connectivity Matrix Analysis (SCMA)*, which we presented at the 34th IEEE International Conference on Software Maintenance and Evolution [SSS18]. With our work, allow for a comprehensive overview of all system variants, and furthermore, for practitioners to focus on arbitrary variant combinations to facilitate strategic decisions prior to portfolio migration. We introduce the *Connectivity Matrix (CM)*, a descriptor, which abstracts salient system information from MATLAB/Simulink models, to reduce their overall complexity to allow for their efficient comparison. Utilizing SCMA and the CMs for comparison respectively, we compare all input models and identify all similar hierarchical structures between them, allowing for a preliminary assessment of the portfolio in its entirety as well as identifying relations between individual variants prior to their SPL migration. We compare all abstracted models, hence all CMs to identify common structures even when respective model parts have been completely relocated during clone-and-own. The CM describes MATLAB/Simulink models, and precisely, each encapsulated sub-model, here *subsystem*, within, by exploiting their inherent graph-structure, and therefore blocks connected via signals. Approximating any such subsystem, the CM reflects which two block functions are directly connected and how often. By that, our CM aims to be easy to extract from the original model and to exhibit a low probability of mismatch, that is, two different models resulting in the same descriptor. With the CM being always of identical dimensions, hence exhibiting the same row and column construction, our SCMA calculates the similarity for any two CMs based the extent of their entries, and thus, similar connections between pairs of functional blocks. Preserving information on the abstracted subsystems' parent-child relation, compared CMs form *trees*, which group together CMs from multiple models within *nodes*, which connect if comprised CMs exhibit a parent-child relation. The output of our SCMA is a *forest*, comprising multiple trees, each reflecting a similar hierarchical structure between multiple input models. Practitioners can tailor results, and for instance, focus on specific variant subsets or define thresholds to evaluate the most similar variants in isolation.

For our evaluation, we apply SCMA to an industrial case-study of MATLAB/Simulink models from the automotive domain to assess performance as well as precision and recall of our approach. Moreover, we investigate the suitability of our descriptor, the CM, to abstract MATLAB/Simulink models for their efficient comparison. Our results show SCMA to exhibit a reasonable runtime given the assessment of 18 model variants, comprising $\approx$15.000 blocks in total. Moreover, our evaluation reveals SCMA to exhibit a high precision, total recall and shows our CM to be a suitable descriptor for MATLAB/Simulink software systems. Overall, we argue that SCMA allows for an efficient comparison of an entire model portfolio at large, recreating information about commonalities across several models to assist practitioners with the daunting task of migrating embedded systems towards an SPL fashion.

Bibliography

[SSS18] Schlie, Alexander; Schulze, Sandro; Schaefer, Ina: Comparing Multiple MATLAB/Simulink Models Using Static Connectivity Matrix Analysis. In: Proc. of the Intl. Conference on Software Maintenance and Evolution (ICSME). IEEE, pp. 160–171, 2018.

Integrated Revision and Variation Control for Evolving Model-Driven Software Product Lines[1]

Felix Schwägerl[2], Bernhard Westfechtel[3]

Abstract: Software engineering projects are faced with abstraction, which is achieved by software models, historical evolution, which is addressed by revision control, and variability, which is managed with the help of software product line engineering. Addressing these phenomena by separate tools ignores obvious overlaps and therefore fails at exploiting synergies between revision and variation control for models. In this article, we present a conceptual framework for integrated revision and variation control of model-driven software projects and its implementation in the tool SuperMod.

Keywords: Model-Driven Software Engineering; Revision Control; Variation Control

1 Background

This work is concerned with the integration of three disciplines of software engineering: In *model-driven software engineering*, software systems are developed from high-level models that are analyzed, simulated, or transformed into source code. *Software product line engineering* denotes a systematic reuse process that supports the efficient development of instances of a product line from a set of shared artifacts. *Software configuration management* is the discipline of managing the evolution of complex software systems; in particular, it includes version control for software engineering artifacts such as models and source code.

To some extent, model-driven software engineering, software product line engineering, and software configuration management have been evolving independently: Model-driven software engineering focuses on models and model transformations without considering evolution in time and space. Software product line engineering addresses evolution in space only, primarily dealing with source code. Software configuration management is concerned with evolution in time, focusing on file-based artifacts. Partial integration of these disciplines has been studied. For example, model version control deals with historical evolution of models, and model-driven software product line engineering applies product line engineering to model- rather than code-based artifacts.

[1] This paper is an extended abstract of [SW19].

[2] Universität Bayreuth, Lehrstuhl für Angewandte Informatik 1, Universitätsstraße 30, 95440 Bayreuth, Germany
felix.schwaegerl@uni-bayreuth.de

[3] Universität Bayreuth, Lehrstuhl für Angewandte Informatik 1, Universitätsstraße 30, 95440 Bayreuth, Germany
bernhard.westfechtel@uni-bayreuth.de

doi:10.18420/SE2020_40

2 Conceptual Framework

In our work, we developed a *conceptual framework* for *managing evolving software product lines*. This framework is based on earlier work on a *uniform version model* for the domain of software configuration management [WMC01] and distinguishes between different layers of abstraction. The *base layer* provides generic support for versioning of artifacts of arbitrary types. Versioned artifacts are represented as *superimposition* of versioned elements. As in conditional compilation or annotative approaches to software product line engineering, versioned elements are decorated with *visibilities*, i.e., expressions that determine the versions in which an element is included. *Version rules* are used to constrain the combination of truth values that may be assigned to the variables occurring in visibilities.

On top of the base layer, user-level models for revision and variation control are realized. Here, familiar abstractions from software product line engineering and software configuration management are reused. *Revision control* (evolution in time) is supported by *revision graphs*. Each revision denotes an immutable snapshot of a software product line. For *variation control* (evolution in space), *feature models* are provided. Feature models are versioned in time; domain artifacts (models and source code) evolve in both time and space.

3 Tool Support

Based on the conceptual framework described above, we developed *SuperMod* (*Super*imposition of *Mod*els), a tool for managing evolving model-driven software product lines. The tool maintains *repositories* of versioned artifacts such as feature models, domain models, and source code. Furthermore, SuperMod offers operations such as *check-out* and *commit* to populate *workspaces* and record changes performed in the workspaces, respectively.

While the check-out/commit cycle was inspired by classical version control tools known from software configuration management, SuperMod differs considerably from version control tools with respect to variation control. On check-out, the user selects a revision first and then proceeds by selecting a variant. Thus, the workspace is populated with a unique product version. Both the feature model and the domain artifacts may be changed in the workspace. On commit, the set of variants is determined which are affected by the changes performed in the workspace. Altogether, this approach is called *dynamic filtered editing*.

Bibliography

[SW19] Schwägerl, Felix; Westfechtel, Bernhard: Integrated revision and variation control for evolving model-driven software product lines. Software and Systems Modeling, 18(6):3373–3420, December 2019.

[WMC01] Westfechtel, Bernhard; Munch, Bjørn P.; Conradi, Reidar: A Layered Architecture for Uniform Version Management. IEEE Transactions on Software Engineering, 27(12):1111–1133, 2001.

Testing 2

Too trivial to test? An inverse view on defect prediction to identify methods with low fault risk

Rainer Niedermayr[1] Tobias Röhm[2] Stefan Wagner[3]

Abstract: To cope with the scarce resources for testing, teams can apply defect prediction to identify fault-prone code regions. However, defect prediction tends to low precision in cross-project prediction scenarios. We take an inverse view on defect prediction and aim to identify methods that can be deferred when testing because they contain hardly any faults due to their code being "trivial". We compute code metrics and apply association rule mining to create rules for identifying methods with low fault risk (LFR) and assess our approach with six Java open-source projects containing precise fault data at the method level. Our results show that inverse defect prediction can identify approx. 32–44% of the methods of a project to have a LFR; on average, they are about six times less likely to contain a fault than other methods. Our approach identifies methods that can be treated with less priority in testing activities and is well applicable in cross-project prediction scenarios.

Keywords: Testing; Inverse defect prediction; Fault risk; Low-fault-risk methods

1 Introduction

In a perfect world, it would be possible to completely test every new version of a software application before it was deployed into production. In practice, however, software development teams often face a problem of scarce test resources. Hence, development teams need to prioritise and limit their testing scope by restricting the code regions to be tested. Defect prediction identifies code regions that are likely to contain a fault and should therefore be tested.

This paper suggests, implements, and evaluates another view on defect prediction: inverse defect prediction (IDP). The idea is to identify code artefacts that are so trivial that they contain hardly any faults and thus can be deferred or ignored in testing. IDP also uses a set of metrics that characterise artefacts, applies transformations to pre-process metrics, and uses a machine-learning classifier to build a prediction model. The difference rather lies in the predicted classes: IDP identifies methods that exhibit a low fault risk (LFR) with high certainty and does not make an assumption about the remaining methods. As a consequence, the objective of the prediction also differs. Defect prediction aims to achieve a high recall, such that as many faults as possible can be detected, and a high precision, such that only few false positives occur. In contrast, IDP aims to achieve high precision to ensure that

[1] CQSE GmbH, Munich, Germany niedermayr@cqse.eu
[2] CQSE GmbH, Munich, Germany roehm@cqse.eu
[3] University of Stuttgart, Stuttgart, Germany stefan.wagner@iste.uni-stuttgart.de

doi:10.18420/SE2020_41

LFR methods contain indeed hardly any faults, but it does not necessarily seek to predict all non-faulty methods. This is a summary of the full article published in PeerJ Computer Science in 2019 [NRW19].

2 IDP Approach

In the IDP approach, we use 39 source code metrics for Java such as maximum nesting depth or the number of if conditions as well as two derived metrics. Furthermore, we take into account whether the method to be analysed is a constructor, a getter/setter, an empty method, delegation method or a toString method. We use these metrics to create rules that indicate when a method is non-faulty using association rule mining.

3 Empirical Study

Using the Defects4J data [JJE14], we could evaluate the IDP approach on six open source Java systems. We found that our approach on average only 0.3% of the methods classified as low fault risk are actually faulty. Those identified methods are 5.7 times less likely to contain a fault. About 15–20% of the lines of code are identified as low fault risk.

4 Conclusion

We show that IDP using association rule mining on code metrics can successfully identify LFR methods. The identified methods contain considerably fewer faults than the average code and can provide a savings potential for QA activities. Our results furthermore suggest that the IDP approach can be applied in a cross-project prediction scenario at the method level.

Bibliography

[JJE14] Just, René; Jalali, Darioush; Ernst, Michael D.: Defects4J: a database of existing faults to enable controlled testing studies for Java programs. In: International Symposium on Software Testing and Analysis, ISSTA '14. ACM, pp. 437–440, 2014.

[NRW19] Niedermayr, Rainer; Röhm, Tobias; Wagner, Stefan: Too trivial to test? An inverse view on defect prediction to identify methods with low fault risk. PeerJ Computer Science, 5:e187, 2019.

Analyzing Scratch Programs with Automated Tests

Andreas Stahlbauer[1] Marvin Kreis[1] Gordon Fraser[1]

Abstract: Block-based programming environments like Scratch foster engagement with computer programming and are used by millions of young learners. Scratch allows learners to quickly create entertaining programs and games, while eliminating syntactical program errors that could interfere with progress. However, functional programming errors may still lead to incorrect programs, and learners and their teachers need to identify and understand these errors. This is currently an entirely manual process. We introduce a formal testing framework that describes the problem of Scratch testing in detail, and instantiate this formal framework with the Whisker tool, which provides automated and property-based testing functionality for Scratch programs. Empirical evaluation on real student and teacher programs demonstrates that Whisker can successfully test Scratch programs, and automatically achieves an average of 95.25 % code coverage. This opens up new possibilities to support learners of programming in their struggles. This summary refers to the article "*Testing Scratch Programs Automatically*" [SKF19] published at the 27th ACM SIGSOFT International Symposium on Foundations of Software Engineering 2019.

1 Introduction

Block-based programming languages [Ba17] like Scratch [Ma10] engage young learners with computer programming. At the time of this writing, almost 50 million users have signed up to Scratch and have publicly shared almost as many Scratch projects, and there are many similar programming languages and environments. An important advantage for young learners is that it is only possible to assemble the blocks that represent programming statements in syntactically valid ways, thus eliminating the need to remember syntactic subtelties. Programs, however, can nevertheless be functionally incorrect, and nothing about the block-based approach makes it easier to identify and correct functional mistakes than in regular programming languages. In fact, the Scratch programming environment offers little support to debug Scratch programs, and the visual arrangement of scripts can easily get confusing with increasing program size, making it very hard to check and fix programs.

Our aim is to enable automated analysis of Scratch programs. To this purpose, we introduce an automated testing framework as a prerequisite to enable dynamic analysis, for example in order to locate bugs or provide suggestions and feedback to learners. An immediate application considered in this paper is the role of a teacher, who needs to identify faulty programs in classroom and during grading. A study on real student programs shows that automated test generation can achieve high code coverage on these programs, and the automated tests highly correlate with manual grading results.

[1] Universität Passau, Lehrstuhl für Software Engineering II, Innstrasse 33, 94032 Passau

doi:10.18420/SE2020_42

2 Whisker: An Automated Testing Framework for Scratch

In Scratch, programs control different objects (*sprites*) that are represented and interact on a graphical *stage*. Programs are created by visually combining blocks that represent syntactical elements of the language, such as control-flow structures. Only blocks that lead to syntactically valid programs can be combined. The stage and each of the sprites can contain several scripts. An event-based programming model is used, and many of the scripts represent event handlers (e.g., message broadcasts, sprite collisions, user input).

A test consists of user events (e.g., mouse clicks and key presses) as well as a specification of the expected behaviour. In our Whisker testing framework, tests are operationalized in JavaScript, and test execution wraps the Scratch virtual machine in order to provide events to the program under test, as well as to check the specified properties. Automated tests can be written manually in JavaScript, or generated automatically by Whisker.

In a study with 37 student implementations of the same program, we demonstrate that automated tests show a strong correlation of 0.893 (p-value < 0.001) with the grading points assigned manually by a teacher. Using a set of 24 popular programming projects frequently used in programming education, we demonstrate that Whisker can achieve an average statement coverage of 95.25% with its automated test generation.

3 Conclusions

Our experiments demonstrate that Whisker tests can be used to automatically test and grade Scratch programs. An important next step is support for deriving specifications, for which we are investigating ways to infer them from executions on a model solution. Furthermore, we envision many further applications of automated tests. The educational nature of Scratch interactions would benefit from the dynamic analysis enabled by a testing framework like Whisker. Applications range from supporting learners with fault localization to producing hints and repairs automatically. To support this area of research, the latest version of Whisker is available as open source at `github.com/se2p/whisker-main`.

Bibliography

[Ba17] Bau, D.; Gray, J.; Kelleher, C.; Sheldon, J.; Turbak, F. A.: Learnable programming: blocks and beyond. Commun. ACM, 60(6):72–80, 2017.

[Ma10] Maloney, J.; Resnick, M.; Rusk, N.; Silverman, B.; Eastmond, E.: The Scratch Programming Language and Environment. TOCE, 10(4):16:1–16:15, 2010.

[SKF19] Stahlbauer, Andreas; Kreis, Marvin; Fraser, Gordon: Testing Scratch Programs Automatically. In: Proceedings of the 2019 27th ACM International Symposium on the Foundations of Software Engineering. ACM, pp. 165–175, 2019.

Predicting How to Test Requirements: An Automated Approach

Jonas Winkler[1] Jannis Grönberg[2] Andreas Vogelsang[3]

Abstract: An important task in requirements engineering is to identify and determine how to verify a requirement (e.g., by manual review, testing, or simulation; also called *potential verification method*). This information is required to effectively create test cases and verification plans for requirements. In this paper, we propose an automatic approach to classify natural language requirements with respect to their potential verification methods (PVM). Our approach uses a convolutional neural network architecture to implement a multiclass and multilabel classifier that assigns probabilities to a predefined set of six possible verification methods, which we derived from an industrial guideline. Additionally, we implemented a backtracing approach to analyze and visualize the reasons for the network's decisions. In a 10-fold cross validation on a set of about 27,000 industrial requirements, our approach achieved a macro averaged F_1 score of 0.79 across all labels. The results show that our approach might help to increase the quality of requirements specifications with respect to the PVM attribute and guide engineers in effectively deriving test cases and verification plans.

Keywords: Requirements Engineering, Requirements Validation, Test Engineering, Machine Learning, Natural Language Processing, Neural Networks

1 Introduction

Verifiability is a quality characteristic for requirements that is mentioned in many normative quality standards such as ISO 29148 and the IREB. One of our industry partners has introduced an explicit requirements attribute called *Potential Verification Method (PVM)* that specifies in which ways a requirement must be verified. Possible PVMs are Review, Simulation, Formal Verification, Process Audit, System Test and Production Control. Setting values for this attribute is a manual, time-consuming, and error-prone task.

We propose an automatic approach to classify natural language requirements with respect to their potential verification methods [WGV19]. Additionally we visualize the importance of parts of the input sentence for the classification decision. We conclude that our automated approach helps to increase the quality of requirements by detecting misclassified PVM attributes or automatically generating classification proposals for unlabeled requirements. In this paper, we present a brief overview of the technique and its applications.

[1] Technische Universität Berlin, DCAITI, jonas.winkler@tu-berlin.de
[2] Technische Universität Berlin, DCAITI, jannis.r.groenberg@campus.tu-berlin.de
[3] Technische Universität Berlin, DCAITI, andreas.vogelsang@tu-berlin.de

doi:10.18420/SE2020_43

2 Classifying requirements with respect to the Potential Verification Method

Our classifier takes a requirement and uses a convolutional neural network (CNN) to assign labels to it, which represent different PVM values. First an input sentence is transformed into vector representation (i.e. word embedding). The first layer in the CNN applies a set of filters by moving them as a sliding window over the sentence vector, producing single values at each position (i.e. convolution). The most important features are concatenated and form a feature vector that is connected to the output layer from which a probability between 0 and 1 is derived for each of the six corresponding possible PVM values.

Classifier	Accuracy	Perfect Match Ratio	Macro-F_1	Micro-F_1
ZeroR baseline	0.8477	0.8293	0.1553	0.8323
CNN	**0.9310**	0.9115	**0.7904**	**0.9368**
Example Sentence (System Test)	The actuators and switches must be activated separately within the control unit.			

Tab. 1: Results for the CNN based PVM classifier

We compared the classifier against a ZeroR baseline. Since 84% of the requirements in our dataset contain the class *System Test*, the ZeroR baseline has high values for *Accuracy*, *Perfect Match Ratio* and *Micro-F1*. Table 1 reveals that our CNN-based classifier performs substantially better than the baseline. We traced back the probabilities in our CNN and derived important key phrases for each class. The highlighting for the class *System Test* in the table shows individual words that are especially important for the classification process of the CNN in this particular sentence.

3 Applications & Conclusions

Our classifier can be used for detecting misclassified PVM attributes or automatically generating classification proposals for unlabeled requirements. As shown in previous studies [WV18], integrating a classifier into a tool may provide certain benefits such as shorter review time and increased number of errors fixed.

Literaturverzeichnis

[WGV19] Winkler, J. P.; Grönberg, J.; Vogelsang, A.: Predicting How to Test Requirements: An Automated Approach. In: 2019 IEEE 27th International Requirements Engineering Conference (RE). S. 120–130, Sep. 2019.

[WV18] Winkler, Jonas P.; Vogelsang, Andreas: Using Tools to Assist Identification of Non-Requirements in Requirements Specifications - A Controlled Experiment. In: Requirements Engineering: Foundation for Software Quality (REFSQ). 2018.

Performance und Benchmarking

Renaissance: Benchmarking Suite for Parallel Applications on the JVM

Aleksandar Prokopec[1], Andrea Rosà[2], David Leopoldseder[3], Gilles Duboscq[4], Petr Tůma[5], Martin Studener[6], Lubomír Bulej[7], Yudi Zheng[8], Alex Villazón[9], Doug Simon[10], Thomas Würthinger[11], Walter Binder[12]

Abstract: Our paper published in the proceedings of the 40th ACM SIGPLAN Conference on Programming Language Design and Implementation (PLDI 2019) [Pr19a] proposes Renaissance, a modern benchmark suite whose aim is to advance compiler and virtual machine (VM) research. The publication is complemented by an accepted artifact [Pr19b]. To demonstrate that a compiler optimization, a memory management algorithm, or a synchronization technique is useful, a VM researcher needs benchmarks that demonstrate the desired behavior, and at the same time capture representative aspects of real-world applications. During the last decade, multiple new programming paradigms appeared on the Java VM (JVM), including functional programming, big-data processing, parallel and concurrent programming, message-passing, stream processing, and machine learning. The JVM has evolved as a platform too: new features, such as method-handles, variable-handles, the invokedynamic instruction, lambdas, atomic and relaxed memory operations, present new challenges for dynamic compilers and runtime environments. Existing benchmark suites do not capture the new applications, because they were made in a time when these workloads did not exist. Renaissance bridges this gap. The Renaissance suite is an ongoing, open-source effort to collect representative real-world workloads, and to advance the research and development of VMs. Renaissance is available at `https://renaissance.dev/`

Keywords: benchmarks; JIT compilation; parallelism; concurrency; JVM; object-oriented programming benchmarks; functional-programming benchmarks; big-data benchmarks

Acknowledgments: This work has been supported by Oracle (ERO project 1332) and by the Swiss National Science Foundation (project 200020_188688).

[1] Oracle Labs, Switzerland aleksandar.prokopec@oracle.com
[2] Università della Svizzera italiana, Switzerland andrea.rosa@usi.ch
[3] Johannes Kepler Universität Linz, Austria david.leopoldseder@jku.at
[4] Oracle Labs, Switzerland gilles.m.duboscq@oracle.com
[5] Charles University, Czech Republic petr.tuma@d3s.mff.cuni.cz
[6] Johannes Kepler Universität Linz, Austria martinstudener@gmail.com
[7] Charles University, Czech Republic bulej@d3s.mff.cuni.cz
[8] Oracle Labs, Switzerland yudi.zheng@oracle.com
[9] Universidad Privada Boliviana, Bolivia avillazon@upb.edu
[10] Oracle Labs, Switzerland doug.simon@oracle.com
[11] Oracle Labs, Switzerland thomas.wuerthinger@oracle.com
[12] Università della Svizzera italiana, Switzerland walter.binder@usi.ch

doi:10.18420/SE2020_44

Bibliography

[Pr19a] Prokopec, Aleksandar; Rosà, Andrea; Leopoldseder, David; Duboscq, Gilles; Tůma, Petr; Studener, Martin; Bulej, Lubomír; Zheng, Yudi; Villazón, Alex; Simon, Doug; Würthinger, Thomas; Binder, Walter: Renaissance: Benchmarking Suite for Parallel Applications on the JVM. Proceedings of the 40th ACM SIGPLAN Conference on Programming Language Design and Implementation (PLDI), Phoenix, AR, USA, June 22-26, 2019, 2019.

[Pr19b] Prokopec, Aleksandar; Rosà, Andrea; Leopoldseder, David; Duboscq, Gilles; Tůma, Petr; Studener, Martin; Bulej, Lubomír; Zheng, Yudi; Villazón, Alex; Simon, Doug; Würthinger, Thomas; Binder, Walter: Supplementatry material - Artifact for the paper "Renaissance: Benchmarking Suite for Parallel Applications on the JVM" published in the proceedings of the 40th ACM SIGPLAN Conference on Programming Language Design and Implementation (PLDI 2019). DOI: 10.1145/3325986. 2019.

Analysis and Optimization of Task Granularity on the Java Virtual Machine

Andrea Rosà[1] Eduardo Rosales[2] Walter Binder[3]

Abstract: Our article published in ACM Transactions on Programming Languages and Systems (TOPLAS) [RRB19] (which extends our work published in the proceedings of the 2018 IEEE/ACM International Symposium on Code Generation and Optimization (CGO 2018) [RRB18]) presents a new methodology to accurately and efficiently collect the granularity of each executed task. Task granularity, i.e., the amount of work performed by parallel tasks, is a key performance attribute of parallel applications. On the one hand, fine-grained tasks may introduce considerable parallelization overheads. On the other hand, coarse-grained tasks may not fully utilize the available CPU cores, leading to missed parallelization opportunities. We implement our methodology in tgp, a novel task-granularity profiler that collects carefully selected metrics from the whole system stack with low overhead, and helps developers locate performance and scalability problems. We analyze task granularity in the DaCapo, ScalaBench, and Spark Perf benchmark suites, revealing inefficiencies related to fine-grained and coarse-grained tasks in several applications. We demonstrate that the collected task-granularity profiles are actionable by optimizing task granularity in several applications, achieving speedups up to a factor of 5.9x. tgp is available open-source at `https://github.com/fithos/tgp/`

Keywords: Task granularity; task parallelism; performance analysis and optimization; vertical profiling; actionable profiles; Java Virtual Machine

Acknowledgments: This work has been supported by Oracle (ERO project 1332) and by the Swiss National Science Foundation (project 200020_188688).

Bibliography

[RRB18] Rosà, Andrea; Rosales, Eduardo; Binder, Walter: Analyzing and Optimizing Task Granularity on the JVM. In: Proceedings of the 2018 International Symposium on Code Generation and Optimization. CGO 2018, ACM, New York, NY, USA, pp. 27–37, 2018.

[RRB19] Rosà, Andrea; Rosales, Eduardo; Binder, Walter: Analysis and Optimization of Task Granularity on the Java Virtual Machine. ACM Trans. Program. Lang. Syst., 41(3):19:1–19:47, July 2019.

[1] Università della Svizzera italiana, Switzerland andrea.rosa@usi.ch
[2] Università della Svizzera italiana, Switzerland rosale@usi.ch
[3] Università della Svizzera italiana, Switzerland walter.binder@usi.ch

doi:10.18420/SE2020_45

Representative Load Testing in Continuous Software Engineering: Automation and Maintenance Support

Henning Schulz[1], André van Hoorn[2]

Abstract: This extended abstract summarizes our work on reducing the maintenance effort for the parameterization of representative load tests using annotations, which we have published in the Journal of Software Testing, Verification & Reliability in 2019 [SHW19].

Representative load testing [JH15] can effectively test and preserve the performance before delivery by mimicking the expected workload. However, it also requires a notable amount of expertise and effort for creating, maintaining, and executing the load test. During load test creation, experts need to take care of different workload scenarios occurring in the production system, which the test needs to replay. Maintenance tasks comprise updating the test due to changing workloads and APIs of the system under test (SUT).

For easing representative load testing and reducing its effort, we aim at automatically generating tailored representative load tests [SAH18]. Based on an expert's request and continuously recorded monitoring data, our approach generates a load test tailored to a particular service and replaying a particular workload observed in production. However, while we can leverage existing approaches for extracting the workload model automatically, generating a load test also requires proper parameterization — e.g., specifying credentials for the simulated users that are valid in the test environment — typically entailing manual effort. The manual effort accumulates if the expert generates multiple load tests or if the workload or APIs change. In the context of continuous software engineering, such manual effort prevents the integration into the highly-automated build and delivery infrastructure.

Therefore, our approach reduces the manual effort for parameterization and completely automates the test generation [SHW19]. As shown in Fig. 1, instead of parameterizing a load test directly, an expert can define all parameterizations in a separate model: the input data and properties annotation (IDPA) (1). Leveraging API specifications, our approach can generate and update parts of an IDPA automatically (2). Also, we provide mechanisms for evolving an IDPA over API changes. Hence, the expert can maintain the IDPA and store it at a central place such as the code repository (3). The application running in production and being used by the end-users collects the user's requests and sessions and stores them into a measurement repository (4). Hence, when a new load test is to be generated — e.g., inside a continuous integration and delivery (CI/CD) pipeline — our approach can retrieve an up-to-date IDPA and measurement data (5). It extracts a workload model — e.g., replaying

[1] Novatec Consulting GmbH, Karlsruhe, Germany, henning.schulz@novatec-gmbh.de
[2] University of Stuttgart, Stuttgart, Germany, van.hoorn@informatik.uni-stuttgart.de

doi:10.18420/SE2020_46

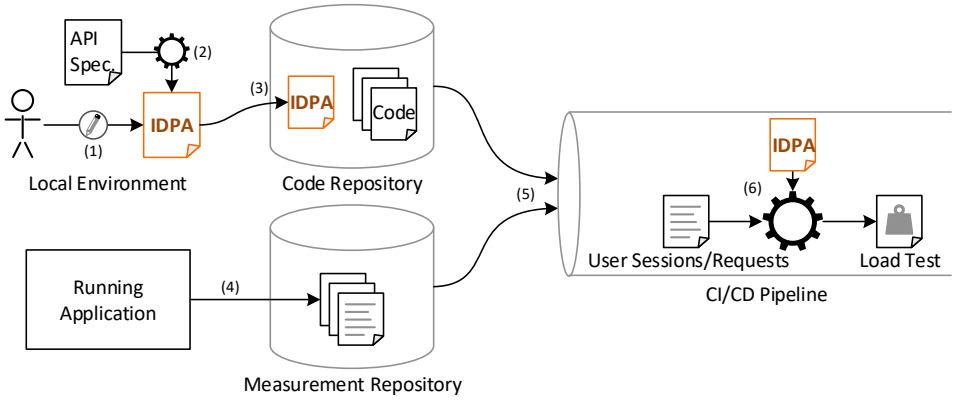


Fig. 1: Overview of the load test parameterization process using IDPAs [SHW19].

the recently typical workload — from the user sessions and requests, transforms it to a load test, and parameterizes it using the IDPA (6). Finally, the pipeline can automatically execute the generated load test without manual intervention.

Our evaluation, consisting of two experimental studies, effort estimation models, and an industrial case study, shows the applicability of our approach. Given the workload is dominated by the order and rate of the requests rather than the input data, it restores the representativeness of generated load tests while it reduces the manual effort from a quadratic to a linear function of time. Details on the evaluation are presented in the original article [SHW19]. The evaluation artifacts and a demo are provided online.[3]

Acknowledgement This work is part of the ContinuITy[4] project, which is supported by the German Federal Ministry of Education and Research (grant no. 01IS17010).

References

[JH15] Jiang, Z. M.; Hassan, A. E.: A Survey on Load Testing of Large-Scale Software Systems. IEEE Trans. on Softw. Eng. 41/11, pp. 1091–1118, 2015.

[SAH18] Schulz, H.; Angerstein, T.; van Hoorn, A.: Towards Automating Representative Load Testing in Continuous Software Engineering. In: Proc. ACM/SPEC International Conference on Performance Engineering (ICPE 2018) Companion. ACM, pp. 123–126, 2018.

[SHW19] Schulz, H.; van Hoorn, A.; Wert, A.: Reducing the Maintenance Effort for Parameterization of Representative Load Tests Using Annotations. Software Testing, Verification & Reliability/, in press (early preview status), 2019.

[3] Artifacts: `https://doi.org/10.5281/zenodo.3255389`, demo: `https://doi.org/10.5281/zenodo.2647976`
[4] ContinuITy web site: `https://continuity-project.github.io/`

Forschungsmethoden im Software Engineering

When Grounded Theory is Not Enough: Additions for Video-Based Analyses of Software Engineering Process Phenomena

A Research Methods Discussion

Franz Zieris[1]

Abstract: I discuss how the practices of the Grounded Theory Methodology are not enough to conduct qualitative research on process phenomena of software development based on video recordings. I present five extensions and auxiliary practices to fill in the gaps.

Keywords: Grounded Theory Methodology; Qualitative Research

1 Introduction and Background

My research group has been investigating the development practice of pair programming (PP) since 2004. To understand how PP actually works, we record sessions (consisting of screencast, webcam, and audio) of industrial developers working in pairs on their everyday tasks. Our ultimate goal is to provide practitioners with actionable advice.

We tried to apply Grounded Theory Methodology (GTM) in its Straussian form [SC90], in particular open, axial, and selective coding, and made observations of two kinds: (1) the GTM coding practices are not readily applicable to our kind of data; (2) the Straussian GTM alone (theoretical sampling and applying the coding practices until theoretical saturation) does not lead to a qualitative study with the desirable properties of naturalistic and holistic inquiry, rich and dense data, and active consideration of the researcher role. Here, I present five additions of the GTM from our group (some already reported in [SPP08]), which I believe to be helpful tools for other researchers considering the GTM to study SE process phenomena.

2 Additions: Extensions and Auxiliary Practices

Data Collection (1): Observations & Interviews. Both Straussian advice on open coding and qualitative SE research in general are often focused on (non-naturalistic) *interviews*.

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

doi:10.18420/SE2020_47

Observations, in contrast, are not limited to what subjects are aware of and able to express. Our research combines both: We record PP sessions as our main data source (observation) and afterwards reflect with the developers on their recent session (interview) which combines capturing their personal accounts and checking findings for resonance. Others may have reversed roles: Rely on interviews supported by observation.

Data Collection (2): Stay in the Field & Blend. The researcher role in the field is not discussed in Straussian GTM. Instead of collecting data on single events, we stay at industrial partners multiple days or weeks on end. In assuming a hybrid role of researcher and software engineer, we also talk to developers on a technical level about their everyday issues, thereby establishing trust and collecting *context* information valuable for interpretation later on.

Sampling: Repository. Establishing trust with a company and collecting data costs time and effort. Theoretical sampling in the sense of *collecting* new data when needed is often impractical. A *repository* built over years organizing raw data (here: session recordings) and metadata is a source for ready-to-analyze data from multiple companies for different studies.

Analysis (1): Filter for Data. Open coding and theoretical sensitivity alone lead to too many concepts when the phenomenon is non-trivial and data is rich. Our case is both: Video recordings of two developers working on a task from their domain at their own speed without the need to explain it to an interviewer are *dense*. Defining a *filter* for what phenomena to look out for and which epistemological stance to take (e.g., to only analyze knowledge transfer based on observable explicit behavior) is necessary to not drown in concepts.

Analysis (2): Reusable Concepts. The GTM requires all concepts to be grounded in data, but studying complex phenomena in one take leads to shallow concepts. To resolve this contradiction, we propose to (1) develop grounded concepts in a reusable way including naming schemes and operationalization rules, and (2) reuse such concepts where fit to avoid reinventing the wheel. For our research, we developed a set of base activities of a PP process to capture developers' primary intentions [SP13], which proved valuable for jump-starting focused analyses (e.g., on knowledge transfer) multiple times.

References

[SC90] Strauss, A. L.; Corbin, J. M.: Basics of Qualitative Research: Grounded Theory Procedures and Techniques. SAGE Publications, Inc, 1990.

[SP13] Salinger, S.; Prechelt, L.: Understanding Pair Programming: The Base Layer. BoD, Norderstedt, Germany, 2013.

[SPP08] Salinger, S.; Plonka, L.; Prechelt, L.: A Coding Scheme Development Methodology Using Grounded Theory for Qualitative Analysis of Pair Programming. Human Technology: An Interdisciplinary Journal on Humans in ICT Environments 4/1, pp. 9–25, 2008, ISSN: 1795-6889, URL: http://urn.fi/URN:NBNfi:jyu-200804151350.

Tailor Made: Situational Method Engineering for Empirical SE Research

An Experience Report

Stefan Sobernig[1]

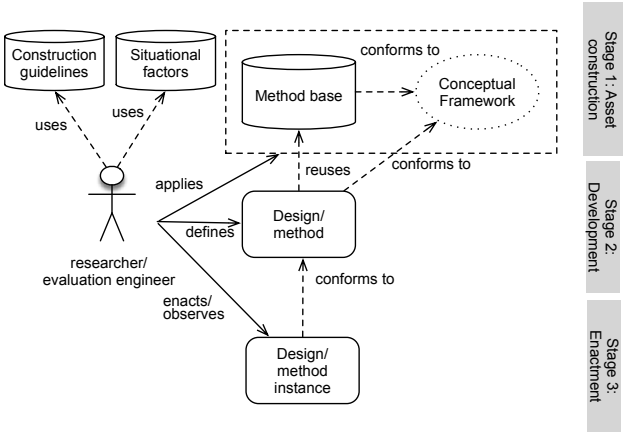


Fig. 1: Basic tenets of engineering situational (contextual) empirical designs and methods (inspired by [He14]).

Custom-Fit Empirical Designs and Methods Research and practice on designing adequate software-development methods and processes ("method engineering") has for long acknowledged the need for *method customization* to match an organization's or team's needs. Numerous approaches to engineering situational development methods and processes have been proposed over the years. See [He14] for a comprehensive overview. As for methods and processes for validating and for evaluating software-engineering artifacts (methods, techniques, and tools) as part of empirical projects with or in industry, one-size-fits-all approaches dominate. This is despite the fact that the necessity of a situated, contextual artifact validation or artifact evaluation has been raised before. Regarding artifact evaluation in industry, situated designs and methods can be blended with existing project and development procedures. As for artifact validation in a research project with industry, situated designs and methods can contribute to explore contextual variation [RSA14] and to

[1] WU Vienna, Wien, Austria, stefan.sobernig@wu.ac.at

doi:10.18420/SE2020_48

mitigating important threats to validity (e.g., construct validity). In praxis, situated methods lower entry barriers for empirical researchers, when engaging with practitioners and their organisations (e.g., by adopting data-collection techniques already accepted within an organisation).

Assessing Domain-specific Developer Tools in Industry Since 2017, we have contributed to a series of industry research projects (e.g., DLUX, HybriDLUX) on introducing domain-specific modeling and programming tools into software-development organisations in the automotive domain [St18]. In this context, we gathered experiences on engineering validation and evaluation methods specific to the partner organisations. In particular, we leveraged review techniques (walkthroughs) accepted among the target audience (firmware developers and testers) as instruments for mixed-method empirical designs. The designs aimed at assessing custom-made developer tools (DSL-based IDEs) for their quality-in-use characteristics.

Outline The talk will reflect on the potential and the limitations to situational method engineering for empirical research and evaluation. This reflection builds on our project experiences and challenges existing approaches to validating/ evaluating domain-specific developer tooling (e.g., FQAD, USE-ME).

- Motivation
- State of things: Situational engineering of empirical designs and methods (e.g., DESMET, 2G, ISO 14102)
- Challenges
  - Inventorising existing practises as a method base
  - Piggybacking onto existing practises (data gathering, daily work activities) versus requirements of empirical designs (participant selection, representation conditions)
  - Change agents: Validation/ evaluation efforts vs. organizational change processes
- An agenda for research and praxis

Bibliography

[He14] Henderson-Sellers, B.; Ralyté, J.; Ågerfalk, P.; Rossi, M.: Situational Method Engineering. Springer, 2014.

[RSA14] Runeson, Per; Stefik, Andreas; Andrews, Anneliese: Variation factors in the design and analysis of replicated controlled experiments. Empirical Software Engineering, 19(6):1781–1808, 2014.

[St18] Stieglbauer, Gerald; Burghard, Christian; Sobernig, Stefan; Korosec, Robert: A Daily Dose of DSL - MDE Micro Injections in Practice. In: Proceedings of the 6th International Conference on Model-Driven Engineering and Software Development (MODELSWARD'18). SciTePress, pp. 642–651, 2018.

Sind wissenschaftliche Praxis und Vergleichbarkeit in der Forschung zum Software Engineering „gut genug“?

Rupert Schlick[1]

Keywords: Vergleichbarkeit; Reproduzierbarkeit; Software Engineering; Praxisbezug

1 Situation

Im Vergleich zu Biologie, Psychologie, Medizin oder Physik, sind die Inhalte und Artefakte mit denen sich Software Engineering und die Computerwissenschaften generell beschäftigen, in hohem Maße detailliert aufzeichenbar, archivierbar und wiederverwendbar. Stärker eingeschränkt wird das nur im Bereich des Requirements Engineering, beim Vergleich der kreativen Prozesse die zu verschiedenen und verschieden korrekten Lösungen führen und wenn das erstellte Software-System mit nicht wiederholbaren (Sensor-)Daten aus der physischen Welt in Kontakt kommt.

Trotzdem ist es nicht generell üblich, die zu einer Publikation gehörenden Artefakte zu veröffentlichen. Konferenzen fordern zwar zunehmend die Einreichung von Artefakten, diese werden der restlichen Forschungsgemeinschaft aber oft nicht zugänglich gemacht. Die Initiative der Europäischen Kommission zu Open Data zeigt noch wenig Wirkung.

Für einzelne Anwendungsbereiche sind Benchmarks und Competitions etabliert. Die generelle Reproduzierbarkeit von Ergebnissen aus der Publikation allein lässt in vielen Fällen einige Wünsche offen.[CP16]

2 Probleme

Neben dem generellen Problem der Verfügbarkeit der Details zur Reproduktion von Ergebnissen, stellt sich die Frage der Vergleichbarkeit. Software Engineering beschäftigt sich mit den Erstellungsprozessen für und dem Lebenszyklus von Software. Um verschiedene Lösungswege in den Prozessen sinnvoll vergleichen zu können, müsste man auf den gleichen „Eingabedaten“ aufsetzen können. Das ist einfach für Code-Analyse-Werkzeuge, weniger für den Vergleich von verschiedenen Umsetzungs- oder Verifikationsmethoden, die eigentlich

[1] AIT Austrian Institute of Technology GmbH, Center for Digital Safety and Security, Giefinggasse 4, 1210 Wien, Österreich, rupert.schlick@ait.ac.at

doi:10.18420/SE2020_49

eine detaillierte Spezifikation oder zumindest detaillierte Anforderungen als Ausgangspunkt benötigen.

Für den in bestimmten Disziplinen allgegenwärtigen Getränkeautomaten als Beispiel zur Erklärung eines Prinzips mag es noch möglich sein, ausreichend Details aus einem Konferenzartikel zu ziehen - für realistisch große Beispiele, die auch die Skalierbarkeit eines Ansatzes demonstrieren, ist das nicht möglich.

3 Lösungsansatz

Zur Diskussion steht die Idee, die Verfügbarkeit von gut dokumentierten Beispielanwendungen so zu erhöhen, dass für Publikationen weniger Aufwand für die Aufbereitung der Beispiele erforderlich ist. Gleichzeitig soll es Anreize und Unterstützung geben, auch die experimentellen Ergebnisse und die verwendeten Mittel so zu publizieren, dass ein Vergleich von Ergebnissen einfacher, im Optimalfall sogar automatisierbar wird.

Mittel dazu sind Vorgaben durch Konferenzen und Journale, aber auch Doktormütter und -väter. Erforderliche Hilfsmittel sind Klassifikationskataloge für Beispielanwendungen, Methoden, Werkzeuge und Experimente sowie Richtlinien für die Erfassung von Beispielen und Experimenten und optimalerweise ein zentrales Archiv[Sc18].

Die erhöhte Vergleichbarkeit würde, erhoffter Weise, sowohl die dokumentierte Verbesserung von Methoden und Verfahren als auch generell vergleichende Publikationen und Metastudien erleichtern. Die erhöhte Zugänglichkeit und Vergleichbarkeit könnte auch der Übernahme von Forschungsergebnissen in die industrielle Praxis Vorschub leisten.

Danksagung Die hier berichteten Beobachtungen und Lösungsansätze entstanden im Förderprojekt *IoT4CPS* mit Unterstützung durch Mittel aus dem Förderprogramm *IKT der Zukunft* von FFG und BMVIT in Österreich.

Literatur

[CP16] Collberg, C.; Proebsting, T. A.: Repeatability in Computer Systems Research. Commun. ACM 59/3, S. 62–69, Feb. 2016, ISSN: 0001-0782, URL: http://doi.acm.org/10.1145/2812803.

[Sc18] Schlick, R.; Felderer, M.; Majzik, I.; Nardone, R.; Raschke, A.; Snook, C.; Vittorini, V.: A Proposal of an Example and Experiments Repository to Foster Industrial Adoption of Formal Methods. In (Margaria, T.; Steffen, B., Hrsg.): Leveraging Applications of Formal Methods, Verification and Validation. Industrial Practice. Springer International Publishing, Cham, S. 249–272, 2018, ISBN: 978-3-030-03427-6.

Software Engineering: Forschung — Innovation — Transfer

Software Engineering: Forschung – Innovation – Transfer

Stefan Sauer[1], Rudolf Ramler[2]

1 Motivation und Zielsetzung

Wie schon der Name Software Engineering vermittelt, lebt die Disziplin von der ingenieursmäßigen Erforschung, Entwicklung und Anwendung von Methoden sowie dem regen Austausch von Wissen und Technologien zwischen Wissenschaft und Praxis. So werden die Relevanz und Anwendbarkeit von Forschungsergebnissen zum beiderseitigen Nutzen gefördert: Neueste Ergebnisse und Erkenntnisse aus der Wissenschaft werden in die Wirtschaft transferiert, um dort Innovationen hervorzurufen. Gleichzeitig gewinnt die Forschung ein besseres Verständnis der Herausforderungen und Randbedingungen beim praktischen Einsatz neuer Technologien und Methoden, die auf diese Weise zudem im realen Anwendungskontext validiert werden können.

Es gibt die unterschiedlichsten Formen der Zusammenarbeit von Unternehmen mit Hochschulen und Forschungseinrichtungen, um gemeinsam Forschung und Entwicklung (F&E) sowie Wissenschaftstransfer zu betreiben. An einigen Hochschulen existieren dafür dedizierte Transferinstitute, die sich der kooperativen und anwendungsorientierten Forschung und dem Wissens- und Technologietransfer zu Themen des Software Engineerings verschrieben haben. Außerdem widmen sich institutsübergreifende Forschungsorganisationen auf nationaler und internationaler Ebene dieser Aufgabe. In der Tradition vergangener SE-Konferenzen bietet die SE 20 den verschiedenen Akteuren aus Praxis und Wissenschaft ein spezielles Programm für den Austausch untereinander – den *Forschungs- und Technologietransfertrack SE FIT 20*. Ziel ist es, Vertreter aus Praxis und Wissenschaft zum Erfahrungsaustausch über Wissens- und Technologietransfer im Software Engineering zusammenzubringen. Der Fokus liegt auf der Diskussion der unterschiedlichen Formen der Forschungskooperation und des Wissenschaftstransfers sowie deren Vor- und Nachteile: Best Practices für die strategische und projektorientierte Zusammenarbeit zwischen Firmen und Forschungseinrichtungen, Transferinstrumente, Co-Working-Modelle, zunehmend aber auch die Kooperation zwischen Forschungseinrichtungen, öffentlicher Hand und gesellschaftlichen Akteuren.

[1] Universität Paderborn, SICP – Software Innovation Campus Paderborn, Fürstenallee 11, 33102 Paderborn, Germany, sauer@sicp.upb.de
[2] Software Competence Center Hagenberg GmbH, Softwarepark 21, 4232 Hagenberg, Austria, rudolf.ramler@scch.at

doi:10.18420/SE2020_50

2 Format und Programm

Praktiker(innen) und Wissenschaftler(innen), die in der praxisorientierten Forschung, dem Wissens- und Technologietransfer im Bereich Software Engineering tätig sind, waren eingeladen, Beiträge einzureichen. Einreichungen für den Transfer-Track SE FIT 20 konnten in zwei Arten erfolgen:

- *Ergebnis-/Erfahrungsberichte*: Forschungstransfer-, Praxis- und Kooperationsberichte sowie hintergründige Profile von Transferinstituten
- *Positionspapiere*: Kurzprofile von Transferinstituten, Transfermodellen und Transferinstrumenten sowie „Wunschzettel“ zu wechselseitigen Erwartungen an einen erfolgreichen Wissens- und Technologietransfer.

Insgesamt wurden 14 Beiträge eingereicht und nach Begutachtung zur Präsentation und Veröffentlichung eingeladen; neun in der Kategorie Ergebnis-/Erfahrungsberichte und fünf Positionspapiere, die Transferinstitute, -modelle und -instrumente vorstellen.

Im Programm des Transfer-Tracks SE FIT 20 wurden die verschiedenen Formate kombiniert. Eingeleitet wurde der Track durch einen *Keynote-Vortrag* von Stefan Wallner, Siemens AG Österreich, Corporate Technology (CT), zum Thema „Success Factors for Industry-Academia Collaboration: Lessons Learned“, der eine Retrospektive der erfolgreichen Forschungskooperation mit dem Christian Doppler Labor „Monitoring and Evolution of Very-Large-Scale Software Systems“ an der Johannes-Kepler-Universität Linz lieferte und auf aktuelle industrielle Forschungsthemen bei Siemens CT einging. Die *Ergebnis-/Erfahrungsberichte* zeichnen ein umfassendes Bild der Herausforderungen und Lösungen in der Kooperation von Hochschulinstituten und Forschungseinrichtungen mit Firmen aus der Industrie und anderen Wirtschaftsbranchen. Das Themenspektrum der vorgestellten Beiträge reicht von Erfahrungen aus der projektorientierten Zusammenarbeit, über verschiedene Kooperationsmodelle, bis zur Diskussion von Organisationsformen für den Forschungs- und Technologietransfer. Die Positionspapiere wurden in Form von *Impulsvorträgen* vorgestellt. Ergänzt wurden die Vorträge um einen *Open Space* zum Kennenlernen sowie als Raum für Diskussionen und interaktiven Austausch, kombiniert mit einem *Poster Space* zur vertieften Darstellung der in den Vorträgen präsentierten Inhalte.

3 Danksagung

Als Co-Chairs des SE FIT 20 danken wir allen Akteurinnen und Akteuren für ihre engagierte Mitwirkung – insbesondere den Autorinnen und Autoren, den Vortragenden, und allen Teilnehmerinnen und Teilnehmern – für die gewährten Einblicke, die präsentierten Ergebnisse, die geteilten Erkenntnisse und die geführten Diskussionen zum Thema Wissens- und Technologietransfer. Unser Dank gilt auch dem Organisationsteam der SE 20, die ein angenehmes und reibungsloses Umfeld für den Track geschaffen haben.

Integrating Microservices from an Ongoing Research Project: A Technology Transfer Experience Report

Andrea Mussmann[1] Michael Brunner[1 2]

Abstract: QE LaB Business Services integrated features focusing on requirements engineering offered by OpenReq, an EU Horizon 2020 project, into an existing commercial software for the management of security requirements. The microservices were integrated as part of a dedicated Open Call project where participants from industry were asked to integrate novel features from OpenReq into their own tools and evaluate them. In this paper, we discuss our experiences with the Open Call format of the project as well as the issues we encountered and lessons we learned following scientifically acknowledge success factors.

Keywords: Technology Transfer; Experience Report; Requirement Engineering; Information Security Management.

1 Introduction

This paper is an experience report of integrating microservices offered by the EU Horizon 2020 project OpenReq (https://openreq.eu) in ADAMANT (https://adamant.work) [Br18b], an Information Security Management tool of QE LaB Business Services. OpenReq is primarily focused on the scientific development of requirements engineering techniques (e.g. finding similar requirements, measuring requirement text quality). While ADAMANT is a tool for the domain of information security management, a key part of it is concerned with the documentation of information security requirements, and thus shares common interests with the requirements engineering domain. Especially features concerning the analysis and quality assurance of natural language requirements were of great relevance to our work and well-aligned with the development roadmap of ADAMANT. QE LaB Business Services successfully applied for the corresponding Open Call with a proposal to integrate the project's microservices into ADAMANT to enhance the collaborative documentation of natural language information security requirements. The Open Call was a promising opportunity to further boost the development of additional quality assurance features that might otherwise have been postponed to a later point in time.

In this paper, we present our experiences as industrial partner in a technology transfer project. In Section 2, we briefly introduce the EU Horizon 2020 project OpenReq, ADAMANT,

[1] QE LaB Business Services GmbH, Technikerstraße 21a, 6020 Innsbruck, Austria firstname.lastname@qe-lab.com
[2] University of Innsbruck, Department of Computer Science, Technikerstraße 21, 6020 Innsbruck, Austria michael.brunner@uibk.ac.at

doi:10.18420/SE2020_51

and the goals of the Open Call. In Section 3, we discuss the technical integration of novel features from the academic setting via provided microservices. In Section 4, we reflect on our project in light of scientific findings on success factors and issues in technology transfer by Brings et al. [Br18a]. We close the paper with a summary of our experiences and lessons learned as industrial partner in a research transfer project.

2 Background

The Open Call project was organised by the EU Horizon 2020 project OpenReq and constituted the academic side of this technology transfer. The project was multinational with members from five countries (Austria, Finland, Germany, Italy, and Spain). The overarching goal was to improve collaborative software engineering by developing algorithms and tools for community-driven requirements engineering and related fields. The project offered different microservices intended to support various requirements engineering tasks and activities. The microservices were open source and could be accessed directly via a platform provided by the project consortium or downloaded from the official OpenReq GitHub repository.

ADAMANT is a multi-user web application developed by QE LaB Business Services that supports companies to introduce and operate an information security management system. Based on a system model of the company (an extended asset catalogue), users define security requirements and controls to reduce identified information security risks to ultimately ensure compliance with the organisation's information security goals. The tool is designed to integrate stakeholders of different domains into the process of information security management. This ensures that a good proportion of departments of a company are directly involved in the process of deriving security requirements for the company.

However, different people from different departments focus on different aspects of information security and potentially use very different language to express what security requirements need to be fulfilled. This can lead to overly complex security requirement definitions or redundant requirements in the system. It can also result in security requirements that depend on or conflict with each other. Therefore, the primary intention behind the integration of services offered by OpenReq was to reduce these sources of errors and to improve existing quality assurance features with novel approaches from academic research.

The technology transfer happened as part of a dedicated Open Call organized and funded by the OpenReq research project, a separate call launched within the project duration to support the integration, extension, and evaluation of developed microservices in existing open source and commercial software. The call aimed at testing the capabilities of the research project's software deliverables. In addition, the call demanded the evaluation of the integrated microservices in real-world settings with data sets and participants from industry.

During the Open Call project, selected microservices were integrated into ADAMANT. Additionally, we conducted a study to capture practitioners' perceptions of developed

features in ADAMANT that were supported by these microservices. Our project ran from March 2019 until September 2019 and involved one part-time software developer and a project manager.

3 Microservice Integration

In the Open Call project proposal, we envisioned the integration of four different features. Table 1 briefly lists these features and their final implementation status in ADAMANT.

Microservice	Implementation Status
Quality check for requirements text	*Direct call of publicly available microservices via REST API.* Quality issues are shown in a separate QA dialog in ADAMANT's security requirement detail and edit view.
Informativeness of requirement text	*Integration of self-hosted microservice via REST API. Underlying classifier had to be trained using a custom-built coprus (that had to be constructed additionally).* Quality issues shown in a separate QA dialog in ADAMANT's security requirement detail and edit view.
Measuring text similarity between requirements	*Integration of self-hosted microservice via REST API. Service integration required additional steps to achieve reliable results in the domain of security management.* Similar requirements are shown in dedicated view with potential follow-up actions.
Requirements dependencies, conflicts resolution and release alternatives	*Not implemented, required ontology could not be produced within budgetary constraints of the Open Call project.*

Tab. 1: Integrated microservices from the research project

While all offered microservices could generally be interfaced easily via their REST APIs, certain complicating factors surfaced during the technology transfer project. By shifting resources and substituting the planned feature for dependency detection with the introduction of a novel multi-dimensional quality rating for security requirements, the project goals could still be met. The Open Call project was completed successfully in September 2019 and the the results were presented at an event organised by the OpenReq project consortium to an audience consisting of experts from academia and industry.

4 Technology Transfer Success Factors

Successful technology transfer depends on several factors, both on the level of the technology to be transferred as well as the development and the transfer process. In a literature study, Brings et al. [Br18a] identified the most important factors for a successful technology transfer in software engineering. Of these success factors, the following are relevant to the Open Call project presented in this article.

Technology: Regarding the technology itself, technology transfer in software engineering is successful if (T1) the technology to be transferred can be adapted, has been (T2) developed in collaboration with practitioners and has been (T3) well evaluated prior to the transfer. Furthermore, the technology to be transferred has (T4) to adequately address industry needs.

Transfer Process: Regarding the technology transfer process, communication is key. There has to be an (P1) adequate information flow between the transfer partners. During the introduction of the software in the company, (P2) researchers should be present. Especially critical for the success of technology transfer is an (P3) awareness for different goals and expectations in research and industry. If the mismatches are not recognized, so Brings et al., "the technology transfer project seems very likely to be condemned" [Br18a].

In this section, we reflect upon our experiences with the Open Call project format according to these success factors. This section both highlights what went well during the six months of the Open Call project and where we had to contend with problems and issues.

4.1 Technology (T1, T2, T3, T4)

The overarching goal of the research project was to improve collaborative requirements engineering approaches and practices. To reach this goal, academic institutions and industry were closely cooperating to develop innovative solutions to concrete real-world problems from industrial practice. The project thus clearly fulfills success factors T2 and T4.

Overall, integrating the services into ADAMANT worked well. However, some of the microservices were not yet mature enough to ensure a smooth technology transfer – a fact that had been clearly and transparently noted in the call for proposals. The fact that both the requirements classifier and the similarity detection service were still under development and changed their APIs during the duration of the project made several changes to the ADAMANT integration of the services necessary. T3 was thus not fully satisfied for some microservices interfaced during this technology transfer project.

What we perceived as the most complicating factor of the transfer process were the unreliable and sometimes unresponsive microservice instances hosted by the project consortium. This made it necessary to install the needed microservices locally to ensure a stable environment for development and subsequent integration of the microservices. While some of the microservices were easy to set up, others had complicated setup and installation instructions that usually required undocumented additional steps. Consequently, the installation process had to be debugged in some cases, which cost time and made the microservices less appealing to use. In this case, success factors T3 and P1 were unsatisfactory in some areas (e.g. missing or outdated deployment documentation).

In general, the developed microservices adapted well to the ADAMANT use cases and data from information security management (success factor T1). However, some issues that seem not to surface with other industrial partners were problematic in the ADAMANT context.

The most seriuous issue we had to deal with occurred during the integration of the similarity detection microservice. Similar requirement texts occur regularly in ADAMANT since the same security requirements and controls may be applied to protect similar assets. Similar requirement texts are thus only an issue if they appear in requirements protecting the same asset. There usually are not that many requirements attached to one asset, however, the offered microservice for similarity detection could not detect identical requirements (nor very similar requirements) if there were only a few requirements to compare. This was not well communicated and cost us some time to figure out and subsequently deal with. Here, both T1 and P1 were not ideal making the technology transfer more difficult.

4.2 Transfer Process (P1, P2)

During the Open Call project, participants were in regular contact with their project coordinator and with people from the development community as needed to allow for a smooth progress. E-mail and Google Hangouts were the main modes of communication. The platform GitHub was used to discuss issues directly related to specific microservices. While the communication with the project coordinator via Google Hangouts and e-mail worked very well with little to no delays, more detailed technical questions had to be posted to the developers directly. Because the microservices were still in active development slight communication delays were inevitable. As a result, P1 was not always satisfactory from our perspective as industry partners. Consequently, P2 could not always be guaranteed.

Similarly, the microservices would have benefitted from a more detailed documentation and a more verbose error code scheme (lack of satisfactory P1). Ultimately, we debugged some of the microservices ourselves to analyse unexpected behaviour and to resolve specific errors.

4.3 Mismatch of Perceptions (P3)

In the initial proposal for the Open Call project, we had planned to integrate the microservice for requirements dependencies, conflicts resolution and release alternatives into ADAMANT. The microservice was advertised to detect dependencies between sets of requirements based on their natural language requirements text. To run the microservice, an ontology that defines relations between terminology used within the domain of the requirements has to be provided for the dependency detection to work. This might be a good solution in terms of research, however, it either presupposes the existence of a domain ontology at the company integrating the service or demands the creation of such an ontology. This is, on the one hand very time consuming, on the other hand it might not even be possible to create such an ontology. Even though ADAMANT is designed as an information security management tool, this does not mean that all requirements in ADAMANT have to be part of the information security domain, or even that ADAMANT has to be used exclusively for

information security management. There cannot be a generic ontology that still allows for dependency detection between requirement texts. Here, an initial mismatch of expectations at the time of writing the proposal made the technology transfer very difficult and meant that we could not use the microservice in question and stay within budgetary constraints.

The feature used to compute sentence-wise informativeness of a requirement was listed in the communicated feature list of the Open Call as a service that automatically identifies sentences of a requirement that are not informative. We chose to integrate the service into ADAMANT assuming that the service would be ready to use. However, there was no informativeness service present, only a requirements classifier (see Section 3) that could be trained to, for example, classify sentences according to their informativeness. While this may be a reasonable task in academia, we had not budgeted for the thorough training of a classifier to perform a task we had assumed would be readily provided by a microservice. This mismatch of expectations additionally strained the technology transfer process.

5 Conclusion

In conclusion, the OpenReq Open Call was a success and a valuable experience that allowed us to integrate cutting-edge technology from research into our own software tool. Of the issues we had to contend with, most resulted from the ongoing development of the microservices and were thus to some degree expected. From our own experience, we can state the following lessons learned regarding technology transfer between research and industry: Good documentation is key, technology should be usable out-of-the-box, any prerequisites for using technology should be clearly communicated to enable informed feature decisions, and on-premise installation of required software components should be well supported.

Acknowledgement

We would like to acknowledge the Horizon 2020 project OpenReq, which is supported by the European Union under the Grant Nr. 732463.

References

[Br18a] Brings, J. et al.: 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.

[Br18b] Brunner, M. et al.: Introduction of a tool-based continuous Information Security Management System: An exploratory case study. In: IEEE International Conference on Software Quality, Reliability and Security Companion (QRS-C). 2018.

Fostering Collaboration of Academia and Industry by Open Source Software

David Baum[1], Pascal Kovacs[2], Richard Müller[3]

Abstract: In 2017 and 2018 we released two of our research prototypes as open source. We explain our motivation and concerns at that time and compare them with our actual experience. We also describe how open source releases enabled collaboration with industrial partners. Finally, we show how research projects can extend their funding through grants for open source software. We share our experiences with the initiative Google Summer of Code and show how we overcame bureaucratical hurdles and how our research has benefited from participating in this program.

Keywords: Open Source; Industry Collaboration; Technology Transfer

1 Introduction

For 12 years our research group *Visual Software Analytics* at Leipzig University has been combining findings and methods from the fields of software analytics, software visualization, data science, and empirical software engineering to extract, visualize, and analyze software-related data. The majority of our work are basic research and empirical evaluation which are usually not of direct interest to practitioners. However, we also develop software prototypes which form the basis of our empirical evaluations. The research project is financed only to a very small extent by third-party funds, but is driven primarily by budget funds and student participation. Therefore, the team consists of only four researchers working part-time for the project, two of them being university employees. The research is supported to some extent by the Institute for Applied Informatics (InfAI), which is an affiliated institute of Leipzig University to promote science and research in the areas of computer science and information systems. Within the research group, we discussed for a long time whether an open source release would be useful for us. Open soure releases have a strong tradition in computer science. Nevertheless, the benefits for basic research projects are not always obvious. For this reason, we would like to share our experiences with systems that we have published as open source. We will briefly present *Getaviz* and *jQAssistant Dashboard* and discuss

[1] Leipzig University, Information Systems Institute, Grimmaische Straße, 04109 Leipzig, Germany david.baum@uni-leipzig.de
[2] GISA GmbH, Consulting, Leipziger Chaussee 191A, 06112 Halle (Saale), Germany pascal.kovacs@gisa.de
[3] Leipzig University, Information Systems Institute, Grimmaische Straße, 04109 Leipzig, Germany rmueller@wifa.uni-leipzig.de

doi:10.18420/SE2020_52

advantages, disadvantages, and recommendations. Afterwards we will discuss opportunities for funding open source software development beyond conventional third-party funding.

2 Getaviz

With Getaviz[4] users can solve software engineering problems visually by exploring software artifacts. Getaviz visualizes the structure and runtime behavior of several programming languages, i.e, Java, Ruby, C#, JavaScript, and ABAP. It can enrich these visualizations with evolutionary information from git- and svn-repositories. Among other things it provides proper visualizations and an interactive user interface for identifying and refactoring architectural antipatterns, locating runtime bottlenecks, assessing software quality, and tracking code changes across multiple versions. Getaviz provides a variety of two- and three-dimensional visualizations, that can be explored via browser, HTC Vive, or Microsoft HoloLens. You can find a selection of showcases on our website[5]. Getaviz includes an evaluation server. Its main objective is to conduct empirical evaluations locally and remotely in an efficient and reproducible way. The complete codebase has about 200.000 lines of code in JavaScript, Java, and Ruby.

We did not develop Getaviz for a one-time research project. Instead, we planned from the beginning to create a long-living system that is developed continuously and gets reused across different research projects. Getaviz has been developed as closed source for about ten years from 2007 – 2017. At this time it was used only internally and there was no collaboration with industry so far, i.e., companies did not took part in the development of Getaviz and they did not use Getaviz in practice. Occasionally, Getaviz was made available to students, so that they could use it and extend it as part of their studies. Triggered by the publication of our tool at the most important conference in our research field [Ba17], we decided to release Getaviz as open source in this year. Doing this we pursued the following objectives:

1. Improve visibility of Getaviz and of our work in general within the scientific community
2. Simplify future collaboration with industry and other research groups
3. Facilitate reproducibility of empirical studies

However, there were also concerns about increased maintenance efforts, too many feature requests, and outflow of ideas. We assumed that it is hard for projects like these to build a large and lively community. So we did not expect any unpaid external contributors, i.e., people who contribute to open source in their leisure time since people prefer to contribute to software they use by themselves. We were also aware that Getaviz is a research prototype

[4] https://github.com/softvis-research/Getaviz
[5] http://home.uni-leipzig.de/svis/showcases/

and hence not ready for production and it will not find suddenly hundreds or thousands of users. Nevertheless, we decided to release Getaviz under Apache License 2.0. We also discussed more restrictive and viral licenses such as GNU General Public License 3 (GPL3) and even GNU Affero General Public License to ensure that modifications to the source code will be published under the original license. However, companies often shy away from this type of license. That is why we decided to use the more liberal Apache License 2.0 which is not viral, i.e., anyone can modify the source code without making the changes available again as open source.

Until its release, we had to overcome some obstacles. Since many contributions came from students as part of their theses, it was not clearly regulated how to publish this code. Therefore we had to contact all students being involved formerly and ask for their permission, which was a lot of work. Next, we had to review all used third-party libraries for their licenses. As a consequence, we had to replace some libraries due to license incompatibility, especially between Apache License 2.0 and GPL3. For this reason we have discarded the version history and started with a new and clean repository. Finally, we released Getaviz 1.0 in November 2017 under Apache License 2.0.

In 2019 GISA GmbH[6] and Visual Software Analytics started a joint initiative named "Visual SAP Analytic Process" (VISAP) based on Getaviz. GISA is an IT full service provider with focus on SAP systems for energy market, contracting authorities, and industrial as well as service companies. The goal of VISAP is to improve quality assessment of customer specific SAP code and to support the migration of custom code to the new SAP S/4HANA platform by using software visualizations. As part of VISAP, Getaviz is used to visualize the custom code written in ABAP, the programming language of SAP systems, which is extracted by an own tool for custom code life cycle management. Currently, GISA employs four people who work part-time to develop VISAP and have already provided many new features for Getaviz that will benefit everyone involved. These include numerous feature improvements as well as improvements in robustness and maintainability, with only a few functions of exclusive interest to GISA. Although the license used, Apache License 2.0, is not viral and GISA is not legally obliged to play back its changes on Getaviz, they do. From GISA's point of view the open source nature of Getaviz is an advantage, because there are no possible lock-in effects and a later usage as a commercial product is also possible.

Since Getaviz is a self-contained system, it is easy to deploy and provide running instances for reviewers and other researchers. We have observed further positive side effects. Previously, the number of installations was very limited and it was possible for us to monitor each installation process personally and provide useful tips. Since this approach does not work for users outside the research group, we had to revise the installation process as well as to improve documentation. Therefore, the quality of our documentation and setup routine has improved significantly since we released Getaviz as open source. The same applies to the code itself. Getaviz has become more robust since it is used on more computers with different

[6] http://www.gisa.de

setups. We also got contributions from external users, but - so far - limited to a manageable extent. In most cases these contributions have been bug reports and improvements of the documentation. We also noticed an increased interest from students in contributing to Getaviz as part of their studies, since they prefer to write code that is actually used.

3 jQAssistant Dashboard

Since 2017, our research group collaborates with BUSCHMAIS GbR[7], an IT consulting company with focus on application integration, deployment, scalability, and persistence. The company has developed *jQAssistant*[8], a quality assurance tool that is based on Neo4j[9]. It scans software artifacts' data, for example source code or test results, stores them as graphs in a Neo4j database, and provides means to analyze the graph data. jQAssistant was released in 2015 under GPL3 and has since formed a significant community.

Through this collaboration we have mutually benefited from each other. On the one hand, BUSCHMAIS was looking for a way to visualize the software data extracted with jQAssistant. On the other hand, our research group needed a unified data source for the visual analytics tools.

Hence, we have developed a dashboard on top of jQAssistant together with a master student. The *jQAssistant Dashboard*[10] supports project leaders and software architects in decision-making. It provides interactive views concerning architecture and dependencies as well as resource, risk, and quality management. Due to our experience with Getaviz, we developed the project as open source from the beginning. Therefore, we had not to consider subsequent license agreements and incompatible libraries and the whole process was much smoother. This successful collaboration resulted in a joint publication at VISSOFT 2018 [Mü18]. Even one year after finishing his master thesis, the student is still maintaining the project. The dashboard will be integrated into regular jQAssistant releases in the near future.

Furthermore, we are developing scanner plugins for jQAssistant licensed under GPL3 to support additional programming languages or data sources. Three plugins are particularly worth mentioning here. The first plugin[11] scans Java source code and was used to solve a feature location challenge at SPLC 2019 [ME19]. The second plugin[12] scans software traces to support application performance monitoring and architecture discovery. It was developed by a student as a seminar project and published at SSP 2019 [MF19]. The third plugin[13] scans data from Jira[14]. It was also developed by a student during Google Summer

[7] https://www.buschmais.de
[8] https://jqassistant.org
[9] https://neo4j.com
[10] https://github.com/softvis-research/jqa-dashboard
[11] https://github.com/softvis-research/jqa-javasrc-plugin
[12] https://github.com/softvis-research/jqa-kieker-plugin
[13] https://github.com/softvis-research/jqa-jira-plugin
[14] https://www.atlassian.com/de/software/jira

of Code (GSoC) 2019. As the jQAssistant Dashboard and the scanner plugins[15] are all open source, we get feature requests, bug reports, and contributions from academia as well as industry and our research can be easily replicated.

4 Open Source Funding

Another advantage of releasing research prototypes as open source is that additional funding sources can be used which exclusively address open source projects, be they scientific or not. Initiatives like GSoC[16], Google Season of Docs (GSoD)[17], and Outreachy[18] support students who would like to start contributing to open source projects. Especially for projects with limited budgets these initiatives are really attractive and a helpful complement to basic funding. However, the application process is different from conventional third-party funding and in our experience the organizational structure of universities is not designed for that. This is because of legal issues, e.g., transfer of rights, but also because of inexperience of the judicial department with US laws in general, which made it impossible for us to participate as Leipzig University in GSoC. To overcome these bureaucratic hurdles we applied for GSoC and GSoD under the umbrella of the InfAI and got accepted for GSoC in 2019. This means that we were able to award paid scholarships to students to further develop our software. Many students from all over the world were interested in our scholarships and applied for it. This is remarkable for such a small project given that our competitors were well-known open source projects with large communities, such as Linux Foundation, KDE, Mozilla, and many others. In our estimation, it is because Getaviz consists of a broad and modern technological basis and the fact that people find software visualizations in general fascinating. We have awarded two scholarships to international master students. Both worked independently for several months on a project based on Getaviz or jQAssistant. One student successfully completed his project and continued to contribute code afterwards. However, studies show that most students take part in GSoC for work experience, career building, and the stipends. Only a minority of the participants will actually become active contributors afterwards [Si19]. Therefore, GSoC only contributes to the building of an active and lively community to a limited extent. But even then open source funding and the contributions so far are very valuable. Taking part in GSoC helped us to increase visibility, to improve our documentation, and to increase the number of external contributions. And last but not least, students also benefit from participating in such programs.

5 Conclusions

For our research group it has been very beneficial to release our software as open source and we regret to have not done this earlier. Releasing our software as open source helped

[15] https://softvis-research.github.io/jqassistant-plugins
[16] https://summerofcode.withgoogle.com
[17] https://developers.google.com/season-of-docs
[18] https://www.outreachy.org

us to increase our visiblity, to provide reproducible software artifacts, and to collaborate with industry. We were even able to expand our funding. Noteworthy, we were not able to identify any negative effects so far.

Nevertheless, we have noticed that the process has its pitfalls and that many decisions have to be taken. Therefore we would like to give the following recommendations.

1. Consider open source release as early as possible in the research process.
2. External transfer institutes can be helpful to overcome limitations of organizational structures of universities and provide additional funding options.
3. Expect additional effort for documentation, code reviews, and support.
4. Limit your expectations with regards to community building.

We want to emphasize that these recommendations are based on our experience given our specific scenario and should be applied to other projects with care.

Acknowledgments

We would like to thank all contributors to our open source systems, especially Stefan Bechert, Matteo Fischer, Tino Mewes, Christina Sixtus, and Lisa Vogelsberg for their valuable and significant contributions. We also thank Ulrich Eisenecker for his comments that greatly improved the manuscript.

References

[Ba17] Baum, David; Schilbach, Jan; Kovacs, Pascal; Eisenecker, Ulrich; Müller, Richard: GETAVIZ: Generating Structural, Behavioral, and Evolutionary Views of Software Systems for Empirical Evaluation. In: IEEE VISSOFT. 2017.

[ME19] Müller, Richard; Eisenecker, Ulrich: A Graph-based Feature Location Approach Using Set Theory: [Challenge Solution]. In: Proc. 23rd Int. Syst. Softw. Prod. Line Conf. - Vol. A. SPLC '19, ACM, New York, NY, USA, pp. 161–165, 2019.

[MF19] Müller, Richard; Fischer, Matteo: Graph-Based Analysis and Visualization of Software Traces. In: 10th Symp. Softw. Perform. Jt. Dev. Community Meet. Descartes/Kieker/Palladio. Würzburg, Germany, 2019.

[Mü18] Müller, Richard; Mahler, Dirk; Hunger, Michael; Nerche, Jens; Harrer, Markus: Towards an Open Source Stack to Create a Unified Data Source for Software Analysis and Visualization. In: Proc. 6th IEEE Work. Conf. Softw. Vis. IEEE, Madrid, Spain, 2018.

[Si19] Silva, Jefferson O; Wiese, Igor; German, Daniel M; Treude, Christoph; Gerosa, Marco A; Steinmacher, Igor: Google Summer of Code: Student Motivations and Contributions. 2019.

Wie aus dem s-lab – Software Quality Lab der SICP – Software Innovation Campus Paderborn wurde

Von der Kunst, einen Forschungscampus zu bauen

Stefan Sauer[1]

Abstract: Das s-lab – Software Quality Lab wurde im Jahr 2005 von fünf Informatik-Professoren der Universität Paderborn als Private-Public-Partnership gegründet, um ihre Aktivitäten im Bereich der Industriekooperation zu Themen der Softwareentwicklung und Softwarequalitätssicherung in einer wissenschaftlichen Einrichtung zusammenzuführen. Trotz seiner strategischen Positionierung ist das s-lab keine 15 Jahre später Geschichte. Es hat Platz gemacht für den SICP – Software Innovation Campus Paderborn und ist darin aufgegangen. Wie dieser Prozess vonstattengegangen ist, welche Ziele dahinterstecken, aber auch welche Chancen und Herausforderungen damit verbunden waren und sind, soll in diesem Beitrag beleuchtet werden. Zudem werden die organisatorischen Veränderungen, neuen Strukturen und Prozesse beschrieben, die die Erforschung und Entwicklung von software- und datengetriebenen Innovationen voranbringen sollen.

Keywords: Wissenstransfer, Technologietransfer, Forschungscampus, transdisziplinäre Forschung, transorganisationale Kollaboration, trianguläres Innovationssystem, software- und datengetriebene Innovationen

1 Einleitung

In einer wissenschaftlich-technischen Disziplin wie der Softwaretechnik – im englischen Sprachgebrauch als Software Engineering bezeichnet, wodurch der ingenieursmäßige Charakter der Informatik-Teilwissenschaft noch deutlicher zum Vorschein kommt – spielt die enge Verzahnung von Wissenschaft und Technik, Theorie und Praxis von jeher eine wesentliche Rolle, um einerseits wissenschaftliche Erkenntnisse schnell und effektiv in die praktische Umsetzung zu führen und um andererseits wissenschaftlich fundierte Antworten auf die Herausforderungen der Praxis zu liefern. Vor diesem Hintergrund wurde an der Universität Paderborn im Jahr 2005 das s-lab – Software Quality Lab von fünf Informatik-Professoren gegründet. Das s-lab wurde als In-Institut der Universität mit mehreren Unternehmen der privaten Wirtschaft als externen Partnern konzipiert, die einen Kooperationsvertrag für das s-lab schlossen. In dieser Multi-Private-Public-Partnership sollten fortan die Aktivitäten der beteiligten Fachgebiete der Informatik und Unternehmen im Bereich der kollaborativen Forschung und des Wissenschaftstransfers zu Themen der

[1] Universität Paderborn, SICP – Software Innovation Campus Paderborn, Germany sauer@sicp.upb.de

doi:10.18420/SE2020_53

Softwareentwicklung und Softwarequalitätssicherung gebündelt werden. Sowohl die Zahl der Hochschullehrer als auch der Unternehmen wuchs über die Jahre stetig an.

Trotz seiner strategischen Positionierung ist das s-lab keine 15 Jahre später Geschichte. Es hat Platz gemacht für den SICP – Software Innovation Campus Paderborn und ist darin aufgegangen. Hauptsächliche Ursachen hierfür waren einerseits der Bedarf der Unternehmen nach disziplinübergreifenden, ganzheitlichen Lösungsansätzen für ihre geschäftlichen Herausforderungen und softwarebasierten Lösungen und Produkte, die nach einer Hinzunahme weiterer fachlicher Expertise auf Seiten der Wissenschaft und der interdisziplinären Bearbeitung verlangten und andererseits die Notwendigkeit der Intensivierung der bilateralen Zusammenarbeit mit den Wissenschaftspartnern und der multilateralen Zusammenarbeit mit weiteren Partnern in Innovations- und Wertschöpfungsnetzwerken, um die komplexen Fragestellung auch lösen zu können. Ein wesentlicher Katalysator für den Transformationsprozess vom s-lab zum SICP war dann die im Jahr 2011 vom Bundesministerium für Bildung und Forschung (BMBF) aufgelegte Förderinitiative „Forschungscampus“.

Wie dieser Prozess vonstattengegangen ist, welche Ziele dahinterstecken, aber auch welche Chancen und Herausforderungen damit verbunden waren und sind, wird in den folgenden Kapiteln dargestellt. Zudem werden die organisatorischen Veränderungen, neuen Strukturen und Prozesse beschrieben, die die Erforschung und Entwicklung von software- und datengetriebenen Innovationen voranbringen sollen. Zunächst werfen wir aber einen kurzen Blick in die Vergangenheit und auf das s-lab als Keimzelle dieser Entwicklung.

2 Das war’s – das s-lab – Software Quality Lab

Das s-lab – Software Quality Lab der Universität Paderborn wurde im Jahr 2005 als wissenschaftliche Einrichtung der Fakultät für Elektrotechnik, Informatik und Mathematik gegründet, als Kompetenzzentrum der Universität Paderborn für kollaborative und anwendungsorientierte Forschung im Bereich Softwaretechnik.

Das s-lab verfolgte die Absicht, die Kooperationen mit Unternehmen im Bereich Softwaretechnik zu bündeln. Es führte die Kompetenzen der beteiligten Fachgebiete zusammen und war darauf angelegt, die resultierenden Synergien nutzen.

Konzipiert wurde das s-lab als ein offenes Multi-Private-Public-Partnership-Institut für Forschungskooperationen mit Unternehmen sowie den Wissens-, Kompetenz und Technologietransfer zwischen Industrie und Wissenschaft. Die offene Struktur des s-lab ermöglichte die Kooperation mit vielen Partnern.

Die Gründung des s-lab fußte dabei auf der zunehmenden und schlussendlich allgegenwärtigen Bedeutung von Softwarelösungen in Wirtschaft, Wissenschaft und Gesellschaft. Im Fokus stand dabei der Qualitätsanspruch an Software und ihre Herstellung. Denn Software sorgt dafür, dass Produkte und Abläufe einwandfrei funktionieren. Eine hohe Qualität der Software ist dabei eine unabdingbare Voraussetzung. Eine Software ist dann qualitativ

hochwertig, wenn sie eine Vielzahl verschiedener Merkmale erfüllt. Sie muss z.B. korrekt, zuverlässig, verständlich und benutzerfreundlich sein, stabil und effizient ablaufen und sicher sein gegenüber unbefugtem Zugriff. Aber auch Eigenschaften, die für die Erstellung und Weiterentwicklung der Software von Bedeutung sind, gehören dazu, z.B. Wartbarkeit, Wiederverwendbarkeit, Anpassbarkeit oder Portierbarkeit.

Diese Softwarequalität zu gewährleisten und zu messen ist eine große Herausforderung. Das s-lab – Software Quality Lab stellte sich genau dieser Aufgabe. Ziel war es, Unternehmen bei der Entwicklung qualitativ hochwertiger Softwareprodukte zu unterstützen. Zielgruppe waren gleichermaßen kleine, mittelständische und große Unternehmen, die anspruchsvolle Software entwickeln, z.B. für technische oder geschäftliche Anwendungsfelder in unterschiedlichen Branchen. Sie wurden dabei unterstützt, Konzepte, Methoden, Techniken, Sprachen und Werkzeuge zu entwickeln oder zu verbessern, um qualitativ hochwertige Software konstruktiv zu entwickeln und/oder die Qualitätseigenschaften der Software analytisch zu überprüfen. Dieses prozessübergreifende Qualitätsverständnis führte zu einem kontinuierlichen Qualitätsmanagement im Softwarelebenszyklus.

In den Projekten des s-lab wurden im Wesentlichen solche Fragestellungen bearbeitet, die einerseits eine hohe Relevanz für die industrielle Softwareentwicklung haben und andererseits eine wissenschaftliche Bearbeitung erfordern.

3 Die Evolution vom s-lab zum SICP

Einige grundlegende Ideen und Konzeptionsentscheidungen des s-lab wurden auch für den SICP aufrechterhalten: In-Institut, keine eigene Rechtsform, Forschungs- und Innovationsverbund auf Grundlage eines Kooperationsvertrags, Offenheit für neue Mitglieder. Auch der SICP ist als offener Forschungs- und Innovationsverbund (F&I) organisiert und basiert formal auf einem Kooperationsrahmenvertrag zwischen der Universität Paderborn und den weiteren Mitgliedern. Er ist keine eigene Rechtsperson. Innerhalb der Universität Paderborn wurde eine wissenschaftliche Einrichtung ins Leben gerufen, das Software Innovation Lab (kurz: SI-Lab), welches die Aktivitäten der Universität im SICP koordiniert und bündelt. Da es sich beim SICP anders als beim s-lab um eine fakultätsübergreifende Initiative handelt, ist das SI-Lab nun eine zentrale wissenschaftliche Einrichtung der Universität.

Neu ist vor allem dieser fakultätsübergreifende, transdisziplinäre (d.h. über wissenschaftliche und Fachdisziplinen hinweg reichende) und transorganisationale (d.h. über Instituts- und Unternehmensgrenzen hinweg gehende) Forschungs- und Innovationsgedanke. Am SICP beteiligen sich mehr als 30 Hochschullehrer(innen) aus der Informatik, Wirtschaftsinformatik, den Wirtschafts-, Ingenieur- und Kulturwissenschaften. Und neu ist natürlich das Forschungscampus-Modell: Unternehmensmitarbeiter(innen) und Wissenschaftler(innen) arbeiten gemeinsam an einem Ort, dem Campus. Unternehmen können ihre Mitarbeiter(innen) dauerhaft, d.h. in eigenen (i.d.R. exklusiv) angemieteten Räumen, Büros und Innovations-Labs auf dem Campus ansiedeln oder temporär, d.h. zeitlich befristet oder

zeitanteilig am Campus arbeiten lassen. Das Raumkonzept sieht hierfür gemeinsam genutzte Büroräume, Labore, Projekt-, Besprechungs- und Konferenzräume vor, aber auch offene Co-Working-Spaces, Multifunktionsflächen und Kommunikationszonen, die dem Austausch der Partner dienen und ebenso von involvierten Studierenden genutzt werden. Deshalb ist es explizit vorgesehen, keine größeren exklusiven Bereiche für einzelne Mitglieder (einzelne Unternehmen oder Fachgruppen der Universität) zu haben, sondern die Partner themenorientiert zu durchmischen, damit ein Höchstmaß an Kommunikation und Kollaboration gefördert wird.

Dabei baut die Kooperation hinsichtlich der Prozesse, Organisations- und Arbeitsformen auf der langjährigen Erfahrung in Forschungskooperationen und Wissenschaftstransfer der Industrie-Wissenschaft-Kooperation, der kollaborativen Forschung und dem Wissenschaftstransfer auf (rechnet man das ebenfalls eingebundene, noch auf Initiative von Heinz Nixdorf zurückgehende C-LAB ein, mehr als 30 Jahre): geeignete Kooperationsformen mit Unternehmen, Identifizieren von F&I-Herausforderungen, Abwicklung gemeinsamer F&I-Projekte, Entwickeln von Innovationen, Herstellen von Win-Win-Situationen. Aber auch dieses F&I-Ökosystem wurde weiterentwickelt, wir sprechen von einem triangulären Innovationssystem: Neben der Wissenschaft und den Technologieunternehmen, welche die software- und datengetriebenen Innovationen entwickeln, sind auch die Anwender – Anwenderunternehmen oder zunehmend andere gesellschaftliche Akteure – strategischer, kontinuierlicher und aktiver Mitspieler im SICP.

Neben den in der Motivation, den Zielen und der gewählten Implementierung liegenden Chancen des triangulären, transdisziplinären und transorganisationalen Forschungs- und Innovationscampus birgt das Modell auch einige Herausforderungen, die nicht zu unterschätzen sind: Zusammenführen verschiedener Fachkulturen, Einbinden von Hochschulbereichen mit weniger Erfahrung im Feld der privatwirtschaftlichen Auftragsforschung oder wissenschaftlichen Dienstleistungen, unterschiedliche Kulturen im Hinblick auf die Einbindung von Studieren als studentische oder wissenschaftliche Hilfskräfte oder durch kollaborative Lehrveranstaltungen, Bachelor- und Masterarbeiten, Regelungen zu Nutzungs- und Verwertungsrechten an Ergebnissen der multilateralen Zusammenarbeit.

Im Jahr 2013 wurde die Strategie des SICP in einem Strategiepapier niedergeschrieben, das Anfang 2020 für den Zeitraum bis 2026 überarbeitet wird. Wichtige Punkte bei der Neuausrichtung sind das fachliche F&I-Profil, die Bedeutung von Anwendungsdomänen (Verticals), Kooperationsbeziehungen im Innovationsökosystem, der Zuschnitt der Kompetenzbereiche, die Adressierung wichtiger gesellschaftlicher Herausforderungen.

4 Der Neue: SICP – Software Innovation Campus Paderborn

Das Leistungsangebot des SICP umfasst die Konsortialbildung, Planung und Durchführung von Forschungs- und Innovationsprojekten in Form von eigenfinanzierten Auftrags-,

Verbund- oder mit Drittmitteln unterstützten Förderprojekten (insbesondere durch die Manager der Kompetenzbereiche und ein eigenes Projektbüro zur administrativen Unterstützung der Beantragung und Abwicklung der Projekte), das Trend- und Förder-Scouting sowie die Erstellung von Technologie-, Markt- und Nutzerstudien, die gemeinsame Durchführung studentischer Bachelor- und Masterarbeiten oder Projekte, Weiterbildungsprogramme (Professional Education, z.B. Schulungen, Workshops, Seminare), Wissenstransfer (Vorträge etc.), Weiterbildungs- und Qualifizierungsangebote für Mitarbeiter(innen) und Studierende, die Mitwirkung in Netzwerken sowie gemeinsame Marketing- und PR-Aktionen (Messen, Wissenschafts- und Pressekommunikation).

Dieses Leistungsspektrum basiert auf dem interdisziplinären Wissen der Beteiligten, den komplementären Kompetenzen der Mitglieder sowie der Anwendung und dem Transfer neuester Forschungsergebnisse, die in organisationsübergreifenden Arbeitsgruppen und maßgeschneiderten Projektteams zur Entwicklung technologieorientierter, innovativer Lösungen genutzt werden. Open Innovation wird zudem durch den Campus und das auf Zusammenarbeit angelegte Raumkonzept gefördert.

Fachlich gliedert sich der SICP (das SI-Lab) in Kompetenzbereiche, die eine grundlegende organisationale Strukturierung geben. Die Kooperation und insbesondere die Projekte liegen aber oft querschnittlich hierzu; denn gerade in der Interdisziplinarität liegt eben die Stärke der Konstruktion. Aktuell gibt es im SICP fünf Kompetenzbereiche: Cyber-Physical Systems, Digital Business, Digital Security, Smart Systems und Software Engineering. Sie bilden das Organisationsgerüst des SI-Lab. Sie werden jeweils von einer Hochschullehrerin oder einem Hochschullehrer als Direktor(in) wissenschaftlich geleitet und in der Regel von einer promovierten wissenschaftlichen Mitarbeiterin bzw. einem promovierten wissenschaftlichen Mitarbeiter als Manager(in) operativ geführt. Darüber hinaus gibt es eine Leitungsstruktur für den SICP aus Vertreterinnen und Vertretern der Universität Paderborn und der weiteren Mitglieder des SICP (vgl. Abb. 1) sowie den Vorstand des SI-Lab. Weitere Details zum Aufbau des SICP finden sich in [SE15].

5 Zusammenfassung und Ausblick

Der SICP – Software Innovation Campus Paderborn ist Ergebnis der konsequenten Evolution des s-lab – Software Quality Lab. Das fachliche Ziel des SICP ist es, software- und datengetriebene Innovationen hervorzubringen und die Entwicklung zunehmend intelligenter, hochgradig vernetzter, software-intensiver Systeme für innovative Produkte, Lösungen und Dienstleistungen zu forcieren. Deshalb spielt die Disziplin Softwaretechnik nach wie vor eine zentrale Rolle, auch wenn sie inzwischen durch zahlreiche andere für die Zielsetzung ebenso wichtige Disziplinen flankiert wird. Mit dem Forschungscampus-Modell ist ein innovativer Rahmen geschaffen worden, um aktuelle Themen mit hoher praktischer Relevanz disziplin- und organisationsübergreifend zu bearbeiten um aktuelle Themen mit hoher praktischer Relevanz gemeinsam zu bearbeiten, die disziplin- und organisationsübergreifende

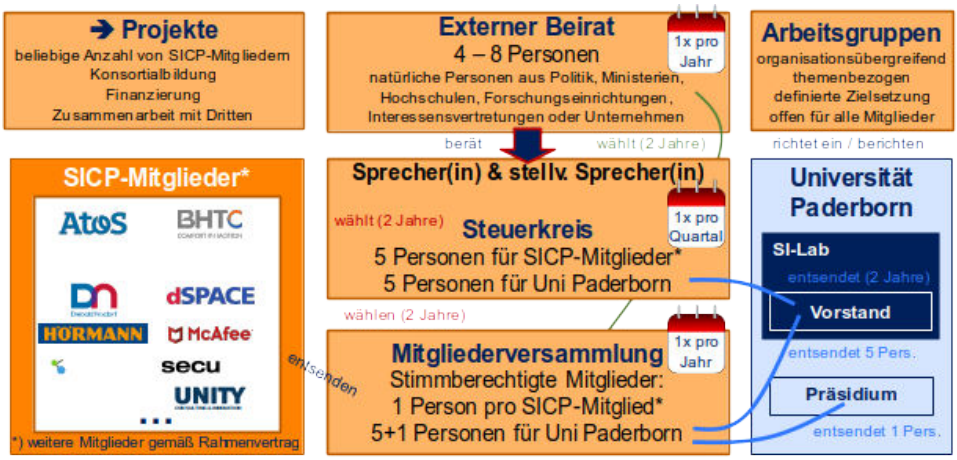


Abb. 1: Aufbau-Organisation und Gremien des SICP

Zusammenarbeit zu intensivieren und durch Faktoren wie Open-Innovation- und New-Work-Modelle zu fördern katalysieren. Das trianguläre Innovationsmodel, die Expertise von 30+ Professoren/-innen Professorinnen und Professoren, die sich ergänzenden Kompetenzen der Mitgliedsunternehmen und das professionelle F&I-Management durch Direktoren und Manager der Kompetenzbereiche sowie 30+ Jahre Erfahrung in Forschungskooperationen und Wissenschaftstransfer bilden hierfür das Fundament.

Um die Idee des SICP auch physisch in die Realität umzusetzen, entsteht aktuell an der Zukunftsmeile Fürstenallee in Paderborn der Gebäudekomplex Zukunftsmeile 2 (siehe Abb. 2), den der SICP im Jahr 2020 beziehen wird.

Abb. 2: Neubau der Zukunftsmeile 2 (Erstbezug 2020) [Bildquelle: © 2019 Goldbeck GmbH

Literatur

[SE15] Sauer, S.; Engels, G.: Neue strategische Forschungspartnerschaften zwischen Wissenschaft und Wirtschaft im deutschen Innovationssystem. In (Koschatzky, K.; Stahlecker; T., Hrsg.). Fraunhofer-Institut für System- und Innovationsforschung ISI, Sttutgart, Kap. SICP – Software Innovation Campus Paderborn: Strategische Forschungspartnerschaft für software-getriebene Innovationen, S. 135–159, 2015.

Software engineering knowledge transfer channels between university and medical device industry: a gap analysis

Zlatko Stapić[1], Nadica Hrgarek Lechner[2], Marko Mijač[3]

Abstract: This paper describes a number of different channels for bidirectional software engineering (SE) knowledge transfer between academia and medical device industry. The paper also brings the results of the gap analysis showing the discrepancy between the medical device industry needs and the curricula of relevant courses. We perform a case study of the Faculty of Organization and Informatics (FOI) at the University of Zagreb (Croatia).

Keywords: innovation; knowledge transfer; medical device industry; regulated industry; software engineering

1 Introduction

In today's world of industrial automation, connected devices, artificial intelligence and digital transformation, more than ever before, software engineering plays an important role in many industries. University education in software engineering as well as ongoing collaboration between university and industry can be viewed as one of success factors for development of innovative products in highly regulated industries such as pharmaceuticals, in-vitro diagnostics, and medical devices. On one hand, higher education institutions such as universities are preparing students to enter the workforce. On the other hand, regulated industries such as healthcare, pharmaceutics or automotive require special knowledge and skills. To close this gap, universities need to make their syllabuses more relevant to the needs of the job market.

In order to narrow this gap in the field of software engineering, we have witnessed few quite different approaches and points of view. For example, Ghezzi and Mandrioli discuss that in engineering *learning by studying at school* is not the same as *learning by doing at work*. They think that even if a field of study is rapidly evolving, education should put focus on principles and long-lasting concepts [GM06]. Oppositely to their point of view, to improve

[1] University of Zagreb, Faculty of Organization and Informatics, Laboratory of Applied Software Engineering, Pavlinska 2, 42000 Varaždin, Croatia. E-mail: zlatko.stapic@foi.hr
[2] MED-EL Elektromedizinische Geräte GmbH, Fürstenweg 77a, 6020 Innsbruck, Austria. E-mail: nadica.hrgarek@gmail.com
[3] University of Zagreb, Faculty of Organization and Informatics, Laboratory of Applied Software Engineering, Pavlinska 2, 42000 Varaždin, Croatia. E-mail: marko.mijac@foi.hr

doi:10.18420/SE2020_54

the education of software engineers, Lethbridge et al. are arguing that curricula should be forward-looking and industrial practices should be effectively communicated to students [Le07]. Third view is taken from Connor at al. who are focusing on requirements engineering to narrow the identified research-practice gap [CBP09]. When focusing on the internships, Almi at al. reveals that even a semester-long internship at companies is not enough for students to meet the industry requirements [Al11]. Similarly, Hanna et al. analyzed more than 1000 job listings to compare SE curricula and job requirements just to find out that academia is not aware of the existing very big discrepancy [Ha14]. A comprehensive meta-analysis trying to align industrial needs and SE education along with the most required skills was done by Garousi et al. who report that software requirements, design and testing are the most important skills and that the greatest gaps are in configuration management, SE models, methods, process, design, architecture and testing [Ga19a, Ga19b].

University-industry interactions include a wide range of channels for knowledge transfer and bring many benefits for both academia and regulated industry. The most common benefits for industry may include: bringing innovations [Re19], easier recruiting through strong cooperation with the universities, top talent retention, reduced initial training time of new employees who have completed an academic degree at the universities with curricula that highly match the industry needs. The benefits for academia may include: providing better education to students who aim to apply for positions in regulated industries, making universities more attractive to potential students, etc.

As part of our case study, we identify different channels of knowledge transfer between the university and medical device industry and examine the curricula of the following three university courses taught at FOI: Software Engineering, Software Analysis and Design, and Information Systems Security. Afterwards we conduct a gap analysis to identify the gaps between the syllabuses and medical device industry needs.

2 Knowledge transfer channels

FOI is a constituent part of the University of Zagreb and has been the first higher education (HE) institution in Croatia that combined information and organizational sciences. It could be considered as the leading HE institution in Croatia providing education in applied information technology and information sciences. Today, the faculty aims to develop its scientific and research activities in the following research fields and topics: ICT application in the private and public sector, information systems, Internet of everything, big data analytics, artificial intelligence, information security and open systems, organizational design, business process re-engineering, decision support (systems), e-learning, electronic and mobile business, software engineering, service-oriented architectures, biometrics, quantitative methods for decision making, risk analysis, project management, and strategic planning. FOI has been involved in several scientific, research and development, and commercial projects. The faculty has close cooperation with the IT companies and is also a co-owner of the Technology Park located in Varaždin.

FOI uses a number of different channels for bidirectional software engineering knowledge transfer between university and industry. In context of the medical device industry, we propose the following channels of knowledge transfer: publication of research results in CECIIS (Central European Conference on Information and Intelligent Systems) conference proceedings and JIOS (Journal of Information and Organizational Sciences) journal published by FOI, participation in CECIIS conference program committee, development of prototypes as part of the industry oriented development projects, student/faculty staff participation in pilot projects, co-patenting, innovation pitch awards, hackathons that aim to solve problem statements for specific industry themes, student internships, student theses, medical device manufacturers providing scholarships to students, innovation incubators at the faculty, medical device manufacturers founding university/academic spin-off companies that transform technological inventions developed from university research, keynotes from industry practitioners at the faculty, meet-ups with industry practitioners, presenting research results from the faculty to medical device manufacturers, attending at the software engineering conferences and workshops organized by the faculty, medical device company presentations at the job fairs organized by the faculty, cooperation with other similar universities/faculties, Erasmus student exchange programs, students conducting their professional practice in the medical device industry, cooperation of Student Support and Career Development Centre with medical device manufacturers, case studies from the medical device industry during faculty lectures and lab exercises, and cooperation with the Technology Park Varaždin.

A novel approach in knowledge transfer that was recently introduced to courses of Software Engineering and Software Analysis and Design, brings industry into classroom in forms of industry-defined project assignments, co-mentoring and infrastructural support [SČM19]. By working in close collaboration with partnering companies on industry-defined and co-mentored projects, students have chance to gain a first-hand insight into challenges, requirements and practices industry faces with, as well as to obtain knowledge and skills from the experienced software engineers.

3 Gap analysis

Medical device manufacturers may develop medical devices that contain software (including firmware) and/or software that is a medical device (including mobile medical applications). Software engineering knowledge that is specific for the medical device industry covers the following topics: global regulations for design and development of health software (e.g., quality (management) system according to ISO 13485 standard [IS16] and 21 CFR 820 [FD19], risk management covering safety and security risks according to ISO 14971 standard [IS19] and technical information report AAMI TIR57 [AA16], software life cycle processes according to IEC 62304 standard [Co06], safety and security of health software products according to IEC 82304-1 standard [Co16], usability engineering according to IEC 62366-1 standard [Co15], etc.), modern software architectures, requirements engineering,

software design specifications, traceability analysis, handling of software changes and bugs, configuration management, creating technical documentation for regulatory submissions, secure coding, applying coding styles, writing unit tests, software verification and validation, regression testing, conducting (peer) code reviews, using agile frameworks (e.g., Scrum, Kanban) and practices during the product development, software project management, cryptography, data protection and privacy, information security, penetration testing, etc.

Based on the medical device industry needs, we identified three related university courses taught at FOI (i.e., Software Engineering, Software Analysis and Design, and Information Systems Security) and compared syllabuses against the industry needs. The Software Engineering course provides a detailed overview of software engineering process and practices and teaches the students methodological approach to develop software products. On the other hand, the Software Analysis and Design course introduces students to the full modern software development life cycle, including domain analysis, requirements specifications, methods and techniques for software design, software development, software testing and debugging. In addition, students learn fundamental approaches that are used in development and engineering of complex software systems and modern tools and procedures that make this process easier.

In context of the medical device industry, the following gaps have been identified if taking into consideration the combined knowledge students get in these two courses: (1) global regulations for design and development of health software are mainly not covered, only software life cycle processes and usability engineering are covered, but without direct considerations of IEC 62304 and IEC 62366-1 standards; (2) traceability analyses along with handling of software changes and bugs are partially covered but configuration management is not part of the curricula; (3) although created, technical documentation is not aligned with requirements for regulatory submissions of medical device products; (4) secure coding practice is not used; (4) testing is covered in theory, but courses only partially cover its practical aspects; (5) cryptography is not part of the curricula; (6) data protection and privacy are only mentioned; (7) information security and penetration testing are not part of the curricula.

However, some topics relevant to medical device industry are addressed by these courses to a large degree: (1) software development life cycle models are discussed in general, including traditional and modern ones; (2) requirements engineering is taught and demonstrated as one of the essential parts of software development process (e.g., IEEE Software Requirements Specification template is used); (3) design activity resulting in software design specifications using UML and other techniques is extensively covered and applied; (4) handling of software changes and bugs is presented through the use of modern platforms for software versioning (such as Git and GitHub); (5) applying coding styles is encouraged by demonstrating good and established coding practices, as well as bad ones resulting in code smells; (6) writing unit tests is covered during discussion on testing in general, test-driven development and regression testing; (7) benefits of code reviews and similar agile practices in terms of software quality, productivity and knowledge sharing are discussed; (8) classical waterfall

model as well as agile frameworks such as Scrum and Kanban are presented and practiced as a means of organizing project activities and efforts in an agile team and software development process; (9) software architectural and structural design are performed on both undergraduate and graduate levels. Other topics related to software engineering in general are also covered, such as software product lines, functional and reactive programming principles, user interfaces and user experience design, etc.

The Information Systems Security course focuses on the area of organizational and technical aspects of information security and covers the majority of medical device industry needs. To fully cover the industry needs, the course would have to integrate security risk management process according to AAMI TIR57.

4 Conclusion

In this paper we have presented the case study results of the bidirectional software engineering knowledge transfer channels between university and industry, along with the gap analysis between the medical industry requirements and academic curricula for the three identified and relevant courses: Software Engineering, Software Analysis and Design, and Information Systems Security.

The analysis of knowledge transfer channels showed the variety of means for bidirectional knowledge transfer. These channels are used on different levels of FOI's structure, from management and administrative, through laboratories and centers to teachers and students at courses and projects. The overall infrastructure for knowledge transfer could be described as good.

However, the gap in undergraduate and graduate students' knowledge, if compared to industry needs, points out some weak aspects, particularly in the curricula itself. As the university study programs are, due to heavy regulations, hard to change, the curricula of observed courses hardly meets industry needs. The general conclusion is that restructuring, in both thematic and time manner, is necessary in order to include the topics that were identified as missing. The identified gap could be narrowed by restructuring and enhancing the current courses to better address the industry needs or by introducing a new course that would cover the topics specific for medical device industry needs. In our future research we plan to propose such curricula for currently relevant and necessary new courses in the field of information science.

References

[AA16] AAMI: Principles for medical device security—Risk management. Technical Information Report AAMI TIR57:2016, AAMI, 2016.

[Al11] Almi, Nurul Ezza Asyikin Mohamed; Rahman, Najwa Abdul; Purusothaman, Durkadavi; Sulaiman, Shahida: Software engineering education: The gap between industry's requirements

and graduates' readiness. ISCI 2011 - 2011 IEEE Symposium on Computers and Informatics, pp. 542–547, 2011.

[CBP09] Connor, Andrew M.; Buchan, Jim; Petrova, Krassie: Bridging the research-practice gap in requirements engineering through effective teaching and peer learning. ITNG 2009 - 6th International Conference on Information Technology: New Generations, pp. 678–683, 2009.

[Co06] Commission, International Electrotechnical: IEC 62304:2006 Medical device software — Software life cycle processes — Amendment 1. International Standard IEC 62304:2006, IEC, 2006.

[Co15] Commission, International Electrotechnical: IEC 62366 Medical devices — Part 1: Application of usability engineering to medical devices. International Standard IEC 62366-1:2015, IEC, 2015.

[Co16] Commission, International Electrotechnical: IEC 82304 Health software — Part 1: General requirements for product safety. International Standard IEC 82304-1:2016, IEC, 2016.

[FD19] FDA: 21 CFR 820 Quality System Regulation. Code of Federal Regulations 21CFR820, 2019.

[Ga19a] Garousi, Vahid; Giray, Görkem; Tüzün, Eray; Catal, Cagatay; Felderer, Michael: Aligning software engineering education with industrial needs: A meta-analysis. Journal of Systems and Software, 156:65–83, 2019.

[Ga19b] Garousi, Vahid; Giray, Gorkem; Tuzun, Eray; Catal, Cagatay; Felderer, Michael: Closing the Gap Between Software Engineering Education and Industrial Needs. IEEE Software, 7459(c), 2019.

[GM06] Ghezzi, Carlo; Mandrioli, Dino: The challenges of software engineering education. In: Proceedings. 27th International Conference on Software Engineering, 2005. ICSE 2005. IEEE, pp. 637–638, 2006.

[Ha14] Hanna, Samer; Jaber, Hayat; Almasalmeh, Ayad; Jaber, Fawze Abu: Reducing the Gap between Software Engineering Curricula and Software Industry in Jordan. Journal of Software Engineering and Applications, 07(07):602–616, 2014.

[IS16] ISO: ISO 13485:2016 Medical devices — Quality management systems — Requirements for regulatory purposes. International Standard ISO 13485:2016, 2016.

[IS19] ISO: ISO 14971:2019 Medical devices — Application of risk management to medical devices. International Standard ISO 14971:2019, 2019.

[Le07] Lethbridge, Timothy C.; Diaz-Herrera, Jorge; LeBlanc, Richard J. Jr.; Thompson, J. Barrie: Improving software practice through education: Challenges and future trends. In: Future of Software Engineering (FOSE '07). volume 2. IEEE, pp. 12–28, May 2007.

[Re19] Reichert, Sybille: , EUA STUDY: The Role of Universities in Regional Innovation Ecosystems. https://www.eua.eu/downloads/publications/eua%20innovation%20ecosystem%20report_final_digital.pdf, 2019. access 2019-10-15.

[SČM19] Stapić, Zlatko; Čižmešija, Antonela; Mijač, Marko: Software Engineering Education in Collaboration with Industry: An Experience Report. In: Accepted for publishing in ICERI 2019 - 12th annual International Conference of Education, Research and Innovation. International Academy of Technology, Education and Development, Seville, Spain, 2019.

Ein neuer Lösungsansatz für agile Produktentwicklung in der Medizintechnik

Alexander Pirker[1] Nadica Hrgarek Lechner [2]

Abstract: Komplexe Medizinprodukte folgen typischerweise einer Systemarchitektur, die aus mehreren Komponenten wie Firmware, Hardware und Software besteht. In vielen größeren Unternehmen aus dem Bereich der Medizintechnik gibt es traditionelle Organisationsstrukturen, welche diese Komponentengliederung reflektieren. Diese Unternehmen stehen nun vor der Herausforderung, neue Arbeits- und Organisationsformen zu finden, welche die Integration agiler Methoden in existierende Produktentwicklungsprozesse ermöglichen. Daraus ergibt sich folgende Problemstellung: Wie kann man ein Medizinprodukt über mehrere Abteilungen und Teams hinweg, effizient mit agilen Projektmanagementmethoden wie Scrum entwickeln? Dieser Beitrag erläutert einen möglichen Lösungsansatz, wie man interdisziplinäre Systemfunktionen mit Hilfe agiler Vorgehensmodelle effizient umsetzen und auf Veränderungen der Anforderungen schnell reagieren kann. Dabei handelt es sich um einen Vorschlag aus der Praxis, welcher noch wissenschaftlich erprobt und optimiert werden muss.

Keywords: Agile Methoden; Medizinprodukte; Medizintechnik; Produktentwicklung; Scrum; Wissenstransfer

1 Einleitung

Die Innovationen, steigende Kundenerwartungen, erhöhter Preisdruck, schnelle Anpassung an geänderte Marktbedingungen und globaler Wettbewerb führen zu komplexeren Medizinprodukten mit immer kürzeren Produktlebenszyklen. Medizinprodukte werden in der Regel in einer äußerst heterogenen Umgebung angewendet, welche andere Softwaresysteme (Server-Systeme, Cloud, etc.), Firmware und auch Hardware beinhaltet. Dies führt zu einer hohen Komplexität solcher Produkte, was eine erhebliche Herausforderung für Medizintechnikunternehmen darstellt. Um trotzdem möglichst schnell zu liefern und Entwicklungskosten zu reduzieren, versuchen viele Medizintechnikunternehmen agile Methoden anzuwenden.

Die Einführung und Anwendung agiler Prinzipien und Methoden birgt Potenziale, aber auch viele Herausforderungen, die in der Regel Veränderungen in der Aufbauorganisation erfordern. Ein typischer Prozess für die Entwicklung von Medizinprodukten beginnt mit der

[1] MED-EL Elektromedizinische Geräte GmbH, Fürstenweg 77a, 6020 Innsbruck, Österreich alexander.pirker@medel.com
[2] MED-EL Elektromedizinische Geräte GmbH, Fürstenweg 77a, 6020 Innsbruck, Österreich nadica.hrgarek@gmail.com

doi:10.18420/SE2020_55

Erfassung der Produktanforderungen, gefolgt von der Betrachtung der Systemschnittstellen und des Systemkontexts. Um diesen zu bestimmen und eine geeignete Systemarchitektur zu entwerfen, ist ein multidisziplinäres Team aus Mitgliedern unterschiedlicher Fachbereiche notwendig. Ein solches Team soll ohne externe Abhängigkeiten in der Lage sein, ein voll funktionsfähiges Produktinkrement zu erstellen [SK17]. Arbeiten in selbstorganisierten und interdisziplinären Teams, die sich aus Mitgliedern verschiedener Fachrichtungen zusammensetzen, fördert das Wissenstransfer und alle Beteiligten verfügen immer über alle aktuellen Informationen. Unglücklicherweise sind viele Medizintechnikunternehmen allerdings traditionell gewachsen, wodurch die Aufbauorganisation eines solchen Unternehmens und die damit verbundene Teamstruktur die technische Schichtung der entwickelten Produkte reflektiert. Dies führt zu einer Wissenskonzentration innerhalb von Teams und einem damit einhergehenden fehlenden Wissenstransfer über Teamgrenzen hinweg. Speziell davon betroffen sind Schnittstellen zwischen verschiedenen Fachbereichen, was die Entwicklungsdauer von Produkten massiv verlängert und nicht selten zu Fehlentwicklungen aufgrund fehlender Information führt.

Hiermit betrachten wir einen neuen Lösungsansatz für die agile Produktentwicklung in größeren Medizintechnikunternehmen und versuchen, einen Überblick zu geben.

2 Neuer Lösungsansatz für die agile Entwicklung von Medizinprodukten

Ein Team für die Entwicklung einer Systemfunktion muss alle Mitglieder beinhalten, welche man benötigt, um den Systemaspekt zu liefern, und wird als Systemteam bezeichnet, siehe Abb. 1. In der Regel besteht ein solches Team aus Fachteams der einzelnen Bereiche Firmware, Hardware, Software, technische Dokumentation, etc.

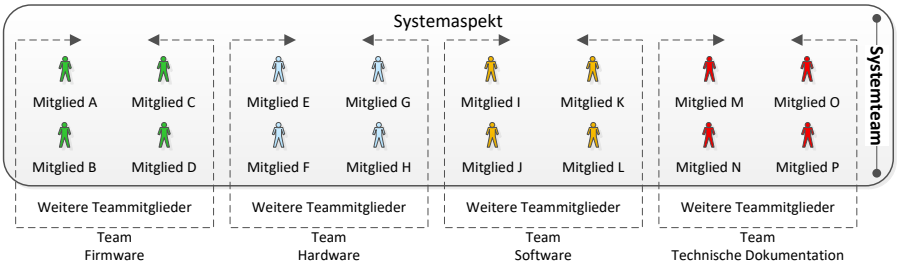


Abb. 1: Ein Systemteam beinhaltet diverse Fachteams, bestehend aus mehreren Personen des jeweiligen Fachbereichs, welche benötigt werden, um eine bestimmte Systemfunktion zu implementieren.

In vielen Medizintechnikunternehmen arbeiten die einzelnen Fachteams, welche die technische Schichtung des Produktes reflektieren, bereits agil. Die einzelnen Fachteams folgen

dabei agilen Entwicklungsprozessen wie zum Beispiel Scrum, und arbeiten in Sprints. Allerdings agieren sie in der Regel sehr unabhängig voneinander, was dem Konzept einer Systemfunktion widerspricht, da eine Systemfunktion nur geliefert werden kann, wenn alle einzelnen technischen Komponenten des Produktes jene unterstützen. Die Unabhängigkeit der jeweiligen Fachteams hat zur Folge, dass Information weder zum richtigen Zeitpunkt noch in der richtigen Form verfügbar ist. Das Kernproblem eines solchen Entwicklungsmodelles liegt meist in den Schnittstellen zwischen den fachlich getrennten Systemen, sowie einem klaren Verständnis des Systemverhaltens. Dem kann zwar durch die Definition von System/Schnittstellen-Standards entgegengewirkt werden, allerdings ist diese Form der Kommunikation zwischen Fachteams sehr eingeschränkt und korrespondiert nicht mit agilen Prinzipien. Es kann weder schnell auf Veränderungen reagiert werden noch ist die Erstellung eines Produktinkrements effizient möglich. Einen vermeintlichen Ausweg stellt die Kommunikation durch die jeweiligen Fachteamleiter dar. Dieser Lösungsansatz wirkt erfolgsversprechend, führt allerdings leider nicht zur gewünschten Agilität. Dies gründet sich darauf, dass die Entwickler der einzelnen Fachteams (welche die Herausforderungen ihres Fachbereichs am besten kennen und beurteilen können) zu wenig Einfluss auf die Definition des Gesamtsystems, die Schnittstellen zwischen den technischen Komponenten, sowie deren Umsetzung haben. Es gilt nun ein agiles Vorgehensmodell zu finden, welches Fehlentwicklungen frühzeitig vermeidet, allerdings die Unabhängigkeit von Fachteams behält. Laut [De18], kann die Arbeit in vernetzten Teams und ein übergreifendes Systemdenken zum Erreichen organisationaler Flexibilität beitragen.

Ein mögliches agiles Vorgehensmodell wäre das Folgende: Die Fachteams, welche eine Systemfunktion entwickeln und nach fachlichen Organisationseinheiten ausgerichtet sind, arbeiten weiterhin in weitgehend unabhängigen Sprints voneinander. Allerdings wird bei jedem System-Sprint, welcher sich aus den einzelnen und miteinander koordinierten Sprints der Fachteams zusammensetzt, ein sogenanntes 'Interface und System Board' als zusätzliches Projektgremium bestimmt. Das 'Interface und System Board' besteht aus jeweils einem Mitglied eines Fachteams und wird mit jedem System-Sprint neu bestimmt, siehe Abb. 2.

Dieses Board hat nun eine Reihe von Aufgaben. Erstens soll es die in diesem System-Sprint benötigten Erweiterungen des Systems/Schnittstellen-Standards festlegen, da ohne jene Festlegungen kein Produktinkrement erfolgreich erstellt werden kann. Außerdem müssen die dort getroffenen Entscheidungen an die entsprechenden Fachteams kommuniziert werden. Der internen Kommunikation in einem Fachteam kommt dabei eine besondere Bedeutung zu, um Fehlentscheidungen sowie Fehlentwicklungen frühestmöglich zu korrigieren. Daher wäre es durchaus sinnvoll, dass die Mitglieder des 'Interface und System Boards' wöchentlich ein Standup-Meeting für ihr Fachteam abhalten, damit alle Mitglieder stets auf dem neuesten Stand der Interface- und System-Entwicklungen sind. Außerdem muss dieses Board den Schnittstellen-Standard warten und gegebenenfalls an neue Gegebenheiten oder Veränderungen anpassen.

Die Neubesetzung des 'Interface und System Boards' zu Beginn jedes System-Sprints ist

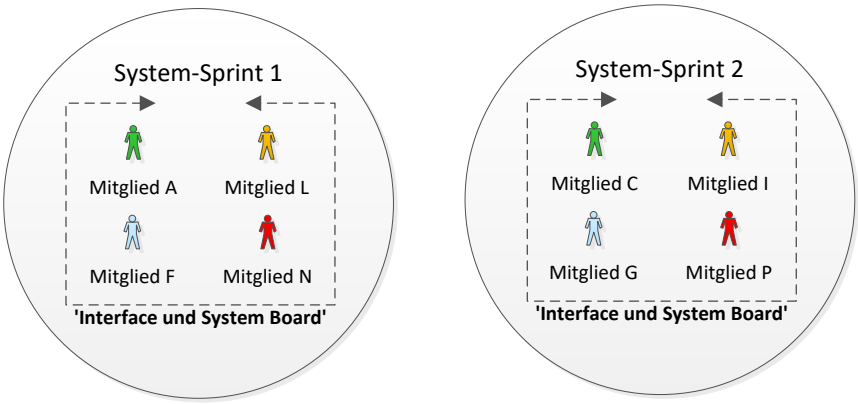


Abb. 2: Ein 'Interface und System Board' besteht aus jeweils mindestens einem Mitglied eines Fachteams und wird mit jedem System-Sprint neu besetzt. Der Neubesetzung dieses Teams kommt dabei eine Schlüsselfunktion zu, da sie es ermöglicht, das Wissen über das Systemverhalten sowie die Systemschnittstellen effektiv zu verteilen.

hier entscheidend. Eine statische Besetzung dieses Boards hat einen großen Nachteil: Das Wissen über das Systemverhalten sowie die Schnittstellen zwischen den Komponenten, ist für jedes Fachteam in einer Person konzentriert. Sollte diese Person aus dem Unternehmen scheiden oder fällt sie über einen längeren Zeitraum aus, ist auch jenes essenzielle Wissen nicht mehr im Fachteam vorhanden. Dies resultiert in einem erheblichen Mehraufwand, da eine andere Person sich jenes Wissen aneignen muss. Diese 'Schlüsselperson' muss an alle Fachteammitglieder die getroffenen Entscheidungen bezüglich des Systemverhaltens kommunizieren, was eine weitere Fehlerquelle birgt. Durch eine Neubesetzung des Boards kann diesen Nachteilen effektiv entgegengewirkt werden, da eine homogene Verteilung der Information über den Schnittstellen-Standard und das Gesamtverhalten des Systems erreicht wird. Außerdem reduziert sich durch die Neubsetzung der vermeintlich große Mehraufwand beim Ausfall einer statischen 'Schlüsselperson' auf einen regelmäßigen kleinen Aufwand, welcher viel einfacher zu bewältigen ist und deutlich weniger Zeit in Anspruch nimmt.

Das 'Interface und System Board' rotiert seine verantwortlichen Mitglieder von System-Sprint zu System-Sprint. Dadurch wird sichergestellt, dass alle Mitglieder aus den beteiligten Organisationseinheiten zwischenzeitlich Teil des 'Interface und System Boards' sind. Aufgrund der Rotation wird ein klares Verständnis des Gesamtsystems (sowie dessen Schnittstellen) von jedem einzelnen Fachteammitglied gewährleistet. Das 'Interface und System Board' ist flexibel und kann daher auf Änderungen (z.B. Abwesenheiten, Einarbeitung neuer Mitglieder, Wechsel im Team, etc.) schnell reagieren. Allerdings kann die Rotation zur Ineffizienz und Ineffektivität durch Informationsverlust oder redundante

Wiederholungen führen. Die regelmäßige Kommunikation mit den Teammitgliedern ist zwar ein kleiner Mehraufwand während eines Sprints bezüglich der getroffenen Entscheidungen im 'Interface und System Board', allerdings tragen sie maßgeblich zum Verständnis des Gesamtsystems sowie dessen korrekter Implementierung und Qualitätssteigerung bei.

3 Praxisbeispiel

Nehmen wir an, dass wir für die Entwicklung eines Medizinprodukts vier Organisationseinheiten brauchen, um einen bestimmten Systemaspekt zu implementieren. Wenn wir aus jeder Organisationseinheit insgesamt vier Mitglieder benötigen, ergibt sich daraus eine Summe von 16 Mitglieder wie in der Abb. 1 dargestellt. Jedes Fachteam implementiert die ihm zugeordnete Komponente bzw. erstellt die ihm zugeordnete technische Dokumentation des Gesamtsystems. Wie aus Abb. 1 ersichtlich ist, erstreckt sich die Implementierung über Software, aber auch Firmware und Hardware. Dies erfordert nun ein klares Verständnis des Systemverhaltens sowie eine klare Definition der Schnittstellen zwischen den einzelnen involvierten technischen Komponenten.

Zu Beginn eines individuellen System-Sprints, welcher sich über das gesamte System erstreckt, könnte man nun ein 'Interface und System Board' definieren. In unserem Beispiel setzt sich dieses sich aus vier Mitgliedern zusammen, siehe Abb. 2. Jedes Mitglied vertritt dort den ihm zugeordneten Fachbereich. Das 'Interface und System Board' ist mit der in diesem System-Sprint erforderlichen Erweiterung des Schnittstellen-Standards sowie dem Definieren des Systemverhaltens betraut. Zum Beispiel sollte dieses Board festlegen, wie Software mit Hardware bzw. Firmware sicher kommuniziert und welche Daten in welchem Format ausgetauscht werden müssen. Außerdem kann das Mitglied des Fachbereichs 'Technische Dokumentation' bereits sehr früh auf mit der Erstellung der entsprechenden Dokumentation, welche am Ende eines System-Sprints bei Behörden eingereicht werden muss, erstellen. Dies verkürzt die Entwicklungszeit bis hin zur Freigabe eines Medizinproduktes stark. Um nun Entscheidungen und Festlegungen dieses Boards an die jeweiligen Mitglieder der Fachteams zu kommunizieren, sollten die 'Interface und System Board' Mitglieder wöchentliche Standup-Meetings abhalten.

4 Zusammenfassung und Ausblick

Im digitalen Zeitalter werden traditionelle Formen der Aufbauorganisation herausgefordert und die Medizintechnikunternehmen suchen nach neuen flexibleren und selbstorganisierten Organisationsformen. Dieser Beitrag schlägt einen neuen Lösungsansatz vor, in dem verschiedene Fachteams in einem Systemteam für einen Produktentwicklungsprozess in der Medizintechnik zusammengeführt werden. Die Hauptidee besteht darin, die Teamstruktur, welche die technische Schichtung der entwickelten Medizinprodukte reflektiert, für die Agilisierung und die damit einhergehende Verbesserung der Kommunikation und inkrementellen Entwicklung über ein sogenanntes 'Interface und System Board' zu vernetzen.

Die aus den Fachteams entsendeten Board-Mitglieder sollen von Sprint zu Sprint rotieren. Die Einführung eines dezidierten, rotierenden Boards zur Wartung und Definition von Systemschnittstellen und Systemverhalten ermöglicht Agilität in der Entwicklung komplexer Medizintechnikprodukte und behält die Unabhängigkeit von Fachteams bei. Ein weiterer Vorteil dieses agilen Vorgehensmodells ist, dass sich das Wissen mit der Zeit in Fachbereichen verteilt. Der vorgeschlagene Lösungsansatz gibt neue Impulse und kann nicht nur in der Medizintechnik, sondern auch in anderen Branchen eingesetzt werden.

5 Danksagung

Wir danken den anonymen Reviewern für ihre wertvollen Anregungen, Kommentare und Korrekturen.

Literaturverzeichnis

[De18] Deloitte: , Organisation neu denken: Flexible Organisationsmodelle für das digitale Zeitalter. https://www2.deloitte.com/content/dam/Deloitte/de/Documents/human-capital/Organisation-neu-denken-flexible-organisationsmodelle-2018.pdf, 2018. Zugriff am 2019-10-14.

[SK17] Schwaber K., Sutherland J.: , The Scrum Guide. https://www.scrumguides.org/docs/scrumguide/v2017/2017-Scrum-Guide-US.pdf, 2017. Zugriff am 2019-09-12.

MoMuT - Eine Transfer-Geschichte über modellbasiertes Testen

Rupert Schlick[1]

Abstract: Forschungstransfer hat viele Gesichter, aber in den seltensten Fällen eine geradlinige Entwicklung. Am Beispiel modellbasierter, mutationsgetriebener Testfallgenerierung wird die Entwicklung einer Methode von der Konzeptionsphase zu (ersten) kommerziellen Verwertungen erzählt. Anhand dessen werden einige der wirkenden Kräfte und Einflussfaktoren betrachtet.

Keywords: Modellbasiertes Testen; Erfahrungsbericht Forschungs-Transfer; Reaktive Systeme

1 Einleitung und Hintergrund

AIT arbeitet an Verfahren für mutationsgetriebene, modellbasierte Testfallgenerierung. Die dabei entstandene Familie von Testfallgeneratoren *MoMuT*[2], wurde in mehreren Etappen um verschiedenste neue Module ergänzt. Im Folgenden betrachten wir deren Werdegang und wie sich Industrieanforderungen auf Entwicklung und Transfer ausgewirkt haben.

Prinzipiell gibt es zwei Ansätze für den Transfer von Forschungsergebnissen in die industrielle Anwendung: *Market-Pull*, wenn ein konkretes Problem von der Industrie an einen Forschungspartner herangetragen und von diesem (hoffentlich) gelöst wird, und *Technology-Push*, wenn eine als nützlich vermutete Technologie umgesetzt und dann dazu Nutzer gesucht werden. Ein reiner Market-Pull, der nicht innerhalb der Forschungsabteilung eines Unternehmens, sondern durch Zukauf der Forschungsleistung bei einem Transfer-Institut erfolgt, ist im Software-Engineering-Umfeld selten, da die Probleme nicht so kundenspezifisch sind.

Die Realität in geförderten, anwendungsorientierten Forschungsprojekten liegt zwischen den beiden Extremen. Um Mittel zu erhalten, muss das Vorhaben eine benennbare, generell nützliche Innovation als Lösung für eine Kategorie von Problemen versprechen. Üblicherweise sind, im Gegensatz zu einem reinen Technology-Push, Anwendungspartner mit an Bord, die Beispiel-Probleme mitbringen. Diese haben Interesse an der Lösung. Jedoch sind die Beispiel-Probleme womöglich gar nicht mit dem Ansatz lösbar oder nicht repräsentativ.

[1] AIT Austrian Institute of Technology GmbH, Center for Digital Safety and Security, Giefinggasse 4, 1210 Wien, Österreich, rupert.schlick@ait.ac.at

[2] `http://momut.org`

doi:10.18420/SE2020_56

Modellbasiertes Testen (MBT) generiert Testsequenzen aus Modellen. Hier verstehen wir darunter, aus einem Modell, das das Verhalten der Software/des Systems spezifiziert, Testfälle mit gegebenen, angestrebten Testzielen, z.B. einer Abdeckung des Modells, zu erstellen. Das Erstellen des (Test-)Modells ersetzt das Erstellen der einzelnen Testfälle. Beides erfordert ein Verständnis der Anforderungen - der Aufwand für die beiden Erstellungsschritte liegt in der gleichen Größenordnung, erfordert aber verschiedene Fähigkeiten und Fertigkeiten.

Ausgehend vom unveränderten Modell, kann die Tiefe/Qualität, bis zu der getestet werden soll, weitgehend frei erhöht werden. Bei direkt erstellten Tests erfordert dies zusätzlichen Arbeitsaufwand, mit MBT nur Rechenaufwand. Am meisten Vorteile bringt MBT in der Wartung und in der Entwicklung abgeleiteter Softwarevarianten. Modelländerungen für neue Funktionen bedeuten einen überschaubaren Aufwand. Dann kümmert sich der Testfallgenerator darum, welche Tests unverändert bleiben, wegfallen, adaptiert werden müssen oder neu hinzukommen. Auch in gut strukturierten Testsuiten gibt es eine Größenobergrenze, ab der es personell nicht mehr sinnvoll möglich ist, zu überprüfen, ob alle Tests nach wie vor ihren ursprünglichen Zweck erfüllen.

Mutationsgetriebenes Modellbasiertes Testen Ein mögliches Testziel im MBT ist Mutationsabdeckung. Dabei wird im Modell jeweils eine kleine Änderung vorgenommen, ein kleiner Fehler eingebracht. Anschließend werden Testsequenzen gesucht, in denen sich das außen (durch den Tester/Benutzer) beobachtbare Verhalten von verändertem und unverändertem Modell unterscheiden. Typische Kontrollflussabdeckungen als Testziel führen dagegen tendenziell zu schlechteren Tests. Ein erfahrener Tester baut, ebenso wie die Mutationsabdeckung, Tests, die ein Problem beobachtbar machen. Bei Strukturabdeckungen kann ein Test nach Erreichen der geforderten Stelle im Modell enden, obwohl ein aufgetretenes Problem nicht beobachtbar wird.

2 Initialzündung

Das FP7-Projekt MOGENTES[3] beschäftigte sich mit fehlerbasierten Methoden zum Test. Mutationsbasierte Testfallgenerierung war im Prinzip als Möglichkeit bekannt [AD06], aber sowohl akademische als auch kommerzielle Werkzeuge für mutationsgetriebenes MBT Testen waren nicht verfügbar. Im Projekt arbeiteten fünf Forschungspartner und ein Hersteller von Entwicklungswerkzeugen an fehlergetriebenen Verifikationsansätzen für Beispielanwendungen von vier Industriepartnern. TU Graz und AIT entwickelten einen ersten mutationsgetriebenen Testfallgenerator für UML [Ai11]. UML wird dazu in die Zwischensprache OOAS (Object Oriented Action Systems) übersetzt. Implementiert wurde mit VIATRA, ANTLR, C# und Prolog.

Der Ansatz funktionierte, hatte aber erhebliche Skalierungsprobleme. Für das Beispiel eines Eisenbahnstellwerks konnte nur eine minimalistische Schienentopologie verwendet werden

[3] Model-based Generation of Tests for Dependable Embedded Systems, `http://mogentes.eu/`, 2008-2011

und die Steuerung einer Baggerschaufel erforderte Hinweise zu Äquivalenzklassen im Input, da der enumerative Ansatz mit großen Datenbereichen überfordert war. Ein einfaches, aber realistisches Beispiel einer Automobilalarmanlage funktionierte hingegen sehr gut und ist repräsentativ für eine ganze Klasse von Aufgaben im Bereich Chassis-Steuerung.

3 Performance

Das national geförderte Projekt TruFal[4] widmete sich der Fragestellung, wie der Ansatz effizienter umzusetzen wäre. Ein wiederum durch die TU Graz in Prolog umgesetzter Generator unter Einsatz von symbolischen Techniken und SMT-Solvern (Z3) lieferte gute Ergebnisse für die Baggerschaufel-Steuerung aus MOGENTES und die Testsystem-Einbindung eines Abgas-Messgerätes [Ai14], scheiterte aber, wegen Datentyp-Einschränkungen des Solvers, wegen des großen Zustandsraums und wegen wenig optimierter Implementierung an größeren Topologien für die Stellwerkssteuerung und an einer Safety-Middleware.

Die Transformation von UML nach OOAS und der OOAS-Parser wurden nach Java migriert, um Erweiterungen und Änderungen zu erleichtern. Die Abkehr von VIATRA war motiviert von der leichteren Verfügbarkeit von JAVA-Programmierern. Gegen Ende von TruFal implementierte AIT einen neuen, wieder enumerativen Generator, der das hochgradig nebenläufig modellierte Stellwerksmodell bewältigen sollte. Er vermeidet die zuvor pro Mutant wiederholten Explorationen und führt Mutanten nur für die jeweils notwendigen Schritte aus. Das Explorations-Framework verwendet zudem einen (später als zu einfach erkannten) Ansatz für Partial Order Reduction, wie ihn Model Checker für nebenläufige Systeme verwenden. Die Architektur wurde gezielt performant gewählt, das Modell wird von OOAS nach LLVM-IR weitertransformiert und dann in Maschinencode übersetzt, der Generator selbst ist in C++ implementiert.

4 Babel der Modelle

Im Rahmen des Projekts MBAT[5] wurde die Anwendung des Ansatzes auf die Datenflusssprache von SCADE evaluiert. Ein Mutations-Mechanismus wurde implementiert, für die Testfallgenerierung wurde der zu SCADE verfügbare Model-Checker verwendet. Abgesehen von Limitierungen des Model-Checkers, konnte die Funktion zwar gezeigt werden - der Anwendungspartner hat jedoch seine Nutzung der Sprache im Zielumfeld zurückgefahren, der Toolhersteller war nicht interessiert, das Thema wurde aufgegeben.

Beide an TruFal beteiligten Industriepartner stellten fest, dass der Ansatz wert ist, weiterverfolgt zu werden, UML aber nicht der passende Modellierungsformalismus ist. AVL

[4] Trust via Failed Falsification of Complex Dependable Systems Using Automated Test Case Generation through Model Mutation, `https://trufal.wordpress.com/`, 2011-2014

[5] Combined Model-based Analysis and Testing of Embedded Systems,`http://www.mbat-artemis.eu/home/`,2011-2014

entwickelte in der Folge unter enger Einbindung der betroffenen Mitarbeiter eine eigene domänenspezifische Sprache (DSL), inklusive Editor-Unterstützung, Anbindung an die interne Geräte-Datenbank, Transformation nach OOAS und Transformation der Tests in das eigene nUnit-Framework. Diese Lösung ist nach etwas zusätzlicher Reifezeit mittlerweile im Produktiveinsatz [St18].

THALES Österreich ging in Richtung formaler Modellierung und begann Event-B und die zugehörige Umgebung Rodin[6] einzusetzen. Die Ziel-Anwendung (interne Projekte) änderte sich aus unternehmensinternen Gründen mehrfach. AIT entwickelte ein Plug-in für Rodin, das eine Untermenge von Event-B nach OOAS übersetzen kann. Rodin unterstützt (teil-)automatische Verifikation von Refinement, das heißt Modelle können, unter Erhalt von z.B. Safety-Zusicherungen, immer weiter detailliert werden, bis das Modell sehr nah an der Implementierung ist. Die aus dem letzten Refinement generierten Tests können dann verwendet werden, um den letzten Schritt zur Implementierung über Tests abzusichern.

5 Performance, die Zweite

Zur weiteren Verbesserung der Performance wurde in mehrere Richtungen gearbeitet. Rapidly exploring Random Trees (RRT) sind nun als Explorations-Strategie verfügbar [Fe19]. Will man (im Security-Umfeld) mittels Model-Checker Fragen zur Vertraulichkeit beantworten, lassen sich diese nicht als Aussagen über einzelne Traces ausdrücken, sondern nur als Aussagen über (mögliche) Paare von Traces, sogenannte Hyper-Properties. Das gilt auch für mutationsgetriebenes MBT, und ein verfügbarer symbolischer Model-Checker für Hyper-Properties wurde auf seine Anwendbarkeit für Mutation Testing untersucht [FBW19]. Die Verwendung von Partial Order Reduction (POR) für Testfallgenerierung ist komplexer als für Model Checking, da in der Exploration als identisch verworfene, alternative Interleavings für den Test rekonstruiert werden müssen. Die bisherige, naive Umsetzung von POR wurde überarbeitet und auf eine solide theoretische Basis gestellt.

6 Mehr als nur funktionales Testen

Initiiert von THALES Österreich und gemeinsam mit der Universität Southampton wurde ein Prozess erarbeitet, der Behavior Driven Development auf Modelle umlegt, die formale Modellierung unterstützt und über inkrementelle Testfallgenerierung einen indirekten Review des Modells durch den Review von Tests ermöglicht [Sn18]. Im Projekt EMBEET[7] wurde UML wieder aufgegriffen - zusammen mit zwei Partnern wurde eine integrierte Modellentwicklungsumgebung entwickelt. Sie unterstützt die Modellierung von Szenearien und Funktion, Rückverfolgbarkeit zu Anforderungen, inkrementelle Testfallgenerierung

[6] http://www.event-b.org/

[7] Environment for Model-Based Embedded Systems Engineering and Testing,http://embeet.com/, 2016-2019

und indirekten Modell-Review ebenso wie On-Target-Debugging auf Modellebene. Eine Produktveröffentlichung / erste Kundenprojekte werden 2020 erwartet.

Verfügbare funktionale Modelle können auch für andere Zwecke genutzt werden. Im neuesten MoMuT steht für Robustheits-Tests eine Smart-Fuzzing-Funktion zur Verfügung. Sie soll demnächst in Projekten zum Protokoll-Testen verwendet werden. Gemeinsam mit der TU Graz wurde ein Ansatz zum modellbasierten Performance-Test entwickelt. Ein erstelltes Verhaltensmodell und ein gelerntes Performance-Modell, das z.B. Reaktionszeiten vorhersagt, werden zur Ableitung von Performance-Aussagen und zur experimentellen, statistischen Prüfung der Vorhersagen verwendet [Ai19].

Im Projekt AQUAS[8] arbeitet AIT derzeit daran, die Ansätze für Performance-Tests, Robustheits- (und Security-)Tests und funktionale Tests zu verbinden. Beispiel ist eine von Integrasys entwickelte Infrastruktur zur Koordination zwischen Unmanned Aerial Vehicles (UAV).

7 Fortsetzung folgt ...

Die Erfahrungen der letzten Jahre auf dem Weg vom technischen Lösungsansatz zu industrieller Anwendung und wirtschaftlicher Verwertung lassen folgende Schlüsse zu:

Die hohe Spezialisierung auch in den Computerwissenschaften führt dazu, dass es schwierig ist, „Probleminhaber“ und die passenden Experten zusammenzuführen. Die Geschichte von MoMut ist doch auch die Geschichte einer Lösung auf der Suche nach einem Problem.

Mit einigen unserer Industriepartner hatten wir großes Glück, weil sie die Geduld hatten, auszuharren, weil sie Vertrauen in den Ansatz und AIT als Partner hatten. Viele begonnene Kooperationen sind trotzdem versandet, aus Gründen, die nichts mit der Anwendbarkeit der Lösung oder der Qualität der Zusammenarbeit zu tun hatten, sondern mit Firmenpolitik, internen Strategien, leeren Budget-Töpfen, oder Personalwechseln. Mitunter hatten einfach andere Probleme, als die mit automatisierter Testfallgenerierung zu lösenden, Vorrang.

Bewährt hat sich, für die Testfallgeneratoren mit OOAS[9] auf eine Zwischensprache unter eigener Kontrolle zu setzen und über Transformationen die benutzerseitigen Modellierungssprachen einzubinden. Das hat spätere Erweiterungen wesentlich vereinfacht.

Die Einführung neuer Lösungen in Unternehmen ist ein komplexes Unterfangen. Ein gelungenes Beispiel dafür ist die Einführung modellbasierter Ansätze und einer DSL bei AVL [St18]. Eine Rückmeldung aus der Industrie ist, dass die Forschungspartner auf fundamental neuen Ansätzen basierende Lösungen (Revolutionen) anbieten, die Industrie aber Evolution mit kalkulierbaren Risiken braucht.

[8] Aggregated Quality Assurance for Systems (AQUAS),http://aquas-project.eu/, 2017-2020

[9] OOAS wurde ursprünglich von der TU Graz entwickelt, Sprachdefinition und Parser-Quellen sind unter Open-Source-Lizenz verfügbar. https://www.momut.org/code/projects/ooastools/repository

Die Förderlandschaft fordert auch für anwendungsorientierte Forschung einen hohen Neuheitsgrad - das führt fast zwingend zu den genannten Revolutionen, während Forschungsgruppen immer neue Themen anreißen, um den neuen Ausschreibungsschwerpunkten zu folgen. Es bleibt eine große Lücke zur Produktreife - Benutzbarkeit und Anwenderakzeptanz, aber auch Dokumentation, finden hier keinen Platz. Auch nützliche Technologien sind oft nicht attraktiv genug für einen Investor, um das Schließen dieser Lücke zu finanzieren.

Im Projekt EMBEET war die Förderschiene EUROSTARS sehr hilfreich, deutlich näher an ein markttaugliches Produkt zu kommen.

Die Bewegung vom Einzelwerkzeug für funktionale Testfallgenerierung zu einerseits der Integration in Entwicklungsumgebungen und Gesamtlösungen (UML wie DSL) und die Ausweitung zum Test nicht-funktionaler Anforderungen unter synergetischer Nutzung von Modellierungsaufwänden zeigt eine Entwicklung von Technologieorientierung hin zur Nutzenorientierung. Es ist zu hoffen, dass dieser Lernschritt weiterhin durch zunehmenden produktiven Einsatz der Werkzeuge belohnt wird.

Literatur

[AD06] Aichernig, B. K.; Delgado, C. C.: From Faults Via Test Purposes to Test Cases: On the Fault-Based Testing of Concurrent Systems. In (Baresi, L.; Heckel, R., Hrsg.): FASE 2006. S. 324–338, 2006.

[Ai11] Aichernig, B. K.; Brandl, H.; Jöbstl, E.; Krenn, W.: Efficient Mutation Killers in Action. In: ICST 2011. S. 120–129, 2011.

[Ai14] Aichernig, B. K.; Auer, J.; Jöbstl, E.; Korosec, R.; Krenn, W.; Schlick, R.; Schmidt, B. V.: Model-Based Mutation Testing of an Industrial Measurement Device. In: TAP 2014. S. 1–19, 2014.

[Ai19] Aichernig, B. K.; Bauerstätter, P.; Jöbstl, E.; Kann, S.; Korosec, R.; Krenn, W.; Mateis, C.; Schlick, R.; Schumi, R.: Learning and statistical model checking of system response times. Software Quality Journal 27/2, S. 757–795, 2019.

[FBW19] Fellner, A.; Befrouei, M. T.; Weissenbacher, G.: Mutation Testing with Hyperproperties. In: SEFM 2019. S. 203–221, 2019.

[Fe19] Fellner, A.; Krenn, W.; Schlick, R.; Tarrach, T.; Weissenbacher, G.: Model-based, Mutation-driven Test-case Generation Via Heuristic-guided Branching Search. ACM Trans. Embed. Comput. Syst. 18/1, 4:1–4:28, Jan. 2019.

[Sn18] Snook, C. F.; Hoang, T. S.; Dghaym, D.; Butler, M. J.; Fischer, T.; Schlick, R.; Wang, K.: Behaviour-Driven Formal Model Development. In: ICFEM 2018. S. 21–36, 2018.

[St18] Stieglbauer, G.; Burghard, C.; Sobernig, S.; Korosec, R.: A Daily Dose of DSL - MDE Micro Injections in Practice. In: MODELSWARD 2018. S. 642–651, 2018.

Coaching on the Job bei Unternehmen des Maschinenbaus

Agil Wissenslücken schließen zur Weiterentwicklung modernisierter IT-Anwendungen

Masud Fazal-Baqaie[1] Jan-Niclas Strüwer[2] David Schmelter[3] Stefan Dziwok[4]

Abstract: Digitalisierung und Industrie 4.0 ist in aller Munde und Anbieter von Maschinen und Anlagen beschäftigen sich zunehmend mit digitalen Geschäftsabläufen und IT-gestützten Produkten. Bezogen auf die steigenden Bedarfe der dazu notwendigen Fähigkeiten und Kenntnisse, mangelt es oft an ausreichend qualifiziertem Personal. Ohne die notwendigen Softwaretechnikkenntnisse erfüllen realisierte Lösungen aber nicht die Erwartungen der Nutzer und sind schlecht wartbar. In diesem Papier berichten wir von unseren Erfahrungen mit zwei Unternehmen aus der Domäne des Maschinen- und Anlagenbaus, mit denen das Fraunhofer IEM Softwarelösungen modernisiert hat. In beiden Projekten war es ebenfalls Ziel, die bestehenden Mitarbeitenden zu befähigen, die modernisierte Software selbstständig zu pflegen und weiterzuentwickeln. Wir berichten von unserem abgeleiteten Lösungskonzept für die Kombination von agiler Entwicklung und „agilem Coaching". Insbesondere mit „Coaching-Stories" gelang es uns, die Themen der Wissensvermittlung gegen die Aufwände für Softwaremodernisierung abzuwägen und für alle Seiten transparent zu priorisieren.

Keywords: Coaching on the Job, Coaching-Stories, Softwaremodernisierung, Arbeit 4.0

1 Einleitung

Die Branche des Maschinen- und Anlagenbaus ist einem grundlegenden Wandel unterworfen, der einen hohen Innovationsdruck erzeugt und zu mehr Flexibilität und Kundenfreundlichkeit zwingt. Um diesen geschäftskritischen Veränderungen gerecht zu werden, müssen sich sowohl die Produkte der Unternehmen als auch die Unternehmen und ihre Mitarbeitenden verändern, um mit modernen Technologien hochqualitative, vernetzte Anwendungen auf agile Weise zu entwickeln [Wa13].

Weder einfache Weiterbildungsmaßnahmen, noch externe IT-Dienstleistungen werden dieser Situation gerecht. Eine Fortbildungsmaßnahme der Mitarbeitenden (z.B. eine Schulung über 2-3 Tage) löst nicht das Legacy-Problem der IT-Anwendungen, da hierfür eine sehr hohe Kompetenz bzgl. Softwaretechnik notwendig ist [SPL03]. Eine Beauftragung von externen Dienstleistern für die Softwaremodernisierung hingegen schließt nicht die

[1] Fraunhofer IEM, Paderborn, Germany masud.fazal-baqaie@iem.fraunhofer.de
[2] Fraunhofer IEM, Paderborn, Germany jan-niclas.struewer@iem.fraunhofer.de
[3] Fraunhofer IEM, Paderborn, Germany david.schmelter@iem.fraunhofer.de
[4] Fraunhofer IEM, Paderborn, Germany stefan.dziwok@iem.fraunhofer.de

doi:10.18420/SE2020_57

Kompetenzlücke der Mitarbeitenden, so dass diese nicht in der Lage sind die IT-Anwendung selbstständig weiterzupflegen und zu betreiben. Unsere Lösung ist daher ein Vorgehen, bei dem die Entwicklungsmodernisierung (Produkt, Entwicklungsinfrastruktur, Organisation und Prozesse) und das Coaching der Mitarbeitenden verzahnt vorangetrieben werden. Dazu begleiten externe Experten das Entwicklungsvorhaben und die Mitarbeitenden über einen Zeitraum von mehreren Monaten. Basierend auf unseren Erfahrungen bei zwei Unternehmen stellen wir dazu unser Lösungskonzept vor, das wir *Coaching on the Job mit Coaching-Stories* nennen.

Im Folgenden beschreiben wir in Kapitel 2 die Ausgangslage unserer Fallbeispiele. In Kapitel 3 stellen wir unsere Lösung vor und berichten von unseren Erfahrungen damit. Wir schließen unseren Beitrag mit einem Fazit und Ausblick in Kapitel 4.

2 Charakterisierung der Ausgangslage

Wir illustrieren die allgemeine Situation im Maschinen- und Anlagenbau anhand zweier konkreter Unternehmensbeispiele (Produkt A und Produkt B). Beide Unternehmen sind Großunternehmen – in KMUs ist die Situation aus unserer Erfahrung aber vergleichbar, teilweise noch gravierender. In beiden Unternehmen (bzw. Abteilungen) ist die Entwicklung von modernen vernetzten, web-basierten Anwendungen bisher nicht Teil des Kerngeschäfts. Mit Produkt B möchte man mit der IT-Anwendungsentwicklung zudem Standortsicherung betreiben, um sich gegen andere Standorte des Unternehmens durchsetzen. Für beide Produkte gilt, dass die entsprechende Abteilung keinen Anwendungsentwicklungs-Schwerpunkt hat. Für Produkt A galt es, zwei bis vier Mitarbeitende über ein Jahr zu begleiten. Für Produkt B wurden ca. vier bis fünf Mitarbeitende mehr als anderthalb Jahre begleitet. Wir unterscheiden im Folgenden die Dimensionen *Produkt* und *Mitarbeitende*. Unser Lösungskonzept berücksichtigt darüber hinaus die Dimensionen *Organisation und Prozesse* sowie *Entwicklungsinfrastruktur* (siehe [Fa19]), die wir hier aus Platzgründen aussparen.

Produkt
Bei beiden zu modernisierenden IT-Anwendungen handelte es sich um monolithisch entwickelte Programme, die ohne eine fundierte Architektur umgesetzt worden waren. Bei Produkt A war ein lokales, prototypisches Administrationswerkzeug der Ausgangspunkt, das zu einem modernen, webbasierten Tool für verschiedene (auch externe) Nutzergruppen entwickelt werden sollte. Bei Produkt B war das Ziel, mehrere statische Websites, die zur Visualisierung und Verwaltung des Supply-Chain-Management-Prozesses dienten, zu einer modernen, verteilten Web-Applikation umzubauen. Beide Produkte sollten den entsprechenden Abteilungen als technische Grundlage für weitere Produkte dienen.

Die bis dato entwickelten IT-Anwendungen waren in beiden Fällen schlecht wart- und erweiterbar. Bei Produkt B war insbesondere auch die Performance unzureichend. Zudem

waren die Produkt A und B aufgrund veralteter Technologien nicht Mobile- bzw. Cloud-ready. Fehlende moderne Softwaretechnikkonzepte in alten Technologien erschwerten zum Beispiel die Erstellung automatisierter Tests. Im Projektverlauf wurden, wie für moderne Web-Anwendungen üblich, ein Unterbau aus Frameworks und Komponenten von Drittanbietern gewählt und unterschiedliche Technologien kombiniert.

Mitarbeitende
Die zu schulenden Mitarbeitenden von Produkt A hatten tiefgehende Erfahrungen im Bereich der Automatisierungstechnik und bei der Entwicklung für Anlagensteuerungen. Das Team von Produkt B hatte Erfahrungen mit skriptbasierter Serverprogrammierung. Beide Teams hatten Expertise zu altbewährten, teilweise veralteten IT-Technologien, aber es fehlte das Wissen zu den relevanten neuen IT-Technologien, mit denen die Produkte im Projektverlauf modernisiert wurden. In beiden Teams fehlten auch fundierte Softwaretechnik-Kompetenzen, z.B. bezüglich Requirements Engineering (RE), Architektur, Gestaltung von Benutzungsoberflächen (UI/UX), Testing und zu agilen Prozessen. Die Mitarbeitenden von Produkt A kann man außerdem als Quereinsteiger für die Entwicklung von IT-Anwendungen bezeichnen. Sie waren bisher sehr lösungsorientiert und fokussiert vorgegangen, die erwähnten Softwaretechnik-Kompetenzen waren nachrangig. Für beide Teams gilt, dass die Mitarbeitenden nicht dediziert an Produkt A oder B arbeiten, sondern stark durch das Tagesgeschäft mit wechselnden Prioritäten belastet sind.

3 Coaching on the Job: Integrierte Produktmodernisierung und Weiterbildung

In diesem Kapitel möchten wir unser Lösungskonzept vorstellen, das wir aus unseren Projekterfahrungen mit Coaching on the Job abgeleitet haben.

3.1 Lösungskonzept

Unserem Lösungskonzept liegt die Adaption eines agilen Vorgehensmodells, wie z.B. Scrum [SB02], zugrunde. Die Kernidee ist an die Beschreibung von Anforderungen durch User-Stories [Co04] angelehnt (vgl. Abbildung 1, Mitte). Die Arbeit zur *Entwicklungsmodernisierung* wird durch sogenannte *Refactoring-Stories* (vgl. Refactors bei SAFe [Le10]) und die Arbeit zum Coaching durch sogenannte Coaching-Stories sichtbar und planbar gemacht. Mit diesen zusätzlichen Typen von Stories kann man ähnlich umgehen, wie mit regulären User-Stories. Beispielsweise lassen sich Akzeptanzkriterien definieren, die für die erfolgreiche Umsetzung gelten. Damit können Refactoring- und Coaching-Stories auch in den entsprechenden Events wie Sprint Planning und Sprint Review mitbehandelt

und eng in die "reguläre" Entwicklung integriert werden. Die Entwicklungsmodernisierung mit Refactoring-Stories umfasst die Veränderungen am Produkt. Das Coaching mit Coaching-Stories beschreibt die Qualifizierung der Mitarbeitenden.

Refactoring- und Coaching-Stories verbessern die Integration von Modernisierung und Coaching in die Produktentwicklung. Allerdings ist damit alleine nur unzureichend inhaltliche Struktur für die beiden Themenbereiche gegeben. Beispielsweise bliebe unklar, wann auf den agilen Prozess umgestellt (Refactoring-Story) und wann dieser geschult (Coaching-Story) werden sollte. Als weitere wesentliche Kernidee des Lösungskonzepts werden daher die notwendigen Arbeiten und ihre Reihenfolge als Handlungsempfehlung systematisiert (vgl. Abbildung 1, unten und oben). Wie in Abbildung 1 illustriert, durchläuft die *Entwicklungsmodernisierung* die Phasen *Analyse*, *Strukturierung* und *Transformation*. Beispielsweise werden erst die bestehenden Abläufe und Verbesserungswünsche des Prozesses analysiert, dann das Scrum-basierte Zielmodell strukturiert und schließlich die bestehenden Abläufe durch neue Abläufe ersetzt. Für die Refactoring-Stories heißt das, dass sich die Anforderungen über die Iterationen hinweg von Analysethemen hin zu Transformationsthemen verlagern. Das *Coaching* der Mitarbeitenden durchläuft die Phasen *Initialisierung*, *Befähigung* und *Übergabe/Konservierung*. Passend zu der Reihenfolge der Modernisierungsaktivitäten liegt in der Initialisierung der Schwerpunkt auf prozessuale Themen wie Scrum und Agiles RE. Daran schließen sich während der Befähigung weitere Grundlagenthemen an, um mit den Verbesserungen der Entwicklungsinfrastruktur umgehen zu können. Über Themen zur Verbesserung der Produktqualität, beispielsweise Softwarearchitektur und Entwicklertests (Unit Tests), bewegt sich die Qualifizierung zunehmend in Richtung Zieltechnologien. Über den Zeitverlauf wird die Autonomie der zu qualifizierenden Mitarbeitenden gesteigert. Dieser Trend ist im untersten Teil der Abbildung illustriert: Während die Mitarbeitenden von Anfang an praktisch umsetzen, was die Coaches konzipiert haben, übernehmen sie über die Zeit auch die Verantwortung über Konzepte, die von den Coaches qualitätsgesichert werden. Letztendlich verantworten die Mitarbeitenden dann die Konzipierung, Realisierung und Qualitätssicherung der eigenen Arbeiten.

3.2 Lessions Learned

Basierend auf der Rückmeldung der Mitarbeitenden und der Coaches, stellen wir hier unsere Lessons Learned dar. Mittels unseres Coaching on the Job mit Coaching-Stories haben wir erste, positive Erfahrungen gesammelt. Die folgenden positiven Aspekte möchten wir dabei herausheben:

Vorteile der Agilität für das Coaching

In dem beschriebenen Kontext ist ein agil organisiertes Coaching notwendig, da Mitarbeitende immer wieder ungeplant an anderen Themen arbeiten mussten. Außerdem zeigten sich viele Wissenslücken erst im späteren Verlauf. Mit dem agilen Coaching konnte das Team damit gut umgehen. Darüber hinaus machen Coaching-Stories die Lernaufwände auch

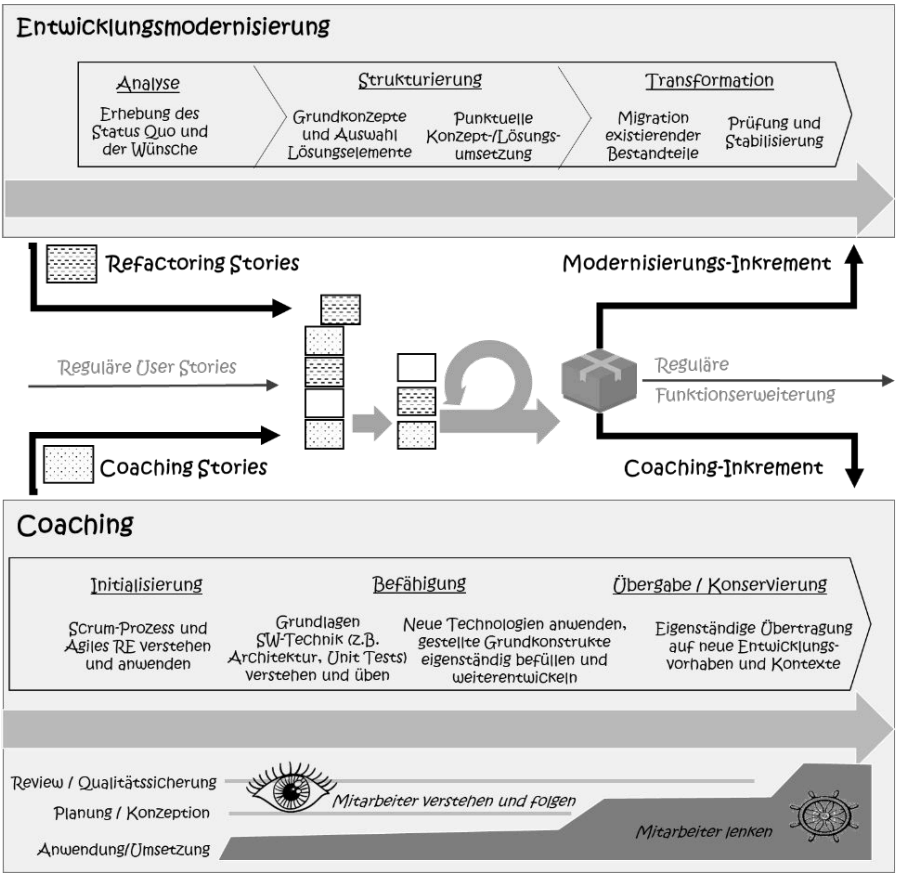


Abb. 1: Entwicklungsmodernisierung

für Stakeholder sichtbar. Wird den Mitarbeitenden wiederholt zu wenig Zeit eingeräumt, werden die Konsequenzen davon (in Form nicht umgesetzter Coaching-Stories) sehr gut sichtbar. Dies lässt auch Produktverantwortliche die Themen Funktionserweiterungen, Modernisierung und Coaching leichter gegeneinander abwägen. Das agile Coaching wird durch die agile Produktentwicklung ideal ergänzt. Es bietet den Mitarbeitenden wiederholt die Möglichkeit, Gelerntes anzuwenden und den Coaches, den Lernerfolg zu evaluieren.

Differenzierung der Dimensionen und Schwerpunktverlagerung über den Zeitverlauf

Die im Lösungskonzept skizzierte Verteilung von Themen über den Zeitverlauf reduziert die Gefahr von Überforderungen durch Veränderungen an zu vielen Fronten.

Durch den zeitlichen Versatz kann man auch Widerständen bei Mitarbeitenden entgegenwirken. Dadurch, dass die Mitarbeitenden in einem Bereich (z.B. Scrum) bereits Fortschritte

gemacht haben, sind sie eher dazu bereit, in anderen Bereichen Veränderungen anzunehmen und die Gefahr von innerer Resignation wird gemindert.

Bedeutung praktischer Lerneinheiten

Die praktischen Lerneinheiten haben geholfen, die Vorteile von neuen Techniken und Technologien zu verdeutlichen und die Motivation der Teilnehmenden zu steigern. Vorbehalte gegen Nutzen konnten so zügig abgebaut und in der Praxis auftretende Probleme im Coaching behandelt werden.

Die folgenden Aspekte stellen weiterhin eine Herausforderung dar und sollen hier deshalb kurz diskutiert werden:

Belastung der Mitarbeitenden durch die Modernisierung

Die vielen und umfangreichen Veränderungen, die durch die Modernisierung bedingt sind, bedeuten für die einzelnen Mitarbeitenden eine sehr steile Lernkurve (vgl. [CF03]). Das Coaching orientiert sich an der Belastungsgrenze, um möglichst viel Wissen zu vermitteln, es gilt aber unter allen Umständen eine Überlastung zu vermeiden.

Schnitt der Coaching-Stories

Analog zu User-Stories in der Produktentwicklung kommt dem Zuschnitt der Coaching-Stories eine besondere Bedeutung zu, denn er beeinflusst die Effektivität und Effizienz des Coaching on the Job. Für die Partitionierung der Lerninhalte und deren logischer Reihenfolge gilt auch zu beachten, wieviel Zeit zwischen zwei Coachings liegt und wie weit die Enwicklungsmodernisierung vorangeschritten ist. Der Zuschnitt von Coaching-Stories sollte, z.B. im Rahmen der Scrum-Retrospektive, regelmäßig anhand der Praxiserfahrungen überprüft werden.

Gefahr von Teamfluktuation und Kopfmonopolen

In beiden Unternehmen hat man die Chance genutzt und für begrenzte Zeit Auszubildende und Werkstudierende in die Teams entsandt, so dass diese von dem Coaching profitieren. Hier besteht die Gefahr, dass Aufgaben und Verantwortlichkeiten (Kopfmonopole) an diese Personen fließen. Dies bedroht die Nachhaltigkeit des Lernerfolgs für das restliche Team. Fluktuation während des Coachings kann ebenso eine Gefahr darstellen.

4 Fazit und Ausblick

Die Digitalisierung zwingt Unternehmen des Maschinen- und Anlagenbaus ihr Produktportfolio durch kundenzentrierte IT-Anwendungen zu ergänzen. Allerdings stellt neben der eigentlichen IT-Modernisierung die hierzu notwendige Qualifizierung von Mitarbeitenden die Unternehmen vor große Herausforderungen. In unserem Beitrag haben wir unser Lösungskonzept vorgestellt, welches sowohl die Modernisierung von IT-Anwendungen

adressiert, als auch die Qualifizierung der Mitarbeitenden. Dabei integriert es sich nahtlos in laufende Entwicklungsvorhaben. Als Qualifizierungsergebnis können die Mitarbeitenden eigenständig modernen IT-Anwendungen (weiter-)entwickeln.

Die Erfahrung mit zwei Unternehmen hat gezeigt, dass ein agiles Vorgehensmodell, bei dem Coaching und Entwicklungsvorhaben verzahnt aufeinander aufbauen, ein zentraler Erfolgsfaktor für die Qualifizierung ist. Dieses parallele Vorgehen bedeutet aber auch eine hohe Lernkurve und mentale Belastung der Mitarbeitenden, auf die es zu achten gilt.

In Zukunft möchten wir unser Lösungskonzept mit weiteren Unternehmen umsetzen und mit den Praxiserfahrungen weiter verfeinern.

Literatur

[CF03] Cohn, M.; Ford, D.: Introducing an Agile Process to an Organization. Computer 36/6, S. 74–78, Juni 2003, ISSN: 0018-9162, URL: http://dx.doi.org/10.1109/MC.2003.1204378.

[Co04] Cohn, M.: User Stories Applied: For Agile Software Development. Addison-Wesley, 2004.

[Fa19] Fazal-Baqaie, M.; Strüwer, J.-N.; Schmelter, D.; Dziwok, S.: Coaching on the Job bei Unternehmen des Maschinen- und Anlagenbaus - Wissenslücken schließen zur Weiterpflege modernisierter IT-Anwendungen. In (Mikusz, M., Hrsg.): Projektmanagement und Vorgehensmodelle 2019 (PVM 2019). Gesellschaft für Informatik, Lecture Notes in Informatics (LNI), 2019.

[Ki16] Kim, G.; Humble, J.; Debois, P.; Willis, J.: The DevOps Handbook:: How to Create World-Class Agility, Reliability, and Security in Technology Organizations. IT Revolution Press, 2016.

[Le10] Leffingwell, D.: Agile Software Requirements: Lean Requirements Practices for Teams, Programs, and the Enterprise. Pearson Education, 2010.

[SB02] Schwaber, K.; Beedle, M.: Agile Software Development with Scrum. Prentice Hall, 2002.

[SPL03] Seacord, R.; Plakosh, D.; Lewis, G.: Modernizing Legacy Systems: Software Technologies, Engineering Processes, and Business Practices. Addison-Wesley, 2003.

[Wa13] Walls, M.: Building a DevOps Culture. O'Reilly Media, 2013.

Softwareentwicklung wie am Fließband

Vorbereitung von Model-Driven Engineering im Maschinen- und Anlagenbau

Thorsten Koch[1], Matthias Meyer[1], Masud Fazal-Baqaie[1], Hubert Runschke[2]

Abstract: Unternehmen des Maschinen- und Anlagenbaus sind bestrebt, Entwicklungs- und Inbetriebnahmezeiten für Anlagen zu reduzieren und damit den Absatz zu erhöhen. Der Steuerungscode dieser Anlagen wird typischerweise nicht auf Basis anerkannter Softwaretechnik-Methoden entwickelt und nur unsystematisch per Copy & Paste wiederverwendet. Die Anwendung von moderner Softwaretechnik und insbesondere die modellgetriebene Entwicklung bergen daher ein großes Potential für die Steigerung der Qualität und Effizienz. Allerdings meiden Unternehmen die damit verbundene Komplexität und Investitionskosten. Wir präsentieren ein Stufen-Modell für die Verbesserung der Softwareentwicklung, das leichter von Unternehmen adaptierbar ist und berichten von unseren Erfahrungen mit dem Industriepartner Venjakob bei dessen Anwendung.

Keywords: Model-Driven Engineering, IEC-61131, Softwarequalität, Maschinen- und Anlagenbau

1 Einleitung

Im Maschinen- und Anlagenbau entwickeln und produzieren Unternehmen häufig Anlagen, die zwar kundenindividuell angepasst werden, aber zu großen Teilen aus sehr ähnlichen Teilsystemen bestehen. Die Steuerung solcher Anlagen erfolgt mit Hilfe von speziell darauf ausgelegten Programmiersprachen gemäß des Standards IEC 61131 [In13]. Die Entwickler haben häufig keine Informatikausbildung und nur wenig Softwaretechnik-Knowhow. Oft handelt es sich um Ingenieure aus dem Maschinenbau oder der Elektrotechnik, die das Programmieren nur in den speziellen Sprachen gelernt haben.

Bei der Entwicklung der Steuerungssoftware werden die Gemeinsamkeiten der Anlagen nur sehr unsystematisch ausgenutzt. Typischerweise wird der Softwarestand der ähnlichsten ausgelieferten Anlage per „Copy & Paste“ als Grundlage genutzt und an die Erfordernisse angepasst. Dieses Vorgehen ist sehr aufwendig und fehleranfällig. Gleiche oder ähnliche Funktionalität wird zum Teil mehrfach entwickelt, und je nach Vorliebe des jeweiligen Entwicklers unterschiedlich strukturiert. Entdeckte Fehler werden so oft nur in einer Anlage behoben anstatt in allen, in denen der Code verwendet wird. Gleichzeitig ist der Quellcode oft plattformabhängig, was wiederum ungewünschte Aufwände für die Reimplementierung bei unterschiedlichen Typen von Steuerungen, Sensorik und Aktorik mit sich bringt [Vo14].

[1] Fraunhofer IEM, Paderborn, Germany, vorname.nachname@iem.fraunhofer.de
[2] Venjakob Maschinenbau GmbH & Co. KG, Rheda-Wiedenbrück, Germany, HRunschke@venjakob.de

doi:10.18420/SE2020_58

Abhilfe kann die Nutzung von modellgetriebenen Entwicklungsmethoden (Model-Driven Engineering, MDE) [St12] schaffen. Dabei werden eine Anlage und ihre Konfiguration auf einer höheren Abstraktionsebene als plattformabhängiger Quellcode modelliert und unter Rückgriff auf wiederverwendbare Softwarebausteine durch Codegenerierung weite Teile der Software für eine Anlage automatisiert erzeugt. Manuelle, fehleranfällige Arbeiten werden so reduziert und die Qualität und Effizienz gesteigert. Die abstraktere Modellierung reduziert die Abhängigkeit von der konkreten Steuerungstechnik und maximiert die Wiederverwendung von plattformabhängigen Softwarebausteinen. Die stärkere Standardisierung reduziert schließlich die Abhängigkeit von einzelnen Entwicklern.

Bei all seinen Vorteilen ist MDE bei der Anlagenentwicklung kaum verbreitet [Vy13], denn den Unternehmen fehlt das notwendige Softwaretechnik-Knowhow. Auch scheuen sie die wesentlichen Investitionen in Modellierungssprache, Softwarebausteine und Codegenerierung. In diesem Papier stellen wir einen Ansatz für eine stufenweise Effizienz- und Qualitätssteigerung vor, der den Weg hin zu MDE systematisiert. Dabei profitiert ein Unternehmen mit jeder Stufe direkt von den Softwaretechnik-Verbesserungen, so dass sich die Aufwände direkt auszahlen. Wir berichten von den Erfahrungen mit der Firma Venjakob Maschinenbau GmbH & Co. KG, die wir mit Hilfe des Modells bei der Adaption hin zu MDE begleitet haben.

Unser Papier ist folgendermaßen strukturiert: In Kapitel 2 stellen wir das Modell vor und berichten in Kapitel 3 von dessen Anwendung bei der Firma Venjakob. In Kapitel 4 präsentieren wir unsere Lessons Learned und in Kapitel 5 fassen wir den Beitrag zusammen.

2 Stufen zur modelgetriebenen Entwicklung von Steuerungen

Die Tatsache, dass Maschinen- und Anlagen häufig aus einer Vielzahl von gleichen oder sehr ähnlichen Teilsystemen bestehen, die kundenindividuell kombiniert und konfiguriert werden, ermöglicht ein sehr effizientes und systematisches Vorgehen in der Entwicklung der Steuerungssoftware, bis hin zur modellgetriebenen Entwicklung. Solche Ansätze finden jedoch im Maschinen- und Anlagenbau bisher nur selten Anwendung. Auf Basis unserer Erfahrungen aus Projekten mit dem Ziel des Transfers von effizienteren Softwaretechnik-Methoden in Unternehmen des Maschinen- und Anlagenbaus, wie z. B. der Firma Venjakob, schlagen wir das in Abb. 1 gezeigte, dreistufige Vorgehen vor. Mit jeder Stufe wird die Effizienz und Softwarequalität gesteigert und damit die Inbetriebnahme verkürzt:

Stufe 1 Codeharmonisierung: Ausgehend von einer, wie eingangs beschriebenen, von Copy & Paste und unsystematischer Wiederverwendung geprägten Entwicklung, sollte in einem ersten Schritt eine Harmonisierung und Standardisierung des erstellten Codes erfolgen, indem Coding Guidelines (einheitliche Sprache, Begrifflichkeiten, Benennung von Variablen und Funktionen, Formatierung) und/oder Templates eingeführt werden. Auf diese Weise werden die Unterschiede zwischen verschiedenen Entwicklern und so die

Abhängigkeit von Einzelpersonen reduziert. Die Einarbeitung in andere Anlagenprojekte, die Wartung und Fehlersuche werden vereinfacht.

Stufe 2 Softwarebausteine: Die nächste Stufe treibt die bereits begonnene Standardisierung weiter, indem wiederverwendbare Softwarebausteine identifiziert und für alle Anlagenprojekte bereitgestellt werden. Die Entwicklung kann so effizienter erfolgen, da Funktionalität nicht aufwändig immer wieder neu entwickelt wird. Die Qualität steigt, da qualitätsgesicherte Implementierungen eingesetzt und Fehler vermieden werden. Entwicklungs- und Inbetriebnahmezeiten werden reduziert.

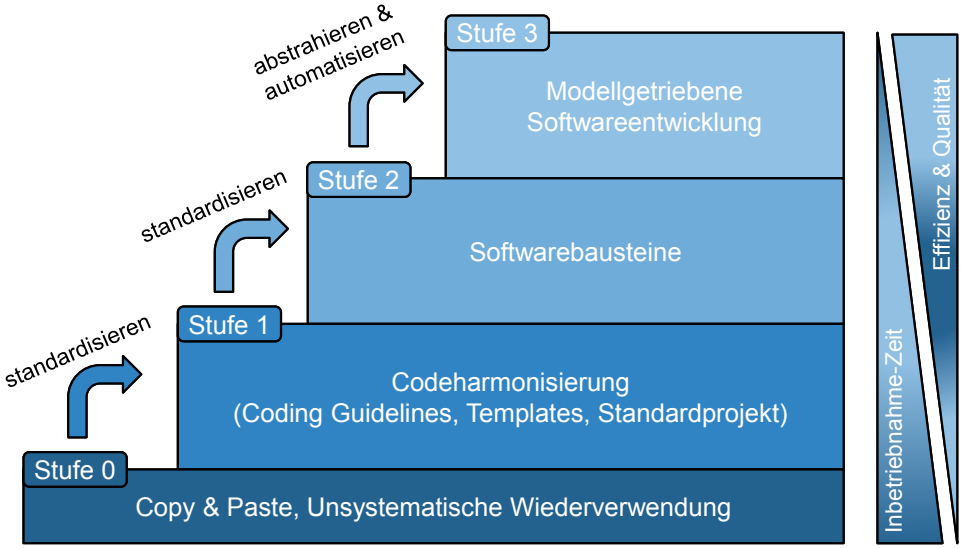


Abb. 1: Stufenmodell der Effizienz- & Qualitätssteigerung

Stufe 3 Modellgetriebene Softwareentwicklung: Weitere Effizienzsteigerungen werden durch Abstraktion und (Teil-)Automatisierung der Softwareerstellung möglich. Wissen über den typischen Aufbau von Anlagen wird genutzt, um eine angepasste Modellierungssprache zu entwickeln, die eine Modellierung und Konfiguration einer Anlage auf einer höheren Abstraktionsebene als von der Steuerungstechnik abhängiger Quellcode erlaubt. Aus solchen Modellen können unter Rückgriff auf die Softwarebausteine durch Codegenerierung weite Teile der Software für eine Anlage automatisiert erzeugt werden. Der Rest wird manuell ergänzt. Die abstraktere Modellierung reduziert die Abhängigkeit von der konkreten Steuerungstechnik und verkapselt diese in Codegeneratoren. Durch das stufenweise Vorgehen können Unternehmen mit zunächst geringem Aufwand und geringen fachlichen Anforderungen an ihre Entwickler bereits Verbesserungen in der Softwareentwicklung erzielen. Mit jeder aufbauenden Stufe wachsen Anforderungen, aber auch Mehrwerte. Stufe 2 erfordert z.B. bereits Software-Werkzeuge für die Verwaltung und Verwendung der

Bausteine. Stufe 3 benötigt dedizierte Teams mit weitreichenden Softwaretechnik-Skills, die Modellierungswerkzeug und die Codegeneratoren entwickeln.

3 Anwendungsprojekt mit Unternehmen Venjakob

Das Unternehmen Venjakob Maschinenbau GmbH & Co. KG ist ein mittelständischer Hersteller für kundenindividuelle Oberflächenanlagen und fördertechnische Produkte, das langfristig auf modellgetriebene Methoden für die Entwicklung der Steuerungssoftware umstellen möchte. Venjakob entwickelte diese bisher auf Basis eines Referenzprojektes. Für jedes Kundenprojekt wurde dieser Code kopiert und aufwendig manuell angepasst, was zu den in der Einleitung beschriebenen Fehlern und Verzögerungen führte.

Dem gemeinsamen Projekt wurde eine Ist-Analyse vorausgeschaltet. Der Softwareentwicklungsprozess wurde in mehreren Workshops aufgenommen und ein Review der Steuerungssoftware von insgesamt 20 Anlagen durchgeführt. Die resultierenden Handlungsempfehlungen wurden in dem anschließenden Hauptprojekt adressiert. Dazu wurden zwei Entwickler von Venjakob sechs Monate lang von einem wissenschaftlichen Mitarbeiter an zwei Tagen pro Woche begleitet. Die Zwischenergebnisse des Projekts wurden regelmäßig den restlichen Entwicklern vorgestellt. Entsprechend dem Stufenmodell sollte die Codeharmonisierung verbessert werden (Stufe 1). Außerdem sollte eine Referenzarchitektur mit Softwarebausteinen abgeleitet und die Mitarbeitenden zur modernen Softwaretechnik geschult werden (Stufe 2).

Für die Codeharmonisierung wurden gängige Coding Guidelines zur IEC 61131-3 Programmierung (z.B. PLCopen) gesichtet und implizite Coding Guidelines aus dem Standardprojekt extrahiert. Anschließend wurden die Vor- und Nachteile einzelner Regeln mit den zwei Entwicklern diskutiert und integrierte, einheitliche Coding Guidelines für Venjakob entwickelt.

Für die Referenzarchitektur und Softwarebausteine wurde die Einführung einer Bibliothek für Softwarebausteine adressiert, um eine systematische Wiederverwendung zu ermöglichen. Dazu wurden die beiden Entwickler in der Modellierungssprache UML [Ob17] und typischen Architekturmustern geschult. Anschließend wurde eine Referenzarchitektur für die Anlagen von Venjakob mit Hilfe der UML modelliert.

Für die Vermittlung moderner Softwaretechnik wurden die beiden Entwickler parallel zu der Erarbeitung der Referenzarchitektur in dem Versionsmanagementsystem TFS und den objektorientierten Erweiterungen der IEC 61131-3 geschult. Die Schulungsinhalte konnten während der Implementierung einzelner Bausteine der Referenzarchitektur angewendet und vertieft werden. Zudem konnten die definierten Coding Guidelines evaluiert und an einigen Stellen angepasst werden. Um die Qualität der einzelnen Softwarebausteine zu gewährleisten, wurden die beiden Entwickler zudem in Konzepten zum Testen geschult. Der Fokus lag dabei auf Unit-Tests, da ein Integrationstest aus Mangel einer realen Anlage nicht möglich war. Der Support von Unit-Tests innerhalb der IEC 61131-3 Sprachen sowie

der Entwicklungsumgebungen ist relativ gering. Aus diesem Grund wurde auf das freie TcUnit-Framework zurückgegriffen.

Als Ergebnisse lagen zum Projektende Coding Guidelines zur Harmonisierung der Steuerungssoftware inklusive der Dokumentation vor. Zudem lag eine vollständige und dokumentierte Referenzarchitektur der Anlagen von Venjakob vor. Des Weiteren wurden erste Softwarebausteine implementiert und mit Hilft des TcUnit Frameworks getestet. Die Entwickler des Projekts wurden zudem befähigt, die verschiedenen Softwaretechnik-Konzepte anzuwenden und ihre Kollegen und Kolleginnen in der Anwendung zu schulen.

Auf diese Weise konnte Venjakob die Implementierung der Softwarebausteine allein fortsetzen. In regelmäßigen Abständen haben wir uns über den Fortschritt informiert. Insgesamt hat sich gezeigt, dass es während der Implementierung kaum Änderungsbedarf an der Referenzarchitektur und den Templates zur Implementierung der Bausteine gegeben hat. Im letzten Quartal des Jahres 2019 wurde zudem mit dem Testen der Bausteine an einer Anlage begonnen.

4 Lessons Learned

Bei der Durchführung des Projekts mit Venjakob haben wir für zukünftige Projekte einige Erfahrungen gesammelt, die wir hier darstellen möchten.

Stufenmodell strukturiert die Verbesserung: Wir haben die Erfahrung gemacht, dass eine Orientierung an dem Stufenmodell zum einen die projektinterne Strukturierung von Verbesserungsvorhaben unterstützt, zum anderen aber auch die Kommunikation an Stakeholder vereinfacht, zum Beispiel beim Erwartungsmanagement für Erfolge.

Verfügbarkeit der Mitarbeitenden sicherstellen: Wenn Mitarbeitende nicht exklusiv und nicht von Beginn an für das Vorhaben zur Verfügung stehen und parallel Anlagenprojekte betreuen und daher kurzfristig ausfallen, kann das den Projektfortschritt verzögern. Das Wissen dieser Teammitglieder steht ggf. nicht zur Verfügung, wenn es benötigt wird, andererseits müssen Trainingseinheiten für sie wiederholt werden.

Überlastung der Mitarbeitenden verhindern: Das Projekt wurde als Chance genutzt, um auf eine neue Version der Entwicklungsumgebung umzustellen. Neben den modernen Softwaretechnik-Methoden mussten die Mitarbeitenden somit auch die neue Umgebung erlernen, was eine sehr steile Lernkurve bedeutete. Für den Projekterfolg ist es daher notwendig, die Belastungsgrenze im Blick zu behalten und Veränderungen einzuschränken.

Lernphasen mitplanen: Neben der Verbesserungen der Codebasis hatte das Projekt auch die Vermittlung von moderner Softwaretechnik als Ziel. Hier gilt es genug Zeit für die praktische Vertiefung einzuplanen und diese auch einzuhalten, damit das Ziel der Wissensvermittlung nicht der Arbeit an der Codebasis untergeordnet wird. Dabei können explizit eingeplante Schulungsphasen hilfreich sein.

Mitarbeitende als Technologie-Botschafter: Während des Projekts wurden den restlichen Mitarbeitenden der Abteilung wiederholt neue Konzepte vorgestellt und deren Anwendbarkeit in der täglichen Arbeit diskutiert. Wir haben die Erfahrung gemacht, dass diese positiver eingeschätzt wurden, wenn Projektmitarbeitende des Unternehmens (und nicht wissenschaftliche Mitarbeitende) Konzepte und Technologien vorstellten. Diese konnten praktische Vorteile am Beispiel vergangener Projektkontexte darstellen.

5 Zusammenfassung

Um die Entwicklung und Inbetriebnahme von Anlagen zu beschleunigen, gilt es die Entwicklung von Steuerungssoftware zu professionalisieren und auf Softwaretechnik-Knowhow zurückzugreifen. Insbesondere modellgetriebene Entwicklung erscheint hier erfolgsversprechend [Li18], den Unternehmen fehlt es aber an Wissen und sie scheuen die hohen Investitionen. Wir präsentierten hierzu ein Modell für die stufenweise Adaption von Verbesserungen, das den Pfad zur modelgetriebenen Entwicklung für Unternehmen strukturiert und erleichtert. Wir berichteten von den gemeinsamen Erfahrungen mit der Firma Venjakob, die langfristig auf modellgetriebene Entwicklung umstellen möchte und mit unserer Hilfe zwei der drei Stufen dazu erklommen hat. In Zukunft planen wir, neben Venjakob auch weitere Unternehmen mit Hilfe des Models hin zu modellgetriebener Entwicklung zu begleiten.

Literatur

[In13] International Organization for Standardization: IEC 61131-3:2013: Programmable controllers - Part 3: Programming languages, 2013.

[Ob17] Object Management Group: Unified Modeling Language – Version 2.5.1, 2017.

[St12] Stahl, T.; Völter, M.; Efftinge, S.; Haase, A.: Modellgetriebene Softwareentwicklung: Techniken, Engineering, Management. dpunkt.verlag, s.l., 2012.

[Vo14] Vogel-Heuser, B. et al.: Challenges for Software Engineering in Automation. Journal of Software Engineering and Applications 07/05, S. 440–451, 2014, ISSN: 1945-3116.

[Vy13] Vyatkin, V.: Software Engineering in Industrial Automation: State-of-the-Art Review. IEEE Transactions on Industrial Informatics 9/3, S. 1234–1249, 2013.

Profil: Software Engineering Research am AIT - Austrian Institute of Technology GmbH

Rupert Schlick[1]

Abstract: Dieser Artikel stellt das Forschungsinstitut AIT Austrian Institute of Technology GmbH vor und beleuchtet seine Rolle als Transfer-Institut für Forschung zum Software Engineering. Dazu werden einerseits die zum Thema passenden, bestehenden Aktivitätsschwerpunkte kurz vorgestellt, andererseits aktuelle Verwertungspfade berichtet und bewertet.

Keywords: Profil Transfer-Institut; Österreich; Infrastrukturbezogene Forschung

1 AIT - ein Transfer-Institut

1.1 AIT - Austrian Institute of Technology GmbH

Das *AIT Austrian Institute of Technology* ist Österreichs größte außeruniversitäre Forschungseinrichtung. Mit seinen acht Centern versteht sich das AIT als hochspezialisierter Forschungs- und Entwicklungspartner für die Industrie. Durch die Forschung und technologischen Entwicklungen des AIT werden grundlegende Innovationen für die nächste Generation von Infrastrukturtechnologien in den Bereichen *Energy, Mobility Systems, Low-Emission Transport, Health & Bioresources, Digital Safety & Security, Vision, Automation & Control* und *Technology Experience* verwirklicht. Ergänzt werden diese wissenschaftlichen Forschungsgebiete um die Kompetenz im Bereich *Innovation Systems & Policy*.

Rund 1.300 MitarbeiterInnen forschen in ganz Österreich - im Besonderen an den Hauptstandorten Wien Giefinggasse, Seibersdorf, Wr. Neustadt, Tulln, Ranshofen und Graz – an der Entwicklung jener Tools, Technologien und Lösungen für Österreichs Wirtschaft, die sie gemäß unseres Grundsatzes „Tomorrow Today“ zukunftsfit halten.

Gesellschafter des AIT sind die Republik Österreich (Bundesministerium für Verkehr, Innovation und Technologie), die 50,46% der Anteile hält und der Verein zur Förderung von Forschung und Innovation (Industriellenvereinigung Österreich) mit 49,54% der Anteile.

[1] AIT Austrian Institute of Technology GmbH, Center for Digital Safety and Security, Giefinggasse 4, 1210 Wien, Österreich, rupert.schlick@ait.ac.at

doi:10.18420/SE2020_59

1.2 Center for Digital Safety & Security

Im *Center for Digital Safety & Security* werden modernste Informations- und Kommunikationstechnologien (IKT) und Systeme entwickelt, um kritische Infrastrukturen im Kontext der umfassenden und globalen Vernetzung und Digitalisierung sicher und zuverlässig zu gestalten.

Dabei fokussiert das Center auf folgende Schlüsseltechnologiebereiche: Cyber-Sicherheit für Industrial Control Systems (ICS), Cyber Physical Systems (CPS), und Internet of Things (IoT), hochsichere und hochverfügbare Software und Systeme sowie höchst zuverlässige Wireless-Kommunikation der nächsten Generation (5G), modernste Verschlüsselungsmethoden (Quantum Safe) für virtuelle IT-Systeme, Data Science für neue Ansätze eines modernen Datenmanagements (Big Data, Blockchain-Technologien), als auch neueste Sensortechnologien und Systeme zum Schutz kritischer Infrastrukturen, Command und Control Systeme für den Einsatz im modernen Krisen- und Katastrophenmanagement sowie Objektschutz kritischer Infrastrukturen und digitales Identity Management durch modernste Biometrie-Sensorik.

In enger Kooperation mit Wirtschaft und Industrie, Wissenschaft und öffentlicher Hand erfolgt eine strategische Technologieforschung sowie Entwicklung von Prototypen bis hin zur Validierung von Anwendungen in disruptiven Öko-Systemen. Das Center for Digital Safety & Security besitzt in nationalen und internationalen Innovationsprogrammen der Sicherheitsforschungsprogramme eine anerkannte Position und baut auf strategischen Partnerschaften mit den wichtigsten nationalen Sicherheitsakteuren (BMI und BMLVS) als auch in internationalen Industrieinitiativen wie beispielsweise ECSO (European Cyber Security Organisation) [2], PSCE (Public Safety Communication Europe) [3], EARTO/EUROTECH Sicherheitsgruppe (Task Force europäischer Forschungsorganisationen im Sicherheitskontext) [4], ARTEMIS/ECSEL (Europäische Technologie- und Forschungsplattform im Bereich eingebetteter und cyberphysikalischer Systeme) [5], EPoSS (Europäische Technologie- und Forschungsplattform im Bereich der Integration intelligenter Systeme) [6] und euRobotics [7] auf.

2 Themenkreise Software Engineering

Forschung im Umfeld Software Engineering findet primär in der *Competence Unit for Secure Communication Technologies* im Forschungsfeld *Dependable Systems Engineering* statt. Die hier bearbeiteten Themen fallen durchwegs in Rand- und Nischenbereiche des

[2] https://www.ecs-org.eu/
[3] http://www.psc-europe.eu/
[4] http://www.earto.eu/
[5] https://www.artemis-ju.eu/
[6] https://www.smart-systems-integration.org/public
[7] https://www.eu-robotics.net/

klassischen Software Engineering und zielen, passend zur Ausrichtung des Unternehmens, auf die Absicherung der Zuverlässigkeit und Betriebssicherheit geschäfts- und infrastrukturkritischer, softwarelastiger Systeme.

Die Arbeiten gliedern sich grob in die Themenkreise *Safety & Security Co-Engineering*, *Model-Based Testing* und *Runtime Verification und Monitoring*. Die zum Einsatz kommenden Methoden bedienen sich im Software-Prozess-Entwurf ebenso wie bei Theorien zum Software-Test, im Maschinellen Lernen und bei der Formalen Verifikation und Formalen Methoden.

2.1 Safety & Security Co-Engineering für Cyber-Physical Systems

Obwohl für Steuerungssysteme generell und für sogenannte Cyber-Physical Systems im besonderen, sowohl Aspekte der Betriebssicherheit (*Safety*, Schutz von Werten, Gesundheit, Leben und Umwelt vor Auswirkungen eines Systemversagens) als auch der Manipulationssicherheit(*Security*, Schutz des Systems, seiner Betriebsfähigkeit und der damit verbundenen Werte vor unberechtigtem Zugriff, Manipulation und Sabotage)[8] besonders wichtig sind und einige Gemeinsamkeiten aufweisen, sind die zugehörigen Forschungs-Communities nach wie vor sehr klar getrennt. Sowohl für Safety als auch für Security spielt die Berücksichtigung im System-Entwurf (Safety and Security by Design) und ihre Sicherstellung in Implementierung und Betrieb eine wichtige Rolle. Die mögliche gegenseitige Beeinflussung (z.B. im Entwurf hinzugefügte Safety-Mechanismen verändern die Angriffs-Oberfläche für Security-Betrachtungen; ein Security-Incident deaktiviert Safety-Mechanismen) legt nahe, die beiden Aspekte im Entwicklungsprozess auch ganzheitlich zu berücksichtigen. AIT ist hier in folgenden Bereichen aktiv:

Safety- und Security-Standardisierung Die Bereiche safety-kritischer Systeme, wie Eisenbahnwesen, Flugverkehr, chemische Anlagen, medizinische Systeme und Automobile, sind hochgradig reguliert, Standards spielen in dieser Regulierung eine wesentliche Rolle. AIT Mitarbeiter sind aktive Mitglieder einer Reihe einschlägiger Standardisierungsgremien in diesem Bereich. Dazu zählen z.B. ISO 26262, ISO/PAS 21448, ISO/SAE 21434, ISO 13849, IEC TR 63074, IEC 61508, IEC TR 63069 und IEC 62443-(1-4). Anzumerken ist hier, dass AIT (und seine Vorgänger-Organisationen) seit mittlerweile Jahrzehnten auf die Berücksichtigung von Security in den Safety-Standards hinwirkt.

Safety & Security Co-Analyse Um Safety & Security by Design sicher zu stellen, arbeitet AIT an Methoden und Werkzeugen sowohl zur Identifikation und Addressierung von

[8] im Folgenden wird, im Interesse einer klareren Begrifflichkeit, mit den englischen Begriffen Safety und Security gearbeitet

Bedrohungen für CPS (Tool *ThreatGet*[9]) als auch zur Identifikation und Fortpflanzung von Safety- und Security-Problemen in CPS [Sc14; SMS14].

Safety-Case Management Verschiedene Safety Standards stellen verschiedene Anforderungen an die Entwicklungs- und Qualitätssicherungs-Prozesse. Das Tool GSFlow [10] unterstützt die Verwaltung von Argumentation und Evidenz für *Safety Cases* passend zu und konfigurierbar für eine Reihe einschlägiger Standards.

2.2 Model-Based Testing

Forschungsthema ist auch die automatisierte Erstellung von Tests, typischerweise aus Modellen. Die Generierung funktionaler Tests wird hier derzeit erweitert um Tests zum Erfüllungsnachweis nicht-funktionaler Anforderungen wie Safety, Performance oder Robustheit. Ergebnis aus und Plattform für diese Forschungsaktivitäten ist die Familie von Testfallgeneratoren *MoMuT*[11]. Aktuelle Arbeiten zielen auf die Unterstützung im Modellierungsprozess und den Review von (formalen und semi-formalen) Verhaltensmodellen durch den Review daraus generierter Tests [SK19; Sn18] sowie die Integration in Modellierungsumgebungen wie *Enterprise Architect* von *SparxSystems*.

2.3 Runtime Verification

Hier werden formale Definitionen von Systemverhalten, zum Beispiel Signalverläufe, verwendet, um im Betrieb automatisch Abweichungen (z.B. zu flache Flanke, Ereignis komt zu früh / zu spät / zu oft) zu erkennen. „Im Betrieb“ bedeutet hier nicht unbedingt im Produktiveinsatz, sondern auch etwa im Bereich des End-Tests von Hardware-Komponenten. Aktuell wird an Möglichkeiten geforscht, die formalen Definitionen automatisch von einem korrekt funktionierenden System abzuleiten oder die Parameter eines Templates anzulernen. [Ni19]

3 Transferpfade

3.1 Ko-finanzierte Forschung

Der Löwenanteil der Arbeiten der Gruppe erfolgt im Rahmen nationaler und europäischer kofinanzierter Forschungsprojekte, vorwiegend in Forschungslinien, die Forschungs-Transfer

[9] https://www.threatget.com/
[10] http://gsflow.ait.ac.at/
[11] http://momut.org

adressieren, wie *IKT der Zukunft*[12] der *FFG* in Österreich und IA- und RIA-Projekte in Projekten des *ECSEL Joint Undertaking*[13] mit gemischt nationaler und EU-Förderung, aber auch *H2020 ICT*[14] und EUROSTARS[15]. Kürzlich abgeschlossene oder aktuell laufende Projekte sind zum Beispiel AQUAS, AutoDrive, Comp4Drones, IoT4CPS, TRUCONF, EMBEET, ENABLE-S3 und SEMI4.0.

3.2 Beratung und Training, Forschungsaufenthalte

Einen deutlich geringeren Teil der Aktivitäten machen Industrieaufträge für Beratung oder die Erstellung von Studien aus. Erwähnenswert ist hier eine Kooperation mit Toyota USA zu spezifikationsbasierten Verifikationsmethoden zu Entwicklungs- und Lauf-Zeit. Forschungsthemen sind hier: Spezifikationssprachen für CPS-Anwendungen (Automobil und Robotik), Sensitivitätsanalysen in Verbindung mit formalen Spezifikationen, Suchbasierte Tests, automatisierte Kausalitätsanalysen bei Ausfällen.

3.3 Lizenzierung

Die im Rahmen der Forschungsarbeiten entstandenen Werkzeuge werden teilweise selbständig oder in Kooperation mit Partnern so weit verbessert, dass sie für den industriellen Einsatz tauglich sind. Die Lizenzierung an Endkunden erfolgt dann vorzugsweise über einen Verwertungspartner. Der Transferpfad wird in der Gruppe erst jetzt, vorwiegend im Rahmen noch junger Kooperationen, genützt.

4 Beobachtungen und Konklusio

Die Erfahrung zeigt, dass man mit der Bearbeitung eines Themas selten warten kann, bis der Markt nach der Lösung für ein Problem fragt (Pull). Aber das Entwickeln von Lösungen für antizipierte Probleme (Push) bringt mit sich, dass man nicht nur inhaltlich, sondern auch zeitlich gründlich daneben liegen kann. So hebt derzeit das in der Gruppe eher junge Thema Safety- und Security Co-Analyse, speziell das Thema modellunterstützte Gefährdungsanalyse, zügig ab, während sich beim schon recht lange betriebenen Model-Based-Testing schleppend erste Lizenzerträge einstellen.

AIT ist ein etablierter und willkommener Partner in wissenschaftlichen Förderprojekten, auch im Umfeld von SE-Themen. Das hat allerdings zur Folge, dass längerfristig zu entwickelnde

[12] https://www.ffg.at/iktderzukunft
[13] https://www.ecsel.eu/
[14] https://ec.europa.eu/programmes/horizon2020/en/h2020-section/information-and-communication-technologies
[15] https://ec.europa.eu/programmes/horizon2020/en/h2020-section/information-and-communication-technologies

Themen von der österreichischen Industrie kaum beauftragt, sondern bevorzugt gemeinsam in geförderten Projekten erarbeitet werden. Die bunte Mischung an Themen, Fertigkeiten und genutzten Verwertungspfaden in der Gruppe bindet zwar Kräfte und erschwert das Erreichen kritischer Massen, bietet aber auch Sicherheit durch mehrere Standbeine in einem inhaltlich sehr bewegten Umfeld.

Literatur

[Ni19] Ničković, D.; Qin, X.; Ferrère, T.; Mateis, C.; Deshmukh, J.: Shape Expressions for Specifying and Extracting Signal Features. In (Finkbeiner, B.; Mariani, L., Hrsg.): Runtime Verification. Springer International Publishing, Cham, S. 292–309, 2019, ISBN: 978-3-030-32079-9.

[Sc14] Schmittner, C.; Gruber, T.; Puschner, P.; Schoitsch, E.: Security Application of Failure Mode and Effect Analysis (FMEA). In (Bondavalli, A.; Di Giandomenico, F., Hrsg.): Computer Safety, Reliability, and Security. Springer International Publishing, Cham, S. 310–325, 2014, ISBN: 978-3-319-10506-2.

[SK19] Schlick, R.; Krenn, W.: Tackling the Challenges of Internet-of-Things-Development Using Models. In. DATE 19, 2nd International ESIIT Workshop. Florence, Italy, S. 40–41, März 2019, URL: http://adt.cs.upb.de/ESIIT2019-tproceedings.pdf.

[SMS14] Schmittner, C.; Ma, Z.; Smith, P.: FMVEA for Safety and Security Analysis of Intelligent and Cooperative Vehicles. In (Bondavalli, A.; Ceccarelli, A.; Ortmeier, F., Hrsg.): Computer Safety, Reliability, and Security. Springer International Publishing, Cham, S. 282–288, 2014, ISBN: 978-3-319-10557-4.

[Sn18] Snook, C. F.; Hoang, T. S.; Dghaym, D.; Butler, M. J.; Fischer, T.; Schlick, R.; Wang, K.: Behaviour-Driven Formal Model Development. In: Formal Methods and Software Engineering - 20th International Conference on Formal Engineering Methods, ICFEM 2018, Gold Coast, QLD, Australia, November 12-16, 2018, Proceedings. S. 21–36, 2018, URL: https://doi.org/10.1007/978-3-030-02450-5%5C_2.

Christian Doppler Laboratory on Security and Quality Improvement in the Production Systems Life Cycle[1]

Dietmar Winkler[2], Stefan Biffl[3]

Abstract: The size and complexity of software components in production systems engineering, such as manufacturing plants or automation systems, requires effective and efficient approaches for security and quality improvement. In industrial practice, engineers from different disciplines, such as electrical, mechanical, and software disciplines typically follow a plan-driven and sequential engineering process approach with parallel engineering activities within a heterogeneous set of methods and tools. Therefore, major challenges concern (a) insufficient data exchange capabilities between disciplines, (b) a lack of consistency evaluation capabilities across disciplines, tools, and engineering phases, (c) insufficient knowledge representation and exchange between disciplines and project stakeholders and (d) limited security considerations. The goal of the Christian Doppler Laboratory on Security and Quality Improvement in the Production Systems Life Cycle (CDL-SQI) is to address these challenges in cooperation with industry partners in the production systems domain. We build on requirements and use case explorations at industry partners and on best-practices from Business Informatics to develop concepts and prototype solutions for the target domain and evaluate these concepts and prototypes in close collaboration with industry partners. We derive requirements, use cases, and test data from industry and provide concepts and prototypes to the industry partner and to related scientific communities.

Keywords: Production Systems Engineering; Software and Systems Engineering; Security; Quality; Engineering Process Improvement; Testing; Variability Management; Software Models

1 Introduction of the CDL-SQI research project

In this presentation we will introduce to the objective and the organization of the CDL-SQI research lab at TU Wien, focus on important challenges in the production systems automation domain and on selected key use cases and prototype solutions from / for involved industry partners [Bi19a]. The research laboratory, located at TU Wien, Austria started in 2018 and is organized in three modules, each supported by an industry partner in the production systems automation domain or related fields. The main focus of the seven-year research project, funded by the Christian Doppler Association[4], is related to basic research in collaboration with industry partners. The mission of the CDL-SQI research lab is to both explore and develop concepts which improve the security and quality in cyber-physical production system lifecycles, where software engineering provides important added value,

[1] CDL-SQI: `www.sqi.at`

[2] CDL-SQI, TU Wien, 1040 Vienna, Austria, dietmar.winkler@tuwien.ac.at
[3] Institute of Information Systems Engineering, TU Wien, 1040 Vienna, Austria, Stefan.biffl@tuwien.ac.at

[4] Christian Doppler Association: `www.cdg.ac.at`

doi:10.18420/SE2020_60

but suffers from heterogeneous requirements and data inputs from a multitude of systems engineering disciplines. We will discuss success and risk factors from the interaction of software engineers and systems engineers.

The research lab is organized in three modules each related to industry partners: (a) *Test Automation* and human-in-the loop testing in the production systems domain (Module 1) [Ec19; WMB18], (b) concepts, mechanism, and prototypes for an *efficient and effective data logistics* concept based on *AutomationML*[5], a standardized data exchange format based on XML (Module 2) in the steel mill production environment (Module 2) [Bi19b], and (c) on *knowledge representation and reuse* in context of a *Product-Process-Resource (PPR)* concept [Sc15] in manufacturing system development.

2 Collaboration between Industry Partners and Academia

The research work follows the *design science cycle* to (a) explore industry partner needs, (b) identify candidate solutions concepts, (c) develop concepts and prototypes, and (d) evaluate prototype solution in academic and industrial contexts. Therefore, regular and individual workshops are organized to discuss and exchange needs, concepts, and prototypes. Industry partners typically provide application use cases, needs, and test data in their individual context and receive concepts, prototypes, and evaluation results. Furthermore, scientific communities can benefit from publications in context of industry use cases and application areas as well as from prototype evaluations in industry settings.

References

[Bi19a] Biffl, S.; Eckhart, M.; Lüder, A.; Weippl, E., eds.: Security and Quality in Cyber-Physical Systems Engineering. Springer, 2019.

[Bi19b] Biffl, S.; Lüder, A.; Rinker, F.; Waltersdorfer, L.; Winkler, D.: Engineering Data Logistics for Agile Automation Systems Engineering. In (Biffl, S.; Eckhart, M.; Lüder, A.; Weippl, E., eds.): Security and Quality in Cyber-Physical Systems Engineering. Springer International Publishing, pp. 187–225, 2019.

[Ec19] Eckhart, M.; Meixner, K.; Winkler, D.; Ekelhart, A.: Securing the testing process for industrial automation software. Computers & Security 85/, pp. 156–180, 2019, ISSN: 0167-4048.

[Sc15] Schleipen, M.; Lüder, A.; Sauer, O.; Flatt, H.; Jasperneite, J.: Requirements and concept for Plug-and-Work. at - Automatisierungstechnik 63/, pp. 801–820, Oct. 2015.

[5] AutomationML: `www.automationml.org`

[WMB18] Winkler, D.; Meixner, K.; Biffl, S.: Towards Flexible and Automated Testing in Production Systems Engineering Projects. In: Proceedings ETFA. Vol. 1, pp. 169–176, Sept. 2018.

Software Competence Center Hagenberg: Wissens- und Technologietransfer in Software und Data Science

Rudolf Ramler[1], Thomas Ziebermayr[1]

Abstract: Das Software Competence Center Hagenberg (SCCH) ist ein außeruniversitäres Forschungszentrum, das seit 20 Jahren umfassende Expertise in der anwendungsorientierten Forschung in den Bereichen Software Science und Data Science aufgebaut hat. Das SCCH fungiert als Schnittstelle zwischen internationaler Forschung und der regionalen Industrie und Wirtschaft. Die Organisation als eigenständiges Zentrum ermöglicht langfristige Forschungskooperationen mit Unternehmen und wissenschaftlichen Partnern im Rahmen von großen, mehrjährigen Projekten in den Bereichen Digitalisierung, Industrie 4.0 und Künstliche Intelligenz.

Keywords: Wissenstransfer, Technologietransfer, Software Science, Data Science

1 Einleitung

Das Software Competence Center Hagenberg (SCCH) ist ein außeruniversitäres Forschungszentrum mit Fokus auf Themen in den Bereichen Software Science und Data Science. Das SCCH wurde im Jahr 1999 als COMET-Kompetenzzentrum[2] gegründet. Das Ziel des Zentrums ist die Kooperation zwischen Industrie und Wissenschaft zu stärken und der Aufbau gemeinsamer Forschungskompetenzen und deren wissenschaftliche und wirtschaftliche Verwertung zu forcieren.

2 Modell für Wissens- und Technologietransfer

Das SCCH versteht sich als Bindeglied zwischen der *wissenschaftlichen Forschung* auf der einen Seite und der *anwendungsorientierten Umsetzung in der Industrie und Wirtschaft* auf der anderen Seite (Abb. 1). Das SCCH arbeitet dazu einerseits mit internationalen wissenschaftlichen Partnern und andererseits mit Unternehmen mit regionalem Schwerpunkt zusammen. Der Transfer von wissenschaftlichen Ergebnissen in die Praxis (in innovative Produkte und Dienstleistungen) erfordert ein umfassendes Verständnis, sowohl der

[1] Software Competence Center Hagenberg GmbH, Softwarepark 21, A-4232 Hagenberg, Austria, https://www.scch.at, {rudolf.ramler,thomas.ziebermayr}@scch.at

[2] Das COMET-Zentrum Software Competence Center Hagenberg (SCCH) wird im Rahmen von COMET - Competence Centers for Excellent Technologies durch BMVIT, BMDW und Land Oberösterreich gefördert. Das Programm COMET wird durch die Forschungsförderungsgesellschaft FFG abgewickelt.

doi:10.18420/SE2020_61

Forschungsdisziplin als auch der Anwendungsdomäne, und darüber hinaus Umsetzungs- und Projektkompetenz. In den Projekten am SCCH arbeiten daher Mitarbeiter/-innen mit universitärer Ausbildung und langjähriger Berufserfahrung.

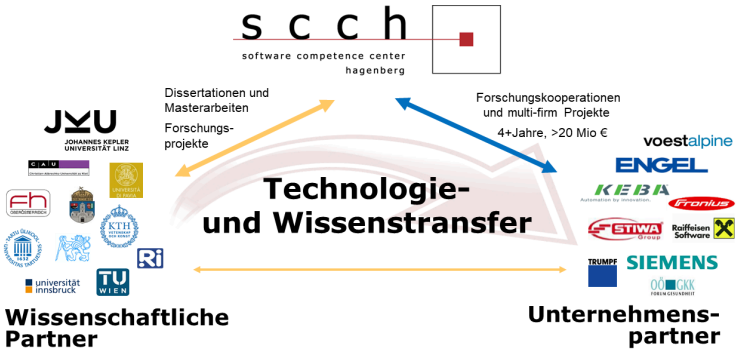


Abb. 1: Technologie- und Wissenstransfer am SCCH

Die Organisation als eigenständiges, unabhängiges Zentrum ermöglicht langfristige Forschungskooperationen mit wissenschaftlichen Partnern zu Grundlagenthemen und die Umsetzung *mehrjähriger Forschungsprojekte* (i.d.R. über einen Zeitraum von 4 Jahren), in denen *mehrere Partner aus der Industrie und Wirtschaft gemeinsam* eingebunden sind. Die Forschungsthemen können so stärker generalisiert und in größerer Breite und Tiefe bearbeitet werden, als dies in Einzelprojekten mit Unternehmen möglich wäre. In Forschungskooperationen gemeinsam mit wissenschaftlichen Partnern beteiligt sich bzw. koordiniert das SCCH nationale und internationale Forschungsprojekte (z.B. H2020) und unterstützt Dissertationen sowie Masterarbeiten. Den Rahmen für anwendungsorientierte Projekte bildet in erster Linie das COMET-Programm des Zentrums. Die Mitarbeiter der akademischen Seite und der Unternehmen sind in Forschungsprojekten aktiv eingebunden, um den direkten Wissensaustausch in beide Richtungen zu fördern.

3 Software Science und Data Science als Erfolgsfaktor

Seit der Gründung des SCCH vor 20 Jahren hat die Bedeutung von *Software und Data Science* enorm zugenommen. Die rasante Entwicklung von Artificial Intelligence (*AI*) im Zusammenhang mit industriellen Anwendungen (Industrie 4.0) und cyber-physischen Systemen hat die beiden Bereiche weiter zusammenwachsen lassen. Viele klassische Anwendungsgebiete im Software Engineering (*SE*) – z.B. die Analyse von Softwaresystemen, die Qualitätssicherung und das Testen – profitieren von dieser Entwicklung und nutzen Künstliche Intelligenz als Unterstützung (*AI für SE*). Zugleich entsteht derzeit durch den Einsatz von Künstlicher Intelligenz in kurzer Zeit eine Vielzahl von neuen Anwendungen,

die durch die Art der Entwicklung und Anpassbarkeit auch neue Herausforderungen für die Qualitätssicherung, Wartung und Weiterentwicklung mit sich bringen (*SE für AI*). Die Verknüpfung von Software Science und Data Science unter einem Dach ist daher heute ein wesentlicher Erfolgsfaktor für das SCCH.

Hamburger Informatik Technology Center e.V. - Technology Transfer at the Department of Informatics of the University of Hamburg

Lothar Hotz[1] Rainer Herzog[2] Stephanie von Riegen[3]

Abstract: This position paper provides an overview of the organisation and activities of the Hamburger Informatik Technology Center e.V. (HITeC). The goal of HITeC is to transfer state-of-the-art scientific results, achieved at the Department of Informatics of the University of Hamburg, to companies, the public administration, and other research institutes.

Keywords: Technology Transfer; Applied Research; Artificial Intelligence

1 Profile

HITeC is the research and technology transfer center of the Department of Informatics at the University of Hamburg and offers the competence of leading computer scientists of the department. Due to HITeC's independent status and professional organization, the cooperation between HITeC and the industry as well as other research groups is flexible and goal-oriented. HITeC is a registered non-profit association, and its members are leading researchers from academia.

HITeC's activities are always research- or education-based. They include research cooperation, consulting and expert studies, subcontractual research and development, as well as administrative project support. The goals include strengthening contacts between companies and the Department of Informatics at the University of Hamburg, dissemination, seminars and workshops. Further activities include or support courses offered jointly by the university and companies, contact fairs for recruiting, seminars for career choice, and spin-off support.

Topics in which HITeC demonstrates excellent expertise, based on experiences from concrete projects are (among others): • Object-oriented Software Development Framework and Component Technology; • Modelling and Simulation of **Logistic Systems**; • Software **Ergonomics and Usability**; • **Intelligent Systems** in Technical Applications including Machine Learning, Computer Vision, and Natural Language Processing; • **Knowledge Management**, Semantic Web, and Multimedia Information Processing; • Information

[1] Hamburger Informatik Technology Center e.V. (HITeC), Universität Hamburg, Vogt-Kölln-Str. 30, 22527 Hamburg, Germany hotz@informatik.uni-hamburg.de
[2] HITeC,herzog@informatik.uni-hamburg.de
[3] HITeC,svriegen@informatik.uni-hamburg.de

doi:10.18420/SE2020_62

Systems for **Enterprise Ecology**; • **Mobile and Location Aware Computing**; • **E-Business Concepts and Technologies**; • **e-Government**.

2 Collaborations

The liaison between HITeC and the Department of Informatics allows for a great variety of collaborations between companies and HITeC: • Advising student theses, • Performing research and providing advice for relevant research questions • Cooperative projects for conceptualisation and prototypical realisation of software technologies and applications. Companies interested in continued cooperation with HITeC may become sustaining members. A list of cooperation partners can be found on our web site[4].

Within EU projects, HITeC processes research activities with international partners. Examples are: • **OPENREQ – Intelligent Recommendation Decision Technologies for Community-Driven Requirements Engineering (EU Horizon 2020 Project 732463, 2017-2019)**, which develops novel context-aware requirements engineering approaches and tools especially for large and distributed software projects; • **RACE – Robustness by Autonomous Competence Enhancement (EU FP7-Project 287752, 2011 - 2014)**, whose aim was to develop a robot which learns from experiences; • **Co-Friend – Cognitive & Flexible learning system operating Robust Interpretation of Extended real scenes by multi-sensors Data fusion (EU FP7-Project 214975, 2008 - 2011)**, whose aim was to develop a vision system for monitoring aircraft service activities at Toulouse Airport. As a result of such projects, in cooperation with the University of Hamburg, HITeC developed an interpretation system that combines low-level object recognition with high-level aggregations of scene interpretations.

3 Experiences and Latest Developments

A main need for effective technology transfer which is lived at HITeC is the dynamic, fast, and responsive interaction with existing and potential project partners, i.e., on • **the economic processing**: quotes, contracts, invoices; • **the technological processing**: selection of appropriate technologies and researchers, performing problem adequate conceptual and experimental research; • **the dissemination processing**: understandable presentation and communication of project results, transferring project results to other projects as well as back to research and education. Furthermore, the physical proximity of persons performing technology transfer and researchers of the university enables the needed dynamics. Through the years (HITeC was founded 1997), a sustainable pool of researchers, processing projects as well as deep know-how in areas such as software development, computer vision, and knowledge-based systems adds fundamental experiences to running and upcoming projects. As a recent development, HITeC is part of the **Artificial Intelligence Center Hamburg**

[4] https://www.hitec-hamburg.de

(ARIC)[5]). ARIC operates as a single point of contact in the region of Hamburg for demands related to Artificial Intelligence of any kind, economically, societally, scientifically. Doing so, it performs networking and project management and, hence, acts as a marketing instrument for HITeC's transfer activities in the field of Artificial Intelligence.

[5] https://www.aric-hamburg.de

Software Engineering am FZI Forschungszentrum Informatik

Von Prozessen zu Architekturen

Jörg Henß,[1] Oliver Denninger[1]

Abstract: Im Folgenden geben wir einen kurzen Überblick über das Profil und die Forschungs- und Transferschwerpunkte des Forschungsbereichs Software Engineering am FZI. Neben ausgewählten Schwerpunkten im Bereich Prozess- und Software-Architektur-Analysen geben wir einen Ausblick auf zukünftige Themen.

Keywords: Transfer-Institut; Profil; Software Engineering

1 Software Engineering am FZI

Das FZI ist eine unabhängige gemeinnützige Stiftung für Informatik-Anwendungsforschung und Technologietransfer. Der Technologietransfer erfolgt schwerpunktmäßig im Bereich kleiner und mittlerer Unternehmen (KMU) durch gemeinsame Forschungs- und Entwicklungsprojekte. Als Innovationspartner des Karlsruher Instituts für Technologie bringt das FZI neuste wissenschaftliche Erkenntnisse in Unternehmen und öffentliche Einrichtungen. Der Bereich Software Engineering des FZI hat seinen Schwerpunkt in der Forschung und Entwicklung von Methoden und Werkzeugen aus dem Bereich Software und digitale Prozesse. Im besonderen Fokus liegen dabei die Analyse und Vorhersage der Qualität von Software sowie die prozess- und nutzerorientierte Gestaltung von Softwaresystemen. Darüber hinaus wird auch an Basistechnologien wie etwa der modellgetriebenen Software-Entwicklung geforscht.

2 Forschungs- und Transferschwerpunkte

Das FZI hat seinen Transferschwerpunkt im Bereich der KMU. In diesem Umfeld finden sich häufig eine Vielzahl von extern entwickelten Bestandssystemen, wobei der Digitalisierungsgrad von Geschäftsprozessen insgesamt häufig noch gering ist. Entsprechend adressieren Beratungsprojekte oft die Frage, wie sich die Bestandssysteme in eine moderne IT-Infrastruktur einbetten lassen und wie die Prozesse und Systeme modernisiert werden

[1] FZI Forschungszentrum Informatik, Haid-und-Neu-Str. 10-14, 76131 Karlsruhe {henss,denninger}@fzi.de

doi:10.18420/SE2020_63

können. Darüber hinaus müssen oft auch die Software-Entwicklungsprozesse modernisiert werden, um schneller auf sich ändernde Anforderungen reagieren zu können.

2.1 Prozessanalysen und -automatisierung

Ein elementarer Bestandteil der Anforderungserhebung für eine modernisierte IT-Infrastruktur ist die (werkzeuggestützte) Erhebung und Bewertung der Prozesse im Unternehmen und ihres Digitalisierungsgrads bzw. -potentials. Auf dieser Basis entwirft das FZI mit Methoden wie Blueprinting von Prozessen und Architekturen iterativ eine Soll-Infrastruktur. Durch leichtgewichtige Techniken wie Robotic-Process-Automation werden schnell Automatisierungspotentiale realisiert.

2.2 Architekturbasierte Software-Analysen

Im Bereich der architekturbasierten Analysen beschäftigt sich das FZI mit der Fragestellung, wie schon zur Entwurfszeit die Qualität eines Software-Systems vorhergesagt werden kann (vgl. Palladio[2]). Auf diese Weise können Entwurfsentscheidungen systematisch getroffen und Faktoren wie die Performance, Skalierbarkeit und Wartbarkeit eines Systems mit einbezogen werden. Darüber hinaus entwickeln und nutzen wir Werkzeuge zur Extraktion, Erhebung und Bewertung der Architekturen von Legacy-Softwaresystemen. Die gewonnenen Informationen bilden die Grundlage für eine nachfolgende systematische Software-Renovierung.

2.3 Modellgetriebenen Technologien

Im Bereich der modellgetriebenen Technologien entwickelt das FZI aktuell Methoden zur Konsistenzhaltung von Engineering-Modellen, z. B. im Bereich des Anlagenbaus oder in der Bauwirtschaft. Hierbei kommt unter anderem der VITRUVIUS-Ansatz[3] zum Einsatz, der es erlaubt, verschiedene Sichten auf ein Modell über ein virtuelles Kernmodell zu vereinen. Weiterhin entwickelt und setzt das FZI verschiedene generative Technologien für den beschleunigten Entwurf von Software-Systemen ein.

3 Ausblick

Durch den momentan starken Boom an KI-basierten Systemen stellt das FZI sich die Frage, wie sich solche lernende und somit sich verändernde Komponenten langfristig in eine existierende IT-Landschaft integrieren lassen. Hierbei kommen Themen aus dem Gebiet Software-Entwicklungsprozesse, Software-Analysen und Software-Evolution besonders zum Tragen.

[2] https://palladio-simulator.com
[3] http://vitruv.tools

Center for Code Excellence

Johannes Kroß[1], Peter Bludau[1], Alexander Pretschner[1,2]

Abstract: Das Center for Code Excellence (CCE) richtet sich sowohl an Softwareentwickler als auch Projekt- und Unternehmensmanager. Wir haben das Ziel, kleine und mittelständische Unternehmen in die Lage zu versetzen, herausragende, nachhaltige und zukunftsweisende Software zu entwickeln. Dieser Beitrag stellt die Motivation und Ansätze des CCE kurz vor sowie die drei Schwerpunkte - Bewertung der Softwaretechnik, Trendanalysen und Erforschung von Lehrmethoden.

1 Motivation und Bedarf

Software Engineering ist die Schlüsseldisziplin in der Digitalisierung. Diese Kompetenz wird für einen Großteil der Unternehmen fortlaufend wichtiger. Auch wenn nicht selbst Software entwickelt wird, benötigt ein Unternehmen tiefes Wissen in der Auswahl und dem Einsatz von Technologien, um Potentiale zu nutzen.

Das notwendige Wissen für Softwareentwicklung wird jedoch kontinuierlich breiter und komplexer. Softwareentwickler, aber auch Führungskräfte, müssen in der Lage sein einzuschätzen, was sich hinter aktuellen Trends verbirgt. Damit ein Unternehmen zukunftsfähig ist, stellen die Beherrschung (1) der Softwaretechnik methodisch und technisch, (2) der Projektorganisation und des Managements, (3) der Software Governance und (4) des Innovationsmanagements unverzichtbare Kompetenzen dar [Br15]. Die Bewertung der eigenen Kompetenzen und der gezielte Aufbau von Breitenwissen innerhalb eines Unternehmens ist daher unumgänglich. Wie genau dieses allerdings hergestellt und vermittelt werden kann, ist oft unklar.

2 Ansatz und Transferaktivitäten

Das Center for Code Excellence (CCE, `https://cce.fortiss.org`) bildet eine unterstützende Anlaufstelle für kleine und mittlere bayerische Unternehmen und richtet sich dabei sowohl an Softwareentwickler als auch Manager. Langfristig ist unser oberstes Ziel, Unternehmen und die dort tätigen Softwareentwickler in die Lage zu versetzen, herausragende, nachhaltige

[1] fortiss, Landesforschungsinstitut des Freistaats Bayern, Guerickestraße 25, 80805 München, Deutschland
{kross|bludau|pretschner}@fortiss.org
[2] Technische Universität München, Fakultät für Informatik, Boltzmannstraße 3, 85748 Garching, Deutschland
alexander.pretschner@tum.de

doi:10.18420/SE2020_64

und zukunftsweisende Software – Code Excellence – zu entwickeln. Um dies zu erreichen, verfolgt das CCE drei Absichten, die nachfolgend vorgestellt werden.

2.1 Bewertung von Softwareentwicklungsaktivitäten

Kernforschungsthema ist die Bewertung von Code Excellence von Unternehmen und Softwareentwicklern (`https://cce.fortiss.org/check`). Code Excellence kann dabei als eine Gesamtheit der nötigen Kompetenzen, die ein Unternehmen/Entwickler besitzen sollte, um bestimmte Aufgaben der Softwareentwicklung zu bewältigen, verstanden werden. Um die Kompetenzen eines Unternehmens gezielt zu erhöhen, erheben wir Entwicklungsaktivitäten, analysieren Potentiale und identifizieren Maßnahmen zur Verbesserung der Unternehmensmethodik.

2.2 Kontinuierliche Trendanalysen

Damit Chancen und Risiken digitaler Technologien richtig eingeschätzt werden können, erheben und visualisieren wir automatisiert Technologietrends (`https://cce.fortiss.org/trends`) und analysieren deren mögliche Auswirkungen auf sich verändernde Kompetenzen. Für entscheidende Entwicklungen ist es unser Ziel, Handlungsweisungen zu entwickeln, wie zum Beispiel die Fortbildung von Mitarbeitern zur Erlernung neuer Technologien. Ein frühzeitiges Erkennen sowie eine frühzeitige Adaption kann entscheidend für den Erfolg eines Unternehmens sein.

2.3 Evaluierung von Lehrmethoden und Vermittlung von Kompetenzen

Zuletzt ist es unser Ziel Lehrmethoden zu entwickeln, die es ermöglichen, Wissen und Kompetenz über die Lösung von komplexen Sachverhalten zu vermitteln. Die zunehmende Bedeutung des Software Engineerings stellt gerade für den Transfer eine große Herausforderung dar, die wir in diesem Forschungsvorhaben systematisch behandeln.

3 Danksagungen

Das CCE wird durch das Bayerische Staatsministerium für Wirtschaft und Medien, Energie und Technologie gefördert.

Literatur

[Br15] Broy, M.: Software Eats The World - Zehn Thesen zur strategischen Bedeutung von Software für Wirtschaftsunternehmen. Swiss Engineering Institute Press, 2015.

Software Engineering im Unterricht der Hochschulen

Software Engineering im Unterricht der Hochschulen 2020

Stephan Krusche[1] Stefan Wagner[2]

Die SEUH ist seit vielen Jahren das Forum im deutschsprachigen Raum, auf dem Lehrende aus Universitäten, Hochschulen für angewandte Wissenschaften sowie dualen Hochschulen ihre Erfolge, Misserfolge und Erfahrungen in der Software Engineering Ausbildung vorstellen, diskutieren und gemeinsam die Qualität der Lehre verbessern. Der Workshop bietet viel Raum für Diskussionen, sowohl während der Sitzungen als auch in den Pausen. Kurze Vorträge mit anschließenden Diskussionsphasen fördern den Austausch der Teilnehmer. Dies haben die Teilnehmer in der Vergangenheit ausgiebig in lebhaften und konstruktiven Gesprächen genutzt. Viele Lehrende haben von der SEUH entscheidende Impulse für ihre Arbeit erhalten.

Thematische Schwerpunkte

Der Tradition folgend, schafft auch die diesjährige Tagung einen Rahmen für die Diskussion zu aktuellen Fragen der Softwaretechnik im Kontext der Hochschullehre. Heinz Züllighoven eröffnet den Workshop mit einer Keynote zum Thema “SEUH und die Erfahrungen eines Softwarehauses”.

Das Programmkomitee wählte aus 22 Einreichungen 15 Beiträge aus, die sich überwiegend folgenden Themenschwerpunkten zuordnen lassen:

- Die kritische Auseinandersetzung mit Modellen und Methoden
- Teambasierte Projektkurse in Kooperation mit der Industrie
- Agile Vorgehensweisen in der Lehre und in Ausbildungsprojekten
- Interaktive und aktivierende Lehrkonzepte für große Kurse
- Werkzeuge zur Lehr- und Lernunterstützung
- Neue Methoden und didaktische Ansätze zur Vermittlung von Fähigkeiten
- Digitale Bildung, Online Kurse und Video basierte Ausbildung
- Automatische Bewertung von Leistungen

[1] Technische Universität München, Deutschland, krusche@in.tum.de
[2] Universität Stuttgart, Deutschland, stefan.wagner@iste.uni-stuttgart.de

doi:10.18420/SE2020_65

Wir hoffen, dass die ausgewählten Beiträge die Grundlage für anregende Diskussionen bilden. Alle Beiträge sind auf `http://ceur-ws.org/Vol-2531` veröffentlicht. Wir danken dem Programmkomitee für die ausführlichen und hilfreichen Gutachten und den Autoren für die interessanten Beiträge.

Programmkomitee

- Jürgen Börstler, Blekinge Institute of Technology
- Ruth Breu, Universität Innsbruck
- Marian Daun, Universität Duisburg-Essen
- Birgit Demuth, Technische Universität Dresden
- Maritta Heisel, Universität Duisburg-Essen
- Elke Hochmüller, Fachhochschule Kärnten
- Carsten Kleiner, Hochschule Hannover
- Marco Kuhrmann, Universität Passau
- Dieter Landes, Hochschule Coburg
- Jürgen Münch, Hochschule Reutlingen
- Barbara Paech, Universität Heidelberg
- Axel Schmolitzky, HAW Hamburg
- Juliane Siegeris, Hochschule für Technik und Wirtschaft Berlin
- Gudrun Socher, Hochschule München
- Matthias Tichy, Universität Ulm
- Nadine von Frankenberg, Technische Universität München

Workshops

2nd Workshop on Avionics Systems and Software Engineering

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

Abstract: Companies are struggling with the complexity of avionics systems. A lot of effort is required for the development of such systems. As appropriate tools and methods are supposed to be an effective lever, there is a high demand for increasing their efficiency. The AvioSE workshop continues to be a forum for people working on increasing the efficiency for the development of avionics systems.

Keywords: avionics; systems engineering; software engineering; formal methods; model-based; requirement; qualification; certification; simulation; process; tool

1 Introduction

Aerospace applications depend heavily on software and hardware, but complexity, both safety and security demands, and regulations make their development ambitious. Research progress in development efficiency can be observed to be of uttermost significance. AvioSE'19[4] demonstrated successful exchange and collaboration in technological applications and in methodological approaches. For both areas there is still a large gap between the provision of research results and their wide industrial adoption. Fostering the cooperation between industry and research is the main objective for achieving significant technological progress and enable enhancements in the development process.

In addition, AvioSE'20 addresses **tools and their usage in aerospace**. The tools' underlying concepts, e. g. textual, model-based; the process, e. g. V-model, agile; the tool implementation, e. g. qualified, proprietary, in-house development, open source; and the tool ecosystem, e. g. manual conversion, seamless-tool chain, one-tool-for-all, differ. It shall be figured out with the participants, if there is a most promising approach for the usage of tools and how tools for new methods must look like to gain most benefit in the avionics domain.

[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
[4] Annighoefer et al., 1st Workshop on Avionics Systems and Software Engineering (AvioSE'19), 2019. Annighoefer et al., Challenges and Ways Forward for Avionics Platforms and their Development in 2019, in IEEE/AIAA 38th Digital Avionics Systems Conference (DASC), 2019.

doi:10.18420/SE2020_66

2 Workshop Objectives

The main objective of the workshop is to accelerate the transfer of knowledge between academia and industry. This workshop provides the enabling platform for these stakeholders to discuss technical, but also process, and educational topics.

The objectives of AvioSE'20 are three-fold: (1) It provides a forum for researchers from both academia and industry to present new methods, tools, and technologies from avionics systems and software engineering, e. g. model-based development, requirements engineering, formal methods, model-based methods, and virtual methods. Those contributions are presented in a scientific format, but the small character of the workshop allows detailed discussion. (2) **Tools and their usage in avionics** are selected to be the main topic of AvioSE'20. This is addressed interactively by inviting all participants to discuss aspects of the tools topic, i. e. tool properties, tool qualification, tool integration into the development process, tool implementation, and tool ecosystems. The objective is to bring people together in small breakout groups. This covers connecting academics and professionals with experts. Each breakout group shall figure out the most important issues of their aspect and propose ways how to address them. The results are made available to all participants with the presentations of the breakout groups' conclusions. (3) The AvioSE'20 also allows for a wild card topic that might show up during the workshop.

Acknowledgements

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, FH-Prof. Dipl.-Ing. Dr. Holger Flühr, FH JOANNEUM Gesellschaft mbH, and Detlef Schiron, Airbus Defence and Space GmbH. Second, we want to express our gratitude to the SE 2020 organizers for supporting our workshop and having the workshop being hosted. Additionally, we are glad that these people (listed in alphabetic order) served as members in the program committee, soliciting papers and writing peer reviews: Prof. María Angeles Martín Prats (Universidad de Sevilla, Spain), Jun.-Prof. Björn Annighöfer (University of Stuttgart), Prof. Dr.-Ing. Steffen Becker (University of Stuttgart), Umut Durak (DLR Braunschweig), Prof. Dr. Ralf God (Hamburg University of Technology), Prof. Dr. Lars Grunske (Humboldt-Universität zu Berlin), Prof. Dr. Eric Knauss (University of Gothenburg), Jürgen Krug (Diehl Aerospace GmbH), Dr. Winfried Lohmiller (Airbus Defence and Space GmbH), Dr. Christian Meißner (Volkswagen AG), Prof. Dr. Alexander Pretschner (Technical University of Munich), Dr. Stephan Rudolph (Northrop Grumman LITEF GmbH), Prof. Dr. Bernhard Rumpe (RWTH Aachen University), Dr. Andreas Schweiger (Airbus Defence and Space GmbH), Katja Stecklina (Philotech Systementwicklung und Software GmbH), and Prof. Dr. Matthias Tichy (Ulm University).

Finally, sincere thanks are given to the authors' management for welcoming and supporting the workshop.

7th Collaborative Workshop on Evolution and Maintenance of Long-Living Systems

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

Die Digitalisierung stellt neue Herausforderungen an die Entwicklung und den Betrieb von Software. Die **Digitalisierung** sozialer, politischer, wissenschaftlicher und ökonomischer Prozesse führt zu gesellschaftlichen Transformationen und verändert die Umgebung, die Nutzung und die Entwicklung von Softwaresystemen. Systeme sollen den wandelnden Bedürfnissen folgen aber dennoch den Qualitätsansprüchen der Nutzer genügen. Die Dynamik und der Umfang von Digitalisierungsvorhaben erfordert daher im steigenden Maße Software, die während ihrer Laufzeit ihren Betrieb sicherstellt. Gleichzeitig aber sollen maschinengestützte Entscheidungsprozesse für den Menschen nachvollziehbar und transparent abgewickelt werden. Konkrete Herausforderungen umfassen daher unter anderem die **Verzahnung der Entwicklungsschritte**, die **Erklärbarkeit** von Software und den zugrunde liegenden Entscheidungen, neue Analyseansätze und -methoden für ein besseres Systemverständnis, Konsistenz der Artefakte, sowie die **Evolution** von Plattformen und Frameworks. Diese sind zentrale Herausforderungen für langlebige softwareintensive Systeme.

Ziel des EMLS-Workshops ist es, diese Herausforderungen gemeinsam aus Wissenschaft und Industrie zu beleuchten und unterschiedliche Sichtweisen zur Evolution und Wartung langlebiger Systeme zusammenzubringen.

Der Workshop nutzt, wo möglich, die thematische Arbeit in Kleingruppen um den Austausch zwischen den Teilnehmern zu fördern. Zu Beginn jeder Session stellen die Autorinnen und Autoren ihre Themen in kurzen Impulsvorträgen vor. Anschließend werden zu diesen Themen Kleingruppen gebildet. Für jede Gruppe stellen die Organisatoren eine Moderatorin oder Moderator zur Leitung der Diskussion. Für die Gruppen stehen Diskussionsmaterialien, wie Flipchart, Moderationstafel und Diskussionskarten bereit, die zur Steuerung der Diskussion und zur Vorbereitung der Ergebnispräsentation genutzt werden können. Am Ende der Session werden die Ergebnisse der Diskussionen in den Kleingruppen zusammengetragen und abschließend 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

[1] Kiel University, Germany, reiner.jung@email.uni-kiel.de
[2] University of Koblenz-Landau, Germany, konersmann@uni-koblenz.de
[3] Information und Technik Nordrhein-Westfalen - IT.NRW, Germany, Eric.Schmieders@it.nrw.de

doi:10.18420/SE2020_67

veröffentlicht. Zudem werden die erarbeiteten Präsentationen fotografiert und auf der Webseite http://rgse.uni-koblenz.de/emls/ bereitgehalten. Der Workshop wird in diesem Jahr mit einer Keynote von Dr. Eric Schmieders (IT.NRW) eröffnet. IT.NRW ist der IT-Dienstleister des Landes Nordrhein-Westfalen und verantwortlich für die Planung und Steuerung der Digitalisierung des Bundeslandes sowie die Entwicklung und den Betrieb von landeseigenen IT-Dienstleistungen. Die Keynote addressiert die *Digitalisierung in der öffentlichen Verwaltung - IT-Architekturmanagement und Geschäftsprozessoptimierung*. Die einzelnen Themen für die Diskussionsrunden lagen zum Zeitpunkt Drucklegung noch nicht vor werden jedoch auf der Workshop Webseite veröffentlicht.

Requirement Management in Enterprise Systems Projects

Christoph Weiss,[1] Johannes Keckeis[2]

1 Motivation

ERP systems and other enterprise systems are the backbone of any company in a digitized world. In almost every company Enterprise Systems are adapted to the needs of the customers within the scope of parameterization, modifications (changes to existing functions and logics) or even extensions (new developments of existing functions and logics). However, many of such Enterprise Systems projects fail due to missing, incorrect, inadequate or incomplete requirements there are "incorrect" expectations, divergents in definition and attitudes on requirements management between customers and suppliers. These challenges will be highlighted, talked over and discussed during this workshop.

2 Expected results (work objectives) of the workshop

Presentation of points of view, ways of thinking, processes and perceptions of Enterprise Systems customers or prospective customers and providers in selection, implementation and further development of Enterprise Systems in the context of requirement management. Sketching of a new requirement management model in an Enterprise Systems project for the further development of the existing system landscape.

3 Workshop Agenda

- Introduction to the topic and keynote by the organizers of the workshop.
- Presentation of the submitted and accepted papers with subsequent discussion.
- Formation of a panel of experts out of the workshop participants (Enterprise Systems users, Enterprise Systems providers, consultants and academic staff from universities and technical colleges).

[1] Andrássy Universität Budapest, Ungarn christoph.weiss@andrassyuni.eu
[2] Universität Innsbruck, Institut für strategisches Management, Marketing und Tourismus, Innsbruck, Österreich, johannes.keckseis@uibk.ac.at

doi:10.18420/SE2020_68

- Cardsorting: Each group of experts uses card-sorting to develop a requirements management model in an Enterprise Systems project to further develop the existing system landscape.
- The models are presented, laid over each other, discussed and subsequently derived a jointly new model.
- Summary and resume by the organizers.

4 Program Committee

- Martin Adam, FH Kufstein Tirol
- Wolfgang Ahammer, VFI GmbH
- Roman Clara, Wuerth Phoenix S.R.L.
- Fritz Fahringer, Standortagentur Tirol GmbH
- Sylvia Geyer, FH Technikum Wien
- Gunther Glawar, EVVA Sicherheitstechnologie GmbH
- Bernhard Göbl, Deloitte Services Wirtschaftsprüfungs GmbH
- Bernd Lessmann, ams.erp Solution GmbH
- Felix Piazolo, Universität Innsbruck
- Kurt Promberger, Universität Innsbruck
- Ludwig Rupp, Rupp AG
- Anton Vorraber, KTM AG

5 Perspective

The results of the development of the new procedure model for requirement management in an Enterprise Systems project for the further development of the existing system landscape are processed in a paper and submitted to the next Software Engineering Conference 2021.

17. Workshop Automotive Software Engineering

Patrick Ebel[1] Steffen Helke[2] Ina Schaefer[3] Andreas Vogelsang[4]

The 17th Workshop on Automotive Software Engineering (ASE'20) addresses the challenges of software development in the automotive sector and consequently with suitable methods, techniques, and tools for this specific area. With the increasing amount of connected vehicles, modern driver assistance systems and the challenges of fully automated driving, automotive software plays an important role today more than ever.

Furthermore, the distraction-free and intuitive operation of vehicle applications via voice control plays an increasingly important role. The trend towards networking has long since reached the vehicle. Driving a car is thus being changed by the advancing "digital cultures": Services such as WhatsApp, Skype or even Facebook will be integrated in the car in the near future and can then be operated by users while driving.

The main objectives of the workshop are the exchange and discussion of how current challenges in automotive software engineering can be mastered. The thematic orientation offers many cross-references to the Software Engineering (SE) conference to which the workshop is colocated. The workshop addresses researchers, developers, and users from the automotive industry as well as scientists from research institutes and universities working in the field of automotive software engineering. Traditionally, the focus is less on theory than on applied research.

As in previous years, the workshop will be opened with a keynote speech. We would like to thank Dr. Alessio Gambi (Chair of Software Engineering II, University of Passau), who agreed to give a keynote. In addition, this year there is an invited talk by Nadine Schimanski and Patrick Göhlich (both German UPA) and a session regarding the role of Automotive Software Engineering in teaching with the objective to create an Automotive Software Engineering curriculum.

[1] Technische Universität Berlin, patrick.ebel@tu-berlin.de
[2] Fachhochschule Südwestfalen, helke.steffen@fh-swf.de
[3] Technische Universität Braunschweig, i.schaefer@tu-braunschweig.de
[4] Technische Universität Berlin, andreas.vogelsang@tu-berlin.de

doi:10.18420/SE2020_69

Program Committee

Dr. Christian Allmann	Audi AG
Prof. Dr. Marcel Baunach	Technische Universität Graz
Dr. Klaus Becker	BMW Group
Dr. Mirko Conrad	samoconsult GmbH
Dr. Heiko Dörr	Model Engineering Solutions 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

Organization

Patrick Ebel	Technische Universität Berlin
Prof. Dr. Andreas Vogelsang	Technische Universität Berlin
Prof. Dr. Ina Schaefer	Technische Universität Braunschweig
Prof. Dr. Steffen Helke	Fachhochschule Südwestfalen

For many years, this workshop has been organized by the GI interest group (Fachgruppe) on "Automotive Software Engineering"[5]. The steering committee was consequently involved in the organization of this workshop as well.

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

Autorenverzeichnis