Combining Safety Engineering and Product Line Engineering

Jean-Pascal Schwinn[1], Rasmus Adler[2], Sören Kemmann[2]

[1]CT RTC SYE DAM-DE
Siemens AG
Otto-Hahn-Ring 6
81739 München
jean-pascal.schwinn@siemens.com

[2]embedded systems quality assurance
Fraunhofer IESE
Fraunhoferplatz 1
67663 Kaiserslautern
rasmus.adler@iese.fraunhofer.de
sören.kemmann@iese.fhg.de

Abstract: Product line engineering and safety engineering for software address current challenges in the development of software-intensive, safety-critical embedded systems. The two engineering disciplines have different goals and the approaches for achieving these goals have been created independently from each other. For this reason traditional safety engineering methods do not fit to traditional methods for software product line engineering. The research project "Safe ReSA (Safe Reusable Safety Artifacts)" between the Fraunhofer IESE and Siemens AG has the goal to extend traditional safety engineering methods so that safety engineering can be applied to the reusable artifacts that are created in product line engineering. Sequentially, we present how we extended methods for analyzing cause-effect relation between failures, for developing a safety concept and a safety case. Additionally, we present lessons learned from industry projects and our tool for applying the extended methods to complex real world systems.

1 Introduction

Product Lines Engineering (PLE) is an important development paradigm allowing companies to realize order-of-magnitude improvements in time to market and other business drivers. All PLE approaches have at least three things in common. First, they try to maximize the benefits of reuse. Second, they try to support reuse by separating and modularizing development artifacts. Third, they use model-based methods in order to achieve a certain degree of formality in the modularization of development artifacts.

Safety PLE focuses on reuse and modularization of artifacts from safety engineering. The safety artifacts include safety analysis results, the safety concept and the safety case. Safety analysis results describe cause-effect relations between failures. They are the input for deriving a safety concept because the safety concept explains all safety measures for breaking cause-effect relations between failures. The safety concept is in turn the input for the safety case, i.e., a defendable argumentation why the system is safe. Due to the dependencies between the three types of safety artifacts, we started with the modularization of safety analyses. In a next step, we integrated the modular safety analyses into development artifacts in order to apply them in a PLE context. Afterwards, we modularized safety concepts in order to support reuse. In the ideal case, it is possible to reuse some "safety element out of context"-safety concepts in order to build a new safety concept. The modularization of the safety concept is the basis for the modularization of the safety case because the safety case is an extension of the safety concept. It extends the safety concept with some evidences for the correct implementation of safety measures. In order to get from a modular safety concept to a modular safety case, it is necessary to formalize evidences and to attach them to the safety concept. The formalization and modularization of safety cases is the basis for checking automatically whether a composition of components is safe in a certain context. For a system with a predefined limited context this could enable a generation of a safety case at design time. Further, it is a powerful means for assuring safety in spite of openness. New technologies enable Car2Car communication or other things that make it impossible to safeguard all possible interaction scenarios at design time. A formal modular safety case could generate at runtime safety assurances according to the concrete runtime situation. However, "safety element out of context"-safety concepts, modular safety cases and safety cases for open adaptive system belong so far to challenges of future work.

This article presents results from our project "Safe Reusable Safety Analysis and Arguments" ("Safe ReSA") which aims at a holistic approach for safety PLE. It explains the evolution of our safety PLE approach and our lessons learned from applying the approach in industry projects. It is structured as follows. First, it presents the modularization of the fault tree analysis technique. Second, it explains how we integrated the modularized fault trees into system models in order to apply the modular fault tree analysis in a PLE context. Third, it presents how we formalized and modularized safety concepts. Based on this, it discusses challenges concerning the formalization and modularization of safety cases. Finally, it concludes with our lessons learned from applying safety PLE in industry projects and summarizes the benefits from our work.

2 Modularization of fault tree analysis

A popular safety analysis technique in safety engineering is Fault Tree Analysis (FTA). The modularization of FTA started in 2003 [KLM03]. The modularized fault trees are called component fault trees (CFTs). A CFT describes the relation between some output events and some input events. The relation is modeled in the same way as a normal fault tree. CFTs can be composed by connecting the output events of one CFT with the input events of another CFT.

For implementing the CFT approach, we choose the UML-tool "MagicDraw" (MD, see [Ma13]) as modeling front end. MagicDraw supports the definition of domain specific modeling languages by adapting typical Unified Modeling Languages. We choose MagicDraw, because it is easy to change the modeling approach if it turns out that it does not fit to the capabilities of the modelers. In [ADH10], it is described in more detail how we integrated CFTs in MagicDraw.

Figure 1 shows a screenshot of a CFT in MagicDraw. The black triangles are output events. The yellow triangles are input events. If one goes into the CFT, the internal view shows how the two output events are caused by the input events and some internal events which are not visible from the external view.

For evaluating CFTs, we implemented interfaces to different calculation engines. For instance, we implemented a backend to the Siemens' proven-in-use calculation engine Zusim. This enables us to apply novel modeling concepts in industry projects. Additionally, we implemented an own calculation backend in order to add calculations. For instance, we added calculations for weighing minimal cutsets according to their criticality, for considering the systems' lifetime, for handling loops in models and for differentiating between several instances of one component and repeated elements. The enhancement of calculations was supported by the flexibility of MagicDraw, because it was easy to adapt the modeling language in cases where additional calculation input was required.

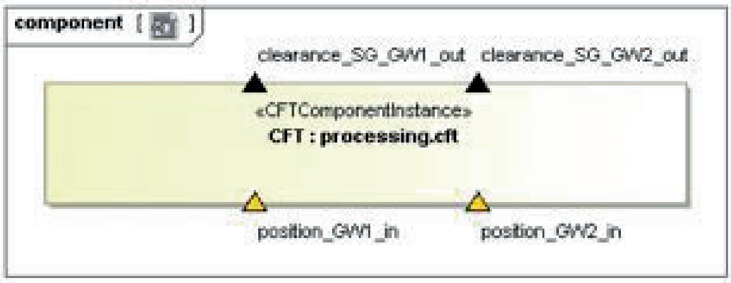

Figure 1: FTA component

3 Application of modular fault tree analysis in product line engineering

In industrial projects, we observed that modelers structured CFTs according to the system. A CFT typically referred to a function, a hardware component or a software component. For instance, the CFT in Figure 1 refers to the function *processing* in Figure 2. The two inputs events of the CFT refer to failure modes of the input signal *fct_proc_in* and the two outputs events refer to failure modes of the output signal *fct_proc_out*.

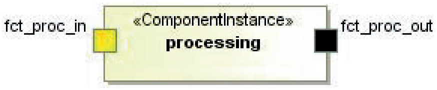

Figure 2: Functional component

The relation between a CFT and development artifacts helps to understand the CFTs and to keep them consistent to the system model. However, we observed that it was effort-intensive to keep the CFTs consistent to the system model. For this reason we integrated CFTs into the blocks of a block definition diagram. The blocks have input ports and output ports. As illustrated in Figure 3, the integration of CFTs was achieved by connecting every input event to exactly one input port and by connecting every output event to exactly one output port. Every event describes a failure mode of the signal that is exchanged via the port. The semantics of the signals depend on the semantics of the blocks. The blocks can represent functions, software components, or hardware components.

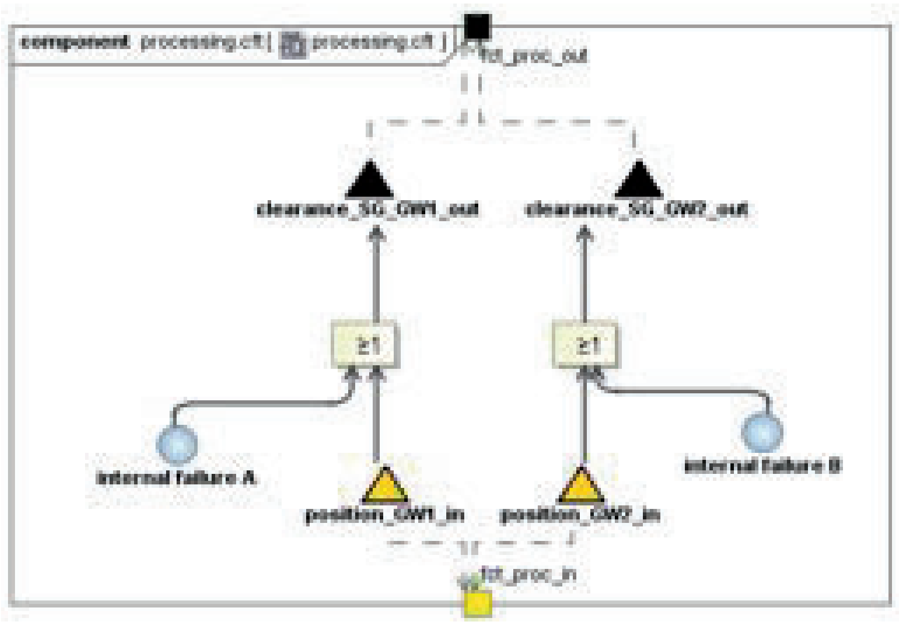

Figure 3: CFT of the functional

If blocks refer to functions then the block definition diagram explains how a composition of functions realizes some “features”, i.e., user-perceivable pieces of functionality. Features are an important aspect of PLE, because a product in a product line is often considered a set of features. In order to apply the Component-integrated Component Fault Trees (C^2FT) in a PLE context, we developed a C^2FT modeling method for modeling the cause-effect relation between failures of functions and failures of features [AKS12].

We applied the C^2FT approach in many different industry projects. The integration reduced the modeling effort because the information about the system structure does not have to be modeled. Once the components are defined, analysis about the propagation of failures can be based on the interfaces defined in the functional model. This method may not only present interactions that would not have been obvious without the functional model, but can also help identify assumptions of failure propagation that are incorrect since for example the needed interface to enable this kind of failure is not existing according to the model. The result can be a highly complex fault tree providing a rather detailed representation of the analyzed system. Because of the different layers of representations of the system, the complexity becomes easier to manage than if only a single fault tree was used. For this reason, the integration of CFTs contributes to the comprehension of the cause-effect relation between failures and to the consistency between the cause-effect relation and the system models.

4 Modularization of safety concepts

After solving safety PLE issues concerning FTA, we focused in the project SafeReSA on the concept of safety validation. Safety validation covers all aspects of the safety process. It starts with the definition of safety goals and ends with the verification showing that the goals are met. In addition, not only technical aspects are considered, but also the process of quality and safety management itself has to be documented. Usually, the documentation of the safety validation is realized with Word files and Excel lists. Plain text is suitable for arguing the appropriateness of process quality and safety management. It is, however, very complex to explain the implementation of the safety goals with plain text. For coping with this complexity, one can use the Goal Structuring Notation (GSN, see [ACC11]). The GSN makes it possible to model how safety goals are iteratively broken down to safety sub-goals. However, the GSN provides no means for modularizing a safety concept and for linking the modular safety concept parts to reusable elements of PLE.

The idea in SafeReSA was thus to adapt GSN in order to model modularly safety concepts of reusable elements. For this purpose we developed first a GSN-like notation called "Safety Concept Tree" (SCT) [DFK09]. The root of a SCT refers to a safety goal. The SCT describes how this top-level safety requirement is refined by other safety requirements. A refinement is modeled with two simple gates. The first one is the AND-gate. When a requirement A is refined with an AND-gate into a requirement B and a requirement C, this means that requirement A is fulfilled when requirements B and C are fulfilled. The second gate is called decomposition gate (D-gate). When a requirement A is refined with a D-gate into a requirement B and a requirement C, this means that requirement A is still fulfilled, even if either requirement B or requirement C is not fulfilled. The gate is called decomposition gate as it is used to model the decomposition of integrity levels. The idea behind the decomposition is to implement both refined requirements in order to create redundancy. As a consequence, the decomposed, redundant requirements can be implemented with a lower integrity level. In order to modularize a SCT, we introduced Safety Concept Parts (SCP). A SCP comprises guaranteed safety requirements (or simply guarantees), demanded safety requirements (or simply demands) and some Boolean gates for explaining the relation between guarantees and demands. Guarantees and demands of a SCP can refer to guarantees and demands of a component or a function. In this case it is desirable to have a formal link between the SCP and the development artifact. For this reason, we integrated the SCP into the blocks of a block definition diagram.

Figure 4 illustrates the SCP of the function *processing* in Figure 2. The SCP has two guarantees assuring that the output signal *fct_proc_out* has none of the two failure modes that are described in the CFT in Figure 1. The reason why the two guarantees are not collapsed to one guarantee is that they are provided with different integrity levels. As each failure mode of the output signal *fct_proc_out* can be caused by another failure mode of the input signal *fct_proc_in*, the SCT has two demands assuring that the input signal *fct_proc_out* has none of the two failure modes.

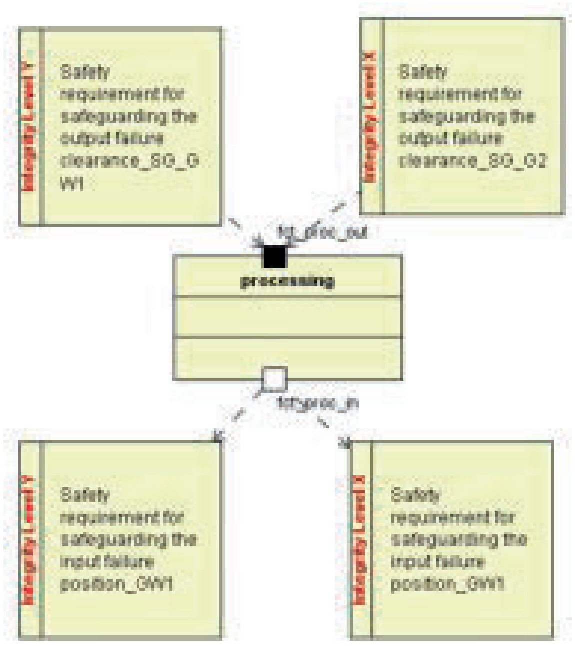
Integrity Level Y
Safety requirement for safeguarding the output failure clearance_SG_GW1
Integrity Level X
Safety requirement for safeguarding the output failure clearance_SG_G2
fct_proc_out
processing
fct_proc_in
Integrity Level Y
Safety requirement for safeguarding the input failure position_GW1
Integrity Level X
Safety requirement for safeguarding the input failure position_GW1

Figure 4: Component representation in safety concept view

We evaluated the modularized and component-integration safety concepts in several industry projects and observed several benefits. First, the modularization and component-integration contributes thus to the comprehensiveness of the safety concept. Even someone, who was not involved in the process of developing the safety validation at all, can capture its content and reasoning more quickly visually than by having to read through many pages of several documents. A second benefit is that it is easier to keep the safety concept consistent to the system models. Consistency is obviously supported by the component-integration because some inconsistencies are avoided by the modeling language. For instance, whenever a block is removed from the diagram its related SCT is automatically removed. However, even the normal SCTs which are not component-integrated contribute to consistency, because any kind of documentation can be referenced from a SCT (e.g. via a hyperlink). If someone works with a document or creates a new one, it can be linked to the SCT and everyone else will work with the linked document. Since there should be only one link, confusion about different versions of the same document can be greatly reduced. The next user working with the model will automatically use the latest link, without the need to make sure first, that he is working on the latest version of the document he expects. Since files can be linked to the SCT using hyperlinks, any type of data can be attached.

The mentioned benefit for better comprehension is of course not only true for people building the safety concept, but also for the experts that need to assess it (e.g. independent assessors, notified bodies, etc.). Specific questions can be answered directly without flipping through many pages of documentation. In addition, once a certain part of the validation is completed, it can already be assessed, even if the whole analysis is still lacking some inputs. Whether this different approach will be feasible will also depend on the assessor and his commitment to perform modular assessments.

5 Challenges

Our modularization and component-integration of fault trees and safety concepts provide a good basis for safety PLE. However, there are still some challenges that have to be resolved for developing a holistic safety PLE approach from our model-based modularization and component-integration of fault trees and safety concepts.

The first challenge is to optimize reuse of component-integrated component fault trees and component-integrated component safety concept trees. The difficulty of reuse arises from the fact that safety is a property of the overall system. If a component is reused for realizing a new feature of the system, it is not self-evident that its related fault tree and its related safety concept can be reused. In particular the reuse of the safety concept is strongly limited because safety requirements are more detailed than failure modes. For instance, a common failure mode for a signal is *too late*. The cause-effect relation between *too late* failures of input and output signals has reuse potential as it is not defined how much too late. Considering a safety requirement for safeguarding a *too late* failure, it is necessary to define precisely in which time frame the signal has to be provided by the component. In case that the guaranteed time frame is quite large, it is likely that the guarantee is not sufficient in a new system context. Consequently, the safety concept cannot be reused. It is often very complex to check which safety models can be reused and which have to be adapted or totally rebuilt. This complexity increases if reasoning is tied to certain constraints, which need to be fulfilled to ensure the consistency of the safety argumentation. The obvious procedure would be to take the component with its defined interfaces from the original safety validation, to transfer it into the new analysis and then to investigate, how the constraints fit in with the new validation. Depending on the amount of constraints, checking each of them one by one can lead to a rather high effort. This can result in a highly ineffective procedure since the outcome may be that none of them are relevant anymore. One approach to address this issue would be, to only allow a set of fixed interfaces, which are mandatory to all components used. While this would certainly ease integrating reused components into new analyses, it would at the same time decrease flexibility significantly and therefore limit new approaches of application.

The second challenge concerns the transition from a modular component-integrated reusable safety concept to a modular component-integrated reusable safety case. In the ideal case it is possible to extend a safety concept with evidences and to generate a safety case from the extended safety concept. However, it is even a challenge to come up with a semi-automated approach where only parts of the safety case are generated. Complete automation is the basis for the big future challenge to generate safety cases at runtime. This future challenge has to be solved in order to deal with the ever increasing openness of systems. Openness due to things like car2car-communication makes it impossible to consider all possible runtime situations at design time. Consequently, safety assurance has to be shifted from design time to runtime. The generation of safety cases at design time is a first step in this direction.

6 Conclusion

Addressing safety cases using a model based approach is a well established research topic, which has led to several publications in recent years (see e.g. [ADH10], [DFK09], [AKS12]). The following chapters describe some examples of new challenges faced and benefits seen putting the academic approaches into practical use.

6.1 Lessons learned

Putting the model driven safety case to practical use gave us additional input into the matter, raising new challenges:

Necessity for precise definition of safety goals: Fairly quickly it became clear that imprecise safety goals will complicate the modular approach for a safety case. The main reason for this is that it is often almost impossible to derive sub safety goals from a safety goal that does not precisely state conditions to be fulfilled. On the other hand, practical experience has shown that often safety goals tend to be formulated in a generic way, which often complicates proving its fulfillment. A paper addressing the aspect of the quality of safety goals is soon to be published. This also leads to the next observation:

Quality and readability of argumentation depends strongly on experience of the safety responsible: If the safety responsible has broad knowledge in the field of safety cases and years of experience to fully compose a safety case, Safe ReSA can be a good support to document the safe development. However, if the responsible is rather new to field of safety, he may finally come to the same conclusion, but it may take him longer to get there. However, the first experiences have also shown that although a model based approach for a safety case is a rather new concept, the learning curve is quite steep.

6.2 Benefits

Putting the tooling into practical use had immediate positive effect on one of the first projects, where it was applied:

Use of different calculation engines: The possibility to use different calculation engines enabled the possibility to verify the results of the analysis. This added greatly to the confidence in this new approach. Additionally it enabled safety experts to switch to the new tooling without the need to switch to a new, unknown and not yet certified calculation engine.

Systematic identification of needless measures: Due to the formal approach of modeling functionality, safety concept trees and CFTs, already planned measures could be identified, which were not needed to reach the safety goal and could get removed. Since these measures would have meant implementation of software on two different subsystems with intricate communication between the two of them, a significant development effort could be saved.

High acceptance with customers: All customers, who got in contact with the graphical representation of the safety cases composed for them embraced the new approach. According to them, grasping even complex matters by seeing them in a graphical display made understanding them much easier, in contrast to having to read the same information in textual form.

References

[KLM03] Bernhard Kaiser, Peter Liggesmeyer, and Oliver Mäckel. 2003. A new component concept for fault trees. In *Proceedings of the 8th Australian workshop on Safety critical systems and software - Volume 33* (SCS '03), Peter Lindsay and Tony Cant (Eds.), Vol. 33. Australian Computer Society, Inc., Darlinghurst, Australia, Australia, 37-46.

[ADH10] Adler, R.; Domis, D.; Höfig, K.; Kemmann, S.; Kuhn, T.; Schwinn, J.-P. and Trapp, M.: Integration of Component Fault Trees into the UML, In: Dingel, J.: Models in software engineering. Workshops and symposia at MODELS 2010. Reports and revised selected papers: Olso, Norway, October 3-8, 2010, pp 312-327.

[Ma13] Magicdraw homepage. URL: http://www.nomagic.com/. Last accessed on 2013/01/24.

[ACC11] Attwood, K.; Chinneck, P.; Clarke, M.; et al.: GSN COMMUNITY STANDARD VERSION 1, Origin Consulting (York) Limited, Nov. 2011.

[DFK09] Domis, D.; Förster, M.; Kemmann, S.; Trapp, M.: Safety Concept Trees, In: IEEE, Reliability and Maintainability Symposium, 2009. RAMS 2009. Annual, 26-29 Jan. 2009, pp 212-217.

[AKS12] Adler, R.; Kemmann, S.; Schwinn, P. and Liggesmeyer, P.: Model-based Development of a Safety Concept, In proceedings 11th International Probabilistic Safety Assessment and Management Conference and the Annual European Safety and Reliability Conference 2012, Helsinki, Finland, 25-29 June 2012, pp. 4200-4208.