Model based Safety Analysis using SysML with Automatic Generation of FTA and FMEA Artifacts

Gaëlle Girard
ROSAS (Robust and Safe Systems) Center / HES-SO, University of Applied Sciences and Arts of Western Switzerland , Fribourg, Switzerland. E-mail: Gaelle.Girard@hefr.ch

Ivan Baeriswyl
ROSAS (Robust and Safe Systems) Center / HES-SO, University of Applied Sciences and Arts of Western Switzerland , Fribourg, Switzerland. E-mail: Ivan.Baeriswyl@hefr.ch

Jonathan James Hendriks
ROSAS (Robust and Safe Systems) Center / HES-SO, University of Applied Sciences and Arts of Western Switzerland , Fribourg, Switzerland. E-mail: JonathanJames.Hendriks@hefr.ch

Roland Scherwey
ROSAS (Robust and Safe Systems) Center / HES-SO, University of Applied Sciences and Arts of Western Switzerland , Fribourg, Switzerland. E-mail: Roland.Scherwey@hefr.ch

Christian Müller
Department of Computer Science / Ulm University of Applied Sciences, Ulm, Germany. E-mail: Christian.Mueller@thu.de

Philipp Hönig
Department of Computer Science / Ulm University of Applied Sciences, Ulm, Germany. E-mail: Philipp.Hoenig@thu.de

Rüdiger Lunde
Department of Computer Science / Ulm University of Applied Sciences, Ulm, Germany. E-mail: Ruediger.Lunde@thu.de

Today's technical systems complexity is increasing in most industrial domains. In combination with the rapid increase in safety regulations and standards, it is therefore necessary to increase the scope and intensity of Safety Assessments of technical systems. Safety Assessment methods like *Failure Mode and Effects Analysis* (FMEA) as well as *Fault Tree Analysis* (FTA) are typically applied in a classical manual form which are not easily reproducible and difficult to update in case of design changes.

To meet these challenges, an automated workflow was developed that combines *Model Based System Engineering* (MBSE) and *Model Based Safety Analysis* (MBSA). This workflow includes the qualitative modelling of the system with its formalized requirements, the nominal and failure behavior using *SysML* (Systems Modelling Language) as a platform. The model is then automatically passed to a tool called *smartIflow Workbench*, which allows an extensive simulation and deep analysis of the nominal and failure behavior of the system based on model-checking techniques and temporal logic for requirement specification. Finally, this workflow allows to automatically generate the FMEA and FTA safety artifacts and such enables to verify in a reproducible way, the critical and in many cases safety-related design aspects of complex technical systems in the preliminary *MIL* (Model in the Loop) concept phase.

Using an example system, the proposed workflow is explained and the automatically generated FMEA and FTA results are compared with those of the classic manual methods.

Proceedings of the 30th European Safety and Reliability Conference and the 15th Probabilistic Safety Assessment and Management Conference
Edited by Piero Baraldi, Francesco Di Maio and Enrico Zio
Copyright © ESREL2020-PSAM15 Organizers.*Published by* Research Publishing, Singapore.
ISBN: 978-981-14-8593-0; doi:10.3850/978-981-14-8593-0

1. Introduction

"*Why model?*" Many industrialists in different fields have asked this question. By introducing modelling into their company, communication between engineers has improved while the complexity of the model increases, making it impossible to remain with traditional methods. Indeed, the development of complex and particularly safety-critical systems require a multidisciplinary collaboration between system engineering departments. Safety equipment development is driven by requirements, and all must be verifiable.

One of the key challenges is to link Model-Based Systems Engineering (MBSE) and Model-Based Safety Analysis (MBSA) in order to combine systems engineering and safety analysis. A partially automated modelling workflow is necessary which allows the coverage of these multidisciplinary aspects in systems engineering and safety analysis. This workflow will enable the development of a set of consistent high-level models. It allows the checking of the system and the safety requirements early in the "V" life-cycle phase and to generate some important safety analysis artifacts documents, e.g. Fault Tree Analysis and Failure Mode and Effect Analysis.

2. State of the Art

2.1 *Model-Based Systems Engineering*

The MBSE is an emerging and powerful methodology for the design of complex systems. The principle of MBSE is to have a system model used as a single source for the engineers and customers to describe the evolution of system design throughout the product life cycle. By aligning people, processes and technology around a single product vision, MBSE promises to significantly reduce the development cost and risks associated with complex systems. One challenge is to reconnect the system engineers and the various technical engineers to enable MBSE to reach its full potential. In addition, this methodology improves the traditional document-centric systems engineering with a model-centric approach using a system modelling language such as SysML, a graphical modelling language that was created as an international standard to support MBSE. The advantages of using MBSE are manifold:

- Validate that the right system is built
- Enable efficient system component building by 3rd parties/outsourcing
- Reduce errors and costs early in the lifecycle
- Accelerate time to market
- Ensure system implementation is correct and reliable at a high-quality level

2.2 *Safety Assessments*

Verification of safety requirements is one important task during the development of safety critical systems. Nowadays the system safety assessment shall be taken in consideration as a required business process. It should be integrated into the product development processes framework of companies. The functional safety is an important part in the safety critical product development, it is becoming more and more significant due to increasing complexity and variability with the wide employment of electrical, electronic components as well as embedded systems. In order to ensure functional safety, several industry sectors have developed their own functional safety standards, which define the functional safety requirements and life-cycle management for the safety-critical components in different phases of development process.

Safety analysis methods like Fault Tree Analysis (FTA) and Failure Mode and Effect Analysis (FMEA) (see Table 1) are already mandatory for performing safety assessment within automotive (ISO 26262) and aviation (ARP4761, RTCA DO 178C/254) industries, and other industries will follow.

Table 1. Comparison between FTA and FMEA

FTA	FMEA
Top-Down-Approach	Bottom-Up-Approach
Deductive analysis	Inductive Analysis
Tries to identify for a critical system failure all possible causes (including combinations of multiple independent faults)	Tries to identify for each possible component failure mode all possible effects on system level
Provides detailed information for the critical system failure including failure probability/failure rate.	Provides useful data to assess the criticality of all component failure modes. Caveat: only single faults are analysed.
Analysis of critical top events of the examined system	Examination of the whole system in individual elements and whose error types
What are the causes of the top events? (and only for these)	What consequences result in the default case? (not limited to top events)
Analysis of signal and effect paths and their logic linkages	Analysis of the effects of failures at higher system levels
Only the relevant default types of the system elements appear in the fault tree and are possible subsets of FME(D)A	All failure types and error sequences of the system elements can be analysed e.g. using FME(C)A respectively FME(D)A

Source: www.fmeaplus.de - 09/2017.

2.3 *Model-Based Safety Analysis*

Nowadays, FTAs and FMEAs are handmade in different fields. These analyses depend on the skills of the safety engineers and the communication between safety engineers and system designers. As a result, it is possible that some aspects may not be covered or that there may be repetitions. In addition, the drastic increase in the complexity of systems will soon no longer allow these artifacts to be generated manually.

MBSA helps to avoid this kind of inconvenience. It is designed to create safety artifacts during the early development phase of critical systems.

MBSA has the benefits of identifying failure scenarios in a repetitive manner prior to the detailed design of technical systems and allows an automated execution of the required safety assessments, hence, further reducing potential "human errors" when analyzing systems safety.

3. Solution

In order to compete with the increasing complexity of technical systems in combination with the faster time-to-market demands, a framework for MBSE using SysML with MBSA and automatic generation of safety artifacts is proposed.

This can be achieved by integrating MBSA into the MBSE development, which in term allows an automatic generation of the safety analysis mandatory to some industrial domain. This helps reducing safety-engineering time in system development.

3.1 *Overview workflow proposal*

A major challenge is to keep these FTA and FMEA analyses up to date in case the system evolves over time. To address these shortcomings, the workflow proposes the automatic generation of FTA and FMEA from a common qualitative system model described with SysML.

To summarize, the proposed workflow combines the following two key elements in the preliminary qualitative MIL (Model-In-the-Loop) concept phase:

- Model of the nominal and failure behaviour of the system using SysML
- Safety Analysis of the model using smartIflow Workbench and the automatic generation of the safety analysis artifacts.

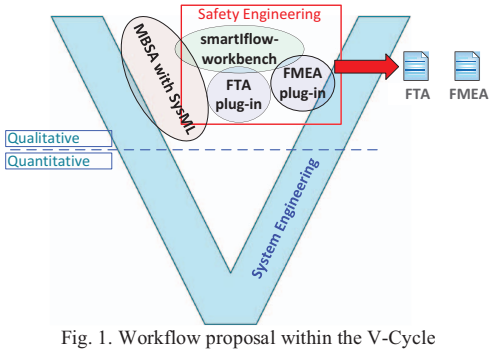

Fig. 1. Workflow proposal within the V-Cycle

In this workflow, fault injection is done in the SysML model while the safety analysis and the automating generation of safety artifacts are done in smartIflow Workbench. In other words, the SysML model that contains the description and the fault injection is translated into smartIflow Workbench by a plug-in. The latter allows to perform safety analysis and to generate automated safety artifacts. The following figure shows the overview of the proposed workflow:

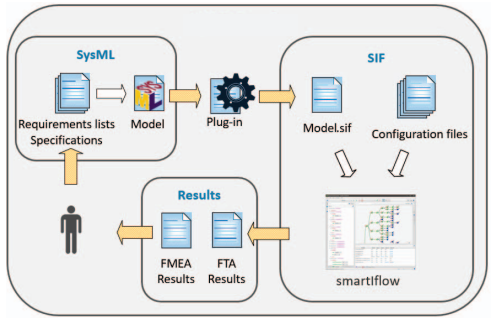

Fig. 2. Workflow proposal

3.2 *Modelling with SysML*

3.2.1 *Plug-In*

A plug-in has been developed in order to translate the SysML language into the smartIflow modelling language. It is divided in three parts (see Fig. 3) in order to be more flexible i.e. minimize the dependencies between input and output:

- Part 1: It is the extraction of all the important information from the SysML model, which is tool dependent. Indeed, practically every tool saves their data in a different format. Basically, the blue part

must be fully rewritten for any new tool to be supported.

- Part 2: This part contains the logic for generating the smartIflow code from SysML diagrams. It was separated from the part 1 so that this algorithmic part is independent of the tool's choice.
- Part 3: This part implements the smartIflow Workbench interface. It is responsible for creating all files of a smartIflow Workbench project which are necessary to perform automated safety analysis.

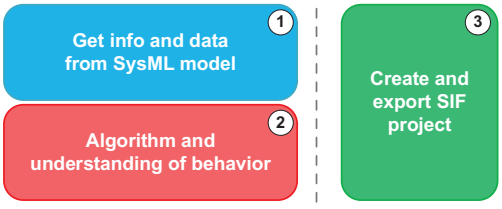

Fig. 3. SysML-plug-in architecture details

3.2.2 *Diagrams in SysML*

SysML is a modelling language for Model based Systems Engineering which enables to visualize and communicate essential aspects of a system like its requirements, its static aspects (structure) and its dynamic aspects (behavior) in a common model and to interlink them together (Delligatti et al. (2019)).

During the workflow development, in order to limit the investment time for both the plug-in developer and the user, research was carried out on the usefulness of all the different SysML diagrams. Indeed, as the proposed approach is located at the beginning of the V-model, not all of the 9 SysML diagrams are necessary. The following five diagrams highlighted in the figure Fig. 4 have been retained:

- Requirements Diagram (req)
- Block Definition Diagram (bdd)
- Internal Block Diagram (ibd)
- Activity Diagram (act)
- State Machine Diagram (stm)

The Use Case Diagram (uc), Sequence Diagrams (seq), Package Diagram (pkg) and the Parametric Diagram (par) are not needed for the rather qualitative description of a system.

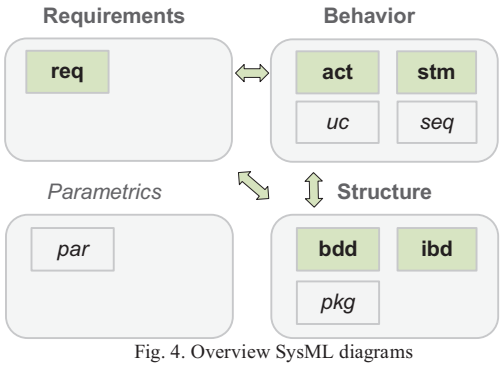

Fig. 4. Overview SysML diagrams

3.2.3 *Failure modelling*

One of the biggest challenges of this research topic is to be able to describe nominal and failure behavior in the same SysML model.

The proposed approach is to separate the nominal and the failure behaviour. A state machine modelling the behaviour of a (sub-) system is divided into two parts (see Fig. 5). The nominal behaviour of the system is described in the state "NominalMode", and the faults are modelled separately in the state "FailureMode" as to avoid redundancy. The faults can then be injected in the nominal model, affecting its intended behaviour. The great advantage of this approach is that the clear separation of both modes facilitates any modification of the nominal mode, and the addition of new faults.

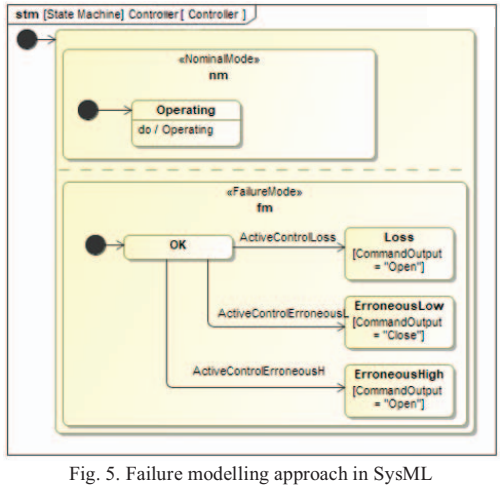

Fig. 5. Failure modelling approach in SysML

3.3 *smartIflow Workbench*

3.3.1 *Introduction*

The smartIflow Workbench[a] is a powerful software tool for model-based safety analysis. It covers a wide range of functionality for model development and analysis, provides a state-of-the-art graphical user interface as well as a scripting interface, and relies on a highly modular and extendable structure which currently consists of about 20 modules of different kinds.

For modelling, the smartIflow language (Hönig et al. (2017)) is used. It is targeted to early phases in the product cycle and provides convenient features for modelling real-world systems on a level of abstraction which is detailed enough to cover the relevant effects for safety but is still suitable for model checking approaches.

Systems are seen as transition systems. Transition systems are directed labelled graphs representing all possible system states and the transitions between them. The state space of real systems is too complex to be described explicitly, but fortunately the theory includes some rather generic representation formalisms such as program graphs (which introduce variables), a concept for combining several program graphs by means of an interleaving operator, and timed automata (which use clocks to quantify delays). See e.g. Baier et al. (2008). Current special purpose languages for model-based safety analysis combine the advantages of those formalisms with additional concepts for reusable context-free component modelling. For example, AltaRica (see Prosvirnova, T. (2014)) adds connections and flow variables to support information exchange between components. The smartIflow modelling language shares many aspects of AltaRica, but adds further concepts, e.g. a fully object-oriented structure, transitions which are triggered by internal state changes and communication, means to calculate flows in arbitrarily structured networks, and features for order-of-magnitude reasoning about flows and time (see Lunde et al. (2018)). As in AltaRica, events are used instead of actions to stimulate the model, e.g. to activate and deactivate a function or a failure mode.

3.3.2 *Analysis of nominal and failure behavior*

This kind of non-deterministic dynamic qualitative models is suitable for a wide range of analysis methods, e.g. classical simulation using a given sequence of external events as input, Markov chain analysis, and model checking. The analysis modules currently available in the smartIflow Workbench concentrate on model checking. An exhaustive simulation (called unfolding) transforms the generic model representation into a transition system. Ideally, the graph contains paths for every possible sequence of input event, so that every possible evolution of the system is covered. Some paths may be infinite if loops are present. But the whole graph is guaranteed to be finite because the state space itself is finite.

Figure 6 shows such a transition system. It was obtained by unfolding the Water Tank Reservoir model which will be described in more detail in Section 4. In fact, this representation is used by the smartIflow Workbench to interactively visualize the result of exhaustive simulation. Nodes represent states and the node on the left-hand side represents the initial state. Edges connect states with possible successor states from left to right. Events causing the state changes are represented by rectangles (red marks external events, yellow marks internal events, red crosses decorate fault activations). Of course, the resulting state graph will not always be tree-structured. Therefore, reference nodes (in blue) are needed to model different paths leading to the same state.

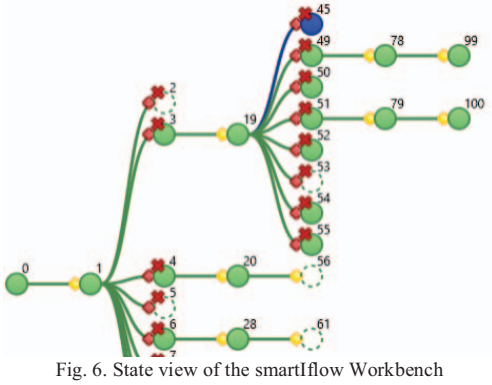

Fig. 6. State view of the smartIflow Workbench

However, complexity can be enormous. So, means to control the size of the transition system have been developed. The most effective one found so far is the so-called event triggering specification (ETS). It provides means to specify restrictions for events on relevant paths. For example, it is possible to limit the unfolding process to paths with two fault mode activations at most:

```
max2Failures := path.filter(type=failure).count() <= 2
```

[a] A test version of the smartIflow Workbench is available at https://smartiflow.bitbucket.io

To limit the number of independent faults on a path makes sense. In fact, for a FMEA only paths with at most one fault mode activation are relevant.

The transition system representation provides a solid base for further analysis. For more or less static system properties, corresponding paths with and without failure mode activations can be identified, and discrepancies in variable values of corresponding states can be interpreted as effects of those failure modes.

In many situations, comparing isolated states is not enough. A typical requirement for a braking system is, that in every possible situation it should stop the car after pressing the pedal to the bottom for some defined duration. Temporal logic is suitable to express this kind of requirements. The example shows a Computation Tree Logic (CTL) variant which is used by the workbench. It is based on order-of-magnitude values for time.

```
AG(pedal.opm == Pushed -> AF{delay<=3} car.speed ==
Zero || pedal.opm == Released);
```

The workbench comes with a special model checker which was designed to analyze smartIflow models in the context of model-based safety analysis. Like other model-checkers, it traverses the states of the transition system and tries to find a counter example, which is a path in which the specified requirement is not fulfilled. But it doesn't stop after one counter example is found. In safety analysis, such a counter example almost always exists. The main question is not whether a counter example exists, but how many of them, which failure modes are involved, and how likely they are. Additionally, the model checker supports order-of-magnitude reasoning about time delays, and path filtering using ETSs.

3.3.3 *Automatic generation of FTA and FMEA artifacts*

The Fault Tree Analysis module strongly relies on the model-checking module. A negated CTL requirement is used as top event. By model checking, counter examples are discovered. They consist of all relevant paths in which the requirement is not fulfilled. The next step is to identify so-called minimal cut sets. Those are the minimal failure mode combinations which can lead to disaster. They are finally used to create a fault tree in disjunctive normal form. Different export formats are available for current FTA tools to visualize the tree and compute corresponding safety numbers such as failure rates.

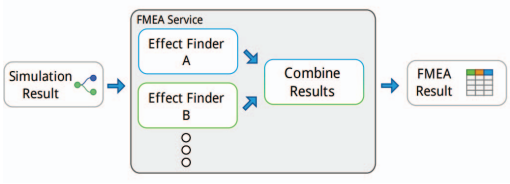

Fig. 7. FMEA-Generator structure

A challenge in FMEA is that various table formats are in use today. So, a major part of the module provides mechanisms to define table formats and text replacement mappings. They support output creation which is very close to the results created manually. However, generic defaults exist, so that first abstract FMEAs can also be created by just pressing a button. The main contribution from the model is the information about the components and their failure modes as well as the relationship between single faults and their possible effects on different integration levels. Components and the possibilities to fail are easily obtained by traversing the model and collecting the failure mode variables with their possible values. For the cause-effect-relation, both analysis methods from 3.3.2 are combined (see Fig. 7). It depends on the model and the requirements, whether effect definitions based on state comparison are meaningful, or whether negated CTL formulas are needed to analyze the behavior over time. In general, it is assumed, that activated failure modes in a counter example are responsible for a violated CTL requirement if no nominal path exists within the set of counter examples. By default, state comparisons are used for local effects and CTL effect specifications for system-wide effects.

4. Use case example

4.1 *Description*

The proposed example is the management of a Water Tank (see Fig. 8).

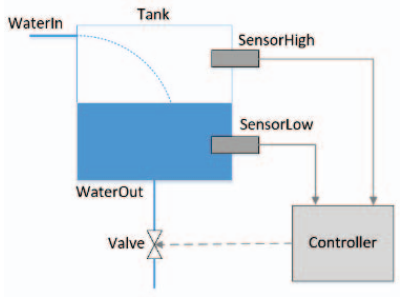

Fig. 8. Water Tank System

The water level is indicated by two sensors SensorLow and SensorHigh which communicate with the Controller. The Controller itself is responsible to open or close the Valve. In nominal mode, the Controller tries to control the Valve in order there is no overflow. Finally, the system has no self-repair and for this example we are only interested in detecting overflow.

4.2 *SysML model*

The internal block diagram (ibd) is shown in Fig. 9. For this simple system the major components from Fig. 8. can be easily identified. The model also shows the necessary data flow between the components.

4.3 *Manual analysis*

In order to compare later on the automatic generated safety artifacts, first a manual FTA and FMEA have been created.

4.3.1 *FTA*

The manually generated FTA (see Fig. 10) shows all events conducting to the top event "Overflow".

This FTA has been done using the system description and the first impression was that all is logical and that the analysis was complete.

4.3.2 *FMEA*

The manually generated FMEA (Table 2) shows all failure modes for each block. In order to limit the space, only the lines concerning the final effect "Overflow" have been presented.

Table 2. Manual FMEA of the Water Tank Reservoir

Function Block	Failure Mode	Failure Effect at Equipment Level
Valve	Loss	Overflow
	Stuck Closed	Overflow
Control	Stuck Low	Overflow
SensorHigh	Erroneous Low	Overflow
	Loss	Overflow
	Stuck Low	Overflow

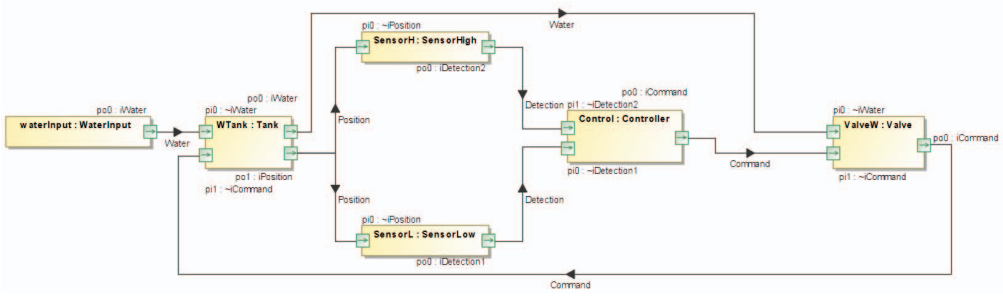

Fig. 9. The internal block diagram (ibd) of the Water Tank

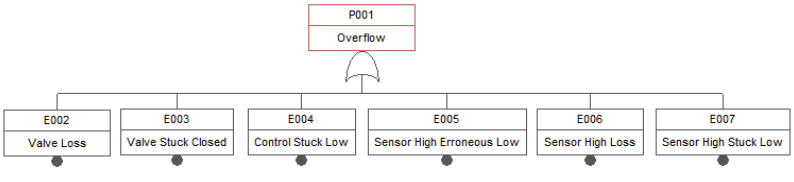

Fig. 10. Manual FTA

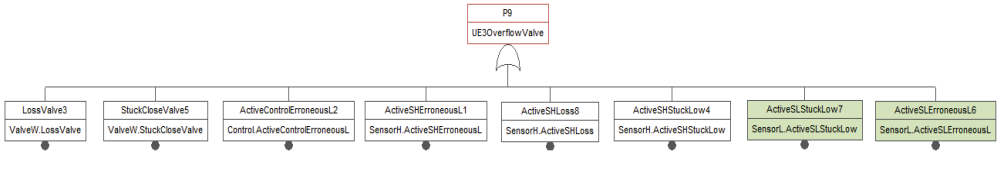

Fig. 11. Automated FTA

4.4 *Automatically generated artifacts*

4.4.1 *FTA*

The automatically generated FTA (see Fig. 11) shows like in the manual FTA, all events conducting to the top event "Overflow". Unlike the manual FTA, the automatic FTA has two additional elements. At the time of the first manual analyses, these two elements had simply been forgotten.

4.4.2 *FMEA*

As for the FTA, the same two elements have also been forgotten for the manual FMEA (Table 3). Even for this simple system the automatic creation of the safety artifacts helps to avoid human errors of omission or misunderstanding of the system.

Table 3. Automatic generated FMEA

Item	Failure Mode	End Effect
Control	ErroneousLow	Overflow
SensorH	ErroneousLow	Overflow
SensorH	Loss	Overflow
SensorH	StuckLow	Overflow
SensorL	ErroneousLow	Overflow
SensorL	StuckLow	Overflow
ValveW	Loss	Overflow
ValveW	StuckClose	Overflow

5. Conclusion and Future Work

In this work, we showed that it is possible to model a system using SysML enabling the utilization of formalized requirements and to connect it thus to the SmartIflow workbench which enables the automatic generation of qualitative safety assessment artifacts (FTA and FMEA).

The safety analysis generated by such an approach enables the system design engineer to identify a) early design flaws and b) safety critical functions and associated elements at a qualitative level very early in the conceptual design phase of a complex technical system. Indeed, with the simple use case presented in this paper it could be demonstrated how such an automated qualitative failure analysis approach can beneficially impact the system design at the very beginning of the system's development cycle where each modelled failure mode is triggered and automatically analyzed.

Consequently, this automated model-based failure analysis will significantly contribute to the identification of design errors and the identification and an initial verification of system safety requirements, thus to reduce engineering and development costs and to optimize the current processes framework in system development of companies in all industrial domains.

Furthermore, it shows that is possible to integrate SysML as an object-oriented modelling language into an automated MIL based failure assessments.

Acknowledgement

This work is based on two projects which have received funding from the NRP (New Regional Policy) program of the Canton of Fribourg/Switzerland. The authors thankfully appreciate the support and sincerely thank the project partners for the fruitful collaboration.

References

Baier, C. and Katoen J-P. (2008). *Principles of Model Checking*. The MIT Press.

Delligatti, Lenny. (2013). *SysML Distilled – A brief guide to the systems modelling language*. Addison-Wesley

Elder, F. (2017). *Effizienz und Akzeptanz durch Kombination von FTA und FMEA.* www.fmeaplus.de - 09/2017

Hönig, P., Lunde R. and Holzapfel F. (2017), *Model Based Safety Analysis with smartIflow*. Information. Vol. 8(1).

Joshi, A. (2008), *Behavioral Fault Modeling and Model Composition for Model-based Safety Analysis*. Minneapolis, MN, USA University of Minnesota.

Lunde, R., Hönig P., and Müller C. (2018). *Reasoning about Different Orders of Magnitude of Time with smartIflow*. IFAC-PapersOnLine, 51(24).

Mhenni, F., Nguyen N., Choley J.-Y. (2014). *Automatic Fault Tree Generation from SysML System Models*

Müller, C., Hönig P., and Lunde R. (2018). *Evaluation of smartIflow based on the Wheel Brake System from ARP4761*. IFAC-PapersOnLine, 51(24).

Prosvirnova, T. (2014). *AltaRica 3.0: A Model-Based approach for Safety Analyses.* Computational Engineering, Finance, and Science [cs.CE]. Ecole Polytechnique

Werdich, M. (2011). *FMEA - Einführung und Moderation*. Springer

Wille, A., Luithardt W, Berns W. (2016). *Designfehler früh erkennen*, Electrosuisse 07/2016