Evaluation of a systematic approach in variant management for safety-critical systems development

Authors:
Michael Käßmeyer
David Santiago Velasco Moncada
Markus Schurius

IESE-Report No. 075.15/E
Version 1.0
December 2015

A publication by Fraunhofer IESE
Fraunhofer IESE is an institute of the Fraunhofer Gesellschaft. The institute transfers innovative software development techniques, methods and tools into industrial practice, assists companies in building software competencies customized to their needs, and helps them to establish a competitive market position.

Fraunhofer IESE is directed by
Prof. Dr.-Ing. Peter Liggesmeyer
(Executive Director)
Prof. Dr. Dieter Rombach
(Director Business Development)
Fraunhofer-Platz 1
67663 Kaiserslautern
Germany
Evaluation of a systematic approach in variant management for safety–critical systems development

Michael Käßmeyer∗, David Santiago Velasco Moncada†, Markus Schurius∗
∗Audi Electronics Venture GmbH, Gaimersheim, Germany, Email: michael.kaessmeyer@audi.de, Email: markus.schurius@audi.de
†Fraunhofer Institute for Experimental Software Engineering IESE, Kaiserslautern, Germany, Email: santiago.velasco@iese.fraunhofer.de

Abstract—The development of highly integrated, safety-relevant automotive functions is faced with the challenge of increasing complexity resulting from product customization and variants in implementation through software-hardware solutions. In order to reduce development time in this scenario, systematic reuse of engineering artifacts is important. This paper introduces a systematic model-based engineering approach that combines architecture design, requirements engineering, and safety analyses with variant management and provides evaluation results to address these challenges. In detail, this tool-supported approach achieves a new level of seamless safety engineering across variants by enabling typical safety lifecycle artifacts to be represented in a homogeneous, UML-compliant model notation. Safety-related information is no longer scattered in various isolated tools and formats, but instead consolidated and integrated.

A further and decisive benefit of this notation is that variability can now be expressed and managed easily by regular variant management tools with UML adapters. Together with change-impact analysis, which is facilitated equally the ultimate goal of developing and maintaining modular safety cases can be achieved. Examples on how to use this model-based safety engineering method for variant-rich automotive functions are presented for a hazard analysis, a fault tree analysis and for a safety concept specification.

Keywords—Product Line Engineering, Functional Safety, Model-based Embedded Systems

I. INTRODUCTION

Ever since software became part of the automotive industry, engineers have been faced with the challenge to keep up with the innovation and complexity of safety-relevant functionalities. In a domain in which “time to market” is such an issue, the ever increasing number of safety-critical services calls for the adoption of new methodologies for coping with approaches such as Product Line Engineering (PLE) and Component-Based Software Engineering (CBSE) [5], [6]. The former has been primarily used to cope with variant management and reusability issues, while the latter has been a means for the evolution of traditional monolithic safety analysis into modular approaches [7], [13].

Traditionally, PLE and safety analysis approaches have been developed separately [7]. However, since the definition of the safety standard ISO 26262 [2], the automotive industry has seen the need for a unified approach that integrates the benefits of both fields. Experience has shown that the common “Clone and Own” reuse approach is inefficient in the long term and that one of the main reasons for that is the need to assure functional safety for individual products [1]. According to ISO 26262, an automobile (i.e., the system) should behave safely under certain operational conditions. However, exchanging a component in a system, for instance, might modify the system’s definition itself. Therefore, the functional safety assessment of the system is not valid anymore and needs to be performed all over again [7]. Furthermore, depending on the functionality being exchanged or integrated, the safety assurance process can be very complex and partially imprecise, especially if tool support is weak.

To this day, there is no holistic approach that supports all safety-relevant assets in the development process of safety-critical systems that could properly integrate PLE. Nevertheless, given the huge interest in this subject, the goal and contribution of this work is to evaluate the Open Safety Model (OSM) [11] approach for the integral development of safety-critical systems with respect to variant management and, consequently, the possibility to reuse safety-critical systems or components. For this purpose, we make use of a case study based upon an electrical driving system. The practical evaluation was performed on the basis of a tool integration including pure::variants†, Enterprise Architect‡, and I-Safe§. This paper is structured as follows: Section II gives an overview of safety and product line engineering. Section III compiles several common requirements and issues to be considered regarding the development of a solution approach. Section IV introduces the Open Safety Model. Section V describes the methodological foundation on which this evaluation is based, followed by the case study itself in section VI. Finally, section VII presents a discussion with respect to previous studies, followed by the conclusions and future work in section VIII.

†http://www.pure-systems.com/
‡http://www.sparxsystems.com
§I-Safe: This tool is being developed at Fraunhofer IESE, http://www.iese.fraunhofer.de/en.html.

Abstract—The development of highly integrated, safety-relevant automotive functions is faced with the challenge of increasing complexity resulting from product customization and variants in implementation through software-hardware solutions. In order to reduce development time in this scenario, systematic reuse of engineering artifacts is important. This paper introduces a systematic model-based engineering approach that combines architecture design, requirements engineering, and safety analyses with variant management and provides evaluation results to address these challenges. In detail, this tool-supported approach achieves a new level of seamless safety engineering across variants by enabling typical safety lifecycle artifacts to be represented in a homogeneous, UML-compliant model notation. Safety-related information is no longer scattered in various isolated tools and formats, but instead consolidated and integrated.

A further and decisive benefit of this notation is that variability can now be expressed and managed easily by regular variant management tools with UML adapters. Together with change-impact analysis, which is facilitated equally the ultimate goal of developing and maintaining modular safety cases can be achieved. Examples on how to use this model-based safety engineering method for variant-rich automotive functions are presented for a hazard analysis, a fault tree analysis and for a safety concept specification.

Keywords—Product Line Engineering, Functional Safety, Model-based Embedded Systems

I. INTRODUCTION

Ever since software became part of the automotive industry, engineers have been faced with the challenge to keep up with the innovation and complexity of safety-relevant functionalities. In a domain in which “time to market” is such an issue, the ever increasing number of safety-critical services calls for the adoption of new methodologies for coping with approaches such as Product Line Engineering (PLE) and Component-Based Software Engineering (CBSE) [5], [6]. The former has been primarily used to cope with variant management and reusability issues, while the latter has been a means for the evolution of traditional monolithic safety analysis into modular approaches [7], [13].

Traditionally, PLE and safety analysis approaches have been developed separately [7]. However, since the definition of the safety standard ISO 26262 [2], the automotive industry has seen the need for a unified approach that integrates the benefits of both fields. Experience has shown that the common “Clone and Own” reuse approach is inefficient in the long term and that one of the main reasons for that is the need to assure functional safety for individual products [1]. According to ISO 26262, an automobile (i.e., the system) should behave safely under certain operational conditions. However, exchanging a component in a system, for instance, might modify the system’s definition itself. Therefore, the functional safety assessment of the system is not valid anymore and needs to be performed all over again [7]. Furthermore, depending on the functionality being exchanged or integrated, the safety assurance process can be very complex and partially imprecise, especially if tool support is weak.

To this day, there is no holistic approach that supports all safety-relevant assets in the development process of safety-critical systems that could properly integrate PLE. Nevertheless, given the huge interest in this subject, the goal and contribution of this work is to evaluate the Open Safety Model (OSM) [11] approach for the integral development of safety-critical systems with respect to variant management and, consequently, the possibility to reuse safety-critical systems or components. For this purpose, we make use of a case study based upon an electrical driving system. The practical evaluation was performed on the basis of a tool integration including pure::variants†, Enterprise Architect‡, and I-Safe§. This paper is structured as follows: Section II gives an overview of safety and product line engineering. Section III compiles several common requirements and issues to be considered regarding the development of a solution approach. Section IV introduces the Open Safety Model. Section V describes the methodological foundation on which this evaluation is based, followed by the case study itself in section VI. Finally, section VII presents a discussion with respect to previous studies, followed by the conclusions and future work in section VIII.

†http://www.pure-systems.com/
‡http://www.sparxsystems.com
§I-Safe: This tool is being developed at Fraunhofer IESE, http://www.iese.fraunhofer.de/en.html.
II. SAFETY AND PRODUCT LINE ENGINEERING

This section is split into two sub-sections: the life cycle of automotive safety engineering and the approach of product line engineering. This section gives a brief background with respect to several core used terms, the activities performed, the work products and the process tasks.

A. Automotive safety engineering

The safety lifecycle in the automotive industry is described in ISO26262. Important work products that provide evidence that a driving function complies with this safety standard are depicted in Fig. 1.

![Figure 1. V-model with safety related work products](image)

The starting point of the safety process is the item definition. Based on this information, a hazard and risk analysis is done and the appropriate Automotive Safety Integrity Level (ASIL), which expresses normative required risk reduction measures, is determined. Next, using techniques such as fault tree analysis (FTA) or failure mode and effects analysis (FMEA), a safety analysis is performed aimed at identifying potential sources causing the violation of safety goals (e.g., the prevention of unintended acceleration). These safety analyses are applied at different architecture and design viewpoints and abstraction levels (e.g., function, system, and component) in order to derive and allocate safety mechanism (e.g., fault detection and reaction), which are then specified as safety requirements. The functional and technical safety concepts aggregate the safety requirements and allocate them to architecture elements that specify which identified failure modes and faults are mitigated by corresponding safety mechanisms. Due to the frequent occurrence of change requests, this is a dynamic and iterative optimization procedure. Finally, the safety case structures the argumentation (i.e., claims and evidences regarding the reasons why the developed item achieved functional safety).

B. Product line engineering

A software product line involves different software systems, distinguished as variants, which share a common set of features and which fulfill domain-specific requirements. A variant is built by instantiating common and specific requirements from the domain specification into a concrete application.

Fig. 2 shows the software product line engineering framework [14]. Domain engineering includes all activities affecting the development of the product line. Requirements engineering, design, realization, and testing of the core assets belong to this part. Domain engineering creates the basis for a software product line so that it becomes possible to configure products. In order to derive a product from the product line, the application engineering process is used. Within this engineering activity, the same steps are performed as in domain engineering, with the difference that they relate to a configured product. One approach for dealing with variability is feature-oriented domain analysis (FODA).

![Figure 2. Software product line engineering framework](image)

FODA [15] is a method for performing domain analysis, where common characteristics of related software systems are analyzed and characterized by the generic description of requirements. A FODA feature tree represents the domain and contains the common characteristics and differences of applications of a software product line. A significant result of the FODA approach is a feature.

![Figure 3. Feature tree](image)

A feature describes a property that differentiates products from a stakeholder’s view. Within a feature tree, the features are arranged hierarchically. The features are classified as “mandatory”, “optional”, “alternative”, and “or” [15]. In Fig. 3, a simple feature tree of a car example is depicted.

III. CHALLENGES

In this section, we describe several recurrent challenges encountered during the integration of the PLE and the safety lifecycle. They will be used later in this paper to evaluate the OSM+PLE approach:
Challenge 1. Modular composition

In order to be able to efficiently reuse parts of a system, modularity is required. This has always been a challenge because safety is a quality attribute that is reflected across the structure of the overall system. Nevertheless, modular approaches for safety analysis have been presented in the past, e.g., Component Fault Trees (CFT) [9].

Challenge 2. Formality and traceability

In order to properly support configuration and change management, more formal relationships need to be established between safety and development artifacts. Today, most fault trees are not constructed directly on the existing architecture model but instead in a separate tool. This results in additional effort to eliminate inconsistencies and ambiguities between the two models. Moreover, traceability is a key factor for achieving automation, modeling, and analysis support. A traceable component-integrated approach for fault trees has been defined in [13].

Challenge 3. Support normative requirements

The safety assessment process is tied to normative requirements. Therefore, an integrated approach for PLE and safety should be tailored to domain-specific safety standards and integrate, for instance, their recommended modeling/analysis techniques, e.g., FTA or FMEA [3].

Challenge 4. Reuse capability and effort estimation

Given the difficulty of reusing all possible developed assets of a subsystem, a solution approach should clarify which ones can be reused as is, and which ones need to be modified or developed anew. This should help practitioners estimate the effort needed for reusing a component, examples are presented in [10].

Challenge 5. Requirement of tool support

In order to adequately support the integration of PLE within the safety lifecycle, it is necessary to provide proper tool support. This is required due to the number of artifacts that need to be maintained throughout the development and assessment phases.

IV. OPEN SAFETY MODEL

In the project Software Platform Embedded Systems eXtream Tailoring (SPES XT4), one of the main research topics was the development of methodologies for modular safety assurance. This was motivated primarily by the difficulty of reusing components in safety–critical systems, which usually demand the repetition of the safety assurance process as explained above. With the aim of reducing costs and facilitating the assessment process, the OSM [11] integrates modularity and compositional concepts for the incremental functional development and assurance of safety–related systems. The OSM was developed as an evolution of the SPES modeling framework [12] and the integration of the results of the Safe ReSA project [7] together with well-known techniques such as the Goal Structuring Notation (GSN) and new experimental approaches for contract-based design and context modeling.

In the OSM, an ontological integration of safety and non-safety artifacts is achieved in which different viewpoints and abstractions are supported, providing modeling flexibility while keeping artifacts traceable. The goal of the OSM is to consolidate a domain–independent safety perspective in which the relationships between the artifacts are clearly defined.

Fig. 4 depicts an abstraction of the ontological integration in which the main blocks of artifacts can be identified. The actual OSM contains hundreds of elements, which are not shown here. The modular safety assurance approach defined through the OSM takes the SPES modeling framework as the basis for the modular design of embedded systems. In this framework, several architectural viewpoints (i.e., Requirements, Functional, Logical, and Technical) specify the system, taking into consideration complementary aspects, which allows a more complete and traceable description of the system. Furthermore, the modeling framework also integrates the concept of abstraction, which allows depicting the system at different levels of granularity. By combining these two dimensions in an artifact-centric manner, the system can be modeled flexibly, while keeping the specification modular and seamlessly traceable at the same time. On top of the SPES framework (see Fig. 5), the OSM maps safety concerns to these modular structured work products, allowing the analysis and specification of safety–related aspects together with the architectural artifacts in a modular and reusable form. In this respect, the safety analysis (e.g., FTA, FMEA, Markov analysis) as well as the safety requirements can be structured modularly according to the specific viewpoint. Furthermore, thanks to the specification of the relationships between safety and non-safety artifacts, the OSM could support such things as change impact analysis and traceability – key factors for facilitating change management, reusability, and therefore modular assessment.

Thanks to the traces defined among the modular and non–modular techniques in the ontological model, it is possible to filter out all safety–related artifacts associated with a component. It is possible, for example, to obtain a list of hazardous events (as defined in ISO 26262) to which the failures of a function contribute. Furthermore, it is possible to validate

---

4"SPES XT" is a project funded by the German Federal Ministry of Education and Research (BMBF), which started in May 2012. http://spes2020.informatik.tu-muenchen.de/
V. APPROACH AND MODELING CONCEPT

The purpose of this work is not to explain the OSM in detail, but to evaluate it with respect to variant management and reuse. Therefore, we followed a systematic approach by integrating several modeling tools. The basis of the integration is formed by Enterprise Architect (EA), I-SaFe – which implements the OSM and builds on EA as an add-in, and pure::variants, which provides variant management and modeling of tailoring features for EA.

With this set of tools, the evaluation of the integration between variant management and safety artifacts follows a typical annotative approach in which all variants are modeled together in one single model, typically called the 150% model [1], [8]. In this model, elements are tagged with variant constraints. These constraints create traces to the variability model and allow a product to be configured. This step is supported by the variant management tool, which automatically removes all unnecessary artifacts, resulting in a 100% model. For 150% models, a common challenge is to keep them maintainable. In this respect, the foundational SPES modeling framework on which the OSM is based is very helpful since the overall complexity gets reduced by introducing abstraction levels (car, system, component, unit, element) and viewpoints (logical, functional, technical).

A holistic meta-model concept is illustrated in Fig. 5 (cf. [13]). In our approach, we integrate a safety perspective (i.e., the OSM) and a variability model into a model-based software engineering framework, which enables defined abstraction layers and different viewpoints. In the following, the model-based software engineering framework will be outlined in detail.

The abstraction layers on the vertical axis range from the abstract level (vehicle) to the detailed level (unit). On the horizontal axis, there are four basic viewpoints. These separate the problem domain (requirements) from the solution domain (functional, logical, and technical). The requirement viewpoint supports the requirements engineering process. The functional viewpoint describes the functional structure of the system under development. The logical viewpoint provides a view of the component structure. The technical viewpoint considers the software and the hardware architecture of the system. The goals of this framework are separation of concerns as well as complexity reduction by abstraction and modularity.

In order to facilitate the development of safety-lifecycle artifacts required by ISO 26262 in a model-based fashion, the engineering framework is extended with the safety perspective (OSM). Moreover, safety artifacts are derived directly from model elements using different analysis methods. Four safety engineering activities can be performed directly on the modeled function and specification, which represents an item definition: hazard and risk analysis (HaRa), functional and technical safety concept development (FSC, TSC), and model-based safety analysis, e.g., component fault tree analysis [10].

The HaRa is performed on the vehicle level. An item definition is typically expected as input in order to start it. The required artifacts must be taken from the available item definition artifacts contained in the requirement and functional viewpoints.

The FSC is a structured model that allocates functional safety requirements to architecture elements and the appropriate safety analyses. The required artifacts must be taken from the system level from the requirements and functional viewpoints.

The TSC is a refinement of the FSC and consists of technical safety requirements. These are allocated to architecture elements and their associated safety analyses in order to argue the reasons for the functional and technical safety requirements to avoid or mitigate fault propagation. The required artifacts must be taken from corresponding artifacts of the logical and technical viewpoints of the system, component, and unit levels.

Safety analyses are conducted across all four viewpoints on the system, components, and unit levels to determine the safety requirements.

Variant management, also realized as perspective in the mod-
eling framework (but for the sake of clarity shown separately in Fig. 5), supports the modeling of multiple products with their dependencies in a feature tree. Moreover, the integration of the dependencies of multiple products in a feature tree is realized via a linking or referencing concept.

Change management and configuration management must be enforced across the three domains. Change requests must be characterized in a software change management system. The change specification refers to the relevant artifacts in the domains variant management, safety, and model-based development. Mature configuration management must provide a comprehensible evolution process, specific SPL evolution stages, engineering artifact evolution stages, and safety artifact evolution stages. Each SPL evolution stage must correlate with a particular stage of the artifacts in the different domains. Safety models as well as feature models and engineering models can be considered as common artifacts.

VI. CASE STUDY E-DRIVE SYSTEM

E-Drive is an electrical driving system whose main purpose is to regulate a vehicle’s acceleration. Due to the criticality of a system failure, it is safety-related. From the functional point of view, the E-Drive system distinguishes between two possible performance configurations, “efficiency” and “dynamic”. The former configuration is part of vehicles with less horsepower. In the latter variant, the opposite case is considered, and it is therefore included in high-performance models.

This section describes the workflow with the tools and the modeling concept (subsection VI.A). The second part illustrates the architecture design of the system, followed by the description of the variant model (subsection VI.B and VI.C). Afterwards, safety analyses are performed and described (subsection VLD). These include hazard analysis and risk assessment (HaRa) and fault tree analysis (FTA). Finally, we present how the safety concept structures safety requirements modularly.

A. Work flow and modeling concept

The workflow in this case study initially targets the development of the feature model and the architecture design in parallel. Afterwards, the safety analyses are applied in order to analyze the different architectures and define the safety requirements. Next, the functional and technical safety concepts are used to aggregate the safety requirements. Due to the frequent occurrence of change requests, this is a dynamic and iterative optimization procedure. Therefore, it is often necessary to switch the tool. The way to approach this issue is to separate safety, variant management, and model-based development into different tools as the different stakeholders are only responsible for their respective domain.

Each artifact is represented as a 150% model and later on the specific variants are extracted. This concept was chosen because it is not yet possible (technically speaking) to develop and modify a variant model (the 100% model) and synchronize the variant model back to the 150% model. In general the overall strategy is clearly to maintain rather one single (large) 150% model for all variants instead of several different 100% models.

In the following, the extraction of a variant from a 150% model is described. It starts with the selection of the features in the variants description model (VDM), which are checked and validated against the feature and families models. If this analysis is positive, it will generate the variants result model (VRM). This VRM is then used for extracting a variant. Each artifact in the VRM has an annotation expressing whether the element is existent in a variant or not. All artifacts in the VRM are analyzed in terms of being existing or not existing in the resulting variant. Finally, the variant with all true artifacts and their interfaces is extracted and generated in an output file.

B. E-Drive architecture design

The logical view of the system can be seen in Fig. 6.

The E-Drive system contains six interconnected components. Three of them are sensors: the Phase Current Sensor, the Rotor Angle Sensor, and the Accelerator Pedal Sensor. They measure the three phases of the electrical engine, its rotation angle, and the reference torque, respectively. There are two controllers, the Micro Controller and the Driver Controller. The former determines the torque and arranges the pulse-width-modulation (PWM), and the latter makes use of the PWM and adjusts the current for the engine and arranges the desired torque. Finally, the Shut off Emergency component is responsible for...
cutting the power supply of the Driver Controller whenever the Micro Controller detects undesired acceleration and calls for an emergency stop.

The Micro Controller component has also been refined, since it contains the main processing logic of the E-Drive system. Refinement details are depicted in the next abstraction layer in the model. Fig. 7 shows the Micro Controller as a 150% model in which the PhaseLimiter component reflects the variant point of the performance configuration.

The sub-components of the Micro Controller are: Torque Controller, Current Plausibility check, and PhaseLimiter. The Current Plausibility check adds the three current phases. The sum of the three phases should be approximately zero; otherwise, a shut-off signal would be emitted. The second component is the PhaseLimiter, which limits the phase current to an adjusted value. Finally, the Torque Controller component calculates a torque value and provides it in the form of PWM.

In Fig. 7, there are two data flow paths targeting the same PhaseCurrents port of the Torque Controller. One goes straight from the Port PhaseCurrents of the Micro Controller. The other path goes through the component (PhaseLimiter). This model illustrates a 150% model, which means the model is over-specified because two configurations are possible for the Micro Controller. In the model, we can see the constraint annotation mechanism used to mark the alternative path. Furthermore, there is also a hidden constraint annotation at the component PhaseLimiter (see Fig. 7). In a 100% model, of the variant “dynamic”, the component PhaseLimiter will be removed.

C. Variant modeling

A feature model (compare to Fig. 3) has been created with the help of the tool pure::variants to specify the configuration of our intended automobile models. Fig. 8 depicts a screen shot of the project (left) and the problem domain of the variants in the system (center).

Three features are mandatory. They describe the common parts of the electrical drive system in different cars. On the right, the features required for the configuration “efficiency” can be seen. The definition of the variant model and the configuration of the final products are entirely defined in pure::variants.

After the features have been modeled and configured in the variant management tool, the existing artifacts in the EA model can be annotated with these variant information as restrictions. pure::variants allows the formulation of restrictions or constraints in an integrated specification language. This can be seen in Fig. 9. After annotating the model elements in EA, pure::variants supports the automatic transformation from the 150% model into the 100% model by removing those modeling elements that do not satisfy a constraint. The final EA model is then generated and stored as an output project.

D. Enriched safety modeling

During the annotation process described in the last section, the definition of constraints does not only apply to the system’s architecture design, but also to the associated safety artifacts. This is necessary in order to integrate the right assets during the transformation process into the 100% model. In the following, we will show how the OSM copes with this scenario. Specifically, we will illustrate the 150% safety artifacts.

As part of the safety lifecycle, we first consider hazard and risk analysis (HaRa). In the OSM approach, HaRa is performed in two steps. In the first step, malfunctions of vehicle-level functions are examined and hazards are identified (see Fig. 10). These functions are traced to the logical model shown in Fig. 6, thanks to a standard UML deployment model.

In the second step, hazardous events are defined as a combination of hazards and operational situations and assessed as displayed in Fig. 11. Finally, safety goals are defined. In the current scenario, the identified hazard was “Undesired vehicle acceleration” and the safety goal was to avoid it. It is automatically updated during the transformation into the 100% model, depending on the variant that remains in the model.
variants are reflected in the HaRa through different hazardous events, due to the controllability parameter, which is more critical in the “dynamic” variant. This is due to the higher horse power that can be released in case of failures. Fig. 11 shows the 100% model of the risk assessment.

In order to investigate the causes of the hazard, modular safety analysis is performed. This is done using the OSM heterogeneous approach, in which CFTs, FMEAs, and Markov analysis can be integrated into a single model. To illustrate this, Fig. 12 shows the failure model of the E-Drive system and Fig. 13 depicts the 150% model of the Micro Controller, where both variants can be seen: one in the form of straight failure propagation from the input failure mode Phasecurrent: too high to the Torque Controller and the other in the form of a failure propagation path from the output failure port across the PhaseLimiter to the Torque Controller. The connection element (i.e., the failure propagation) as well as the failure model instance are annotated with the variant constraint of the PhaseLimiter. The latter hides its variant constraint in its realization (similar as in Fig. 9).

A typical analysis that could be performed over this model is the Minimal Cut Sets analysis (MCS), which computes the minimal combination of faults that could cause the hazard. This is represented by the top-most outgoing failure mode (i.e., a black triangle).

It should be noted that in a 150% model, the Minimal Cut Sets (MCS) analysis performs erroneously. This is due to a semantic error, circled in red in Fig. 13. This is not a problem of the method, as a 150% model, containing alternative elements, does not represent a concrete product. Therefore, the MCS have to be computed on the 100% model, which can be generated from the VRM.

The computation of the minimal cut sets for the heterogeneous model is performed on the basis of fault tree analysis. Initially, the modular failure models are translated into CFTs, which are later on integrated and transformed into individual fault trees for each selected top-most outgoing failure mode. In the current scenario, there is only one, but there could be more depending on the number of hazards being investigated. The failure models are displayed here from a black-box perspective, depicting only the propagation of failures between components. This is possible thanks to the OSM, which defines a common modular approach with standardized fault interfaces. Detailed fault logic propagation can be seen on the next lower abstraction level, which is not illustrated in this paper due to size restrictions.

Another step in the safety lifecycle is to build the safety concept. In the OSM, it can be modeled in association with the logical view, so that requirements are aggregated in the same logical structure. In this way, the logical view creates a reusable package at the component level, where components aggregate its safety requirements and failure models. Fig. 14 presents the safety concept of the Micro Controller. Here again, it is possible to see the annotated variants in the 150% model.

The transformation into a 100% model can be seen in Fig. 15, where the requirements module of the PhaseLimiter in the requirements module of the Micro Controller has been removed.
With the modeling of the safety concepts and their variants, we conclude this first case study regarding the integration of safety aspects and variant management. In the near future, we also plan to evaluate other OSM-supported methodologies, including safety context, safety cases, and safety contracts. The architecture of the SPES XT meta-model allows not only extensions for safety and variability as demonstrated in this case study; timing models can be integrated in the same way. A practical use case to facilitate further integration is the systematic analysis and allocation of fault tolerance time intervals in highly integrated, safety-relevant, and multi-variant driving functions.

Figure 14. Detail of a 150% technical safety concept model

Figure 15. Detail of a 100% technical safety concept

VII. DISCUSSION AND RELATED WORK

This section refers to the challenges described in section III. One by one, they will be used to evaluate the outlined approach.

Challenge 1. Modular composition

The presented OSM approach is, to the best knowledge of the authors, one of the broadest and most domain-independent approaches integrating safety artifacts, variant management, and model-based development in a modular way. Safety analysis as well as safety concepts can be performed modularly on the basis of the architecture design. This can be defined very flexibly since the SPES modeling framework, which forms the basis of the approach, allows working with different abstraction levels (as displayed in Fig. 5) as well as with different viewpoints. Component orientation is reflected in the safety analysis (Fig. 12 and Fig. 13) as well as in the safety concepts (Fig. 14 and Fig. 15). One drawback has been identified since the model is over-specified (i.e., 150%), semantic issues appear and analysis cannot be performed directly. Analysis currently requires transformation into the 100% model.

Challenge 2. Formality and traceability

Semi-formality is supported in the presented approach as given by the default UML level. The OSM approach defines the associations between artifacts and establishes proper traceability mechanisms. This is reflected in the failure models as well as in the safety concept models, for instance in the associations established between failure modes and logical ports, as well as with demands and guarantees (i.e., lollipop notation). These component-integrated approaches are known from [8], [14]. Examples of traceability are modeled in Fig. 13 and Fig. 15. In comparison, other approaches such as found in [1] present a semi-formalization view and a traceability dependency view applied throughout the different safety models. However, it is not clear whether such semi-formal descriptions are strong enough to support automated analysis. An alternative approach to traceability is presented in [5]. There, a framework is presented in which the dependencies among causes, design/architecture, and hazards are defined through a dependency matrix, which is used to trace backward to the functional change request in the design/architecture.

Challenge 3. Support normative requirements

According to SPES_XT [12], the OSM should be considered to be domain-independent. The automotive as well as the aerospace industry should be able to get along with its definition. However, this seems to have been achieved only partially. Although the architecture and the safety analysis may be domain-independent, there are still techniques that are domain-specific. This is the case, for instance, for the hazard and risk assessment of ISO 26262 and its aerospace counterpart, Functional Hazard Analysis. This has been demonstrated in the implementation of the I-SaFe tool, in which, depending on the selected domain, the definition of an assurance level for requirements and safety goals takes the form of either an ASIL or a DAL level. Other approaches have opted for a dedicated domain. This is the case in [1], where the presented safety engineering tool focuses on the fulfillment of the ISO 26262 requirements.

Challenge 4. Reuse capability and effort estimation

In most of the approaches we found in the literature [1], [3], it was not possible to clearly identify which safety-related assets can be reused in a modular way. In [4], [6], an approach for software fault tree product lines was defined in which it is only possible to reuse leaves of the fault tree. In the case study, it could be seen that the OSM approach is clearer and broader with respect to the artifacts that can be reused, since safety requirements as well as failure models are structured modularly according to the architecture design of the system. However, some techniques were not explicitly modularized.
Another very important benefit of these approaches is that they can now be modeled in OSM, instead of in a separate EXEL-sheet.

**Challenge 5: Requirement of tool support** The tool integration approach pursued in this work is lightweight and did not require major integration effort, given the fact that both tools are based on the EA platform. Ideally, one single tool user interface should exist, that aggregates all the functionalities. However, this is rather difficult in practice due to the legacy of very specialized tools, for example in the requirements specification area (e.g., DOORS). Most practitioners are nowadays forced to work with tool chains and hope for good integration. This drawback became clear during our case study: the EA, I-SafE, and pure::variants tools need to be obtained separately; training as well as dedicated support is needed to handle the tools and the model–based approach. Although not as modular as I-SafE, the approach presented in [1] also provides good tool support. Another very important benefit of these approaches is that they are scalable because variability models can be developed “on-the-job”. There is no unnecessary, a-priori variant modeling overhead.

**VIII. CONCLUSIONS AND FUTURE WORK**

The integrated approach presented in this work, although not completely evaluated for all parts of the automotive safety lifecycle yet, represents a promising approach for the reuse of variant-rich and safety-critical automotive functions. Although other approaches exist in the field, the modularity offered by the OSM is far more structured and it is easier to identify the reusable assets. Additionally, being based on UML, OSM can easily benefit from enhancements of regular UML tooling, like model differencing and change impact analysis. The annotation approach in which a 150% model is created has two primary disadvantages. First, rich variant systems become very complex. Second, analysis is restricted to 100% models. These issues might nevertheless be resolved by further development of the tools since this is not necessarily a methodological problem. Moreover, in the current implementation we identified some major challenges with respect to (semi-)automated consistency and completeness checks. Although they exist to a certain extent as part of I-SafE, they do not yet support the definition of variant constraints. Therefore, parts of the analysis can only be performed over a 100% model or have to be performed manually.

In the near future, we plan to implement and evaluate variant management support for safety model consistency and completeness checks ourselves. Regarding the safety perspective, we plan to evaluate the missing methods of the OSM: safety context, safety cases, and safety contracts. Moreover, we will elaborate and integrate rule-based impact analysis to synchronize changes throughout the different safety artifacts. Regarding the tooling support, we plan to look for alternatives to improve the usability of the 150% model approach. From the product line engineering perspective, we plan to evaluate modeling approaches with respect to the design work flow and tool support. We hope to provide methodological recommendations with respect to the construction steps required in the definition of the variant model and the realization of the safety analysis as well as to come up with precise metrics to measure the effort required in the construction and reuse of safety–critical components.

**ACKNOWLEDGMENT**

Partially funded by the projects CRYSTAL: ARTEMIS Innovation Pilot (AIPP N332830) and SPES XT of the German Federal Ministry of Education and Research (grant no. 01IS12005C).

**REFERENCES**


<table>
<thead>
<tr>
<th>Title:</th>
<th>Evaluation of a systematic approach in variant management for safety-critical systems development</th>
</tr>
</thead>
<tbody>
<tr>
<td>Date:</td>
<td>December 2015</td>
</tr>
<tr>
<td>Report:</td>
<td>IESE-075.15/E</td>
</tr>
<tr>
<td>Status:</td>
<td>Final</td>
</tr>
<tr>
<td>Distribution:</td>
<td>Public Unlimited</td>
</tr>
</tbody>
</table>