Variability Management in Product Lines of Safety Critical Embedded Systems

Stephan Baumgart, Xiaodi Zhang
E&E System Architecture Department,
Volvo Construction Equipment,
Eskilstuna, Sweden
Email: (stephan.baumgart, xiaodi.zhang)@volvo.com

Joakim Fröberg, Sasikumar Punnekkat
School of Innovation, Design and Engineering,
Mälardalen University,
Västerås, Sweden
Email: (joakim.froberg, sasikumar.punnekkat)@mdh.se

***Abstract*—The product line engineering approach is a promising concept to identify and manage reuse in a structured and efficient way and is even applied for the development of safety critical embedded systems. Managing the complexity of variability and addressing functional safety at the same time is challenging and is not yet solved. Variability management is an enabler to both establish traceability and making necessary information visible for safety engineers. We identify a set of requirements for such a method and evaluate existing variability management methods. We apply the most promising method to an industrial case and study its suitability for developing safety critical product family members. This study provides positive feedback on the potential of the model-based method PLUS in supporting the development of functional safety critical embedded systems in product lines. As a result of our analysis we suggest potential improvements for it.**

***Keywords*—Product Lines; Functional Safety; Commonality and Variability; Model-Based Development; Embedded Systems**

I. Introduction

Software has become the key enabler for new functionalities in industrial products due to its vast potential, ease and cost effectiveness for innovations and prototyping. The boundless possibilities with software are also gradually making the application software as well as the final products increasingly complex. There is a high demand to get the products faster to the market in order to be competitive. This leads to strong requirements to achieve a sufficient high level of quality in shorter time while the complexity of the products are increasing. Industry attempts to solve this challenge by reusing already developed parts or artifacts to the maximum extent feasible. This strategy aims for faster time to market and reduced development cost. But reuse itself may not always be beneficial and may not always automatically lead to faster time to market. Product lines and especially software product line engineering concepts are dealing with identifying the artifacts and components and evaluate the potential for reuse.

When developing safety-critical products, standards help to establish confidence in the products developed. Functional safety is a part of the overall product safety and is concerning failures and faults in the products Electrical and Electronic (E&E) system. Functional safety standards like IEC 61508 or the automotive standard ISO 26262 provide a set of requirements and process steps to identify critical functions and classify their criticality. The common approach among all functional safety standards is to identify possible hazards that are caused by malfunctioning of the embedded system and classify the criticality of such a malfunctioning. All standards state requirements on the safety life-cycle, i.e., how the critical function or component should be developed. Throughout the development phases, evidences need to be collected to be able to argue for the safety of the system.

When developing safety critical products in product lines, all product line members need to meet the safety goals if safety critical functions are used in their configurations. Depending on the configuration, different functional safety strategies may be used. Today functional safety practitioners are facing the problem of having limited information on possible variability and configurations, to be able to assess the compliance of a safety standard comprehensively. This leads to an uncertainty when the design has not been analyzed thoroughly. Project delays and higher cost when critical configurations are identified too late have been reported. It is important to find ways to collect the relevant information and visualize the variability of the system from the beginning.

The focus of this study is to identify an appropriate method capable of visualizing the variability for product lines of safety critical embedded systems and to evaluate its applicability. Specifically, we investigate the appropriateness of Product Line UML-based Software Engineering (PLUS) approach presented by Hassan Gomaa [1] in this context.

The main contributions of this paper are the identification of specific requirements for variability management methods for the development of safety critical embedded systems in product lines, an investigation on the applicability of the PLUS method by applying it on a typical industrial example, and suggestions regarding potential improvements to PLUS for addressing safety critical products.

Based on our experience and earlier studies we set up requirements for variability management methods in section II, followed by studying and evaluating existing variability management methods in III. We describe a typical case from the construction equipment domain in section IV. In section V

978-1-4799-5026-3/14/$31.00

we apply the PLUS method to the described case and present results from our analysis. We review the identified requirements in VI and conclude the paper in VII.

II. Requirements

Product line theory is focused on methods that capture commonalities to achieve cost-efficiency. In this study, we are also interested in capturing the variability aspects and their relation to functional safety work products for assuring the safety requirements. We envision a method that allows both the commonality and the variability analysis to support a safety centric development and analysis of product lines. For such a method to be effective, several requirements are essential such as general methodological requirements *(G)*, reuse-centric requirements *(R)* and safety-centric requirements *(S)*. Based on previous studies [2][3] and our industrial experience, we have identified the following non-exhaustive list of requirements (against each requirement we also indicate within parenthesis their category):

(R1) Safety analysis techniques - An effective method should be capable to support/integrate multiple safety analysis techniques as prescribed by the domain specific safety standards. *(S)*

(R2) Variation points and configurations - In order to perform an effective safety analysis a method must correctly identify all variation points and relevant configurations. Multiple configurations and modes will need to be supported. *(S, R)*

(R3) Life-cycle coverage - Because functional safety regards the complete product life-cycle, i.e. development, production, maintenance and decommissioning, the method should be capable of supporting analysis throughout the life-cycle phases of products to enable a mapping of functional safety requirements. *(G, S, R)*

(R4) Visualization - The method would enable an effective visualization and support communication and understanding, e.g. to communicate requirements among designers or to development partners more efficiently. *(G, S, R)*

(R5) Static and dynamic characteristics - The method should support both static and dynamic views of the embedded system including the system boundary description. This information is necessary to trace design decisions, analyze the impact of changes and perform safety analyses considering the systems characteristic from different perspectives. *(S)*

(R6) Scalability A method must scale and support large product lines and a large set of complex configurations. *(G)*

(R7) Commonality A method should exhibit the ability to capture and depict commonality for product line members and also for safety related concepts. *(S, R)*

(R8) Evolution Possibility to capture and trace new or changed existing variation points and variations introduced during evolution of the product line. This enables an impact analysis to identify potential violations of the safety concepts. *(S)*

(R9) Traceability The effects of changes should be traceable to show all affected parts of the systems and possible configurations. *(S, R)*

III. Literature Study

A. Introduction

The software product line engineering (SPLE) concept provides a paradigm for developing products with higher quality, lower cost and faster time to market [4]. With this approach, reusing components and development artifacts is planned and managed, thus enabling companies to perform cost projections and evaluate the reuse of components and development artifacts [5]. One key activity to be performed when developing products using product lines is the commonality and variability analysis, which provides the foundation for reuse [6]. A wide range of variability management techniques has been proposed in the past. Chen et al. [7] provide an overview on variability management methods proposed in literature and explain the challenges the different approaches aim to address. Variability can either be adapted to existing development artifacts and models or can be extracted and visualized in separate models. In the following we provide an overview on typically used variability management methods and discuss the fulfillment of the requirements.

B. Variability Management - feature orientation

Many feature based modeling approaches are focusing on modeling the characteristics of a system to improve communicating variability to customers. Kang et al. [8] propose the Feature-Oriented Domain Analysis (FODA), which aims at detecting the common and variable features and its potential reuse between different products. The authors define the term feature as a user-visible characteristic or quality of a system. The feature tree method aims to ”bridge the gap between requirements and technical design decisions” [9]. FODA uses a tree structure to visualize features, its relation and depicts the product features covered by the product line in a hierarchical and straightforward way. Features and their relation are analyzed and documented during concept phase, investigating the potential reuse of the features. The method does not allow deeper analysis as it is proposed, once the common and variable features are identified to a certain level. Information about product related assets, e.g. hardware and context information are not covered and extension have been proposed to cover other development phases [10]. The feature modeling itself is useful for designing a product line, but lacks in covering the complete product life-cycle.

C. Variability Management - extraction to external models

Variability can be managed by adding a variability notation to existing development artifacts and model-based approaches. Some researchers argue that the resulting models are becoming too complex and propose to extract information about variability into a specific model. Prominent examples of external models for variability are the concept of Pohl et al. presented in [11] and the concept COVAMOF presented by Sinnema et al. in [12]. In COVAMOF the product family artifacts are separated from the introduced ”variation point view” and the ”dependency view”. The importance of this method is

	Feature oriented variability management	Extracting variability	Life-cycle based variability Management
R1	partly	partly	partly
R2	x	x	x
R3	-	partly	partly
R4	x	partly	x
R5	-	-	x
R6	x	x	x
R7	x	partly	x
R8	x	x	x
R9	-	partly	x

Fig. 1. Mapping of requirements to categories of variability management techniques

that development levels from concept phase until implementation are supported, which enables a uniform representation of variability in some phases of the development process. Furthermore dependencies and their tracing through different abstraction levels and even interactions between dependencies are represented.

D. Variability Management - life-cycle coverage

In comparison to FODA, model-based approaches based on UML like PLUS [1] and the SysML-adaption presented in [13] for product line development are proposed aiming to cover the benefits from the feature tree concept and providing methods to derive a product line at the same time. In the PLUS method Gomaa is proposing techniques to design software product lines by using standard UML diagrams that are extended by new stereotypes for kernel, alternative and optional elements and relations. A range of UML models for different development phases covering requirements identification, domain analysis and design are part of the approach. Supporting the product life-cycle is important since variability is managed in an uniform way, which enables considering functional safety for the product line at the same time.

E. Pros and cons of different approaches

Many variability management approaches focus on representing variability on a high abstraction level to communicate the variability to customers. These feature-modeling approaches are lacking concepts for tracing variability through the complete development process. The requirements defined in section II help us to identify potential variability management methods by reviewing the concepts and their requirements fulfillment (Fig. 1). To our knowledge none of the investigated techniques is considering safety analysis techniques and supporting safety analysis techniques needs to be shown first (R1).

1) Feature oriented variability management: The main focus of the feature oriented variability management methods is to communicate variability to the customers. Therefore the methods do not aim to cover the whole life-cycle and not all required safety analysis techniques can be related (R1) and functional safety requirements cannot be mapped. Because of just covering the concept phase of the development process, only changes on feature level can be traced (R9). Different systems characteristics are not being able to depict (R5).

2) External models for variability management: In this category fall variability management methods which extract information about variability and manage the information separated from the other development artifacts. Tools and methods are required to keep the links between development artifacts and variability model consistent in order to be able to trace changes (R9). Communicating the variability effectively seems challenging (R4), when variability is separated from the development artifacts. Different characteristics of the system are not considered, if not explicitly managed in other development artifacts (R5). Methods like COVAMOF explicitly visualize variability. How well commonality is depicted needs to be further investigated (R7). Even the covering of the product life-cycle, as required in (R3), needs to be shown.

3) Life-cycle based variability management: The variability management methods we group here, cover parts of the development process and aim for example to identify a product line architecture. Variability is not extracted and instead the information on variability is added to development artifacts. The concepts analyzed promise to fulfill most of the requirements we set up. Mapping the safety analysis methods (R1) and how a complete product life-cycle is covered (R3) needs to be shown. Generally the variability management methods that apply a model-based approach seem to be capable manage commonality and variability even for safety critical product line members.

F. Other relevant approaches

Generally variability management concepts like FODA and PLUS are not covering concepts or approaches to support the development of safety critical systems. Nonetheless extensions to safety analysis techniques like fault tree analysis (FTA) and failure mode and effects analysis (FMEA) have been proposed in literature [14] to be applicable for product lines. The PL-FTA presented in [15] is adding information about commonality and variability to the leaf nodes in the fault tree, enabling product specific tree pruning during application development depending on the product configurations. The proposed techniques require already complete information about identified commonality and variability, which are not always available in the industrial cases. Before considering applying product line safety analysis techniques such as PL-FTA, visualizing the variability of the product line is necessary.

Out of available variability management methods, we choose PLUS for this study. The rationale is that the PLUS method is fulfilling many of the requirements and is therefore most promising in terms of applicability and adaptability in an industrial context. In the following we describe a typical case from industry and apply the model-based concept PLUS to it in order to gain insight information and reveal further requirements and necessary extension to be applicable for the development of safety critical products in product lines.

IV. Case Description

In this section we describe an illustrative example from the construction equipment domain and the potential config-

urations. We choose a steer-by-wire system in two different configurations - Variant 1: using a lever for steering to left and to right, and Variant 2: a joystick steering supporting both forward and backward driving and the left-right steering.

A. Variant 1 - Left-right steering

The forward and backward movement of the machine is realized through selecting the driving direction and by using the gas-pedal for acceleration, while a lever can be used for left-right steering. During operation or when driving on public roads a wrong calculated or unintended steering may lead to serious accidents. Transported goods might fall onto other workers or other traffic might be hit when traveling on public roads.

B. Variant 2 - Joystick Steering

The joystick steering is a more complex system that applies on top of the above-described left-right steering function also the forward-backward driving. Interfaces to the automatic gearbox and the engine are necessary, where the first is receiving inputs to select forward or backward gears and the latter is receiving inputs on the targeted speed. Potential failures might result in hazards like unintended acceleration and unintended change in the direction of motion (forward to backward). Higher safety goals involving more parts of the embedded system need to be reached.

Both variants are implemented using safety concepts that utilize different parts of the distributed embedded system e.g. using an independent on-line monitoring on a different ECU.

C. Product line scenarios and impact on functional safety

Both variants seem to be easy to develop, manage and configure. But in an industrial context those variants might be implemented in different ways requiring different functional safety concepts. In the following we list typical scenarios for product lines of construction equipment machinery and their impact on the implemented functional safety concepts.

Scenario 1: Market differences A machine type might be sold in different configurations on different markets. Functional safety concepts valid in one country might not be applicable in others.

Scenario 2: Differences between product line members Even though features might look similar, the architecture of the embedded systems can differ. Larger machines utilize more ECUs in comparison to smaller machines of the same product line. This results in different functional safety concepts that need to be applied.

Scenario 3: Product application Different usage scenarios of product line members will result in different types of hazards. While smaller machines may be used in public areas are larger machines are often used off-road. Different usage scenarios and hazards require different functional safety concepts to meet the derived safety goals.

Scenario 4: Evolution The next generation of the product line may require to remove the mechanical steering wheel for the joystick variant. For those products rebooting the ECU is no safe state and instead the joystick steering needs to be implemented as a fail-safe system.

Scenario 5: Reuse across product lines Features that are successfully implemented in one product line may be applied in a different product line as well. Simply using a copy-paste strategy may lead to unexplored safety goal violations in the new product line.

We want to explore the variability and identify violated safety goals already in early stages. In the following we apply the PLUS method in the steer-by-wire case.

V. Application of PLUS

Gomaa [1] proposes the Product Line UML-based Software Engineering (PLUS), which is a method extending the standard UML notation for capturing product line specific characteristics in UML diagrams and in [16] the author describes the PLUS process. Requirements for the product line are specified and used as an input for the PLUS process. Gomaa is foreseeing a Product Line Reuse Library, where product line artifacts are stored. Those artifacts are reused and integrated during application engineering phase. Through a feedback loop, changes made in the application, will result in updates of product line development artifacts in the library. The PLUS concept is covering the development phases: requirements identification, domain analysis and modeling the design. In the course of this study we focus on the requirements identification and domain analysis parts.

A. PLUS Requirements Model

The PLUS requirements model is aiming to capture the requirements for the product line by applying use case models and feature models.

(a) *Use case model*
Concept: The use case model depicts the functional requirements, which are addressed in terms of actors and use cases. To distinguish between common and variable use cases, PLUS introduces use case stereotypes: kernel, optional and alternative. The product line commonality is captured by the kernel use cases, while the variability is captured by the optional and alternative use cases.
Case: We applied the steer-by-wire example for the use case model and extracted the necessary information from existing documentation. The described use cases can be applied for feeding the Preliminary Hazard Analysis (PHA), because typical application scenarios of the final products are specified. Highlighting the critical scenarios in the use case models is not possible in the PLUS concept as it is today. New stereotypes for the use cases will make this possible.

(b) *Feature model*
Concept: A feature is a requirement or characteristic that is provided by one or more product line members. The PLUS approach extends the UML class diagrams to create a feature tree adding feature groups and static dependencies and relations. The feature model depicts the commonality and variability of the software product line

system using either of the stereotypes, common, optional or alternative. Features have many-to-many relationship with the use cases. When a feature is mapped to a group of use cases, those use cases have the potential to be reused together. When many features are mapped to a use case, those features relate to the use case variation points.

Case: Applying the proposed feature tree concept to the steer-by-wire case was demanding, since the required information about variability and commonality and even the grouping of features was not easy to derive from existing documentation. The PHA results in identifying safety critical features and according criticality levels (e.g. SIL, ASIL). The safety critical features will lead to specific architecture solutions and therefore this information needs to be captured. Safety critical features might depend on inputs from other features and both static and dynamic dependencies will need to be captured. Especially dynamic dependencies are not defined in PLUS yet.

(c) *Feature-Use case dependencies*

Concept: In order to be able to trace information from the use case diagrams to the feature tree, a table is proposed to map the different use cases to the features. For each feature described in the feature tree, the according use case including variation point is identified and mapped.

Case: We created the table mapping the use cases for the steer-by-wire system to the identified features. For large product lines, mapping features to use cases will require tool support in order to both establish traceability and to perform consistency checks when changes are made. The safety critical use case are not yet highlighted and therefore tracing to related features is not possible.

B. PLUS Analysis Model

In addition to the requirements model, the PLUS model includes an analysis model, which is covering both static and dynamic views of the system.

(a) *Static model*

Concept: The static model depicts the product line static structure, describing both internal and external components. Specifying the product line boundary is done through identifying product line external components e.g. external user, external device, external system or external time, and their communication direction.

Case: In our study identifying the boundary of the system has shown to be useful. It is of importance that the interfaces of the embedded system to the environment are up to date and changes in sensors or actuators can be immediately traced to the according software parts in the embedded system. Especially the complexity of large scale product lines makes it more and more difficult to oversee the impact of a change to all potential configurations. This traceability helps performing functional safety assessments and checking the safety arguments.

(b) *Dynamic model*

Concept: Objects are derived from the static models. The dynamic interaction model addresses the interaction between those objects, which satisfies the product line functional requirements depicted in the use case model. The interaction model has a one-to-one relationship to the use case in the use case model. Furthermore the state transition diagram depicts the finite state machine, being referred to as statechart diagram in UML, which describes the control and sequencing view of a system. The state transition in the statechart is caused by input events, which are identical with the messages in the dynamic interaction diagram.

Case: Especially addressing the dynamic characteristics of the product line has shown to be useful when functional safety standard compliance needs to be achieved for product lines. This information is necessary to trace and assess the timing behavior of the distributed embedded system. The flow of information can be analyzed thoroughly and a clear representation of safety concepts for different variants is necessary. The concept of merging the statecharts of different product family members is beneficial to get an overview, which product variant is using which states. The reachability of safe states can be checked and even be simulated.

We also performed a detailed fault tree analysis to investigate the quality and availability of information provided in the PLUS models, which confirmed the potential of the PLUS method to be applied in developing safety critical products in product lines [17].

VI. Discussions

We have modeled the steer-by-wire system according to PLUS procedures. Here we evaluate the outcome by discussing the fulfillment of the requirements R1-R9.

(R1) Safety analysis techniques - Both PHA and FTA are commonly used hazard analysis techniques in an industrial context. The information captured in the diagrams of the PLUS Requirements Model can be used for conducting a PHA. The static and dynamic information specified in the analysis model are useful for conducting a FTA. Generally adding the results of the hazard analyses to the proposed diagrams can be useful for the following developing stages.

(R2) Variation points and configurations - In the models we applied in this study commonality and variability is addressed. Nonetheless we realized in our study that static and dynamic dependencies between features and variation points are necessary to capture in order to derive constraints for deriving for example an architecture that shall support all safety goals.

(R3) Life-cycle coverage In our case we did not explicitly perform any analysis on life-cycle phases like software implementation, verification and maintenance. In order to test applicability for life-cycle purposes, we will need to define realistic scenarios covering more life-cycle phases to evaluate the model. The main goal of the PLUS method is to provide

guidance to derive a product line architecture. Therefore the PLUS model does not cover all product life-cycle phases. Extensions of the PLUS model could help achieving this requirement.

(R4) Visualization - When applying the PLUS method it was challenging to find the right information in available specifications. Information on commonality and variability is managed by the experts and is partly hidden. This makes it difficult to perform a thorough analysis. The PLUS method is useful to communicate and discuss the characteristics of a product line.

(R5) Static and dynamic characteristics - Both static and dynamic characteristics of the product line members can be captured by the PLUS method. Extending the diagrams to manage even variability for functional safety work-products to enable traceability even to those development artifacts is necessary.

(R6) Scalability - As far as we can see from applying PLUS to the case, the model is scaling well. A tool support is needed in order to trace information between the different diagrams and keep the different parts of the model consistent. Especially when more properties are added to the different diagrams aiming to address functional safety and dependencies between features and variation points are also captured, the diagrams become more complex.

(R7) Commonality - The PLUS method aims to manage the commonalities as well. A potential extension would be, to clarify the safety concepts that are commonly applied.

(R8) Evolution - We applied the PLUS method to a simplified case. In order to evaluate how PLUS applies to evolution a product line a more advanced case study is needed.

(R9) Traceability - The specific characteristics specified in the different diagrams is traceable. The traceability is established through tables, where elements of the different diagrams are mapped and traceability is established.

Generally the PLUS method seems to be a good starting point for investigating how functional safety can be achieved in product lines. There is a range of potential extensions we can foresee.

VII. Conclusions

The main goal of the product line engineering approach is to identify commonality to achieve higher efficiency during development. From the functional safety perspective the variation points among the product variants and their dependencies are of higher interest. Inability to capture sufficient details of the variability (planned, hidden and evolutionary) could potentially result in faulty designs, costly development efforts or even lead to unsafe products. It is important to have an adequate method, which makes it easier to visualize and analyze all possible configurations from a safety perspective. In this paper, we have presented a non-exhaustiveness list of key requirements for such a method and investigate the use of PLUS, a model-based product line engineering approach in this context. We describe a representative example from industry and provide an analysis how PLUS will need to be extended to be suitable for developing functional safety critical embedded systems in safety critical product lines. We provide pointers towards enhancing PLUS to be more relevant in the analysis of safety critical products.

VIII. Acknowledgments

The research leading to these results has received funding from the ARTEMIS Joint Undertaking under grant agreement no 295373, Vinnova and the KKS-funded ITS-EASY Post Graduate School for Embedded Software and Systems and SSF funded Synopsis project.

References

[1] H. Gomaa, *Designing software product lines with UML*. Addison-Wesley Boston, USA, 2004.

[2] S. Baumgart, J. Froberg, and S. Punnekkat, "Towards efficient functional safety certification of construction machinery using a component-based approach," in *Product Line Approaches in Software Engineering (PLEASE), 2012 3rd International Workshop on*, June 2012, pp. 1–4.

[3] S. Baumgart, J. Froberg, and S. Punnekkat, "Industrial challenges to achieve functional safety compliance in product lines," in *Software Engineering and Advanced Applications (SEAA), 2014 40th EUROMICRO Conference on*, Aug. 2014 (in publication).

[4] D. Weiss and C. Lai, *Software Product-Line Engineering: A Family-Based Software Development Process*. Addison-Wesley Professional, 1999.

[5] P. Clements and L. Northrop, *Software product lines*. Addison-Wesley, 2001.

[6] J. Coplien, D. Hoffman, and D. Weiss, "Commonality and variability in software engineering," *Software, IEEE*, vol. 15, no. 6, pp. 37–45, Nov 1998.

[7] L. Chen, M. Ali Babar, and N. Ali, "Variability management in software product lines: A systematic review," in *Proceedings of the 13th International Software Product Line Conference*, ser. SPLC '09. Pittsburgh, PA, USA: Carnegie Mellon University, 2009, pp. 81–90. [Online]. Available: http://dl.acm.org/citation.cfm?id=1753235.1753247

[8] K. C. Kang, S. G. Cohen, J. A. Hess, W. E. Novak, and A. S. Peterson, "Feature-oriented domain analysis (FODA) feasibility study," DTIC Document, Tech. Rep., 1990.

[9] M. Svahnberg, J. Van Gurp, and J. Bosch, "A taxonomy of variability realization techniques," *Software: Practice and Experience*, vol. 35, no. 8, pp. 705–754, 2005.

[10] K. C. Kang, S. Kim, J. Lee, K. Kim, E. Shin, and M. Huh, "Form: A feature oriented reuse method with domain specific reference architectures," *Annals of Software Engineering*, vol. 5, no. 1, pp. 143–168, 1998.

[11] K. Pohl, G. Böckle, and F. Van Der Linden, *Software product line engineering: foundations, principles, and techniques*. Springer-Verlag New York Inc, 2005.

[12] M. Sinnema, S. Deelstra, J. Nijhuis, and J. Bosch, "Covamof: A framework for modeling variability in software product families," in *Software Product Lines*. Springer, 2004, pp. 197–213.

[13] I. Habli, I. Ibarra, R. Rivett, and T. Kelly, "Model-based assurance for justifying automotive functional safety," in *Proc. 2010 SAE World Congress*, 2010.

[14] Q. Feng and R. R. Lutz, "Bi-directional safety analysis of product lines," *Journal of Systems and Software*, vol. 78, no. 2, pp. 111 – 127, 2005. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0164121205000105

[15] J. Dehlinger and R. Lutz, "Software fault tree analysis for product lines," in *High Assurance Systems Engineering, 2004. Proceedings. Eighth IEEE International Symposium on*, March 2004, pp. 12–21.

[16] H. Gomaa, "Designing software product lines with uml 2.0: From use cases to pattern-based software architectures." IEEE Computer Society, 2006.

[17] X. Zhang, "Methods for modeling of product lines for safety-critical systems," Master's thesis, School of Innovation, Design and Engineering, Mälardalen University, 2013.