embeddedworld2017
Exhibition&Conference
... it's a smarter world

Automating Safety Engineering with Model-Based Techniques

Juha-Pekka Tolvanen
MetaCase
Jyväskylä, Finland
jpt@metacase.com

***Abstract*— Fault Trees and Failure Models and Effects Analyses are well known methods in safety and reliability engineering. Their use, however, requires a considerable amount of work, in particular when the system evolves and grows. We describe an approach that automates parts of safety design flow. First, existing architecture models can be translated to dependability and error models. Safety engineers can then adapt the models for various safety cases and finally run analysis calling a suitable tool. We demonstrate the approach within automotive domain: System is specified with domain-specific languages and the created models are translated to analysis tools. This approach provides several benefits. It helps to ensure that safety analysis is done for the intended/designed architecture. It also makes safety analysis faster as it is partly automated, reduces error-prone routine work and makes safety analysis easier to use and accessible.**

Keywords—safety design; fault trees; failure models and effect analysis, functional safety; model-based design

I. INTRODUCTION

Fault Trees Analysis (FTA) and Failure Models and Effects Analysis (FMEA) are well known methods in safety and reliability engineering. In FTA [1] the system is describe from top-down to understand the logic leading to the top event having undesired state. In FMEA [2] then individual components and their failures are identified and the effect of a failure is then analyzed on system operations. Both approaches rely on understanding deeply the system analyzed and their use requires a considerable amount of work, in particular when the system evolves and grows. For example, creating failure trees or fishbone diagrams to specify events, faults and failures manually takes time and maintaining them along with the system specifications is demanding. This leads easily then companies to create safety models only to a few selected parts - those considered critical. Similarly it leads companies to perform safety analysis late, once the system is already almost finished, diminishing the input from safety design into the final product. The cost of creating safety analysis at early level, for different system design alternatives, or for performing analysis each time the system is updated is simply too high.

We present here an approach that aims to make safety engineering easier and cheaper to perform. Key elements of the approach are:

- Existing systems designs can be translated to models applicable for safety design. This not only makes the move from system design to safety design easier but better ensures that what is analyzed for safety is also what is designed. Lowering the effort to start safety design also enables earlier acknowledgement of safety issues and allows them to influence the system design at a stage when it is cheaper to make changes.
- Safety models directly express issues relevant to safety design, such as hazards, safety goals, costs and failure rates. For example, in the case of ISO 26262 for functional safety in automotive applications, the modeling language directly supports standard concepts like Item, Hazard and ASIL levels. This makes safety design and safety standards directly visible to all participants, and also allows checking the completeness of safety design. Contrast this with traditional specification languages like SysML or UML: these do not even recognize safety issues, so model-based safety design becomes hard if not impossible.
- Models based on safety concepts are applied directly for automated safety analysis linked with analytical tools. This allows you to run alternative scenarios, removes tedious manual analysis work, and generally makes the analysis process faster and easier.

We demonstrate the approach with an example from the automotive domain: The safety design of a power window controller is performed following ISO 26262, and error models are created for automated safety analysis. For both system modeling and safety modeling we use the EAST-ADL language [3] in the MetaEdit+ modeling tool [4]. The models are then passed to the HiP-HOPS tool [5] for automated safety analysis, providing Fault Tree Analysis (FTA) and Failure Mode and Effects Analysis (FMEA). The approach is not limited to any particular modeling language or tool.

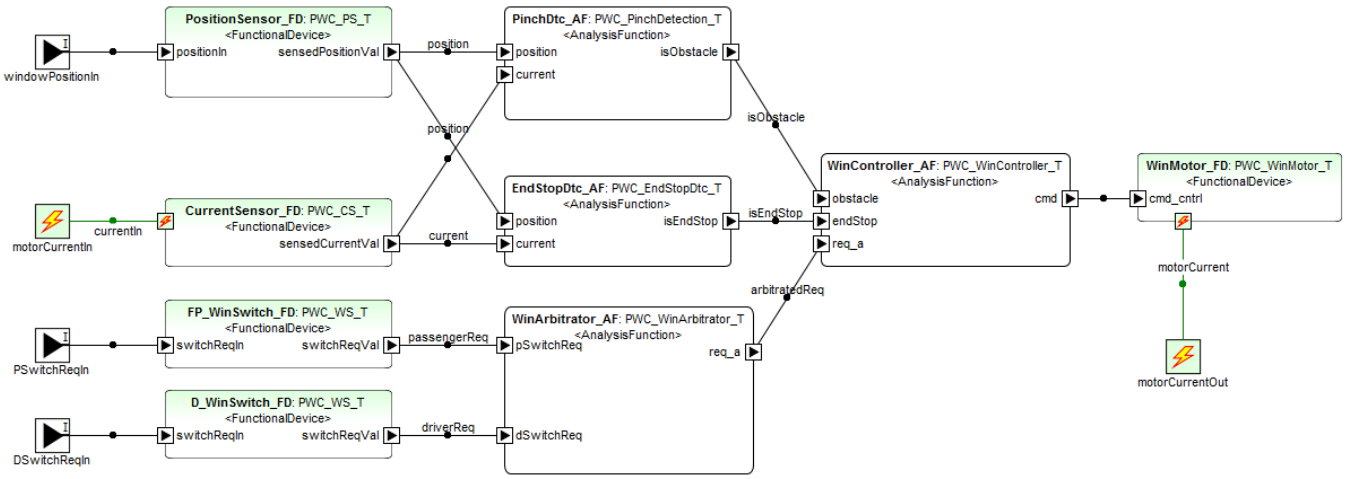

Fig 1. Functional architecture of the PowerWindow controller

II. Utilizing System Specifications For Safety Engineering

Safety standards, like ISO 26262, IEC 61508 and ISO 13849-1, emphasize the use of models in safety design and management. Design models, however, are not directly suitable for safety design as they do not identify and capture typical safety aspects like faults, failures or failure rates. Design models also include information that is not relevant for safety work, adding unnecessary complexity for safety engineers. Consider for example the small system illustrated in Figure 1. It shows the functional architecture of a PowerWindow controller: its functions, ports and connections among them. Functions are classified, have a more detailed internal hierarchical structure, and their ports have interfaces and data types. Some of these aspects, like classification of functions or data type definitions, are not directly relevant in safety design, but are necessary to generate the implementation, like AUTOSAR, Simulink models, configuration etc.

Adding safety-related information, like faults or failure logic, directly to the design model is often not practical because it would quickly make the model complex with too many aspects. For safety analysis, unlike design, we also usually need several models covering different analysis scenarios. One solution is to generate the initial safety models from design specifications. For example, Figure 2 shows the result of such a transformation: an initial error model produced from the system design shown in Figure 1. The error model is made for more detailed safety analysis and includes the same system architecture but focusing only on failures, faults, and error types. Aspects needed for safety engineering are then added to this model, e.g. a FailureOut port is added to analyze an error on obstacle detection (see Section 4 for more details).

This kind of model transformation makes the move from system design to safety design easier [6]. It also better ensures that what is analyzed for safety is directly related to what is designed. Lowering the costs for starting safety engineering also enables faster feedback to system development. For safety engineers the creation of specifications becomes easier, as models focus only on safety characteristics such as errors, failures etc.

For any realistic system, the size of the error model is larger than a single diagram like Figure 2. For this case the generated error models covers further details, like the detailed structure of functions or ports and error logic. Figure 3 shows such an example: it describes a portion of the model hierarchy illustrating the error model for pinch detection (PinchDTC_AF in Figure 2) specifying the Boolean logic. In this case the omission failure for detecting an obstacle is defined to be caused by a fault in the ‘position’ or ‘current’ input values, or an internal process error that fails to update the variable.

During safety analysis the automatically produced error logic can be extended with further details, or it can be used as it is, e.g. for producing FTA and FMEA (see Section 4). The automatically generated error models may also include error logic that is generated based on selected patterns, like providing default omission cases as internal failures for functions which do not receive input from outside. Safety models can also include information on component specific failure rates, repair rates, costs of the components etc. as needed for the safety analysis.

Often the exact formalism for the specification of failure logic is chosen according to the analysis methods of interest and naturally on the analysis tool. However, rather than specifying error behavior with certain syntax of a specific analysis tool, often better is to create a specification in a format that can be then translated to the analysis tool specific format. For example, in MetaEdit+ it is possible to enter directly the format of HiP-HOPS or use a more generic specification language (see Figure 3) that is translated to analysis tool specific format (here HiP-HOPS). The benefit of this approach is that different analysis tools can be applied to the same safety models – rather than recreate them for each analysis tool.

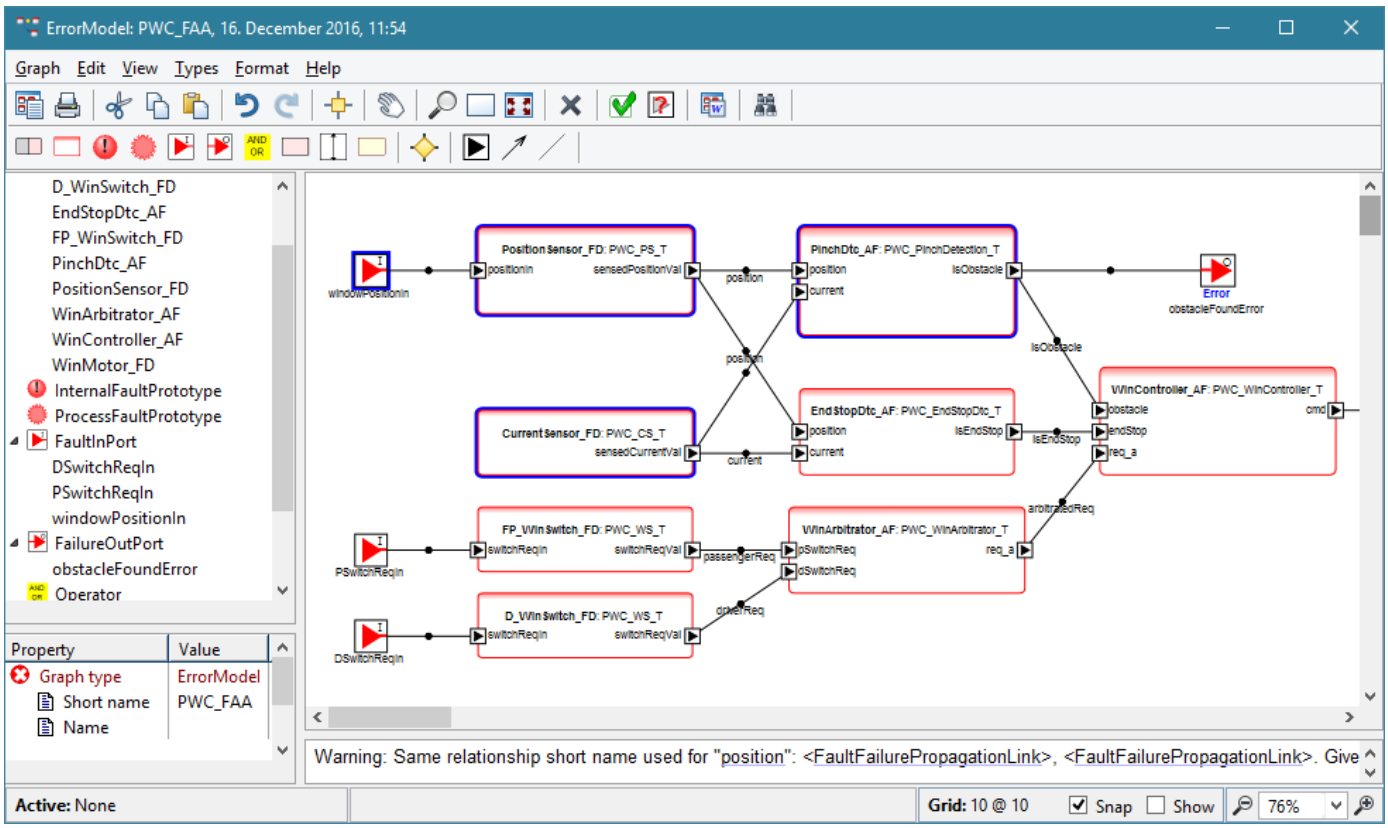

Fig 2. Automatically produced error model for safety engineering

To enable trace and links to the system engineering phase, the generated error models may also include references to the requirements and original system design. If system models include links to requirements then also automatically produces error models can provide the equivalent links. This way safety engineers can reason on why certain components and related functionality is needed in the first place.

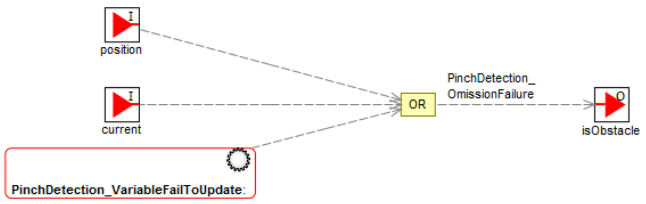

Fig. 3. Error model with error logic for Pinch Detector in PowerWindow controller

III. Apply Safety Concepts Directly in Model-Based Specifications

Since safety standards emphasize the use of a model-based approach, an obvious way would be to apply safety concepts directly as modeling concepts. This is in analogy to using relevant design concepts in system design languages. For example, according to the ISO 26262 functional safety standard, the safety engineering should consider various hazards, safety cases, ASIL levels, failures etc. With domain-specific languages, such as EAST-ADL for automotive systems, these very same concepts are directly supported in the language, guiding the engineer in making safety models. Figure 4 illustrates the safety design flow with an example from the Braking System.

This dependability model of EAST-ADL is directly based on ISO 26262 concepts. At the top is an Item for ‘Basic Braking’, with a FeatureFlaw on ‘Unwanted deactivation’: Hazardous event shows the assessed risk in terms of severity of injuries, exposure and controllability following the classification from Automotive Safety Integrity Level (ASIL). In accordance with ISO 26262, SafetyGoals are defined for the hazardous events aiming to reduce the risk to a tolerable level.

SafetyGoals are also related to requirements that can be linked with the system architecture and its individual components, supporting traceability and validation. In terms of error modeling, the specified feature flaws are analyzed along with the specification of the faults and fault propagations.

In terms of error modeling, the ‘L2_AA’ error model element in Figure 4 contains the error model been originally generated from the nominal architecture been models like show in Figures 2 & 3. In this way the initially created error models can be linked for different cases of hazard analysis. To give a better indication on the size of error models and their complexity Figure 5 shows a high-level view to the error model of braking system – also been originally generated from the nominal architecture model.

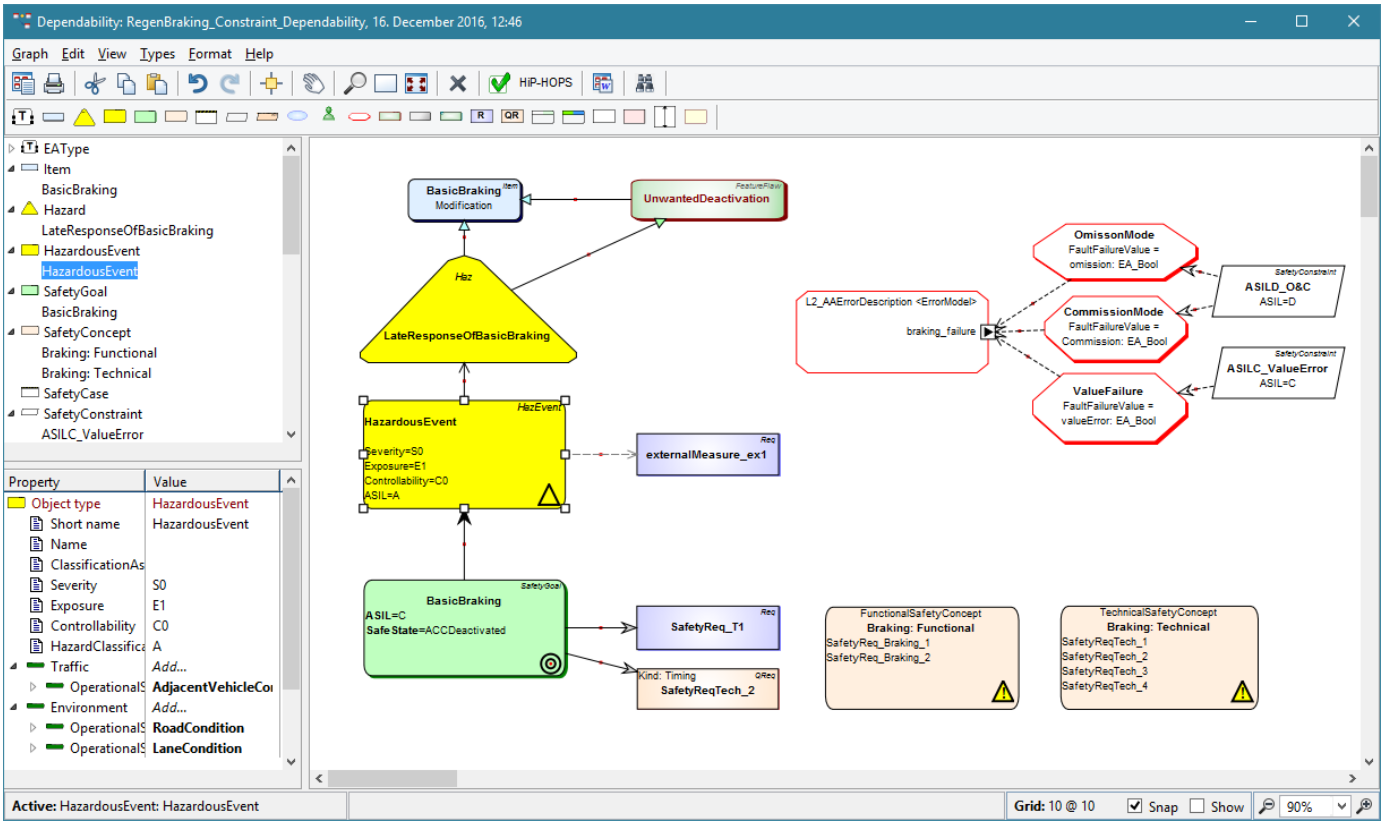

Fig 4. Dependability model structuring information related to a safety goal.

It is notable that the modeling language directly provides the ISO 26262 concepts for safety engineers, and treats safety aspects as first class citizens. Not only is the vocabulary of functional safety applied, but also the safety models are checked on their consistency and completeness. For example, each HazardousEvent is usually a result of some use case.

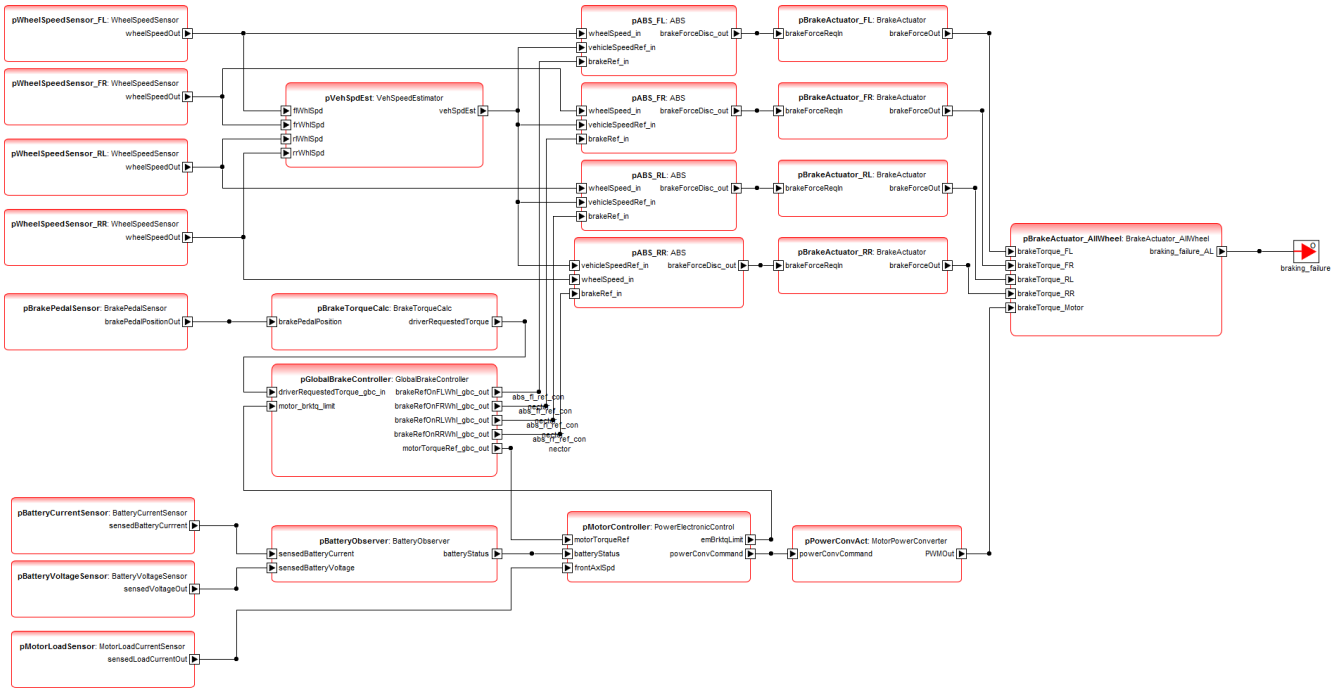

Fig 5. Error model defining the faults and error propagations within a braking system.

IV. LINKING WITH ANALYTICAL TOOLS

Manual safety analysis, such as creating fault trees or fishbone diagrams and performing related analyses, easily becomes error prone and tedious when applied to larger systems. This is simply because such systems have a vast amount of alternatives, so exploring them manually is no longer feasible. This challenge is multiplied further in change situations, as the analysis should be carried out again when the system specification changes. Often companies then limit their focus on safety, concentrating only on a few parts that are considered most critical. All this calls for increasing the use of automation in safety engineering.

Models created during safety design, like those illustrated in Figures 2–5, can be used for automating safety analysis with analytical tools. One approach is to trace failures within the system by inspecting the possible causes for a particular point of failure. Figure 2 illustrates this analysis by showing parts of the system annotated in blue from the point of failure detected (ObstacleFoundError at the top right).

Another approach is an automatic synthesis of fault trees and performing Failure Models and Effects Analyses (FMEAs) based on the safety data given in error models. Figure 6 shows the results of such automated analysis with the HiP-HOPS tool. The error model of PowerWindow is exported to the analysis tool, which then takes each failure and calculates the possible causes based on the structure and logic specified in the error model. From the point of view of the safety engineer this is fully automated: pressing the 'HiP-HOPS' button in the modeling tool (Figure 4) starts the analysis process, resulting in the analysis report showing FTAs and FMEA results with minimum cut sets. The step from safety model to analyzed results takes only a few seconds, making it fast and cheap to perform – especially in cases where the system updates and failure analysis need to be updated too.

Figure 6 gives a small example for the case illustrated in the error models (Figures 2 and 3). It shows the portion of the fault tree for the top level failure on Obstacle Detection and related cut sets. The fault tree can be further studied by inspecting details when opening the gate symbols (e.g. the position of pinch detection in our example). For any realistic size system the automated analysis quickly becomes an invaluable tool.

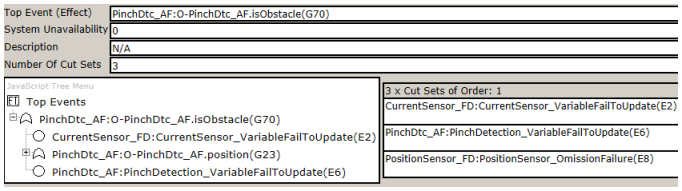

Fig. 6. Automatically produced fault tree and calculated cut sets.

Figure 7 illustrates the FTA been generated from the larger error model related to the braking system (Figure 5). Figure 8 then shows the results of FMEA been produced by HiP-HOPS. It is also based on the same error model of the braking system.

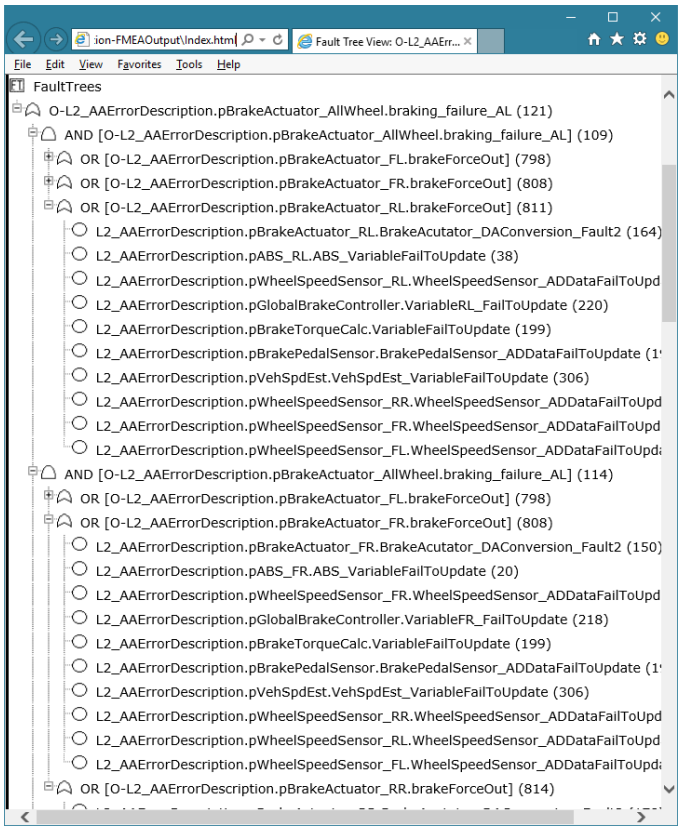

Fig. 7. One of the fault tree and some cut setsgenerated by HiP-HOPS

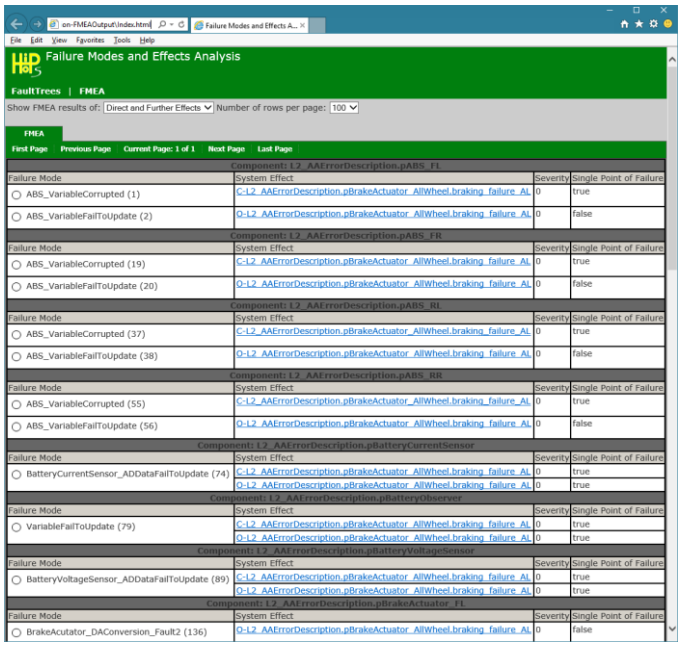

Fig. 8. FMEA table generated by HiP-HOPS according to the EAST-ADL error specifications

Depending on the tools used the analysis can be further extended by providing the cost data for each element type in the error model, and by giving information on failure rates and repair rates.

V. CONCLUSIONS

Safety engineering can greatly benefit from a model-based approach. We have presented an approach in which safety vocabulary and related guidelines are directly part of the specification language. While we illustrated the approach with an automotive example, with EAST-ADL applying directly ISO 26262, we have also experiences targeting other analytical tools and safety standards (e.g. Sistema). They require similarly having a domain-specific language that directly applies the safety concepts, along with connections with the desired analysis tool.

With the model-based approach presented, the specifications can be used for automating safety analysis tasks, like creating fault trees and performing FMEA analysis. To support the move towards using models in safety design, we also applied model transformations to automatically produce initial error models from standard system and software models. The approach presented provides several benefits, such as:

- Ensures that safety analysis is done for the intended/designed architecture
- Makes safety analysis faster as it is largely automated
- Feedback from safety analysis improves: results can be used earlier
- Reduces error-prone tasks

Currently reported experiences by others show that the presented approach is very effective [6]: The modeling provides a user-friendly approach to specification and management of safety concerns. The translation to external analytical models as well as the analysis by HiP-HOPS is automatic and efficient. For a case containing 23 error model instances and almost 100 error behavior descriptions, the generation of corresponding HiP-HOPS input file by MetaEdit+ takes less than one second in a PC with Windows 7, Intel Core i7-4800MQ and 16 GB memory [6]. Clearly, safety engineering can benefit from automation.

ACKNOWLEDGMENT

For the development of the presented approach and providing feedback on its use we want to thank Stefano Cerchio, Carlo LaTorre, DeJiu Chen, Frank Hagl, Henrik Lönn, Luis P Azevedo, Mark – Oliver Reiser, Martin Walker, Renato Librino, Sandra Torchiaro, Sara Tucci-Piergiovanni and Tahir Naseer Qureshi.

REFERENCES

[1] Lee, W. S., Grosh, D. L., Tillman, F. A., Lie C. H., Fault Tree Analysis, Methods, and Applications - A Review. IEEE Transactions on Reliability, Volume: R-34, Issue: 3, Aug. 1985.

[2] Reifer, D., Software Failure Modes and Effects Analysis, IEEE Transactions on Reliability, Volume: R-28, Issue: 3, 1979.

[3] EAST-ADL association, http://east-adl.info/ (accessed Dec 2016)

[4] MetaEdit+, http://www.metacase.com (accessed Dec 2016)

[5] HiP-HOPS, hip-hops.eu/ (accessed Dec 2016)

[6] Maenad project, Case study analysis and safety assessment, Deliverable D6.1.3, 2014 (http://cordis.europa.eu/docs/projects/cnect/7/260057/080/deliverables/001-D613V10.pdf)