# Computerized Model Based Functional Safety Analysis

**Muhammed Zoheb Hossain** [*] **Mattias Nyberg** [**]
**Olena Rogovchenko** [***] **Peter Fritzson** [****]

[*] *Scania, Sweden, (e-mail: zohebmuhammed.hossain@scania.com).*
[**] *Scania, Sweden, (e-mail: mattias.nyberg@scania.com)*
[***] *Linköping University, Sweden, (e-mail: olena.rogovchenko@liu.se)*
[****] *Linköping University, Sweden, (e-mail: peter.fritzson@liu.se)*

**Summary**

**Abstract:** With the growing complexity of the hardware models, the verification of the functional safety of each individual component as well as of the entire system becomes increasingly complex. In this paper the authors present a novel approach to functional safety analysis, based on the integration of functional tests into the model itself and the analysis of resulting model 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, Modeling

## 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 interrelationships (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 modes and effects 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*

Now-days 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. For this reason we chose the OpenModelica platform in our implementation.

## 3. IMPLEMENTATION

The proposed tool is implemented in C++ and Matlab is used for the generation of the causal nodes for the components of the model, which are written in Modelica

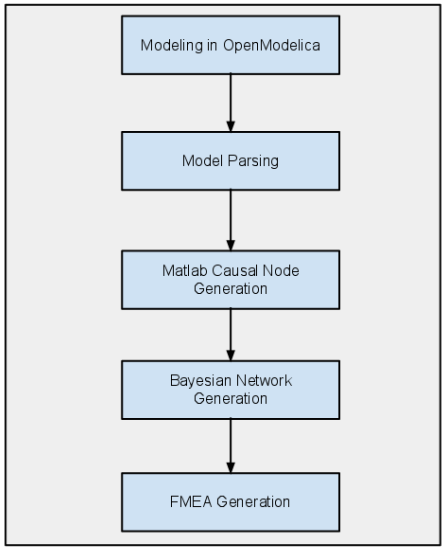


Fig. 1. Work Flow

using the OpenModelica tool. 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 1 illustrates the relationship between the various tools.

The joint probability distribution can answer any question about the domain of some random variables but can become intractable 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.

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.

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 2 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.

The components and the services are all modeled by classes, in the sense of classical Object-Oriented programming languages. These components are interconnected through their interfaces.

Figure 3 illustrates the dependencies between the nodes, where the nodes with double ring are the Service Nodes and the nodes with single ring are the actual hardware and software nodes. The arrows represent the dependencies and the direction of information flow between the connected nodes.

We 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. Figures 3 and 4 show how other components will be affected by a faulty analog to digital converter unit and a faulty magnetic valve respectively. Figure 5 shows the inferences for a fault free model, where an unavailability is observed at the top level of the controller. The inferences help detect the probable causes of the observed unavailability.

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.

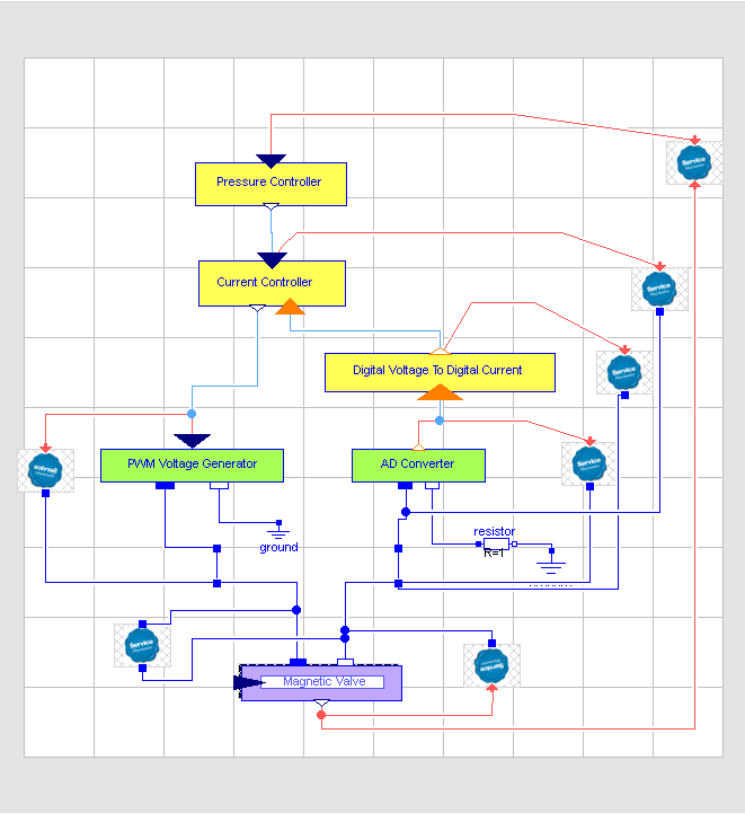


Fig. 2. Magnetic Valve

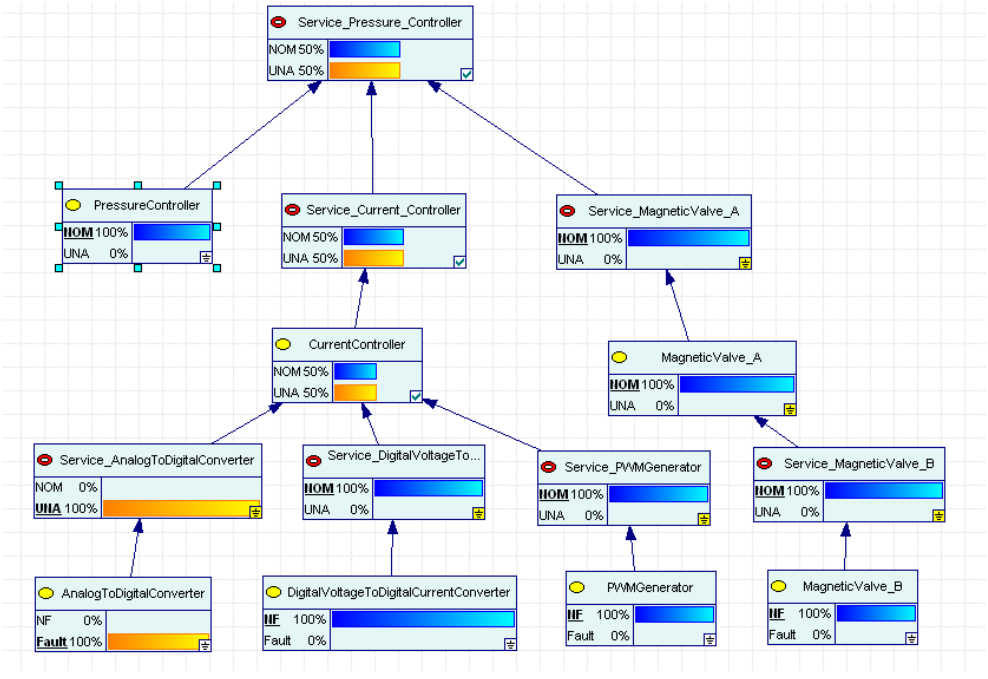


Fig. 3. A model with a faulty analog to digital converter component

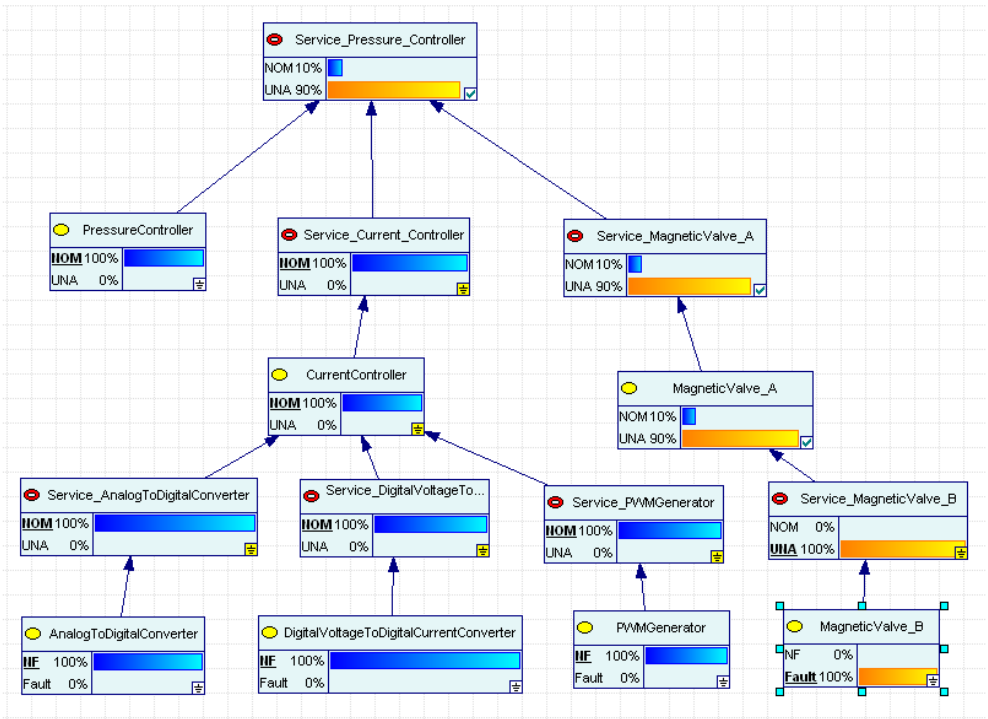


Fig. 4. A model with a faulty magnetic valve

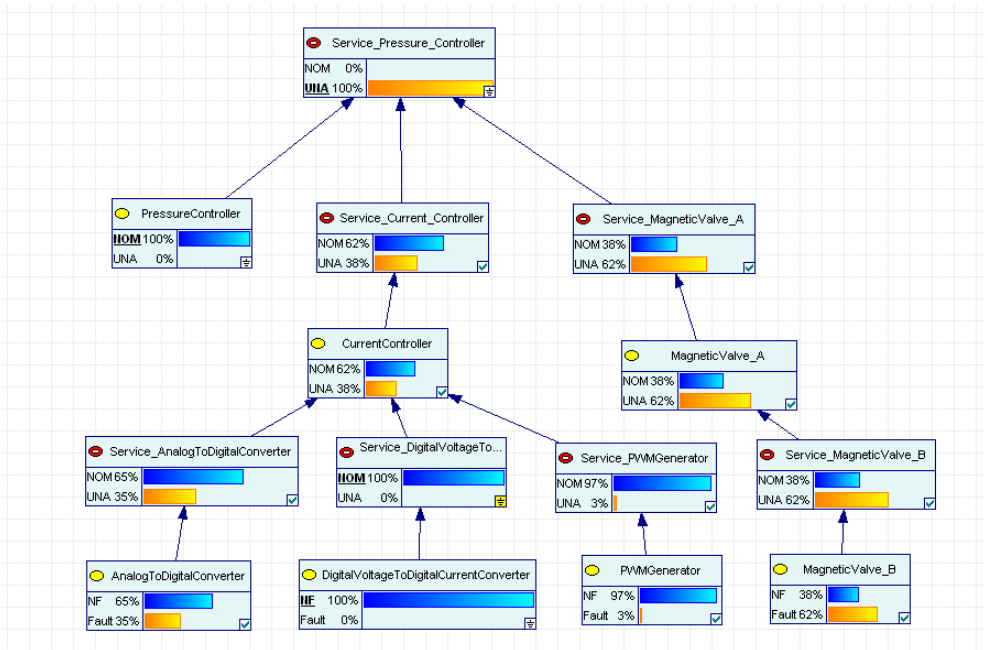


Fig. 5. A fault free network with an unavailability observed at the top-level

## 4. TOWARDS A MORE STABLE IMPLEMENTATION

The current implementation choice of combining Matlab and C++ and setting up an interface between Matlab and a C++ program is quite cumbersome as well as error prone at times.

Moreover the combined architecture is very sensitive to individual versions and the platform on which they are installed and executed. For instance in our work we have used Visual Studio 2008 and Matlab 2008b on the Windows XP operating system, running on a Pentium 4 machine. Changing the version number of either of the tools or changing the operating system to a newer generation of Windows requires a nontrivial implementation effort. This means that the tool is not easily portable.

Therefore we are now working on eliminating the Matlab interface, in order to depend solely on the C++ implementation for the generation of the causal nodes from the modelled Modelica components. This will not only greatly reduce the work load of maintaining future versions of ME tool but also reduce the complexity of coding and optimization.

## 5. CONCLUSION

Using Bayesian Networks for diagnostics in the context of functional safety verification 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 facilitates the maintenance of the model since the addition/removal of a real world component translates to the addition/removal of corresponding network nodes.

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

## 6. FUTURE WORK

In the current implementation we depend upon the commercial tool Matlab for the generation of the causal nodes. In order to have a fully open-source and more portable distribution, Matlab will be replaced by a node generation algorithm implemented in C++, which is now under development.

The next step, is to implement fault-tolerant control and diagnosis through a service oriented architecture in a more complex and realistic system, as opposed to the simplified example used for this paper. This can be then used 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.

## 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.

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.