Thesis for The Degree of Doctor of Philosophy

Understanding, Measuring, and Evaluating Maintainability of Automotive Software

Jan Schroeder

UNIVERSITY OF GOTHENBURG

Division of Software Engineering
Department of Computer Science & Engineering
University of Gothenburg
Göteborg, Sweden, 2020

Understanding, Measuring, and Evaluating Maintainability of Automotive Software

Jan Schroeder

ISBN: 978-91-8009-009-4
Technical Report No. 187D
Division of Software Engineering
Department of Computer Science & Engineering
University of Gothenburg
Göteborg, Sweden

This thesis has been prepared using LaTeX.
Printed by Chalmers Reproservice,
Göteborg, Sweden 2020.

To Viktor

Abstract

Context: The importance of software maintainability is well-addressed by software engineering research, in general. Particularly for object-oriented and open-source software, measurements as a means to represent maintainability are well-established. Nevertheless, there is a lack of a similar understanding for software maintainability of executable models, which are widely used in the automotive industry, predominantly, using Simulink. Maintainability for automotive software is the main setting of this thesis. Software growth and complexity which are concepts related to maintainability are also investigated.
Objective: In this thesis, we aim to investigate maintainability for model-based software in the automotive domain. We explore the aspects it consists of, elicit maintainability measurements, and assess their applicability in practice. Additionally, we investigate two approaches to evaluate existing measurement data. First, we show how outliers with a significant impact on software quality can be identified in measurement data. Second, regarding software growth in the context of Simulink models, we show which predictions are relevant to practitioners, how these can be reliably conducted, and which environmental factors software growth is affected by. Lastly, in this thesis, we aim to present a practical implementation of software quality-focused design and evaluation of an automotive software architecture.
Method: As Simulink models are widely used in the automotive industry, we always work in close collaboration with practitioners from industry. Hence, the majority of the work presented in this thesis has been performed in the form of case studies within the automotive industry in Sweden and Germany. In addition, we always associate findings from the industry with current research using literature. We use multiple qualitative and quantitative research methods. This includes literature reviews, interviews and workshops with practitioners in industry, surveys, and software measurement with consecutive data analysis and hypothesis testing.
Results: In this thesis, we present a categorized list of aspects related to the maintainability of Simulink models, as well as a list of measures for the maintainability of Simulink models ranked by practitioners from industry. We provide evidence that simple size measures can be more applicable maintainability measures in practice than more complex measures. We present an approach to detect impactful outliers in measurement data.
Furthermore, concerning software growth, we list environmental factors affecting software growth measurement and prediction. We further provide a collection of practitioners' expectations towards growth predictions and rank prediction approaches for growth measurements by their applicability in industry. Lastly, we present an approach to the design and evaluate a software architecture in the automotive industry.
Conclusion: With these results, we provide a taxonomy of maintainability for Simulink models and respective measurements. Together with the methods for data analysis, we move a step towards a common understanding of maintainability for Simulink models which is presently missing. Next to that, we present approaches for maintainability measurement and analysis applicable in practical work environments. Thereby, we facilitate the application of rigorous measurements and analysis in the domain of automotive software.

Keywords

Software Engineering, Industrial Case Studies, Empirical Research, Software Quality, Maintainability, Software Measurement, Automotive Software, Simulink, Software Architecture

ISBN: 978-91-8009-008-7 (PRINT)
ISBN: 978-91-8009-009-4 (PDF)

Acknowledgments

My biggest thanks go to my supervisor Christian Berger, for his tireless encouragement, guidance, and positive energy. Without his support and feedback I might not have pursued my PhD studies and definitely would not have made it that far. I am grateful to have you as a friend and mentor.

I also want to thank my co-supervisor Miroslaw Staron, for his constructive and honest advice at any time I needed it. His constant endeavor to strengthen my work with scientific rigor was always highly appreciated.

A special thanks goes to Thomas Herpel, my co-supervisor and friend. Thank you for sharing your technical knowledge, experience, and ideas. I could always rely on your talent to show me new ways to my work where I could not see them.

I am very grateful to my colleagues at the Software Engineering Division. Particularly, Ivica Crnkovic, Michel Chaudron, Regina Hebig, Alessia Knauss, Hang Yin, Abdullah-Al Mamun, Antonio Martini, Vard Antinyan, Lukas Gren, Grischa Liebel, Hugo Sica de Andrade, Federico Giaimo, Darko Durisic, Hiva Alahyari, Salome Maro, and Emil Alégroth as they were always available for constructive discussions and valuable reviews.

I also want to thank my friends and colleagues from industry Christoph Funda, Okan Ecin, Ibrahim Alagöz, Herman Abt, Christian Baar, Martin Schalau, Radhakrishna Kothamasu, Wilhelm Bairlein, Balázs Gábor, and Robert Dämbkes for their willingness to share their technical knowledge, their patient availability to answer all my questions, and for always making me feel welcomed on site.

I owe a special thanks to Bára, for being an indefatigable source of inspiration and motivation for improvement, but also for her patience in stressful times and her constant support.

Finally, I want to send a big “Danke” to my family. First, to my parents Frank and Felicitas, who supported me with all means available and never stopped believing in me, even in difficult times. Second, to my sister Julia for her encouragement and decision support in times of doubt.

List of Publications

Appended Papers

This thesis is based on the following papers.

[A] J. Schroeder, C. Berger, V. Antinyan, F. Hajredini, S. A. Manesh
"Understanding and Measuring Maintainability of Simulink Models"
In submission to the International Journal on Software and Systems Modeling (SoSyM), 2020.

[B] J. Schroeder, C. Berger, T. Herpel, M. Staron
"Comparing the Applicability of Complexity Measurements for Simulink Models during Integration Testing – An Industrial Case Study"
Proceedings of the 2nd International Workshop on Software Architecture and Metrics (SAM), Florence, Italy, May 16, 2015.

[C] J. Schroeder, C. Berger, M. Staron, T. Herpel, A. Knauss
"Unveiling Anomalies and their Impact on Software Quality in Model-Based Automotive Software Revisions with Software Metrics and Domain Experts"
Proceedings of the 25th International Symposium on Software Testing and Analysis (ISSTA), Saarbrücken, Germany, July 18-22, 2016.

[D] J. Schroeder, C. Berger, A. Knauss, H. Preenja, M. Ali, M. Staron, T. Herpel
"Prediction of Software Model Growth in Practice"
Proceedings of the 33rd International Conference on Software Maintenance and Evolution (ICSME), Shanghai, China, September 17-24, 2017.

[E] J. Schroeder and C. Berger
"Environmental Factors for Measurement and Prediction of Software Growth in the Automotive Industry"
In submission to the Journal of Software: Evolution and Process, 2020.

[F] J. Schroeder, D. Holzner, C. Berger, C. J. Hoel, L. Laine, A. Magnusson
"Design and Evaluation of a Customizable Multi-Domain Reference Architecture on top of Product Lines of Self-Driving Heavy Vehicles"
Proceedings of the 37th International Conference on Software Engineering (ICSE), Florence, Italy, May 16-24, 2015.

Other Papers

The following papers are published during my PhD studies but not appended to this thesis.

Simulation & Validation in HIL environments

[a] J. Schroeder, C. Berger, T. Herpel
"Challenges from Integration Testing using Interconnected Hardware-in-the-Loop Test Rigs at an Automotive OEM"
In Proceedings of the First International Workshop on Automotive Software Architecture (WASA), Montréal, QC, Canada, May 4, 2015.

[b] J. Schroeder, C. Berger, T. Herpel
"A Methodology for Simulation and Validation of a Safety-Critical Electronic Control Unit for Integration Testing in Connected Hardware-in-the-Loop Environments"
Proceedings of the 3rd International Symposium on Future Active Safety Technology Towards Zero Traffic Accidents (FASTzero), Gothenburg, Sweden, September 9-11, 2015.

[c] T. Herpel, T. Hoiss, J. Schroeder
"Enhanced Simulation-Based Verification and Validation of Automotive Electronic Control Units"
Proceedings of the 5th International Conference on Electronics, Communications and Networks (CECNet), Shanghai, China, December 12-15, 2015.

[d] I. Alagöz, J. Schroeder, C. Funda, R. German, C. Berger
"Validating Test Cases for Safety Relevant ECUs using Simulation Models"
Proceedings of the 4th International Symposium on Future Active Safety Technology Towards Zero Traffic Accidents (FASTzero), Nara, Japan, September 18-21, 2017.

Testing Autonomous Vehicles

[a] A. Knauss, J. Schroeder, C. Berger, H. Eriksson
"Paving the Roadway for Safety of Automated Vehicles: An Empirical Study on Testing Challenges"
Proceedings of the Intelligent Vehicles Symposium (IV), Redondo Beach, California, USA, June 11-14, 2017.

[b] A. Knauss, C. Berger, H. Eriksson, J. Schroeder
"Proving Ground Support for Automation of Testing of Active Safety Systems and Automated Vehicles"
Proceedings of the 4th International Symposium on Future Active Safety Technology Towards Zero Traffic Accidents (FASTzero), Nara, Japan, September 18-21, 2017.

[c] I. R. Jenkins, L. O. Gee, A. Knauss, H. Yin, J. Schroeder
"Accident Scenario Generation with Recurrent Neural Networks"
Proceedings of the 21st IEEE International Conference on Intelligent Transportation Systems (ITSC), Maui, Hawaii, USA, November 4-7, 2018.

Others

[a] H. S. Andrade, J. Schroeder, I. Crnkovic
"Software Deployment on Heterogeneous Platforms: A Systematic Mapping Study"
IEEE Transactions on Software Engineering (Early Access), 2019.

[b] R. Hebig, T. H. Quang, R. Jolak, J. Schroeder, H. Linero, M. Ågren, A. M. Raj, S. Maro, K. Al-Sabbagh
"How do Students Experience and Judge Software Comprehension Techniques?"
In Proceedings of the 28th International Conference on Program Comprehension (ICPC), Seoul, South Korea, October 5-6, 2020.

Research Contribution

For all appended papers, except Paper F, I contributed with the study design, data collection, data analysis, and the majority of writing. The remaining co-authors contributed with discussions, reviews, survey and interview coding, and improvement suggestions. The ideas for the papers usually emerged during discussions with Christian Berger, Thomas Herpel, and Miroslaw Staron.

Study design, data collection, and data analysis for paper F are conducted in evenly balanced collaboration with the co-author Daniela Holzner, under the supervision of Christian Berger. The writing of the final publication was equally split up among the three main authors.

In Paper A and D, two co-authors conducted the initial literature reviews in form of thesis work supervised by me. In both cases, I re-evaluated and extended results, discussions, threats, and related work leading to substantial changes or extensions.

Contents

Chapter 1

Introduction

Automotive Software is growing in size and complexity as vehicles are required to become more and more intelligent. Modern premium cars can easily accommodate over 100 million lines of software code (cf. [1] and [2]). Next to the challenge of growing software size, automotive software is often complex. For example, Pretschner et al. [3]) mention high complexity due to hardware/software interaction and the distributed nature of automotive software. Even though there is an ongoing change of paradigms towards multi-purpose computers, which is highlighted, for example, by Zerfowski and Buttle [4], currently software in cars is distributed over multiple electronic control units (ECUs) specialized for specific components of the car. In his study, Vogelsang [5] has recently shown that such automotive architectures contain numerous feature dependencies, which may largely be unknown to the developers. Accordingly, architectural design for automotive software is required to embrace the interaction of possibly 100 computing units working and interacting simultaneously. Lastly, in addition to the challenges of growth and complexity, Pretschner et al. [3] found that there is a large number of tools used in automotive software development and Bock et al. [6] highlight that there is a constant change in development methods, processes, and tool chains. Altogether, this shows that developing software in the automotive domain poses various challenges.

Considering ongoing trends in the field, currently predominantly electrification and autonomous driving, we expect above challenges to exacerbate. For example, Vdovic et al. [7] list a set of challenges in regard to electrification. In particular, they mention the challenge of increasing architectural complexity and the need for better software quality. Regarding autonomous driving, Mallozzi et al. [8] found that the uncertainties when driving autonomously and the related need for constant adaptation pose a challenge towards future automotive software and its architecture. Furthermore, they mention the vast amount of data received from sensors and the environment which in turn motivates the wide use of machine learning approaches to be a challenge for automotive software development.

Software engineering research provides investigations and solutions addressing software quality, maintainability, and architectural design in general. However, on multiple occasions during our work in the automotive industry, we identified a lack of such coverage in the specific field of automotive software

and model-based software development. According to Altinger et al. [9], model-based software, using the Matlab/Simulink development environment, makes up for on third of the development activities in the field. Hence, there is a clear need for further investigations to which extent currently existing approaches are applicable in the context of model-based software development with Simulink in the automotive industry. Considering this scope, in this thesis, we look into the aspects of software quality and architecture design with a particular focus on maintainability and its associated concepts of software complexity and software growth.

Maintenance is a major part of a software project life-cycle. Low maintainability has been associated to a low motivation and productivity among developers, as well as high project costs (cf. [10] and [11]). Furthermore, the constant change with the field of automotive software development seems to be reflected in the development teams, too. We observed change and fluctuation in the teams which may lead to the constant need to instruct new team members. Maintainable software may reduce respective efforts. Lastly, we have seen that software in automotive is often continuously evolved from existing versions. A car, evolving to a new version, usually still contains the same major parts like, for example, the engine, brakes, airbags, etc. Hence, reuse is important. Maintainable software may facilitate this extensive need for reuse in the field. All these factors motivated us to conduct research in this area.

In this thesis, our main approach to evaluate software quality and maintainability is by using software measurements as it is an established approach to quantify the concept of software quality. For example, Fenton and Bieman [12] describe how the size and other structural attributes of software relate to its quality and how measuring those attributes provides us with a way of describing quality according to clear, predefined rules. Measuring provides deterministic and replicable results for our studies.

1.1 Background and Scope

As software quality and maintainability are main aspects of this thesis we follow clear and established definitions in all of the included studies. Throughout the thesis, we use the definitions from the ISO/IEC 25000 series standard for software quality characteristics. The standard defines maintainability as follows

> ***Maintainability***
> *"[The] degree of effectiveness and efficiency with which a product or system can be modified by the intended maintainers."*

These modifications include corrections, improvements or adaptation. Modifications may occur during initial software development and during later maintenance phases. We study both in this thesis and treat them as equally important. The ISO/IEC 25000 standard series was the most apparent choice for our definitions. It is well-established and used in research as well as in practice. Whenever we needed to determine which quality characteristic a practitioner or a paper refers to, we mapped their descriptions to the definitions from the standard.

Maintainability comprises seven sub-characteristics, modularity, reusability, analysability, changeability, modification stability, testability, and maintainability compliance. All sub-characteristics are considered when investigating maintainability. This applies, for example, when judging studies or replies from practitioners.

During this thesis we also discuss two other concepts related to maintainability, which are (a) software complexity and (b) software growth. The following will introduce both concepts. Complexity (a) has multiple definitions. For example, Antinyan et al. [13] interpret complexity as "[...] an emergent property of code that is magnified by the addition of more elements and/or interconnections, changing the existing elements and interconnections, or not clearly specifying the function of existing elements." In this definition, the connection to maintainability is explicit. In this thesis we understand complexity as one possible aspect affecting software maintainability.

In general, we are led by the concept that independent of its cause, complexity always is related to how the software can be understood by the developers. Furthermore, in some situations, complexity might be unavoidable, for example, if artifacts demand extensive logic. In a recent work on complexity with a focus on cyber-physical systems, Kopetz [14] discusses these concepts in detail. Kopetz claims that complexity can be categorized as "object complexity", focusing on the properties of a software artifact, and "cognitive complexity", focusing on how an observer comprehends the artifact. In this thesis, we do not make this distinction as we consider both of the concepts of complexity to be strongly related. Instead, we understand complexity as always related to an observer.

Furthermore, Kopetz elaborates on the terms of essential complexity and accidental/incidental complexity, following the terms created by Brooks and Kugler [15] and Dvorak [16]. Complexity is essential if it is unavoidable to create the functionality of a system. Complexity is accidental or incidental if created unintentionally or because of inadequate design decisions. Separating between these two categories is out of the scope of this thesis as we are mostly interested in the connection of complexity and maintainability. We are aiming to understand which aspects of a software artifact make it more difficult to maintain, independent of its accidental or essential nature.

This leads to two observations. First, a software artifact which is highly complex only becomes a problem if it has to be maintained. Second, complexity can become problematic if it is not controlled, for example, if it is unintended or unmanaged. The concept of the recently emerging term technical debt is relate to these unintended complexities which may accumulate over time. The connection of technical debt and maintainability is shown, for example by Li et al. [17]. Most of the studies that they investigated mentioned a negative effect of technical debt on the maintainability of a software system. Accordingly, we understand complexity, technical debt, and maintainability as strongly related and far from being fully understood. Particularly, we see a high need to investigate how complexity affects the maintainability of model-based software in the automotive industry.

In industrial practice, complexity is a widely used term. At times, this leads to biased interpretations, for example related to the well-known concept of cyclomatic complexity. To illustrate that, we observed that some engineers may

perceive large software artifacts as complex, while others concern interdependencies or the levels of nested blocks inside a model. In practice, it is critical to first investigate what exactly makes a software artifact complex before using existing metrics. Hence, we expect that practitioners will benefit from in-depth studies aimed to improve understanding of maintainability and complexity in the domain of automotive software development.

Software size (b) is the second concept used in this thesis, related to maintainability. Many studies indicate this relation, for example, Sjøberg et al. [18] and more recently Gil and Lalouche [19]. In practice, software exceeding size limits set by hardware or software requirements causes long loading, build, and deployment times. Hence, while software grows in size and complexity, situations arise where refactoring and software maintenance becomes necessary. Hence, size and growth measurement is an important part of this thesis.

For the majority of studies comprising this thesis, we focus on software used in the automotive industry as subject of study; examples from our industrial collaborations include Simulink models used during the development of embedded systems for multiple vehicle control functions. Whereas vehicle control functions in this thesis include all functionality used to operate functionality within the vehicle. For example, this includes steering, braking, and safety, but also climate management and entertainment functions. In general, research on model-based software is broad and established. Liebel et al. [20] outlined current trends regarding model-based software engineering and found that various languages are used. UML and SysML are common but there is also a broad spectrum of domain-specific languages. The amount of available tools is vast and Liebel et al. found Matlab with Simulink and State Flow, Eclipse-based tools, and Enterprise Architect. The wide set of languages and tools indicates that depending on the domain or context, models may be understood differently. In this thesis, we use the following classification created by Zheng and Taylor [21]: the goal of any modeling in this thesis is to create compilable and executable models so they can replace source code as main development artifact. Throughout the thesis, we call these models executable models. Simulink models are one example of such executable models which are heavily used in the automotive industry. Most of the studies in this thesis focus particularly on Simulink models. As Simulink models are largely used to implement control software in vehicles, our interpretation of automotive software is limited to control software and generally excludes higher-level components like, for example, infotainment systems. Simulink models usually rely a set of tools and artifacts like code generators, version control, connected libraries, etc. In this thesis, we are trying to isolate the models and investigate them independent of the surrounding tool chain.

In Paper F, we raise the level of abstraction and investigate software siting on top of the vehicle control layer. Specific details can be found in Chapter7. Figure1.1 summarizes the scope for a better overview. It shows that we focus on Simulink in Papers A to E and on higher-level software in Paper F. It also shows that automotive software is not limited to these fields but hosts a variety of different tools, languages and application areas.

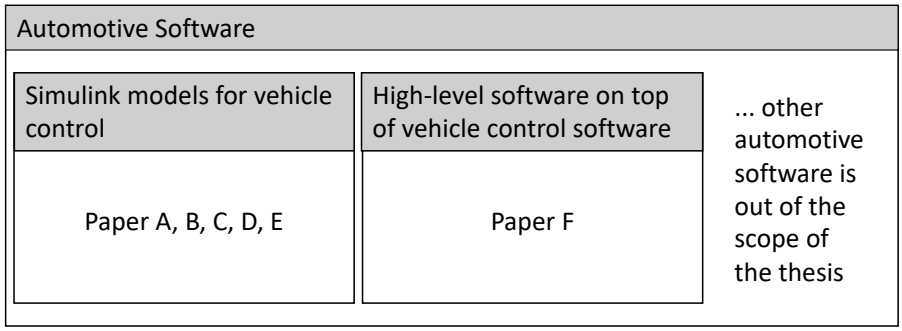

Figure 1.1: The scope of the thesis, including Simulink models and higher level software used in the automotive industry.

1.2 Study Goals

The overall goal of this thesis can be summarized to investigating maintainability and related concepts for model-based software in the automotive domain. We combined the specific study goals from each paper to the following three goals which we address in this thesis.

G1 Understanding and measuring maintainability of Simulink models in the automotive industry

G2 Interpreting and evaluating measurement results received in the automotive industry

G3 Designing and Evaluating a reference architecture in the automotive industry

In Table 1.1, we assigned the papers in this thesis to a goal and highlight the particular focus of the study. For example, the focus of Paper A is on aspects of and metrics for maintainabilty which in turn addresses Goal 1. The table visualizes how the studies are connected.

Papers A to E all have either maintainability or a related concept in focus. In Paper F software quality, including maintainability, is part of the design and evaluation of the architecture. Software measurement is a major aspect of our studies, too. In all our studies, we either use software measures, compare their performances, or evaluate their results. Software measurement is particularly prominent in Papers A to C where we compare and evaluate metrics. Papers C to E build on top of results from software measurements. In Paper F we use measurements to evaluate a reference architecture.

The following research questions are derived from the goals and summarized from the individual papers. All research questions are limited to automotive software development.

RQ1 How is maintainability of Simulink models interpreted in practice?

- Answered in Papers A and B

Table 1.1: Goals and focus for each paper presented in this thesis

Goal	Study	Focus	Details
G1	Paper A	Maintainability	Metrics
			Aspects
G1	Paper B	Complexity	Metrics
			Aspects
G2	Paper C	Quality	Metrics
			Outlier analysis
G2	Paper D	Growth	Prediction approaches
G2	Paper E	Growth	Environmental aspects
G3	Paper F	Architecture	Requirements
			Design
			Evaluation

RQ2 What are applicable measurements for maintainability, complexity, and growth of Simulink models?

- Answered in Papers A, B, and C

RQ3 What are methods to interpret and evaluate measurement results in practice?

- Answered in Papers C and D

RQ4 What are environmental aspects affecting software growth measurement results?

- Answered in Paper E

RQ5 How to design and evaluate a reference architecture?

- Answered in Paper F

1.3 Related Work

In this section we focus on related work respective to the main goal of the thesis of investigating maintainability and related concepts. Related work on other more specific aspects of the thesis are presented in the respective papers. For related work, the scope is not reduced to automotive software and Simulink. We looked into approaches from other domains as well as looking for work on general model-based software or even architecture design.

As mentioned in Section 1.1, there are multiple standards addressing software maintainability and it is a well-defined concept. Next to the standards, we were not able to find extensive research covering the interpretation of maintainability in the automotive industry. Practitioners may interpret the

quality characteristic according to their understanding and possibly focus on specific aspects of it relevant in their projects. We identified a gap in research investigating aspects of maintainability in the automotive industry.

Complexity on the other hand is not as well-defined. Instead, there are multiple attempts to formalize the concept using theoretical frameworks (cf. [22] and [23]) and using empirical approaches as in Antinyan [13]. The general conclusion being that more research is necessary, particularly regarding the actual application of the presented approaches. We deem both, complexity and maintainability in the automotive industry to be a worthwhile research area.

The concept of software quality is usually quantified using software measurement. There is extensive research available investigating metrics for software quality characteristics and complexity. Next to metrics for common object-oriented software, Plaska et al. [24], Boström et al. [25], and Card and Agresti [26], for example, all look into metrics for models and architectures. In this thesis, much of the initial work builds on these three studies.

A dissertation by Scheible [27] addresses similar to this thesis automotive software development. The focus on metrics and the broad spectrum of software quality. While they list a big collection of possible metrics and thresholds, the evaluation is limited to a single case study. Similarly, Vogel et al. [28] have recently published a literature review on metrics for automotive software development. They look at the broad spectrum of all quality characteristics but find that maintainability is the one which is most often addressed. They found 19 studies investigating model-based software containing 29 metrics somehow related to measuring models. We could not identify an overview of which single metrics are model metrics. Hence, even the most recent work shows us that maintainability is a relevant topic and that there is still a gap for investigating model-based software in the automotive industry.

1.4 Research Methodology

The thesis work was conducted in close collaboration with industry. All studies investigate research questions in practice. In our research and data-collection methods, we always relate to practitioners by investigating existing documents, conducting interviews, workshops, or surveys. These methods are used either as an initial step to collect data to be confirmed or evaluated consecutively or as a confirming step after collecting data from other sources. This work with practitioners is always paired with literature studies or measurement and data analysis methods. Thereby, in each study, we achieve method/data triangulation. Table 1.2 contains an overview of all data collection methods used in the included studies.

In most studies, we start with a comprehensive investigation of existing literature. Four of our studies involve some measurements of software artifacts at a case company and consecutive data analysis and thereby have a very applied focus. This includes Papers B, C, D, and F which can be best described as case studies. Paper A and E also address research questions emerged from practice. They have a strong focus on the underlying concepts.

In their paper, Stol and Fitzgerald [29] categorize studies according to four scales. They aim to provide a holistic view over existing research methods in

Table 1.2: Data collection methods used in this thesis

	Methods	Data sources
Paper A	Literature study	8 final papers
	Survey	45 participants
Paper B	Software measurement	65 artifacts, 3 measures
	Stakeholder interviews	8 participants
Paper C	Software measurement	71 artifacts, 4 measures
	Stakeholder interviews	6 participants
Paper D	Literature study	3 existing literature reviews
	Stakeholder interviews	6 participants
	Software measurements	48 artifacts, 1 measure
Paper E	Literature study	5 final papers
	Survey	22 participants
Paper F	Literature study	16 final papers
Part 1	Stakeholder interviews	13 participants
	Existing document analysis	4 documents
Paper F	Literature study	5 final papers
Part 2	Stakeholder workshops	2 participants

software engineering. To provide a better understanding of the set of studies attached with this thesis, we categorize them using the scales proposed by them. The four scales are obtrusiveness, research focus, generalizability, and setting. They are explained in the following list

- **Obtrusiveness**
  This scale describes the extent of control researchers have on the study setting. For example, in controlled experiments, researchers have full control over variables and confounding factors while in case studies in the field researchers usually merely observe.

- **Focus**
  This scale has three categories, 'actor', 'context', and 'behavior'. This in an indicator as to which aspect of a study is represented as most realistic or most generalizable. For example, in survey studies, the actors are in the focus and as realistically represented as possible. In field studies, on the other hand, the context is in focus. Lastly, experimental studies are a typical example for a strong focus on a behavior.

- **Generalizability**
  This scale describes how universally results are applicable. The scale goes from 'universal' to 'particular'.

- **Research setting**

Table 1.3: Categorization of the studies included in the thesis

Source	Stol and Fitzgerald [29]				Runeson and Höst [30]
Scale	obtrusiveness	context	setting	focus	purpose
Paper A	unobtrusive	universal	natural	actors	descriptive
Paper B	unobtrusive	particular	natural	actors	exploratory
Paper C	unobtrusive	particular	natural	context	exploratory
Paper D	unobtrusive	particular	natural	context	descriptive
Paper E	unobtrusive	universal	natural	actors	descriptive
Paper F	unobtrusive	particular	natural	context	exploratory

> This scale contains for four categories describing how realistic the environment is in which the study is conducted. It reaches from 'natural' if the study is conducted in the field, to 'contrived', if the study environment is more artificial, like for example in a research lab, and 'no empirical setting' which corresponds to studies may also be 'setting independent' which may, for example, be achieved using a survey with a large sample size.

In Table 1.2, we decided to select the categories that fit our studies best. There are naturally always nuances to this particularly considering that we combine at least two research methods in each study. In addition to the scales provided by Stol and Fitzgerald, we list each study's purpose, according to the four types presented by Runeson and Höst [30]. They are 'exploratory', 'descriptive', 'explanatory', and 'improving'.

We always aimed to be as unobtrusive as possible in our studies. As we always investigate real world problems and actual practitioners in the field, we wanted to ensure to investigate the context as realistic as possible and avoid bias induced by us as researchers. This concept is, for example, highlighted in the work of Zelkowitz et al. [31]. Our studies are always in the automotive domain and we always concern observations in practice. Accordingly, our setting for each study is categorized as 'natural'. Accordingly, we portrait the current status of the study setting as realistically as possible while we usually do not provide hard evidence for causal relationships. We also do not exercise control over our study setting.

While all studies are very similar regarding the above mentioned two scales, they are diverse regarding the focus and context scales. Regarding the study focus, the levels of obtrusiveness and setting practically exclude a focus on behavior due to the lack of control. The other two categories, namely 'context' and 'actors' are addressed to an equal extent. While studies A, B, and E are particularly focused on how practitioners in the field act, behave, and think, studies C, D and F rather focus on the phenomena and processes observable in the field. In studies A and E we aim for a high generalizability while the other studies have a very particular context.

The above categorizations are probably inherited by the purpose we seek to fulfill with our studies. The study purposes can be categorized as exploratory and descriptive (cf. [30]). In our studies, we usually aim to portrait the current situation in the field. Even though, we attempt to highlight associations we usually do not intent to reveal causal relationships as this would require to enforce a higher level of control.

1.5 Study Summaries

In this section we shortly introduce the studies attached with this thesis. We describe what was the motivation to conduct each study, which methods were used and what are the main outcomes.

1.5.1 Chapter 2: Understanding and Measuring Maintainability of Simulink Models (Paper A)

During our work with different Simulink modeling departments in industry and investigation of related literature, we identified a lack of comprehensive overviews for Simulink model maintainability. Accordingly, in this study we aimed to establish a taxonomy of the concept and to provide a list of metrics which can assess it. We used a literature review to collect existing metrics. We extracted 23 individual metrics, provided definitions, and discussed related maintainability sub-characteristics. In the subsequent survey, we selected the 17 most-relevant metrics for a ranking. 44 practitioners from industry assessed which of the metrics they believe addresses maintainability of Simulink models best. Hence, we present a ranked list of maintainability metrics which, according to our sample work best in practice. Additionally, the practitioners provided a list of positive and negative aspects which they believe affect maintainability. We used the 256 collected aspects to create a taxonomy of maintainability for Simulink models.

1.5.2 Chapter 3: Comparing the Applicability of Complexity Measurements for Simulink Models During Integration Testing – an Industrial Case Study (Paper B)

Similarly to the first study, this work addresses the need of understanding complexity in Simulink models. We aimed to compare three existing metrics to assess complexity in practice. We selected two size measurements and a complexity measurement based on existing work. Those measures were applied to 78 Simulink models currently used in practice. In consecutive interviews with the team developing the models, we were able to collect ratings of perceived complexity for each model. These ratings were independent of the measurement results and therefore unbiased. They could be compared directly with the measurement results. We showed that practitioners favor simple size metrics like block count and lines of code over more sophisticated metrics assessing the model structure and signal routing. This is particularly interesting as it was counter-intuitive to the practitioners. They favored the more complex

metrics, when asked to rank the metrics based on their underlying concepts in the beginning of the interviews. Hence, we showed not only which metric works better in practice but also that practitioners may be biased when rating metrics based on their description.

1.5.3 Chapter 4: Unveiling Anomalies and Their Impact on Software Quality in Model-Based Automotive Software Revisions with Software Metrics and Domain Experts (Paper C)

In the third study, after having a reasonable understanding of applicable metrics, we decided to investigate data analysis approaches. We investigated measurement results taken over time using version histories of the models. The goal was to develop a methodology to automatically highlight software versions where changes with high impact on software quality are made. We applied four metrics to 65,000 software revisions of 71 Simulink models. We then applied two different approaches from the fields of time series analysis and statistics to highlight outliers within the measurements. The identified outliers were then shown to the developers in individual interviews and follow-up workshops. They rated the impact of the changes made in the outlying software version regarding six software quality characteristics. Hence, we could verify if detected outliers actually reveal changes with high impact on software quality and which metrics reveal the most relevant outliers. We show that detected outliers are associated to the assessment of the practitioners. Hence, the automated outlier detection approach is applicable in practice. It is able to reveal software changes with high impact on software quality. We further showed that each metric reveals outliers with impact on different software quality characteristics. This approach can be used to analyze historic changes but also to survey current development.

1.5.4 Chapter 5: Predicting and Evaluating Software Model Growth in the Automotive Industry (Paper D)

In Chapter 5, we continued the work on data analysis by investigating how measurement results can be predicted in practice. Next to the best prediction approach, we aimed to investigate what practitioners actually expect from a prediction of measurement data. In this study we decided to focus on size measures as they exhibit certain practical advantages and also showed to be well-qualified to assess maintainability of Simulink models. We started with a literature review to collect existing approaches to predict time series of measurement data. At the same time, in interviews and workshops we collected the expectations towards the predictions from Simulink developers in practice. By extending the measurements conducted in Paper B, we collected 4,547 size measurements using the version history of 48 Simulink models. We found five different approaches currently used to predict measurement data grouped into linear, statistical approaches and non-linear machine learning approaches including neural networks and support vector machines. Further, we found that practitioners require long-term prediction, up to one month into the future

using approaches which require low maintenance effort. Speed is not a strong requirement. The prediction itself may run over night. When comparing the prediction results, we show that both, statistical and machine learning approaches are applicable to our data. Only support vector machines predicted our data significantly worse than the other approaches. The results for the statistical approaches were slightly better. They also require less maintainance effort as they do not have to be trained. Accordingly, in this study we show how to predict future development of measurement results in practice.

1.5.5 Chapter 6: Environmental Factors for Measurement and Prediction of Software Growth in the Automotive Industry (Paper E)

When predicting and analyzing software change and growth in the previous work, we found that functionality is not the only aspect influencing it. We saw that there is a continuous impact of external, environmental events affecting change within software which are not possible to predict when focusing only on functional or artifact-related factors. Research addressing non-artifact-related factors is scarce. We identified a need for a rigorous investigation of this problem.

Accordingly, in Chapter 6, we aimed to investigate environmental factors influencing software growth. Using a survey within automotive industry, we elicit factors from practitioners. A literature review complements the practitioners' opinion with research findings. As result we created a synthesized list which includes, for example, factors like change in requirements, hardware, team size and the development environment.

1.5.6 Chapter 7: Design and Evaluation of a Customizable Multi-Domain Reference Architecture on top of Product Lines of Self-Driving Heavy Vehicles An Industrial Case Study (Paper F)

In Chapter7, we develop and evaluate a reference architecture in an industrial context. The architecture is meant to capture all possible transport mission and route planning a heavy vehicle or a fleet thereof may perform. It focuses on high level software and does not include vehicle control software. While following the functional needs, the main goal was to design strictly following the major software quality requirements of software maintainability and portability. The evaluation, which was the second main goal, evaluated the same quality characteristics but specifically the sub-characteristics adaptability, changeability, and stability. All design and evaluation steps are based on rigorous analysis of current literature, stakeholder involvement, and existing document analysis. Hence, in this study we present a systematic process to design and evaluate reference architectures in practice. The process supports the design of the functional architecture while ensuring software quality.

1.6 Contributions

In this thesis, we contribute to the software engineering body of knowledge by answering five research questions. Regarding research question 1 on how maintainability is interpreted in the field, we found aspects explaining the concept in Papers A and B. Paper A resulted in 256 maintainability aspects, grouped into eleven categories and 77 sub-categories. We also ranked the aspects by the number of mentions. As the results are received from a broad survey, we are convinced that we provide an extensive classification of maintainability for the whole field of automotive software development with Simulink models. Paper B provides a ranked list of maintainability aspects from one industrial case company. Together, these results provide a comprehensive taxonomy and address the lack of rigorous literature assessing maintainability for Simulink models. Thereby, we contribute towards the understanding of software maintainability of Simulink models used in the automotive industry.

To answer research question 2, in Paper A, we collected 16 metrics to assess maintainability of executable models which we found applicable in the field. Therefrom, we extracted 23 re-occurring base measures for a better understanding of the building blocks used to design maintainability measurements and to increase generalizability as those base measures can be compared with measurements from other fields and the maintainability aspects collected for research question 1. Furthermore, we provide a ranking of the measures collected from 44 survey participants and are able to answer which metric is perceived to work best considering automotive Simulink models in practice. As highlighted in Section 1.3, rigorous and comprehensive overviews of measures for maintainability are missing. Hence, we understand that our identified measures are relevant to both, academia and industry; in addition, industry contributes from the directly applicable metrics presented in this thesis as they are broken down to their basic building blocks and are defined in detail. In Paper B, we compare the performance of three maintainability measures for Simulink, in practice. We present how well the metrics address the practitioners' view of maintainability. By answering the first two research questions, we are convinced to present a realistic state-of-the-art view on maintainability in the field.

Research question 3 asks for the evaluation and interpretation of measurement results. In Paper C, we present a novel approach to uncover anomalies among measurement data with an impact on software quality. Hence, using measurements and outlier detection, we can show when changes to a software artifact have a high impact on software quality characteristics. In the same study, we verify the results using a single case study in the automotive industry. Our focus in Paper D are predictions of measurement data. We present the most relevant prediction approaches for model growth from literature, including traditional statistical approaches and more recent machine learning approaches using neural networks and suport vector machines. We further collect a set of expectations towards predictions in practice from practitioners. This enables us to finally present a ranking of the prediction approaches by accuracy and other criteria important to the field. Hence, the results reported in this thesis contribute to industry by providing approaches that can be directly applied to evaluate measurement results. In addition, these approaches have also been

evaluated in practice. We could not find any study reporting a similar anomaly detection approach applied to measurement data. Hence, the approach also constitutes a novel contribution to academia.

Paper E addresses research question 4 by listing and explaining environmental factors affecting predictions and measurement data, in general. Thereby, we aim to improve future predictions and measurements by incorporating possible confounding factors. In Paper F, answering the last research question, we design and evaluate a complete reference architecture in the field. In the paper, we present a systematic process to designing reference architectures for automotive heavy vehicles. This includes applicable approaches to elicit and document requirements, make and document architectural design decisions, handle variability, and evaluate architectural quality.

1.7 Discussion

With the answers to the five research questions, we contribute towards the study goal of understanding maintainability of Simulink models in the field of automotive software engineering. Still, the combined results raise a set of questions to be discussed in the scope of this thesis.

1.7.1 What is the best metric to assess maintainability of Simulink models used in the automotive domain?

In this thesis, we present a set of metrics which may be used to measure maintainability of automotive software. The combined results of Papers A and B provide a ranked list of possibly applicable metrics. Some metrics may perform better than others but there is no single best metric to recommend in general. In practice, a set of highly metrics can be combined in a function. For example, one could combine the number of signal line crossings in a Simulink model in relation to the total number of signals. Both are highly ranked by practitioners in Paper A.

1.7.2 Are maintainability metrics from other domains applicable to Simulink?

We found that metrics from other domains partly follow similar concepts as metrics used for Simulink models. Regarding maintainability metrics in general, Jabangwe et al. [32] report that complexity, cohesion, coupling, and size measures correlate with maintainability of object oriented software. Similarly, when investigating object-oriented Java projects, Kozlov et al. [33] found that the number of code lines, data types, inputs, outputs, and cyclomatic complexity correlates with maintainability. For source code in general, Riaz et al. [34] found that size, complexity, and coupling are the most commonly used metrics. Next to the concept of size, the cohesion, and the coupling of components are recurring.

We found that these concepts can also be found in Simulink metrics. The concept of size, for example, appears in our best ranked metrics “Number of Signals/Transitions”, ‘Number of n-Ports”, and “Number of Out-Ports”.

While the concepts are similar, it is important to note that the realization of these concepts into measurements may differ for Simulink. For example, the measure "number of code lines generated" from the model, an equivalent to the common size measurement lines of code, received a low ranking. Other metrics which were highly ranked in the context of Simulink are not typical in research on traditional software. These are measures specific to Simulink. This is, for example, the number of blocks in a model without proper configuration. Accordingly, we conclude that metrics from other domains are usually not directly applicable. They may need to be adapted to fit the Simulink domain.

1.7.3 How useful are prediction approaches for Simulink measurement results in practice?

Predicting measurement results is used in this thesis to predict the development of software quality of Simulink models. We found, that practitioners do not necessarily look for the most precise prediction approach. Especially not for only one next point in time but are rather interested in how a model will develop through the coming months. This has to be taken into account when choosing prediction approaches.

Further we observed that prediction approaches are susceptible to the number of inputs provided. This is particularly problematic considering the measurements presented in this thesis, which mostly assess structural attributes of software, like size or coupling. The problem arises if such attributes change only occasionally. This leads to respectively few data points and may affect the prediction performance. We found that particularly in these cases, linear prediction approaches seem to outperform machine learning approaches.

1.7.4 Threats to Validity

In Table 1.4, we collected the major threats from the studies collected in this thesis. As previously shown in Section 1.4, most of our studies are particular in context and have a natural setting. Specifically, Papers B, C, D and F can clearly be classified as case studies and consequently share a common threat to generalizability. Being aware of this, we always tried to mitigate using data and method triangulation but naturally, fully generalizable results cannot be achieved with this type of study. Hence, our results in these studies is limited to Simulink models or executable models within the automotive industry.

Paper F is a case study, too, but only the design of the actual architecture is specific to the case company. Other steps in the presented process, like the requirements and evaluation include a wide range of external information sources. Accordingly, we expect the presented concepts to be applicable in a broader context, too.

Partly, we are working with small samples which has an effect on the validity of our data analysis, as in Paper B and C. When working with small samples, we make our conclusions carefully, not to overestimate results. We also mitigate the threat to generalizability by making the sample as diverse as possible. For example, by interviewing practitioners taking on different roles in the company and with different experience.

Paper A	
Convenience sampling in the survey	Broad sample, snowballing of practitioners
Manual, extensive data extraction	At least two researchers per task
Survey participants biased from existing metrics and approaches	Survey design and formulation to avoid bias
Paper B	
Interviews with only one interviewer	Consecutive confirmatory workshops
Data analysis on few data points	Triangulation of results with qualitative data
Generalizability (single case)	Data and method triangulation
Paper C	
Generalizability (single case)	Interview experienced engineers; clearly outline data collection and analysis for replications
Human aspects in interviews	Introductions before interviews; open interviews with possibility to ask questions; confirmation with consecutive workshops
Small sample	Data and method triangulation
Paper D	
Generalizability (single case)	Clearly outline data collection and analysis for replications
Variability in machine learning methods	Provide step by step descriptions and used libraries
Small interview sample	Practitioners with different roles and experience
Paper E	
Convenience sampling in the survey	Broad sample, snowballing of practitioners
Bias start set in snowballing literature review	Analysis of start set topics and citation hierarchy
Paper E	
Generalizability (single case)	Extensive set of methods and samples (internal and external to the case company)

Table 1.4: Summary of the major threats to the validity of this thesis and their mitigation. Threats are on the left side and mitigations on the right.

Our two broader studies A and E, which are less prone to generalizability issues, rely on a survey in which convenience sampling was used to collect the participants. This could not be avoided to reach the intended population of practitioners. To reduce bias in the sample, we contacted a broad set of practitioners, diverse regarding experience, job title, and educational background.

To summarize, the results in our thesis are mostly confirmed withing the scope of Simulink models within the automotive industry. Still, we have no reason to believe that the metrics and other findings may not be applicable to other modeling approaches, too. During the survey in paper A, we received answers from developers using other modeling languages like Modelica, Dymola, and UML. The number of respondents is not large enough for a deeper analysis but their statements indicate that similar concepts are important for other modeling languages, too.

1.8 Conclusions

In this thesis we investigated maintainability of automotive software. We extracted three goals, reaching from understanding and measuring maintainability, to interpreting and evaluating measurement results, to designing and evaluating a complete reference architecture. Five research questions were answered to address these goals. In the process, we collected an extensive set of measures and aspects describing maintainability in the field. Ranking of these measures and aspects allows us to recommend the metrics that shall be favored over others and highlight the aspects that describe maintainability best. We propose methods to predict the growth of software artifacts and to detect impactful outliers among measurement results. We provide a set of environmental factors affecting measurements allowing for the creation of more accurate measures. Lastly, we are able to present an applicable approach to designing and a reference architecture in the field. Even though, future work may provide an even deeper and more rigorous understanding of the applicability of maintainability measurements, we believe that results presented in this thesis present a considerable step to improve understanding, measuring, and evaluating maintainability of automotive software.

As whole, with this thesis, next to explaining maintainability in the field, we provide the metrics which are most applicable to measure it. Using the received measurement data, we show how to further evaluate and interpret results using outlier detection and predictions. We also show how prediction and measurement results can be further improved by determining environmental influences. Combining these results enables practitioners to establish a reliable assessment and more rigorous investigation of maintainability for Simulink models.

When interpreting these results we are able to address some of the challenges still present in the automotive industry. As confirmed, for example, by Dajsuren and van den Brand [1], Altinger [2], and Pretschner et al. [3], still currently, growth in size and complexity is challenging the domain and is aggravated by the trends towards electrification and automation. Establishing a rigorous understanding of what makes automotive software maintainable may help to

address the challenge of growth and complexity, while applicable measurements help to reliably assess where in the software these problems are most prominent. Together with the findings on interpreting and improving measurement results we have established further applicable and reliable approaches in the field to eventually cope with growing size and complexities.

Close collaboration with industry in all our studies increases the applicability of our results, which are at parts currently used in industry practice. Hence, we are convinced that the whole domain of Simulink software development in the automotive industry may benefit from this work. The use of method and data triangulation throughout the thesis is our strongest means to produce reliable results which contributes an academical advancement of the domain.

Bibliography

[1] Y. Dajsuren and M. van den Brand, "Automotive software engineering: Past, present, and future," in *Automotive Systems and Software Engineering*, Y. Dajsuren and M. van den Brand, Eds. Springer International Publishing, 2019, ch. 1, pp. 3–8.

[2] H. Altinger, *State-of-the-Art Tools and Methods Used in the Automotive Industry*. Cham: Springer International Publishing, 2019, pp. 59–73. [Online]. Available: https://doi.org/10.1007/978-3-030-12157-0_4

[3] A. Pretschner, M. Broy, I. H. Kruger, and T. Stauner, "Software engineering for automotive systems: A roadmap," in *Future of Software Engineering (FOSE '07)*, 2007, pp. 55–71.

[4] D. Zerfowski and D. Buttle, "Paradigmenwechsel im automotive-softwaremarkt," *ATZ - Automobiltechnische Zeitschrift*, vol. 121, no. 9, pp. 28–35, 2019.

[5] A. Vogelsang, "Feature dependencies in automotive software systems: Extent, awareness, and refactoring," *Journal of Systems and Software*, vol. 160, 2020.

[6] F. Bock, C. Sippl, S. Siegl, and R. German, "Status report on automotive software development," in *Automotive Systems and Software Engineering*, Y. Dajsuren and M. van den Brand, Eds. Springer International Publishing, 2019, ch. 3, pp. 29–57.

[7] H. Vdovic, J. Babic, and V. Podobnik, "Automotive software in connected and autonomous electric vehicles: A review," *IEEE Access*, vol. 7, pp. 166 365–166 379, 2019.

[8] P. Mallozzi, P. Pelliccione, A. Knauss, C. Berger, and N. Mohammadiha, *Autonomous Vehicles: State of the Art, Future Trends, and Challenges*. Cham: Springer International Publishing, 2019, pp. 347–367. [Online]. Available: https://doi.org/10.1007/978-3-030-12157-0_16

[9] H. Altinger, F. Wotawa, and M. Schurius, "Testing methods used in the automotive industry: Results from a survey," in *Proceedings of the 2014 Workshop on Joining AcadeMiA and Industry Contributions to Test Automation and Model-Based Testing*, ser. JAMAICA 2014. New York, NY, USA: Association for Computing Machinery, 2014, p. 1?6. [Online]. Available: https://doi.org/10.1145/2631890.2631891

[10] J. T. Nosek and P. Palvia, "Software maintenance management: changes in the last decade," *Journal of Software: Evolution and Process*, vol. 2, no. 3, pp. 157–174, 1990.

[11] B. P. Lientz and E. B. Swanson, "Software maintenance management: a study of the maintenance of computer applications software in 487 data processing organizations," 1980.

[12] N. Fenton and J. Bieman, "Software metrics: A rigorous and practical approach, 3rd edition," Feb 02 2015, copyright - Copyright Ringgold Inc Feb 2, 2015; Last updated - 2015-02-06.

[13] V. Antinyan, M. Staron, and A. Sandberg, "Evaluating code complexity triggers, use of complexity measures and the influence of code complexity on maintenance time," *Empirical Software Engineering*, vol. 22, no. 6, pp. 3057–3087, 2017.

[14] H. Kopetz, *Simplicity is Complex*. Springer, 2019.

[15] F. Brooks and H. Kugler, *No silver bullet*. April, 1987.

[16] D. Dvorak, "Nasa study on flight software complexity," in *AIAA Infotech@ Aerospace Conference and AIAA Unmanned... Unlimited Conference*, 2009, p. 1882.

[17] Z. Li, P. Avgeriou, and P. Liang, "A systematic mapping study on technical debt and its management," *Journal of Systems and Software*, vol. 101, pp. 193–220, 2015.

[18] D. I. K. Sjøberg, B. Anda, and A. Mockus, "Questioning software maintenance metrics: A comparative case study," in *Proceedings of the 2012 ACM-IEEE International Symposium on Empirical Software Engineering and Measurement*, Sep. 2012, pp. 107–110.

[19] Y. Gil and G. Lalouche, "On the correlation between size and metric validity," *Empirical Software Engineering*, 2017.

[20] G. Liebel, N. Marko, M. Tichy, A. Leitner, and J. Hansson, "Model-based engineering in the embedded systems domain: an industrial survey on the state-of-practice," *Software & Systems Modeling*, vol. 17, no. 1, pp. 91–113, Feb 2018.

[21] Y. Zheng and R. N. Taylor, "A classification and rationalization of model-based software development," *Software & Systems Modeling*, vol. 12, no. 4, pp. 669–678, Oct 2013.

[22] E. Weyuker, "Evaluating software complexity measures," *IEEE Trans. Softw. Eng.*, vol. 14, no. 9, pp. 1357–1365, Sep 1988.

[23] L. C. Briand, S. Morasca, and V. R. Basili, "Property-based software engineering measurement," *IEEE Trans. Softw. Eng.*, vol. 22, no. 1, pp. 68–86, 1996.

[24] M. Plaska, "Simulink-specific design quality metrics," Turku Centre for Computer Science, Tech. Rep. TUCS Tech. Rep. 1002, February 2011.

[25] P. Boström, R. Grönblom, T. Huotari, and J. Wiik, "An approach to contract-based verification of simulink models," Turku Centre for Computer Science, Tech. Rep. TUCS Technical Report 985, 2010.

[26] D. Card and W. Agresti, "Measuring software design complexity," *Journal of Systems and Software*, vol. 8, no. 3, pp. 185 – 197, 1988.

[27] J. Scheible, *Automatisierte Qualittsbewertung am Beispiel von MATLAB Simulink-Modellen in der Automobil-Domäne.* Tübingen, Germany: Universität Tübingen, 2012.

[28] M. Vogel, P. Knapik, M. Cohrs, B. Szyperrek, W. Püschel, H. Etzel, D. Fiebig, A. Rausch, and M. Kuhrmann, "Metrics in automotive software development: A systematic literature," *Journal of Software: Evolution and Process*, 2020, preprint.

[29] K. Stol and B. Fitzgerald, "A holistic overview of software engineering research strategies," in *IEEE/ACM 3rd International Workshop on Conducting Empirical Studies in Industry.* IEEE, 2015, pp. 47–54.

[30] P. Runeson and M. Höst, "Guidelines for conducting and reporting case study research in software engineering," *Empirical Software Engineering*, vol. 14, no. 2, pp. 131–164, 2009.

[31] M. V. Zelkowitz and D. R. Wallace, "Experimental models for validating technology," *Computer*, vol. 31, no. 5, pp. 23–31, May 1998. [Online]. Available: http://dx.doi.org/10.1109/2.675630

[32] R. Jabangwe, J. Börstler, D. Šmite, and C. Wohlin, "Empirical evidence on the link between object-oriented measures and external quality attributes: a systematic literature review," *Empirical Software Engineering*, vol. 20, no. 3, pp. 640–693, Jun 2015.

[33] D. Kozlov, J. Koskinen, M. Sakkinen, and J. Markkula, "Assessing maintainability change over multiple software releases," *Journal of Software Maintenance and Evolution: Research and Practice*, vol. 20, no. 1, pp. 31–58, 2008.

[34] M. Riaz, E. Mendes, and E. Tempero, "A systematic review of software maintainability prediction and metrics," in *Proceedings of the 2009 3rd International Symposium on Empirical Software Engineering and Measurement*, ser. ESEM '09. Washington, DC, USA: IEEE Computer Society, 2009, pp. 367–377.