Product Line Adoption in Industry: An Experience Report from the Railway Domain

Muhammad Abbas
muhammad.abbas@ri.se
RISE Research Institutes of Sweden
Västerås, Sweden

Robbert Jongeling
robbert.jongeling@mdh.se
Mälardalen University
Västerås, Sweden

Claes Lindskog
claes.lindskog@rail.bombardier.com
Bombardier Transportation AB
Västerås, Sweden

Eduard Paul Enoiu
eduard.paul.enoiu@mdh.se
Mälardalen University
Västerås, Sweden

Mehrdad Saadatmand
mehrdad.saadatmand@ri.se
RISE Research Institutes of Sweden
Västerås, Sweden

Daniel Sundmark
daniel.sundmark@mdh.se
Mälardalen University
Västerås, Sweden

ABSTRACT

The software system controlling a train is typically deployed on various hardware architectures and must process various signals across those deployments. The increase of such customization scenarios and the needed adherence of the software to various safety standards in different application domains has led to the adoption of product line engineering within the railway domain. This paper explores the current state-of-practice of software product line development within a team developing industrial embedded software for a train propulsion control system. Evidence is collected using a focus group session with several engineers and through inspection of archival data. We report several benefits and challenges experienced during product line adoption and deployment. Furthermore, we identify and discuss improvement opportunities, focusing mainly on product line evolution and test automation.

CCS CONCEPTS

• **Software and its engineering → Software product lines**.

KEYWORDS

Software product-line engineering, Challenges and opportunities, Overloaded assets

ACM Reference Format:
Muhammad Abbas, Robbert Jongeling, Claes Lindskog, Eduard Paul Enoiu, Mehrdad Saadatmand, and Daniel Sundmark. 2020. Product Line Adoption in Industry: An Experience Report from the Railway Domain. In *24th ACM International Systems and Software Product Line Conference (SPLC '20), October 19–23, 2020, MONTREAL, QC, Canada.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3382025.3414953

1 INTRODUCTION

Software running on train systems has to allow a high degree of customization to address the varying regional standards and certifications, as well as the varying software and hardware needs of different trains. The variance across products is mostly generated by the varying capacity and speed requirements of the vehicles. To meet these customization requirements, Bombardier Transportation (BT) is moving towards the adoption of Software Product Lines (SPLs). SPLs are expected to help the company in delivering (variant-rich) quality products quickly at scale.

BT develops various products for different target seat capacity and speed requirements, as illustrated in Figure 1. At the heart of all these products is a software system controlling the train's engine, the Power Propulsion Control (PPC) software. The PPC software needs to be developed in variants, each addressing the specific hardware configuration, including the motors and controllers of a product. Nevertheless, PPC software variants are sharing many common properties. The PPC team at BT in Sweden successfully exploited these commonalities and adopted a software product line for the development of its propulsion software.

The benefits and costs of product line adoption can vary based on the adoption strategy used. A commonly perceived benefit in the industry is a reduced product development time [27]. The upfront costs depend mainly on the adoption strategy. There are four main approaches for the transition into product line engineering, namely incremental, big bang, tactical, and pilot project [27]. As the name suggests, the incremental transition approach allows the introduction of the product line using lightweight steps. The assets in the incremental strategy are developed over time, usually in parallel with product development. In contrast, the big bang approach requires the domain engineering phase to be completed first to develop a common base for future products. A pilot project approach usually does not consider existing product families for migration. In the PPC team, a lightweight incremental transition strategy was chosen because of the low initial investment needed and because it allows for the development of multiple products in parallel. Similar transition strategies have been used by other companies in the literature [3, 13, 22, 37]. Given that SPLE adoption is an inherently complex organizational change, industrial experiences can provide valuable insights for other organizations making this transition.

Contributions & Results. In this paper, we report the current practices of the Software Product Line Engineering (SPLE) process at BT. The findings in this study are collected using mixed-method

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
SPLC '20, October 19–23, 2020, MONTREAL, QC, Canada
© 2020 Association for Computing Machinery.
ACM ISBN 978-1-4503-7569-6/20/10…$15.00
https://doi.org/10.1145/3382025.3414953

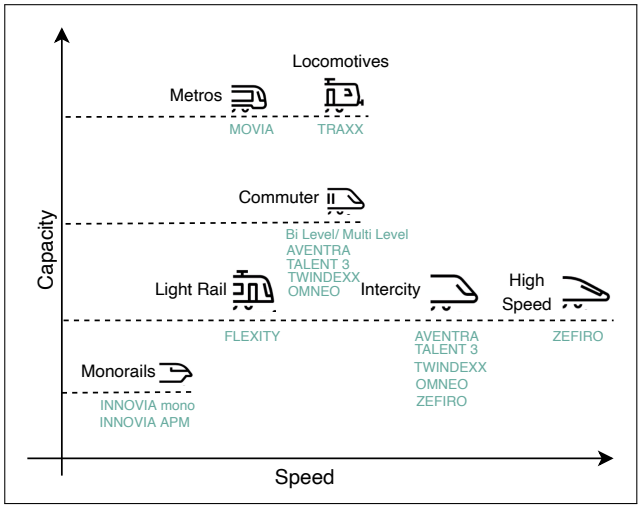

Figure 1: Overview of the product family at BT. Product names are shown in light green, class of train in black.

research. Our findings show that the company was able to significantly reduce the time spent on the development and testing of software components, as well as on safety assessments of products. In addition, engineers report increased confidence in the developed product. Moreover, the results identified a well-defined and standardized way of working and the development of a knowledge-base as the benefits of adopting SPLE practices. Among the most prominently experienced challenges, participants recognized the identification of reuse opportunities, the evolution of the product line and the derived products, tool configuration and migration, and SPLE awareness. As future vision of the company, we identified several themes that need to be considered: the creation of separate domain and application engineering teams, creation of product-wide regression test suites, application of automated test case generation. In addition, the results suggest that there is a need for more research on methods related to test automation, test reuse, automated test repair, and change impact analysis in SPLE.

The remainder of the paper is structured as follows. Section 2 presents a summary of the research method used to obtain the results; this is later presented in detail in Appendix A, which also addresses threats to validity. Our findings are presented in Section 3 and then discussed in Section 4 along with related work on product line adoption. Finally, the paper is concluded in Section 5.

2 RESEARCH METHOD

We used a mixed-method research approach to obtain our results. Document analysis [9] and focus group research [19] are employed to study the SPLE process of a team responsible for developing PPC software at BT. The underlying goal of this research is to improve the current state-of-practice of SPLE in this setting. By applying various research methods and studying different data sources, we aim to obtain reliable industrial insights experienced during the adoption and deployment of SPLE in the railway domain.

We break down the research goal into the following three research questions:

RQ1. *What are the experienced benefits of the adoption and deployment of software product line practices in the railway domain?* This research question aims at collecting raw data about experienced benefits of SPLE at different levels of abstraction, such as at the level of requirements, software components, or tests in one PPC team inside BT.

RQ2. *What are the perceived challenges during and after the adoption of the product line engineering in the railway domain?* We aim at gathering data about the challenges the company has faced during the process of moving towards the product line engineering process. The question also is focused on collecting improvement opportunities in SPLE.

RQ3. *What is the future vision of the company for their SPLE process?* We collect data about the company's future vision for the SPLE process.

Data about the current state-of-practice inside BT (reported in Section 3.1) is collected through document analysis. After collecting the current practices, the data were analyzed by the researchers and subsequently validated by the team's manager. This manager was not part of the performed focus group session. Data about the experienced benefits, perceived challenges, and future vision is collected using the focus group research method. Appendix A explains the details of our research method and how data was collected.

3 RESULTS

We present findings in five parts. First, we present the current practices of the PPC team for SPLE. Then, experienced benefits (see Table 1) and perceived challenges (see Table 2) of the introduction of SPLE are listed (corresponding to RQ1 and RQ2). In addition to the perceived challenges, the fourth part presents improvement opportunities (see Table 3). The last part lists the findings related to planned future work on improving the SPLE process (which corresponds to RQ3, see Table 4). Note that to distinguish them from the main text, direct quotes from interviewees are included in italics throughout the remainder of this paper.

3.1 Current Development Practices

Today's challenge for BT is that every train for any given customer is unique. In practice, this means that both software and hardware vary across different products, since the software has a close dependency on the hardware. In particular, it is crucial to handle the electrical system interface with the software modules such that the software can handle a variety of hardware configurations. Nevertheless, every train software shares a large number of common components and features. The PPC team at BT decided to exploit these commonalities to enhance reuse and reduce the lead time of products. One way to achieve a high degree of reuse is via the adoption of a SPL containing overloaded assets (150% SPLs). The incremental adoption strategy was used to introduce the product line of overloaded assets [27, 30]. In the studied setting, the product line assets are always under evolution, and new functionality is added when needed. Reuse in BT's SPL is realized using a *clone-and-own* way of working, i.e., existing projects are copied and modified to meet customization requirements. This ad-hoc manner of reuse helps companies to avoid high upfront investment in SPL adoption

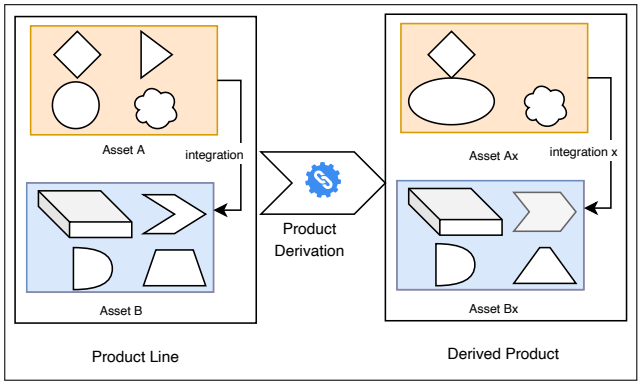

Figure 2: Overview of product derivation process from overloaded assets based SPL

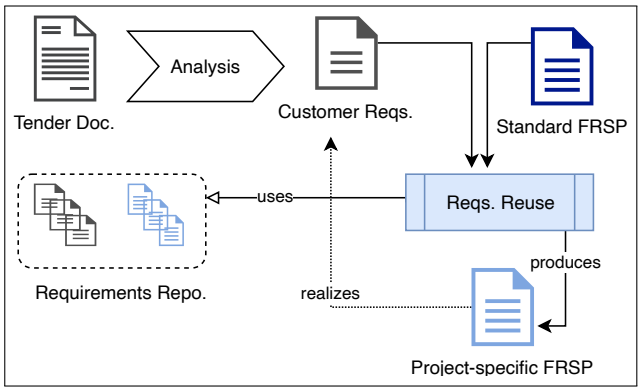

Figure 3: Overview of requirements-level process at BT

and engineering since most of the required assets are already developed in past projects. The same incremental adoption strategy has been found useful in overcoming high upfront investment in other cases too [3].

The product line at PPC is based on clone-and-own based reuse. As shown in Figure 2, the assets are trimmed (trapezoid shape of Asset Bx), in some cases modified or removed (triangle shape in Asset A, Arrow in Asset Bx) in the final derived product. An asset is usually a combination of several software components. Features can be mapped to one or more assets. The integration of the assets may also vary in the derived products (as shown in Figure 2, *integration x* in derived product). The functional variants (modified versions of the assets) of the components are maintained separately for each derived product, meaning that a standard component can have multiple functional variants. Due to the safety-critical nature of the product line, requirements have to be documented and traced down to implementation models and test cases. In the remainder of this section, we present the current way of working at PPC after the introduction of SPLE. We structure the section into three levels describing the process of requirements engineering, development, and testing.

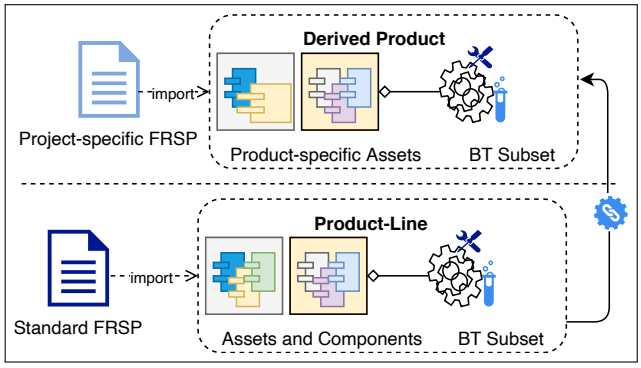

Figure 4: Overview of the product development process at BT

Requirements Engineering. The process of product derivation starts with a set of customer requirements to be realized. The requirements come from a high-level requirements analysis phase from a different team at the company. Over the years, a standard Functional Requirements Specification (standard FRSP in Figure 3) document is created as generic domain requirements. The FRSP describes the overloaded assets and also serves as a features description document. When a new product is to be derived, the standard FRSP is reused/tailored to accommodate the new product requirements. The reuse analysis process also considers existing similar product requirements in the repository. Note that the project-specific FRSF derivation is done manually and might require several iterations.

Development. The content of the standard FRSP is imported in Simulink Requirements to map the requirements into different components and ensure traceability. Note that these traceability links are created manually. The overload assets contain many components. Each component is a Simulink model that can be used to generate C/C++ code for target computers using Embedded Coder. The Simulink components are created by combining building blocks from a library of well-tested commonly-used components (e.g., Mathematical operations) developed by BT (known as BT Subset in the company). Software components are tested, and each is associated with its own test harness. The whole SPL itself is not executable, and thus it undergoes a series of modifications (e.g., removal of components, the addition of functionality to a component) driven by new customer requirements. In practice, the components in the derived products are integrated differently than their integration in the SPL. Every functional variant of the component can be mapped to one Simulink model and can be traced back to its standard component in the product line. The product line itself is maintained and evolved in parallel to the derived products. Whenever a new product is derived, a SPL baseline is created for that product. Some changes to the SPL, e.g. bug-fixes, may also be propagated to the derived product(s).

Testing. Testing is performed in three different phases and at four different levels (shown in Figure 5). In order to achieve a Safety

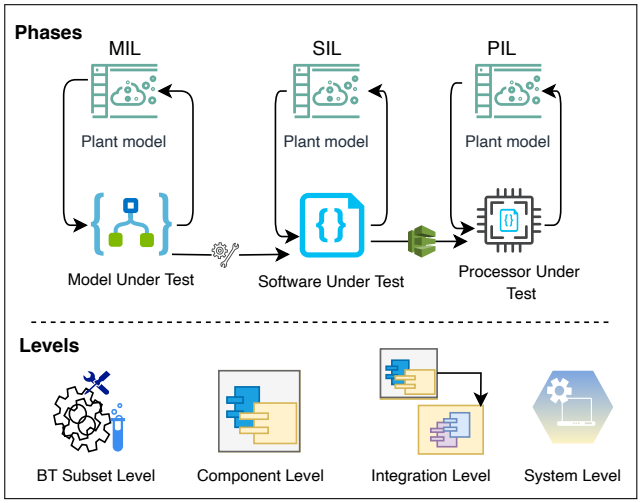

Figure 5: Overview of the testing process at BT

Integrity Level (SIL) 2 compliance for the propulsion software, testing on some test levels has to conform to the EN50128 standard[1] for development of rail control software. Note that the restrictions placed on the test process only apply to some levels and not all. Specifically, the standard applies to hardware-software integration, software integration, and components.

The developed artifacts for the derived products (mostly Simulink models) are tested in-house in three different phases. If an artifact has any hardware/environment dependencies, a plant model is created to simulate the required behavior of the hardware/environment. The artifact is the first tested in a Model-In-the-Loop (MIL) environment inside Simulink. Embedded Coder is then used to generate C/C++ code from the Simulink model. The generated code is then tested in a Software-In-the-Loop (SIL) environment. For safety-critical components, the generated code is also subjected to code reviews. The reviewed code is deployed on the target processor for a Processor-In-the-Loop (PIL) testing.

These phases are performed on four different levels. First (unit level), the subset is tested, and the source code is reviewed. Then the components (created from combining the sub-set elements) are subjected to all phases of the testing. Thirdly, the components are integrated in a way that serves a system function. The integrated components are tested in all three phases for integration errors. Lastly, the propulsion system is tested in all three phases at the system level. In addition, a final Hardware-In-the-Loop (HIL) testing is performed off-site. Note that the SPL itself is only tested at the BT subset level and partially at the component level. Components that are integration dependent are only tested in the derived products.

3.2 Experienced Benefits

Saving time and resources. Since the introduction of SPLE, the lead time of software changes has decreased significantly: *"The time needed for implementing new functionality is about a month, whereas it was estimated to be about six months to a year."* This decrease can be attributed to shorter development cycles as introduced in SPLE, and a smaller scope of the products as compared to the previous organization with development divided into projects with long release cycles: *"The product development was taking around six months to a year to complete. It used to require many changes, and managing those changes within the project time-frame was difficult"*. Although these time gains are rough estimates and based on data from one development team, they do indicate the suitability of SPLE for development in short cycles, which is popularized by the publishing of the manifesto for agile software development [5].

After the introduction of the product line, the testing process has been shortened. In testing components, the majority of the effort resides in writing test cases. Hence, the reuse of test harnesses and test cases was an important factor in increasing the productivity of the development team. The highest productivity gain is obtained in cases where all tests could be reused, and therefore only need to be re-run for a derived product but not altered. In cases where the test cases do need to be altered, having an existing test harness and test cases also improves the development productivity by reducing the effort of designing test cases from scratch. Indeed, one domain expert estimated the related time effort to be at least 50% more efficient: *"The time it takes to complete or create a new test case was about a day. Now, you do a modification and retest, and you are going from a standard day or eight hours to anywhere between one to four hours"*.

Another time-saving aspect is related to the required safety assessments. The reuse of products implies that a large portion of the safety assessment can also be reused because it is performed on the product level, and the changes to derive the product are expected to address only small portions of that product. *"The idea is that if you have a product already safety assessed, you can then do the next safety assessment a lot easier since the safety assessment should focus on the changed part of the product."* This is, in particular, important in the studied setting because of the required certification for all developed software.

Confidence boosting. Some other experienced benefits are not directly related to savings of time or resources. The common denominator of these benefits is the increased confidence in the developed products. Thereby, they indirectly improve the productivity of the development team.

As discussed earlier, deriving the products from the SPL means that these are based on software that is already tested and safety-assessed. In addition, the way of creating such a derived product is highly standardized with the introduction of SPLE, which increases the confidence of developers in the resulting artefacts: *"What also helps is the fact that in the SPL, we are following the entire development process as defined by the standard team. So we know how certain things should be done, especially testing, for example, software-in-the-loop testing, how to setup everything for processor-in-the-loop testing, etc. Once you then go to the derived product, all you have to do is to follow the development process."* The shaping of this way-of-working hence seems to be an important by-product of the introduction of SPLE.

Also, the current way of organizing reuse inherently means that a knowledge-base is created that is built upon project after project: *"you can develop the product so that the next product has all the benefits*

[1] EN 50128 - Railway applications - Communication, signalling and processing systems - Software for railway control and protection systems.

Table 1: Identified themes and sub-themes corresponding to experienced benefits (RQ1).

	Theme		Sub-theme
1	Saving development time and resources	1.1	Reduced development time
		1.2	Less time spent on writing test cases.
		1.3	Easier safety assessment when performed incrementally.
2	Confidence boosting	2.1	Increased confidence in software for derived products
		2.2	Well-defined "way-of-working"
		2.3	Inherent knowledge-base built through reuse.

from the lessons learned of the past derived products". This works since all products are now derived from a shared common SPL that is developed separately from the products. Whereas, in the former project-based setting, every new project was only based on its direct predecessor. Furthermore, the ongoing product development can benefit from knowledge built during the development of any other product, since in some occasions, new functionality or a patch introduced in a product is propagated back from derived products into the SPL: *"We can have multiple product derivations running in parallel. All their feedback is then pushed into the SPL. The past way of working where you are just taking the latest project doesn't allow this"*.

3.3 Perceived Challenges

Deriving products. As we discussed in the previous subsection, the experienced benefits are mostly related to the way the product derivation is organized. Although the process is clearly delivering benefits to the company, some challenges remain.

Reuse begins by identifying opportunities for reuse between the SPL and the products. The problem faced in identifying these opportunities was a high number of new requirements coming in, and at the same time, a large existing set of artifacts spread out over many derived products. Some of these artifacts could, unbeknown to the engineer, already be addressing the new incoming requirements: *"The problem that we faced was that many new requirements were coming in. We didn't know if we had handled this requirement before, if this was something we already had, or if it has a small difference from what we already had. Because the amount of requirements is so large, there are so many past derived products; there are so many different designers and architects. Nobody could know all the requirements that are already being handled by control software"*.

After artifacts are identified and reused, they are not final. Instead, they keep being subject to evolution, for example, due to changed requirements or bug fixes. A challenging aspect currently under research in a different study within the development team is how to propagate changes made in the SPL (after a product has been branched off) to the products in which those (parts of) software components or test cases are reused. For each product, it needs to be decided if the changes are desired to be propagated or not. In some cases, it might also occur that a change in a derived product is later "pushed back" to the SPL. In conclusion, since the changes are made to both the SPL and the derived products, continuous maintenance is required.

A product consists, among other artifacts, of a set of software components, tests, and component integration. In general, it is challenging to assess the impact of changes in the SPL on the derived products, as outlined by one participant of the focus group: *"One issue we had is that the tester has to have a quick and easy way of identifying what changes were made, where they were made. And if there were any changes, have a quick confirmation that there was no functional change, and everything is the same as it was before or behaving the same as it was before. (...) If this is all manual work, then that builds extra time and overhead"*. This quote specifically has test case development in mind, but for changes in the component integration, impact analysis is experienced as even more challenging: *"The issue is when a derived product makes changes to the component integration, it is hard to identify the impact of those changes on other parts. (...) In such cases, the tester has to identify these changes and know where they were made. The tester also has to find the impact of such changes on functionality and behavior"*. Also, for both scenarios, automated means could help the developers and testers to be more efficient. The literature contains proposals for automated impact analysis techniques, but these can be challenging to adopt in particular industrial settings. One of the aspects of this challenge lies in the specific tools used in a particular setting. We now discuss such tool-related challenges.

Automation. In general, setting up development tooling for a SPL is difficult. Each company and maybe even each project has its own specific needs. Therefore, tool vendors provide highly configurable tools that can be used in many situations. Within the team, Simulink and Embedded Coder are used for modeling systems and generating code from these models. One of the encountered challenges in using this tooling appropriately was its configuration, and the team was sometimes not helped by the documentation, on the contrary: *"the information that we found online was either wrong or misleading"*. Hence, in some cases, some more thorough investigations than a quick search through the documentation were required to find out how exactly to use the tool: *"If you look online for example, this not the sort of information that you are going to find."*

Another aspect of this tool flexibility is that multiple developers can opt for different alternative ways of performing similar tasks. This is planned to be resolved by separating domain and application engineering, as explained by one of the interviewees: *"I think it's good to be very clear and concise about what we are doing, what we are aiming for, what you can do in the tool, what you cannot do in the tool. For the people using it for a particular purpose, if you are a developer and looking at improving the process and improving the product line, improving upon how we use the tool, then you have all this flexibility and customizations. So, I think there should be a clear distinction between the people doing that improvement work and people who just apply the way-of-working with this product"*.

Table 2: Identified themes and sub-themes corresponding to Perceived Challenges (RQ2).

	Theme		Sub-theme
3	Product Derivation	3.1	Identification of reuse opportunities
		3.2	Propagating changes form the SPL to the derived products
		3.3	Impact analysis of behavioral changes on test cases
4	Automation	4.1	Domain-specific configuration of tooling
		4.2	Incomplete migration to central tool
5	SPLE Awareness	5.1	Shift from projects to product-line thinking not propagated throughout organization
		5.2	Software architecture choices negatively impact re-usability opportunities

Table 3: Identified themes and sub-theme for Improvement Opportunities (RQ3)

	Theme		Sub-theme
8	Software Testing	8.1	Methods for automated software verification (e.g., model checking).
		8.2	Test case reuse between MIL, SIL and PIL.
		8.3	Automated test case repair.
9	Artifacts Evolution	9.1	Automated change impact analysis on test cases.
		9.2	Traceability link recovery between requirements and model elements.
10	SPLE Tooling	10.1	Migration of development actions from auxiliary tools to Matlab environment
		10.2	Architecture formalization and use CI pipeline to check consistency between it and the implementation.

This is also noted as one of the improvements to the development process in the future, as shown in Table 4.

SPLE has been introduced five years ago within the team, but the process of transitioning to the development tools is ongoing. As one interviewee noted, this is a much needed next step towards the full introduction of the product line and specifically the team's main development tool Simulink. Although the current development of software components and test cases is performed in Simulink, several other steps in the process are performed in different ways: *"I would like us to integrate more steps into Simulink. We have software components, assembly, and test cases in Simulink (but also) we are using a lot of Excel sheets."*

SPLE Awareness. In addition to these more technical challenges, some organizational challenges were identified. One of them is the challenge of shifting the way of thinking throughout the organization from project-based development to product lines. A lack of this is experienced by one of the interviewees: *"nobody really had a clear definition of what a product is"*. The consequence of the ignorance of SPLE concepts is that the interviewees notice challenges to motivating the required development steps and associated costs. Furthermore, the interviewee notes that the company is not yet reaping the full benefits of reuse for these reasons: *"it helps a lot if you deliver a lot of products because then you see more potential reuse. There are work steps or processes that are rapidly reoccurring. If you can shorten the steps, there is motivation to do so. But if you have long-running product derivation cycles, where you only start sort of maybe a few per year, then these concepts come into difficulty. It's difficult to measure, and it's difficult to motivate why you should do it."*

Some architectural decisions have, in hindsight, been preventing optimal reuse: *"at the moment we are only sending over components. It's more difficult to send over high-level integrations. And that's based on architectural decisions that should be reviewed"*. This could be a consequence of the initial unawareness of SPLE and the continuing move towards it from a project-based way-of-working.

3.4 Additional Improvement Opportunities

In addition to the perceived challenges, we present some opportunities for improvement of the encountered state of practice in the studied setting.

Product Testing. Writing test cases consumes resources and requires domain-level expertise. One participant in the focus group suggested that the company is looking for automated approaches for testing and verification, such as model checking (Sub-theme 8.1 in Table 3). Besides, it seems that the company can benefit from test reuse between the different testing phases such as MIL, SIL, and PIL (Sub-theme 8.2 in Table 3). One of the participants suggested that: *"There is a huge potential in reusing test cases of MIL in SIL and PIL testing."* Methods supporting reuse between MIL, SIL, and PIL would be of high interest for BT. We also noted that in many cases, a small modification to the components results in test case breakages. To fix these, currently, test cases are manually reviewed and updated. In this case, there is a need for approaches for automated test case classification and repair (Sub-theme 8.3 in Table 3).

Evolution of the product line and products. The evolution of the product line and derived products brings several challenges. Particularly, there is manual effort involved in assessing whether or not to co-evolve derived products upon changes in the product line. We found that if a decision is made in favor of such co-evolution, it is very hard to know the impact of possible changes on the resulting test cases and the behavior of the derived product. Similar lessons learned have also been reported in the literature (e.g., in [36]).

There is a need for approaches and methods for variability-aware change impact analysis (Sub-theme 9.1 in Table 3). In addition, compliance with safety standards requires the demonstration of

Table 4: Identified themes and sub-themes corresponding to Future Vision (RQ3).

	Theme		Sub-theme
6	Test automation	6.1	Automated test generation
		6.2	Creation of product-wide regression suite
7	Development process	7.1	Adoption of CI/CD
		7.2	Separation of domain and application engineering

traceability from the customer requirements to models and test cases. Currently, the requirements are imported into Simulink and are linked manually with the model elements satisfying them. We see an improvement potential and a need for automated traceability link recovery between natural language requirements and model elements (Sub-theme 9.2 in Table 3).

Tooling. The supporting tools for product development and management should allow for a very high degree of flexibility. Consequently, obtaining the right development setup is not straightforward. Indeed, configuring a tool for company-specific needs is challenging.

As mentioned earlier, some development artifacts and steps are organized in auxiliary tools such as Microsoft Excel. It is desired to include those steps in the Simulink environment to make the way-of-working less dependent on error-prone steps in which data from one tool is used as an input for another one.

Currently, there is no automated support for handling the variability in the team. There have been talks with the pure-systems'[2] team on the use of pure-variant at BT. However, it is desired to utilize the Simulink environment with their Variant Manager.

In addition, the product architecture is described through informal diagrams, e.g., Powerpoints or Visio diagrams. Since the architecture does not evolve much, due to new incoming requirements being similar to some extent, there is no current need for diagrams or formal models in formats that are more maintainable. However, upon the creation of more different products and shorter development cycles, some more formal architecture models might be useful, particularly to allow automated checking of the consistency between Simulink models and the intended architecture. It is an interesting research opportunity to formalize the architectural models and check, for example, within the CI pipeline, whether the Simulink models conform to the intended architecture or violate it.

3.5 Future Vision

At the end of the focus group, we discussed some future plans of the team for improving the way-of-working in product line engineering. These are additional to the plans that have been discussed in previous subsections.

One of the areas of interest is test automation. A pilot study has shown interesting results of applying automated test generation: *"surprisingly, some interesting issues have popped up by just applying automatic test case generation at the component-level. (...) For example, some components have dead code, and some input values for test cases were not optimal in the sense that they would allow for an output that was outside the set boundaries"*. Since a test generation plugin is included in the Simulink tool suite, through its *Design Verifier*, the team plans to explore its deployment in the near future. Another future vision for testing is to create a complete test suite for the product at the integration level. Although it is currently not certain if that will be established, one participant states: *"I hope that we would standardize the platform, that we would also come with the complete overall test suite"*. A clear benefit of this standardization would enhance the reuse opportunities at the integration and system-level testing of PPC software.

Other topics of interest mentioned in the focus group include improving software engineering practices and establishing a continuous integration pipeline, which could allow for optimizations in the development process. It is noteworthy that more involved practices such as continuous deployment and delivery are of less interest in the railway domain, due to the need for certification and safety assessment of the products. As discussed earlier, an important part of improving software engineering practices is the separation of domain and application engineering.

4 DISCUSSION

In this section, we discuss our results, provide an analysis of our observations, and provide an overview of the related work. The product development lead time in our case is reduced significantly after the product line adoption. Mainly, our results suggest that the adoption of product line practices reduced the lead time by lowering the product development time, test case creation time, and time required for activities related to safety standard compliance. All this was achieved by a high degree of reuse of the core assets. Moreover, the reduction in lead time was made possible by introducing a standardized and well-documented way-of-working for each team. This was originally a by-product of SPLE adoption but turned out to be highly beneficial. Similar experiences of reduced development time are also reported in the literature (e.g., [31, 35]).

We also observed that an evolutionary and incremental adoption strategy leads to successful adoption in small teams due to its low up-front cost. However, this evolutionary way of working can result in a product-line growth issue for product derivation. We found that, when the number of derived products grows (the number of modified assets will also grow), engineers find it hard to know if a new customer requirement is already satisfied by a modified component in an existing derived product. Similar findings have also been reported in the literature regarding asset accessibility and management [32]. Currently, a tool for requirements-level reuse analysis and recommendation is under development [1], and evaluation. In addition, the requirements-level configuration (deriving the project-specific FRSP in Figure 3) is a resource-intensive task. The company is looking for automating the process with tools such as Zen-ReqConfig [21].

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

4.1 Related Work

Over the years, software reuse through SPLs has been the focus of many researchers and companies. The expected benefit of the adoption of a product line is reduced efforts in the development of quality products [27]. However, the shift towards a software product line faces challenges. Various product line adoption strategies and experience reports can be found in the literature [2, 15]. We provide an overview of the related work on product line adoption, and link reported findings to our own findings as presented in Section 3.

Product Line Adoption in Small and Medium Companies. Bastos *et al.* used a mixed research method to report the product line adoption in small companies [3]. Based on systematic mapping, an industrial case study, and expert opinion, the authors reported 22 findings. Some similar findings to ours revealed that the adoption road-map is necessary, and the incremental introduction of SPLE is more appropriate in small companies or teams. Nazar and Rakotomahefa conducted an ethnographic study to identify the challenges of product line adoption in a small Chinese company [25]. The study's suggestions have some overlap with our outlined future vision, which includes having a separate domain and application engineering teams. Njima and Demeyer interviewed four participants from two start-ups to identify the motivational factors and challenges of product line adoption [26]. Results from the interviews show that start-ups see the reduced development time and cost of quality products as a benefit. The study also reported challenges that start-ups might encounter while adopting software product lines. Knauber *et al.* reported their experiences in applying the product line engineering method in small companies [17]. Their experience shows that there was a lack of documentation in the companies, and a clear vision about the product evolution is essential to successful adoption. Verlage and Kiesgen reported experiences of the transition of a small software company toward software product lines [36]. The study summarizes the current product line, the changes made to the process for product line adoption, and lessons learned. Initially, no separate application and domain engineering teams were formed, as in our case. Other overlaps in findings show that analyzing the impact of changes can be challenging.

Opinions, Guidelines and Experiences of Product Line Adoption in Large Enterprises. Catal *et al.* presented common barriers in adoption of product line engineering [12]. The study also categorizes ten findings in three categories based on three views (Sponsor and customer view, company or development group view, and SPLE community view). The sponsor and customer view include overlapping challenges on a lack of SPLE knowledge. The development view includes challenges about the practitioners not having enough knowledge regarding SPLE, issues in the integration of hardware/software, high up-front cost, organizational structure change, and training. Finally, the community view includes an overlapping finding on the lack of practical resources on SPLE, specifically testing and high training cost. The authors also provided suggestions on how to avoid these barriers. Bilic *et al.* used overloaded models in a model-driven product line in the automotive domain [6, 7]. The outlined challenges include creating and maintaining a legacy variant, lack of traceability, product configuration, and no test optimization. The authors also reported lessons learned and management, and the evolution of assets was suggested to be done in an incremental manner. The assets evolution is a similar overlapping finding to our work. Similar experience reports also reporting reduced product development time can be found in other domains as well [31, 35]. Kuvaja *et al.* presented challenges and guidelines related to product line adoption based on interviews with ten practitioners [20]. The related identified finding includes requirements' data not utilized fully, lack of training, and lack of adoption plans. Lack of education and training is also reported as a barrier to software reuse [32]. Sherif and Vinze's findings also show some overlap with our findings regarding asset management and asset accessibility (similar to our sub-theme 3.1 in Table 2) [32]. Jha *et al.* surveyed 29 practitioners to find issues in software reuse in context software product lines [16]. Some overlapping findings indicate a high up-front and maintenance cost as an issue, need for reuse planning, confidence boost as a benefit, and knowledge-base development as a benefit. Böckle *et al.* presented guidelines on the transition towards product line engineering process [8]. The work reports and discusses various adoption strategies and emphasizes on preparing a business case and an adoption plan. Moreover, the authors recommend transforming the organizational culture to product line engineering. Our findings also show that there is a need for product-oriented thinking in the company. Staples and Hill reported their experience of adopting product line development via variation provided in configuration management [34]. This allows the assets reuse developed with no product line architecture in mind. The approach improved product development time and quality by allowing the propagation of fixes across the products. Similar fast feedback cycles (to the standard product line, in our case) has been noted in our findings. Mohagheghi and Conradi discussed the adoption of product families at Ericsson [23]. The adoption was introduced incrementally, and requirements management for the components was seen as a challenge. The adoption resulted in reduced development time and cost. Bauer reported the challenges of structured reuse adoption in the context of a big undisclosed company [4]. The paper presented two failed adoptions of reuse and reported the barriers that made the adoption unsuccessful. The authors also outlined lessons learned during different cases. Dordowsky and Hipp reported the adoption of software product line principles at an avionics company [14]. The study presents the company's process and also reported experiences. Related findings show that incremental adoption was used to avoid failure. The finding also revealed that product development time is reduced to a quarter of the development time with no product line.

There is some overlap between our findings and those presented in the aforementioned literature. Commonly reported benefits include SPLE awareness and reduced lead time. A common experienced challenge is change impact analysis. In addition to the literature, we report challenges to identifying reuse opportunities at the requirements level, in particular certain product line evolution scenarios and migration to specific SPLE tooling. Moreover, we have identified new research opportunities in the domain, as outlined in Section 4.

5 CONCLUSIONS

We report the state-of-practice of SPLE at the PPC team of BT. Apart from the experienced benefits, we also identified several current challenges and research opportunities related to the SPLE process in one team inside a large enterprise in the railway domain.

Our study of RQ1 has yielded the following perceived benefits: reduced lead time for product development, testing, and safety assessment, increased confidence in the product. We also conclude that successful adoption of SPLE was made possible by following a lightweight evolutionary and incremental transition strategy.

In relation to RQ2, we identified that analyzing the impact of changes in the evolving product line and derived products continues to be challenging in industrial practice. Moreover, identifying reuse opportunities remains challenging with the growth of software development complexity, including an increasing number of products addressing an increasing number of requirements. Engineers experienced that tools used for the development and management of the product line are hard to configure, and their versatility may be counterproductive if not adequately controlled. Finally, our findings underscore the importance of product-line-oriented thinking throughout the entire organization.

As a general finding with respect to RQ3, our results suggest that researchers and tool vendors should develop novel automated methods to support product line engineering and management, specifically addressing the challenging topics of reuse recommendation, test reuse, variability-aware testing and verification, change impact analysis, and traceability link recovery. More specific to the studied setting, the future vision of the PPC teams is to establish a product-wide regression suite, introduce a continuous integration pipeline, and apply automated test generation via model checking. The company is also planning to use separate domain and application engineering teams. In addition, the team has on-going initiatives in collaboration with other companies and universities on change impact analysis, variability testing, and reuse recommendation.

Based on the identified challenges and research opportunities, there is a need to improve the variability-aware change impact analysis and recommendation methods and incorporate these as part of a product line engineering process.

A FOCUS GROUP PROTOCOL

The focus group session was conducted to gather data about the experienced benefits, perceived challenges, and the future vision (corresponding to each research question). Focus group research is a well-established practice in empirical software engineering research used to gather practitioners' experiences and opinions about new tools and methods [18, 19, 24, 33]. In planning and executing the focus group session, we followed the guidelines presented by Breen [11]. The recorded audio was afterwards transcribed and then analyzed following the thematic analysis guidelines proposed by Braun and Clarke [10]. The process followed in conducting the focus group session is presented in Section A, also shown in Figure 6.

A.1 Focus Group Planning

Instrument. A study plan to manage the focus group session was created by three authors of this paper. The plan contained a focus group instrument with three topical research questions and five sub-questions. The plan was reviewed in an online workshop by the authors, and the topical questions were chronologically ordered. As a substitute for a pilot, the instrument was validated by one practitioner affiliated with the company and thereafter revised based on received comments. The focus group session was conducted with three members of the PPC team who are experts in different domains of SPLE. The final focus group instrument covered five open-ended sub-questions related to three pre-defined themes corresponding to each of our research questions.

Confidentiality and Ethical Concerns. At the start of the focus group, consent was obtained from all participants for taking audio recordings and notes. The recordings were transcribed and subsequently deleted. Within the transcript, the practitioners have been made anonymous, but names of tools and development practices have not, since they are relevant to the presented findings.

A.2 Session and Transcription

Participants. Three members of the PPC team were selected to participate in the focus group session. The selection ensures a diversity of age, gender, roles, and experience. Roles vary between requirements engineering, testing, and software design. The participants have two, seven, and thirty years of experience in the control software development and testing.

Session. The focus group session was conducted in an online meeting using Skype for Business. The session was moderated by one of the authors, and two authors were mainly tasked with note-taking but also with asking for clarifications when needed. We started the session with an introduction on the pre-defined goal of this study. We record one hour and ten minutes of audio material. The total length of the session was one hour and thirty minutes, since we started the recording only after a short introduction and a discussion about obtaining consent for this recording.

Transcription. The audio recording of the session was transcribed in a text document containing around seven thousand words. The transcription process included also the anonymization of confidential and personal information (such as names of the participants and names of confidential projects). The anonymized terms were replaced with labels in brackets (e.g., a confidential project name was replaced with {Project followed by a letter}). The anonymized transcript was subjected to a thematic analysis.

A.3 Thematic Analysis

Qualitative data analysis can be performed in many different ways. We perform a thematic analysis using the guidelines presented by Breen [11] since these are commonly followed for qualitative research studies in different research communities. Terminologies used in the remainder of this section are first defined here:
Theme is an abstraction of a pattern within a data-set. In our case, we pre-defined a set of three themes from our research questions (Experienced Benefits, Perceived Challenges, and Future Vision).

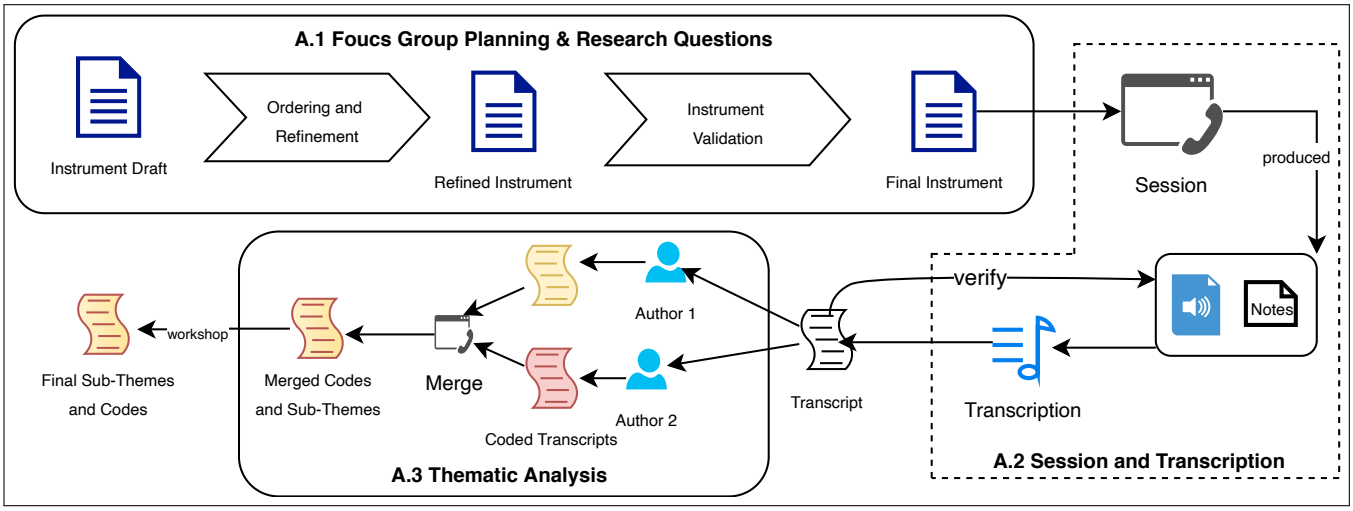

Figure 6: An overview of the research process followed to gather data from the focus group session.

Pre-defined themes are common in qualitative research and are referred to as a priori themes.
Sub-Theme is a theme within a theme.
Codes are labels assigned to data (such as sentences) to index them.

The thematic analysis was performed by two authors to encourage the diversity of the sub-themes and codes (as shown in Figure 6, 5.3). Each author performed coding of the transcript and grouped the codes to general sub-themes representing an abstract concept. The sub-themes were grouped into the three pre-defined main themes derived from the research questions. Codes that did not fit into the pre-defined themes were considered for identifying research opportunities (reported in Section 4). This process of thematic analysis and coding was repeated one more time to refine the codes and sub-themes and identify sub-themes and codes that we might have missed during the process. The codes were merged and ranked to drop codes that might not be relevant to the focus of this study. The final sub-themes and codes were validated and verified by two other authors of this paper. The resultant document was shared with the focus group participants to allow them to correct any possible misunderstandings.

A.4 Validity Threats

We address the validity threats of our results following the classification and guidelines proposed by Runeson and Höst [28].

Construct Validity. Construct validity reflects the extent to which the studied operational measures are representative of what the researchers intend to study. To tackle potential threats to construct validity of our results, we designed our instrument using terminologies known by the participants. The instrument was subjected to thorough iterations, refinement and was validated by both researchers and an engineer in the team who did not take part in the focus group session.

Internal Validity. Internal validity threats affect the validity and credibility of our results. Potential internal validity threats were tackled by using multiple data sources (we obtain our findings through document analysis and participant observation) and by asking engineers to validate our take-aways from these sources.

External Validity. External validity reflects the extent to which the results can be generalized. The presented findings in this study are obtained from studying the case of one team in a company, we therefore do not claim any findings generalized beyond the domain of the team. Furthermore, the reported findings come from the development of safety-critical software in a real industrial context and can, therefore, be valuable to the community as well as other practitioners in a similar domain.

Reliability. This aspect reflects the extent to which the data and findings are independent of the researchers that performed the study. We address this aspect of the validity threats by following well-established qualitative research methods and guidelines. In addition, we involved industrial professionals and multiple researchers to validate the research process and the obtained findings.

ACKNOWLEDGMENTS

This work has been supported by and received funding from the ITEA3 European XIVT [29][3], and ARRAY[4] projects.

REFERENCES

[1] Muhammad Abbas. 2020. Variability Aware Requirements Reuse Analysis. In *The 42nd International Conference on Software Engineering Companion (ICSE '20 Companion)*. ACM. http://www.es.mdh.se/publications/5734-
[2] Jonatas Ferreira Bastos, Paulo Anselmo da Mota Silveira Neto, Eduardo Santana de Almeida, and Silvio Romero de Lemos Meira. 2011. Adopting software product lines: a systematic mapping study. In *15th Annual Conference on Evaluation & Assessment in Software Engineering (EASE 2011)*. IET, 11–20.
[3] Jonatas Ferreira Bastos, Paulo Anselmo da Mota Silveira Neto, Pádraig O'Leary, Eduardo Santana de Almeida, and Silvio Romero de Lemos Meira. 2017. Software product lines adoption in small organizations. *Journal of Systems and Software* 131 (2017), 112–128.

[3] https://itea3.org/project/xivt.html
[4] https://www.es.mdh.se/projects/497-ARRAY

[4] Veronika Bauer. 2015. Challenges of structured reuse adoption—Lessons learned. In *International Conference on Product-Focused Software Process Improvement*. Springer, 24–39.

[5] Kent Beck, Mike Beedle, Arie Van Bennekum, Alistair Cockburn, Ward Cunningham, Martin Fowler, James Grenning, Jim Highsmith, Andrew Hunt, Ron Jeffries, et al. 2001. *Manifesto for Agile Software Development*. http://agilemanifesto.org

[6] Damir Bilic, Daniel Sundmark, Wasif Afzal, Peter Wallin, Adnan Causevic, and Christoffer Amlinger. 2018. Model-Based Product Line Engineering in an Industrial Automotive Context: An Exploratory Case Study. In *International Systems and Software Product Line Conference*. ACM, 56–63. https://doi.org/10.1145/3236405.3237200

[7] Damir Bilic, Daniel Sundmark, Wasif Afzal, Peter Wallin, Adnan Causevic, Christoffer Amlinger, and Dani Barkah. 2020. 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*. 1–11.

[8] Günter Böckle, Jesús Bermejo Muñoz, Peter Knauber, Charles W Krueger, Julio Cesar Sampaio do Prado Leite, Frank van der Linden, Linda Northrop, Michael Stark, and David M Weiss. 2002. Adopting and institutionalizing a product line culture. In *International Conference on Software Product Lines*. Springer, 49–59.

[9] Glenn A Bowen et al. 2009. Document analysis as a qualitative research method. *Qualitative research journal* 9, 2 (2009), 27.

[10] Virginia Braun and Victoria Clarke. 2006. Using thematic analysis in psychology. *Qualitative Research in Psychology* 3, 2 (2006), 77–101. https://doi.org/10.1191/1478088706qp063oa

[11] Rosanna L Breen. 2006. A practical guide to focus-group research. *Journal of Geography in Higher Education* 30, 3 (2006), 463–475.

[12] Cagatay Catal. 2009. Barriers to the adoption of software product line engineering. *ACM SIGSOFT Software Engineering Notes* 34, 6 (2009), 1–4.

[13] Krzysztof Czarnecki and Michał Antkiewicz. 2005. Mapping Features to Models: A Template Approach Based on Superimposed Variants. In *Generative Programming and Component Engineering*. Springer Berlin Heidelberg, 422–437.

[14] Frank Dordowsky and Walter Hipp. 2009. Adopting Software Product Line Principles to Manage Software Variants in a Complex Avionics System. In *Proceedings of the 13th International Software Product Line Conference (SPLC '09)*. Carnegie Mellon University, USA, 265–274.

[15] M. A. Jabar, M. B. Zarei, F. Sidi, and N. F.M. Sani. 2013. A review of software product line adoption. *Journal of Theoretical and Applied Information Technology* 57, 1 (2013), 88–94.

[16] Meena Jha and Liam O'Brien. 2009. Identifying Issues and Concerns in Software Reuse in Software Product Lines. In *Proceedings of the 11th International Conference on Software Reuse: Formal Foundations of Reuse and Domain Engineering (ICSR '09)*. Springer-Verlag, Berlin, Heidelberg, 181–190. https://doi.org/10.1007/978-3-642-04211-9_18

[17] P. Knauber, D. Muthig, K. Schmid, and T. Widen. 2000. Applying Product Line Concepts in Small and Medium-Sized Companies. *IEEE Software* 17, 5 (2000), 88–95.

[18] Jyrki Kontio, Johanna Bragge, and Laura Lehtola. 2008. The focus group method as an empirical tool in software engineering. In *Guide to advanced empirical software engineering*. Springer, 93–116.

[19] Jyrki Kontio, Laura Lehtola, and Johanna Bragge. 2004. Using the focus group method in software engineering: obtaining practitioner and user experiences. In *Proceedings. 2004 International Symposium on Empirical Software Engineering, 2004. ISESE'04.* IEEE, 271–280.

[20] Pasi Kuvaja, Jouni Similä, and Hanna Hanhela. 2008. Software product line adoption–guidelines from a case study. In *IFIP Central and East European Conference on Software Engineering Techniques*. Springer, 143–157.

[21] Yan Li, Tao Yue, Shaukat Ali, and Li Zhang. 2019. Enabling automated requirements reuse and configuration. *Software and Systems Modeling* 18, 3 (2019), 2177–2211. https://doi.org/10.1007/s10270-017-0641-6

[22] N. Matsuda. 2004. Problems and suggestions for adopting product line software engineering from modification style development. In *11th Asia-Pacific Software Engineering Conference*. 568–571.

[23] Parastoo Mohagheghi and Reidar Conradi. 2003. Different aspects of product family adoption. In *International Workshop on Software Product-Family Engineering*. Springer, 429–434.

[24] Jefferson Seide Molléri, Michael Felderer, Emilia Mendes, and Kai Petersen. 2019. Reasoning about Research Quality Alignment in Software Engineering. *Journal of Systems and Software* (2019).

[25] Najam Nazar and TMJ Rakotomahefa. 2016. Analysis of a Small Company for Software Product Line Adoption–An Industrial Case Study. *International Journal of Computer Theory and Engineering* 8, 4 (2016), 313.

[26] Mercy Njima and Serge Demeyer. 2019. An Exploratory Study on Migrating Single-Products towards Product Lines in Startup Contexts. In *Proceedings of the 13th International Workshop on Variability Modelling of Software-Intensive Systems (VAMOS '19)*. Association for Computing Machinery, New York, NY, USA, Article Article 10, 6 pages. https://doi.org/10.1145/3302333.3302347

[27] Klaus Pohl, Günter Böckle, and Frank J van Der Linden. 2005. *Software product line engineering: foundations, principles and techniques*. Springer Science & Business Media.

[28] Per Runeson and Martin Höst. 2009. Guidelines for conducting and reporting case study research in software engineering. *Empirical Software Engineering* 14, 2 (2009), 131–164. https://doi.org/10.1007/s10664-008-9102-8

[29] Holger Schlingloff, Peter M. Kruse, and Mehrdad Saadatmand. 2020. Excellence in Variant Testing. In *Proceedings of the 14th International Working Conference on Variability Modelling of Software-Intensive Systems (VAMOS '20)*. Association for Computing Machinery, New York, NY, USA, Article 12, 2 pages. https://doi.org/10.1145/3377024.3377028

[30] K. Schmid and M. Verlage. 2002. The economic impact of product line adoption and evolution. *IEEE Software* 19, 4 (2002), 50–57.

[31] D. Sellier, M. Mannion, G. Benguria, and G. Urchegui. 2007. Introducing Software Product Line Engineering for Metal Processing Lines in a Small to Medium Enterprise. In *11th International Software Product Line Conference (SPLC 2007)*. 54–62.

[32] Karma Sherif and Ajay Vinze. 2003. Barriers to adoption of software reuse: A qualitative study. *Information & Management* 41, 2 (2003), 159–175.

[33] Williamson Silva, Igor Steinmacher, and Tayana Conte. 2019. Students' and instructors' perceptions of five different active learning strategies used to teach software modeling. *IEEE Access* 7 (2019), 184063–184077.

[34] M. Staples and D. Hill. 2004. Experiences adopting software product line development without a product line architecture. In *11th Asia-Pacific Software Engineering Conference*. 176–183.

[35] Stefan Strobl, Mario Bernhart, and Thomas Grechenig. 2010. An Experience Report on the Incremental Adoption and Evolution of an SPL in EHealth. In *Proceedings of the 2010 ICSE Workshop on Product Line Approaches in Software Engineering (PLEASE '10)*. Association for Computing Machinery, New York, NY, USA, 16–23. https://doi.org/10.1145/1808937.1808940

[36] M. Verlage and T. Kiesgen. 2005. Five years of product line engineering in a small company. In *Proceedings. 27th International Conference on Software Engineering, 2005. ICSE 2005.* 534–543.

[37] G. Zhang, L. Shen, X. Peng, Z. Xing, and W. Zhao. 2011. Incremental and iterative reengineering towards Software Product Line: An industrial case study. In *2011 27th IEEE International Conference on Software Maintenance (ICSM)*. 418–427.