AN INTEGRATED TOOLCHAIN FOR MODEL BASED FUNCTIONAL SAFETY ANALYSIS

Muhammed Zoheb Hossain[(a)], Olena Rogovchenko [(b)], Mattias Nyberg[(c),] Peter Fritzson[(d)]

[(a)] Scania, Sweden
[(b)] Linköping University, Sweden
[(c)] Scania, Sweden
[(c)] Linköping University, Sweden

[(a)]zohebmuhammed.hossain@scania.com, [(b)]olena.rogovchenko@liu.se, [(c)]mattias.nyberg@scania.com, [(c)]peter.fritzson@liu.se

ABSTRACT

The significant increase in the complexity and autonomy of the hardware systems renders the verification of the functional safety of each individual component as well as of the entire system a complex task and underlines the need for integrated, model based tools that would assist this process. In this paper the authors present such a tool, coupled with an approach to functional safety analysis, based on the integration of functional tests into the model itself. The analysis of the resulting model is done through a stochastic Bayesian model. This approach strives to both bypass the necessity for costly hardware testing and integrate the functional safety analysis into an intuitive component development process.

Keywords: Bayesian Networks, Safety Analysis, Model-based Design, Functional testing

1. INTRODUCTION

Functional safety is a key concern in all industry sectors, be it nuclear plants, medical appliance manufactures or the automotive industry. The functional correctness of a component is the guarantee that the component behaves the way it should and fulfills all the functional requirements of the system. In order to ensure functional correctness of a component it is necessary to perform a series of rigorous tests on the target device in the appropriate environment context. Skipping this phase and allowing for a component to be tested based on its design specification, without an actual hardware implementation, would make a significant contribution to reducing the skill, labor, time and money required to develop the component.

In this paper we present a novel approach to Functional Safety verification, where we integrate functional tests as full-fledged components into a model based architecture developed using OpenModelica (Fritzson 2004). This model can then be used to generate a stochastic Bayesian model which in turn can be used to produce a Failure Mode Effect and Analysis (FMEA) table.

2. A COMBINED MODELING APPROACH

2.1. Bayesian Networks

Bayesian Networks (BN) allow for the specification of risk models that represent the key factors and their inter- relationships (a qualitative model) with probability distributions based on expert judgment or from observed data (a quantitative model). Bayesian Networks are already used for the verification of functional validity, but the existing approaches require the use of dedicated tools which means that there is a gap to be breached between the tool used to design and program the component and the verification tool. Our approach is to combine in a single tool suite the design and verification stages of the development process.

2.2. Failure mode and effect analysis

Failure modes and effects analysis (FMEA) (McDermott et al. 2009) is a step-by-step approach to identifying all the possible failures in a design. With this approach failures are prioritized according to how serious their consequences are, how frequently they occur and how easily they can be detected. FMEA is applicable right from the conception stages of a component and throughout its entire life-span. This approach is particularly popular with the automotive and aerospace industries. The FMEA table produced by analyzing the component model can thus be used to predict possible failures and prevent them right in the design stages.

2.3. Component Modeling

Nowadays a large choice of design and simulation tools is available. Tools like Matlab, Simulink or SimulationX provide efficient support for the mathematical aspects of component modeling. However these tools lack the support for intuitive modular design aspects. OpenModelica, on the other hand, provides a complete modeling editor (OMEdit) as well a structured, intuitive approach to modeling and simulation of complex multi-domain systems in the

Modelica language. For this reason we chose the free and open-source OpenModelica platform for our implementation.

3. USE-CASE SCENARIO

To illustrate our modeling approach we have chosen to model a Magnetic Valve that is used as a sub-component to start the ignition of an automotive vehicle. Figure 1 provides a representation of the complete model. The rectangular blocks represent the various components and the squares with a circle in the center are the services associated either with an individual component or with the whole environment.

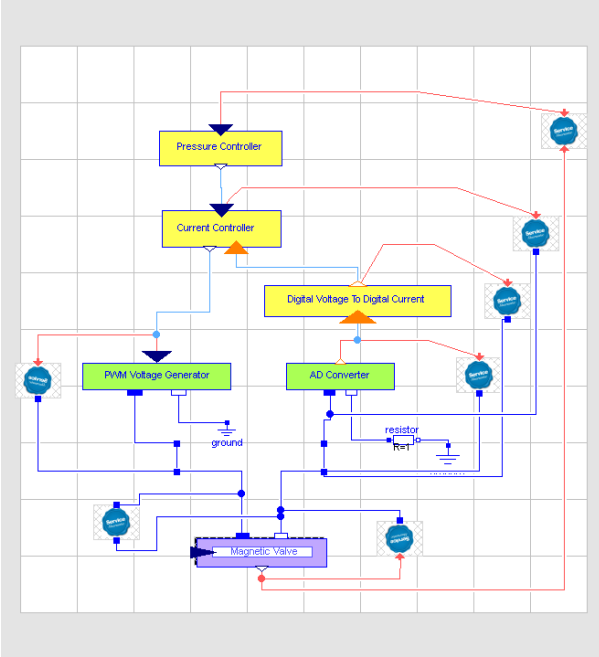

Figure 1: Magnetic Valve Model

The components and the services are all modeled through classes, in the sense of classical Object-Oriented programming languages. Services are simply a special kind of components that formalize what is required from one component by another in order to perform the task correctly. These components are interconnected through their interfaces.

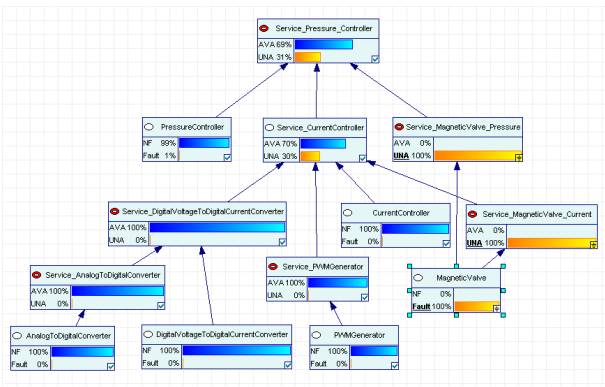

Figure 2: Model with a fault

Based on the diagram in Figure 1, a dependency graph will be generated, by analyzing the dependencies of the services provided on the different components of the model. We then introduce probabilities at the leaf nodes and perform inference over the Bayesian Network which allows us to analyze how and where a node is affected when the node 'PWM Generator' fails.

In the first scenario in Figure 2, we analyze the probability of a component's success and failure, upon fault injection in leaf nodes. Once the Bayesian Network is generated it is automatically opened in Genie, a viewing tool for Bayesian Networks. Here, one can insert probability values into the nodes of the network in order to carry out an inferencing over the network. Thus in Figure 2, we have injected a fault into the Magnetic Valve node, which lies at the bottom right of the figure and after inferencing we can see that the fault is propagated to all the nodes that are dependent on the Magnetic Valve node.

In the second scenario in Figure 3 we investigate the probabilities of the model's success and failure when no fault is injected. For this purpose, we have kept the model free of fault and hence we see a clean slate reflection of our model. The "AVA", "UNA" and "NF" represents Available, Unavailable and No Fault respectively.

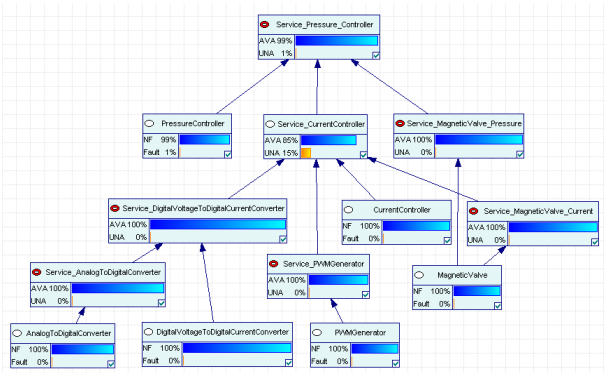

Figure 3: Model with no faults

Once we have the inference of the Bayesian Network we generate the FMEA table with the following information: Failed Component, Failure Mode, Potential Effect of Failure and Severity of the Failure. This table helps the designers or engineers to adapt their design in order to mitigate the possibility of the failure mode.

4. IMPLEMENTATION

The implementation of the tool presented in this paper relies on C++ code for the generation of the causal nodes for the components of the model, which are written in Modelica by using the graphical editors provided by the OpenModelica tool suite.

Previously the implementation relied on Matlab code for the generation of Bayesian Networks, but since one of our goals is to create a free and open source tool-chain, this code was ported to C++, which required some minor alterations to the OpenModelica API.

4.1. Model-based design implementation

First of all, we needed some means to relate the Services with the corresponding Behavior-components in terms of the order of message flow in the system. In OpenModelica it is not possible to assign causality in hardware components, hence we had to add an optional dependency parameter in the API through which it would become possible to get our desired information. The following example syntax corresponds to the components of our model:

```
annotation(
__OpenModelica_ComponentsHierarchy =
{{"Pressure Controller","Current Controller"},
 {"Current Controller","PWM Voltage Generator"},
{"PWM Voltage Generator", "Magnetic Valve"},
{"Magnetic Valve","AD Converter"},
{"AD Converter","DigitalVoltageToDigitalCurrent
Converter"},
{"DigitalVoltageToDigitalCurrent Converter","Current
Controller"}});
```

The component names inside the curly braces are to be read as, "Pressure Controller is parent to Current Controller", "Current Controller is parent to PWM Voltage Generator", and so on.

Now, the model also contains the service to behavior relationship, from which we can derive the overall dependencies for the Service-Behavior relation of the model. Later on, we use this information to generate the Bayesian Network of the corresponding Modelica model.

In order to make our model in Modelica adaptable for future extensions we have created base classes for the Service components. The base classes consist of all the basic attributes the Service components are required to have. The basic attributes in this case are the Behavior_Input interfaces i.e. the input interfaces that take Success/Failure values from the behavior components they are connected to. As well as Service_Input and Service_Output interfaces where the input interface takes in input from the Service on which this Service is dependent on, and the output interface gives out dependency information to the Service it is parent of.

4.2. Failure verification implementation

The implementation also relies on the Structural Modeling, Inference and Learning Engine (SMILE) library (Druzdzel 1999) for the creation of the Bayesian Network and the companion tool GeNie for visualizing the resulting BN. Figure 4 illustrates the relationship between the various tools.

The joint probability distribution can answer any questions regarding the domain of some random variables but can become intractably large as the number of variables grows. Independence and conditional relationships among variables can greatly reduce the number of probabilities that need to be specified in order to define the probability distribution.

Once the Bayesian network is generated and visualized in Genie, we can introduce probability values in the component nodes, i.e. the non-service nodes. Upon inclusion of the Success/Failure probabilities, we can perform inference over the BN in order to troubleshoot the model.

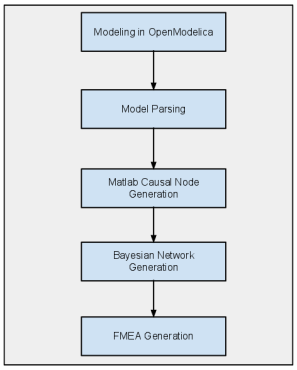

Figure 4: Workflow

Building the network consists of three tasks. The first of these is to identify the variables of importance along with their possible states. Once these are identified the next task is to identify the relationships between the variables and to express them through a graphical structure. The third and final task is to obtain the probabilities for the quantitative part of the network.

The troubleshooting problem in the context of a model can be expressed as follows: “Given a set of observations, which component is the most likely to be faulty?”. A Bayesian Network is well suited for this kind of reasoning under uncertainty. We have used FMEA, a top-down approach, to evaluate out model, assigning failure probability to the top node along with probabilistic values in the depending lower nodes. The faulty components are then computed on the condition that the top component level has failed to perform its task.

5. RELATED WORKS

The advantages of applying model based design techniques to the field of reliability and safety analysis are clear, however so far only a few works in the domain exist.

One of these is RAMSAS, a model-based method for system reliability analysis, which combines the power of modeling languages such as UML or SysML with the Mathworks simulation environments. It allows to analyze the system as a whole by decomposing it into

subsystems. This method is centered around the classical iterative engineering process with four steps: Objectives Definition, System Modeling, System Simulation and Results Assessment.

Although united in the goal of applying model based techniques and combining the advantages of established modeling tools with more formal analysis methods, there are some differences in the two approaches.

The RAMSAS approach (Garro et al. 2012) relies on simulation based analysis and verification against description of the required behavior, whereas in our work we rely on static analysis and probabilities to run through possible failure scenarios.

Another distinction is that in our implementation we made a voluntary decision to rely on free and open source software, whereas the RAMSAS implementation relies on proprietary tools.

6. CONCLUSION AND FUTURE WORKS

Model-based design brings the advantages of a modular, object-oriented language for system design such as Modelica to the safety verification process.

Choosing Bayesian Networks for diagnostics in the context of functional safety verification on the other hand has both advantages and drawbacks.

A clear advantage is the direct correspondence of the network nodes to the real world components of the ECU, this makes the model more reliable and easier to modify.

The use of expert knowledge should not be underestimated since the probabilities are outcomes of expert knowledge, however it may become counter-effective when several experts are involved and a large amount of probabilities has to be elected.

In our current implementation we output only the basic information that is needed to make the FMEA worth using. In our future work we would like to add more specific fields like "Severity Rating", "Occurrence Rating" and "Detection Rating" and eventually these three parameters will lead us in calculating the Risk Priority Number (RPN).

The next big step is to implement fault-tolerant control and diagnosis through a service oriented architecture in a more complex and realistic systems, and use the results as a basis for comparison with other academic and/or commercially available tools such as Hip-Hops (Papadopoulos et al. 2001) to provide an evaluation of the model.

Another work in progress is the generation of a Fault Tree Analysis (FTA) (Clifron 1999), a top-down, deductive failure analysis in which an undesired state of the system is analyzed using Boolean Logic to combine a series of low-level events. This approach will complement the current bottom-up analysis scheme.

The current case studies are based on the automotive industry standards, and international standards such as IEC61508 and ISO26262 require both FTA (fault tree analysis) and FMEA to be done. The proposed approach will be of great help since it will allow to perform them automatically or at least semi-automatically.

In a general manner, this work is part of an ongoing project with a larger context for developing a model-based approach for system-requirement verification and fault tolerance.

ACKNOWLEDGMENTS

This work has been supported by the Swedish Strategic Research Foundation in the EDOp and HIPo projects and Vinnova in the RTSIM and ITEA2 OPENPROD projects. The Open Source Modelica Consortium supports the OpenModelica work.

REFERENCES

Clifron, E. (1999). Fault tree analysis - a history. In Proceedings of the 17th Inernational Systems Safety Conference.

Druzdzel, M.J. (1999). Smile: Structural modeling, inference, and learning engine and genie: a development environment for graphical decision-theoretic models. In Proceedings of the sixteenth national conference on Artificial intelligence and the eleventh Innovative applications of artificial intelligence conference innovative applications of artificial intelligence, AAAI '99/IAAI '99, 902–903. American Association for Artificial Intelligence, Menlo Park, CA, USA.

Fritzson, P. (2004). Principles of Object-Oriented Modeling and Simulation with Modelica 2.1. Wiley-IEEE Press, 1., auflage edition.

A. Garro and A. Tundis, March 2012, A Model-Based method for System Reliability Analysis, Proc. of the Symposium On Theory of Modeling and Simulation (TMS), Orlando, FL (USA)

McDermott, R., Mikulak, R., and Beauregard, M. (2009). The basics of FMEA. CRC Press.

Papadopoulos, Y., McDermid, J., Sasse, R., and Heiner, G. (2001). Analysis and synthesis of the behaviour of complex programmable electronic systems in conditions of failure. Reliability Engineering System Safety, 71(3), 229–247.