# D2.2.6 – BN/FTA/FMEA prototype in OpenModelica

## Final M45 Deliverable. Reliability modeling and analysis through OpenModelica and Figaro integration

## MODRIO (11004)

**Version** 1.0

**Date** 23/04/2016

**Authors**

| | |
|---|---|
| Lena BUFFONI | LiU |
| Peter FRITZSON | LiU |

## Executive Summary

This document presents the prototype developed in OpenModelica to support the generation of reliability models using Figaro models especially from augmented Modelica models. This is an even more general approach to capabilities such as BN/FTA/FMEA (Bayesian Network, Fault tree analysis, Failure mode and effects analysis). Reliability properties are expressed in Modelica[1] models by using dedicated classes with Figaro specific information. Such models are then exported into a Figaro[2] model, which can be then used to generate different models for reliability analysis. A more general presentation of this approach (applicable to other Modelica tools as well) is available in a document written by EDF (Electricite de France) et al for deliverable 2.2.1 "Specification of Modelica extensions and interfaces for Bayesian networks and Fault trees"[4].

# 1. Abstract

This document presents the prototype developed in OpenModelica to support the generation of reliability models using Figaro models from especially augmented Modelica models. This is an even more general approach to capabilities such as BN/FTA/FMEA. Reliability properties are expressed in Modelica[1] models by using dedicated classes with Figaro specific information. Such models are then exported into a Figaro[2] model, which can be then used to generate different models for reliability analysis. A more general presentation of this approach (applicable to other Modelica tools as well) is available in a document written by EDF et al for deliverable 2.2.1 "Specification of Modelica extensions and interfaces for Bayesian networks and Fault trees"[4].

In order to implement the prototype, the OpenModelica compiler has been extended with an "export to Figaro" functionality[3] and the OpenModelica Graphical Editor interface has been extended to include graphical support for configuring and exporting Modelica models.

# 2. Introduction

A reliability model represents the probability distribution of all possible combinations of component failures that lead to a system failure. Fault trees and Bayesian networks in general are examples of reliability models.

Instead of having several different reliability models, Figaro was developed by EDF (Électricité de France) to be a general representation formalism. Figaro is a reliability modeling language. Although there is a strong connection to logic, Figaro is not entirely logic-based, as it is also influenced by object-oriented languages. Figaro is mainly used with its graphical user interface KB3. To do modeling in Figaro, there are essentially three steps. First, you develop a library (knowledge base). Second, you make a so-called list of objects that will be in the model. Third, you invoke the Figaro compiler to make a full model in Figaro 0 out of the list of objects with respect to the database. That is, you instantiate the objects. Figaro 0 is a sublanguage of Figaro. The full model can then be used to generate different reliability models such as a fault tree.

An example of a Figaro code snippet with an object definition:

```
OBJECT link1 IS_A bi_dir_link;
INTERFACE
    extremity
        =n1 Source;
```

Modelica models can be used for dynamic model verification through simulation. By enriching them with Figaro properties, these models can then be used to generate reliability Figaro models, which in turn can be used for static reliability analysis.

Generating these models automatically simplifies the verification process, improves code maintainability as there is an automatic mapping between the Modelica and reliability Figaro models, and reduces the risk of introducing errors compared to a manual transformation.

# 3. Figaro Export in OpenModelica

A more detailed description of Figaro to Modelica mapping can be found in [4].

## 3.1. Augmented Modelica Model

The reliability Figaro specific information in the Modelica model is expressed by inheriting from two abstract classes:

```
model Figaro_Object
  parameter String fullClassName;
```

```
  parameter String codeInstanceFigaro;
end Figaro_Object;


connector Figaro_Object_connector
  parameter String fullClassName;
  parameter String codeInstanceFigaro;
end Figaro_Object_connector;
```

It is necessary to have two classes with the same contents because in Modelica, a connector can only inherit from a connector.

The string parameters of these abstract classes will contain reliability Figaro-specific information used only to generate the reliability Figaro model. This String fields are used to specify the types of the Modelica components in the Figaro database, and any instantiation parameters. Through the use of inheritance the default values for these parameters can be specified once for a set of components.

## 3.2. Figaro Model Generation

To generate the reliability Figaro model from the augmented Modelica model, the abstract syntax tree of the Modelica model is traversed to extract the relevant data. This is done via the following call in the OpenModelica scripting API :

```
function exportToFigaro
  input TypeName path "the class to be exported";
  input String database "the Figaro database";
  input String mode "the type of file to be generated";
  input String options "options for the generation of FTA";
  input String processor "the Figaro processor";
  output Boolean success;
end exportToFigaro;
```

The argument path is the name of the class to be exported to Figaro. The argument database is the name and location of an XML file that specifies the database (that can be made of one or two files) that the Figaro processor will use. Argument mode defines the type of result to be generated by the export, it can either be a *figaro0* file or a *fault-tree*. The argument options is the name of an XML file that specifies the options for fault tree generation. This argument is only relevant if mode is *fault-tree*. Finally, the processor argument specifies the location of the Figaro compiler.

When exportToFigaro is done, it returns a Boolean value that indicates whether the call was successful or not.

The transformation is done in two steps, first by generating pairs of class names with their corresponding Figaro types and then by generating triplets consisting of class names, Figaro types and Figaro instantiation code for the classes. This transformation is split in two steps for efficiency purposes. Once all the triplets are retrieved, they are used to generate the Figaro model.

## 3.3. Processing of Results

Once the reliability Figaro model (i.e. list of objects) is generated, the script calls the Figaro compiler, passing it as arguments the generated model, the database and configuration information.

The Figaro compiler will store the log of the compilation file in an XML file. This file is then parsed to retrieve any error messages.

# 4. Graphical Support

The OMEdit editor in OpenModelica has been extended with graphical support for the Figaro export extension. A tab has been added for configuring the type of result generated, the location of the Figaro executable and the databases used by the Figaro processor (Figure 1).

The processor is shipped with the OpenModelica installation in the current version, but a different processor path can be set for testing for example. The processor treats the file exported from the Modelica model and generates a fault tree.

The library used is the library that describes the safety components that the model maps to in Figaro. It can be a general library such library or a domain specific library with extended rules for special component types. Linking to this kind of library reduces the amount of reliability information that needs to be specified in the Modelica model.

The options for the tree generation will define the node that will serve as a root to the tree, the type of generation algorithm to be used and other parameters.

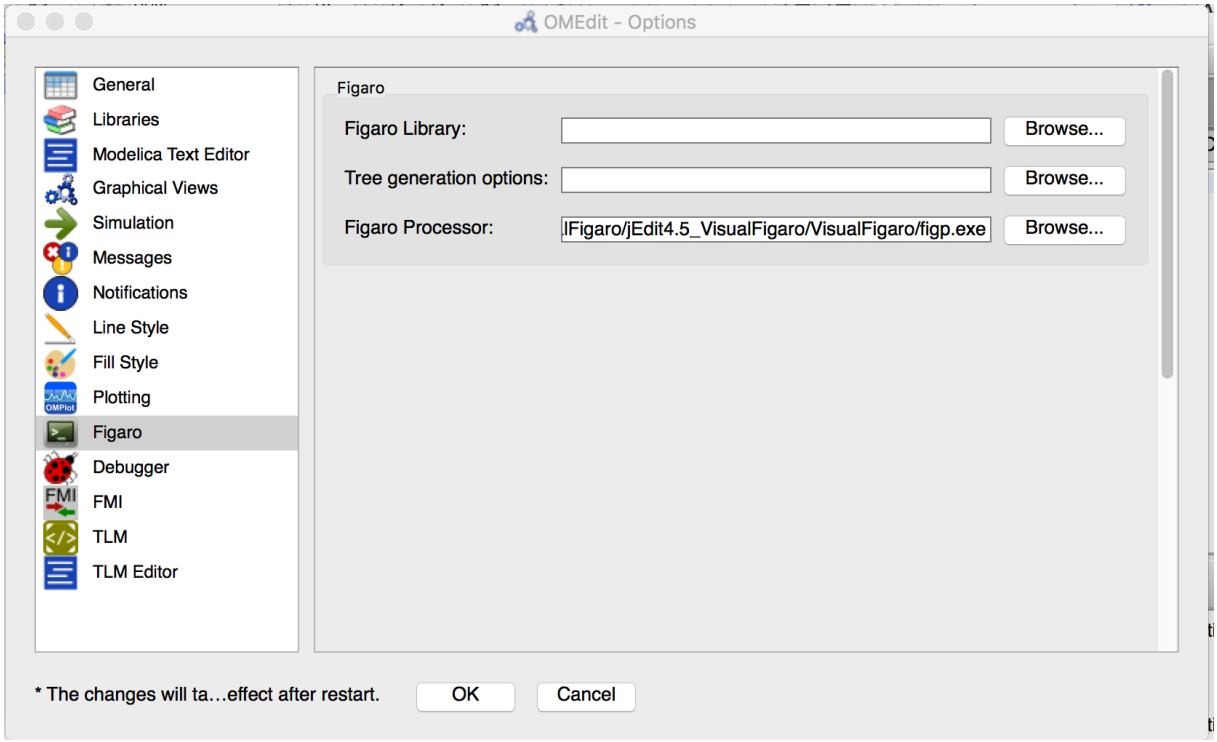


Figure 1: Configuration of FigaroExport parameters

Once the Figaro executable is configured, a reliability Figaro model can be generated by selecting the Modelica model and choosing "Export Figaro" in the right-click menu (Figure 2).

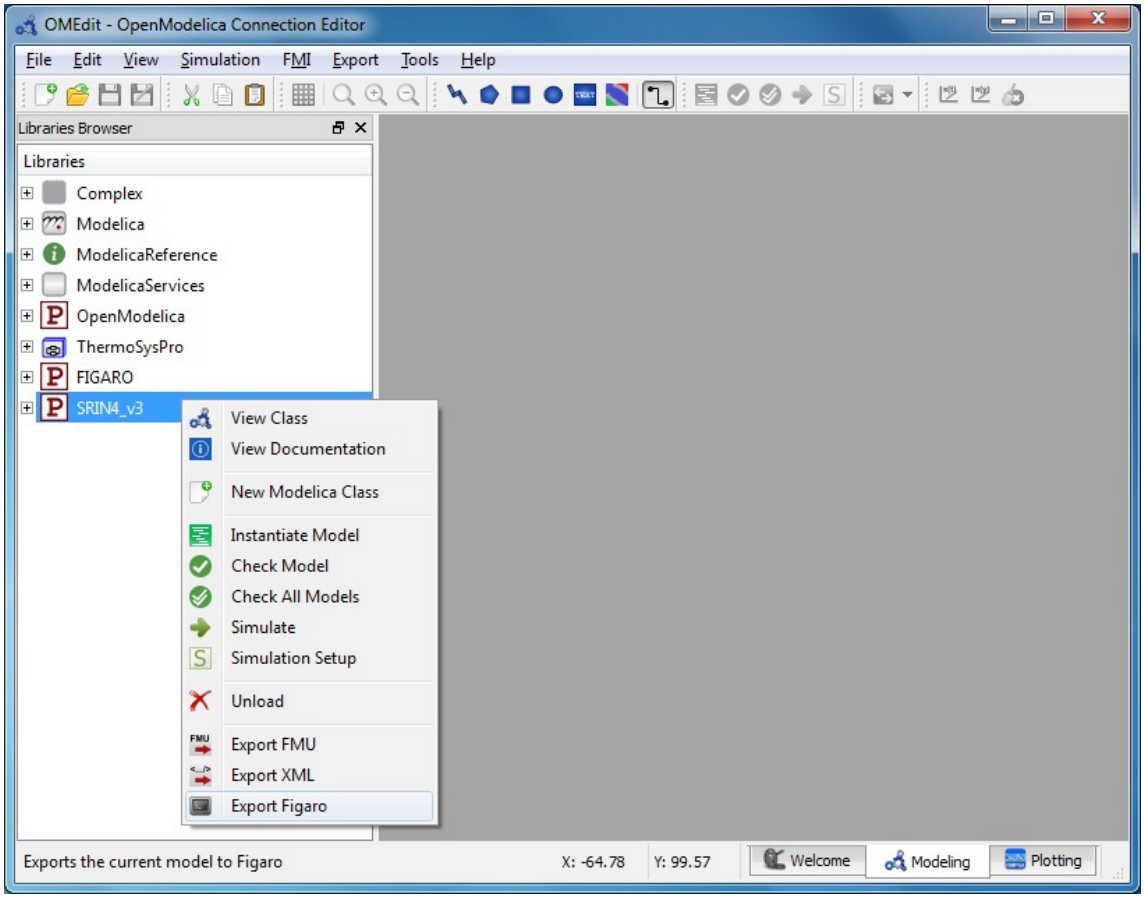


Figure 2: Right click and select Export Figaro to export the Figaro model from a given Modelica model

Possible errors are parsed from the results file and displayed in the console window.

Export can be chosen in Figaro0 or FaultTree formats described in the previous section (Figure 3).

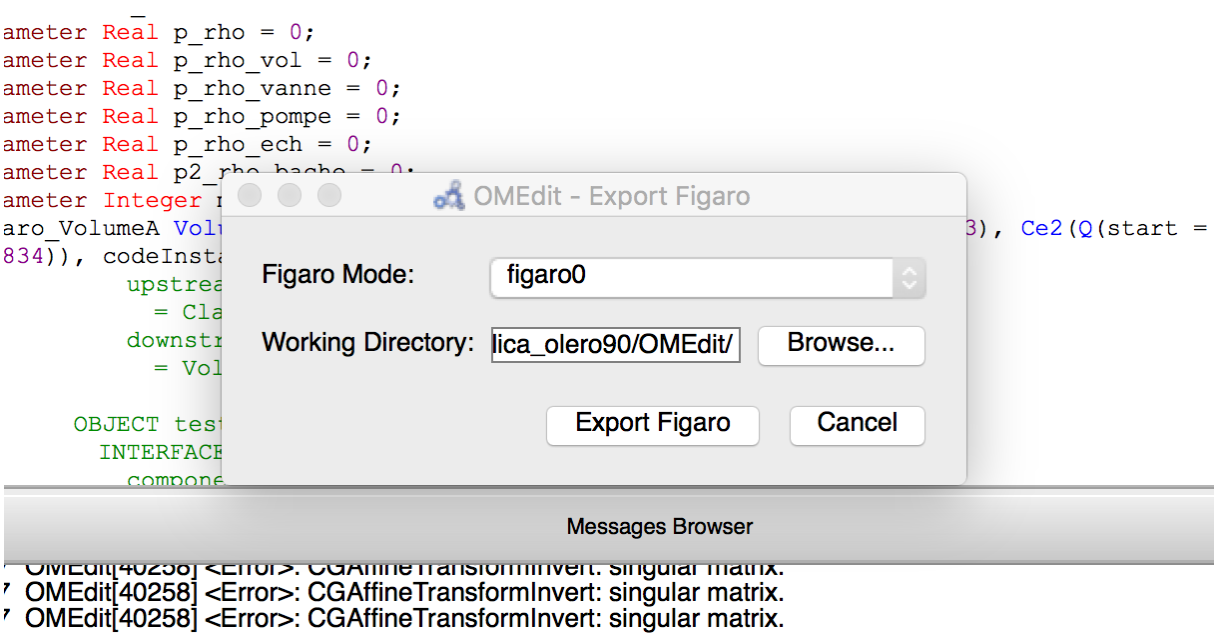


Figure 3: Select the type of file that will be generated and the directory it will be saved to

# 5. Case Study

The SRI case study described in detail in WP 2.1 has been used to test the Figaro extension. This is a cooling system case study developed by EDF (Figure 4).

First the SRI model was enriched with the Figaro information, by extending the SRI building blocks with FIGARO classes:

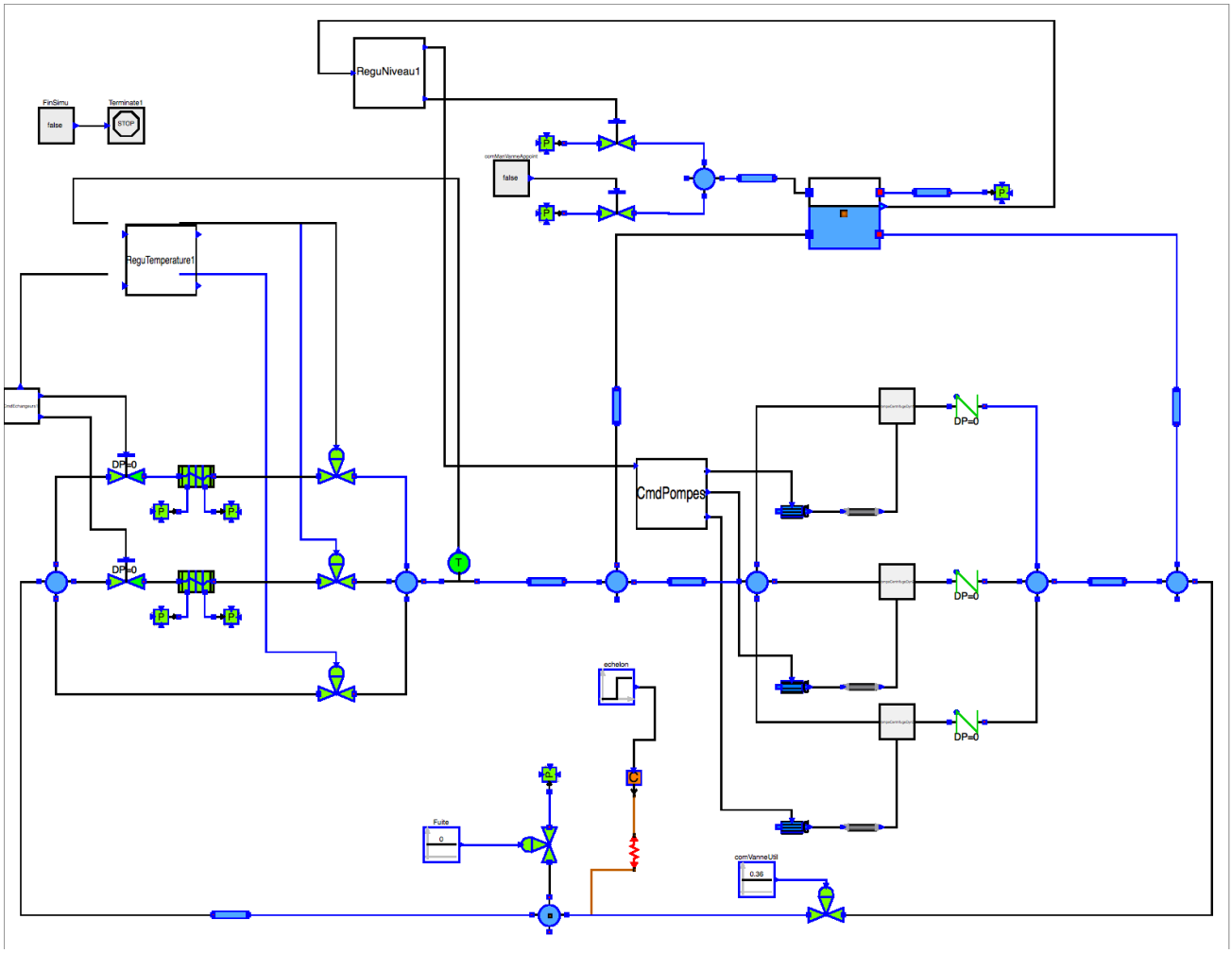


Figure 4: Modelica model of the SRI system

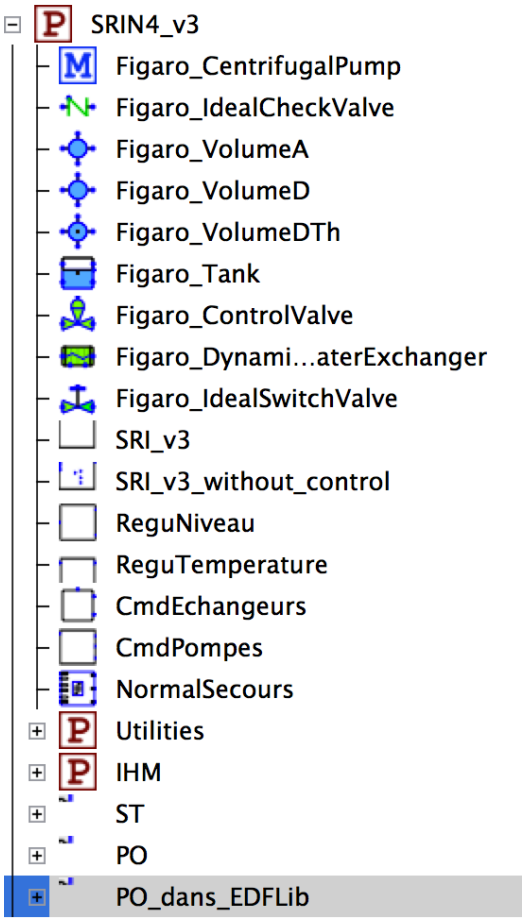


Figure 5: SRI component library

For example:

```
model Figaro_Tank
  extends FIGARO.Figaro_Object(fullClassName = "tank");
  extends ThermoSysPro.WaterSteam.Volumes.Tank;
end Figaro_Tank;
```

This enriched model can then be used to generate the reliability Figaro model.

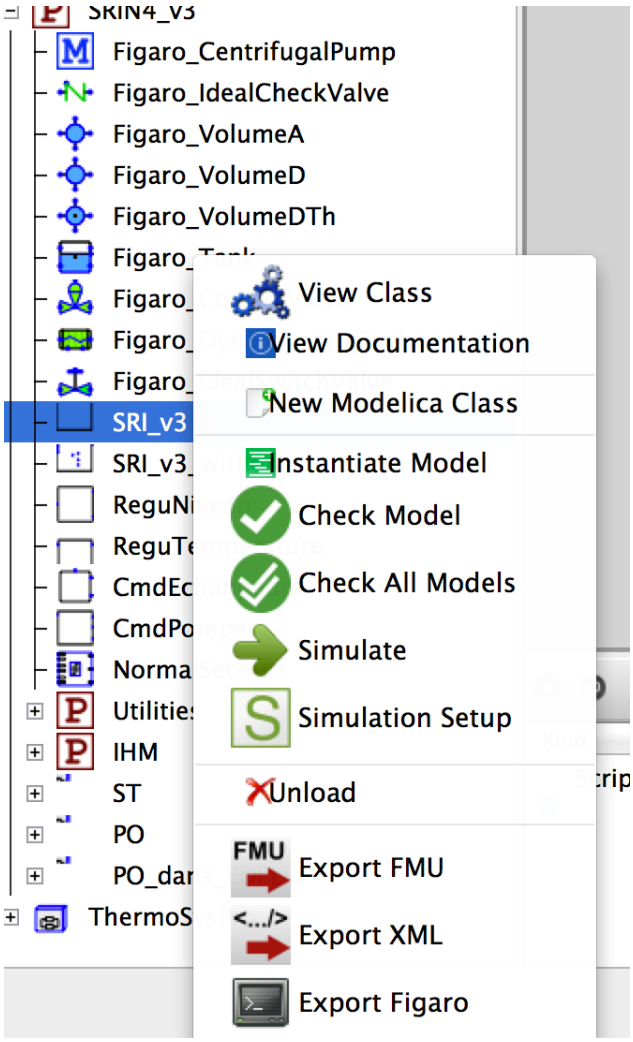


Figure 6: Export of Figaro information for the SRI_v3 model

Once the Figaro export is called, the following Figaro model is generated:

| Kind | Time | Resource | Location | Message |
|---|---|---|---|---|
| Scriptin | 00:13:06 | | 0:0-0:0 | The FIGARO is generated. |

Figure 7: OMEditor message after the operation is succesfull

All the files of this use case are available on the SVN of the MODRIO project.

# 6. References

[1] Peter Fritzson. Principles of Object Oriented Modeling and Simulation with Modelica 3.3: A Cyber-Physical Approach. 1250 pages. ISBN 9781-118-859124, Wiley IEEE Press, 2014.

[2] M. Bouissou et al., "Knowledge modelling and reliability processing: presentation of the FIGARO

language and associated tools," in Safecomp, Trondheim, Norway, Nov. 1991.

[3] Alexander Carlqvist, "OpenModelica Support for Figaro Extensions Regarding Fault Analysis," Bachelor thesis. LIU-IDA/LITH-EX-G--14/042—SE. June 2014.

[4] M, Bouissou et al, MODRIO Deliverable D2.2.1 – Specification of Modelica extensions and interfaces for Bayesian networks, Fault trees and hybrid stochastic models, April 2016.