Technical Report

Functional Safety and Variability
Can it be brought together?

Michael Schulze
pure-systems GmbH
Agnetenstraße 14
39106 Magdeburg
Germany
michael.schulze@pure-systems.com

Jan Mauersberger
ikv++ technologies ag
Dessauer Strasse 28/29
10963 Berlin
Germany
mauersberger@ikv.de

Danilo Beuche
pure-systems GmbH
Agnetenstraße 14
39106 Magdeburg
Germany
danilo.beuche@pure-systems.com

ABSTRACT
Today's product development creates multiple products over time, often by using reuse strategies like "Clone and Own", leading to very inefficient reuse of artifacts in the long term since synergy effects between the products e.g. from testing cannot be utilized. Applying a product line approach with explicitly modeling the commonalities and variabilities of system artifacts and deriving products from that common base is a way to tackle the problem. High variant complexity can often be found in the development of embedded systems, which in turn often control safety critical functions. For these systems functional safety is a major concern not only since the ISO 26262 got relevant for the automotive industry. The arising question is: Can variability in functional safety related assets be treated in the same way as for other artifacts like requirements, models, and source code? In this paper we demonstrate on the example of two commercial tools and an automotive use case that from the technical/tool point of view safety related artifacts can be treated like other artifacts regarding variability. This means linking with variability information and visualizing as well as deriving of variants is feasible. This is a big step forward, because now not only ordinary artifacts but also functional safety related assets can be reused in the same way as other product line artifacts. However, we have identified and will discuss challenges with respect to variable safety analyses, regulations, and reuse of certifications, which need further research and elaboration, in this paper.

Categories and Subject Descriptors
D.2.9 [**Software Engineering**]: Management – *Life cycle*; D.2.13 [**Software Engineering**]: Reusable Software – *Reuse models;* J.2 [**Computer Applications**]: Physical Sciences and Engineering – *Engineering.*

Keywords
Functional-Safety, Variant Management, Tool Support.

1. INTRODUCTION
Traditional product development creates multiple products over time, often by using reuse strategies like "Clone and Own". With "Clone and Own" new projects start by copying/branching most/all of an existing product and start their development from there. It is cheap in the beginning but eventually costs are often not much lesser or even higher than for a systematic reuse approach which avoids copying. One of the reasons for this reuse "inefficiency" is that effort for activities as testing, maintenance, and certification (w.r.t. ISO 26262 [1] better confirmation measures, which include reviews, assessments, etc. e.g. for safety analysis, functional and technical safety concept) does not reduce significantly by copying the assets, as they have to be carried out for each of the clones individually.

Especially in development processes driven by the ISO 26262 and similar regulations such as EN 50128, the safety activities typically start with a hazard analysis based on the intended and functionality use of the system which might be different for individual variants already. The traceability and requirements coverage from the initial hazard analysis to the various safety concepts to the final development of hardware and software components need to be fully established. To support a variable design with safety constraints integrated measures are needed from the beginning. Of course, the scope of the hazard analysis could be extended to fit all possible use cases for all variants, resulting in an extremely expensive development, leading the idea of system re-use ad absurdum. If it is intended to use an item as a base-design for a product family, the possible effect of design variations to the safety activities need to be stated while performing the different verification and validation activities, to allow an exact determination of the influence of that variations. Another often seen approach for new products is combining functionality from different versions/branches of the system in an unmanaged ad-hoc kind which sometimes takes more time than developing them from scratch (which is what developers will do after a few failed attempts to reuse by merging) leading to same drawbacks as mentioned before.

To tackle the reuse inefficiency problem, firstly it is needed to separate changes over time (aka versions/evolution) from combining functionality into products (aka variants) and treat version management and variant management as different disciplines from which both are necessary. Secondly, a *product line* approach shall be applied where common shareable assets and variability-containing adaptable assets forming the base of all products often also termed as platform. Changes on that platform in order to support a new product have to be done in a way that already existing products can be created from the same asset

versions. Maintaining such flexible 150% solutions, from which a number of products each of them a 100% solution can be derived, over time, is much simpler than later trying to merge fixes across the different version of the assets used to build the individual products [2]. For sure, some of the variants might get irrelevant over time, so caring for them is stopped eventually.

At the moment we have not defined what we understand under variability. In [3] the term software variability is specified, however, we do not just consider software but also other artifacts like safety goals, etc. and as such we define variability following the definition in [3] but in a more general and broader sense: *Variability describes the ability of an artifact or of a system to be used in different contexts by changing or customizing some characteristics of it.* Those changeable characteristics are somewhere located within the artifacts and a notion that is usually used in all product line/variant management approaches is the term *variation point*. Variation points identify all places where members of a product line may differ from each other. Such difference may be the existence of certain model artifacts (optional or alternative artifacts) or *parameters*. A variation point describes all possible instantiations available at this point. Usually there are some conventions or even specific formal ways to express variation points. Furthermore, a variation point may have constraints, describing dependencies between instances of the variation point and to other variation points. They usually have a binding time such as compile time, link time or run time, at which the variation point instances have to be selected (the decision has to made).

To sum it up variation points describe where variations occur, what the choices are and how these variations are related to each other and very important is also that variation points in the different spaces are connected to each other.

The aim of variant management capabilities in the context of safety related artifacts is to enable modeling and specifying variable artifacts or parts of such artifacts in order to align them to the variable software or hardware generally speaking to the variable architecture. That means with our proposed approach it is possible to describe at which locations variability exists and under which conditions what variation can be applied. The benefit for the end user is that he can specify a specific architecture variant exploiting the captured variant knowledge and he will obtain related safety artifacts which are based on the architecture automatically. For example hazards will be there or not depending on which architecture element is there or not since hazards are based on functions that are provided by the architecture artifacts.

The contribution of the paper is twofold: Firstly, we show that from the technical point of view the integration of variability into functional safety models and deriving variant specific functional safety models can be done in the same way as it works for common artifacts like requirements, system architecture models, source code, tests, etc. Secondly, we have identified challenges that need to be addressed in further research to exploit the potential of that integration that can currently not be lifted. The rest of the paper is structured as follows. Next we describe the tool landscape and an airbag system example where functional safety and variability are relevant aspects. In Section 3 we present the chosen approach to combine these two followed by the application of the approach on the presented example. Subsequently we explain the identified challenges of this combination in Section 5. Related word is regarded in Section 6. Finally, that paper is summarized in Section 7.

2. TOOL LANDSCAPE AND SCENARIO

Experiences from the field have shown that an adequate management of variability that goes beyond “Clone and Own” is usually only feasible with appropriate tools. The same is for sure valid for functional safety analysis too. So in order to prove our assumptions we first have identified the necessary tools to technically apply the approach later.

The tool landscape in functional safety in general, and in particular in the automotive sector, is quite "clear". A few main tools are currently in use, most of them historically coming from the hardware sector and dedicated to specific safety analysis methods. With the new safety norm ISO26262 that was published December 2011 not only the complexity of the functional safety process has increased but also the number and the degree of requirements on the used tools themselves.

ISO 26262 is the actual standard for Functional Safety of automotive E/E (Electric/Electronic) systems. One of the mentioned challenges in the application of the standard is the alignment with modern model based development processes that put architecture and design models in the center of system engineering activities. A tight integration of safety analysis methods with the engineering activities is required. Preliminary and detailed safety analysis methods are commonly applied at the conceptual, system, component configuration and software levels in order to identify harmful behavior and to demonstrate the limitation of risks to a tolerable level. Hazard Analysis and Risk Assessment (HARA), Fault Tree Analysis (FTA), Failure Mode and Effects Analysis (FMEA), and hardware diagnostic coverage metrics are prominent examples for such safety analysis methods.

At present many Original Equipment Manufacturers (OEM) and suppliers as mentioned still use separate techniques and tools for the design specification and for the execution of the safety analyses. Even though tool support in general is widely accepted, there are still gaps between the tools in use (i.e. information cannot be exchanged automatically). Thus continuous tool chains are not yet available and some activities miss suitable tools at all. The typical application of office tools like Excel to overcome these shortcomings can only be considered as a temporary workaround as these tools neither fulfill the model based approach nor do they provide any native support for the varying safety analysis techniques. Duplication of data as well as a weak traceability between architecture and analyses is often the result.

The presented solution consists of a combination of the medini analyze tool [4], a tool for covering parts of ISO 26262 norm; and pure::variants [5], a variant management tool.

medini analyze is an integrated model based solution for all parts of the ISO 26262 norm. It approaches these problems by using the architecture model as a single source of information for design and safety analyses, enhancing the architecture model with information like failure rates and failure modes by the safety engineer and consequently doing safety analysis directly on the architecture model (see Figure 1).

Typically the architecture is the main source of variability, so at the latest when multiple variants exist in the system architecture, the question is interesting to see how a functional safety tool is affected by it and even more how the user of the tool – the functional safety engineer – can benefit from it. At the moment variant support is very poor or even does not exist in most tools, including medini analyze. However, the later comes with facilities to firstly exemplary introduce variant handling in several places and to secondly immediately evaluate and test the implications on

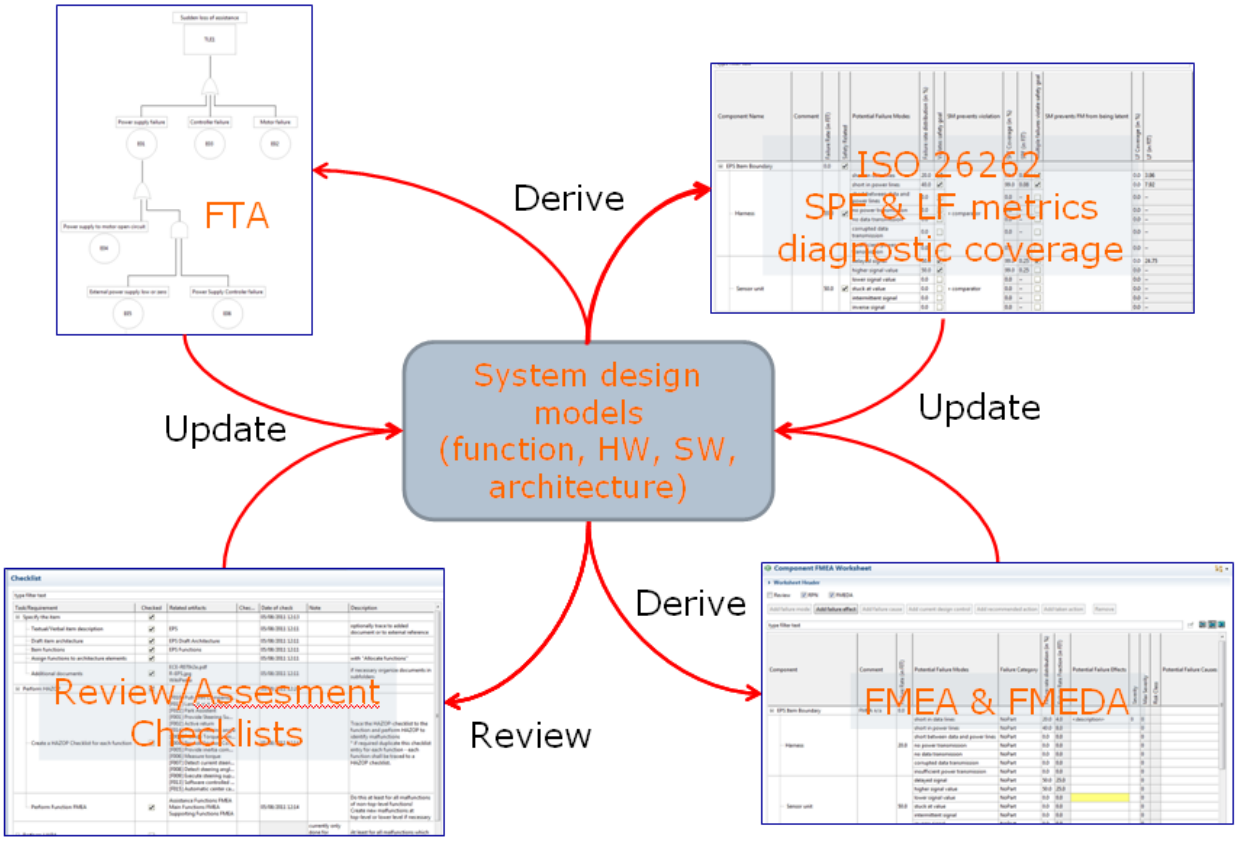

Figure 1 Integrated safety analysis techniques

most of the safety engineering activities, artifacts and work products

pure::variants [5] is a tool for efficient variant management supporting the whole life cycle of product lines starting from the requirements phase through design, architecture, and realization phase up to the testing phase. Due to its independence for example of programming languages or modeling tools it integrates nearly seamlessly into existing development processes.

The basic idea of pure::variants is the separation of concerns and as such the distinction between a problem domain that considers the variability of the system from an abstract point of view and a solution domain that deals with implementation artifacts like requirements, architecture models, source code, safety analysis, etc. pure::variants captures the problem space within feature models and the solutions with family models separately and independently in a structured way. Additionally, those models allow defining inter-relations and restrictions on the building artifacts that cannot be expressed by just structuring the information in the right manner. From the management perspective the models allow a uniform representation of all variabilities and commonalities of the products of a product line and the therein captured knowledge provides the basis to perform informed, comprehensible and valid decisions during the definition of variants.

In order to later prove our approach and to be able to achieve feasible results within a reasonable time, we did choose the Airbag – a well-known vehicle safety device – as a rather simple but adequate example system. Having its origin back in the middle of the last century the Airbag system is very long on the market and therefore well known, yet it was constantly improved and new variations were introduced over time, for example with weight sensors in the front passenger seat, ignition monitoring etc. which among other things do also contribute to new risks and to different estimations in the risk analysis. In addition the system is available in many different variations for the different car types and series even in one car maker. Same holds for the suppliers which typically deliver their Airbag system to different car makers or even cross domain for example to motorcycles or other vehicles. All in all it is a good example to study the implications of variability on functional safety and the possibilities to bring both together.

Modern cars may contain multiple Airbag modules but a simplified and sufficient architecture consists of a set of sensors (like accelerometers, impact sensors, and seat occupancy sensors), an airbag control unit (ACU) and a set of actual airbags that are inflated in case it is triggered by the ACU. So in this basic example we have variability in the number and type of sensors as well as in the number and type of airbags as visualized in Figure 2. The overall Airbag system has a set of main and supporting functions. The first includes the observation of the main crash detection sensors and the calculation of the ignition trigger, the later for example the observation of optional sensor values like the passenger weight.

All functions and their malfunctions have to be considered during the initial hazard and risk analysis to analyze severity, controllability and exposure of hazards that occur in certain driving situations and to finally get to a certain automotive safety integrity levels (ASIL) that more or less defines the objectives for all the further design and development activities. In our scenario some sensors and consequently the supporting functions based on

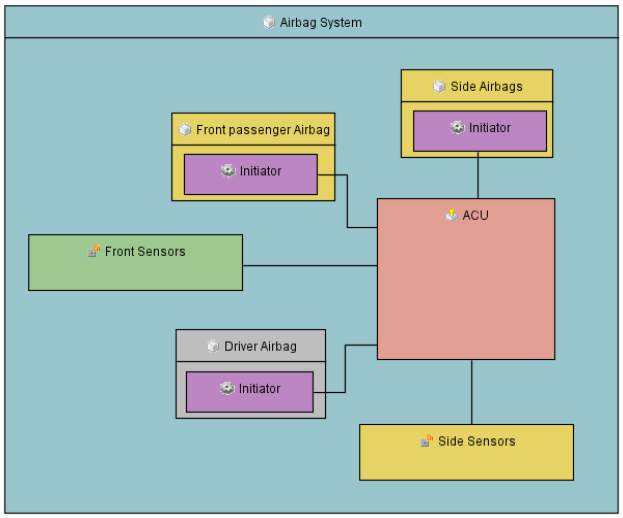

Figure 2 Simplified Airbag system architecture model with optional artifacts as Side Sensors, Front passenger Airbag, and Side Airbags (yellow parts)

those sensors are optional too, most likely bound to the same variation point as the system design artifact. Functions that aren't part of a certain variant do not have to be considered in the risk analysis, however, it may not be sufficient to have them optional – their presence or absence may have an effect to other functions and malfunctions, to risk classifications and thus the complete hazard and risk analysis. However, these kinds of challenges are tackled later in a separate chapter.

Besides the variability in the architecture and the used modeling language, it would be of great help to further model variability in follow-up safety activities, for example when deriving safety goals and requirements or doing qualitative and quantitative fault tree analyses. A safety requirement for example that claims a certain sensor must have a minimum reaction time or a minimum failure rate is only valid in case the sensor is really part of the variant. This kind of variability is directly linked to the variability in the architecture and can be modeled by the safety engineer before the derivation of a variant. There are many other opportunities fostered by the high degree of inter dependencies between the various safety artifacts as depicted in Figure 3.

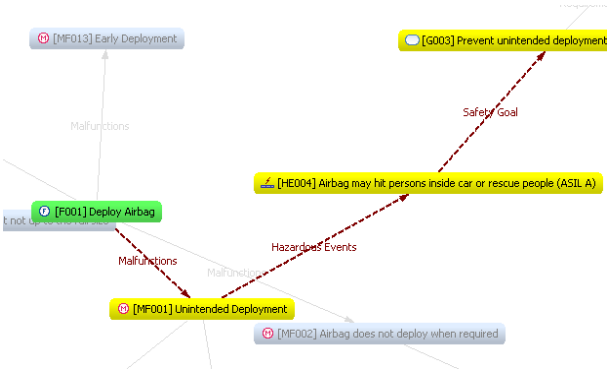

Figure 3 Excerpt from the dependency tree of safety analysis artifacts of the Airbag system

3. APPROACH

If the artifact's language does not provide basic means for the description of variations by itself as it is the case e.g. for FMEA and FTA, an approach needs to be taken that enables denoting variability on artifacts level. That can be done in two ways either by annotating artifacts or by referencing them. We have chosen the latter one the asset referencing approach, which links variant information like features from the problem domain with artifact from the solution domain, using an additional model, the referencing model (see Figure 4), for storing the linking information. That had two reasons. First reason: we need the additional model due to the fact that the solution domain models within medini analyze like SysML, FMEA or FTA models do not provide generic support for embedding additional information on artifacts level. That is, the data model is only extensible on project level and per artifact type separately but not on general scope. Second reason: the chosen approach supports different model types of the respective functional safety tool at once in a generic way and is minimally invasive, too. This is why it is well suited to be used in prototypical applications. However, these strengths come at a price: additional artifacts and the referencing model have their own life cycle, meaning one has to ensure that the information in all connected artifacts is consistent e.g. by synchronizing between the related models and using an appropriate configuration management approach.

Nevertheless, the asset referencing approach is always applicable for languages or tools or generally speaking for those Meta Models not having adequate support for variant handling mechanisms. That means, the Meta Model in question stays as it is and neither variation points nor conditions are integrated. However, that information is modeled in the referencing model. The variation points defined in the referencing model contain references to artifacts and existence conditions for them. If those conditions containing of a single feature or a feature expression are being evaluated in the context of a specific variant, the need for an instantiation of the respective model element is decidable meaning it will be part of the variant at the end. Since the tools that implement the respective Meta Model do not know anything about variability they are not able to perform the transformation from such variant-rich artifact model to a variant-specific artifact model through binding variation points by resolving the particular

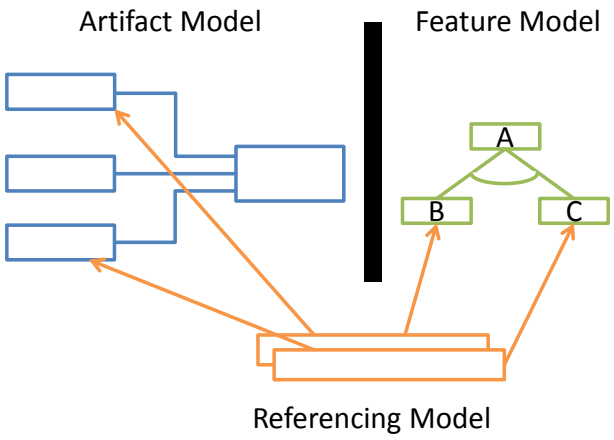

Figure 4 Artifacts are linked with variant management information via a referencing model.

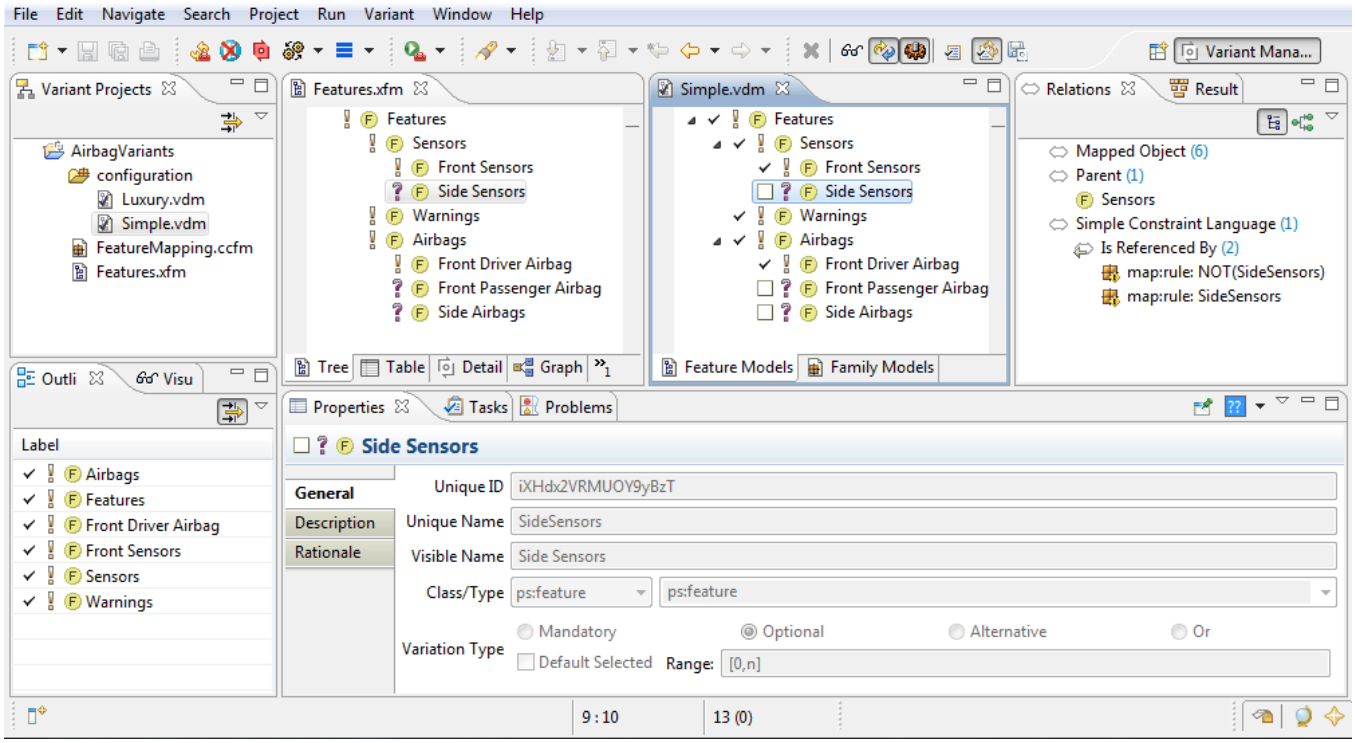

Figure 5 Airbag variants project with the variant management perspective of pure::variants.

conditions and selecting the necessary parts and removing the irrelevant ones.

Hence, from the perspective of the variant management two actions have to be carried in order to materialize specific variants:

- Variation points have to be bound with variations by resolving the variability for the referenced artifacts.
- Variant specific artifacts need to be created by transforming the 150% model into a 100% model through pruning unnecessary artifacts.
    - Either done by the variant management tool itself, meaning some knowledge about the specific meta model e.g. artifact dependencies has to exist,
    - or done by the modeling tool under guidance of the variant management tool, which delivers the information of to be pruned artifacts. Potential dependencies are resolved and needed erasing actions are done by the modeling tools.

Within pure::variants the generic EMF feature mapping extension works according to the asset referencing approach and as the models of medini analyze are based on EMF models, the connection of functional safety models and variant management is realized using this extension.

4. APPLICATION OF THE APPROACH

As a first step pure::variants and medini analyze were combined in one Eclipse application. Even though that kind of "live" integration is not absolutely necessary from a technical viewpoint, it perfectly fits the "integrated solution" approach of both tools and the opportunity that arose from having Eclipse and EMF as the common base for both platforms offers a seamless collaboration within the tool chain. Doing so, the problem of keeping feature models, variant models, domain models and referencing models in sync can be more easily solved. Having both tools connected in this way, it is possible to express variability in the system architecture, functional safety concept, safety analysis, etc. of the item under analysis as part of a single workflow.

In alignment with the chosen airbag example scenario, we analyzed the variability of the airbag system next and modeled it using the feature modeling capabilities of pure::variants. Since the example is indeed not complex, the resulting feature model depicted in Figure 5 contains just nine features at all. Six features are mandatory, which describe the common parts of all airbag system variants and three are optional that denote the really existing variability of our example. Overall, eight variants are configurable. Given that our example has very limited variability, the question might come up: Will what we presenting here scale for real application scenarios? Yes, our experiences from the field have been shown that the presented concepts scale very well for other artifact types and we assume the same will hold for safety related artifacts.

Right to the feature model in Figure 5, a variant configuration is visualized. The name of the variant is "Simple" and only the mandatory features are chosen, resulting in an airbag system that has just the driver airbag. The given screenshot shows further the properties of the currently selected feature "Side Sensor" (lower part of Figure 5) and the relations of that feature to other artifacts like parent element and where this is referenced (right side of Figure 5).

In a following step, the variation point modeling capabilities of pure::variants are used to connect features with e.g. architecture and functional safety concept artifacts of medini models using the generic EMF feature mapping extension. The extension works according to the asset referencing approach and the resulting

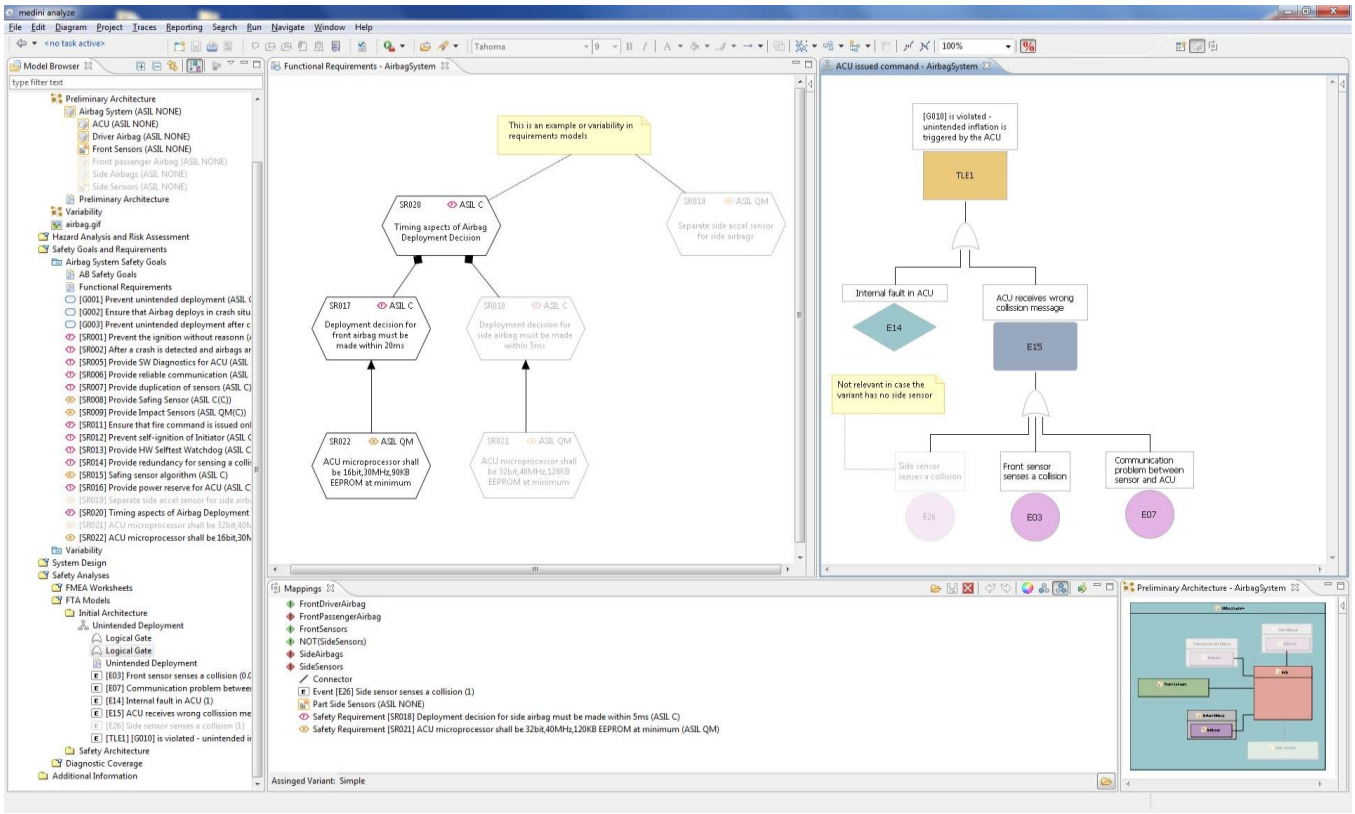

Figure 6 Medini analyze, pure::variants and the EMF feature mapping extension in a combined application supporting variability in functional safety concept, FTA and architecture models and user guidance

reference model "FeatureMapping.ccfm" (see Figure 5 upper left) that holds the linking information (feature expression and related model artifacts) is part of the airbag variant project. The end user gets support and guidance while performing connections through a special Mapping view (see Figure 6 lower center). The view provides a special editor for writing correct feature conditions that conform to the defined features and also enables to drag arbitrary EMF model artifacts and drop them on feature conditions to establish the connection. The view shows below each feature condition in hierarchical manner which model artifacts relate to that.

Beside the basic modeling functionality for linking features with medini model artifacts, the Mapping view provides three different visualization capabilities to help the safety engineer to easier recognize the implications of feature conditions. To activate a special visualization, a mode, symbolized through the three icons beginning with the color circle in the head of the Mapping view, has to be selected (see Figure 6 lower center).

For example feature conditions can get assigned a color and if the engineer activates the color circle mode, medini model artifacts will be colored according to the condition's color to which they are connected to. The mode next to the color circle on the right side greys out all artifacts that do not belong to the feature condition that is selected in this moment.

The last mode, the variant realization mode, works in contrast to the former not just on single feature conditions but instead on a configured variant, meaning on the evaluation of all feature conditions. For that it is required to assign a configured variant to the view. In Figure 6 the variant Simple is assigned to the Mapping View and Simple's configuration is shown in Figure 5. Depending on the variant configuration, the feature conditions have either green colored diamonds in case the condition evaluates to true or red otherwise. The activation of the variant realization mode results in graying out all medini model artifacts that are connected to feature conditions, which evaluate to false and as such the mode realizes a preview of a variant how it would look like if it would be instantiated. An important characteristic of the tight integration of medini analyze and pure::variants within one eclipse environment is that if the safety engineer does a live reconfiguration of a variant, all trees, views, and editors will be updated immediately to reflect the change and allow the engineer to get an impression on the change's impact.

Using this tooling infrastructure the safety analyst has several advantages at hand. He is able to take the variability in the underlying architecture into account when performing further safety analyses upon it. Because of the described features, it is possible to recognize variable parts of the architecture at all times. Safety analyzes for common parts of the architecture that will apply to all variants, may be created as far as they are possible and allowed. In addition, variability may also be introduced in a controlled way into the safety analyzes themselves. Subtrees in fault trees, as shown in Figure 6 on the right, as well as optional risks and hazardous events in the risk analysis are only some practical conceivable examples. However, the safety analyst must always be aware of the variant management and the implications of using it to derive variants of safety analysis, why such tool support is essential to tackle that task. However, currently there are not yet enough field experiences to know the limits of that approach within the legal scope of the regulations.

5. CHALLENGES

The benefits of product line development are usually a short time to market for new product variants, in general higher artifacts quality with overall lesser effort than having every time a single product development. These advantages persist due to the managed reuse of artifacts because synergy effects can be exploited given the variant knowledge captured in the variant management. Furthermore, treating functional safety artifacts in the same way as other artifacts delivers the same benefits for these, too. However, some challenges exist that are special to the functional safety topic and hence need to be considered accordingly.

Challenge 1: To the best knowledge of the authors the existing safety analyses e.g. FMEA or FTA do not know anything about variability and as such they are not able to work if a safety model contains variability. An open research question is whether the analyses can be extended to handle variability or whether we need new analyses, which addresses this issue.

Challenge 2: Reusing functional safety artifacts within a new context is possible but it is not just applying them because certification or confirmation measures as for ISO 26262 and the safety case argumentation cover not only the artifact itself but also its context. An exception to that is the safety element out of context (SEooC) to which for example a certified operating system counts. However, the SEooC concept cannot be used everywhere and therewith further concepts like a modular and composed safety case might be a promising way.

Challenge 3: A really major aspect is regulations through the certification bodies. They have to be convinced that for instance variable safety analyses and modular safety cases, if existent, are possible means to demonstrate and to argue the functional safety of a product variant. In the automotive case it has to be ensured that the approach is compliant with the relevant safety norms such as ISO 26262.

The three presented challenges are just the tip of the iceberg and we are sure that there exist lots more. However until these are clarified and answered the approach we have shown can be used. Using just the product line approach for all artifacts works even if functional safety is a topic but it requires the certification of each new product variant. In this respect, the last development steps regarding certification are not really different from single product development. Nevertheless, applying the product line development facilitates still a saving in time and money and solving the described challenges only increases the amount of that saving.

6. RELATED WORK

Related works which investigate combining functional safety and variability to increase the reuse potential is very limited. Most of this works are pure research papers where the tooling aspect is just a side note if any.

Goméz et al. describes in [8] an approach for reusing Component Fault Trees (CFT) of similar systems as input for the construction of the CFTs for new systems. At first it seems convincing but the procedure works according the described "clone and own" strategy, which we have discussed in the introduction and it has the discussed drawbacks.

The work [9] of Dehlinger and Lutz describes how software fault tree analysis can be adapted to be used for an entire product line and in [10] Dingding and Lutz extend that approach further to fault contribution trees analysis, where beside the software the hardware is considered too. Both approaches assume to have a common hazard analysis and each identified hazardous event is a root node of an individual tree. Below the root node a fault tree is constructed where variability can be integrated. However, a functional safety analysis can only be done on a specific product line member's fault trees and not for the entire product line. That means, it is a step forward but the challenges presented in Section 5 still exist.

Burton and Habermann show in [11] what the challenges of ISO 26262 are and how the automotive systems engineering can be aligned with it. In their position paper they discuss on a very abstract level what has to be done to bring product lines and functional safety together and the drawn conclusion is: Tool support is essential.

7. CONCLUSION

Customers want to have products that fit exactly their need to a reasonable price. Therefore tailoring products is usually done by the industry to satisfy this demand and to be profitable on the same side. In the automotive industry for example the effect is that the system development has to cope with variability on all abstraction levels and exploiting a product line approach in combination with variant management has appeared to be a promising way to tackle the variant complexity and to circumvent shortcomings of other approaches as e.g. inefficient reuse. Since December 2011 as the functional safety standard ISO 26262 was published, new challenges arose in general and especially if variability is involved since variability may influence not only arbitrary artifacts but also the functional safety of a product.

We have shown that from the technical point of view the integration of variability into functional safety models and deriving variant specific functional safety models can be done in the same way as it works for common artifacts like requirements, system architecture models, source code, tests, and so on. For that we integrated the functional safety tool medini analyze with the variant management tool suite pure::variants tightly within one eclipse environment to enable a lot of valuable use case for safety engineers. Besides the basic modeling work of both tools, it is now possible to provide sophisticated guidance and visualization support to the safety engineer while creating a 150% safety model and before the derivation of variant projects to execute the remaining safety analysis. Based on available variants and feature condition mappings, artifacts that are not relevant for a selected variant can be grayed out to support the safety analyst in doing the safety analysis like fault tree analysis providing already a significant visual improvement for the user. Since the functional safety analysis activities and the one defined in ISO 26262 for automotive in special are hardly automatable, guidance is a key issue for supporting tools.

Beside this technical feasibility study we identified three major challenges that are targeted to the functional safety topic with respect to reuse and variability and therefore they have to be considered in this interesting area of tension accordingly. In the future we will work on these challenges in research projects like SAFE [6] or OPENCOSS [7].

8. ACKNOWLEDGMENTS

This document is based on the SAFE and SAFE-E projects. SAFE is in the framework of the ITEA2, EUREKA cluster program Σ! 3674. The work has been funded by the German Ministry for Education and Research (BMBF) under the funding ID 01IS11019, and by the French Ministry of the Economy and Finance (DGCIS). SAFE-E is part of the Eurostars program,

which is powered by EUREKA and the European Community (ID 01|S1101). The work has been funded by the German Ministry of Education and Research (BMBF) and the Austrian research association (FFG) under the funding ID E!6095. The research leading to these results has received funding from the FP7 program under grant agreement n° 289011 (OPENCOSS). The responsibility for the content rests with the authors.

9. REFERENCES

[1] www.iso.org

[2] JEPSEN, Hans Peter; DALL, Jan Gaardsted; BEUCHE, Danilo. Minimally invasive migration to software product lines. In: *Software Product Line Conference, 2007. SPLC 2007. 11th International*. IEEE, 2007. S. 203-211.

[3] J. van Gurp, J. Bosch, and M. Svahnberg. On the Notion of Variability in Software Product Lines. In *2nd Working IEEE/IFIP Conference on Software Architecture (WICSA)*, 2001.

[4] http://www.ikv.de

[5] http://www.pure-systems.com/pure_variants.49+M5eb736ffe60.0.html

[6] http://www.safe-project.eu/

[7] http://www.opencoss-project.eu/

[8] Gómez, Carolina, Peter Liggesmeyer, and Ariane Sutor. "Variability management of safety and reliability models: an intermediate model towards systematic reuse of component fault trees." *Computer Safety, Reliability, and Security* (2010): 28-40.

[9] Dehlinger, Josh, and Robyn R. Lutz. "Software fault tree analysis for product lines." *High Assurance Systems Engineering, 2004. Proceedings. Eighth IEEE International Symposium on*. IEEE, 2004

[10] Dingding Lu., and Lutz, R. R. (2002). Fault contribution trees for product families. In *Software Reliability Engineering, 2002. ISSRE 2003. Proceedings. 13th International Symposium on* (pp. 231-242). IEEE.

[11] Burton, Simon, Habermann, Albert. “Automotive Systems Engineering und Functional Safety: The Way Forward.” *In: ERTS 2012 – Embedded Real Time Software and Systems*, Toulouse, France (February 2012)