

The qSafe Project – Developing a Model-based Tool for Current Practice in Functional Safety Analysis

Florian Grigoleit¹, Sebastian Holei³, Andreas Pleuß², Robert Reiser¹, Julian Rhein¹, Peter Struss¹, and Jana v. Wedel³

¹Technische Universität München, Munich, Germany

e-mail: {robert.reiser, struss, julian.rhein, florian.grigoleit}@tum.de

²EnCo Software GmbH, Munich, Germany

e-mail: a.pleuss@enco-software.com

³Invensity GmbH, Munich, Germany

e-mail: {sebastian.holei, jana.wedel}@invensity.com

Abstract

Functional safety analysis (FSA), that is checking whether a designed artifact will perform safely even under the presence of failing components, has gained significant importance in different areas, including aeronautic and automotive systems. The same applies to failure-modes-and-effects analysis (FMEA) and fault-tree analysis (FTA) as the major contributing processes. FSA is labor- and time-consuming as well as error-prone, and would benefit from computer-based tool-support. Work on qualitative-model-based systems has developed principled solutions, particularly to FMEA, but not made the step to industrial practice. Rather than novel technical contributions, this paper discusses reasons for this fact and describes the qSafe¹ project, which aims at overcoming the obstacles and at making a major step towards tools that can support current practice.

1. Introduction - Functional Safety Analysis - Current Process and Standards

System and safety analyses have become an integral part of the development of complex systems. They are requested by standards and norms across most areas of industry, especially when dealing with functional safety relevant mechatronic systems: EN 5012x for rail systems, IEC 60601 1-4 for medical equipment, IEC 61513 for nuclear technologies, IEC 61511 for process industries, ISO 26262 for automotive, SAE ARP 4761 for aerospace and MIL 881 for military. In addition, they are an effective means for the iterative optimization of systems regarding important characteristics such as functional safety, robust-

ness, and cost-effectiveness, even at early stages of development.

During the design phase, engineers use system and safety analyses to determine whether the system under development will perform safely, both under nominal conditions as well as under the occurrence of a fault. This is of particularly high importance when system failures may cause injury or even death of humans or other severe damage to the system environment. Analyses may also have to be carried out repeatedly for different versions and variants during the design of a system.

The basic principle of system and safety analyses is to identify the causal relationships between failures of system components or subsystems and potential hazardous behavior of the entire system. Such critical effects must be prevented or mitigated by design changes and/or the introduction of safety functions, both physically and through software. Two of the most commonly used analysis techniques are

- **Bottom-up: FMEA (Failure Modes and Effects Analysis):** For each potential fault of a system component (“failure mode”) it is determined whether the fault may cause any undesired effects, i.e. violate any requirement on the behavior of the system. Typically, the analysis is performed for different scenarios (environmental conditions, system states, ...) and at different levels in a subsystem hierarchy.
- **Top-down: FTA (Fault Tree Analysis):** The starting points are the critical system behaviors; the possible causation of such symptoms by (alternative) combinations of faults is represented in an AND-OR graph across the subsystem hierarchy.

The focus of this paper is **not** presenting novel theoretical results or techniques, but rather shedding a light on what is needed to turn qualitative modeling and basic mod-

¹ The project is funded by the German Federal Ministry of Economics and Technology under the ZIM program (ZF4086001BZ5).

el-based techniques (here: FMEA) into tools that provide a benefit in actual work processes.

In the next section, we illustrate the task, focusing on FMEA and using a benchmark example from one of the standards. Section 3 discusses the main problems in the current FSA process. In section 4, we briefly survey the current tool landscape, highlighting model-based approaches, whose limited utility in current practice is analyzed in section 5. Based on this, we present the attempt to overcome these limitations in the qSafe project in section 6. First results are presented briefly in section 7.

2. The Current Benchmark Example

As a first benchmark study for the discussion of fundamental concepts and challenges and for the evaluation of the qSafe tool we have chosen the transport aircraft wheel brake system (WBS), an example given in the SAE ARP 4761 standard [1], because the standard provides a variety of exemplary safety assessment results, which can be used for the verification of the automatically generated results. Additionally, the WBS is a popular example for other model-based approaches, for example [2] [3] [4] [5], which allows the direct comparison of the qSafe approach to the related work.

The system is composed of two functional subsystems: Braking is electrically controlled by the braking system control unit (BSCU), which receives the pedal deflection signals from the flight deck and sends actuation commands to the hydraulic system. The hydraulic portion of the system contains the components required for the transduction of hydraulic power and the actuation of the wheel brakes (for details on the system architecture see [1]).

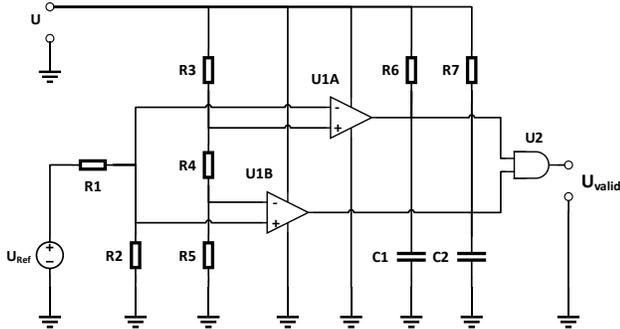


Figure 1: Voltage Monitor Circuit [1]

The preliminary study presented in this paper focuses on the voltage monitoring function of the power supply subsystem of the BSCU. The circuit level implementation of this function is shown in figure 1. The circuit is designed as a window comparator. The ICs U1A and U1B compare the resistively divided test voltage U to the reference voltage provided by the voltage source U_{Ref} in order to detect over (U1B) and under (U1A) voltage conditions. The capacitors C1 and C2 together with the resistors R6 and R7 implement simple first order lag filters in order to effec-

tively delay the circuit output in the presence of noise or ripple on the input signal.

The analysis task during the FMEA is to determine the (potential) effects of every possible failure mode for all components of the system under study. Examples for typical failure modes are “open”, “increased resistance”, “decreased resistance” for resistors or “output open” and “output shorted to ground” for the comparator ICs. The resulting effects describe how the intended function of the overall circuit is affected in the presence of failures. Typical effects would be:

- **Monitor stuck valid:** The voltage monitor always produces a high (valid) output signal, regardless the actual input voltage
- **Monitor stuck tripped:** The voltage monitor always produces a low (invalid) output signal
- **Window shift:** The thresholds for the validation of the input voltage both decrease or increase
- **Trip window widens/tightens:** The thresholds for the validation of the input voltage shift in opposite direction, such that the monitoring function becomes less/more sensitive

Figure 2 shows a section of the FMEA table given by the standard as a reference.

Component ID	Part type	Failure Mode	Failure Mode Rate (E-6)	Failure Effect Code	Failure Effect	Detection Method
R1	Film Resistor	open	.0009	3	over volt monitor trips	P/S shutdown
		increase resistance	.0005	4	trip window shifts down	
		decrease resistance	.0004	4	trip window shifts up	
R2	Film Resistor	open	.0009	3	under volt monitor trips	P/S shutdown
		increase resistance	.0005	4	trip window shifts up	
		decrease resistance	.0004	4	trip window shifts down	
R3	Film Resistor	open	.0009	3	under volt monitor trips	P/S shutdown
		increase resistance	.0005	4	trip window shifts up	
		decrease resistance	.0004	4	trip window shifts down	
R4	Film Resistor	open	.0009	1	monitor stuck valid	bench test
		increase resistance	.0005	2	trip window tightens	bench test
		decrease resistance	.0004	1	trip window widens, may cause monitor stuck valid	bench test
R5	Film Resistor	open	.0009	3	over volt monitor trips	P/S shutdown
		increase resistance	.0005	4	trip window shifts down	
		decrease resistance	.0004	4	trip window shifts up	

Figure 2: Partial results of the reference FMEA from [1]

3. The current Process: High Efforts and Error Proneness

The analyses mentioned in section 1 and 2 are carried out by experts with different backgrounds, from design to after-sales and require a significant amount of time and, hence, precious labor cost. This is due to

- the **combinatorics** - the number of cases to be analyzed in FMEA is determined by the product of the number of components, the average number of faults per component, the number of scenarios and of effects/requirements that must be considered - and

- the **repetitive** nature of the analysis, which has to be redone (at least in theory) after each design change. Even a small change may render a large portion of the previous analysis obsolete.

The second critical aspect is the **quality** of the results. Even though experts are involved and the fact that each single analysis is usually not too complicated, FMEA tables may be incomplete and contain mistakes due to the sheer number of cases.

4. Current Tool Support

Most commercially available tools are confined to supporting bookkeeping of the process and editing, storing, displaying and retrieving data and analysis results. They usually support multiple FSA methods such as FMEA, FMEDA (which includes diagnostic capability information to the analysis), FMECA (where an additional classification of hazard criticality is performed), FTA and other probabilistic modeling methods (e.g. RBD and Markov analyses) as well as scenario-based analysis techniques such as HARA. Examples of such tools are: Safety Office X2 (SOX2) [6], Medini Analyze [7], APIS IQ-FMEA [8], and Isograph Reliability Workbench [9]

While they are useful in avoiding mistakes or omissions, they provide no support to the key task of the analysis: determining the causal interdependencies of component faults and their impact on system behavior. Since this requires reasoning about the structure of a system and the behavior of its building blocks to infer the overall system behavior, using models to support the analysis is an obvious idea. A broad overview of the recent advances in model-based safety assessment and a classification of the different approaches is given in [10]. Most approaches focus on the automated generation of fault trees. AltaRica [11] is a formal language for the specification of complex, hierarchical systems in terms of automata and equations, which allows for modeling both functional and dysfunctional behavior of a system (for details see [12] [13] [14] [15]).

The FSAP/NuSMV-SA [16] approach, which was developed as part of the ESACS project [17], supports failure mode definition based on a library of commonly used failure models, automatic fault injection and automatic fault tree construction, based on a model checker.

HiP-HOPS (Hierarchically Performed Hazard Origin and Propagation Studies, [18]) is a tool that supports the annotation of hierarchical system models with failure logic information, which can then be used for the automatic generation of fault trees. A MATLAB toolbox allows to integrate HiP-HOPS and Simulink models [19].

As illustrated by the example, the analysis is essentially qualitative, in linking classes of faults (“increased resistance”) with classes of effects (“trip window shifts down”). For this reason, research on qualitative modeling

and model-based systems has considered FMEA quite early as a challenge and potential application domain. The Autosteve system [20] performed automated FMEA of electrical circuits for cars, while the AUTAS project delivered a generic algorithm based on model libraries of arbitrary kinds of physical systems (evaluated on electrical, hydraulic, and mechanical aeronautic and automotive systems [21]). Further case studies applied the approach to an anti-lock braking system [22] [23] the drive-train of a truck [24] and extended it to include software and cyber-physical systems [25]. Although this work produced positive results in the case studies performed and even lead to commercial products (SimCertify, Rodon, Raz’r’s FMEA engine [26]), it has not yet been adopted in industrial practice to a significant degree.

The current manual process and expert communication as depicted in figure 3 is replaced by the generation and use of formal behavior models based on the system design and the formalized requirements, effects, and scenarios (fig. 4).

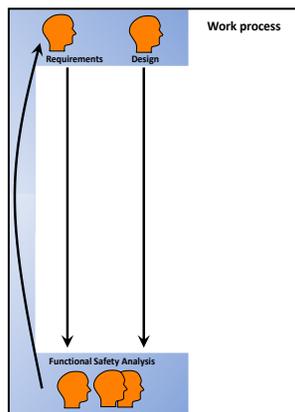


Figure 3: Status quo

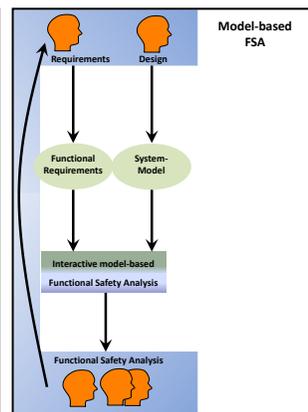


Figure 4: qSafe process

In the context of model-based systems, FMEA can be formalized as the task of determining whether an effect EFF_k (which may be given as the negation of a requirement) may occur under a model of the system with an assumed fault, $MODEL_f$, in a scenario $SCEN_j$:

$$MODEL_f \cup SCEN_j \cup EFF_k \neq \perp,$$

or, stronger, is entailed by them:

$$MODEL_f \cup SCEN_j \models EFF_k.$$

The consistency and entailment checks can be implemented by constraint satisfaction algorithms, as in the Raz’r FMEA engine, which is used in the qSafe project. Compositional modeling is an enabling technology for this approach, since manual generation of system models for all potential component failures is prohibitive.

5. The Limitations of Model-based Solutions

Despite the successful case studies mentioned above, the impact of model-based solutions on industrial practice is, stated cautiously, very limited, which has several reasons.

- **Modeling effort:** developing the required repository of component models is a challenge and requires high efforts, even more so, if they are to support a compositional approach and, hence, have to be generic and context-independent to be re-usable.
- **Multiple Models:** Composing system models that are appropriate for a particular analysis task may require different scope, expressive power, granularity etc., dependent on the set of requirements to be checked, relevant thresholds, the level of detail of the design etc.
- **Creation of multiple models:** There is a caveat concerning the creation of re-usable component models. In practice, it is impossible to anticipate and enumerate all potentially useful or needed variants of a component model.
- **Formalization of requirements:** automated model-based analysis of (safety) requirements can only work, if the latter are stated in a formal manner, something that is usually far away from current practice. Even stronger: what is needed is not only a formal representation, but a formal representation that is **related to the models in the library**.
- **Selecting and composing tailored system models.** Not only do the formal requirements have to „speak the language of the available models“, but also the dual holds: for the analysis of a set of requirements, the model elements that are suited to perform this analysis have to be selected from the multiple model library, and again, it cannot be expected that this is done manually by the engineer.
- **Performing the analysis in an interactive process:** the main contribution of model-based systems in this area is providing algorithms for the key steps of an automated analysis, i.e. performing model composition and the inferences stated formally in section 4. What is lacking is embedding these steps into the current work process which reflects various standards and comprises much more than establishing associations between component failures and critical system behavior.

6. qSafe – Tackling the Obstacles

The consortium of the project is a mixture of academic and industrial partners with complementary expertise:

- EnCo Software GmbH develops and markets a tool, Safety Office X2 (SOX2), which supports the entire FSA process including requirements, system design, tests, and traceability.
- Invensity GmbH is a technology consultancy company with vast expertise in FSA processes.

- The Flight System Dynamics chair at the Technical University of Munich (TUM) brings in expertise in the areas of numerical modeling
- The Model-based systems and qualitative modeling (MQM) group of TUM provides the techniques of model-based systems
- Finally, the project exploits model-based software for diagnosis and FMEA from OCC’M Software GmbH [26].

The experience of the consortium covers, in particular, aeronautic, automotive, and rail systems.

6.1. qSafe Architecture

qSafe exploits the core of existing model-based inference engines for FMEA and diagnosis (for FTA) and embeds them in an environment of modules (see figure 5).

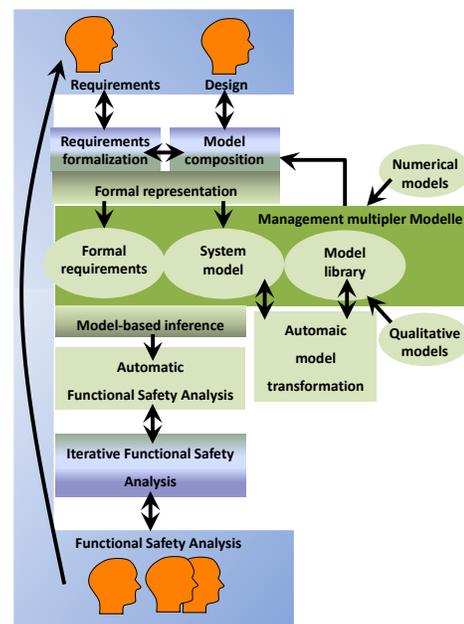


Figure 5: Modular qSafe Architecture

These modules address the barriers described in section 5:

- the organization of **multiple models** to face the need of exploiting various models dependent on the type of analysis task, system, etc. It includes the representation of a high-level ontology and its mapping to the specific modeling environments and model instances implemented in these environments.
- a collection of several operators for **transforming models**, predominantly, but not exclusively, for generating qualitative models from numerical equation-oriented models (e.g. in Modelica or SimScape) to reduce the cost of producing adequate models for the analysis
- support to, on the one hand, the **formulation of formal requirements**, effects, and scenarios without requiring in-depth information about details of the

available models (exploiting the ontology provided by the first module) and, on the other hand, to the extraction of model fragments and the **construction of a model that is tailored** to the needs of the analysis.

- The **interaction of experts with the automated analysis** tools, including hierarchical analysis, persistent corrections and modification of automatically generated results, conforming to the respective standards etc.

In the following, we discuss challenges to be addressed by these modules.

6.2. Multiple Modeling

The basis for this has been described in [27]: tool support for creating and managing models must provide more than just libraries for common modeling environments (such as Matlab or Modelica): The tool must be able to deal with models of different granularity (numerical vs. qualitative) and expressiveness (e.g., steady state vs. transient, absolute vs. relative descriptions) to address the need for different model variants in different development phases. For instance, early phases in the development process might require coarse-grained and purely qualitative models while later phases might require specification of numerical threshold values. In particular, the required granularity and expressiveness of the models depends on the system requirements to be analyzed. For instance, the same component (e.g., a resistor) might have to be analyzed based on qualitative deviations or numerical threshold values and with or without consideration of the influence of temperature. Moreover, it should be possible to make use of models specified in different modeling environments or formalisms (e.g., Matlab, Modelica, constraint systems, etc.), to avoid practical limitation of the approach.

Consequently, it must be possible to manage a variety of models of different formats, granularity and expressiveness and to distinguish and select them by the following criteria:

- the represented physical units (and their derivatives),
- the handling of deviations,
- the selected domains (sign, numerical, intervals, etc.),
- the modeling assumptions (e.g., steady state, constant temperature, etc.), and
- the modeling environments the models are implemented in and that are used for calculation or simulation.

It must be possible to select, combine, and evaluate models with respect to the above criteria based on the requirements (e.g., specified effects and scenarios) to be analyzed for the system-under-development. Hence, an appropriate tool must provide an abstract interface to the different models to access them without knowledge of their specific format and content (such as variable names). This requires definition of the common high-level concepts (ontology) which can then be mapped to the specific occurrences in the different models.

This module implements these requirements and provides the following functionality:

- Storing and accessing component and system models, structuring them into models that specify the normal behavior and failure models,
- Indexing models by the above criteria such as the physical units considered, chosen variables (absolute vs. derivative vs. deviation), and modeling assumptions.
- Representing relationships between component models such as abstraction and refinement
- Providing an interface to different modeling environments to execute models
- Creation and representation of hierarchical model structures, in particular, failure models on higher level of abstraction (e.g. aggregated components)

6.3. Automated Model Transformation

The applicability of a model-based solution is strongly dependent on the effort required for generating an appropriate model. In the first place, this is related to the creation of the model library. The introduction of new and complex modeling languages generated from an academic perspective must be avoided because they simply will not be accepted in current practice. Qualitative models can be obtained by abstraction of fine-grained mathematical models which are often already implemented in common modeling environments: In past work [28], we developed automated abstraction of directed numerical models (Matlab). Besides extending this to equation-based models (Modelica, SimScape), the aim of qSafe is the development of a set of algorithms for automated model transformation, as depicted in Figure 6.

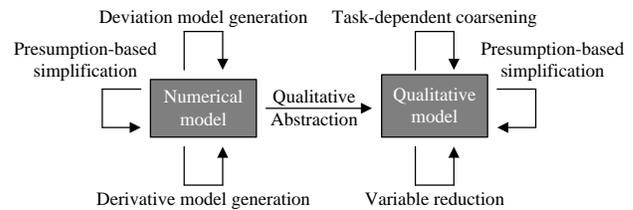


Figure 6: Overview of the model transformations

This comprises, in particular:

- Generation of qualitative models by automated discretizing of implemented numerical models
- Automated generation of (qualitative) deviation models from equation-based, numerical models
- Generation of models for the derivation of effects over time (e.g. system response after an error)
- Task-dependent behavior abstraction: coarsening of a system model without losing the force of expression regarding the task given by system, requirements and scenarios [29].
- Variable reduction
- Assumption-based simplification

Another part of this module is the structural aggregation of models.

In qSafe, not only directed but also undirected numerical models are used as base for the abstraction process. This leads to new challenges, e.g. which ports can be used as input and output to calculate the abstraction and how the variables of an undirected model are related among themselves.

6.4. Formalization of Requirements and Model Composition

To enable automated model-based system and safety analyses, not only a model of the system under consideration but also the (safety) requirements imposed on it must be available in a formalized manner. Interdependencies between the chosen formal representation of the requirements and that of the model exist. On the one hand, the formulation of requirements has to be compliant with the model, i.e. a mapping of aspects mentioned in requirements to constructs and variables of the model must be possible. On the other hand, a system model that allows checking the compliance with the system requirements in different situations must be generated.

To ensure that these interdependencies are taken into account and to support and ease the process as much as possible, the Requirements Formalization Module of qSafe provides tool support for transforming informally written requirements into a suitable formal representation. The module also handles the composition of a system model from multiple models available in the qSafe model library, each of which might only represent certain aspects or components of the overall system.

Example:

The informally written effect “over voltage monitor stuck tripped” must be formalized in such a way that it is possible to check for this effect using an existing model. To enable this even if the user does not have detailed knowledge of the model, and to account for the fact that, once formalized, a requirement could be checked by different model implementations, the user is guided through several steps:

- “tripped” must be described in terms of a quantity of the model ontology, e.g. “voltage”
- The expression must be linked to a component of the system model, e.g. “over voltage monitor” refers to the component U1B of the voltage monitor circuit. The expression can be further localized to the output terminal O of the comparator.
- “stuck” refers to a deviation of the voltage level from a reference value and the direction of this deviation, i.e. the voltage level is high in a situation where it should be low, so overall a “+” deviation of the voltage

This results in a formalized effect

$$\text{sign}(U1B.O.V.\text{deviation}) > 0$$

At the same time, we can derive that any model used to verify this requirement must include the concepts of voltage, sign, and deviation. Moreover, the system model must be coherent and expressive enough to ensure that inferences regarding those concepts can be made. In the example under consideration, the effect “over voltage monitor stuck tripped” implies indirect requirements towards the comparator model, e.g. deviation of output voltage.

The module realizes the model-compliant formalization of requirements and the requirements-compliant composition of system models by supporting the formal representation of requirements through

- Offering of templates and patterns for requirements, scenarios, effects, and hazards
- Offering of modeling primitives (physical quantities, variables, and the mapping in between) to fill out the templates
- Supporting the creation of user defined refinements of templates, patterns and modeling primitives to best support his needs
- Suggesting additional types of requirements that might still be needed to fully describe the system at hand (based on the already defined requirements)
- Creation and storage of requirements based on templates, patterns and modeling primitives

and by enabling model composition via

- Identification of models from the library which are suitable for the analysis of the given requirements
- Automated generation of a system model out of the library models
- Transformation of the system model into an internal representation which is suitable for the automated model-based analysis

6.5. Interactive Model-based Functional Safety Assessment

In addition to the core inferences explained in section 4, effective tools need to embed these automatic analyses in the conventional safety assessment workflow. This includes analysis procedures which are in line with the applicable industrial standards (such as hierarchical analyses) as well as techniques for the transparent and comprehensible presentation and modification of the automatically generated results. A challenge that arises in this context is the generation of explanations for the results of the automatic safety assessment: As the analysis is carried out based on consistency checks over the model variables and their respective constraints, it provides no causal information whatsoever and potentially all model variables and components can contribute to a deduced relation. An intuitive explanation however calls for a causal chain (i.e. a directed relation) that associates the observed effect with its causes through a series of system components which are involved in the failure propagation.

Furthermore, methods for the manual modification and refinement of the generated artifacts are required: The analyzed qualitative models are coarser than their numerical counterparts and might lead to an over-estimation of certain effects (a positive deviation may be indicated that an expert – or a numerical analysis - considers insignificant), and effects that are correctly identified might be considered irrelevant in a given context.

The module accounts for this challenges by introducing the following concepts:

- **Hierarchical analysis:** The environment for the interactive functional safety assessment features the specification of analysis tasks based on the hierarchical system structure. Depending on the definition of effects and requirements, a hierarchical system model can be flattened for the analysis (which is required by the inference engine). The traceability between analysis artefacts and system model components is always ensured. Fault propagation through the system hierarchy can be displayed using common graphical representation tools such as failure nets and fault trees. Analysis results can be modified by the application of annotations and filters, which ensure that modification decisions persist in subsequent analyses following design model changes.
- **Dependency analysis and model slicing:** Program slicing is a method for the extraction of a partition of a computer program that affects or is affected by the statements at a specified point in the program (for details refer to [30] and [31]). In [32] an application of program slicing techniques to component oriented data flow models is given, where the data and control flow of the blocks is evaluated to construct a dependency graph. Model slices can then be extracted by performing reachability analyses on the dependency graph. This approach is one candidate. In the context of MBSA, interactive failure dependency is introduced, which evaluates the analysis results for a given failure configuration in order to determine whether the values of a components variables depend on the failure of another component. The combination of data dependency and failure dependency allows for the construction of fault propagation traces.
- **Harness modeling:** A harness model is an executable model of a failure configuration (i.e. a single FMEA row, a trace in a failure net or a single cut set of a fault tree) that was identified during the automatic analysis. The module provides a feature for the automated generation of harness models in the supported simulation environments. This facilitates the intuitive explanation and comprehension of the automatically generated results, as the harness models can be simulated to enrich the results of the automatic analysis by numerical simulation results as required.: In figure 7, the envisioned analysis process is compared to the conventional manual safety assessment using the FMEA process according to [1] as an example.

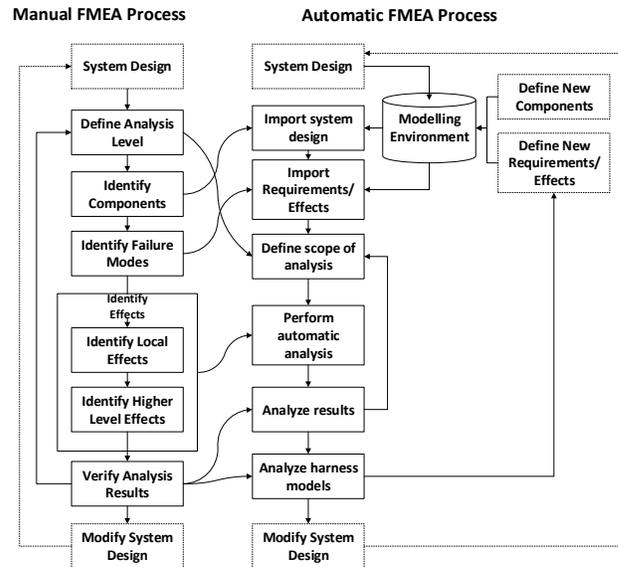


Figure 7: The automatic FMEA process compared to the manual FMEA process according to [1] including the mapping of the different process stages

7. First Results

As a proof of concept, and in order to produce a reference example to highlight the requirements on these modules and to provide test cases, a qualitative model was constructed for the circuit described in section 2, and the automatic FMEA was carried out using Raz’r’s FMEA engine. The analyzed model is a qualitative abstraction of the equation network that can be derived from the circuit shown in figure 1. Table 1 summarizes the equations of a resistor model, which describe the nominal component behavior as well as the applicable failure modes. The modeling approach used here includes both the magnitude of the respective model variables as well as their deviation from a reference value. The deviation is defined as $\Delta x = x - x_{ref}$. Note that the “Background” mode behavior corresponds to physical laws that hold under any mode.

Mode	Magnitude	Deviation
Background	$R \geq 0$	$R - \Delta R \geq 0$
Nominal	$U = R \cdot I$	$\Delta U = R \cdot \Delta I$ $\Delta R = 0$
Open	$I = 0$	$\Delta I = -I_{ref}$ $\Delta R \rightarrow \infty$
Decreased Resistance	$U = R \cdot I$	$\Delta U = \Delta I \cdot (R - \Delta R) + I \cdot \Delta R$ $\Delta R < 0$
Increased Resistance	$U = R \cdot I$	$\Delta U = \Delta I \cdot (R - \Delta R) + I \cdot \Delta R$ $\Delta R > 0$

Table 1: Exemplary resistor model

These equations and inequalities are then abstracted to a qualitative model, where for all model variables only the

sign of numerical value and deviation is considered (for further details on the principles of model abstraction and qualitative (deviation) models refer to [23] and [33]). The resulting set of finite constraints over the model variables can then be checked for its consistency with the formalized effects to derive the relations between failure modes and effects (refer to section 4). Examples for such formalized effects are (using the symbols from figure 1):

MonitorStuckValid (MSV):

$$MSV = (\Delta U \neq 0) \wedge (U_{valid} > 0)$$

MonitorStuckTripped (MST):

$$MST = (\Delta U = 0) \wedge (U_{valid} = 0)$$

Table 2 shows the partial results of the automatic analysis to be compared to the table given by the standard (see section 2).

Part	Failure Mode	Failure Effect
R1	DecreasedResistance	MonitorStuckTripped
R1	DecreasedResistance	UnderVoltMonitorStuckTripped
R1	DecreasedResistance	TripWindowShiftsUp
R1	Open	MonitorStuckTripped
R1	Open	OverVoltMonitorStuckTripped
R1	Open	TripWindowShiftsDown
R1	IncreasedResistance	MonitorStuckTripped
R1	IncreasedResistance	OverVoltMonitorStuckTripped
R1	IncreasedResistance	TripWindowShiftsDown
[...]		
R4	DecreasedResistance	TripWindowTightens
R4	Open	MonitorStuckValid
R4	IncreasedResistance	TripWindowWidens
R5	DecreasedResistance	TripWindowShiftsUp
R5	Open	OverVoltMonitorStuckTripped
R5	IncreasedResistance	TripWindowShiftsDown

Table 2: Partial results of the automatic FMEA

For resistor R1, the results of the standard document are completely included, for the remaining components only the most relevant effects are depicted in the table, while spurious lines are filtered after manual review (as described in section 6.5 and discussed below). It can be noted that all effects, which are identified in the reference according to Figure 2 are represented. In fact, **even an error in the reference has been detected** by the automated analysis: For resistor R4, the failure mode “Decreased resistance” causes the window for the validation of the input voltage to tighten, as one can easily show by manual calculation. An increase of the Resistance R4 causes the window to widen. In the standard document (Figure 2), these effects are swapped. Stating this fact is not meant to indicate that the models are sophisticated and superior to expert knowledge, but, rather, that even carefully performed FMEA does produce errors that can be easily avoided. Besides, it can be noticed that Table 2 includes additional effects as shown for R1, where “MonitorStuck-

Tripped” is identified for increased and decreased resistance. This is an example for the conservative over approximation of an effect: The qualitative models describe the failure mode “Increased resistance” by a positive deviation $\Delta R > 0$. However, they make no statement about the magnitude of this deviation. In the case of a very high resistance, the behavior approaches the state of an open circuit, which indeed causes the over volt monitor (and the entire circuit) to trip. This is a good example for an effect that might be considered spurious by a safety analyst, highlighting the need for effective methods for the manual modification of the results.

8. Discussion:Expected Benefit

qSafe aims at cost reduction in two ways:

- reduction of the cost for FSA during the design process through automation
- reduction of costs through early detection of design faults.

Regarding the first aspect, the objective is not completely automating the entire process, which is not feasible, but rather reducing the efforts to be spent on a significant part of the analysis, which is more or less a routine kind of work, estimated to be 70-80% of the entire FMEA analyses. Case studies suggest that these efforts can be reduced from several days to seconds of computing time plus manual assessment of the results (single digit man hours). qSafe aims, conservatively estimated, at a coverage by the automated analysis of 50%. This reflects that, for instance, certain component fault behaviors or complex scenarios may not be adequately representable.

The automatically generated results need to be manually inspected and, potentially corrected. Since computation time is negligible w.r.t. to these efforts, the inspection time basically determines the cost of the automated analysis.

We estimate that this time may be about 20% of the time needed for a manual analysis. With these estimates, the overall saving in time will be

$$\text{Degree of automation} * (1 - \text{time factor manual inspection}) \\ 50\% * 80\% = 40\%$$

Based on figures for the investments of German automotive companies alone in the analysis of functional safety, the estimated 40% cost reduction through automation translates into savings of 280 million € per year.

Regarding the second aspect, completeness and correctness of the results will be increased by the automated analysis. A conservative estimate is that 5% of the potential entries of a manually created FMEA table are faulty or missing. We cautiously estimate that at least one third of such faults can be avoided. Thus, savings through early avoidance of design faults is calculated as

$$\text{Degree of automation} * \text{ratio fault avoidance} \\ = 50\% * 33\% = 16.5\%$$

Acknowledgements

We thank Oskar Dressler from OCC'M Software for his support by making Raz'r's inference engines available at this stage.

References

- [1] Society of Automotive Engineers, "ARP4761: Guidelines and Methods for Conducting the Safety Assessment Process on Civil Airborne Systems and Equipment," SAE International, 1996.
- [2] A. Joshi and M. P. Heimdahl, "Model-Based Safety Analysis of Simulink Models Using SCADE Design Verifier," in *The 24th International Conference on Computer Safety, Reliability and Security*, Fredrikstad, 2005.
- [3] T. Peikenkamp, A. Cavallo, L. Valacca, E. Böde, M. Pretzer and M. Hahn, "Towards a Unified Model-Based Safety Assessment," in *Computer Safety, Reliability, and Security: 25th International Conference, SAFECOMP 2006*, Gdansk, Poland, 2006.
- [4] A. Joshi and M. Heimdahl, "Behavioral Fault Modeling for Model-based Safety Analysis," in *10th IEEE High Assurance Systems Engineering Symposium*, Washington, DC, USA, 2007.
- [5] M. Bozzano, A. Cimatti, A. Fernandes Pires, D. Jones, G. Kimberly, T. Petri, R. Robinson and S. Tonetta, "Formal Design and Safety Analysis of AIR6110 Wheel Brake System," in *Computer Aided Verification: 27th International Conference, CAV 2015, Proceedings, Part I*, San Francisco, CA, USA, 2015.
- [6] "ENCO Software," [Online]. Available: <http://www.enco-software.com/software/>.
- [7] "medini analyze," [Online]. Available: <http://www.medini.eu/index.php/de/products/functio-nal-safety>.
- [8] "APIS IQ-Software," [Online]. Available: <https://www.apis.de/software/produkte/>.
- [9] "Isograph Reliability Workbench," [Online]. Available: <https://www.isograph.com/software/reliability-workbench/>.
- [10] O. Lisagor, T. Kelly and R. Niu, "Model-Based Safety Assessment - Review of the Discipline and its Challenges," in *9th International Conference on Reliability, Maintainability and Safety (ICRMS), 2011*, 2011.
- [11] A. Arnold, G. Point, A. Griffault and A. Rauzy, "The AltaRica Formalism for Describing Concurrent Systems," *Fundam. Inf.*, pp. 109-124, 1999.
- [12] A. Rauzy, "Mode automata and their compilation into fault trees," *Reliability Engineering & System Safety*, pp. 1-12, 2002.
- [13] P. Bieber, C. Castel and C. Seguin, "Combination of Fault Tree Analysis and Model Checking for Safety Assessment of Complex System," in *Proceedings of the 4th European Dependable Computing Conference on Dependable Computing*, London, UK, 2002.
- [14] P. Bieber, C. Bagnol, C. Castel, J.-P. Heckmann, C. Kehren, S. Metge and C. Seguin, "Safety Assessment with Altarica - Lessons Learnt Based on Two Aircraft System Studies," 18th IFIP World Computer Congress, Topical Day on New Methods for Avionics Certification, 2004.
- [15] C. Kehren, C. Seguin, P. Bieber, C. Castel, C. Bagnol, J.-P. Heckmann and S. Metge, "Advanced simulation capabilities for Multi-systems with Altarica," in *International System Safety Conference*, 2004.
- [16] M. Bozzano and A. Villafiorita, "Improving System Reliability via Model Checking: The FSAP/NuSMV-SA Safety Analysis Platform," in *Computer Safety, Reliability, and Security. SAFECOMP 2003*, Edinburgh, UK, 2003.
- [17] M. Bozzano, A. Villafiorita, O. Akerlund, P. Bieber, C. Bagnol, E. Böde, M. Bretschneider, A. Cavallo, C. Castel, M. Cifaldi, A. Cimatti, A. Griffault, C. Kehren, B. Lawrence, A. Lüdtkke, S. Metge, C. Papadopoulos, R. Passarello, T. Peikenkamp, P. Persson, C. Seguin, L. Trotta, L. Valacca and G. Zacco, "ESACS: an integrated methodology for design and safety analysis of complex systems," in *ESREL*, Maastricht, 2003.
- [18] Y. Papadopoulos and J. A. McDermid, "Hierarchically Performed Hazard Origin and Propagation Studies," in *Computer Safety, Reliability and Security. SAFECOMP 1999*, Toulouse, France, 1999.
- [19] Y. Papadopoulos and M. Maruhn, "Model-based synthesis of fault trees from Matlab-Simulink models," in *2001 International Conference on Dependable Systems and Networks*, 2001.
- [20] C. Price, "Autosteve: Automated Electrical Design Analysis," in *Proceedings of the 14th European Conference on Artificial Intelligence (ECAI)*, Berlin, Germany, 2000.
- [21] C. Picardi, L. Console, F. Berger, J. Breeman, T. Kanakis, J. Moelands, S. Collas, E. Arbaretier, N. De Domenico, E. Girardelli, O. Dressler, P. Struss and B. Zilbermann, "AUTAS: a tool for supporting FMECA generation in aeronautic systems," in *16th*

European Conference on Artificial Intelligence,
2004.

- [22] P. Struss and A. Fraracci, "Automated Model-based FMEA of a Braking System," in *8th IFAC Symposium on Fault Detection, Supervision and Safety of Technical Processes (Safeprocess 2012)*, Mexico City, 2012.
- [23] P. Struss and A. Fraracci, "Modeling Hydraulic Components for Automated FMEA of a Braking System," in *Proceedings of the Annual Conference of the Prognostics and Health Management Society 2014*, Fort Worth, Tx, USA, 2014.
- [24] P. Struss and S. Dobi, "Automated Safety Analysis of Vehicles Based on Qualitative Behavior Models and Spatial Representations," in *The 24th International Workshop on Principles of Diagnosis (DX-2013)*, Jerusalem, Israel/Palestine, 2013.
- [25] P. Struss, "Model-based Analysis of Embedded Systems: Placing it upon its Feet instead of on its Head - An Outsider's View," in *8th International Conference on Software Engineering and Applications (ICSOFT-EA 2013)*, Reykjavik, Iceland, 2013.
- [26] "OCC'M Software," [Online]. Available: <http://www.occm.de/>.
- [27] P. Struss and T. Regassa, MOM - An Environment for Multiple Modeling, 24th International Workshop on Qualitative Reasoning, Portland, USA, 2010.
- [28] P. Struss, Automated Abstraction of Numerical Simulation Models - Theory and Practical Experience, Sixteenth International Workshop on Qualitative Reasoning, Sitges, Catalonia, Spain, 2002, 2002.
- [29] M. Sachenbacher and P. Struss, Task-dependent qualitative domain abstraction, *Artificial Intelligence* 162 (2005) pp. 121-143, 2005.
- [30] F. Tip, "A Survey of Program Slicing Techniques," *Journal of Programming Languages*, pp. 121-189, 1995.
- [31] J. Silva, "A vocabulary of program slicing-based techniques," *ACM Computing Surveys*, 2012.
- [32] R. Reicherdt and S. Glesner, "Slicing of SIMULINK Models," in *Proceedings of the 34th International Conference on Software Engineering*, Zurich, Switzerland, 2012.
- [33] P. Struss, "Models of Behavior Deviations in Model-based Systems," in *Proceeding of the 16th European Conference on Artificial Intelligence*, Valencia, Spain, 2004.