Faculty of Informatics
Masaryk University

Experimental Research Towards Software Systems Quality

Habilitation Thesis
(Collection of Articles)

Bruno Rossi

April 2023
Brno, Czech Republic

Abstract

Software Systems are playing a significant role in nowadays society. Their importance is even more relevant nowadays as Smart Environments have been envisioned and deployed. The integration of cyber and physical systems brings new needs in terms of the qualities of such systems in terms of availability, reliability, safety, security, and resilience. Software development processes need to adapt to consider these quality concerns, leading to the adaptation of software development approaches for designing such systems and taking automation needs into account.

This Habilitation Thesis brings forward my contributions with a focus on improving the quality of designing such systems, discussing the properties and impact of qualities in the development process and at the level of software reliability of such systems. The contributions are supported by empirical Software Engineering methods with the evaluation of automation for improving system qualities at design time (software mutation testing, code reviewers recommendation, technical debt identification, automation of issue severity recommendation and triaging) and for the evaluation of the reliability of software systems utilizing Software Reliability Growth Models (SRGMs). The journey in this direction adopts Smart Grids as a representative specimen of cyber-physical systems that can be studied for the evaluation of qualitative properties in terms of co-simulations and software testing activities.

The Habilitation Thesis is structured as a commentary to a collection of articles discussing the contributions to the state of the art, linking them in a cohesive view related to software systems qualities.

Keywords: Empirical Software Engineering, Software Quality, Software Systems, Smart Grids.

Acknowledgements

I thank Tomáš Pitner for welcoming me to the Lasaris Lab after my application as Assistant Professor at Masaryk University in late 2013. I am thankful for all the suggestions and guidance in all these years. I am also grateful to my colleagues Barbora Bühnová and Radek Ošlejšek for all the support and ongoing collaborations during these years as "senior" members of the lab.

I am thankful to all colleagues and Lasaris members over all these years. Especially the "historical" members: Stanislav Chren, Karolína Burská, Martin Macák, Lukáš Daubner – those that were keeping the lab alive all this period. I am also indebted to all my colleagues from the Center of Applied Software Engineering (CASE) at FUB for all the research collaborations, projects, and productive discussions carried on more than nine years ago. Without my family, all this would not have been possible with all those sacrifices that I hope I will be able to repay.

Most of the studies collected here would not have been possible without access to the resources from CERIT-SC. Part of the research was supported from European Regional Development Fund Project CERIT Scientific Cloud (No. CZ.02.1.01/0.0/0.0/16 013/0001802), and from access to the CERIT-SC computing and storage facilities provided by the CERIT-SC Center under the programme "Projects of Large Research, Development, and Innovations Infrastructures" (CERIT Scientific Cloud LM2015085).

Most of the results were achieved in the context of the ERDF project "CyberSecurity, CyberCrime and Critical Information Infrastructures Center of Excellence" (No. CZ.02.1.01/0.0/0.0/16_019/0000822).

Bruno Rossi

Contents

Part I

Commentary

Chapter 1

Introduction

This Habilitation Thesis consists of a collection of 17 selected peer-reviewed publications: 11 conference papers and 6 journal papers. These represent a selected list to showcase my effort in the area of Software Systems Quality improvement in the range of nine years in the period 2014-2022.

In this introductory chapter, there will be a discussion about the main motivations behind this thesis, a summary of the areas covered, and the main goals and structure of the thesis. The following chapters will go into more detail about the contributions in the areas covered in the thesis.

1.1 Motivation

The implementation of software systems has become nowadays a challenging activity based on the size, complexity, and interrelations of modern software systems [82]. There has been a paradigm shift from the 70s, where the definition of algorithms was the main challenge, to more recent years in which System-of-Systems (SoS) and System-of-Systems Engineering (SoSE) started to be the focus of research, attempting to develop methods and support for large scale software development and deployment.
Lehman's Laws of software evolution were originally defined in the '80s [69, 70] as a set of laws for the software evolution of software systems, stating in the second law that evolving systems are subject to inevitable effort if activities are not undertaken to reduce such complexity. While the last laws were formalized in the 90's [70], modern systems represent complexities that go beyond the laws, taking into account the complexity of the interactions among different components, systems, and sub-systems can lead to emerging behaviors difficult to fully control and understand by software development teams [118].

1.2 Focus of the thesis

To better support and understand the software development processes in terms of the impact on software systems' qualities, my research dealt with the definition of automation and the application of automated data analysis methods to support quality during the software development processes (software mutation testing, code reviewers recommendation, technical debt identification, automation of issue severity recommendation and triaging) and for the evaluation of the reliability of software systems utilizing Software Reliability Growth Models (SRGMs). The concrete domain of application was the context of Smart Grids in which anomaly detection and co-simulations, and testing processes were investigated. This progresses from my PhD thesis [101] in which I was delving into path-dependent processes to understand open-source software development processes.
This Habilitation Thesis focuses on providing an overview of my achievement in the area of Software Quality and the connection with cyber-physical systems with a specific focus on Smart Grids. The main results of my research in software development process quality and systems qualities are summarized in Chapter 2, while Chapter 3 is focused on the application domain of Smart Grids in which we investigated anomaly detection in data analysis processes and the application of co-simulations for testing. Each of the chapters is divided by research topic, which is then supported by the research articles that are framed in the discussion. A brief overview of my own contribution is described for each paper.

1.3 Methods

In this Habilitation Thesis, several methods from empirical software engineering – more concretely Evidence-Based Software Engineering (EBSE) [63, 29] – have been applied. EBSE aims to improve software development practices through the use of empirical data and scientific methods [63]. The methods seek to provide evidence-based answers to questions related to software engineering, such as what software development methods are most effective, what quality attributes are most important for different types of software systems, and how to measure the effectiveness of software development processes. EBSE uses data from various sources, such as experiments, surveys, and case studies, to gain insights into software engineering practices and to develop best practices for software development. The goal of EBSE is to make software development more efficient, effective, and reliable by relying on evidence and data, rather than intuition and opinion [29].

In this thesis, various methods of empirical software engineering has been adopted, from action research, to mining software repositories research, data analysis, controlled experiments, mapping studies and systematic literature reviews.

1. *Quasi experimental designs* [16]. As in software engineering context is difficult to conduct controlled experiments, *quasi experimental designs* are typically adopted missing

the randomization method for assignment and also less direct control of the treatment;

2. *Mining Software Repositories* [47]. Processing, collecting, analyzing data from source control systems and bug repositories represents an opportunity for understanding software development models but also to build models about the reliability of software systems based on the stored information. This approach has been adopted in several large-scale studies in this research collection. For example, building Software Reliability Growth Models (SRGMs) on a large set of open source software projects [85].

3. *Prediction Models.* Several machine learning models have been adopted during the research. Either building models such as Software Reliability Growth Models (SRGMs) [105, 24, 85] or adopting models such as Naive Bayes (NB) classifier (e.g., in [58]) or Support Vector Machines (SVM) (e.g., in [26]) for the classification of severity of software defects or for the automation of the bug triaging recommendation process. These quantitative models have been useful for the automation of the evaluation of phenomena in the context of software development efforts.

4. *Action research* [114, 9] is a research method that has been adopted when the researcher interacts with the industrial partner, directly participating to the activities while conducting the research – this is very different from typical case study-based research.

5. *Replications* [111] are also a relevant instrument in EBSE to gather in some cases, replicating part of teh study can give insights about the quality of the research and if there are any issues. Especially replicability of results is a complex problem in Software Engineering so the aim was in recent years to publish the experimental packages – however, such effort is often undervalued in conference submission, even though there has been a trend of increasing importance in recent years.

6. *Systematic Mapping Studies (SMS)* [98] and *Systematic Literature Reviews (SLR)* [14]. These methodologies of reviewing literature in a systematic way have been adopted in a consistent way thorughout the research. They have been adopted in the context of EBSE to gather more relevant research.

One key aspect is the collaboration with industry that is relevant in the context of EBSE [17]. This was reflected in some of the articles, in which the applied component is fundamental for knowledge transfer (e.g.,[57, 26]).

1.4 Thesis structure

Each chapter and section of this Habilitation Thesis represents the area in which the described research problems that have been tackled. To help navigate through each contribution, Fig. 1.1 summarizes the main areas of contribution of the collection of articles included in this Habilitation Thesis.

In Chapter 2, we outline the main state of the art in software quality and specifically research into software development process quality and systems qualities with specific focus on the software reliability improvements. Scaling Agile Software Development, Software Code Reviews Automated Recommendation, Mutation Testing to improve Test Quality, Technical Debt Identification, Software Quality & Reliability Engineering with SRGMs, but also Software Quality in Software Engineering Education are discussed.

In Chapter 3, we outline the application domain of Smart Grids, that is how the quality considerations introduced in the previous chapter have been discussed in the context of Smart Grids in terms of anomaly detection and testing and co-simulation platforms that have been researched and implemented during the research.
In Chapter 4 there are the main conclusions, summarizing the main results derived from the collection of research works included in the Habilitation Thesis.

In Appendix A, we present the list of the articles included in this Habilitation thesis divided by type of contribution and impact.

1.5 Papers Collection

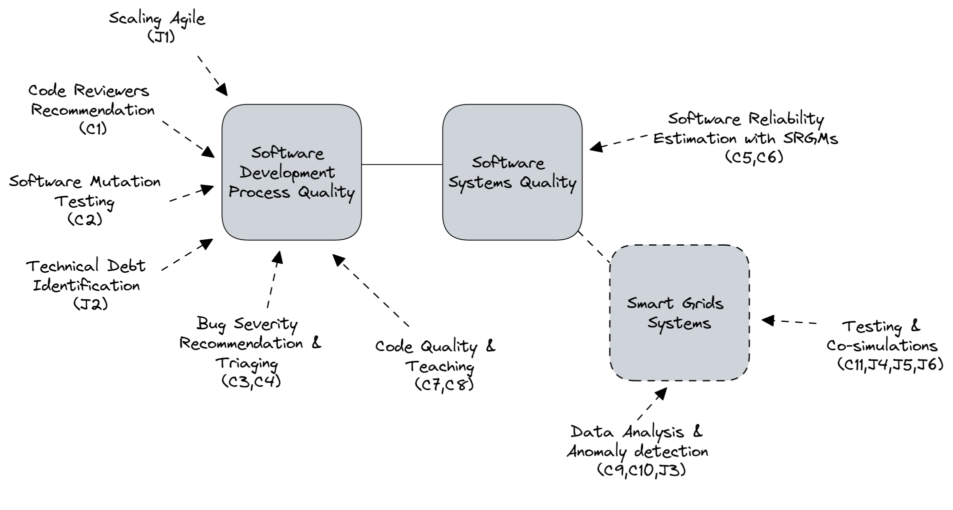
Scaling Agile (J1)
Code Reviewers Recommendation (C1)
Software Mutation Testing (C2)
Technical Debt Identification (J2)
Bug Severity Recommendation & Triaging (C3,C4)
Code Quality & Teaching (C7,C8)
Software Development Process Quality
Software Systems Quality
Software Reliability Estimation with SRGMs (C5,C6)
Smart Grids Systems
Testing & Co-simulations (C11,J4,J5,J6)
Data Analysis & Anomaly detection (C9,C10,J3)

Figure 1.1: Mapping of the articles included in the Habilitation Thesis to different quality areas (the full list of articles is available in Appendix A)

C1. (CORE B). J. Lipcak and B. Rossi. A large-scale study on source code reviewer recommendation. In *2018 44th Euromicro Conference on Software Engineering and*

Advanced Applications (SEAA), pages 378–387, 2018

C2. (CORE B). J. Možucha and B. Rossi. Is mutation testing ready to be adopted industry-wide? In P. Abrahamsson, A. Jedlitschka, A. Nguyen Duc, M. Felderer, S. Amasaki, and T. Mikkonen, editors, *Product-Focused Software Process Improvement*, pages 217–232, Cham, 2016. Springer International Publishing

C3. (CORE B). N. Kanti-Singha Roy and B. Rossi. Towards an improvement of bug severity classification. In *2014 40th Euromicro Conference on Software Engineering and Advanced Applications*, pages 269–276, 2014

C4. (CORE B). V. Dedík and B. Rossi. Automated bug triaging in an industrial context. In *2016 42th Euromicro Conference on Software Engineering and Advanced Applications (SEAA)*, pages 363–367, 2016

C5. (CORE A). S. Chren, R. Micko, B. Buhnova, and B. Rossi. Strait: A tool for automated software reliability growth analysis. In *2019 IEEE/ACM 16th International Conference on Mining Software Repositories (MSR)*, pages 105–110, 2019

C6. (CORE B). R. Mičko, S. Chren, and B. Rossi. Applicability of software reliability growth models to open source software. In *2022 48th Euromicro Conference on Software Engineering and Advanced Applications (SEAA)*, pages 255–262, 2022

C7. (CORE A). S. Chren, B. Buhnova, M. Macak, L. Daubner, and B. Rossi. Mistakes in uml diagrams: Analysis of student projects in a software engineering course. In *2019 IEEE/ACM 41st International Conference on Software Engineering: Software Engineering Education and Training (ICSE-SEET)*, pages 100–109, 2019

C8. (CORE A). S. Chren, M. Macák, B. Rossi, and B. Buhnova. Evaluating code improvements in software quality course projects. In *Proceedings of the International Conference on Evaluation and Assessment in Software Engineering 2022*, EASE '22, page 160–169, New York, NY, USA, 2022. Association for Computing Machinery

C9. (CORE B). B. Rossi, S. Chren, B. Buhnova, and T. Pitner. Anomaly detection in smart grid data: An experience report. In *2016 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, pages 2313–2318, 2016

C10. (CORE B). P. Lipčák, M. Macak, and B. Rossi. Big data platform for smart grids power consumption anomaly detection. In *2019 Federated Conference on Computer Science and Information Systems (FedCSIS)*, pages 771–780, 2019

C11. (CORE B). M. Schvarcbacher and B. Rossi. Smart grids co-simulations with low-cost hardware. In *2017 43rd Euromicro Conference on Software Engineering and Advanced Applications (SEAA)*, pages 252–255, 2017

J1. (Q2 - IF2018: 2.046). M. Kalenda, P. Hyna, and B. Rossi. Scaling agile in large organizations: Practices, challenges, and success factors. *Journal of Software: Evolution and Process*, 30(10):e1954, 2018. e1954 smr.1954

J2. (Q3, IF2020: 1.025). P. Strečanský, S. Chren, and B. Rossi. Comparing maintainability index, sig method, and sqale for technical debt identification. *Scientific Programming*, 2020:14, Jul 2020

J3. (Q1, IF: 10.215). B. Rossi and S. Chren. Smart grids data analysis: A systematic mapping study. *IEEE Transactions on Industrial Informatics*, 16(6):3619–3639, 2020

J4. (Q2). V. Lamba, N. Šimková, and B. Rossi. Recommendations for smart grid security risk management. *Cyber-Physical Systems*, 5(2):92–118, 2019

J5. (Q1, IF: 4.028). P. Mihal, M. Schvarcbacher, B. Rossi, and T. Pitner. Smart grids co-simulations: Survey & research directions. *Sustainable Computing: Informatics and Systems*, 35:100726, 2022

J6. (Q1, IF2018: 2.217). M. Schvarcbacher, K. Hrabovská, B. Rossi, and T. Pitner. Smart grid testing management platform (sgtmp). *Applied Sciences*, 8(11), 2018

Chapter 2

Software Systems Quality

Software Quality research started in the late 60s when the implementation of large-scale mission-critical software systems led to questioning the importance of methods and techniques to improve and care about the quality management in software processes [11]. This led to the creation of the first Software Quality Models (e.g., Boehm et al. [11, 10] and McCall model [79]). All these models later converged towards several standards, like the IEC/ISO 91260 Software Quality Model [51] and the most recent update ISO/IEC 25010:2011 Standard [52].

The rigorous study of Software Quality acquired importance in the 80s with the publication of seminal works about Software Reliability [88, 89, 77], Software Metrics [35, 37], and Software Testing [42]. The common view of the software quality community and represented in the standards (Fig. 2.1) is that process quality can impact software product quality that can be measured by both internal (e.g., source code metrics) and external quality measures (e.g., number of failures during the testing process). Such aspects then impact the software product from the side of the users that can perceive the quality of the final software product.

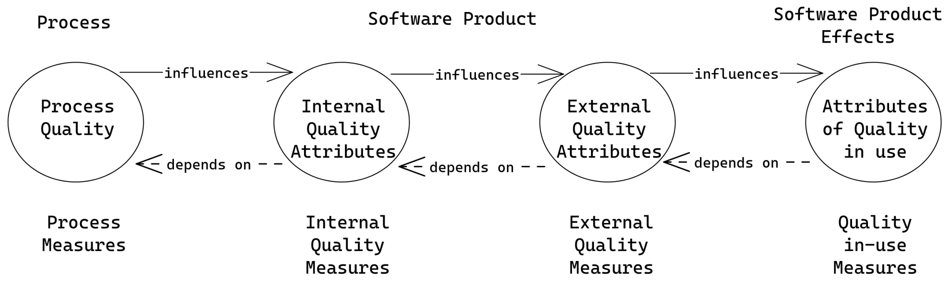

Figure 2.1: Software Quality impact, adapted from [51]

Software Quality has been defined as an elusive target [62], as it needs to be defined with precise metrics and measurements [36]. Furthermore, the whole concept can be interpreted from a different point of view, like the conformance to software specifications or the fitness for purpose [62]. As such, software systems quality cannot be seen as an independent aspect of software development process quality but is rather impacted by the quality of the development processes.

2.1 Contributions

The selected contributions in the area of Software Quality cover a thread that spans from software development process quality (adoption of scaling agile methods in Section 2.1.1, automated recommendation of code reviews in Section 2.1.2, mutation testing in Section 2.1.3) to product quality improvement (study of technical debt identification in Section 2.1.4), to the application of Software Reliability Growth Models (SRGMs) and experimentation for the evaluation of systems reliability (in Section 2.1.5). Ending this thread in evaluating Software Quality in the context of Software Engineering Education (Section 2.1.6) based on the impact of a Software Quality university course initiated several years ago at Masaryk University.

2.1.1 Scaling Agile Software Development

Agile methods have acquired large importance in recent years for developing software systems. Scaling software development has acquired high importance recently for overall quality. Nowadays, several frameworks that can be used to guide the scaling process in organizations [100, 30, 27, 86], several methods, practices, and frameworks were derived from extending traditional agile methods: Scrum of Scrums (SoS), Scaled Agile Framework (SAFe), Large-Scale Scrum (LeSS), Disciplined Agile Delivery (DAD), Lean Scalable Agility for Engineering (LeanSAFE), Recipes for Agile Governance in the Enterprise (RAGE) [2, 30, 107, 68].
In this work [57], we conducted a one-year Action Research in the context of Kentico's company to derive the benefits and challenges of scaling agile. We derived 20+ factors compared in the context of existing research. We have been using Action Research feedback loops to see the impact during the implementation to evaluate the effect of several practices that were adopted.
We studied the company's concrete scaling practices, which success factors the company experienced, and which challenges the company faced. Agile culture within the company and prior agile and lean experience, management support, and unification of views and values were key success factors during the action research process. Resistance to change, too quick roll-out, quality assurance issues, and the integration with previous non-agile parts of the organization were found to be critical challenges in the scaling process.
This action research was considered one as one of the exemplary applications of Action Re-

search to be listed in the Software Engineering Empirical Standards[1] [99]. This article is also one of the most influential papers in the area of research about Scaling Agile methods.

> **Article:** [57] M. Kalenda, P. Hyna, and B. Rossi. Scaling agile in large organizations: Practices, challenges, and success factors. *Journal of Software: Evolution and Process*, 30(10):e1954, 2018. e1954 smr.1954
>
> I designed the study, granted the collaboration with the company and coordinated the action research process. Contributed to writing the final journal version. Contribution ∼**30%**.

2.1.2 Software Code Reviews Automated Recommendation

One of the main activities related to the management of large-scale software development is running source code reviews to improve the quality of the implemented solutions. Reviewers recommendation area implies the automated suggestion of the most useful reviewer for source code pull requests. There are many algorithms that have been suggested over the years. We considered the most important categories:

- Heuristic-based approaches: Such approaches are based on finding similarities based on some assumptions. For example, if a developer has in the past dealt with a library, the developer might be a candidate for the suggestion, according to the similarity of the reviewing task (e.g., RevFinder [117]).
- Machine Learning-based: Usage of different *Machine Learning* techniques for the recommendation of code reviewers. A model will need to be built based on a training set (e.g., [55]).
- Social Networks-based: Social networks have also been used to determine similarities in communication between developers (e.g., *Comment Network (CN)* [127])
- Hybrid approaches: use a combination of approaches (e.g.,CoreDevRec [56] that builds a model based on a different set of features, also considering Social Network metrics).

In this work [73], we implemented a Naive Bayes-based approach for reviewers' recommendation and ran a large-scale evaluation of with the RevFinder implementation to identify the best code reviewers (51 projects, more than 293K pull requests analyzed, 180K owners and 157K reviewers). At the time it was written, this was the largest study about source code reviewers' recommendations. Overall, we found that no model can be generalized as best for all projects and that the usage of a different repository (Gerrit, GitHub) can have an impact on the recommendation results. Within the 51 projects, RevFinder generally provides better results in terms of Mean Reciprocal Ranks (MRR), but the effect size is

[1] `https://acmsigsoft.github.io/EmpiricalStandards/docs/?standard=ActionResearch`

small, and differences might be limited. RevFinder reaches in this dataset 60% MRR. Considering the 14 Gerrit projects with additional features, sub-project information can bring significant improvements to the recommendation results for the NB-based approach with a medium effect size.

> **Article:** [73] J. Lipcak and B. Rossi. A large-scale study on source code reviewer recommendation. In *2018 44th Euromicro Conference on Software Engineering and Advanced Applications (SEAA)*, pages 378–387, 2018
>
> I designed the study, supervised the student for the evaluation of the recommendation algorithms, contributed to the final version of the article. Contribution ∼**30%**.

2.1.3 Mutation Testing to improve Test Quality

Software Quality measurement is based on identifying and collecting metrics about the process and product quality. One critical area in the context of software testing is the quality of the written tests. Relying on code coverage is a common misunderstanding, as it cannot give a real evaluation of the quality of the tests. This is one of the major reasons why Software Mutation Testing has been proposed to improve the quality of software tests. However, it has not yet reached a wide consensus for industry-wide adoption, mainly due to missing clear benefits and computational complexity for the application to large systems.
In this work [87], we conducted an experiment to look at the support of mutation testing in the context of Java Virtual Machine (JVM) environments. In particular, we looked at the application of Higher Order Mutation (HOM): taking into account the original mutants as the First Order Mutants (FOMs), the technique creates mutants with more than a single mutation. We evaluated four different algorithms (Last2First, DifferentOperators, ClosePair and RandomMix):

- Last2First: the first mutant in the list of FOMs is combined with the last mutant in the list, the second mutant with next to last, and so on;
- DifferentOperators: only FOMs generated by different mutation operators are combined;
- ClosePair: two neighboring FOMs from the list are combined;
- RandomMix: any two mutants from the FOMs are combined;

We found out that while default configurations are unbearable for larger projects, strategies such as selective operators, second-order mutation, and multi-threading can increase the approach's applicability. However, we also found a trade-off concerning the quality of the achieved results of the mutation analysis process. Using Selective Operators can bring benefits in terms of runtime performance, however, at the expense of lower mutation scores.

Article: [87] J. Možucha and B. Rossi. Is mutation testing ready to be adopted industry-wide? In P. Abrahamsson, A. Jedlitschka, A. Nguyen Duc, M. Felderer, S. Amasaki, and T. Mikkonen, editors, *Product-Focused Software Process Improvement*, pages 217–232, Cham, 2016. Springer International Publishing

I designed the study, supervised the student, redone graphical representations and data analysis, rewritten for conference article. Contribution ∼**40%**.

2.1.4 Technical Debt Identification

Technical Debt (TD) is a metaphor for how technical decisions made at development time coined can have a long-term impact similar to economic debt. The main origin of such a concept was the so-called Lehman's laws of software evolution [70] – especially in the second law that states that *"as a system evolves, its complexity increases unless work is done to maintain or reduce it"*. There is still no clear definition of TD, and many models have been proposed over the years for the measurement. OF the many models that were proposed over the years for the calculation of TD, we were interested in the following :

- the Maintainability Index (MI) defined through the average Halstead Volume per module, the average Cyclomatic Complexity per module, the average lines of code per module, and the average lines of comments per module [94];
- SIG TD model that quantifies TD based on an estimation of repair effort and estimation of maintenance effort, which provides a view about the cost of repair and the expected payback period [93];
- Software QuALity Enhancement (SQALE) that provides an operationalization of the ISO 9126 Software Quality standard utilizing several code metrics that are attached to the taxonomy defined in the ISO/IEC 9126 standard [71];

In this work [115], we compared three techniques of TD identification that were proposed over time: (i) the Maintainability Index (MI), (ii) SIG TD models, and (iii) SQALE analysis. Considering 20 open source Python libraries, we compared the TD measurements time series in terms of trends and evolution according to different sets of releases (major, minor, and micro) to see if the perception of practitioners about TD evolution could be impacted. While all methods report generally growing trends of TD over time, there are different patterns. SQALE reports more periods of steady states compared to MI and SIG TD. MI is the method that reports more repayments of TD compared to the other methods. SIG TD and MI are the models that show more similarity in how TD evolves, while SQALE and MI are less similar. The implications are that each method gives a slightly different perception of TD evolution, which can have an impact when selecting a method to adopt.

Article: [115] P. Strečanský, S. Chren, and B. Rossi. Comparing maintainability index, sig method, and sqale for technical debt identification. *Scientific Programming*, 2020:14, Jul 2020

I designed the study, supervised the student, contributed to writing of the article for the journal. Contribution ∼**30%**.

2.1.5 Software Quality & Reliability Engineering

Being able to predict the severity of software defects has been found in previous studies to improve the overall resolution process. Many classification/prediction approaches emerged to provide automated reasoning over severity classes.
In the first seminal paper from Menzies and Marcus from 2008, the authors proposed to use text mining approaches to determine the severity levels for NASA projects [83]. The approach proposed was based on entropy and information gain, supported by a rule learner. Utilizing the top-terms, the results were in the range of 65%-98% in terms of F-Measure. Some initial attempt was also based on the use of the Naïve Bayes classifier to classify issues considered as severe, non-severe [66]. The main results had a precision and recall from 65% to 75% (Mozilla and Eclipse) and 70%-85% (GNOME).
Lamkanfi et al. also provided a comparison of several classifiers by using their previously proposed approach [67], comparing Naïve Bayes (NB), Naïve Bayes Multinomial (NBM), K-Nearest Neighbor (K-NN) and Support Vector Machines (SVM) applied to Eclipse and GNOME projects. Feature selection was adopted to improve the results of severity prediction algorithms [125] – comparing three different feature selection schemes: Information Gain, Chi-Square, and Correlation Coefficient.
In this work [58], we used text mining together with bi-grams and feature selection to improve the classification of bugs in severe/non-severe classes. We adopt the Naive Bayes (NB) classifier considering Mozilla and Eclipse datasets commonly used in related works. Overall, the results show that applying bi-grams can slightly improve the classifier's performance, but feature selection can be more effective in determining the most informative terms and bi-grams.

Article: [58] N. Kanti-Singha Roy and B. Rossi. Towards an improvement of bug severity classification. In *2014 40th Euromicro Conference on Software Engineering and Advanced Applications*, pages 269–276, 2014

I designed the study. Participated to the analysis and contributed with most of the writing. Contribution ∼**50%**.

The bug triaging process recommendation automation represents the automation of the assignment of the most appropriate developers for solving issues in bug reports. With this aim, we devised an automation method. We cooperated with an industrial partner to define an automated bug triager that could automate the process of assigning developers to software defects for resolution.
In this work [26] After reporting the requirements and needs that were set within the

industrial project, we compare the analysis results with those from an open source project used frequently in related research (Firefox). Overall, we found that more easily configurable models (such as SVM+TF–IDF) are preferred, and top-x recommendations, number of issues per developer, and online learning can all be relevant factors when dealing with industrial collaboration.

> **Article:** [26] V. Dedík and B. Rossi. Automated bug triaging in an industrial context. In *2016 42th Euromicro Conference on Software Engineering and Advanced Applications (SEAA)*, pages 363–367, 2016
>
> I designed the study and supervised the student for the conduction of the analysis. Coordinated the cooperation with the industrial partner. Contributed to the writing. Contribution ~**35%**.

Software Reliability Growth Models (SRGM) emerged in the '70es when Jelinski and Moranda proposed the JM model for fitting cumulative failure data to define a methodology to analyze and predict project failures over time [77, 54]. Failure events are encountered during the testing and early operation phases of the software life cycle. They are recorded, and the underlying faults that caused them are removed, which results in a process called *reliability growth*. SRGMs are regression-based models whose purpose is to estimate the parameters of a mean value function $m(t)$ based on the input data: $m(t)$ represents the cumulative number of faults detected by the given time t. SRGMs have been my interest since my PhD studies. We had one of the first papers with the application of SRGMs to open source software in 2010 [105] - considered one of the starting points in literature for the application of SRGMs in this context. SRGMs are regression-based models that use historical failure data to predict the reliability of software projects. The main contributions were to provide a dedicated tool to support the whole process of SRGMs data preparation and application from issue repositories that can be used for data analysis. The tool was then adopted to conduct the largest study of the application of SRGMs in the context of open source software projects.
In this work [24], to support the replicability of results and analyze large datasets, we implemented one tool (STRAIT) that can help with the analysis of software projects for the definition of SRGMs. The tool provides features such as downloading, filtering, and processing of data from provided issue repositories for use in multiple SRGMs, supporting fitting SRGMs with different snapshots in the tool. The tool allows customization of the filtering capabilities and the implementation of custom SRGMs.

> **Article:** [24] S. Chren, R. Micko, B. Buhnova, and B. Rossi. Strait: A tool for automated software reliability growth analysis. In *2019 IEEE/ACM 16th International Conference on Mining Software Repositories (MSR)*, pages 105–110, 2019
>
> I participated to the tool design. Contributed to the writing. Contribution ~**20%**.

In this work [85], we analyzed the performance of SRGMs for the understanding of open

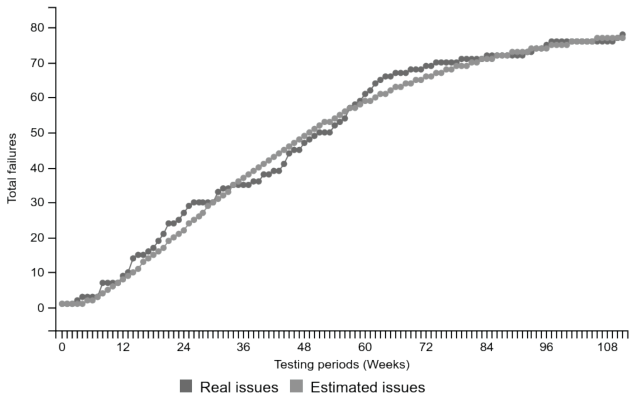

Figure 2.2: Sample output from STRAIT fitting cumulative failure data with the Goel-Okumoto S-Shaped model

source software projects. The main issues in the applicability of SRGMs to OSS are based on the assumptions that are made about the defects-fixing process. Some of these assumptions might be violated, like the fact that faults are repaired when they are discovered, fault repairs are perfect, or that no new code is introduced during the defects fixing process [122]. We compared nine SRGMs (Goel-Okumoto (GO) & Goel-Okumoto S-Shaped (GOS) [43], Hossain-Dahiya (HD) [49], Musa-Okumoto (MO) [90], Duane (DU) [28], Weibull (WE) [89], Yamada Exponential (YE) [124], Yamada Raleigh (YR) [124], Log-Logistic (LL) [44]) . We present an experimental study of SRGMs applicability to a total of 88 OSS projects, comparing nine SRGMs, looking at the stability of the best models on the whole projects, on releases, on different domains, and according to different projects' attributes. This represents a large-scale study (1 053 SRGMs, $\sim 400K$ software defects) with the aid of the STRAIT tool [24]. We found good applicability of SRGMs to OSS, but with different performance when segmenting the dataset into releases and domains. Based on 792 fitted SRGMs, considering the R^2 metric, we found that LL, YR, WE, HD, DU are the best models. GO, GOS, MO, YE show the highest variance than other models. GOS is, in general, the worse model in terms of R^2.
Considering several domains of the systems, the results in terms of R^2 metric report some models with good consistency across domains (e.g., LL, WE, YR). The GOS model, while statistically worse than all other models when considering the whole dataset, has some domains in which it has low variance and good R^2 rankings.
Considering the effect of releases on SRGM has limited impact in terms of R^2 (the top-3 models remain the same). Considering different performance metrics can lead to different rankings of models, signaling that some models, such as YR might be penalized for overfitting the data. The GOS model remains the worse, both considering releases and projects as a whole.

Article: [85] R. Mičko, S. Chren, and B. Rossi. Applicability of software reliability growth models to open source software. In *2022 48th Euromicro Conference on Software Engineering and Advanced Applications (SEAA)*, pages 255–262, 2022

I participated to the design of the study. Conducted additional data analysis. Contributed to the writing. Contribution ~**30%**.

2.1.6 Software Quality in Software Engineering Education

In recent years, new courses targeting Software Quality have emerged in universities to make students aware of concerns and techniques related to Software Quality (e.g., [53, 81]). Although studies looked at the quality of students' contributions (e.g., [123, 46, 76, 64, 59, 13]), there was still no comprehensive evaluation of the impact of such a course on the quality of the produced source code.

In this work [23], we performed an extensive study about the quality of source code related to students of the Software Quality course given at the Faculty of Informatics of Masaryk University. The aim was to see if the efforts in teaching various Software Quality aspects affect the quality of the produced code. As a source, we designed a quasi-experimental design, one group pre-test post-test ($O1XO2$), in which a group of students performs the initial project implementation ($O1$). The treatment (X, Software Quality course) is applied, afterward the implementation by students is repeated ($O2$). Manual inspections and code metrics are used for the evaluation. We analyzed 54 project submissions from 27 students using manual and automated quality assessment methods. We have employed 30 manual and 22 automated quality characteristics related to coding style, architecture design, and general development practices. We examined which characteristics of the code have improved the most and what were the most common issues. Overall, we found that Code Complexity was improved for all categories of students. There were good improvements in SRP and OCP categories of SOLID principles. Code Smells and Checkstyle normalized errors were reduced in the final submission. The most common issues were related to error handling, structuring of methods, and issues related to code documentation. Violations of Single Responsibility and Open-Closed principles with insufficient separation of responsibilities were also common concerns. Test automation was generally absent in the initial submission and very limited in the final version.

Article: [23] S. Chren, M. Macák, B. Rossi, and B. Buhnova. Evaluating code improvements in software quality course projects. In *Proceedings of the International Conference on Evaluation and Assessment in Software Engineering 2022*, EASE '22, page 160–169, New York, NY, USA, 2022. Association for Computing Machinery

I participated to the design of the study. Analysis of one research question. Contributed to the writing. Contribution ~**20%**.

In this work [22], we experimented to understand the impact of the usage of the Unified

Modelling Language (UML) in terms of the most common errors and misconceptions. The outcome was a catalogue of the most common issues in learning UML models that can be useful for future generations of students. Overall, 146 types of mistakes in eight types of diagrams were represented in the catalogue. We utilized the catalogue to analyze students' projects in a software engineering course with 2,700 diagrams submitted by 123 students. We reviewed the frequency of mistakes, the correlations of the mistakes between different diagram types, and the correlation of the quality of student projects to exam results, among other aspects.

> **Article:** [22] S. Chren, B. Buhnova, M. Macak, L. Daubner, and B. Rossi. Mistakes in uml diagrams: Analysis of student projects in a software engineering course. In *2019 IEEE/ACM 41st International Conference on Software Engineering: Software Engineering Education and Training (ICSE-SEET)*, pages 100–109, 2019
>
> I participated to the design of the study. Analysis of one research question. Contributed to the writing. Contribution ∼**15%**.

2.1.7 Ongoing Research Directions

Several research directions are still to be explored. I mention here the two main areas I plan to focus the research:

- *Automation in Software Engineering*: many works in this collection deal with the automation of processes to improve the quality of processes and systems. With the recent emergence of generative AI language models [119] many new possibilities go beyond the initial design of the models and studies. The application can be at several levels and impact most of the studies in this Habilitation Thesis – to exemplify, the models can have a potential impact on mutation testing for the definition of mutants [87], at the level of source code reviewers recommendation thinking about agents that can self-perform recommendation tasks and bug triaging [73, 26], but the applications can be many that can be explored to evaluate the positive and negative effects.

- *Software Reliability and SRGMs*: the application of SRGMs has been based on data collection by means of Mining Software Repositories studies. This will still be my research focus in comparing quality models of open source software projects versus SRGMs models. However, my focus will be more on applying such models to Microservices-based systems for the improvement of reliability and resilience by using the outputs of the models to provide improvements for Microservices as self-adaptive systems. This will bring more interesting results in terms of architectural and software systems engineering approaches to take into account knowledge learned from the models analyzing the current run of the systems. I plan to connect this part to the simulation parts discussed in the upcoming Section 3.1.2 as running the models in a simulated environment can bring benefits to evaluating the properties of the running systems and modifying them in production systems based on the results of the simulations. At the moment, many theses are on-going to investigate these matters – for example, about reconstructing resilience patterns from a Microservice system to a Microservice simulator for running models about the runtime availability of the services.

Chapter 3

Application Domain: Smart Grids

Software Quality is a critical aspect of Smart Grid Systems, as it directly impacts the reliability, security, and efficiency of these systems [33]. Smart Grids (SGs) are cyber-physical systems that use the information to provide reliable, efficient, and sustainable electricity to end-users [33]. They play a significant role in the Smart Cities concept by enabling the integration of Smart Energy systems that interconnect utilities and end-users through a Smart Infrastructure [41, 12, 31]. SGs enhance the decision-making process, enable self-healing and automation of the energy grid, and facilitate the integration of renewable energy sources [41]. In Smart Grids, the software is used to manage, monitor, and control the flow of electricity in the grid, as well as to support customers and power load management. As such, high-quality software is essential for the efficient and reliable functioning of Smart Grids.
In particular, Software Quality attributes such as reliability, maintainability, scalability, and security are particularly important. Software must be able to withstand high levels of stress and variability in the electrical grid, as well as to be quickly and easily updated as needed. Additionally, software must be secured to prevent security threats that could disrupt the flow of electricity.

The US National Institute of Standards and Technologies (NIST) was one of the first organizations to create a structure for a Smart Grid (SG), developing a conceptual model and a more specific reference architecture model [91, 92]. The CEN-CENELEC-ETSI standardization Group later modified these models for the European context, resulting in the Smart Grid Architecture Model (SGAM) Framework (Fig. 3.1). This framework has multiple interoperability layers mapped to the SG pane, which is formed by physical, electrical domains and information management zones. The SGAM model aims to represent the zones of information management where interactions between domains occur.
SGs face several challenges arising from the integration of physical infrastructure with in-

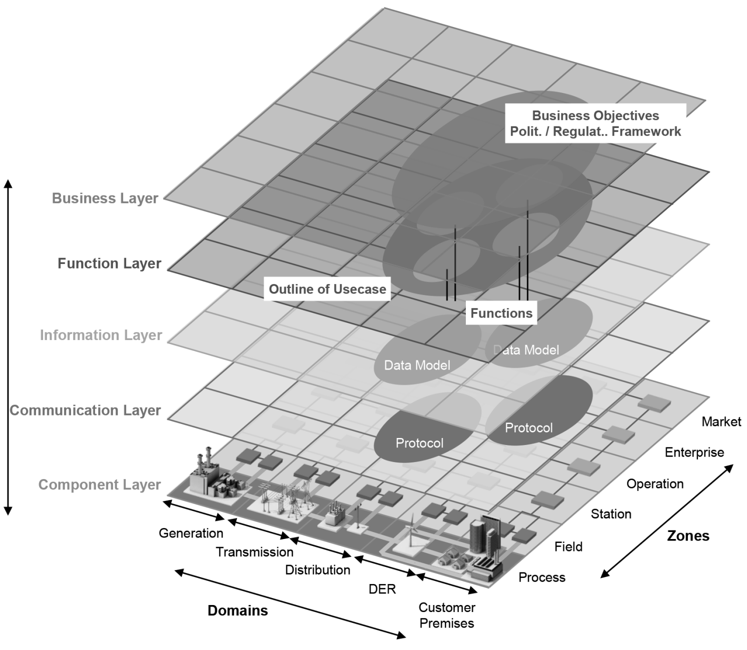

Figure 3.1: The Smart Grid Architecture Model (SGAM) [18]

formation and communication technologies [18]. Addressing these challenges requires a holistic approach that considers all layers of the SG ecosystem [18]. These challenges include prioritizing communication network availability over traditional confidentiality and integrity aspects [5], ensuring customers' privacy and infrastructure security [80], finding reliable ways to integrate renewable energy sources [121], and utilizing information/data for self-healing and self-monitoring purposes [3, 34, 4].

The interoperability layers define viewpoints through which the SG has to be considered [18]. The *Component layer* specifies the physical distribution of all participating components in the SG, including actors, applications, power system equipment, protection and control devices, network infrastructure, and any kind of devices. The *Communication layer* describes protocols and mechanisms for the exchange of data between components within the context of usage scenario, function, or service. The *Information layer* details the information that is exchanged between functions, services, and components. It contains information objects and the underlying data models.
The *SGAM physical electrical domains* capture the electrical energy conversion chain and consist of five parts. Bulk generation represents the energy generation in large quantities, for example, by fossil, nuclear or hydro-power plants. Such generation is connected to the transmission system. Transmission represents the infrastructure and organization for long-distance energy transportation. The distribution represents the infrastructure and organization that distributes the electricity to customers. Distributed Energy Resources

(DER) describes the distributed electrical resources connected to the public distribution grid. Customer premises include the consumers of electricity and the local producers.

3.1 Contributions

In the area of Smart Grids, the contributions span from the evaluation of data analysis techniques, data quality concerns, and anomaly detection that have been applied in the context of ongoing projects (Section 3.1.1). Afterward, the knowledge acquired has been applied in the context of Smart Grids co-simulations (Section 3.1.2) culminating in the evaluation of a low-cost hardware-in-the-loop prototype and the creation of a platform (SGTMP) to support Smart Grids testing with the Mosaik co-simulation framework.

3.1.1 Anomaly Detection in the Smart Grids Context

The emergence of Smart Grids has created a large set of opportunities for data analytics initiatives. The vast amount of data collected from the smart infrastructure can be utilized for decision support and predictive algorithms to enhance the services provided [103]. One such initiative is power load forecasting, which predicts customers' electricity consumption over time [61]. Another example is Demand Response (DR), which involves balancing energy supply and demand load during peak hours [39].

In this work [102], we conducted a large Systematic Mapping Study (SMS) in the area of Smart Grids, gathering methods that have been adopted in the area for analyzing various aspects of SG.
We gathered techniques that were applied in different application subdomains (e.g., power load control), aspects covered (e.g., forecasting), used techniques (e.g., clustering), tool support, research methods (e.g., experiments/simulations), and replicability/reproducibility of research. We identified ten main sub-domains for SG data analysis: C1. customer profiling, C2. energy output forecasts, C3. events analysis, C4. load segregation, C5. Power loads/consumption analysis, C6. power quality, C7. pricing, C8. privacy, C9. security, C10. Smart Grid failures. The themes that are covered in order of numerosity of research are loads forecasting, energy pricing forecasting, forecasting production from renewable power sources, power loads clustering, users power consumption profile clustering, false data injection attacks, users power consumption pattern recognition, power quality disturbances classification, non-intrusive appliance load monitoring, power data compression, energy theft detection, and SG faults detection.
As interesting findings, simulations and experiments are crucial in many areas. The replicability of studies is limited concerning the provided implemented algorithms and, to a lower extent, due to the usage of private datasets. This paper has been published in IEEE Transactions in Industrial Informatics, which has an IF of over 10: with 358 articles being the largest study conducted in the area.

Article: [102] B. Rossi and S. Chren. Smart grids data analysis: A systematic mapping study. *IEEE Transactions on Industrial Informatics*, 16(6):3619–3639, 2020

I designed the study. Shared the conduction of data analysis and review. Contributed to the writing. Contribution ∼**50%**.

In the context of the collaboration with industrial partners, we worked on defining anomaly detection based on Smart Metering events.
Anomaly detection is a complex task, requiring identifying various types of anomalies. Point, context, and collective anomalies have been categorized in the literature [19, 1]. Point anomalies are single instances that deviate from the rest of the data, while context anomalies consider external factors in determining whether a data point is anomalous. Collective anomalies consider specific patterns identified over time and require a different approach for detection. Therefore, identifying anomalies by looking at threshold intervals might not be sufficient in certain cases.
There are several approaches for detecting anomalies, such as probabilistic models that define properties for anomalies based on data distributions or models that use data proximities, such as k-nearest neighbor algorithms, which examine distances between k-data points in space. Time series properties can also be used for anomaly detection [1].
Power consumption anomaly detection focuses on identifying anomalous data traces that could indicate relevant domain events, such as energy theft, tampering with smart meters, or device failure. Typically, time series are used to represent power traces that change over time, and multiple time series, such as power traces and weather data, may be used to identify anomalies.
The application of different models for anomaly detection varies significantly. Typically, most models incorporate temporal aspects and involve statistical properties. At a fundamental level, there are several categories of anomaly detection models, including linear models (e.g., regression-based), proximity models (e.g., k-nearest neighbour), statistical models (e.g., two sigma rule), and density-based models (e.g., clustering). However, it can be difficult to categorize models strictly as many approaches are often combined in ensemble models (e.g., time series forecasting methods with extreme value anomaly detection approaches).
Several models have been proposed for power consumption anomaly detection. Initial models, like the one presented in [128], used regression models to estimate power consumption and temperature, flagging data points that deviated from the real values by a predefined threshold as anomalies. A similar approach, based on AutoRegressive Integrated Moving Average (ARIMA) models, was used in [21]. Other models, such as the Periodic Auto Regression with eXogenous variables (PARX) proposed in Ardakanian et al. [8] and later improved in Liu et al. [75], focused on clustering power consumption data by temporal properties. Saad et al. [106] proposed an approach based on clustering. In contrast, Sial et al. [112] used heuristics based on the domain to group data traces by time and type of the day and then used distance-based anomaly detection techniques (e.g., kNN) to detect anomalies. Buzau et al. [15] used several machine learning approaches (k-NN, SVM, Logistic

Regression, XGBoost) to identify anomalies in users' power consumption profiles. Recent models have focused on the stochastic nature of the underlying processes and the importance of detecting concept drifts, such as in Fenza et al. [38].

In this work [103], we proposed a method based on the concept of itemsets of events that may be anomalous based on their patterns of appearance. This moves from the concept of single events identified as an anomaly to the concept of collective anomaly [1]. By analyzing Smart Meters data streams, we used frequent itemset mining and categorical clustering with clustering silhouette thresholding to detect anomalous behavior. In this study, we consider Smart Meters [25] as data sources of data streaming, allowing the analysis of all data derived from the Smart Meters' operations. The Smart Meter events are used to notify the data center about important state changes that happened at the level of the Smart Meter, such as powering up or down of the meter, tariff and rate switching, and time synchronization. Each event belongs to one of the possible 76 event types.

> **Article:** [103] B. Rossi, S. Chren, B. Buhnova, and T. Pitner. Anomaly detection in smart grid data: An experience report. In *2016 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, pages 2313–2318, 2016
>
> I designed the study. Conducted the main data analysis part. Contributed to the writing. Contribution ∼**45%**.

When looking at the Smart Grids, we examined the impact of data quality in the different parts of the system to suggest adopting a data quality model as a frame of reference.
New devices in Smart Grids, such as smart meters and sensors, have emerged to become a massive and complex network where a large volume of data flows to the smart grid systems. Data can be real-time, streaming, and originating from various smart devices. However, data quality issues might cause the delayed, inaccurate analysis of results in the context of Smart Grid Systems [40]. Based on the previous works [103, 102, 40, 104], we dealt with the adoption of Big Data Platforms for Anomaly Detection of Smart Grids data. The processing in the Smart Grid context requires real-time data analysis for several applications (e.g., intrusion and data injection attacks detection, electric device health monitoring). Two popular architectures were proposed over time, Lambda and Kappa [78, 72], based on the importance of batch and stream processing.
In this work [74], we provided the implementation of a Big Data platform for power consumption anomaly detection with the main components mapped to the reference architecture proposed by Pääkkönen and Pakkala [95]. We showcased the results of a scenario run with public datasets to assess the applicability of batch-oriented (Apache Spark), stream-oriented (Apache Storm), and hybrid (Apache Flink) frameworks. The platform is based on an ingestion layer with data densification options, Apache Flink as part of the speed layer, and HDFS/KairosDB as data storage layers. We also tested Lambda vs. Kappa architectures for power consumption data processing.

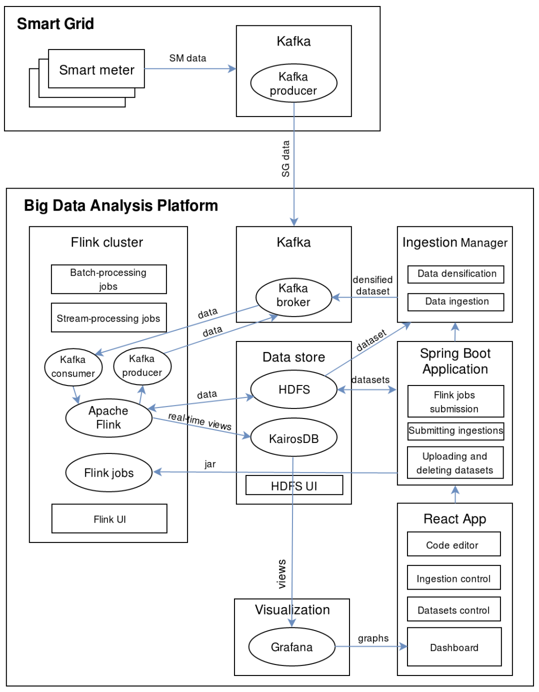

Figure 3.2: Architecture of the Big Data Platform for Power Consumption Anomaly Detection [74]

Article: [74] P. Lipčák, M. Macak, and B. Rossi. Big data platform for smart grids power consumption anomaly detection. In *2019 Federated Conference on Computer Science and Information Systems (FedCSIS)*, pages 771–780, 2019

I designed the study, supervised the student for the system design, implementation, data analysis. Contributed to the writing. Contribution ∼**30%**.

3.1.2 Smart Grids Co-Simulations

The area of co-simulation is one of the most interesting aspects in the context of Smart Grids evaluation, as highlighted from the review we conducted about data analysis in the context of SGs [102]. Integrating renewable sources, communication, and power networks with information and communication technologies is one of the main challenges in Smart Grids (SG) for large-scale testing [84]. To test for the multiple layers explained in the SGAM model (Fig 3.1), we need to let multiple systems interact, with the need to simulate and emulate part of the hardware and software devices. Different simulators, each running in their runtime environment, are commonly referred to as the concept of *co-simulation*. A co-simulator allows the connection of multiple software simulators and hardware emulators to enable multiple unified simulation scenarios [120, 84].

In this work [84], we provided one large Systematic Mapping Study to define how co-simulations has been used in the context of Smart Grids. For this reason, the coupling of simulators is commonly used to dynamically simulate several aspects of the SG infrastructure in the so-called co-simulations. We provided a scoping review of research of co-simulations in the context of Smart Grids: focusing on the research areas and problems addressed by co-simulations, any specific co-simulation aspects that are the focus of research, the coupling of simulators in co-simulation studies. We mapped the problem addressed in different areas reliability and wide-area awareness, customer energy efficiency, energy resource distribution, grid energy storage, electric Transportation, Advanced Metering Infrastructure, Management of the distribution grid, Cybersecurity, and Network communications. While several co-simulation frameworks exist (e.g., HELICS [97], Daccosim-NG [32]), we mainly adopted the Mosaik framework [110] based on discrete-events and an ad-hoc architecture for our research.

> **Article:** [84] P. Mihal, M. Schvarcbacher, B. Rossi, and T. Pitner. Smart grids co-simulations: Survey & research directions. *Sustainable Computing: Informatics and Systems*, 35:100726, 2022
>
> I designed the study, participated to the review analysis of the model. Contributed to the writing for the journal article. Contribution ∼**30%**.

Due to the complexity of the Smart Grids infrastructure, many Smart Grids laboratories have emerged over the years to provide partially virtualized environments for simulations and testing (e.g., Erigrid [48]). However, the costs for setting-up such environments are quite high, representing a huge barrier for newcomers and educational purposes.
In this work [109], we investigated a hardware-in-the-loop (HIL) architectural solution based on Arduino and Raspberry PI boards, supported by the Mosaik framework to simulate different Smart Grids scenarios on a small and cost-effective scale. We highlighted the educational benefits the solution can bring for understanding simulations and HIL in an affordable & effective way in an easy-to-deploy environment. Furthermore, we had some modifications to the Mosaik co-sim framework - we studied the possibilities of simulating node failure scenarios with a modification of the Mosaik co-simulation platform to allow for dynamic topologies changes. We show how co-simulations can help determine the impact of different failure patterns using a sample scenario of households and PV units [45].

> **Article:** [109] M. Schvarcbacher and B. Rossi. Smart grids co-simulations with low-cost hardware. In *2017 43rd Euromicro Conference on Software Engineering and Advanced Applications (SEAA)*, pages 252–255, 2017
>
> I designed the study, supervised the student for the creation of the prototype and the data analysis design. Contributed to the writing for the conference paper. Contribution ∼**40%**.

This led to the integration of a platform for testing Smart Grids. As a first step, we defined the need for introducing a Testing Management Framework [50] – the framework is

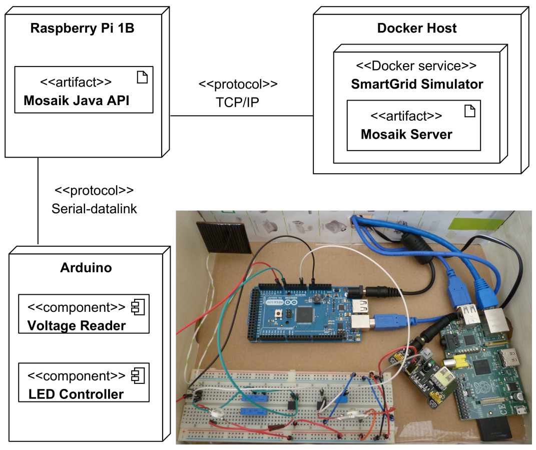

Figure 3.3: Smart Grids simulation with low cost hardware [109]

based on the concept of the definition of risks. For this reason, we investigated risk management in the Smart Grids infrastructure and the definition of a testing process that could be adopted in the platform.
In this work [65], we collected common suggestions from practices and other frameworks to provide a common risk management framework for Smart Grids. In particular, we focused on planning, identifying, assessing, prioritizing, monitoring, and controlling security risks involved in Smart Grids management. Following the SGAM model [18], the Smart Grids' multiple layers and domains need to be considered to accurately predict, manage, and mitigate the risks involved in Smart Grids management. The proposed recommendations cover traditional risk management activities, such as defining the purpose and scope of the risk assessment, developing risk models, and defining threats and impact analysis. These aspects should consider the communication layers with many interactions, protocols, and devices with a definition of recommended security mechanisms such as integrated security for field devices, secure access control, and secure communication protocols.

Article: [65] V. Lamba, N. Šimková, and B. Rossi. Recommendations for smart grid security risk management. *Cyber-Physical Systems*, 5(2):92–118, 2019

I designed the study, supervised the student. Contributed to the writing and revision for the journal article. Contribution ∼**30%**.

This led to integrating different parts in the so-called Smart Grids Testing Management Platform (SGTMP). Software Testing is crucial to ensure reliable and safe systems [126]. However, testing in the context of SGs is highly complex due to the multi-layered structure of the infrastructure. Each layer, responsible for different areas such as communication protocols and service provision, must be tested individually and as a whole to guarantee a faultless and dependable energy supply [18]. Additionally, as SGs are cyber-physical systems

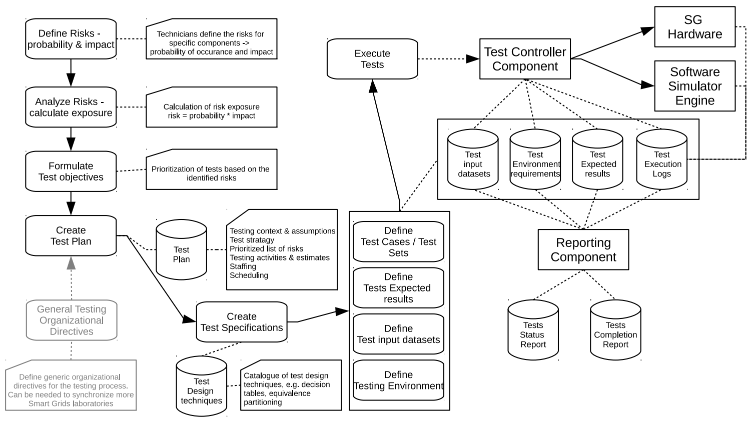
Define Risks - probability & impact
Technicians define the risks for specific components -> probability of occurance and impact
Analyze Risks - calculate exposure
Calculation of risk exposure risk = probability * impact
Formulate Test objectives
Prioritization of tests based on the identified risks
Create Test Plan
Test Plan
Testing context & assumptions
Test stratagy
Prioritized list of risks
Testing activities & estimates
Staffing
Scheduling
General Testing Organizational Directives
Define generic organizational directives for the testing process. Can be needed to synchronize more Smart Grids laboratories
Create Test Specifications
Test Design techniques
Catalogue of test design techniques, e.g. decision tables, equivalence partitioning
Define Test Cases / Test Sets
Define Tests Expected results
Define Test input datasets
Define Testing Environment
Execute Tests
Test Controller Component
SG Hardware
Software Simulator Engine
Test input datasets
Test Environment requirements
Test Expected results
Test Execution Logs
Reporting Component
Tests Status Report
Tests Completion Report

Figure 3.4: Proposed Testing process for SGTMP, based on ISO/IEC/IEEE 29119 [108]

that integrate with hardware devices, modeling, and simulations are essential to detecting integration issues on a large scale [96, 60]. Many SG testing/simulation platforms emerged over the years, focusing on simulating part of the complexity within SGs, both at the power and at the communication network level [120]: SmartGridLab [113], GridSim [6], Smart-Grid Common Open Research Emulator (SCORE) [116], Smart Grid Testbed [7], and GridLAB-D [20] are some of the main platforms available. The main goal of such platforms is to provide a common testbed for SG implementation capabilities supported by "co-simulations": the evaluation of multiple independent simulations integrated using a software interface [120].

In this work [108], we proposed a platform, Smart Grid Testing Management Platform

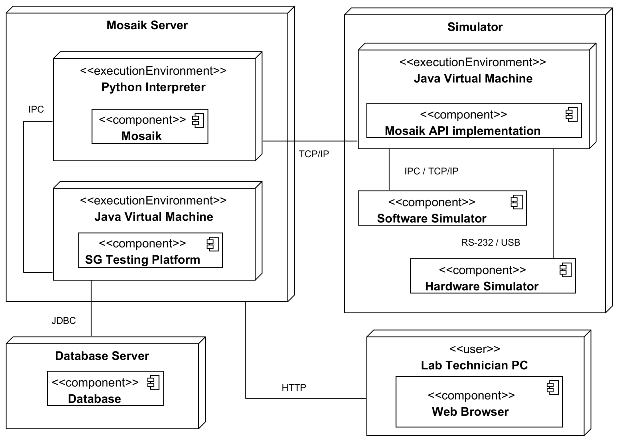

Figure 3.5: SGTMP Platform for the integration of Mosaik simulator [108]

(SGTMP), to take into account co-simulations for reducing the complexity of testing in a Smart Grids infrastructure. SGTMP can allow the execution of real-time hardware-in-the-loop SG tests and experiments that can simplify the testing process in the context of interconnected SG devices. We provided the context of usage, the system architecture, the interactive web-based interface, the provided API, and the integration with co-simulation frameworks to provide virtualized environments for testing. Furthermore, we presented one main scenario for stress-testing SG devices that can showcase the applicability of the platform. SGTMP comprises several components, mainly the Mosaik Interface written in Python and the overall Java-based Testing Platform for the support, including test management and execution, viewing test/simulation results, and the web interface. For the developers, there is an extension of the Mosaik high-level Java API to integrate existing simulators into this system.

[108] M. Schvarcbacher, K. Hrabovská, B. Rossi, and T. Pitner. Smart grid testing management platform (sgtmp). *Applied Sciences*, 8(11), 2018

I supervised both students. Together we designed the platform. Structured the article and contributed to the writing of the journal article. Contribution ∼**30%**.

3.1.3 Ongoing Research Directions

Several research directions are worth investigating in the application domain of Smart Grids:

- *Anomaly Detection*: The models implemented, and the Big Data Platform for anomaly detection were separated from the SGTMP platform created to support co-simulations running by means of the Mosaik framework. One research direction can be integrating the two platforms, allowing anomaly detection on the data collected from the SGTMP platform. This can bring a useful scenario in which data from various simulation environments can be analyzed for anomalies, supporting not only multiple algorithms that can be plugged-in but also potentially supporting large data streams, as the experiments on the platforms showed that it can scale on a massive number of data points with adequate reliability. Furthermore, a more extensive evaluation of anomaly detection algorithms when plugged into the platform would be useful. This will imply the evaluation and implementation of algorithms reviewed in some of the articles included in this Habilitation Thesis (e.g., [102]). For experimentation purposes, the platform also contained a data densification part [74] that can be improved based on the needs of the implemented algorithms.

- *Co-simulations*: As mentioned in Section 2.1.7 about ongoing research directions in software systems' qualities, the knowledge acquired in the research about co-simulations can be exploited for the evaluation of Microservice-based systems to improve self-adaptive properties, in particular the availability of the systems based on the models for software reliability discussed in the previous chapter. The plan is to use

Microservice simulators for deriving properties of runtime systems. Another direction is to refactor and revise the SGTMP platform and integrate it with the newest versions of the Mosaik framework to run more accurate experiments about the system's adoption.

Chapter 4

Conclusion

This Habilitation Thesis is structured as a commented collection of 17 selected peer-reviewed publications: 11 conference papers and 6 journal papers. These represent a selected list to showcase my research efforts in nine years in the period 2014-2022.

The area of software quality has evolved during the last years, requiring higher levels of automation. Furthermore, aspects such as the scalability of the software solutions and the software development models have acquired key importance. There is a discussion of several contributions in this direction, under the light that process improvement can lead to software solutions improvements. Starting with the discussion and an action research on scaling agile software development to understand which practices can lead to more qualitative software development processes. The automation of processes such as source code reviews, technical debt identification, mutation testing, software defect severity identification, and automated triaging were discussed in the context of process improvement. Adopting models such as Software Reliability Growth Models (SRGMs) with tool support can provide a way to understand the cumulative failure rate of software projects towards improving the processes and final quality of the software products. Finally, several contributions were made in the area of software quality for software engineering education, namely, through quasi-experimental designs understanding which are the aspects that students find more challenging and more useful to adopt high-quality practices and principles in their software projects. The application domain was the area of Smart Grids, in which the contribution was in the area of data analysis anomaly detection based on experience in the software engineering automation domain. Due to the Smart Grids infrastructure's complexity, testing has to consider the interactions between the different layers. For this reason, we adopted and proposed a model and platform based on the Mosaik frameworks for testing based on the concept of co-simulations. To keep in line with the educational aspects, we also proposed a low-cost solution that can allow us to more easily understand the concepts of co-simulations with the support of commodity hardware.

The full texts of the articles are excluded from the public version of this thesis collection to avoid copyright violation.

Bibliography

[1] C. C. Aggarwal. *An Introduction to Outlier Analysis*, pages 1–34. Springer International Publishing, Cham, 2017.

[2] M. Alqudah and R. Razali. A review of scaling agile methods in large software development. *International Journal on Advanced Science, Engineering and Information Technology*, 6(6):828–837, 2016.

[3] M. Amin. Challenges in reliability, security, efficiency, and resilience of energy infrastructure: Toward smart self-healing electric power grid. In *Power and Energy Society General Meeting-Conversion and Delivery of Electrical Energy in the 21st Century*, pages 1–5. IEEE, 2008.

[4] S. M. Amin and B. F. Wollenberg. Toward a smart grid: power delivery for the 21st century. *IEEE power and energy magazine*, 3(5):34–41, 2005.

[5] E. Ancillotti, R. Bruno, and M. Conti. The role of communication systems in smart grids: Architectures, technical solutions and research challenges. *Computer Communications*, 36(17-18):1665–1697, 2013.

[6] D. Anderson, C. Zhao, C. Hauser, V. Venkatasubramanian, D. Bakken, and A. Bose. Intelligent design" real-time simulation for smart grid control and communications design. *IEEE Power and Energy Magazine*, 10(1):49–57, 2012.

[7] M. Annor-Asante and B. Pranggono. Development of smart grid testbed with low-cost hardware and software for cybersecurity research and education. *Wireless Personal Communications*, pages 1–21, 2018.

[8] O. Ardakanian, N. Koochakzadeh, R. P. Singh, L. Golab, and S. Keshav. Computing electricity consumption profiles from household smart meter data. In *EDBT/ICDT Workshops*, volume 14, pages 140–147, 2014.

[9] D. E. Avison, F. Lau, M. D. Myers, and P. A. Nielsen. Action research. *Communications of the ACM*, 42(1):94–97, 1999.

[10] B. W. Boehm. Trw systems engineering and integration division. characteristics of software quality. *TRW Software Series: TRW-SS*, 1973.

[11] B. W. Boehm, J. R. Brown, and M. Lipow. Quantitative evaluation of software quality. In *Proceedings of the 2nd international conference on Software engineering*, pages 592–605, 1976.

[12] R. Bonetto and M. Rossi. Smart grid for the smart city. In *Designing, Developing, and Facilitating Smart Cities*, pages 241–263. Springer, 2017.

[13] D. M. Breuker, J. Derriks, and J. Brunekreef. Measuring static quality of student code. In *Proceedings of the 16th annual joint conference on Innovation and technology in computer science education*, pages 13–17, 2011.

[14] D. Budgen and P. Brereton. Performing systematic literature reviews in software engineering. In *Proceedings of the 28th international conference on Software engineering*, pages 1051–1052, 2006.

[15] M. M. Buzau, J. Tejedor-Aguilera, P. Cruz-Romero, and A. Gómez-Expósito. Detection of non-technical losses using smart meter data and supervised learning. *IEEE Transactions on Smart Grid*, 10(3):2661–2670, 2018.

[16] D. T. Campbell and J. C. Stanley. *Experimental and quasi-experimental designs for research.* Ravenio books, 2015.

[17] J. C. Carver and R. Prikladnicki. Industry–academia collaboration in software engineering. *IEEE Software*, 35(5):120–124, 2018.

[18] S. G. C. CEN-CENELEC-ETSI. Group.(2012). *Smart Grid Reference Architecture*, pages 1–107, 2012.

[19] V. Chandola, A. Banerjee, and V. Kumar. Anomaly detection: A survey. *ACM computing surveys (CSUR)*, 41(3):15, 2009.

[20] D. P. Chassin, K. Schneider, and C. Gerkensmeyer. Gridlab-d: An open-source power systems modeling and simulation environment. In *Transmission and distribution conference and exposition, 2008. t&d. IEEE/PES*, pages 1–5. IEEE, 2008.

[21] J.-S. Chou and A. S. Telaga. Real-time detection of anomalous power consumption. *Renewable and Sustainable Energy Reviews*, 33:400–411, 2014.

[22] S. Chren, B. Buhnova, M. Macak, L. Daubner, and B. Rossi. Mistakes in uml diagrams: Analysis of student projects in a software engineering course. In *2019 IEEE/ACM 41st International Conference on Software Engineering: Software Engineering Education and Training (ICSE-SEET)*, pages 100–109, 2019.

[23] S. Chren, M. Macák, B. Rossi, and B. Buhnova. Evaluating code improvements in software quality course projects. In *Proceedings of the International Conference on Evaluation and Assessment in Software Engineering 2022*, EASE '22, page 160–169, New York, NY, USA, 2022. Association for Computing Machinery.

[24] S. Chren, R. Micko, B. Buhnova, and B. Rossi. Strait: A tool for automated software reliability growth analysis. In *2019 IEEE/ACM 16th International Conference on Mining Software Repositories (MSR)*, pages 105–110, 2019.

[25] S. Chren, B. Rossi, and T. Pitner. Smart grids deployments within eu projects: The role of smart meters. In *2016 Smart Cities Symposium Prague (SCSP)*, pages 1–5, 2016.

[26] V. Dedík and B. Rossi. Automated bug triaging in an industrial context. In *2016 42th Euromicro Conference on Software Engineering and Advanced Applications (SEAA)*, pages 363–367, 2016.

[27] T. Dingsøyr and N. B. Moe. Towards principles of large-scale agile development. In *International Conference on Agile Software Development*, pages 1–8. Springer, 2014.

[28] J. T. Duane. Learning curve approach to reliability monitoring. *IEEE Transactions on Aerospace*, pages 563–566, 1964.

[29] T. Dyba, B. A. Kitchenham, and M. Jorgensen. Evidence-based software engineering for practitioners. *IEEE software*, 22(1):58–65, 2005.

[30] C. Ebert and M. Paasivaara. Scaling agile. *IEEE Software*, 34(6):98–103, November 2017.

[31] M. Eremia, L. Toma, and M. Sanduleac. The smart city concept in the 21st century. *Procedia Engineering*, 181:12–19, 2017.

[32] J. Évora Gómez, J. J. Hernández Cabrera, J.-P. Tavella, S. Vialle, E. Kremers, and L. Frayssinet. Daccosim ng: co-simulation made simpler and faster. In *Linköping electronic conference proceedings*, 2019.

[33] X. Fang, S. Misra, G. Xue, and D. Yang. Smart grid—the new and improved power grid: A survey. *Communications Surveys & Tutorials, IEEE*, 14(4):944–980, 2012.

[34] H. Farhangi. The path of the smart grid. *IEEE power and energy magazine*, 8(1), 2010.

[35] N. Fenton. Software measurement: A necessary scientific basis. *IEEE Transactions on software engineering*, 20(3):199–206, 1994.

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

[37] N. E. Fenton and M. Neil. Software metrics: roadmap. In *Proceedings of the Conference on the Future of Software Engineering*, pages 357–370, 2000.

[38] G. Fenza, M. Gallo, and V. Loia. Drift-aware methodology for anomaly detection in smart grid. *IEEE Access*, 7:9645–9657, 2019.

[39] M. Frincu and Z. Gima. A bottom-up approach to sustained curtailment and comfort for controlled demand response. In *2014 IEEE Conference on Technologies for Sustainability (SUSTECH)*, pages 61–68. IEEE Oregon Sect; REGION 6; IEEE USA; IEEE Consumer Elect Soc; IEEE Soc Social Implications Technology, 2014.

[40] M. Ge, S. Chren, B. Rossi, and T. Pitner. Data quality management framework for smart grid systems. In W. Abramowicz and R. Corchuelo, editors, *Business Information Systems*, pages 299–310, Cham, 2019. Springer International Publishing.

[41] K. Geisler. The relationship between smart grids and smart cities. *IEEE newsletter, May*, 2013.

[42] D. Gelperin and B. Hetzel. The growth of software testing. *Communications of the ACM*, 31(6):687–695, 1988.

[43] A. L. Goel and K. Okumoto. Time-dependent error-detection rate model for software reliability and other performance measures. *IEEE Transactions on Reliability*, R-28(3):206–211, 1979.

[44] S. S. Gokhale and K. S. Trivedi. Log-logistic software reliability growth model. In *Proceedings Third IEEE International High-Assurance Systems Engineering Symposium (Cat. No.98EX231)*, pages 34–41, 1998.

[45] L. Gryga and B. Rossi. Co-simulation of smart grids: Dynamically changing topologies in failure scenarios. In *Proceedings of the 6th International Conference on Complexity, Future Information Systems and Risk - Volume 1: COMPLEXIS,*, pages 63–69. INSTICC, SciTePress, 2021.

[46] S. Hamer, C. Quesada-López, and M. Jenkins. Students projects' source code changes impact on software quality through static analysis. In *Quality of Information and Communications Technology*, pages 553–564. Springer International Publishing, 2021.

[47] A. E. Hassan. The road ahead for mining software repositories. In *2008 frontiers of software maintenance*, pages 48–57. IEEE, 2008.

[48] K. Heussen, C. Steinbrink, I. F. Abdulhadi, V. H. Nguyen, M. Z. Degefa, J. Merino, T. V. Jensen, H. Guo, O. Gehrke, D. E. M. Bondy, et al. Erigrid holistic test description for validating cyber-physical energy systems. *Energies*, 12(14):2722, 2019.

[49] S. A. Hossain and R. C. Dahiya. Estimating the parameters of a non-homogeneous poisson-process model for software reliability. *IEEE Transactions on Reliability*, pages 604–612, 1993.

[50] K. Hrabovská, B. Rossi, and T. Pitner. Software testing process models benefits & drawbacks: a systematic literature review, 2019.

[51] International Standard Organization (ISO). International standard iso/iec 9126, information technology - product quality - part1: Quality model, 2001.

[52] ISO/IEC 25010. ISO/IEC 25010:2011, systems and software engineering — systems and software quality requirements and evaluation (square) — system and software quality models, 2011.

[53] M. L. Jaccheri. Software quality and software process improvement course based on interaction with the local software industry. *Computer Applications in Engineering Education*, 9(4):265–272, 2001.

[54] Z. Jelinski and P. Moranda. Software reliability research. In *Statistical computer performance evaluation*, pages 465–484. Elsevier, 1972.

[55] G. Jeong, S. Kim, T. Zimmermann, and K. Yi. Improving code review by predicting reviewers and acceptance of patches. *Research on Software Analysis for Error-free Computing Center Tech-Memo (ROSAEC MEMO 2009-006)*, pages 1–18, 2009.

[56] J. Jiang, J.-H. He, and X.-Y. Chen. Coredevrec: Automatic core member recommendation for contribution evaluation. *Journal of Computer Science and Technology*, 30(5):998–1016, 2015.

[57] M. Kalenda, P. Hyna, and B. Rossi. Scaling agile in large organizations: Practices, challenges, and success factors. *Journal of Software: Evolution and Process*, 30(10):e1954, 2018. e1954 smr.1954.

[58] N. Kanti-Singha Roy and B. Rossi. Towards an improvement of bug severity classification. In *2014 40th Euromicro Conference on Software Engineering and Advanced Applications*, pages 269–276, 2014.

[59] H. Keuning, B. Heeren, and J. Jeuring. Code quality issues in student programs. In *Proceedings of the 2017 ACM Conference on Innovation and Technology in Computer Science Education*, pages 110–115, 2017.

[60] S. K. Khaitan and J. D. McCalley. Design techniques and applications of cyberphysical systems: A survey. *IEEE Systems Journal*, 9(2):350–365, 2015.

[61] H. A. Khan, A. C. M. Tan, Y. Xiao, V. Sreeram, and H. H. C. Iu. An implementation of novel CMAC algorithm for very short term load forecasting. *Journal of Ambient Intelligence and Humanized Computing*, 4(6):673–683, 2013.

[62] B. Kitchenham and S. L. Pfleeger. Software quality: the elusive target [special issues section]. *IEEE software*, 13(1):12–21, 1996.

[63] B. A. Kitchenham, T. Dyba, and M. Jorgensen. Evidence-based software engineering. In *Proceedings. 26th International Conference on Software Engineering*, pages 273–281. IEEE, 2004.

[64] F. Koetter, M. Kochanowski, M. Kintz, B. Kersjes, I. Bogicevic, and S. Wagner. Assessing software quality of agile student projects by data-mining software repositories. In *CSEDU (2)*, pages 244–251, 2019.

[65] V. Lamba, N. Šimková, and B. Rossi. Recommendations for smart grid security risk management. *Cyber-Physical Systems*, 5(2):92–118, 2019.

[66] A. Lamkanfi, S. Demeyer, E. Giger, and B. Goethals. Predicting the severity of a reported bug. In *MSR*, pages 1–10, 2010.

[67] A. Lamkanfi, S. Demeyer, Q. Soetens, and T. Verdonck. Comparing mining algorithms for predicting the severity of a reported bug. In *2011 15th European Conference on Software Maintenance and Reengineering (CSMR)*, pages 249–258, 2011.

[68] C. Larman and B. Vodde. *Practices for scaling lean & Agile development: large, multisite, and offshore product development with large-scale scrum.* Pearson Education, 2010.

[69] M. M. Lehman. On understanding laws, evolution, and conservation in the large-program life cycle. *Journal of Systems and Software*, 1:213–221, 1979.

[70] M. M. Lehman, J. F. Ramil, P. D. Wernick, D. E. Perry, and W. M. Turski. Metrics and laws of software evolution-the nineties view. In *Proceedings Fourth International Software Metrics Symposium*, pages 20–32. IEEE, 1997.

[71] J.-L. Letouzey. The sqale method for evaluating technical debt. In *Third International Workshop on Managing Technical Debt (MTD)*, pages 31–36. IEEE, 2012.

[72] J. Lin. The lambda and the kappa. *IEEE Internet Computing*, 21(5):60–66, 2017.

[73] J. Lipcak and B. Rossi. A large-scale study on source code reviewer recommendation. In *2018 44th Euromicro Conference on Software Engineering and Advanced Applications (SEAA)*, pages 378–387, 2018.

[74] P. Lipčák, M. Macak, and B. Rossi. Big data platform for smart grids power consumption anomaly detection. In *2019 Federated Conference on Computer Science and Information Systems (FedCSIS)*, pages 771–780, 2019.

[75] X. Liu and P. S. Nielsen. Regression-based online anomaly detection for smart grid data. *arXiv preprint arXiv:1606.05781*, 2016.

[76] Y. Lu, X. Mao, T. Wang, G. Yin, and Z. Li. Improving students' programming quality with the continuous inspection process: a social coding perspective. *Frontiers of Computer Science*, 14(5):1–18, 2020.

[77] M. R. Lyu. *Handbook of software reliability engineering.* IEEE Computer Society Press, 1996.

[78] N. Marz and J. Warren. *Big Data: Principles and best practices of scalable real-time data systems.* New York; Manning Publications Co., 2015.

[79] J. A. McCall, P. K. Richards, and G. F. Walters. Factors in software quality. volume i. concepts and definitions of software quality. Technical report, General Electric CO Synnyvale, CA, 1977.

[80] P. McDaniel and S. McLaughlin. Security and privacy challenges in the smart grid. *IEEE Security & Privacy*, 7(3), 2009.

[81] N. R. Mead, T. B. Hilburn, and R. C. Linger. Software assurance curriculum project volume 2: Undergraduate course outlines. Technical report, Carnegie-Mellon University Pittsburgh PA Software Engineering Institute, 2010.

[82] T. Mens. On the complexity of software systems. *Computer*, 45(08):79–81, 2012.

[83] T. Menzies and A. Marcus. Automated severity assessment of software defect reports. In *Software Maintenance, 2008. ICSM 2008. IEEE International Conference on*, pages 346–355, 2008.

[84] P. Mihal, M. Schvarcbacher, B. Rossi, and T. Pitner. Smart grids co-simulations: Survey & research directions. *Sustainable Computing: Informatics and Systems*, 35:100726, 2022.

[85] R. Mičko, S. Chren, and B. Rossi. Applicability of software reliability growth models to open source software. In *2022 48th Euromicro Conference on Software Engineering and Advanced Applications (SEAA)*, pages 255–262, 2022.

[86] N. B. Moe and T. Dingsøyr. Emerging research themes and updated research agenda for large-scale agile development: a summary of the 5th international workshop at xp2017. In *Proceedings of the XP2017 Scientific Workshops*, page 14. ACM, 2017.

[87] J. Možucha and B. Rossi. Is mutation testing ready to be adopted industry-wide? In P. Abrahamsson, A. Jedlitschka, A. Nguyen Duc, M. Felderer, S. Amasaki, and T. Mikkonen, editors, *Product-Focused Software Process Improvement*, pages 217–232, Cham, 2016. Springer International Publishing.

[88] J. D. Musa. A theory of software reliability and its application. *IEEE transactions on software engineering*, 1(03):312–327, 1975.

[89] J. D. Musa, A. Iannino, and K. Okumoto. *Software reliability: measurement, prediction, application.* McGraw-Hill, Inc., 1987.

[90] J. D. Musa and K. Okumoto. A logarithmic poisson execution time model for software reliability measurement. In *Proceedings of the 7th International Conference on Software Engineering*, pages 230–238, Piscataway, NJ, USA, 1984. IEEE Press.

[91] NIST. Nist framework and roadmap for smart grid interoperability standards, release 1.0 (draft), jan. *Dept. of Commerce, USA, Framework and Roadma*, 2010.

[92] NIST. Roadmap for smart grid interoperability standards, release 2.0. *NIST special publication 1108R2*, 2012.

[93] A. Nugroho, J. Visser, and T. Kuipers. An empirical model of technical debt and interest. In *Proceedings of the 2Nd Workshop on Managing Technical Debt*, MTD '11, pages 1–8, New York, NY, USA, 2011. ACM.

[94] P. Oman and J. Hagemeister. Construction and testing of polynomials predicting software maintainability. *Journal of Systems and Software*, 24(3):251 – 266, 1994. Oregon Workshop on Software Metrics.

[95] P. Pääkkönen and D. Pakkala. Reference architecture and classification of technologies, products and services for big data systems. *Big data research*, 2(4):166–186, 2015.

[96] P. Palensky, E. Widl, and A. Elsheikh. Simulating cyber-physical energy systems: Challenges, tools and methods. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 44(3):318–326, 2014.

[97] B. Palmintier, D. Krishnamurthy, P. Top, S. Smith, J. Daily, and J. Fuller. Design of the helics high-performance transmission-distribution-communication-market co-simulation framework. In *2017 Workshop on Modeling and Simulation of Cyber-Physical Energy Systems (MSCPES)*, pages 1–6. IEEE, 2017.

[98] K. Petersen, R. Feldt, S. Mujtaba, and M. Mattsson. Systematic mapping studies in software engineering. In *12th International Conference on Evaluation and Assessment in Software Engineering (EASE) 12*, pages 1–10, 2008.

[99] P. Ralph, N. b. Ali, S. Baltes, D. Bianculli, J. Diaz, Y. Dittrich, N. Ernst, M. Felderer, R. Feldt, A. Filieri, et al. Empirical standards for software engineering research. *arXiv preprint arXiv:2010.03525*, 2020.

[100] D. J. Reifer, F. Maurer, and H. Erdogmus. Scaling agile methods. *IEEE software*, 20(4):12–14, 2003.

[101] B. Rossi. *Towards a Simulation Model including Network Externalities in Free / Libre Open Source Software (FLOSS) Adoption.* PhD thesis, Free University of Bozen-Bolzano, 2007.

[102] B. Rossi and S. Chren. Smart grids data analysis: A systematic mapping study. *IEEE Transactions on Industrial Informatics*, 16(6):3619–3639, 2020.

[103] B. Rossi, S. Chren, B. Buhnova, and T. Pitner. Anomaly detection in smart grid data: An experience report. In *2016 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, pages 2313–2318, 2016.

[104] B. Rossi et al. A large-scale replication of smart grids power consumption anomaly detection. In *IoTBDS*, pages 288–295, 2020.

[105] B. Rossi, B. Russo, and G. Succi. Modelling failures occurrences of open source software with reliability growth. In P. Ågerfalk, C. Boldyreff, J. M. González-Barahona, G. R. Madey, and J. Noll, editors, *Open Source Software: New Horizons*, pages 268–280, Berlin, Heidelberg, 2010. Springer Berlin Heidelberg.

[106] A. Saad and N. Sisworahardjo. Data analytics-based anomaly detection in smart distribution network. In *2017 International Conference on High Voltage Engineering and Power Systems (ICHVEPS)*, pages 1–5. IEEE, 2017.

[107] I. Scaled Agile. Safe 4.0 introduction a scaled agile, inc. white paper july 2016 overview of the scaled agile framework for lean software and systems engineering. Technical report, Scaled Agile, Inc., 5480 Valmont Rd, Suite 100, Boulder CO 80301 USA, July 2016.

[108] M. Schvarcbacher, K. Hrabovská, B. Rossi, and T. Pitner. Smart grid testing management platform (sgtmp). *Applied Sciences*, 8(11), 2018.

[109] M. Schvarcbacher and B. Rossi. Smart grids co-simulations with low-cost hardware. In *2017 43rd Euromicro Conference on Software Engineering and Advanced Applications (SEAA)*, pages 252–255, 2017.

[110] S. Schütte, S. Scherfke, and M. Tröschel. Mosaik: A framework for modular simulation of active components in smart grids. In *2011 IEEE First International Workshop on Smart Grid Modeling and Simulation (SGMS)*, pages 55–60, Oct 2011.

[111] F. J. Shull, J. C. Carver, S. Vegas, and N. Juristo. The role of replications in empirical software engineering. *Empirical software engineering*, 13:211–218, 2008.

[112] A. Sial, A. Singh, A. Mahanti, and M. Gong. Heuristics-based detection of abnormal energy consumption. In *International Conference on Smart Grid Inspired Future Technologies*, pages 21–31. Springer, 2018.

[113] W.-Z. Song, D. De, S. Tan, S. K. Das, and L. Tong. A wireless smart grid testbed in lab. *IEEE Wireless Communications*, 19(3), 2012.

[114] M. Staron. *Action research in software engineering.* Springer, 2020.

[115] P. Strečanský, S. Chren, and B. Rossi. Comparing maintainability index, sig method, and sqale for technical debt identification. *Scientific Programming*, 2020:14, Jul 2020.

[116] S. Tan, W.-Z. Song, Q. Dong, and L. Tong. Score: Smart-grid common open research emulator. In *Smart Grid Communications (SmartGridComm), 2012 IEEE Third International Conference on*, pages 282–287. IEEE, 2012.

[117] P. Thongtanunam, C. Tantithamthavorn, R. G. Kula, N. Yoshida, H. Iida, and K.-i. Matsumoto. Who should review my code? a file location-based code-reviewer recommendation approach for modern code review. In *Software Analysis, Evolution*

and Reengineering (SANER), 2015 IEEE 22nd International Conference on, pages 141–150. IEEE, 2015.

[118] V. Tomic. A bionic view on complex software systems-and the consequences for digital resilience. Master's thesis, Wien, 2021.

[119] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[120] M. Vogt, F. Marten, and M. Braun. A survey and statistical analysis of smart grid co-simulations. *Applied Energy*, 222:67–78, 2018.

[121] M. Wolsink. The research agenda on social acceptance of distributed generation in smart grids: Renewable as common pool resources. *Renewable and Sust. Energy Reviews*, 16(1):822–835, 2012.

[122] A. Wood. Software reliability growth models. Technical report, Tandem Computers Inc., Cupertino, CA 95014, 1996.

[123] R. Włodarski, A. Poniszewska-Marańda, and J.-R. Falleri. Impact of software development processes on the outcomes of student computing projects: A tale of two universities. *Information and Software Technology*, 144:106787, 2022.

[124] S. Yamada, M. Ohba, and S. Osaki. S-shaped reliability growth modeling for software error detection. *IEEE Transactions on Reliability*, R-32(5):475–484, 1983.

[125] C.-Z. Yang, C.-C. Hou, W.-C. Kao, and I.-X. Chen. An empirical study on improving severity prediction of defect reports using feature selection. In *Software Engineering Conference (APSEC), 2012 19th Asia-Pacific*, volume 1, pages 240–249, 2012.

[126] M. Young. *Software testing and analysis: process, principles, and techniques.* John Wiley & Sons, 2008.

[127] Y. Yu, H. Wang, G. Yin, and T. Wang. Reviewer recommendation for pull-requests in github: What can we learn from code review and bug assignment? *Information and Software Technology*, 74:204–218, 2016.

[128] Y. Zhang, W. Chen, and J. Black. Anomaly detection in premise energy consumption data. In *2011 IEEE Power and Energy Society General Meeting*, pages 1–8. IEEE, 2011.

Part II

Collection of Articles

Appendix A

Selection of Articles

This appendix lists all the articles that have been included in this Habilitation Thesis. Acronyms in the index distinguish between journal (J) and conference (C) papers. The full-texts of the articles follow this section[1].

A.1 (C1.) J. Lipcak and B. Rossi. A large-scale study on source code reviewer recommendation. In *2018 44th Euromicro Conference on Software Engineering and Advanced Applications (SEAA)*, pages 378–387, 2018

A.2 (C2.) J. Možucha and B. Rossi. Is mutation testing ready to be adopted industry-wide? In P. Abrahamsson, A. Jedlitschka, A. Nguyen Duc, M. Felderer, S. Amasaki, and T. Mikkonen, editors, *Product-Focused Software Process Improvement*, pages 217–232, Cham, 2016. Springer International Publishing

A.3 (C3.) N. Kanti-Singha Roy and B. Rossi. Towards an improvement of bug severity classification. In *2014 40th Euromicro Conference on Software Engineering and Advanced Applications*, pages 269–276, 2014

A.4 (C4.) V. Dedík and B. Rossi. Automated bug triaging in an industrial context. In *2016 42th Euromicro Conference on Software Engineering and Advanced Applications (SEAA)*, pages 363–367, 2016

A.5 (C5.) S. Chren, R. Micko, B. Buhnova, and B. Rossi. Strait: A tool for automated software reliability growth analysis. In *2019 IEEE/ACM 16th International Conference on Mining Software Repositories (MSR)*, pages 105–110, 2019

A.6 (C6.) R. Mičko, S. Chren, and B. Rossi. Applicability of software reliability growth models to open source software. In *2022 48th Euromicro Conference on Software Engineering and Advanced Applications (SEAA)*, pages 255–262, 2022

[1]full-texts are omitted from the public version of the thesis to avoid copyright infrigement.

A.7 (C7.) S. Chren, B. Buhnova, M. Macak, L. Daubner, and B. Rossi. Mistakes in uml diagrams: Analysis of student projects in a software engineering course. In *2019 IEEE/ACM 41st International Conference on Software Engineering: Software Engineering Education and Training (ICSE-SEET)*, pages 100–109, 2019

A.8 (C8.) S. Chren, M. Macák, B. Rossi, and B. Buhnova. Evaluating code improvements in software quality course projects. In *Proceedings of the International Conference on Evaluation and Assessment in Software Engineering 2022*, EASE '22, page 160–169, New York, NY, USA, 2022. Association for Computing Machinery

A.9 (C9.) B. Rossi, S. Chren, B. Buhnova, and T. Pitner. Anomaly detection in smart grid data: An experience report. In *2016 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, pages 2313–2318, 2016

A.10 (C10.) P. Lipčák, M. Macak, and B. Rossi. Big data platform for smart grids power consumption anomaly detection. In *2019 Federated Conference on Computer Science and Information Systems (FedCSIS)*, pages 771–780, 2019

A.11 (C11.) M. Schvarcbacher and B. Rossi. Smart grids co-simulations with low-cost hardware. In *2017 43rd Euromicro Conference on Software Engineering and Advanced Applications (SEAA)*, pages 252–255, 2017

A.12 (J1.) M. Kalenda, P. Hyna, and B. Rossi. Scaling agile in large organizations: Practices, challenges, and success factors. *Journal of Software: Evolution and Process*, 30(10):e1954, 2018. e1954 smr.1954

A.13 (J2.) P. Strečanský, S. Chren, and B. Rossi. Comparing maintainability index, sig method, and sqale for technical debt identification. *Scientific Programming*, 2020:14, Jul 2020

A.14 (J3.) B. Rossi and S. Chren. Smart grids data analysis: A systematic mapping study. *IEEE Transactions on Industrial Informatics*, 16(6):3619–3639, 2020

A.15 (J4.) V. Lamba, N. Šimková, and B. Rossi. Recommendations for smart grid security risk management. *Cyber-Physical Systems*, 5(2):92–118, 2019

A.16 (J5.) P. Mihal, M. Schvarcbacher, B. Rossi, and T. Pitner. Smart grids co-simulations: Survey & research directions. *Sustainable Computing: Informatics and Systems*, 35:100726, 2022

A.17 (J6.) M. Schvarcbacher, K. Hrabovská, B. Rossi, and T. Pitner. Smart grid testing management platform (sgtmp). *Applied Sciences*, 8(11), 2018

A.1 A Large-scale Study on Source Code Reviewer Recommendation (C1)

J. Lipcak and B. Rossi. A large-scale study on source code reviewer recommendation. In *2018 44th Euromicro Conference on Software Engineering and Advanced Applications (SEAA)*, pages 378–387, 2018

Abstract. Context: Software code reviews are an important part of the development process, leading to better software quality and reduced overall costs. However, finding appropriate code reviewers is a complex and time-consuming task. Goals: In this paper, we propose a large-scale study to compare performance of two main source code reviewer recommendation algorithms (RevFinder and a Naive Bayes-based approach) in identifying the best code reviewers for opened pull requests. Method: We mined data from Github and Gerrit repositories, building a large dataset of 51 projects, with more than 293K pull requests analyzed, 180K owners and 157K reviewers. Results: Based on the large analysis, we can state that i) no model can be generalized as best for all projects, ii) the usage of a different repository (Gerrit, GitHub) can have impact on the the recommendation results, iii) exploiting sub-projects information available in Gerrit can improve the recommendation results.

A.2 Is Mutation Testing Ready to be adopted Industry-wide? (C2)

J. Možucha and B. Rossi. Is mutation testing ready to be adopted industry-wide? In P. Abrahamsson, A. Jedlitschka, A. Nguyen Duc, M. Felderer, S. Amasaki, and T. Mikkonen, editors, *Product-Focused Software Process Improvement*, pages 217–232, Cham, 2016. Springer International Publishing

Abstract. Mutation Testing has a long research history as a way to improve the quality of software tests. However, it has not yet reached wide consensus for industry-wide adoption, mainly due to missing clear benefits and computational complexity for the application to large systems. In this paper, we investigate the current state of mutation testing support for Java Virtual Machine (JVM) environments. By running an experimental evaluation, we found out that while default configurations are unbearable for larger projects, using strategies such as selective operators, second order mutation and multi-threading can increase the applicability of the approach. However, there is a trade-off in terms of quality of the achieved results of the mutation analysis process that needs to be taken into account.

A.3 Towards an Improvement of Bug Severity Classification (C3)

N. Kanti-Singha Roy and B. Rossi. Towards an improvement of bug severity classification. In *2014 40th Euromicro Conference on Software Engineering and Advanced Applications*, pages 269–276, 2014

Abstract. Predicting the severity of bugs has been found in past research to improve triaging and the bug resolution process. For this reason, many classification/prediction approaches emerged over the years to provide an automated reasoning over severity classes. In this paper, we use text mining together with bi-grams and feature selection to improve the classification of bugs in severe/non-severe classes. We adopt the Naive Bayes (NB) classifier considering Mozilla and Eclipse datasets commonly used in related works. Overall, the results show that the application of bi-grams can improve slightly the performance of the classifier, but feature selection can be more effective to determine the most informative terms and bi-grams. The results are in any case project-dependent, as in some cases the addition of bi-grams may worsen the performance.

A.4 Automated Bug Triaging in an Industrial Context (C4)

V. Dedík and B. Rossi. Automated bug triaging in an industrial context. In *2016 42th Euromicro Conference on Software Engineering and Advanced Applications (SEAA)*, pages 363–367, 2016

Abstract. There is an increasing need to introduce some form of automation within the bug triaging process, so that no time is wasted on the initial assignment of issues. However, there is a gap in current research, as most of the studies deal with open source projects, ignoring the industrial context and needs. In this paper, we report our experience in dealing with the automation of the bug triaging process within a research-industry cooperation. After reporting the requirements and needs that were set within the industrial project, we compare the analysis results with those from an open source project used frequently in related research (Firefox). In spite of the fact that the projects have different size and development process, the data distributions are similar and the best models as well. We found out that more easily configurable models (such as SVM+TF–IDF) are preferred, and that top-x recommendations, number of issues per developers, and online learning can all be relevant factors when dealing with an industrial collaboration.

A.5 Strait: A Tool for Automated Software Reliability Growth Analysis (C5)

S. Chren, R. Micko, B. Buhnova, and B. Rossi. Strait: A tool for automated software reliability growth analysis. In *2019 IEEE/ACM 16th International Conference on Mining Software Repositories (MSR)*, pages 105–110, 2019

Abstract. Reliability is an essential attribute of mission- and safety-critical systems. Software Reliability Growth Models (SRGMs) are regression-based models that use historical failure data to predict the reliability-related parameters. At the moment, there is no dedicated tool available that would be able to cover the whole process of SRGMs data preparation and application from issue repositories, discouraging replications and reuse in other projects. In this paper, we introduce STRAIT, a free and open source tool for automatic software reliability growth analysis which utilizes data from issue repositories. STRAIT features downloading, filtering and processing of data from provided issue repositories for use in multiple SRGMs, suggesting the best fitting SRGM with multiple data snapshots to consider software evolution. The tool is designed to be highly extensible, in terms of additional issue repositories, SRGMs, and new data filtering and processing options. Quality engineers can use STRAIT for the evaluation of their software systems. The research community can use STRAIT for empirical studies which involve evaluation of new SRGMs or comparison of multiple SRGMs.

A.6 Applicability of Software Reliability Growth Models to Open Source Software (C6)

R. Mičko, S. Chren, and B. Rossi. Applicability of software reliability growth models to open source software. In *2022 48th Euromicro Conference on Software Engineering and Advanced Applications (SEAA)*, pages 255–262, 2022

Abstract. Software Reliability Growth Models (SRGMs) are based on underlying assumptions which make them typically more suited for quality evaluation of closed-source projects and their development lifecycles. Their usage in open-source software (OSS) projects is a subject of debate. Although the studies investigating the SRGMs applicability in OSS context do exist, they are limited by the number of models and projects considered which might lead to inconclusive results. In this paper, we present an experimental study of SRGMs applicability to a total of 88 OSS projects, comparing nine SRGMs, looking at the stability of the best models on the whole projects, on releases, on different domains, and according to different projects' attributes. With the aid of the STRAIT tool, we automated repository mining, data processing, and SRGM analysis for better reproducibility. Overall, we found good applicability of SRGMs to OSS, but with different performance when segmenting the dataset into releases and domains, highlighting the difficulty in generalizing the findings and in the search for one-fits-all models.

A.7 Mistakes in UML Diagrams: Analysis of Student Projects in a Software Engineering Course (C7)

S. Chren, B. Buhnova, M. Macak, L. Daubner, and B. Rossi. Mistakes in uml diagrams: Analysis of student projects in a software engineering course. In *2019 IEEE/ACM 41st International Conference on Software Engineering: Software Engineering Education and Training (ICSE-SEET)*, pages 100–109, 2019

Abstract. The Unified Modelling Language (UML) is being widely accepted as a modelling notation for visualizing software systems during design and development. UML has thus become part of many software engineering course curricula at universities worldwide, providing a recognized tool for practical training of students in understanding and visualizing software design. It is however common that students have difficulties in absorbing UML in its complexity, and often repeat the same mistakes that have been observed by course tutors in previous years. Having a catalogue of such mistakes could hence increase the effectiveness of both teaching and learning of UML diagrams. In this paper, we introduce such a catalogue, consisting of 146 types of mistakes in eight types of diagrams. As the main contribution of this study, we use this catalogue to guide the analysis of student projects within a software engineering course. In total, over 2,700 diagrams submitted over 12 weeks of a semester by 123 students were analysed to identify the frequency of mistakes (from the catalogue), correlations of the mistakes between different diagram types, correlation of the quality of student projects to exam results, student behaviour in terms of introducing and fixing the mistakes over time, and other interesting insights. The analysis is described together with its setup and execution, and all datasets and detailed guidebook to the catalogue of all mistakes is made available for download.

A.8 Evaluating Code Improvements in Software Quality Course Projects (C8)

S. Chren, M. Macák, B. Rossi, and B. Buhnova. Evaluating code improvements in software quality course projects. In *Proceedings of the International Conference on Evaluation and Assessment in Software Engineering 2022*, EASE '22, page 160–169, New York, NY, USA, 2022. Association for Computing Machinery

Abstract. Software quality sits at the core of software engineering as a discipline. Yet, although each university software-engineering and the software development course covers software quality to some extent, practitioners still lament on graduates' readiness for practise for this very reason—poor quality of their code. As a result, we have engaged university industrial partners in designing a master-degree Software Quality course that puts the key software quality topics in one place. In this paper, we report on the effects of the course on the quality of students' coding projects. To this end, we have analysed a total of 54 project submissions from 27 students, with both manual and automated quality assessment methods. We have employed 30 manual and 22 automated quality characteristics related to coding style, architecture design and general development practices. In particular, we examine which characteristics of the code have improved the most and what were the most common issues. Additionally, we investigate how the code quality improvement is related to external aspects such as students' prior coding experience, interest and their time spent on the assignments. We use the results to formulate a set of lessons learned in order to improve the design of the course and to inspire educators who consider introducing a similar type of course.

A.9 Anomaly Detection in Smart Grid Data: An Experience Report (C9)

B. Rossi, S. Chren, B. Buhnova, and T. Pitner. Anomaly detection in smart grid data: An experience report. In *2016 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, pages 2313–2318, 2016

Abstract. In recent years, we have been witnessing profound transformation of energy distribution systems fueled by Information and Communication Technologies (ICT), towards the so called Smart Grid. However, while the Smart Grid design strategies have been studied by academia, only anecdotal guidance is provided to the industry with respect to increasing the level of grid intelligence. In this paper, we report on a successful project in assisting the industry in this way, via conducting a large anomaly-detection study on the data of one of the power distribution companies in the Czech Republic. In the study, we move away from the concept of single events identified as anomaly to the concept of collective anomaly, that is itemsets of events that may be anomalous based on their patterns of appearance. This can assist the operators of the distribution system in the transformation of their grid to a smarter grid. By analyzing Smart Meters data streams, we used frequent itemset mining and categorical clustering with clustering silhouette thresholding to detect anomalous behaviour. As the main result, we provided to stakeholders both a visual representation of the candidate anomalies and the identification of the top-10 anomalies for a subset of Smart Meters

A.10 Big Data Platform for Smart Grids Power Consumption Anomaly Detection (C10)

P. Lipčák, M. Macak, and B. Rossi. Big data platform for smart grids power consumption anomaly detection. In *2019 Federated Conference on Computer Science and Information Systems (FedCSIS)*, pages 771–780, 2019

Abstract. Big data processing in the Smart Grid context has many large-scale applications that require real-time data analysis (e.g., intrusion and data injection attacks detection, electric device health monitoring). In this paper, we present a big data platform for anomaly detection of power consumption data. The platform is based on an ingestion layer with data densification options, Apache Flink as part of the speed layer and HDFS/KairosDB as data storage layers. We showcase the application of the platform to a scenario of power consumption anomaly detection, benchmarking different alternative frameworks used at the speed layer level (Flink, Storm, Spark).

A.11 Smart Grids Co-Simulations with Low-Cost Hardware (C11)

M. Schvarcbacher and B. Rossi. Smart grids co-simulations with low-cost hardware. In *2017 43rd Euromicro Conference on Software Engineering and Advanced Applications (SEAA)*, pages 252–255, 2017

Abstract. Smart Grids have nowadays gained wide diffusion and relevance. Due to the complexity of the grid, many Smart Grids laboratories have emerged over the years to provide partially virtualized environments for testing and co-simulation testbeds for the modern grid. However, the costs for setting-up Smart Grids laboratories are substantial, representing a barrier for newcomers and for educational purposes. In this paper, we propose an hardware-in-the-loop (HIL) architectural solution based on Arduino and Raspberry PI boards, supported by the Mosaik framework to simulate different Smart Grids scenarios on a small and cost-effective scale. We highlight the educational benefits that the solution can bring for understanding simulations and HIL in an affordable & effective way in an easy-to-deploy environment.

A.12 Scaling Agile in Large Organizations: Practices, Challenges, and Success Factors (J1)

M. Kalenda, P. Hyna, and B. Rossi. Scaling agile in large organizations: Practices, challenges, and success factors. *Journal of Software: Evolution and Process*, 30(10):e1954, 2018. e1954 smr.1954

Abstract. Context: Agile software development has nowadays reached wide adoption. However, moving agile to large-scale contexts is a complex task with many challenges involved. Objective: In this paper, we review practices, challenges, and success factors for scaling agile both from literature and within a large software company, identifying the most critical factors. Method: We conduct a focused literature review to map the importance of scaling practices, challenges, and success factors. The outcome of this focused literature review is used to guide action research within a software company with a view to scaling agile processes. Results: Company culture, prior agile and lean experience, management support, and value unification were found to be key success factors during the action research process. Resistance to change, an overly aggressive roll-out time frame, quality assurance concerns, and integration into preexisting nonagile business processes were found to be the critical challenges in the scaling process. Conclusion: The action research process allowed to cross-fertilize ideas from literature to the company's context. Scaling agile within an organization does not need to follow a specific scheme, rather the process can be tailored to the needs while keeping the core values and principles of agile methodologies.

A.13 Comparing Maintainability Index, SIG Method, and SQALE for Technical Debt Identification (J2)

P. Strečanský, S. Chren, and B. Rossi. Comparing maintainability index, sig method, and sqale for technical debt identification. *Scientific Programming*, 2020:14, Jul 2020

Abstract. There are many definitions of software Technical Debt (TD) that were proposed over time. While many techniques to measure TD emerged in recent times, there is still not a clear understanding about how different techniques compare when applied to software projects. Thee goal of this paper is to shed some light on this aspect, by comparing three techniques about TD identification that were proposed over time: (i) the Maintainability Index (MI), (ii) SIG TD models, and (iii) SQALE analysis. Considering 20 open source Python libraries, we compare the TD measurements time series in terms of trends and evolution according to different sets of releases (major, minor, and micro), to see if the perception of practitioners about TD evolution could be impacted. While all methods report generally growing trends of TD over time, there are different patterns. SQALE reports more periods of steady states compared to MI and SIG TD. MI is the method that reports more repayments of TD compared to the other methods. SIG TD and MI are the models that show more similarity in the way TD evolves, while SQALE and MI are less similar. The implications are that each method gives slightly a different perception about TD evolution.

A.14 Smart Grids Data Analysis: A Systematic Mapping Study (J3)

B. Rossi and S. Chren. Smart grids data analysis: A systematic mapping study. *IEEE Transactions on Industrial Informatics*, 16(6):3619–3639, 2020

Abstract. Data analytics and data science play a significant role in nowadays society. In the context of smart grids, the collection of vast amounts of data has seen the emergence of a plethora of data analysis approaches. In this article, we conduct a systematic mapping study aimed at getting insights about different facets of SG data analysis: application subdomains (e.g., power load control), aspects covered (e.g., forecasting), used techniques (e.g., clustering), tool support, research methods (e.g., experiments/simulations), and replicability/reproducibility of research. The final goal is to provide a view of the current status of research. Overall, we found that each subdomain has its peculiarities in terms of techniques, approaches, and research methodologies applied. Simulations and experiments play a crucial role in many areas. The replicability of studies is limited concerning the provided implemented algorithms, and to a lower extent due to the usage of private datasets.

A.15 Recommendations for Smart Grid Security Risk Management (J4)

V. Lamba, N. Šimková, and B. Rossi. Recommendations for smart grid security risk management. *Cyber-Physical Systems*, 5(2):92–118, 2019

Abstract. Smart grids (SGs) represent a paradigm shift for the traditional electric power infrastructure in terms of generation, transmission, and distribution of electricity in real time. The vast use of Information and Communication Technology (ICT) is a key enabler for the provision of smart energy services to customers. For such provision and sustainability of services, the SG infrastructure has a high level of complexity that brings an increased risk of security threats that need to be properly accounted and managed. The goal of this article is to provide recommendations for security risk management for SGs, discussing aspects of SG risk management, and the peculiarities for the planning, identification, assessment, prioritisation, monitoring, and control of security risks.

A.16 Smart Grids Co-Simulations: Survey & Research Directions (J5)

P. Mihal, M. Schvarcbacher, B. Rossi, and T. Pitner. Smart grids co-simulations: Survey & research directions. *Sustainable Computing: Informatics and Systems*, 35:100726, 2022

Abstract. The integration of renewable sources, communication and power networks with information and communication technologies is one of the main challenges in Smart Grids (SG) large-scale testing. For this reason, the coupling of simulators is commonly used to dynamically simulate several aspects of the SG infrastructure, in the so-called co-simulations. In this paper, we provide a scoping review of research of co-simulations in the context of Smart Grids: i) research areas and research problems addressed by co-simulations, ii) specific co-simulation aspects focus of research, iii) typical coupling of simulators in co-simulation studies. Based on the results, we discuss research directions of future SG co-simulation research in each of the identified are

A.17 Smart Grid Testing Management Platform (SGTMP) (J6)

M. Schvarcbacher, K. Hrabovská, B. Rossi, and T. Pitner. Smart grid testing management platform (sgtmp). *Applied Sciences*, 8(11), 2018

Abstract. The Smart Grid (SG) is nowadays an essential part of modern society, providing two-way energy flow and smart services between providers and customers. The main drawback is the SG complexity, with an SG composed of multiple layers, with devices and components that have to communicate, integrate, and cooperate as a unified system. Such complexity brings challenges for ensuring proper reliability, resilience, availability, integration, and security of the overall infrastructure. In this paper, we introduce a new smart grid testing management platform (herein called SGTMP) for executing real-time hardware-in-the-loop SG tests and experiments that can simplify the testing process in the context of interconnected SG devices. We discuss the context of usage, the system architecture, the interactive web-based interface, the provided API, and the integration with co-simulations frameworks to provide virtualized environments for testing. Furthermore, we present one main scenario about the stress-testing of SG devices that can showcase the applicability of the platform.