Mälardalen University Licentiate Thesis 295

Managing Variability in SysML Models of Automotive Systems

Damir Bilic

MÄLARDALEN UNIVERSITY
SWEDEN

Mälardalen University Press Licentiate Theses
No. 295

MANAGING VARIABILITY IN SYSML MODELS OF AUTOMOTIVE SYSTEMS

Damir Bilic

2020

MÄLARDALEN UNIVERSITY
SWEDEN

School of Innovation, Design and Engineering

ISBN 978-91-7485-479-4
ISSN 1651-9256
Printed by E-Print AB, Stockholm, Sweden

U sjećanju na tatu

Abstract

Organizations developing software-intensive systems inevitably face increasing complexity of developed products, mainly due to rapid advancements in all domains of technology. Many such organizations are considering model-based systems engineering (MBSE) practices to cope with the increasing complexity. The use of models, as a central role during product design, promises to provide benefits such as enhanced communication among system stakeholders, continuous verification, improved design integrity, traceability between requirements and system artifacts and many more. Additionally, products are often built in many variants. That is especially obvious in the automotive domain, where customers have the ability to configure vehicles with hundreds of configuration options. To deal with the variability, a product line engineering approach is often used. It allows the development of a family of similar software-intensive systems that share a common base while being adapted to individual customer requirements.

In this thesis, the overall goal is to evaluate and facilitate the combination of the two mentioned approaches, model-based systems engineering and product line engineering, in an industrial environment. To achieve the main thesis goal, it was divided into three separate research goals. The first goal was to identify challenges when applying an annotation-based approach for variant management in SysML models on a use case provided by Volvo Construction Equipment. The aim was to identify and understand challenges when using existing tool support to manage variants in implementation artifacts of existing products. The second research goal was to identify reuse-related challenges in the "clone-and-own" based development process of Volvo CE. Moreover, we assess the effects of model-based product line engineering on the identified challenges. Lastly, the third research goal was to develop an approach for consistency checking between variability- and system models. To achieve that, we develop an integrated toolchain for model-based product line engineering that allows the integration of variable artifacts, which are not documented in system models, into the development process. Secondly, we define and develop an approach for consistency checking between variability models that describe the system in terms of features and implementation models that describe how variability is implemented in the product itself, since such support does not exist in current state-of-the-art tools.

Based on the results from the results of case studies at Volvo CE, it was shown that model-based product line engineering has the potential to improve communication and highlight implications of variability to stakeholders (e.g. to non-technical staff), improve traceability between variability in requirements and variability in design and implementation, improve consistency through constraints between variants and automate repetitive activities. In other words, it shows potential for improving product quality while reducing the development lead time. However, the evaluation and measurement of improvement will be left for future work because measuring the product quality and lead time requires an organizational rollout of model-based product line engineering.

Sammanfattning

På grund av dagens snabba tekniska föränding, möter organisationer som utvecklar programvaruintensiva system ofta växande komplexitet i sin produktutveckling. Ett sätt att hantera denna växande komplexitet är genom att använda modellbaserad systemutveckling.

Användningen av modeller som en central del vid utveckling av programvaruintensiva system kan ge fördelar som förbättrad kommunikation, kontinuerlig säkerställning av kvalitet, förbättrad design, och spårbarhet mellan krav och systemkomponenter. Dessutom är produkter ofta byggda i olika varianter. Detta är särskilt tydligt inom bilindustrin, där kunderna har möjlighet att konfigurera fordon med hundratals olika konfigurationsalternativ. För att kunna hantera variationen används ofta produktlinjer för utveckling av dessa produkter. Produktlinjer möjliggör utveckling av en produktfamilj av liknande programvaruintensiva system som kan konfigureras utifrån individuella kundbehov.

Avhandlingens övergripande mål är att utvärdera och förbättra hur modellbaserad systemutveckling och produktlinjer kan kombineras i en industriell kontext. För att utreda den överliggande problemställningen har tre forskningsmål tagits fram. Det första målet var att identifiera utmaningar när produktlinjer används för hantering av variabilitet i SysML-modeller av produkter från Volvo Construction Equipment. Syftet var att identifiera och förstå utmaningar när man använder befintliga verktyg för att hantera varianter i modeller av befintliga produkter. Det andra forskningsmålet var att identifiera utmaningar i användningen av “clone-and-own" baserade utvecklingsprocessen på Volvo CE. Dessutom utvärderar vi effekterna av modellbaserade produktlinjer på de utmaningar som tidigare har identifierats. Det tredje forskningsmålet var att utveckla ett tillvägagångssätt för konsistenskontroll mellan variabilitets- och systemmodeller. För att uppnå detta utvecklade vi ett verktyg för modellbaserade produktlinjer som möjliggör integrering i en utvecklingsprocess för variabla produktkomponenter som inte är dokumenterade i systemmodeller. Vi definierar och utvecklar ett tillvägagångssätt för konsistenskontroll mellan variabilitetsmodeller och implementeringsmodeller. Variabilitetsmodeller beskriver systemets variabilitet, men implementeringsmodeller beskriver hur variabilitet implementeras i produkten.

Avhandligen visar att användingen av modellbaserade produktlinjer kan förbättra kommunikationen, belysa konsekvenserna av variabilitet för ingenjörer men även personal utan teknisk bakgrund. Det kan även förbättra spårbarhet mellan variabilitet i systemkrav, design och implementering, samt förbättra konsistensen med begränsningar mellan variabla produktkomponenter och automatisera repetitiva aktiviteter.

Modellbaserade produktlinjer möjliggör förbättringar av produktkvaliteten samtidigt som utvecklingstiden minskas. Utvärdering och mätning av förbättringar lämnas för framtida forskning, då företagsorganisationer först behöver införa modellbaserad produktlinjeteknik i sin verksamhet.

Acknowledgments

To my parents, for all the immense sacrifices that they made throughout the years. Without their colossal efforts, I wouldn’t be where I am now. Dad, although you aren’t with us any more, I will always carry you in my heart. Thank you for teaching me to always keep asking questions, to never take anything for granted and to never fear challenges. Thank you for teaching me to dream big and to always stay optimistic. To my sisters, for all the support and encouragement through thick and thin.

I would like to express my sincere gratitude to my supervisors for giving me the opportunity to enroll at Mälardalen University as a PhD student. Thank you Daniel for having answers to my questions which I didn’t even know how to ask. Thank you Wasif for making sure that I always pay attention to even the smallest details. Thank you Peter for the always on point sanity checks whenever I needed them. Thank you Adnan for taking care of the complicated things so I didn’t have to.

As a large part of my research work has been done in collaboration with Volvo Construction Equipment. I would like to thank all the amazing people at the Engine Controls department, especially Marino for always keeping me safe in the "real world" and providing me with any resources I needed for my research. Many thanks to Christoffer and Dani for always having time to resolve any uncertainties I had during my time at Volvo CE. Special thanks to Patrik, Martin and Jagadish for their input to my research and countless hours spent in various discussions, often spanning far longer than the scheduled meetings.

Many thanks to the whole MDH gang, I enjoyed every activity we did together. It was always interesting to hear about the stories, (and try out the sweets), from all around the world during our fika times. It is an honour to have you as friends!

Eldar, I am very grateful to be able to call you a friend. Semir, Sejo and Hajra, thank you for treating me as a part of your family since the first day we met. I appreciate everything you did for me.

Selma, thank you for sticking by my side.

List of Publications

Papers included in thesis[1]

Paper A: D. Bilic, D. Sundmark, W. Afzal, P. Wallin, A. Causevic and C. Amlinger. Model-Based Product Line Engineering in an Industrial Automotive Context: An Exploratory Case Study In *Proceedings of the 22nd International Systems and Software Product Line Conference*. Volume 2, ACM, 2018.

Paper B: D. Bilic, E. Brosse, A. Sadovykh, D. Truscan, H. Bruneliere, U. Ryssel. An Integrated Model-based Tool Chain for Managing Variability in Complex System Design In *2019 ACM/IEEE 22nd International Conference on Model Driven Engineering Languages and Systems Companion (MODELS-C)*. IEEE, 2019

Paper C: D. Bilic, W. Afzal, D. Sundmark, P. Wallin, A. Causevic, C. Amlinger and D. Barkah. Towards a Model-Driven Product Line Engineering Process: An Industrial Case Study In *Proceedings of the 13th Innovations in Software Engineering Conference (ISEC2020)*. ACM, 2020.

Paper D: D. Bilic, J. Carlson, D. Sundmark, W. Afzal, P. Wallin. Detecting Inconsistencies in Annotated Product Line Models In *24th International Systems and Software Product Line Conference (SPLC2020)*

[1]The included papers have been reformatted to comply with the thesis layout.

Contents

Part I

Thesis

Chapter 1

Introduction

Successful and useful products are always built around the needs and requirements of customers. However, not all customers are identical. Some might need subtle differences in the product, while others might need highly specialized solutions. Besides customer requirements, legislations are often different in various countries, regions and continents. One of the domains where such concerns become apparent is the automotive domain. Often, customers request a product that is configured specifically based on their needs. Then, the manufacturer needs to produce such a customer-specific and configured vehicle. At the same time, the legislation region in which a vehicle is sold defines the legal standards, such as emissions, that need to be fulfilled. Developing a standardized product that is configurable for all customers can be a challenging task, especially in software-intensive systems, as software increases the complexity and variability of such systems [72].

Complex systems such as in the automotive domain are rarely developed from scratch. Rather, they are adaptations of one or several previous projects and products. Expectations from reuse practices are often similar, regardless of the domain where the product is developed. Reduction of development and product costs is very often the main driver for reuse with the assumption that the product quality will be transferred or even increased through reuse [10, 35]. Improved maintenance as well as reduced lead time and training costs are other reuse benefits, which are reported in literature as well [10, 79]. Many organizations are reaping the reuse benefits and that has been extensively researched and documented [57]. Such success stories commonly conclude that reuse practices have to be system-

Figure 1.1: Volvo Construction Equipment Machinery

atically performed in order to achieve the expected benefits.

One such organization that is developing highly configurable and constantly evolving systems is the Engine Controls department at Volvo Construction Equipment (Volvo CE). Volvo CE, a subsidiary to the Volvo Group, develops and manufactures construction equipment and heavy machinery including articulated haulers, wheel loaders, excavators, pavers and forestry machinery, illustrated in Figure 1.1. Within Volvo CE, the Engine Controls department delivers a range of diesel engines for such heavy machines.

The engine department of Volvo CE joined the MegaM@Rt2 [1] research project as a use case provider whose focus was on Model-based Systems Engineering (MBSE, [65]). MBSE proposes the use of models as a means to represent the complete system throughout various phases of its development cycle. System engineering activities, which are traditionally performed and documented in textual documents, are to be performed by utilizing models. Models in MBSE are often graphical representations of different aspects of the system that are to be developed [33], such as requirements, the system architecture, detailed design and even test specifications. A common language for graphical modelling in MBSE is SysML [38].

The main goal of Volvo CE, within MegaM@Rt2, was to exploit MBSE as an attempt to cope with the increasing complexity of engine systems and improve upon their reuse practices. An initial case study performed at Volvo CE has shown that MBSE has the potential to address several issues during systems engineering: (i) ambiguity of natural language requirements, (ii) communication within the organization, (iii) traceability between requirements, design, implementation and testing [75].

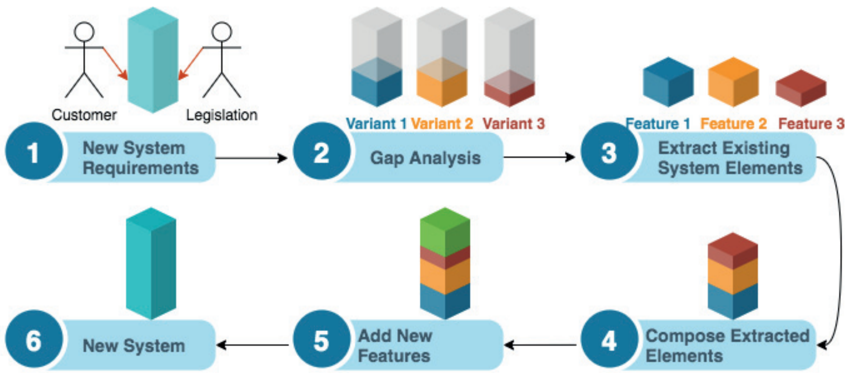

Figure 1.2: Illustration of the clone-and-own process

1.1 Research context

A research objective within MegaM@Rt2 was to explore variability modeling in model-based development approaches. This objective was also closely aligned to the objectives of Volvo CE. To get a better understanding of the use case, we have analyzed the systems engineering process at the engine department. It was shown that during the systems engineering process, engineers rely heavily on reusing already existing artifacts (such as requirements, component design, tests, etc.) when building new engine products. This suggested that reuse and variability could be one of the major factors in MBSE for the engine department at Volvo CE. A major factor in the engine system development was the large number of variants, where all engine variants were built on top of a small number of highly configurable and extensible engine platforms.

The current reuse process can be more or less described as a clone-and-own approach. Clone-and-own is a practice when currently existing systems, or parts of them, are copied and adapted in order to create a new variant of the system. The new product that is created using this approach usually has its own lifecycle and is independent from the system (or systems) from which it was cloned. As illustrated in Figure 1.2, the process starts with customer requirements for a product. In addition to customers, for engine systems, the legislation requirements must be considered as well (e.g. exhaust emission regulations). Once requirements are established, engineers proceed to step 2, the gap analysis. The existing system variants are analysed in order to understand which parts can be reused for the new project. The "clone" part of clone-and-own starts with step 3. In this step, the system elements to be reused are extracted and copied based on the results of the gap analysis. Once copied, the lifecycle of the cloned elements (or features as illustrated in Figure 1.2) is separated from the lifecycle of systems of origin. Further, in step 4, the extracted elements are composed together and adjusted if necessary, to form a base on which the requirements for new features will be implemented. In step 5, new features are added to the cloned base. In step 6, the "and-own" part of the clone-and-own practices takes effect. The lifecycle of both, cloned elements and new features, is now owned by the new product. Since the new product has its own lifecycle, the consequences are for example that bug fixes in a cloned system component will not necessarily be performed in all system variants which use the same component.

In reality, these steps are not always performed in the presented order. The cloning is often done ad-hoc without a proper gap analysis as it is usually a time consuming task. Without a detailed gap analysis, some aspects that could have been cloned can be overlooked, leading to situations where an already existing feature will need to be implemented again. Furthermore, since cloned features might be changed in step 4 of Figure 1.2, it is not always obvious from which system to clone a feature as there can be multiple versions of it in various system variants. Another issue is the interaction of system features. A part of the system can behave differently depending on the presence or absence of other features in the system. At Volvo CE, the engine system is based on a platform, therefore the clone-and-own approach is not used for the complete system. Rather, engine components, which are used to extend the platform and create variants, are built by following the clone-and-own process. In other words, there exists a base that is the same for all engines (the engine platform) on which variable features are added to create individual engine variants. These variable elements, which are added to the base engine, are mainly developed by following the clone-and-own process.

1.2 Problem Formulation

Due to the fact that there exists a common base for all systems and variants are built by extending the base with variable components, product line engineering (PLE) came as a natural choice for the management of commonality and variability in this environment. In PLE, a set of similar products is built from a number of core components, which are included in each product variant. Variable features are then added to the core in order to build application specific variants [63]. In the context of the described engine systems, the core for the complete product family is the base engine platform. Here, auxiliary components[1] are equivalent to features in PLE. Since the goal is to explore variability management in MBSE, our research will focus on the combination of MBSE and PLE, also known as model-based product line engineering (MB-PLE) [86].

The described industrial use case was shown to be a fitting context for the analysis of some open challenges in PLE as described in [56]. For example, Volvo CE is looking into MBSE, where managing reuse in all phases of the development process is an important aspect. At the same time, an open challenge in PLE is the interrelation between variability models, system requirements, the design, testing, etc. Furthermore, it is crucial to manage the consistency of variability across the different development artifacts.

Since SysML is the most commonly used modeling language for MBSE [22], it was necessary to explore modeling tools and approaches that support variability modeling with SysML. When looking at the state-of-art and state-of-practice, the majority of tools and approaches, which support variability modeling in SysML, implement the annotation based approach for modeling product lines [85, 43, 30]. This approach itself will be described in more detail later in the background section, but for now it is important to note that the majority of the studies suggests the benefits of such an approach as: (i) enabling mass-customization: each product can be customized to individual customers, (ii) reduced cost by building products from reusable parts, (iii) improved quality, (iv) reduced time-to-market, etc. However, very few discuss potential impediments when creating models that include variability [30, 56]. It is not clear whether such approaches, for example, increase the complexity of models or how they scale when a large number of features is added to a single system model.

Having the open challenges in mind and the opportunity to apply and evaluate model-based product line engineering on the described industrial use case, in this thesis we have: (i) evaluated currently existing tool support in order identify to understand challenges of annotation-based product line models, (ii) assessed the requirements and consequences of migrating from the current clone-and-own

[1]Auxiliary component is the name for all physical components that are added to the base engine in order to make a product variant, for example: Sensors, engine turbochargers, etc. In software, they are reflected through configuration and calibration parameters.

development process to a model-based PLE process, (iii) identified limitations in the state-of-art tools to support model-based PLE and (iv) developed a consistency checking approach for product line models based on SysML.

1.3 Thesis Overview

This thesis is based on a collection of research papers and is divided into two parts. The first part of the thesis introduces the research topic, the research process and the relations between the individual research papers. It starts with the motivation and general introduction to the topic, which was described earlier in the current chapter. Chapter 2 provides the theoretical background to the main concepts that are the main research foci in the thesis: PLE and MBSE. Inline with the background, Chapter 2 gives an overview of the related work on the relevant aspects of MBSE and PLE.

Chapter 3 and Chapter 4 describe the research goals and contributions of this thesis. Generally, the thesis aims to evaluate and facilitate the application of model-based product line engineering (MB-PLE) in an industrial environment. Firstly, we identify challenges when an annotation-based approach on SysML models is used to to manage variants in existing industrial systems. Secondly, we systematically identify reuse-related challenges in the clone-and-own based development process of Volvo CE. Then we assess the effect of model-based product line engineering on the identified challenges. We propose a PLE process that is aligned with the MBSE practices which are being under pilot evaluations at the Engine Controls department. We discuss the implications of MB-PLE on the development activities as well as system artifacts.

Despite the fact that the modeling tool at Volvo CE supports variability modeling, it was noted that it limits the applicability of the PLE process. Although we did show that most of the development process phases can be documented in the modeling environment, the organization expressed a need for flexibility, i.e. to be able to include other variable system artifacts in the PLE process such as software calibration files. The third objective was to create a MB-PLE toolchain that allows for such flexibility. Lastly, we define an approach for consistency checking between the variability model, which describes the variability of the system and the system model, which describes how the system is implemented. SysML was used as it was the preferred language for the ongoing MBSE activities within Volvo CE.

The first part of the thesis concludes with Chapter 5, a short discussion and conclusion with a reflection on possible directions for future work. The second part of the thesis contains the included papers whose abstracts are listed below:

Paper A

Title:

Model-Based Product Line Engineering in an Industrial Automotive Context: An Exploratory Case Study [19]

Authors:

Damir Bilic, Daniel Sundmark, Wasif Afzal, Adnan Causevic, Peter Wallin, Christoffer Amlinger

Venue:

The 1st International Workshop on Variability and Evolution of Software-intensive Systems (VariVolution'18), Colocated with The 22nd International Systems and Software Product Line Conference (SPLC'18)

Abstract:

Product Line Engineering is an approach to reuse assets of complex systems by taking advantage of commonalities between product families. Reuse within complex systems usually means reuse of artifacts from different engineering domains such as mechanical, electronics and software engineering. Model-based systems engineering is becoming a standard for systems engineering and collaboration within different domains. This paper presents an exploratory case study on initial efforts of adopting Product Line Engineering practices within the model-based systems engineering process at Volvo Construction Equipment (Volvo CE), Sweden. We have used SysML to create overloaded models of the engine systems at Volvo CE. The variability within the engine systems was captured by using the Orthogonal Variability Modeling language. The case study has shown us that overloaded SysML models tend to become complex even on small-scale systems, which in turn makes scalability of the approach a major challenge. For successful reuse and to, possibly, tackle scalability, it is necessary to have a database of reusable assets from which product variants can be derived.

Paper B

Title:

An Integrated Model-based Tool Chain for Managing Variability in Complex System Design [18]

Authors:

Damir Bilic, Etienne Brosse, Andrey Sadovykh, Dragos Druscan, Hugo Bruneliere,
Uwe Ryssel

Venue:

The 13th International Workshop on Models and Evolution (ME'19), co-located with The 22nd IEEE/ACM International Conference on Model Driven Engineering Languages and Systems (MODELS'19)

Abstract:

Software-intensive systems in the automotive domain are often built in different variants, notably in order to support different market segments and legislation regions. Model-based concepts are frequently applied to manage complexity in such variable systems. However, the considered approaches are often focused on single-product development. In order to support variable products in a model-based systems engineering environment, we describe a tool-supported approach that allows us to annotate SysML models with variability data. Such variability information is exchanged between the system modeling tool and variability management tools through the Variability Exchange Language. The contribution of the paper includes the introduction of the model-based product line engineering tool chain and its application on a practical case study at Volvo Construction Equipment. Initial results suggest an improved efficiency in developing such a variable system.

Paper C

Title:

Towards a Model-Driven Product Line Engineering Process – An Industrial Case Study [20]

Authors:

Damir Bilic, Daniel Sundmark, Wasif Afzal, Adnan Causevic, Peter Wallin, Christoffer Amlinger, Dani Barkah

Venue:

The 13th International Conference on Innovations in Software Engineering (ISEC'20)

Abstract:

Many organizations developing software-intensive systems face challenges with high product complexity and large numbers of variants. Partly, due to large numbers of both legal and customer-specific requirements. In order to effectively maintain and develop these product variants, Product-Line Engineering (PLE) methods are often considered, while Model-based Systems Engineering (MBSE) practices are commonly utilized to tackle product complexity. In this paper, we report on an industrial case study concerning the ongoing adoption of PLE in the MBSE environment at Volvo Construction Equipment (Volvo CE) in Sweden. In the study, we identify and define a PLE process that is aligned with MBSE activities at the Engine Controls department of Volvo CE. Furthermore, we discuss the implications of the migration from the current development process to a model-driven PLE-oriented process. This process and its implications are derived by conducting and analyzing interviews with Volvo CE employees, inspecting artifacts and documents, and by means of participant observation. Based on the results of a first system model iteration, we were able to document how MBSE and variability modeling will affect development activities, work products and stakeholders of the work products.

Paper D

Title:

Detecting Inconsistencies in Annotated Product Line Models

Authors:

Damir Bilic, Daniel Sundmark, Wasif Afzal, Adnan Causevic, Peter Wallin

Venue:

The 24th International Systems and Software Product Line Conference (SPLC'20)

Abstract:

Model-based product line engineering applies the reuse practices from product line engineering with graphical modeling for the specification of software intensive systems. Variability is usually described in separate variability models, while the implementation of the variable systems is specified in system models that use modeling languages such as SysML. Most of the SysML mod-

eling tools with variability support implement the annotation-based modeling approach. Annotated product line models tend to be error-prone since the modeler implicitly describes every possible variant in a single-system model. To identifying variability-related inconsistencies, in this paper, we firstly define restrictions on the use of SysML for annotative modeling in order to avoid situations where resulting instances of the annotated model may contain ambiguous model constructs. Secondly, inter-feature constraints are extracted from the annotated model, based on relations between elements that are annotated with features. By analyzing the constraints, we can identify if the combined variability and system model can result in incorrect or ambiguous instances. The evaluation of our prototype implementation shows the potential of our approach by identifying inconsistencies in the product line model of our industrial partner which went undetected through several iterations of the model.

Chapter 2

Background and related work

In this chapter, we introduce the concepts of model-based systems engineering, product line engineering and a combination of the two. The related work is discussed inline with the theoretical background.

2.1 Model-based Systems Engineering (MBSE)

Systems engineering is a multidisciplinary set of methods and processes that facilitates the transformation of stakeholder requirements into a system whose objectives are to accomplish the said requirements [46]. The technical activities in the system engineering process include requirement elicitation and analysis, specification and design of the system, as well as verification and validation, which ensures that the objectives of the system are met and that stakeholder requirements are satisfied [42].

MBSE is an approach that facilitates systems engineering. It consists of a set of concepts which propose the use of models as a means to represent information about the system that is being developed instead of the traditional, document-based approach. Previously, models and related techniques have been used extensively as a part of the document-based approach. These might include control flow diagrams, behavioral diagrams, schematics etc. However, these are mostly used for representation of specific aspects of a system. In the document-based approach, individual models are usually self-contained and not integrated with models of other parts of the system to allow for an overall system view [38].

MBSE presents a shift from the document-based representation to a model-based representation of the complete system [65] and includes the application of modeling activities as a central role to support the specification of requirements, design, integration, validation and operation of a system. It emphasizes the use of models in order to carry out systems engineering activities that are traditionally performed using a document-based approach [33]. The suggested benefits of MBSE compared to the traditional approach are: enhanced communication, possibility of continuous verification and validation of the design, enhanced design integrity, improved traceability between design artifacts, reuse of models for design evolution, automated document generation, etc. [81].

In order to support MBSE, SysML has been introduced. It is a general-purpose graphical modeling language that facilitates the representation of specification and architecture of systems [25]. It can represent the following aspects of systems: structural composition, system behavior, constraints on physical and performance properties, allocation between the different aspects, requirements and relationship between other requirements as well as their relationships to other design elements [38]. Systems can include hardware, software, data, personnel, specific procedures etc. In order to represent these system elements, a set of nine diagram types is available.

A comprehensive overview of SysML can be found in [38]. Through initial case studies at Volvo CE, MBSE helps to address issues such as: (i) ambiguity of natural language requirements, (ii) communication within the organization, (iii) traceability between requirements, design, implementation and testing [75]. These benefits of MBSE are not only related to the use case of Volvo CE, but are also documented in the literature [51, 23].

Industrial embedded systems are often developed by reusing already developed artifacts from previous products, and this is also true for Volvo CE. At the Engine Controls department, new engine systems are developed by evolving and improving on the functionality of previous systems. Although the new engine systems are always an evolution of a previous system, the old systems still must be maintained. The reason is the very long life cycle of engine systems that are used in Volvo CE machines. Reuse of system artifacts, such as design specifications and test procedures, is a fairly common practice in the industry, however reuse strategies in systems engineering are not so well defined [82].

Moreover, SysML does not support mechanisms for modeling of variability. Nevertheless, it is possible to express variability by creating SysML profiles to extend the language with variability concepts, as it can be seen in [83]. However this approach struggles to represent variability in behavioral aspects.

In order to address the needs of organizations that are using MBSE and developing variant-rich products, such as the previously mentioned Volvo CE example, we apply and evaluate PLE as an approach to improve reuse and manage variability in the MBSE environment.

2.2 Product Line Engineering (PLE)

PLE is a development process that allows for mass production of similar customized systems while enabling individualism for each single product variant. A group of similar products is developed and built from a reusable set of artifacts. These reusable artifacts can be common and variable. Common artifacts are building blocks of the product line which are included in every product variant, while variable artifacts are combined together with common ones in order to create the individual, customer-specific variants.

The research by Kang et al., [44], is what kick-started the topic of software product line engineering. It was then that the term feature model was actually introduced and the notion of a feature was defined. The notion of feature-oriented domain analysis (FODA) has been one of the main topics in the PLE research community.

An example where such development methods are necessary is the automotive domain. Vehicles are being mass-produced, yet each final product is customized based on the needs of the customer that ordered the vehicle. This mass-customization is enabled by allowing the variable artifacts to be combined in different ways based on the customer requirements.

The main benefits and expectations of PLE are, [6]: (i) mass-customization: instead of a small set of preconfigured products, products are configured for individual customers, (ii) reduced cost: instead of developing each product from scratch for each customer, systems are built by combining reusable parts in different ways based on customer needs, (iii) improved quality: frequently used and standardized product artifacts are tested in many products, potentially leading to more reliable products, (iv) reduced time-to-market: compared to developing each individual product completely from scratch, building a system based on a set of already developed and reusable assets can result in significant time savings.

In [6], the theoretical principles behind software PLE are explained. The main two driving principles are the concepts of problem domain and solution domain. For illustration purposes, we show a simplified version of the PLE de-

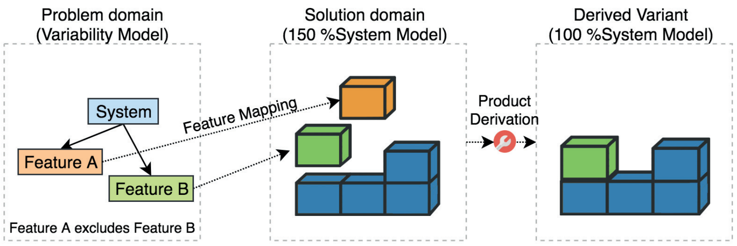

Figure 2.1: A generic approach for combining variability modeling with system modeling

velopment approach in Figure, 2.1 [18]. The left side in the figure depicts the problem domain. It defines the “what" aspect of the system, i.e., it describes the system in terms of features that include always-present (common) and variable features, the relations and the constraints between them.

The term "feature" for PLE was initially defied by Kang et. al. in [44] as "a prominent or distinctive user-visible aspect, quality, or characteristic of a software system or systems". Throughout the thesis, when referring to a feature, we refer to the same definition except that the feature does not necessarily have to be user-visible. Some features might be there, for example, due to legislation requirements, which require changes in aspects of the systems that are not always user-visible.

The information about features in the system (problem domain) is usually captured in separate models [29]. Feature models [44] and decision models [26] are a common way of representing variability in the problem domain. Several feature-based modeling methods can be identified in literature and comprehensive studies on feature models can be found in [14, 72, 29]. Feature models describe the common and variable aspects or features of a product line, with relationships between them.

After the system common and variable characteristics are described as features with relations between them in the variability model, the variability must also be implemented in the solution domain. The middle part of Figure 2.1 depicts the solution domain. This part describes the “how". This is where the system model is created and it describes how the system is implemented together with all the system features.

When considering graphical models, three common approaches to represent variability in the solution domain are: annotative approaches [28], compositional approaches [77, 7] and delta-based approaches [71]. These approaches are described in more detail later in the text.

It is important to point out that in this thesis, the focus is mainly on the annotative approach [28], i.e. all system variants are modeled within a single overspecified model. In the solution domain, the goal is to implement both variable and common features such that they can be combined together in order to create individual product variants. Ideally, features should be built as standardized parts of the system that need not to be changed when creating an individual variant of the product line.

Apel et al. [6] explain possible approaches for implementing and analyzing the problem domain and solution domain. Moreover, they define a PLE-based development process which illustrates how the problem and solution domain should be integrated together in order to be able to develop products in such an environment.

Variability models in the problem domain serve as a starting point for customers during the definition of a product variant. Customers select variable product features based on their needs. Constraints and relations between features restrict how features can be combined. As illustrated in Figure 2.1, features from the problem domain are mapped to implementation artifacts, which realize the individual features in the solution domain.

Based on the customer selection of features and mapping between the problem domain and solution domain, the product derivation is performed. During product derivation, the individual parts that represent the selected features are combined together to form a complete product variant. When new features are requested by customers or when currently existing features need to be adapted, the work is to be done within the solution domain. To prevent architecture erosion, in PLE, one must be careful when implementing new features or modifying currently existing features, especially during application engineering [87].

As illustrated in the right part of Figure 2.1, the derived system should include only features from the solution domain that were selected during feature selection. As the work in this thesis deals with PLE with SysML models, the derived variant includes for example: requirement models, the system architecture, component design, verification procedures, etc. However, PLE is not only a task that is reflected in the development process of an organization. It is necessary to consider the business aspect of PLE. Moreover, the architecture of the system has to be adapted to support the variability. A change in all of these aspects usually also requires an organizational restructuring. The business, architecture, process and organization aspects are discussed in the book by Linden et al. [78]. Furthermore, they discuss industrial applications of PLE and discuss the effect of PLE on the adopting organizations by providing feedback from industrial experiences.

2.3 Model-based Product Line Engineering

To manage variability in the solution domain with models, several different approaches are proposed. One possibility is the annotative approach in which all system variants should be modeled within a single over-specified model (sometimes called *150% model*) [28, 52]. The annotated product line model contains all system variations within a single model. In this approach, variants are created by removing features, which are not selected from the over-specified model, leaving only the desired parts in the system. An annotative approach for the management of variability in SysML models is also the main focus of the research in this thesis.

In an annotative approach, SysML can be extended with profiles and stereotypes to support variability [83]. This approach does not require a separate variability management tool. However, it does not support modeling of variability in SysML behavior diagrams, which are an important part in the system design. Another approach for modeling variability with SysML is presented in [54]. But then again, it is not clear how variability is captured in behavior models. As stated in an extensive literature survey [84], activity and state machine diagrams are widely used in systems engineering.

An approach that allows the combination of variability models from the problem domain with over-specified models from the solution domain is proposed in [63]. Orthogonal Variability Modeling (OVM) is an annotative approach which enables the annotation of SysML models with variability constructs. In this case, both design and variability are presented in the same model with explicit links between variants and parts of the design models.

In contrast to annotative approaches, there exist compositional approaches in which features are created as reusable assets and then combined together to create a single product variant. Such approaches are mainly based on feature-oriented programming, [12, 8], but applied on graphical models (instead of code) [77, 7, 80]. In such approaches, variants are, usually, created through automated model transformations which compose the individual models of features into a complete, interconnected, system variant that contains only the desired features.

In addition to compositional and annotative approaches, there exists the delta-oriented programming approach [71] or delta oriented modeling when applied to graphical models instead of code [70]. In the delta-based approach, the product line is firstly implemented as a single system, which implements the common features of all systems in the product line. Then, a set of deltas, i.e. a set of changes and steps on how to apply the changes is defined. The deltas represent different features of the product line. Depending how the base system is changed, i.e. which deltas are applied to the system, it is possible to derive different variants.

2.4 Analysis of Models in Product Lines

An important aspect to consider in PLE is quality assurance. In contrast to traditional single-product development, in PLE, the whole product family (i.e. all variants) must be verified and validated. When looking from a problem domain perspective, where feature models are the most commonly used way of representing variability [17], extensive research has been done on their analysis [15, 16]. Moreover, the same analysis approaches can be extended to other types of variability models such as OVM [67, 64]. These variability models can be expressed as a Boolean satisfiability problems and SAT solver-based methods are the most common approach for analysing and verifying the variability models [55, 13].

Other than analysing the variability models, one needs to consider that the implementation of the product line might introduce additional dependencies and constraints between features. Model checking in the PLE domains has been a topic of interest almost since model-based product line engineering has been introduced [69]. When looking at model-based product lines, work has been done on the mentioned types of product lines, i.e. over-specified product lines [4, 27], composition-based product lines [3], as well as delta oriented product lines [48].

Shortly after introducing the concept of over-specified product line models in [28], the authors have noted that such types of product line models tend to become complex even with a small number of features [4]. The reason is that every possible product variant is implicitly described in a single-system model. Furthermore, they noted that such errors usually happen in only a small number of system variants. To deal with these inconsistencies, the authors have introduced new semantics for OCL, which they then apply on the specification of UML class diagrams, instead of their instances as it is intended with OCL and UML. Similar work has been done on UML class diagrams by Buchmann et al. in [24].

However, in the work of this thesis, the annotative approach differs in the way how relations on model elements are handled. A relation can be removed from the over-specified model when instantiating a variant even if it is not explicitly annotated itself. In our approach, a relation will be removed if any of the two related blocks are to be removed when creating a variant. Thus in contrast, our approach can result in faulty variants even though the variant model is well-formed with respect to the underlying metamodel. Section 4.2 describes our approach for handling relations during the derivation process as well as the analysis approach for over-specified product line in more detail.

2.5 Product Line Engineering in the Industry

MBSE with SysML has been widely used in the industry in several domains, for example, in aerospace [5] and in automotive domains [40].

Related to the adoption of PLE in industrial settings, a number of articles can be found. In [36], experiences from introducing PLE at General Motors are reported. In [9], the application of PLE and exploration of MBSE was reported. It was done in a domain related to the use case organization in this thesis: aircraft engine systems. They explored MBSE to model at the system level but reported limited success due to the fact that intended users of the model were not familiar with the modeling language. Significant time savings, however, were reported during requirement elicitation. Another study [58], published by researchers from the Mitsubishi Electric Corporation, was also done in a very similar context to ours. In [58], the development of the actual product itself was outsourced to a technology provider. Over a five-year period, they report a gradual decrease of productivity after applying PLE, attributed to architecture erosion during application engineering and improper implementation of mechanisms for product derivation. The product line at Testo AG was evaluated in [45]. Results indicated that PLE allows organizations to develop more complex products while maintaining the costs and development lead time. A study on the use of MBSE within product lines from the rail transportation industry is presented in [39]. They report on first steps towards introducing MBSE into PLE by creating a product line of a metro train. The product line includes development artifacts from which all possible alternatives can be derived. The Orthogonal Variability Modeling language [63] was used to define problem space variability. They estimate cost savings on fixed engineering costs in the specification phase for up to 50%.

Moreover, a number of industrial experience reports can be found in the book by Linden et al. [78]. The book summarizes the effects of PLE on business, architecture, process and organizational aspects. It also includes the results of applying PLE in the combustion engine domain at Bosch [74, 76], reporting that a clear understanding of the process prior to the introduction of PLE was an important success factor. Addressing the process part at Volvo CE is also a large part of the research work performed in this thesis.

A systematic mapping study on the adoption of PLE in industrial practice can be found in [11]. The authors describe several adoption approaches based on the current literature and compile a summary of adoption barriers that should be considered in practice. The experience report from Danfoss [37] is another long-term evaluation report of PLE, reporting on the complete PLE experience from planning to technical execution and evaluation. In the industrial report from [59], it was noted that most of the challenges when adopting product lines and variability modeling in an automotive context arise from the lack of modeling guidelines and limited scalability of current approaches.

Chapter 3

Research Overview

This chapter provides the research problem formulation through a description of the industrial context in which the work for the thesis was performed. Furthermore, the research goals which have driven the work are described. The reasoning for the use of research goals was to allow flexibility in the research studies. The research followed an exploratory process, meaning that the results of the one research goal prompted the definition of subsequent research goals.

3.1 Research Goals

The overall goal of this thesis, which we aim to accomplish is stated as follows:

Thesis Goal: *Apply, facilitate and evaluate model-based product line engineering with SysML in an industrial environment.*

The main motivation for the migration from a traditional, clone-and-own-based, reuse and development approach towards a model-based PLE development approach was, firstly, to manage complexity with graphical models and secondly, to improve upon the currently inefficient reuse practices within the use case provider.

As it is with any complex problem, there is rarely a silver bullet that can solve all of the problems' aspects. Therefore, we approach the variability management in MBSE challenge carefully by, firstly, describing the challenge related to variability in more detail and evaluating how currently existing tool support at Volvo

CE can cope with the challenge.

Research Goal 1: *Identify challenges when overloaded variability models are used to manage variants in existing industrial systems.*

This goal has driven research that explored how the annotative modeling approach with over-specified (also known as 150%) models can be used to describe variability in SysML diagrams of already implemented systems. The intent was to explore potential benefits as well as issues and challenges, which might arise when such an approach is applied in the described industrial context.

Research Goal 2: *Identify reuse-related challenges in a "clone-and-own" based development process, and assess the effects of model-based product line engineering on the identified challenges.*

The second goal expands the scope of the first research goal. over-specified models significantly increase complexity of models, even when simple systems are represented. This research goal has driven research that explored the feasibility of applying annotative modeling in other phases of the development cycle, from requirement engineering to testing.

Research Goal 3: *Define and develop an extensible toolchain for model-based product line engineering that allows integration of non-model artifacts with SysML models and supports consistency checking between variability models and SysML implementation models.*

The third research goal has two parts. Firstly, it states the need for a model-based PLE tool that allows integration of non-model elements to the product line. Secondly, it emphasizes the fact that it is necessary to define a method for maintaining the consistency between variability models in the problem domain and system models in the solution domain.

3.2 Research Process

Initially in this thesis, the research was primarily based on qualitative research methods [41], i.e. research methods which help us understand and describe the

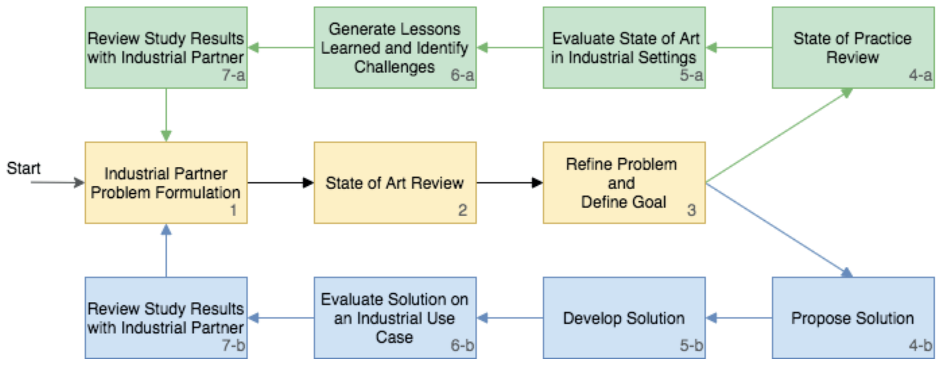
Review Study Results with Industrial Partner 7-a
Generate Lessons Learned and Identify Challenges 6-a
Evaluate State of Art in Industrial Settings 5-a
State of Practice Review 4-a
Start
Industrial Partner Problem Formulation 1
State of Art Review 2
Refine Problem and Define Goal 3
Review Study Results with Industrial Partner 7-b
Evaluate Solution on an Industrial Use Case 6-b
Develop Solution 5-b
Propose Solution 4-b

Figure 3.1: Illustration of the research process for this thesis

characteristics and concepts applied during the development of engine systems at Volvo CE.

The two case studies (papers A and C) were aimed at understanding the reuse phenomena in industrial settings by performing empirical exploratory case studies and generating research problems and research questions in an inductive manner. Through observation and identification of patterns, we have tried to reason about the reuse phenomena, derive conclusions and define further research questions. These studies were following the yellow-green colored loop in Figure 2.1 and were performed by following guidelines from Runeson and Höst [68]. As the case organization was already known beforehand, the unit of analysis was defined based on case organizations' initial problem formulation.

- The first step in both case studies (A and C) was to formulate the initial problem that was intended to be evaluated at Volvo CE. Based on the problem formulation, a suitable unit of analysis was defined.

- After the initial problem formulation, a review of the state of the art was carried out in step 2 of Figure 3.1.

- In step 3, based on the identified research in the state of the art, we have refined the problem formulation and defined research goals.

- The next point (step 4-a) was to identify the state of practice that is currently present in the organization under analysis.

- Then in step 5-a, the state-of-the-art approaches were evaluated in the industrial settings and compared to the current state of practice.
- Lessons learned were generated and general challenges were pointed out in step 6-a.
- In the end, in step 7-a, the studies were further reviewed by the industrial partner and actions for further research were identified.

On the other hand, in studies B and D, the focus was on the implementation of solutions for identified challenges from papers A and C. These have followed the yellow-blue colored loop in Figure 2.1.

- After reviewing the lessons learned an challenges when applying state-of-the-art tools, together with the industrial partner, a new problem formulation was defined in step 1. The problem formulation was intended to drive research towards addressing the previously identified challenges.
- Again, a state-of-the-art analysis on that particular problem was performed in step 2.
- Further, in step 3, the problem formulation was refined and more specific research goals were defined.
- Based on the state-of-the-art review results and research goals, a potential solution was proposed in step 4-b.
- Then, the solution was developed, as illustrated with step 5-b. After the development, the solution was evaluated on a small-scale study in step 6-b.
- In the end, in step 7-c, the solution was reviewed by the industrial partners and room for further improvement was identified.

The yellow-blue loop was an iterative loop in which one solution proposal could be refined a number of times, with the extension of the study's scope until the industrial partner needs were satisfied.

Chapter 4

Thesis Contribution

This chapter gives an overview of the contributions which are a direct result of research work driven by the previously described research goals. This chapter is divided into two sections. Figure 4.1 illustrates the relations between the individual research goals, publications and how these resulted in the contributions presented in this chapter.

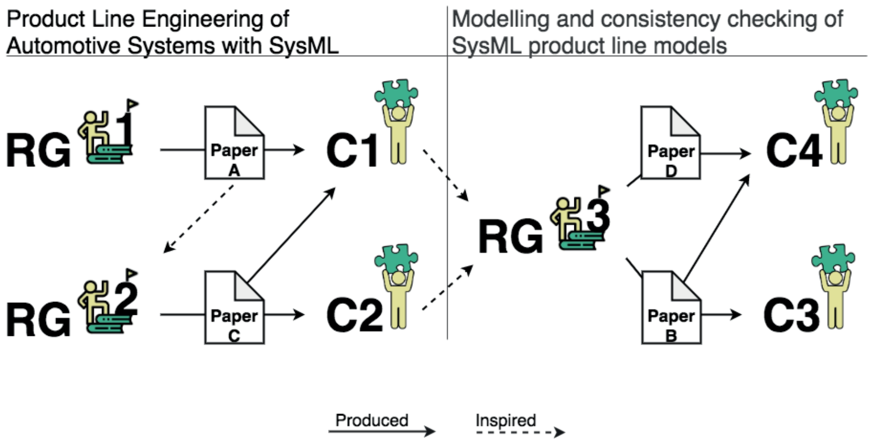
Product Line Engineering of Automotive Systems with SysML
Modelling and consistency checking of SysML product line models
RG 1
Paper A
C1
RG 2
Paper C
C2
RG 3
Paper D
C4
Paper B
C3
Produced
Inspired

Figure 4.1: Relation between research goals, papers and contributions.

Illustrated on the left side of Figure 4.1, the first part of this chapter describes two contributions about the state-of-practice in model-based PLE as well as feedback from an industrial perspective on the challenges and benefits of its application in an industrial pilot study. We have initially defined RG1, and the work toward it resulted in Paper A. These results are also the basis for Contribution 1: an evaluation of the annotative approach for variability management in SysML models of hardware and software in an industrial case study. The decision to go with the annotation-based approach is driven by the fact that this approach is commonly implemented in commercial SysML modeling tools which support variability modeling [85, 43, 30].

The results of Paper A inspired RG2, which in turn resulted in Paper C. The results of Paper C were the basis for Contribution 2, but also contributed towards C1. In Contribution 2, we describe the challenges of the clone-and-own practice throughout the complete development process of Volvo CE. Then, we perform an industrial pilot study and assess the potential benefits and challenges when model-based PLE is used to manage variability throughout the complete development process, from requirement engineering to testing.

The second part of this chapter is illustrated on the right side of Figure 4.1. Both C1 and C2 suggested that not all aspects of a system can be modeled with SysML, and that it might be necessary to allow for inclusion of non-model artifacts in the product line. Moreover, a need for automated consistency checking of the annotated SysML product line models was identified. Thus, RG3 has firstly driven the work in Paper B, where a toolchain for variability management with SysML was developed that supports integration of non-model artifacts, but also changes the way how certain relations in SysML diagrams are handled. Contribution 3 is mainly based on the results from Paper B. Lastly, the work towards Contribution 4 extended the work from Paper B and resulted in Paper D: a consistency-checking approach for the detection of variability-related anomalies such as dead features, false optional features and dead relations, which can appear in variants of the SysML product line model.

4.1 Product Line Engineering of Automotive Systems with SysML

Contribution 1

The first contribution is based on findings from two case studies (Papers A and C), which were conducted on the Urea Dosing System (UDS) at Volvo CE. The UDS is a part of a diesel engine used in Volvo CE machinery. Its role is to reduce

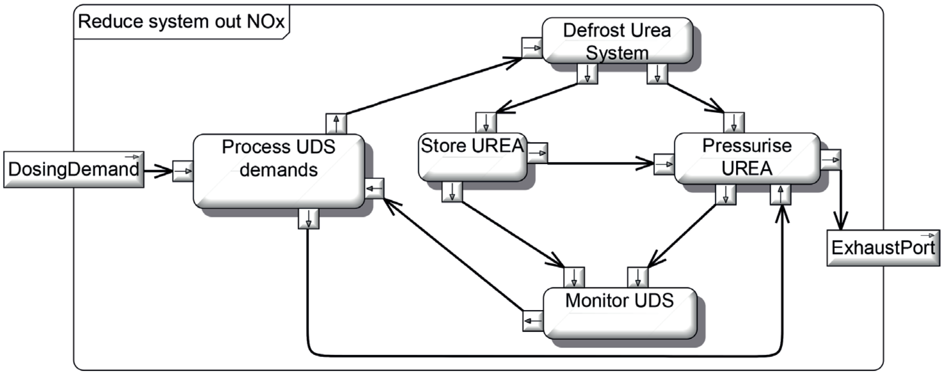

Figure 4.2: Urea Dosing System Functional Architecture

the amount of Nitrogen Oxides (NOx) in the exhaust gasses of the diesel engine. The functional architecture of the UDS is shown in Figure 4.2. It is injecting a special fluid (urea) into the exhaust stream and initiating a chemical reaction that reduces harmful emissions. Urea is stored in a tank within the UDS until a demand from the engine system is received. In addition to storage and injection, the UDS has a defrost functionality, which ensures that the system can operate in cold conditions.

The UDS system has been modeled with all variable features and combined into a single overloaded SysML model. The model was annotated with Orthogonal Variability Modeling constructs, which maps variability from the variability diagram to the corresponding system artifacts. A successful derivation of individual UDS variants from the overloaded mode has been demonstrated, however, with some challenges:

- *Complexity.* The process of creating the overloaded models was complex and prone to errors. The system, which was modeled for Contribution 1, was rather small with 11 features in total. When combined into a single model, it was difficult to manually analyse, especially for engineers that do not have experience with annotative modeling. The added complexity of annotative modeling makes scalability a great challenge.

- *Scalability.* The added complexity of models, and partially, the scalability issues can be addressed through introducing several layers of abstractions during the design process of models. It is necessary to organise the model in a way, which will reduce the scattering of elements belonging to the same feature. Although cross-cutting feature elements are explicitly exposed with annotations, they tend to increase complexity of models. The possibility to separate concerns mainly depends on the system itself, whether it can be separated into cohesive components or not. In cases where the system architecture requires a large number of cross-cutting features, it might be more feasible to redesign the architecture rather than dealing with many such features afterwards, as it increases complexity of models significantly. With many cross-cutting features, it is also difficult to understand feature dependencies.

These challenges were mainly observed when modeling the physical architecture of the UDS. Modeling all hardware components within a single model and having multiple parts connect to the same interface in different ways significantly increases the perceived complexity of the system, i.e. the model becomes more complex than what the actual system is. However, the benefit of such models is that they facilitated the analysis of feature dependencies. By having all features in a single SysML model, with all relations towards other elements, explicitly modeled, it is possible to (manually) trace feature-related model elements to all parts (and features) with which a certain feature interacts.

In contrast to the physical architecture, modeling the software in this approach did not introduce additional complexity. The reason is that the software was already designed in an over-specified fashion. All variants of the UDS were based on the same software, which was developed in an approach where various parts software platform could be enabled or disabled by setting necessary parameters. That is also the basic idea behind annotative product lines, features are enabled or disabled based on whether feature presence conditions evaluate to true or false.

The necessity of a development process that takes variability into account at early stages of development was brought to light by the lessons learned from Research Goal 1. It became obvious that configuration management of the implementation artifacts alone is not enough and that variability must be taken into account starting right from the project planning and customer requirements. This means that management of variability should be taken into account in a systematic fashion across different phases of the system development process.

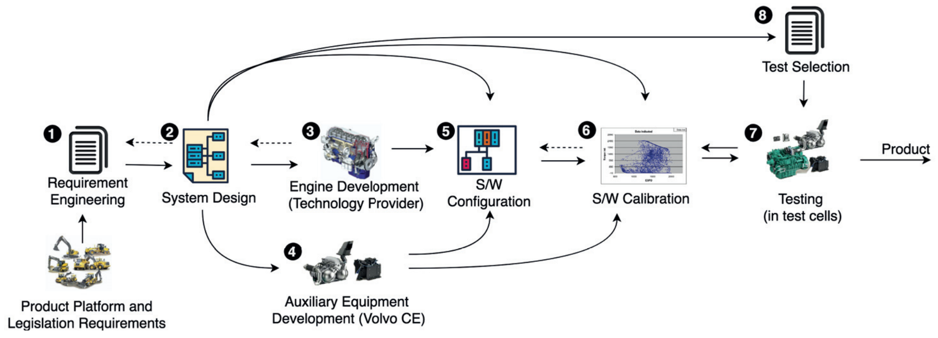
8
Test Selection
1
Requirement Engineering
2
System Design
3
Engine Development (Technology Provider)
5
S/W Configuration
6
S/W Calibration
7
Testing (in test cells)
Product
Product Platform and Legislation Requirements
4
Auxiliary Equipment Development (Volvo CE)

Figure 4.3: Current development process at the Engine Controls department at Volvo CE

Contribution 2

In the second contribution, we strive to, firstly, document the current development process of the Engine Controls department at Volvo CE and secondly, to propose a model-based PLE approach based on the current development process. The contribution is a systematic definition of a model-driven PLE process at the Engine Controls department of Volvo CE.

The current development process is illustrated in Figure 4.3. Based on an interview study through which the process was mapped, several reuse-related challenges were identified.

During requirements engineering in the first phase of the process, the major challenges were attributed to the fact that expert knowledge was not always readily accessible during requirement elicitation. This lead to issues like requests for obsolete and difficult-to-maintain features. Dependencies between various variable parts of the system are not explicitly documented, leading to requirement change requests in later phases of the process, inducing delays and requiring rework of some aspects of the system. It was noted that there was a large amount of duplicate documents for each system variant individually.

During the system design, challenges were related to the fact that specifications are based on natural language where graphical illustrations have no unified format. It was reported that reuse of system components required extensive experience in the domain, mainly due to outdated and ambiguous specifications and lack of traceability between requirements and system design. Improper reuse of one system part often resulted in unexpected behavior in other aspects of the

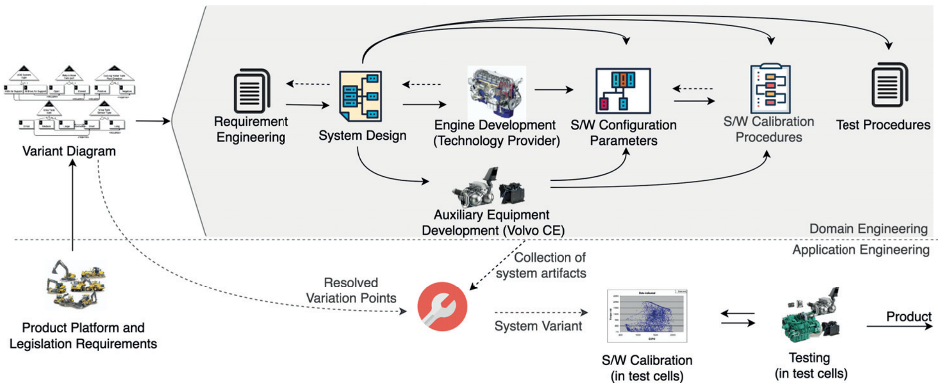
Variant Diagram
Requirement Engineering
System Design
Engine Development (Technology Provider)
S/W Configuration Parameters
S/W Calibration Procedures
Test Procedures
Auxiliary Equipment Development (Volvo CE)
Domain Engineering
Application Engineering
Collection of system artifacts
Resolved Variation Points
Product Platform and Legislation Requirements
System Variant
S/W Calibration (in test cells)
Testing (in test cells)
Product

Figure 4.4: Proposed product line engineering process

system, requiring significant testing efforts to isolate the source of errors. Dependencies between variable system features were not explicitly documented in the current development process, resulting in repetitive bottom-up analysis of the system in order to understand dependencies.

Software calibration and configuration (Phases 5 and 6 in Figure 4.3) are time consuming tasks as engine system have more than ten thousand software parameters. Incorrect performance goals (due to lacking requirement elicitation) can result in a need to re-calibrate, inquiring a significant cost as physical engine calibration cells are a scarce resource. The source of such issues is often the large variability of requirements for performance parameters. Software calibration and configuration maps are manually cloned from previous projects, however, it is not always obvious from which variant these should be cloned.

Challenges regarding testing are mainly related to the fact that testing often focuses on aspects of the system, which did not change compared to the variant from which the system was cloned. This issue comes from the fact that feature dependencies are not explicitly documented, resulting in the need to test every functionality even if it did not change.

In the second part of Contribution 2, we have proposed an adaptation of the current development process towards a PLE-based process, as illustrated in 4.4. Further, we discuss the implications model-based product line engineering (MB-PLE) may have on the identified challenges as well as system artifacts.

Requirement elicitation should start with the variability diagram as illustrated in Figure 4.4. The variability diagram describes the variability of the complete system and traces to all phases of the development process, from requirements to testing and test procedures. It has been proven as a useful tool for managers, as it provides them with an overview of the dependencies and implications between features without requiring extensive domain knowledge.

MBSE can help with issues such as outdated, ambiguous, incomplete information and traceability, especially during the system design phase [75]. However, variability still needs to be taken into account by tracing the variation points to all relevant system artifacts. Constraints and dependencies in the variability model from the problem domain restrict the system such that incompatible components can never appear in any variant of the system. This correctness is ensured by the mapping from the problem domain variability model to the system model in the solution domain. The mapping, however, is performed manually and must also be maintained manually.

PLE has also shown a positive effect during the testing phase. Usually, test engineers need support from system engineers to understand the expected outcome of test cases, a process that can sometimes take a significant amount of time. With PLE however, the test procedures can be automatically derived based on the features, which are present in the system variant.

After comparing the existing document-based development process to the model-based PLE process, it was noted that although it might be possible to represent the necessary system information in models, it would have been useful if flexibility was provided that allows to include system artifacts, which are maintained by other means, such as calibration parameters files or documents that are developed by other departments, which do not follow the MBSE approach.

Moreover, it was noted that the consistency between the variability model and SysML model annotated with variability had to be maintained manually. To address these needs, we had defined research goal 3, which was addressed through contributions 3 and 4.

4.2 Modelling and Consistency Checking of Product Line Models

It was noted in both industrial case studies (Papers A and C) that over-specified models can introduce additional constraints on features in the product line, e.g. due to well-formedness requirements for all variants of the product line. However, existing modeling tools that support variability modeling and SysML did not provide us with the ability to extend their functionality, e.g. to add consistency checking. Moreover, the SysML modeling tools do not provide the ability to integrate non-model feature artifacts into the product line. We firstly started by addressing the latter requirement in Contribution 3. The former requirement has been addressed through Contribution 4.

Contribution 3

We have extended an existing SysML modeling tool to support variability modeling and integrate it with state-of-the art variability management tools in order to create a toolchain that allows us to extend it with additional functionality such as consistency checking, but also supports the integration of variable non-model-based development artifacts (e.g. code, documents etc.). This is done by using the pure::variants[1] tool to manage variability in the problem domain and the Modelio[2] modeling tool to model the system in the solution domain. In order to integrate the two tools, the Variability Exchange Language (VEL) was used. It allows us to exchange variability information between the problem and solution domain. The toolchain has been applied in industrial settings and an evaluation according to the quality criteria, [78], for product lines was performed. The main aspects of this contribution are:

- The definition of mechanisms for how an 100% model should be instantiated from an 150% model based on the feature selection. Compared to other approaches ([28, 50]), we have defined that during the derivation process, removing a block removes all its properties, associations, aggregations, compositions, generalizations and dependency links connected to the block. This approach ensures well-formedness, but comes with the drawback that there might be a chance that model variants can be derived with missing features. This issue is later addressed in Contribution 4.

- The evaluation according to the family evaluation framework for product lines [6]. Regarding the *pre-planning effort*, our approach enables the product line to be annotated as system models are evolving, with little pre-planning. For *traceability*, a feature can be mapped to any number of model elements that implement the given feature. Considering the *separation of*

[1]https://www.pure-systems.com/
[2]https://www.modelio.org/

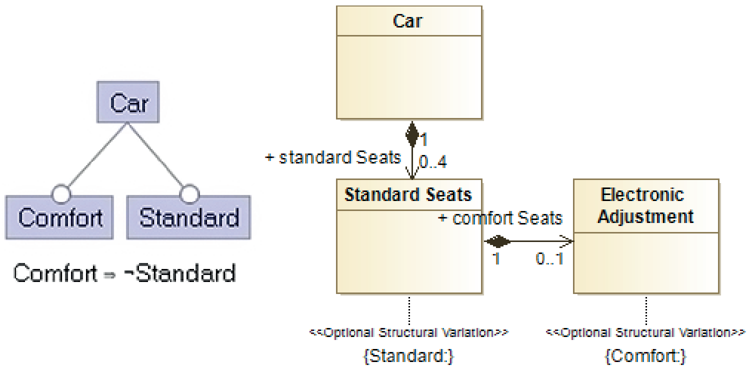

Figure 4.5: Left: Problem domain, Right: Solution domain

concerns, annotating model elements with variability information explicitly exposes cross-cutting features, thus facilitates separation of concerns even when there exist cross-cutting features. Encapsulated model elements of any feature can be annotated with variability, thus enabling *information hiding* with regards to features. When looking at *granularity*, our approach allows annotation on all model element types and properties of such elements, as well as annotation on the relationship links between elements. And finally, the toolchain provides good *uniformity* as all product artifacts are SysML model elements and are annotated in the same fashion.

By extending the system modeling tool with variability mechanisms in Contribution 3, we have also created a setup, which we can use to define and test methodologies for consistency checking between the variability model and SysML model, which was done in the last contribution.

Contribution 4

As variability models constrain how the implementation (SysML) models can be configured, it is crucial for the models to be consistent. Features with constraints in the problem domain (e.g. in a feature model) might implicitly have additional conflicting constraints resulting from the over-specified product line model. These issues are not always obvious and are difficult to identify manually.

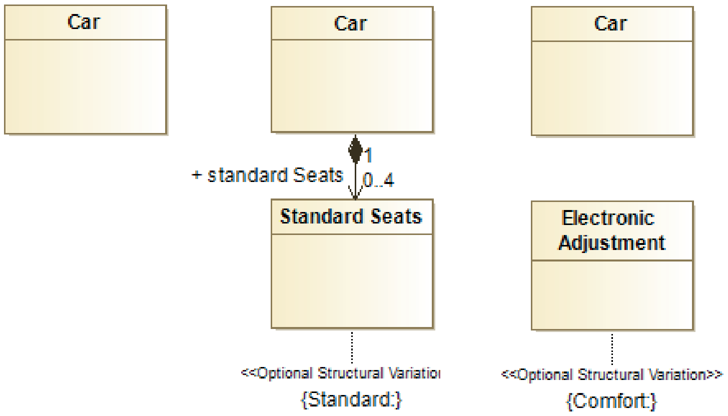

Figure 4.6: Left: V_1, Middle: V_2, Right: V_3

Consider the example in Figure 4.5. The feature model (left) describes that the *Car* has two mutually exclusive features: *Standard* and *Comfort*. The system model (right) describes the implementation of the system. The *Car* contains *Standard Seats* (which are mapped to the *Standard* feature) and the *Electronic Adjustment* (mapped to feature *Comfort*) is specified as part of the *Standard Seats*. There are three possible variants of the system, and each is well-formed when modeled in the toolchain from Contribution 3. The variants are listed below and also illustrated in Figure 4.6:

$$V_1 = \{Car\}$$

$$V_2 = \{Car, Standard\}$$

$$V_3 = \{Car, Comfort\}$$

Standard and *Comfort* are mutually exclusive, yet, elements to which these features are mapped, *Standard Seats* and *Electronic Adjustment*, are related through a SysML composition relation. The *Electronic Adjustment* is specified as part of *Standard Seats*. Therefore, the variant V_3, although a well-formed model and a valid combination of features is suspicious from a system perspective, it does not make sense to have electronic adjustment of seats if the seats do not even exist in the system. Due to the constraint from the problem domain, the composition relation between *Standard Seats* and

Electronic Adjustment (shown in Figure 4.5) can never exist in any variant as illustrated in Figure 4.6.

Although obvious in this small example, considerable effort is necessary to manually detect such issues in larger systems.

In Contribution 4, we define an approach to identify inconsistencies between the variability model and feature relations in the over-specified system model. We firstly restrict the use of certain SysML relations for annotative modeling with the aim to avoid ambiguous model constructs. Secondly, we extract feature constraints from the over-specified system model and analyze them together with feature constraints from the variability model with the goal of identifying variability-related inconsistencies.

The rules for use of SysML in annotative modeling are as follows (quoted from Paper D):

- *Each relation in the over-specified model must be able to exist in at least one instance of the product line model.*
- *If a block is a composite part of (owned by) at least one other block, then it requires the presence of exactly one owning block in each product-family configuration (variant) in which it exists itself.*
- *If a block is a subtype of at least one other supertype, then it requires the presence of exactly one supertype in each of the product-family configurations in which it exists itself.*
- *A feature-annotated block property requires the presence of the block, which owns the property in all product-family configurations in which the property exists itself.*
- *A feature-annotated block property requires the presence of the data type of that property.*
- *A feature-annotated operation within a block requires the presence of the block which owns the operation in all product-family configurations in which the operation exists itself.*
- *If a client block A has a dependency to supplier block B, then it requires the presence of the supplier block in all the product-family configurations in which the client exists.*

Based on these rules, propositional logic constraints are extracted from the system model (model on the right side of Figure 4.5). These are then combined with constraints from the variability model (left side of Figure 4.5). By combining the two sets of constraints, we can analyse the system to detect:

- Dead features – features which can never exist in any variant of the product line
- False optional features – features, which are specified as optional in the variability model, but must actually be present in all variants of the product line.
- Dead relations – relations between model entities, which can never exist in any variant of the product line (an example is the composition relation between *Standard Seats* and *Electronic Adjustment* from Figure 4.5).

Promising results were obtained on an prototype evaluation on the product line models from Papers A and C. During the analysis, we have detected several inconsistencies on models that have gone through several iterations and which were considered as complete by the engineers within the organization that provided the use cases.

Contribution 4 also concludes the research work in this Licentiate thesis.

Chapter 5

Summary

5.1 Discussion

PLE makes the greatest promises in terms of a reduction in development efforts while increasing product quality [10, 78]. Although these studies describe success stories with PLE, all of them report that there is no silver bullet. In order to succeed, the processes and methods must be tailored to suit the specific needs of the organization that is adopting the practices.

In [78], nine large organizations report on their experiences on the adoption of PLE. Yet, each of them described a slightly different approach of doing it. For example, looking only at the highest level of documenting variability, two organizations reported the use of standard feature models or a very similar concept, while others developed their custom approaches for variability modeling. This was also the case during the research in Papers A, B and C. We looked into two common approaches, feature modeling and OVM. Although they are essentially very similar, it was noted that both had their benefits in different situations. For example, OVM was seen as more useful when adding new variation points especially when getting into fine-grained variation points, while feature models were seen as a better approach for management due to the overview of the complete variability space, which it provides.

Going further into the product line implementation, in Papers A and C, we have used an annotative approach to model a subsystem of the engine. Initially, the annotative approach worked well for the software part of the system. The reason was that the software already had been developed for quite some time in an approach where various parts of the software platform could be enabled or disabled by setting necessary parameters. That is also the basic idea behind annotative product lines: features are enabled or disabled based on whether feature presence conditions evaluate to true or false. However, from a hardware perspective, the annotative approach causes the models to become quite complex even though a small number of features was modeled. A composition-based approach could have worked better as variants of engine systems were built on a platform on which parts were either replaced with different ones or completely new parts were added. The interfaces between physical parts and the engine platform did not change for the majority of feature variations. Thus, instead of an annotative approach, modeling each feature separately from the platform would allow the system model to be composed based on selected features. This compositional type of approach would avoid the issue of having multiple features connect to the same interfaces in a single model, as well as avoid the need of consistently removing features which were selected from the model.

Moreover, the need for automated analysis of annotated product line models was identified in Papers A and C and addressed in Paper D. We have developed a consistency checker that ensures well-formedness of variants of the product line model, but also ensures that the product line model itself follows our SysML modeling rules as described in Section 4.2. Although the consistency checker has proven to be useful, it can still detect only syntax errors. To this end, we plan to extend our work and allow users to create custom rules that can be used to check different properties on the complete product line.

From a tooling perspective, the most comprehensive commercial MBSE tool is the Integrity Modeler[1], which supports the UML and SysML standards as well as OVM as a variability modeling language. Other specialized tool solutions for variability modeling are BigLevers Gears[2], and pure::variants[3]. These variability modeling tools enable variant configuration management and can be connected to other modeling tools through means of plugins. Other academic tools include FeatureIDE[4] and the BVR tool[5], tools for configuration management and feature modeling. They can be used, for example, with Eclipse Modeling Framework[6]

[1]Integrity Modeler by PTC - https://www.ptc.com
[2]BigLever Software Gears - https://biglever.com/
[3]pure::variants - https://www.pure-systems.com/
[4]FeatureIDE, https://featureide.github.io/
[5]BVR tool https://github.com/SINTEF-9012/bvr
[6]Eclipse Modeling Framework, https://www.eclipse.org/modeling/emf/

compatible models to enable PLE. A comprehensive overview of variability management tools was written by Pereira et al. and can be found in [62].

Nevertheless, technical solutions are only a part of the actual problem solution. The questions of solution ownership, education, training, etc., sometimes are more important than the actual technical solutions such as the ones proposed in this thesis. Organizational change is not only a technical task. After all, it is a cultural shift with humans in the center of change.

5.2 Conclusion

The research had started with a more general objective: to apply model-based PLE in an industrial environment. Thus, the research was initially based on an industrial use case where we performed pilot studies with the aim to understand how model-based PLE can help with configuration management in hardware and software of already existing variable systems (Contribution 1, Contribution 2).

The main point was that PLE increased the complexity of system models significantly when variability in physical components was considered, while it enabled more efficient variability management in software. Then, the scope of the industrial case study was expanded to the complete development process at Volvo CE Engine Controls. It was shown that model-based PLE can be particularly useful during requirement engineering as it makes the system variability the main factor for decision making. Moreover, it was demonstrated to be of great benefit during testing because it allowed for automated test procedure derivation.

Based on results and feedback from the industrial case studies, we continued the work on developing an open source toolchain that allows integration of non-model elements into the model-based product line. Compared to currently existing approaches, we have changed the mechanism for handling relations and model elements when deriving individual variants from an over-specified model (Contribution 3).

Furthermore, we have defined modeling rules for creating over-specified models in order to avoid ambiguities during the derivation process. Then, based on these rules, we have developed a consistency checking approach, which ensures that all system variants will be well-formed and that there exist no anomalies in the product line such as dead features, false optional features and dead relations between model elements (Contribution 4).

5.3 Future Work

In the close future, the plan is to extend the consistency checking approach beyond block definition diagrams, to ensure well-formedness for over-specified models across multiple model views. Further, we want to define modeling guidelines and a modeling methodology for over-specified SysML product lines.

As mentioned in the background section, there exist several methods for modeling product lines, the main three being: annotation-based, composition-based and delta-based product lines. An interesting study would be to perform an industrial case study and compare modeling approaches belonging to these three groups and to try to assess their benefits and drawbacks for representing various system aspects, e.g. requirements, system (SW and HW) design, tests, etc.

Another potential direction is the exploration of automated test generation from over-specified product line models. Model-based testing has been extensively studied in the literature ([73, 53, 34, 21, 32]) as well as product line testing ([31, 47, 2]). Since over-specified product line models implicitly describe all variants within a single model, the number of generated test cases can be very large. Although there is research on model-based testing of product lines ([61, 49, 66, 60]), it would be interesting to consider if the extracted constraints as described in Contribution 4 can be utilized for test prioritization or reduction of the generated test sets, etc.

Bibliography

[1] W. Afzal et al. The MegaM@Rt2 ECSEL Project: MegaModelling at Runtime - Scalable Model-Based Framework for Continuous Development and Runtime Validation of Complex Systems. *Microprocessors and Microsystems*, 61:86 – 95, 2018.

[2] M. Al-Hajjaji, S. Krieter, T. Thüm, M. Lochau, and G. Saake. Incling: efficient product-line testing using incremental pairwise sampling. *ACM SIGPLAN Notices*, 52(3):144–155, 2016.

[3] M. Alférez, R. E. Lopez-Herrejon, A. Moreira, V. Amaral, and A. Egyed. Supporting consistency checking between features and software product line use scenarios. In *International Conference on Software Reuse*, pages 20–35. Springer, 2011.

[4] M. Alférez, R. E. Lopez-Herrejon, A. Moreira, V. Amaral, and A. Egyed. Consistency checking in early software product line specifications-the vcc approach. 2014.

[5] H. Andersson et al. Experience from introducing unified modeling language/systems modeling language at saab aerosystems. *Systems Engineering*, 13(4):369–380, 2010.

[6] S. Apel, D. Batory, C. Kästner, and G. Saake. *Feature-oriented software product lines*. Springer, 2016.

[7] S. Apel, F. Janda, S. Trujillo, and C. Kästner. Model superimposition in software product lines. In *International Conference on Theory and Practice of Model Transformations*, pages 4–19. Springer, 2009.

[8] S. Apel and C. Kästner. An overview of feature-oriented software development. *Journal of Object Technology*, 8(5):49–84, 2009.

[9] M. J. Atherton and S. T. Collins. Developing product lines in engine control systems: Systems engineering challenges. In *INCOSE Intl. Symp.* Wiley, 2013.

[10] J. L. Barros-Justo, F. Pinciroli, S. Matalonga, and N. Martínez-Araujo. What software reuse benefits have been transferred to the industry? a sys-

tematic mapping study. *Information and Software Technology*, 103:1 – 21, 2018.

[11] J. F. Bastos, P. A. d. M. S. Neto, E. S. de Almeida, and S. R. de Lemos Meira. Adopting software product lines: A systematic mapping study. In *15th Annual Conference on Evaluation & Assessment in Software Engineering*, pages 11–20. IET, 2011.

[12] D. Batory, J. N. Sarvela, and A. Rauschmayer. Scaling step-wise refinement. *IEEE Transactions on Software Engineering*, 30(6):355–371, 2004.

[13] S. Ben-David, B. Sterin, J. M. Atlee, and S. Beidu. Symbolic model checking of product-line requirements using sat-based methods. In *2015 IEEE/ACM 37th IEEE International Conference on Software Engineering*, volume 1, pages 189–199. IEEE, 2015.

[14] D. Benavides, S. Segura, and A. Ruiz-Cortés. Automated analysis of feature models 20 years later: A literature review. *Information Systems*, 35(6):615–636, 2010.

[15] D. Benavides, S. Segura, and A. Ruiz-Cortés. Automated analysis of feature models 20 years later: A literature review. *Information systems*, 35(6):615–636, 2010.

[16] D. Benavides, S. Segura, P. Trinidad, and A. R. Cortés. Fama: Tooling a framework for the automated analysis of feature models. *VaMoS*, 2007:01, 2007.

[17] T. Berger, R. Rublack, D. Nair, J. M. Atlee, M. Becker, K. Czarnecki, and A. Wąsowski. A survey of variability modeling in industrial practice. In *Proceedings of the Seventh International Workshop on Variability Modelling of Software-intensive Systems*, pages 1–8, 2013.

[18] D. Bilic, E. Brosse, A. Sadovykh, D. Truscan, H. Bruneliere, and U. Ryssel. An integrated model-based tool chain for managing variability in complex system design. In *2019 ACM/IEEE 22nd International Conference on Model Driven Engineering Languages and Systems Companion (MODELS-C)*, pages 288–293. IEEE, 2019.

[19] D. Bilic, D. Sundmark, W. Afzal, P. Wallin, A. Causevic, and C. Amlinger. Model-based product line engineering in an industrial automotive context: An exploratory case study. In *Proceedings of the 22nd International Systems and Software Product Line Conference-Volume 2*, pages 56–63, 2018.

[20] D. Bilic, D. Sundmark, W. Afzal, P. Wallin, A. Causevic, C. Amlinger, and D. Barkah. Towards a model-driven product line engineering process: An industrial case study. In *Proceedings of the 13th Innovations in Software Engineering Conference on Formerly known as India Software Engineering Conference*, pages 1–11, 2020.

[21] R. V. Binder, B. Legeard, and A. Kramer. Model-based testing: where does it stand? *Communications of the ACM*, 58(2):52–56, 2015.

[22] M. Bone and R. Cloutier. The current state of model based systems engineering: Results from the omg™ sysml request for information 2009. In *Proceedings of the 8th conference on systems engineering research*, 2010.

[23] S. Bonnet, J.-L. Voirin, V. Normand, and D. Exertier. Implementing the mbse cultural change: organization, coaching and lessons learned. In *INCOSE International Symposium*, volume 25, pages 508–523. Wiley Online Library, 2015.

[24] T. Buchmann and B. Westfechtel. Mapping feature models onto domain models: ensuring consistency of configured domain models. *Software & Systems Modeling*, 13(4):1495–1527, 2014.

[25] R. Cloutier, B. Sauser, M. Bone, and A. Taylor. Transitioning systems thinking to model-based systems engineering: Systemigrams to sysml models. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 45(4):662–674, April 2015.

[26] S. P. Consortium et al. Reuse-driven software processes guidebook. *SPC-92019*, 1993.

[27] M. Cordy, X. Devroey, A. Legay, G. Perrouin, A. Classen, P. Heymans, P.-Y. Schobbens, and J.-F. Raskin. A decade of featured transition systems. In *From Software Engineering to Formal Methods and Tools, and Back*, pages 285–312. Springer, 2019.

[28] K. Czarnecki and M. Antkiewicz. Mapping features to models: A template approach based on superimposed variants. In *International conference on generative programming and component engineering*, pages 422–437. Springer, 2005.

[29] K. Czarnecki, P. Grünbacher, R. Rabiser, K. Schmid, and A. Wasowski. Cool features and tough decisions: A comparison of variability modeling approaches. In *Proceedings of the Sixth International Workshop on Variability Modeling of Software-Intensive Systems*, VaMoS ’12, pages 173–182, New York, NY, USA, 2012. ACM.

[30] H. Dubois, V. Ibanez, C. Lopez, J. Machrouh, N. Meledo, P. Mouy, and A. Silva. The product line engineering approach in a model-driven process. 2012.

[31] E. Engström and P. Runeson. Software product line testing–a systematic mapping study. *Information and Software Technology*, 53(1):2–13, 2011.

[32] E. P. Enoiu, A. Čau•ević, T. J. Ostrand, E. J. Weyuker, D. Sundmark, and P. Pettersson. Automated test generation using model checking: an industrial evaluation. *International Journal on Software Tools for Technology Transfer*, 18(3):335–353, 2016.

[33] J. A. Estefan et al. Survey of model-based systems engineering (mbse) methodologies. *Incose MBSE Focus Group*, 25(8):1–12, 2007.

[34] M. Felderer, P. Zech, R. Breu, M. Büchler, and A. Pretschner. Model-based security testing: a taxonomy and systematic classification. *Software Testing, Verification and Reliability*, 26(2):119–148, 2016.

[35] D. Flemström, D. Sundmark, and W. Afzal. Vertical test reuse for embedded systems: A systematic mapping study. In *2015 41st Euromicro Conference on Software Engineering and Advanced Applications*, 2015.

[36] R. Flores, C. Krueger, and P. Clements. Mega-scale product line engineering at general motors. In *Proceedings of the 16th Intl. Software Product Line Conference*. ACM, 2012.

[37] T. Fogdal, H. Scherrebeck, J. Kuusela, M. Becker, and B. Zhang. Ten years of product line engineering at danfoss: lessons learned and way ahead.

In *Proceedings of the 20th Intl. Systems and Software Product Line Conf.* ACM, 2016.

[38] S. Friedenthal, A. Moore, and R. Steiner. *A practical guide to SysML: the systems modeling language*. Morgan Kaufmann, 2014.

[39] H. G. C. Góngora, M. Ferrogalini, and C. Moreau. How to boost product line engineering with mbse - a case study of a rolling stock product line. In F. Boulanger, D. Krob, G. Morel, and J.-C. Roussel, editors, *Complex Systems Design & Management*, pages 239–256, Cham, 2015. Springer International Publishing.

[40] H. G. C. Góngora, T. Gaudré, and S. Tucci-Piergiovanni. Towards an architectural design framework for automotive systems development. In *Complex Systems Design & Management*, pages 241–258. Springer, 2013.

[41] A. Håkansson. Portal of research methods and methodologies for research projects and degree projects. In *The 2013 World Congress in Computer Science, Computer Engineering, and Applied Computing WORLDCOMP 2013; Las Vegas, Nevada, USA, 22-25 July*, pages 67–73. CSREA Press USA, 2013.

[42] C. Haskins, K. Forsberg, M. Krueger, D. Walden, and D. Hamelin. Systems engineering handbook. In *INCOSE,*, 2006.

[43] J. Hummell and M. Hause. Model-based product line engineering-enabling product families with variants. In *2015 IEEE Aerospace Conference*, pages 1–8. IEEE, 2015.

[44] K. C. Kang, S. G. Cohen, J. A. Hess, W. E. Novak, and A. S. Peterson. Feature-oriented domain analysis (foda) feasibility study. Technical report, Carnegie-Mellon Univ Pittsburgh Pa Software Engineering Inst, 1990.

[45] R. Kolb, I. John, J. Knodel, D. Muthig, U. Haury, and G. Meier. Experiences with product line development of embedded systems at testo ag. In *10th International Software Product Line Conference (SPLC'06)*, pages 10–pp. IEEE, 2006.

[46] A. Kossiakoff, W. N. Sweet, et al. *Systems engineering: Principles and practices*. Wiley Online Library, 2003.

[47] J. Lee, S. Kang, and D. Lee. A survey on software product line testing. In *Proceedings of the 16th International Software Product Line Conference-Volume 1*, pages 31–40, 2012.

[48] M. Lochau, S. Mennicke, H. Baller, and L. Ribbeck. Incremental model checking of delta-oriented software product lines. *Journal of Logical and Algebraic Methods in Programming*, 85(1):245–267, 2016.

[49] M. Lochau, I. Schaefer, J. Kamischke, and S. Lity. Incremental model-based testing of delta-oriented software product lines. In *International Conference on Tests and Proofs*, pages 67–82. Springer, 2012.

[50] R. E. Lopez-Herrejon and A. Egyed. Detecting inconsistencies in multi-view models with variability. In *European Conference on Modelling Foundations and Applications*, pages 217–232. Springer, 2010.

[51] R. Malone, B. Friedland, J. Herrold, and D. Fogarty. Insights from large scale model based systems engineering at boeing. In *INCOSE International Symposium*, volume 26, pages 542–555. Wiley Online Library, 2016.

[52] S. Mann and G. Rock. Dealing with variability in architecture descriptions to support automotive product lines: Specification and analysis methods. In *Proc. embedded world Conference 2009*, pages 3–5. Citeseer, 2009.

[53] A. Marques, F. Ramalho, and W. L. Andrade. Comparing model-based testing with traditional testing strategies: An empirical study. In *2014 IEEE Seventh International Conference on Software Testing, Verification and Validation Workshops*, pages 264–273. IEEE, 2014.

[54] S. Meacham, F. Gioulekas, and K. T. Phalp. Sysml based design for variability enabling the reusability of legacy systems towards the support of diverse standard compliant implementations or standard updates: the case of ieee-802.15. 6 standard for e-health applications. *EAI Endorsed Transactions on Pervasive Health and Technology*, 2(5):e1, 2016.

[55] M. Mendonca, A. Wąsowski, and K. Czarnecki. Sat-based analysis of feature models is easy. In *Proceedings of the 13th International Software Product Line Conference*, pages 231–240, 2009.

[56] A. Metzger and K. Pohl. Software product line engineering and variability management: achievements and challenges. In *Future of Software Engineering Proceedings*, pages 70–84. 2014.

[57] P. Mohagheghi and R. Conradi. Quality, productivity and economic benefits of software reuse: a review of industrial studies. *Empirical SW Eng.*, 12(5):471–516, 2007.

[58] M. Nagamine, T. Nakajima, and N. Kuno. A case study of applying software product line engineering to the air conditioner domain. In *Proceedings of the 20th International Systems and Software Product Line Conference*. ACM, 2016.

[59] O. Oliinyk, K. Petersen, M. Schoelzke, M. Becker, and S. Schneickert. Structuring automotive product lines and feature models: an exploratory study at opel. *Requirements Engineering*, 22(1):105–135, Mar 2017.

[60] S. Oster, A. Wübbeke, G. Engels, and A. Schürr. A survey of model-based software product lines testing., 2011.

[61] S. Oster, I. Zorcic, F. Markert, and M. Lochau. Moso-polite: tool support for pairwise and model-based software product line testing. In *Proceedings of the 5th Workshop on Variability Modeling of Software-Intensive Systems*, pages 79–82, 2011.

[62] J. A. Pereira, K. Constantino, and E. Figueiredo. A systematic literature review of software product line management tools. In *International Conference on Software Reuse*, pages 73–89. Springer, 2015.

[63] K. Pohl, G. Böckle, and F. J. van Der Linden. *Software product line engineering: foundations, principles and techniques*. Springer Science & Business Media, 2005.

[64] M. Pol'la, A. Buccella, and A. Cechich. Automated analysis of variability models: The sevatax process. In *International Conference on Computational Science and Its Applications*, pages 365–381. Springer, 2018.

[65] A. L. Ramos, J. V. Ferreira, and J. Barceló. Model-based systems engineering: An emerging approach for modern systems. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 42(1):101–111, Jan 2012.

[66] A. Reuys, E. Kamsties, K. Pohl, and S. Reis. Model-based system testing of software product families. In *International Conference on Advanced Information Systems Engineering*, pages 519–534. Springer, 2005.

[67] F. Roos-Frantz, J. A. Galindo, D. Benavides, A. R. Cortés, and J. Garcıa-Galán. Automated analysis of diverse variability models with tool support. *Jornadas de Ingenierıa del Software y de Bases de Datos (JISBD 2014), Cádiz. Spain*, page 160, 2014.

[68] P. Runeson and M. Höst. Guidelines for conducting and reporting case study research in software engineering. *Empirical software engineering*, 14(2):131, 2009.

[69] A. R. Santos, R. P. de Oliveira, and E. S. de Almeida. Strategies for consistency checking on software product lines: a mapping study. In *Proceedings of the 19th International Conference on Evaluation and Assessment in Software Engineering*, page 5. ACM, 2015.

[70] I. Schaefer. Variability modelling for model-driven development of software product lines. *VaMoS*, 10:85–92, 2010.

[71] I. Schaefer, L. Bettini, V. Bono, F. Damiani, and N. Tanzarella. Delta-oriented programming of software product lines. In *International Conference on Software Product Lines*, pages 77–91. Springer, 2010.

[72] I. Schaefer, R. Rabiser, D. Clarke, L. Bettini, D. Benavides, G. Botterweck, A. Pathak, S. Trujillo, and K. Villela. Software diversity: state of the art and perspectives. *International Journal on Software Tools for Technology Transfer*, 14(5):477–495, Oct 2012.

[73] I. Schieferdecker. Model-based testing. *IEEE software*, 29(1):14, 2012.

[74] M. Steger, C. Tischer, B. Boss, A. Müller, O. Pertler, W. Stolz, and S. Ferber. Introducing pla at bosch gasoline systems: Experiences and practices. In *International Conference on Software Product Lines*, pages 34–50. Springer, 2004.

[75] J. Suryadevara and S. Tiwari. Adopting mbse in construction equipment industry: An experience report. In *25th Asia-Pacific Software Engineering Conference*, 2018.

[76] C. Tischer, A. Muller, T. Mandl, and R. Krause. Experiences from a large scale software product line merger in the automotive domain. In *2011 15th International Software Product Line Conference*, pages 267–276. IEEE, 2011.

[77] S. Trujillo, D. Batory, and O. Diaz. Feature oriented model driven development: A case study for portlets. In *29th International Conference on Software Engineering (ICSE'07)*, pages 44–53. IEEE, 2007.

[78] F. J. Van der Linden, K. Schmid, and E. Rommes. *Software product lines in action: the best industrial practice in PL engineering*. Springer & Business Media, 2007.

[79] J. Varnell-Sarjeant, A. A. Andrews, J. Lucente, and A. Stefik. Comparing development approaches and reuse strategies: An empirical evaluation of developer views from the aerospace industry. *Info. and SW Tech.*, 61:71 – 92, 2015.

[80] M. Voelter and I. Groher. Product line implementation using aspect-oriented and model-driven software development. In *11th International Software Product Line Conference (SPLC 2007)*, pages 233–242. IEEE, 2007.

[81] A. Vogelsang, T. Amorim, F. Pudlitz, P. Gersing, and J. Philipps. Should I stay or should I go? on forces that drive and prevent MBSE adoption in the embedded systems industry. *CoRR*, abs/1709.00266, 2017.

[82] G. Wang, R. Valerdi, and J. Fortune. Reuse in systems engineering. *IEEE Systems Journal*, 4(3):376–384, Sept 2010.

[83] T. Weilkiens. *Variant modeling with SysML*. Lulu. com, 2012.

[84] S. Wolny et al. Thirteen years of sysml: a systematic mapping study. *Software & Systems Modeling*, May 2019.

[85] B. Young, J. Cheatwood, T. Peterson, R. Flores, and P. Clements. Product line engineering meets model based engineering in the defense and automotive industries. In *Proceedings of the 21st International Systems and Software Product Line Conference-Volume A*, pages 175–179, 2017.

[86] B. Young and P. Clements. Model based engineering and product line engineering: Combining two powerful approaches at raytheon. In *INCOSE International Symposium*, volume 27, pages 518–532. Wiley Online Library, 2017.

[87] B. Zhang, M. Becker, T. Patzke, K. Sierszecki, and J. E. Savolainen. Variability evolution and erosion in industrial product lines: a case study. In *Proceedings of the 17th International Software Product Line Conference*, pages 168–177, 2013.