International System Safety Society Congress 2015

A Model-Based Safety and Dependability Methodology for Missile Safety Engineering

Dipl.-Ing. Dieter Fasol; MBDA Deutschland GmbH/System Safety; Schrobenhausen, Germany
Dr.-Ing. Burkhard Münker; icomod – munker consulting; Freudenberg, Germany
Peter Bunus, ESI-Group, Sollentuna, Sweden
Keywords: model-based, safety, dependability, analysis, safety assessment, safety engineering, missile

Abstract

Model-based methods are becoming more prevalent to support system development processes while in the field of Safety and Dependability (S&D) this adoption is slower. This contribution reports about model based S&D engineering along the V-model from concept to operation phase in an industrial application. In the concept phase, the design engineer has the complex task to find an architecture fulfilling safety, reliability, availability and testability targets. We propose a qualitative modeling approach in which, during the early stage of design, models can be quickly built to support the full range of S&D analyses such as: computation of cause-effect relationships, automatic generation of FTAs, automatic generation of RBDs, system availability prediction or systematic evaluation of the Diagnostic Coverage. At this stage, the qualitative model supports the system and subsystem specifications' validation process and provides a systematic framework to reduce the risk of not meeting the RAMST targets. Later in the development cycle a full quantitative model is built (semi-automatically from importing the ECAD-net lists) using component model libraries. While maintaining drawing set consistency of analysis results, this final model quickly unveils the safety impacts of design changes, enables automated computation of even double-fault FMECAs and supports model-based diagnostics during operation of the system.

Introduction

With the publication of the Development Standards for IT Systems of the Federal Republic of Germany in 1997 the V-Model 97 entered into force as standard for all civil and military federal agencies. The current V-Model (V-Modell XT 2006) shown as the blue path in Figure 1 is designed as guidance for planning and executing development projects, taking into account the entire system lifecycle.

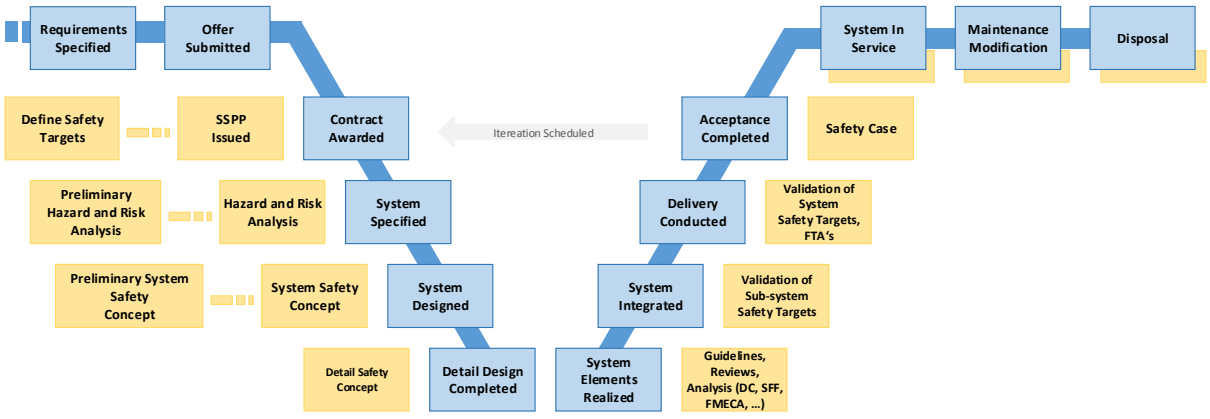

Figure 1: Safety Engineering Activities along the V-Model

The V-model defines the results to be achieved in a project and describes the actual approaches for developing these results.
All safety engineering activities, i.e. development of system functionalities to reduce all safety risks to an acceptable level, are allocated along the V-Model as illustrated in the yellow path in Figure 1. Starting from the concept phase,

the safety engineer has to find a safety concept to mitigate the outcome of the preliminary hazard and risk analysis. As information is highly dynamic and quick evaluations are needed, he needs to be embedded in the core project team. On the other hand, however, he has the task to find an S&D architecture fulfilling all safety, reliability, availability and testability targets and to define the full set of S&D related requirements. This ambivalent situation and the effort required for repeating and updating analyses and documentation often leaves little time for further iterations, resulting in sub-optimal designs. In this paper we present an approach, how the necessary S&D analyses such as cause-effect relationships, Fault Tree Analysis (FTA), Reliability Block Diagrams (RBDs), specification of Mean Time Between Failure (MTBF)-values, availability prediction or evaluation of the Diagnostic Coverage (DC) can be computed in a modular, quick and reusable way, thus freeing engineering resources for higher level tasks like system optimization.

Model-Based Analysis for the Design Phase: A Qualitative Approach

Introductory Example: In spite of many norms, standards and internal guidelines defining an obviously seamless process, safety development and assessment in many organizations is marked by many proprietary tool solutions and data repositories. Reuse of determined results in later analyses often requires manual work causing effort and potential consistency errors.

It is obvious that the advantages of centralized model-based support as already harvested in the classical system engineering process will also provide benefit when analyzing the faulty behavior of systems.

This section introduces a first view into a modeling approach considering local component faults and their propagation to potential failure of system functions in a purely qualitative way. Nevertheless, the approach already covers the full hierarchical system architecture and supports the full range of RAMST-analyses based on the same model being quickly set up. Especially during the concept phase, it supports system and subsystem specifications and definition of consistent safety requirements. Here we first introduce the modeling approach and the potential analyses along a small, illustrative example, before showing its benefit on an industrial example in a later chapter.

The example is the simplified redundant auxiliary power supply of a nuclear power station; see Figure 2a). The bus bar, that provides auxiliary power to the internal operation system (bottom right) is fed by the power generated by the turbine or alternatively from an external source (top). Here both switches are assumed to be activated.

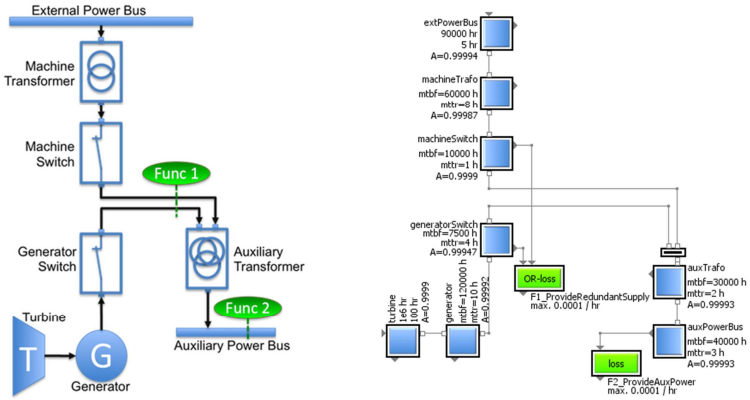

Figure 2: Auxiliary Power Supply: a) Block Diagram b) System model

The block diagram visualizes designated hardware components composed in an anticipated architectural design. However, system requirements specify not hardware solutions, but designated functions on top level. This functionality is provided by certain chains of hardware components at certain locations within the architecture, here indicated as green ovals in the block diagram (Figure 2a) respectively rectangles in the model (Figure 2b),

- Func 1 / F1 (OR-loss): “Provide redundant supply” (i.e. both switch outputs energized)
- Func 2 / F2: (loss): “Provide Aux power” (i.e. auxiliary bus is energized)

The state of the blocks F1 and F2 express the status of functional chains w.r.t. correct functionality. Each function can fail if the component at the end of the functional path doesn’t perform as expected. This is the case either a) if it

is defective or b) if one or several of its own suppliers fail. From this simplified reasoning viewpoint – similar to troubleshooting conclusions – the system behavior may be expressed by Boolean logic.
Dependent on the number of its local inputs and its internal logical behavior, we have implemented a library of generic model classes using a *Modelica*©-like approach (Modelica 2014).
This allows quick assembly and evaluation of anticipated architectural concepts by graphical instantiation of each system component in the modeling environment. As shown in Figure 2b), the model clearly reflects the original system structure, including the 2 functions, represented as rectangular boxes.
Model blocks of each element may be enhanced by additional parameters, like *mtbf* values or thresholds for failure probabilities for the top-level functions, see annotations in Figure 2b). Insight into the modeling approach and support of RAMST-analyses will be provided in the following sections.

General Functionality: The model example above describes each of the two power supply paths by chains of "*SiSo*"-models, representing components following the "single input, single output"-pattern. All classes in the library are structured in an object-oriented way, delegating general declarations to base classes and allowing any level of hierarchy. The following piece of code (see Figure 3) shows some features of the modeling approach.

```
                FailureMode     fm (max = 1, mapping = "ok, fails", rate = 1.e6/mtbf);
  parameter     Real            mtbf (unit = "h") = 10000 ; // Mean time between failure

behavior  // specification of functional dependencies of local quantities:
  outl.signal := ( (fm ==0) & inl.signal );
```

Figure 3: Example model code excerpt of *SiSo*-class

1. The declaration of a Discrete-valued Failure Mode-variable *fm* allows distinguishing different behavior modes of the component. With a value range starting at 0, indicating nominal behavior, in this example the maximal value is 1, here meaning that the component fails according case a) in the previous section. The variable type allows specification of several sub-properties, e.g. assigning the inverse of parameter *mtbf* as default of the failure rate. By default a value of 0 is assumed for *fm*, unless it is modified by the internal algorithm (see below) or externally by the user, allowing explicit fault injection into a system.
2. The specification of functional constraints between the interface and internal variables in the *behavior*-section of the model class. Whereas also complex dependencies can be specified, in case of the qualitative model library only one statement suffices to determine the Boolean value of the output-signal in relation to the input-signal and the local *fm*-value.

Dependent on the value of local variables specific graphical appearances of the model icons can be assigned. The blue color in the evaluated model shown in Figure 2b) indicates the expected nominal status "energized ".

Reliability Analysis: Based on a) the system topology assembled in the model diagram as shown above and b) the logical functionality assigned to the component classes and thus instantiated in each particular component model, the top-level functions can be analyzed automatically. After selecting their status-variable within a special Reliability-module of the model-based evaluation environment the inherent mathematical dependencies are traversed backward symbolically, up to the starts of the functional chains.
After optimizing the implicit Boolean structure the reliability analysis results for each function are shown graphically as interactively evaluable Fault Tree (see Figure 5a below) and Reliability Block Diagram (see Figure 4a) for each function. The numeric value table displays the specified max threshold of the failure probability, the actual value and the reserve (see Figure 4b) per function. Function specific highlighting of violated reliability values immediately points out weaknesses of the designated architecture.

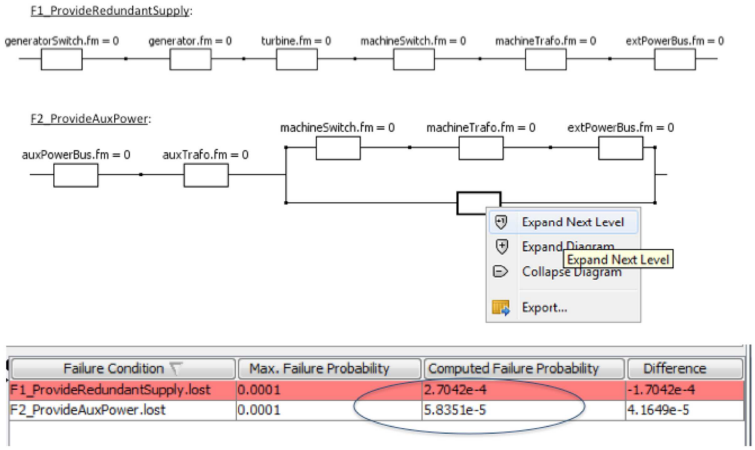

Figure 4: Results of Reliability Analysis a) Hierarchical RBD for F1 and F2 b) Comparison of failure probabilities

Safety Analysis: If the consequence of a malfunction in functional chains is safety critical, safety engineering has to assure that the risk of the occurrence of this malfunction is mitigated to an acceptable level. The engineer can modify the safety architecture or increase and decrease the reliabilities of the components to achieve the safety targets. The established method for this task is FTA, a graphical method supported by numerical evaluation engines, for the evaluation of the failure logic and the probability of system malfunctions.
The usage of the above described system model as a basis to calculate fault trees allows a more interactive and intuitive analysis method. Due to the component-oriented setup of the model, its hierarchical topology can directly be browsed in the model viewer. The environment not only displays computed values of system variables but also to set them interactively.
One option is to overwrite the expected value derived in a nominal-behavior simulation by a value observed in reality or just assumed. After having specified that function *F2_ProvideAuxPower* is lost (see Figure 5b), this obviously conflicting value nevertheless can be propagated through the network of Boolean – or more generally: physical – constraints.

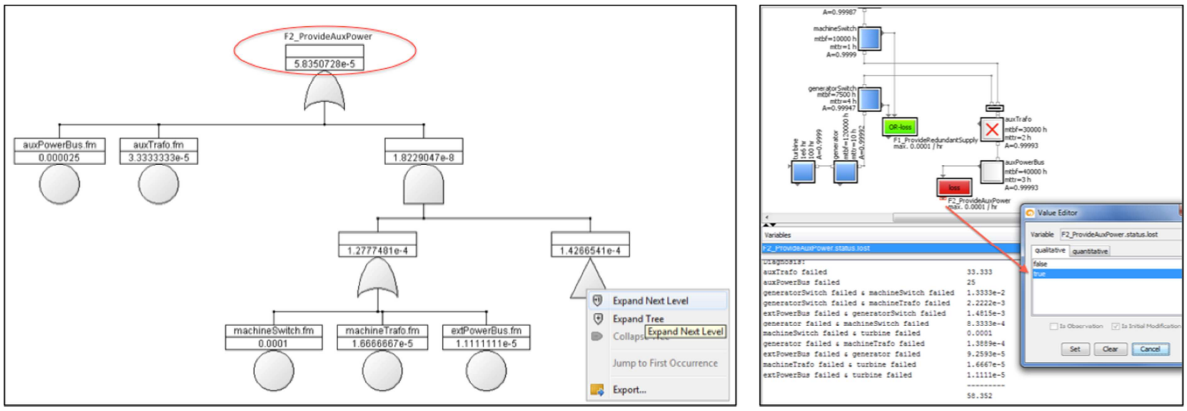

Figure 5: Safety Analyses for lost function F2 a) via Fault Tree b) via Model-based Diagnosis

Since the nominal behavior does not give an explanation, the inherent fault model of each component is also taken into account. The evaluation engine follows the extended General Diagnostic Engine "GDE+" for model-based diagnosis (Heller, Struss 2001). Automatically and systematically it checks the explainability of the specified symptom by any fault model or a combination thereof and lists them, together with the individual probability of this diagnostic candidate (see Figure 5b, bottom). Animations in the model graphics display the root cause component (here indicated with red X) and the affected system parts (here white for "de-energized components" or red for "lost functions"), giving the analyst an immediate impression on other lost branches or functions. The displayed probability sum of all root causes in *fpmh* matches the failure probability of the top event in the FTA above; compare top-node value as displayed in a) and total sum resulting in b).

Another method to support the engineer in early concept evaluation is the automatic consecutive activation of each modeled fault mode followed by a system simulation in a batch process and the logging of potentially interesting top-level behavior variables. From this pool of pre-computed analysis results automatically an FMEA-table can be generated and formatted according to tailored templates. Since this process is highly automated, consideration of single, double or higher order faults is just a matter of settings in the evaluation environment and computing power.

Testability Analysis: Once a component-oriented system model exists, any variable within the model may be analyzed for its benefit in testing the system resp. its contribution to Diagnostic Coverage. This is simply done by marking all the superset of potentially helpful variables by a textual identifier. Due to the object-oriented structure of the modeling language, each variable is considered as object and has sub-attributes (see also see Figure 3). Text labels can be arbitrary chosen or selected from some pre-defined categories, e.g. for “observations”, “fault codes”, “voltage measurements” (in full physical system models, see below), etc.
Simulating and logging a wide range of nominally or irregularly possible system behaviors by the same batch process as described in the previous section results in a database of analysis results. Smart filtering mechanisms allow to evaluate easily, if there are potential faults not detected by any measurement, leading to NFF-problems (no fault found) or if a specific sensor reading or BITE (built-in-test equipment) signal does not contribute to fault detection procedures at all and may be skipped, thus opening paths to reduce hardware in the system.

Availability Analysis: Finally the qualitative model also allows availability analysis of defined system functions early in the development process. In addition to the already defined *mtbf*-parameter another parameter representing *mttr* (mean time to repair) is added to the declaration part of the component base class. As an extension to the Boolean behavior description mentioned above (see Figure 3) also the local arithmetic to compute the numerical values of availability A and non-availability N can be assigned to the generic model classes. Their standalone values for this particular component are determined according the well-known equations (1) and (2).

$$A = mtbf / (mtbf + mttr) \quad (1)$$

$$N = mttr / (mtbf + mttr) \quad (2)$$

Besides these, the availability of the output signal of this component also depends on the supply availability according to the arithmetic shown in Figure 6. Results of this extension in the context of the overall model can be seen in Figure 2b).

```
// --------------------
// Availability layer:
// --------------------
    outl.A := A * inl.A;
    outl.N := 1 - outl.A;
// ========================
end SiSo;
```

Figure 6: Arithmetic to compute Availability (*A*) and Non-Availability (*N*) within component class

Industrial application - missile system safety architecture: Figure 7 shows the application of the qualitative method for the safety concept of a guided missile with the subsystems (from left to right): seeker (not modelled, as not safety critical), weapon computer, data link, power supply, SAU and warhead, dual stage propulsion and actuator. The main system hazards, like premature warhead initiation, inadvertent motor ignition and malfunction of flight termination are modeled as external function-blocks as described in the introduction (see Figure 7, upper right part).

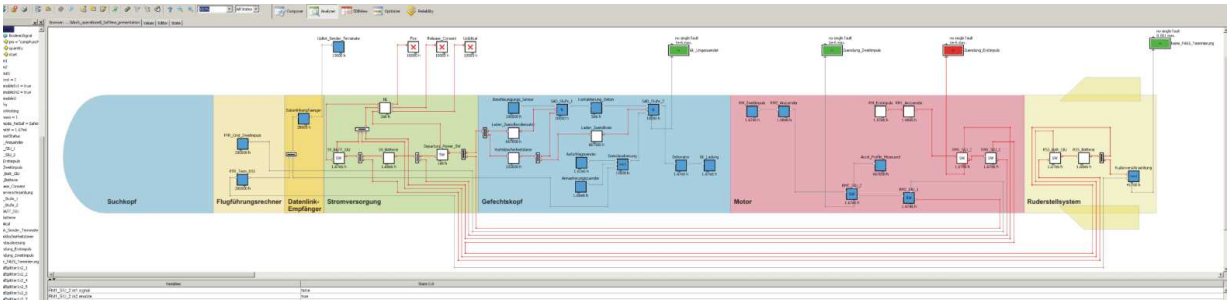

Figure 7: Qualitative Missile Safety Architecture

The fault trees of these events can be generated and displayed on the fly and the safety engineer can evaluate the results interactively. In Figure 7 the consequence of the inadvertently set input signals “release consent”, “fire” and “umbilical separation” (the three boxes with red X) are analyzed correctly to fire the booster motor. This example shows the advantage to interactively analyze the consequence of selected house events i.e. component failures. Figure 8 shows the computed fault tree for the system hazard “warhead detonation”. It would also easily be possible to pre-inject faults into the virtual system and evaluate the remaining safety or reliability margin.

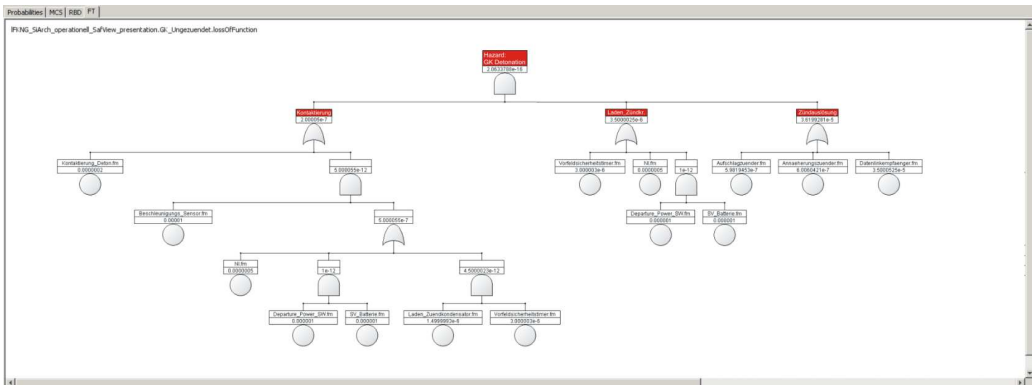

Figure 8: Computed Fault Tree for Hazard "Unintended Initiation of Warhead"

Once the model is established and approved in discussions with the relevant component specialists, the safety engineer can interactively optimize the system safety architecture. He can then determine and consolidate the safety concept and specified values for the component reliability properties.

Model-based Analysis for the Realization Phase: A Quantitative Approach

After the development of the system has started with the construction and implementation of hardware and software components (i.e. the design phase is moving up on the right side of the V-Model), the methods for safety engineering change significantly. It is no longer sufficient to continue with the qualitative model. The needed models from this phase on have to represent the physical behaviour with its nominal and erroneous functionalities.

Quantitative Modeling of components: Following the approach for physical system modeling similar to *Modelica*©, the physical behavioral description of a certain type or category of elements is specified in exactly one place, i.e. the model class representing this category. This class description contains a declaration part and a behavior-part.

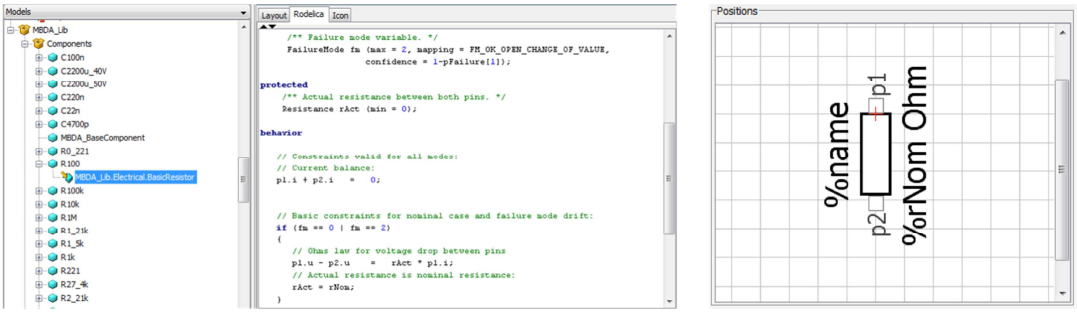

Figure 9: Physical component modeling: a) Model Library b) Code for physical model c) Graphical icon definition

All local variables, parameters and other internal properties are declared in a typed manner, as well as the external interfaces. These ports may recursively contain variables themselves – e.g. for voltage *U*, current *I* in electrical or force *F* in mechanical applications – and are the handles that allow to connect instances of separate classes to each other and to communicate and propagate values. As can be seen in the upper part of Figure 9b), the failure mode variable *fm*, already introduced in Figure 3, in this example is declared to indicate two fault modes, both of which may be assigned a value for their occurrence probability (i.e. the failure probability).

Each behavioral mode is described by its mathematical-physical constraints in the behavior section of the class. The example code in the figure shows a current balance equation – referring to the current symbols *i* of the two ports *p1* and *p2* – and in the lower part the formulation of Ohm's law, bound to the condition that it is only applicable for fault modes 0 and 2. It is important to understand that all internal descriptions are only based on local information.
Besides internal behavior description also the external appearance can be defined, i.e. the icon representing instances of this class in the superior system model together with its legend and other optional decoration; see Figure 9c).
This approach of standardized and context-free model classes is a prerequisite to assemble the knowledge about technical part categories in centralized model libraries, from where they easily can be reused in future projects. This not only improves engineering efficiency, especially in developing variants or updates of a system, but also conserves the expertise of system engineers in a modular way. Based on already existing generic libraries a user-specific adaption and extension has been set up in context of this missile project, see Figure 9a), *MBDA_lib*".

Industrial application - Assessment of a cable cutting system: The quantitative modeling methods described above were applied to analyze the firing control system of a missile. The system consists of several connected circuit boards, a simplified signal transmission path for the commands received from ground station and several actuators. Due to the high number of electronic components on the boards their modelling process was supported by an automated method to import topological information incl. instance names seamlessly from the existing ECAD-netlists. Driven by the bill of materials, model classes have been set up to map all required component categories. The physical behavior descriptions were then added for the nominal behavior as well as for one or several fault modes, including their specific probability. While the internal topology of the boards came with the import, their external graphical appearance and their arrangement on top level of the system model was done manually.
To validate the model several datasets for typical nominal scenarios were applied, forcing the system through sequences of internal states to defined actuator behavior. After successful validation the datasets have been extended to describe hazardous events like "no termination on command" and the model-based diagnosis function (see above) was used to determine possible root causes of this event and the resulting probability.
Figure 10 displays the system top-level view, as inset the internal view of one board and as another inset the icon of a MOSFET as one of the two single-fault candidates. The suspected components are highlighted in the system hierarchy and their explaining fault modes are displayed in the result console (bottom).

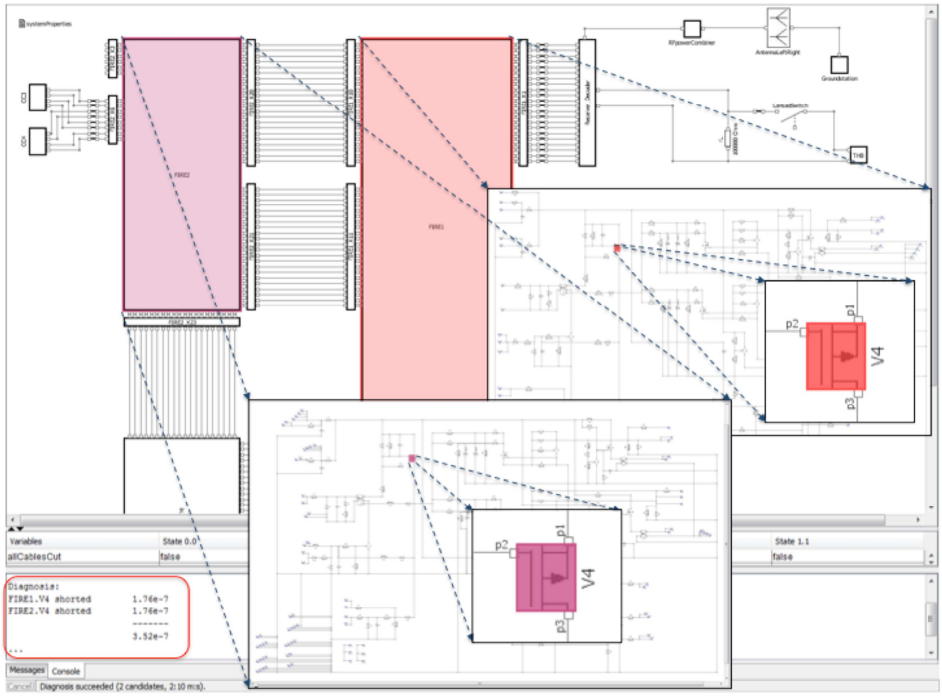

Figure 10: Hierarchical system with display of two candidates from a model-based diagnosis.

The diagnostic analyses resulted in complete lists of all single- and double-fault root-causes for the injected hazardous events. The determined overall failure probabilities were successfully compared with existing figures resulting from a “manual” historical FMECA. This comparison added trust to the model-based FMECA methodology. The project was a success as it has formed the analysis basis for future design modifications of the system and proved the model-based approach.

Conclusions

This paper describes a model-based methodology for safety and dependability analysis. The creation process of the model assures that the safety engineer is in the state of having a full and complete understanding of the system. Another advantage is that he can use the model in the design team for the discussions of nominal and erroneous system behaviours with the component specialists. In the projects, where this method was applied, the model soon turned out to be used as the “golden reference”. Once the catalog (i.e. parts lists) of used devices is settled, subsystem modifications are easily updated via the above described ECAD netlist import functionality around the method. This assures drawing set consistency and rapid turnaround times of the analysis cycle in order to be able to evaluate proposed design modifications for Change Control Board decisions.
Within the projects, limitations of the model-based approach were also identified to be in the complexity of systems electronics hardware. Whereas the modelling of discrete electronics (resistors, capacitors, switches, diodes, transistors, ...) was mastered, it was difficult to develop models for components with complex integrated circuits (e.g. DC/DC converters, FPGA). On the one hand, the character of the realized safety critical functionalities helps to mitigate the performance drawbacks, as they tend to be realized as simple as possible. On the other side the usage of complex elements e.g. anti-fuze FPGA technology in safety critical functionalities becomes more and more prevalent, so models have to catch up. First modeling steps to cover the logic realized in an FPGA were successful.
The described method was applied for safety assessments in several projects within MBDA Germany and in various research projects. It has been proven that the model-based approach is a promising way ahead. Although modelling work forces the safety engineer to spend extra effort at project start, this investment pays back by the process ensured complete understanding of the assessed system, leading to a higher degree of confidence in the analysis results. In addition payback for the initial extra efforts for the model-based safety assessment comes from faster analysis results after design iterations.

Acknowledgements

We very much want to express our thanks to the inventor and initiator of the reasoning system, Dr. Werner Seibold, for his vision and power in building the basis for the described model-based approach and Beate Frey for her great work in setting up the model library and assembling the system models of our projects. We also want to thank Falk Müller and Dr. Gerhard Elsbacher of MBDA Deutschland GmbH for their trust and support towards a new approach. Then there are of course many colleagues and partners in uncountable discussions that we want to express our deep gratitude for their scepticism and encouragement, without which this work would not have been possible. And last but definitely not least, there are our families, who always have to pay for the extra time spent at work.

References

1. V-Modell® XT, Version 1.3 English, "Fundamentals of the V-Modell" February 9, 2009, 18. Accessed May 15, 2015. http://v-modell.iabg.de/dmdocuments/V-Modell-XT-Gesamt-Englisch-V1.3.pdf.

2. Heller Ulrich and Peter Struss. "GDE+ - The Generalized Diagnosis Engine." *Proceedings of the 12th International Workshop on Principles of Diagnosis*, Via Lattea, Italy, March 7-9, 2001.

3. Modelica Association "Modelica - Language Specification Version 3.3, rev.1" July, 2014, https://modelica.org/documents

4. Lunde Karin, Rüdiger Lunde and Burkhard Münker "Model-Based Failure Analysis with RODON", *Proceedings of the ECAI 2006 - 17th European Conference on Artificial Intelligence*, Riva del Garda, Italy, August 29 - September 1, 2006.

Biography

Dieter Fasol: MBDA Deutschland GmbH, Hagenauer Forst 27, D-86529 Schrobenhausen, Germany, Tel: +49 8252 99 8158, email: dieter.fasol@mbda-systems.de

Dieter Fasol is a safety engineer at MBDA Deutschland in the role of the "project responsible for product safety" on several projects with experience in the safety field of nearly ten years. His experience in the company as a development engineer for missile algorithms and simulations has a history of over 25 years. Prior to his engagement in safety he led the design team for guidance and flight control algorithms on a long term missile project. On this project he gathered experience with safety critical functional chains.
Dieter graduated in 1987 from the Ruhr-Universität-Bochum, Germany with receiving the Dipl.-Ing. for Mechanical Engineering in the field of Control and Automization.

Burkhard Münker, icomod – münker consulting, Olper Strasse 53, DE-57258 Freudenberg, Tel: +49 2734 5503897, email: burkhard.muenker@icomod.de, www.icomod.com
Burkhard Münker is a freelance service provider of model-based services, system- and safety-engineering. He works as author, coach and consultant in various industries with the key focus to optimize system development processes and to support safety management by the application of model-based technologies and tools. Prior to that role, he was engaged for many years in the development and application of model-based diagnosis and reasoning technology, being responsible e.g. for fault and risk analyses and new modeling concepts according to the Modelica© language.
Burkhard studied Mechanical Engineering and received his PhD from the Technical University of Berlin for developing a tool for automatic generation of dynamic models to observe chemical reaction systems. He propagates the idea of model-based, collaborative engineering and is interested to adapt proven solutions from modeling and analysis to new application fields. For several years he has also been a guest lecturer on physical system modeling and model-based fault analyses and diagnosis at the University of Siegen, Germany.

Peter Bunus: ESI-Group, Sjöängsvägen 15, SE-192 72 Sollentuna, Sweden. Tel: +46 70999 1874, email: *peter.bunus@esi-group.com*
Peter Bunus is a Director at the Center of Excellence in Virtual Electronics and Systems at ESI-Group, a company specialized in virtual product engineering software and services. In his current position he is managing the execution of systems prime solutions project related to model-based and system engineering solution portfolio. Prior to joining ESI-Group, Peter worked as a Research Scientist at Palo Alto Research Center and also held an Associate Professor position in Computer Science at Linköping University Sweden where he taught Computer Science and Software Engineering courses and performed research in the area of model-driven engineering.
Peter has received his Ph.D. in Computer Science from Linköping University Sweden in 2004. He has published over 40 peer-reviewed articles and papers in leading journals and conferences in the field of modeling and simulation and program analysis and verification. Peter has also served as co-chair or program committee members of more than 15 conferences in his area of research and he is actively involved in the modeling and simulation community. Currently, his research is focused on modeling techniques, modeling language development and model-driven engineering with direct applications to diagnostics and automated program and model verification.