MODPROD 2016

Dependability studies, focus on fault trees and Figaro language

Marc Bouissou

EDF R&D, dépt Management des Risques Industriels

Lena Buffoni

PELAB, Linköping University

LINKÖPINGS UNIVERSITET

MODRIO

OpenModelica

EDF

Installation for the tutorial

- For Windows only
- USB keys with OpenModelica and dependability analysis tools
- IMPORTANT: copy the whole contents of the USB key before installing

OpenModelica

EDF Figaro tools

2

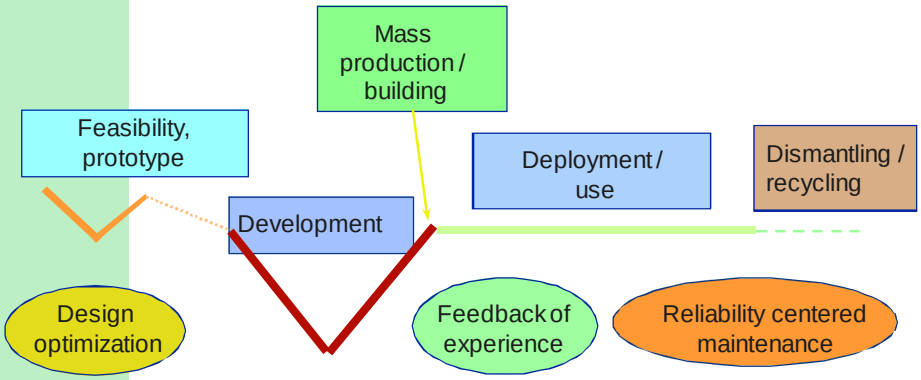
Outline

- Dependability studies and lifecycle
- General approach of a dependability study
- Static and dynamic systems & models
- Introduction to fault trees
- Principles to build fault trees
- Processing methods for fault trees
  - Minimum cut sets generation
  - Probability calculation
- Automatic generation of fault trees
  - The Figaro modeling language
  - Demonstration of the KB3 tool
- Figaro and OpenModelica

3

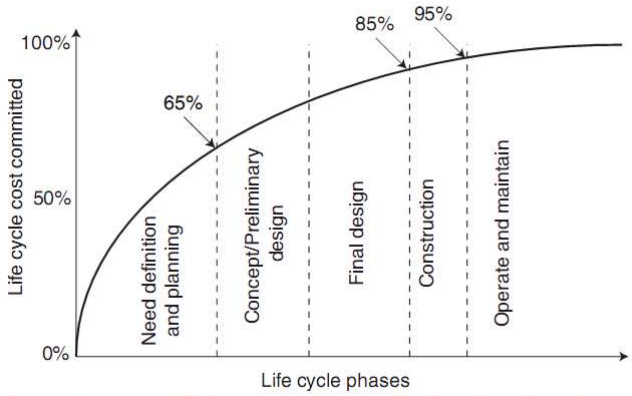
Dependability (RAMS) studies & lifecycle

Life cycle of a product or installation & related dependability activities



5

The importance of early phases

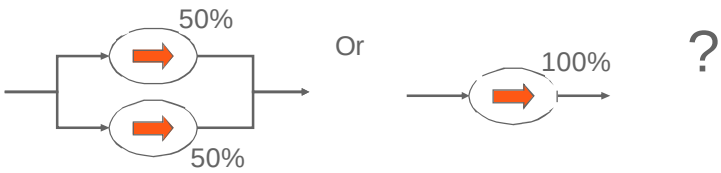


Source: Blanchard, B.S., Design and Manage to Life Cycle Cost, Forest Grove, OR, MA Press, 1978

6

Design optimization

- Find the best trade-off: costs / dependability
- Application of the ALARP principle
- Example:



The choice is not so simple: 2 pumps require check-valves in addition to avoid inverse flows => additional failures are possible
Pumps with the same reliability & maintainability, or not ?
The first solution is more "survivable", the second one requires less maintenance (but forbids online preventive maintenance)

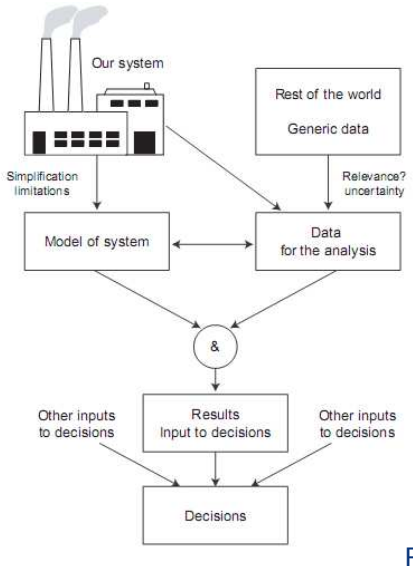
7

Design optimization

- Choices to be made:
  - architecture (redundancies)
  - spare stocks
  - technologies (e.g. hydraulic or electric or pneumatic motors for valves ?)
  - materials (resistance to stress, corrosion, radioactivity…)
- Architecture alternatives are not easy to compare automatically
- Usually the architecture is decided by the designers according to functional considerations rather than dependability considerations
- However, they may wish to compare a few possibilities => **tools are required for quick evaluations**

8

General approach for a *system* dependability study



From the book by Rausand & Hoyland

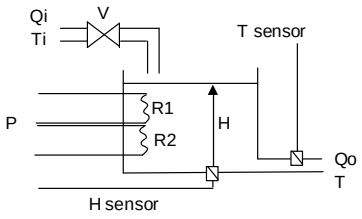
9

First modeling of the system

Steps

- Determine the system bounds with regard to other systems, in particular to fix the support systems
- Acquire a correct knowledge of the functioning of the system
  - Functional analysis
  - Preliminary risk analysis, HAZOP, FMECA
- Choose the level of detail

=> first modeling of the system, for example in the form of a diagram, accompanied with the main functional hypotheses



Mission: heat the water at temperature T=65°C, and maintain level H between two bounds…

10

Choice of the detail level of the model

A crucial issue

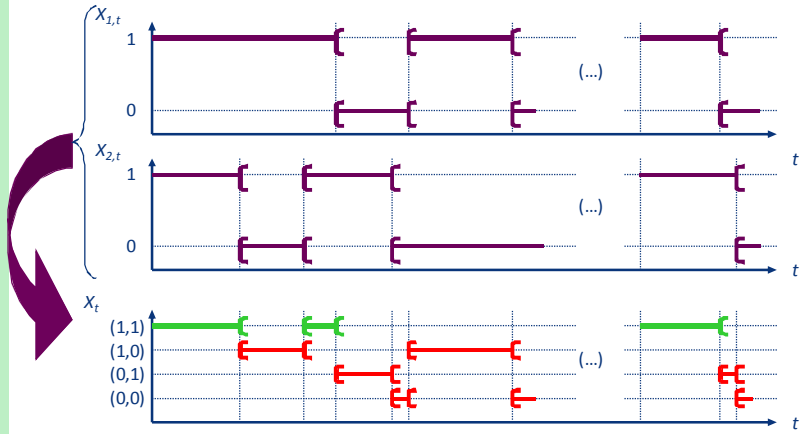
- too coarse => we cannot answer all the questions
- too detailed =>
  - Model is intractable
  - Reliability data may not be available

11

Static and dynamic systems & models

(for dependability analysis)

Combining two components: series system



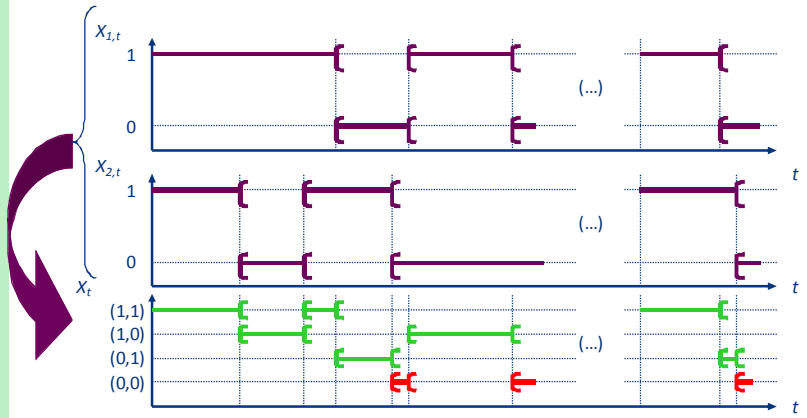
Example of history of the studied system

$$A(t) = A_1(t)A_2(t)$$

$$R(t) = R_1(t)R_2(t)$$

13

Combining two components: parallel system (active redundancy)



Example of history of the studied system

$$A(t) = 1 - (1 - A_1(t))(1 - A_2(t))$$

!

The reliability cannot be computed as a function of reliabilities or availabilities

14

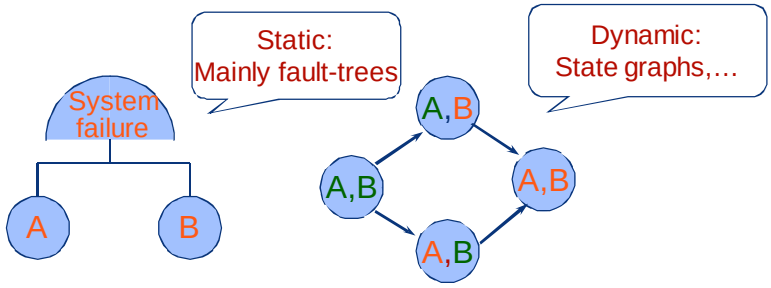
Combining two components: parallel system (standby redundancy)

- The history previously given as an example is no longer possible
  - When Cpt 1 works, Cpt 2 is in standby and cannot fail
  - Example of dependency between the components
- A global approach of the system is needed
  - Impossible to reason component by component
  - Calculations are more complex…
- Other examples of dependencies
  - Reconfigurations
  - Shared maintenance resources
  - Common cause failures
  - …

15

Characteristics of reliability models

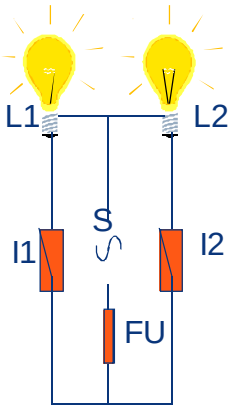
- Combinatorial explosion of the possible failure combinations
- Discrete state models – often Boolean
- Two main categories of models

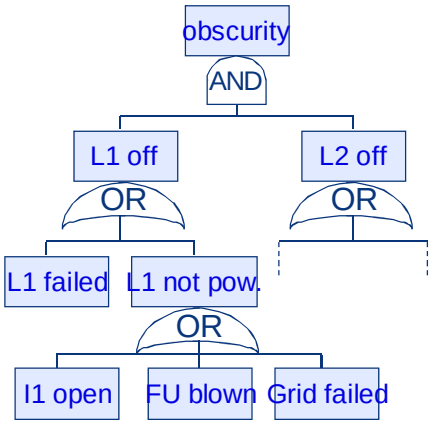


16

Fault-tree

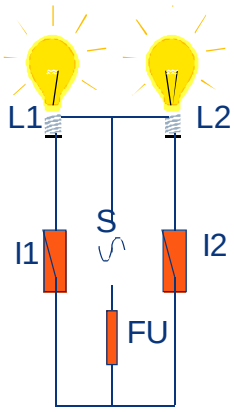
Explains an undesirable event as a function of failures of components in the form of a Boolean function

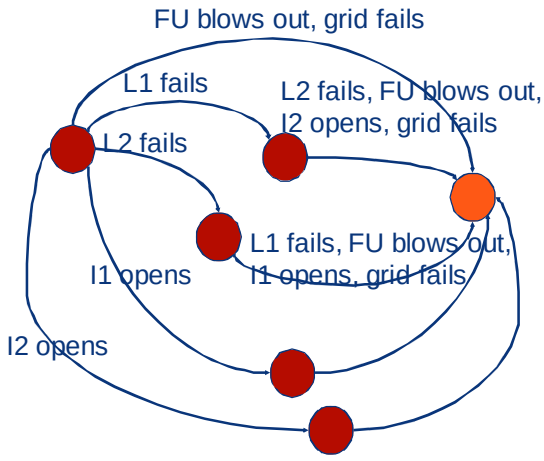




17

State graph





Simple because no repair

18

The limits

- Static models assume the independence of components -> (more and more) precise calculations on approximate models

- Dynamic models: more accurate, but combinatorial explosion -> imprecise (or incomplete) calculations

19

Fault trees

A powerful method,

facilitating the **construction** of the model and its **processing**

Scope of fault trees

- One (or more) Boolean undesirable event(s)
- Two physical (or intrinsic) states for components: working or failed
- Two states for the functions fulfilled by the components
- Independence of **components** (w.r.t. to their failure and repair processes)
- Deterministic interactions between **functions** (e.g. loss of power supply => loss of the function of all components using this power supply)

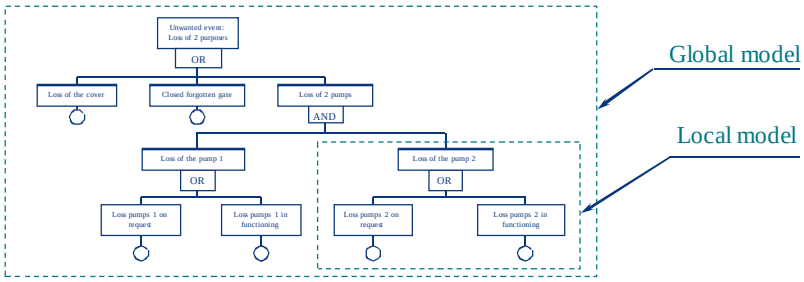
21

Local model and global model

- Repetitive process of determination of the causes:

  The fault tree is formed by successive levels such that every event is generated by other events, through logical operators

- The complete tree represents the global model of the system (for a particular undesirable event)
- A part of the tree represents a local model



22

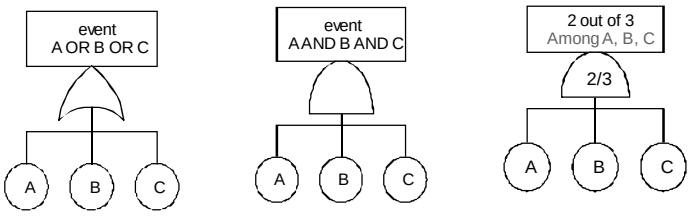
Definition of a fault tree

- Mathematical model connecting the occurrence of a **top event** to those of a set of **basic events**
- The events are associated to Boolean indicator variables (X=1 means (by abuse of notation): the event X is realized); we identify the union, the intersection of the events with the Boolean operators OR, AND
- A fault tree is thus a Boolean function
- Normalized graphic representation
- The basic events must be independent, if we want to use the fault tree for probability calculations

23

Graphic representation

Most common gates



OR gate

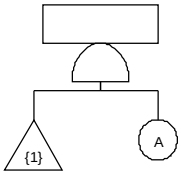
AND gate

K/N gate
(equivalent to one OR on a set of AND)

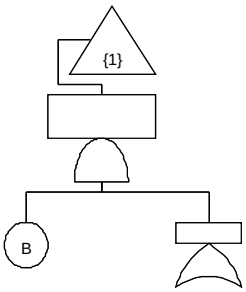
Less often used gates: NOT, NOR, NAND...
they create "non coherent" trees, more difficult to process

24

Use of the transfer gates



the sub-tree { 1 } is defined once and can be referenced in multiple places in the fault tree



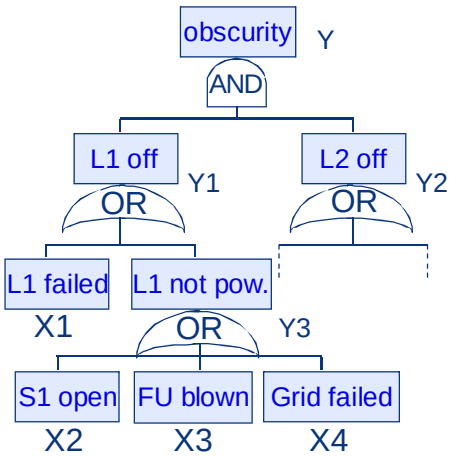
The possibility of having repeated events or sub-trees makes the fault "tree" in fact a **graph** without circuit

25

Textual representation (example)

Y = Y1 & Y2;
Y1 = X1 ¦ Y3;
Y3 = X2 ¦ X3 ¦ X4;
Y2 = X5 ¦ Y4;
Y4 = X6 ¦ X3 ¦ X4;

Y = F (X1, X2... Xn)



26

Method of construction

- Start from the top event (usually a loss of function of the system)
- Look for the immediate, necessary and sufficient causes of the under development event; when this inventory is ended,
- Choose a new event to develop, and repeat the previous point

Advice

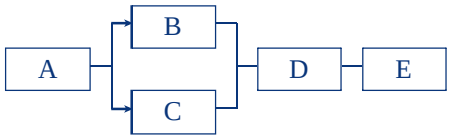
- Continue the decomposition upon reaching independent basic events of known probability
- Develop in width (level by level) rather that in depth first
- Give labels to each gate; this is required for the readability of the tree

27

Examples

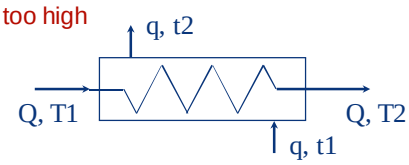
Example 1

U.E. = no output signal for E



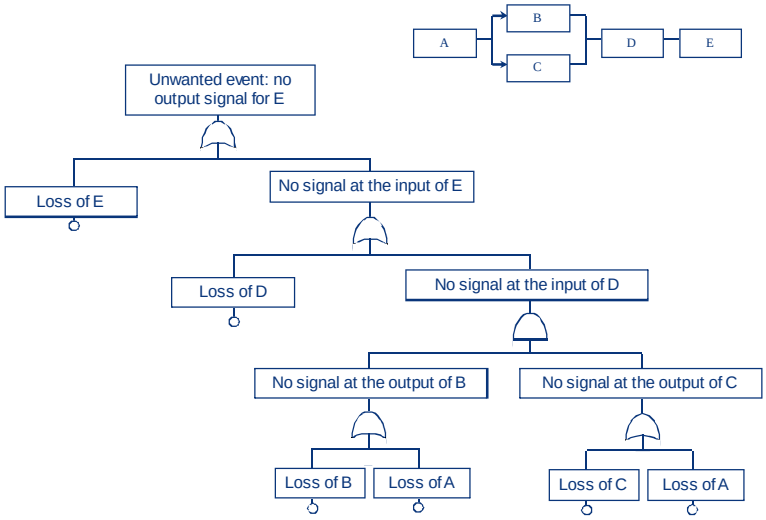
Example 2

U.E. = T2 too high



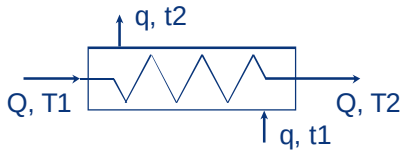
28

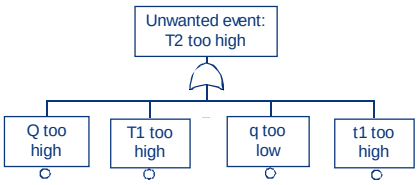
Example 1



29

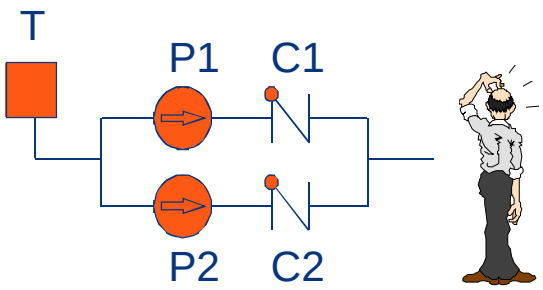
Example 2





30

Exercise



Unwanted event: no water at the output of the system

31

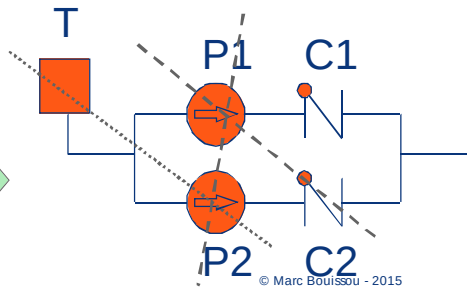
Qualitative processing of (coherent) fault trees

search for minimal cut sets

Definition of the minimal cut sets

- CUT SET: set of basic events whose (simultaneous) occurrence ensures the occurrence of the top event
- MINIMAL CUT SET : cut set containing no other cut set
- ORDER of a cut set: number of events that it contains
- The set of all the minimal cut sets constitutes a canonical representation of the fault tree

Example: two minimal and one non minimal cut sets



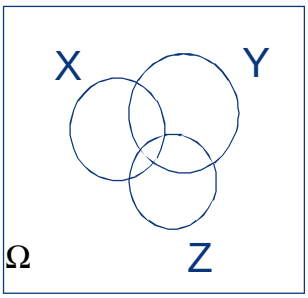
33

Calculation of the minimal cut sets

Consists in rewriting the Boolean function, using Boole's algebra laws

X+X = X.X = X

- X+1 = 1, X+0 = X
- X.0 = 0, X.1 = X
- X.(Y+Z) = X.Y + X.Z
- X+Y.Z = (X+Y) . (X+Z)

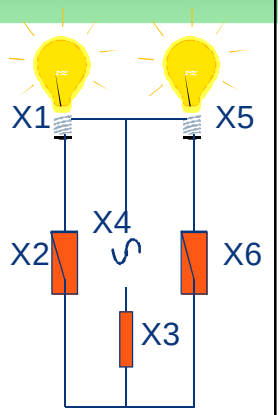


Boolean product ←→ intersection of events
Boolean sum←→ union of events

34

Example

Y = Y1.Y2
Y1 = X1 + Y3
Y3 = X2 + X3 + X4
Y2 = X5 + Y4
Y4 = X6 + X3 + X4



Y = (X1 + X2 + X3 + X4). (X5 + X6 + X3 + X4)
Y = [(X1 + X2) + (X3 + X4)]. [(X5 + X6) + (X3 + X4)]
Y = (X1 + X2). (X5 + X6) + X3 + X4
Y = X1. X5 + X1. X6 + X2. X5 + X2. X6 + X3 + X4

35

Use of the minimal cut sets

- Validation of the model
- They allow to detect the weak points of the system (but be careful: according to probabilities, the cut sets of smaller order are not necessarily the most likely)
- They allow the identification of false redundancies and potential common cause failures (this requires that we can associate characteristics to the basic events, such as: place, conditions of environment, manufacturer…)
- Support of the probability calculation (see next chapter)

36

Beware of combinatorial explosion!

- Typically: millions of cut sets for a highly redundant medium sized system (<300 components)
- Truncation criteria: order, probability

37

Quantitative processing of fault trees (coherent or not)

Calculations of probability, importance factors

The Poincaré formula

Also called the inclusion-exclusion principle

$$\Pr(UE) = \Pr(C_1 \cup C_2 \cup \ldots \cup C_i \cup C_n)$$

$$\boxed{\begin{aligned}\Pr(UE) = \sum_{i=1}^{n} \Pr(C_i) - \sum_{i<j} \Pr(C_i \cap C_j) + \sum_{i<j<k} \Pr(C_i \cap C_j \cap C_k) \\ - \ldots + (-1)^n \Pr(C_1 \cap C_2 \cap \ldots C_n)\end{aligned}}$$

Intractable for large numbers of minimal cut sets
In practice, the approximation given on next slide is used

39

Calculation from the minimal cut sets (coherent trees)

Theorem of Poincaré:

$$\sum \Pr(C_i) - \sum_{i \neq j} \Pr(C_i \cap C_j) \leq \boxed{\Pr(\bigcup C_i) \leq \sum \Pr(C_i)}$$

The probability of every cut set is given by the product of probabilities of the basic events which compose it (because they are independent). And for an intersection of cut sets?

This approximation is good only if **all basic events are of low probability**. The fact that minimal cut sets have a low probability is not sufficient!

40

Summary...

- fault trees are a very practical type of models by their readability, their capacity to model large systems, with complex structures
- there are numerous software* to process them; the most recent are based on BDD techniques, that allow exact calculations (thus **valid whatever the probabilities, and for non coherent as well as coherent fault-trees**)
- their essential limitation is their **incapability to take into account dependences between components** (e.g: functioning in normal/standby, limitation of the number of repairmen…)

* But very few of them are free:

- XFTA http://www.lix.polytechnique.fr/~rauzy/xfta/xfta.htm
- OpenFTA http://www.openfta.com/ (poor GUI)
- GRIF workshop http://grif-workshop.com/ (limited in size)

41

Automatic construction of fault trees

Principles of the KB3 tool developed by EDF

Demonstration

Why automation tools?

- Needs for the PSA of nuclear power plants
  - Consistency
  - Traceability
- Needs for the studies to be performed during the design of new systems
  - Speed
  - Accessibility to non specialists

43

Principles of the KB3 workbench

Knowledge Base
(Figaro language)

Description of components

Figaro 0
(textual model)

If static Model

Fault tree generator

Towards commercial tools (Aralia, Risk-Spectrum)

- ✓ **Minimal cut sets**
- ✓ **Reliability**
- ✓ **Availability**

Automatic generator of Sequences: FIGSEQ

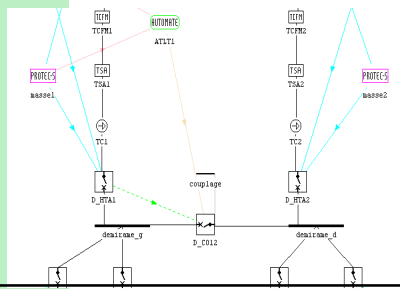
- ✓ **The most likely sequences**
- ✓ **Reliability, MTTF**
- ✓ **Availability**

Monte Carlo simulation software : YAMS

- ✓ **The most likely sequences**
- ✓ **Reliability, Availability**
- ✓ **Mean values of random variables**

If dynamic Model

Graphic input of system models



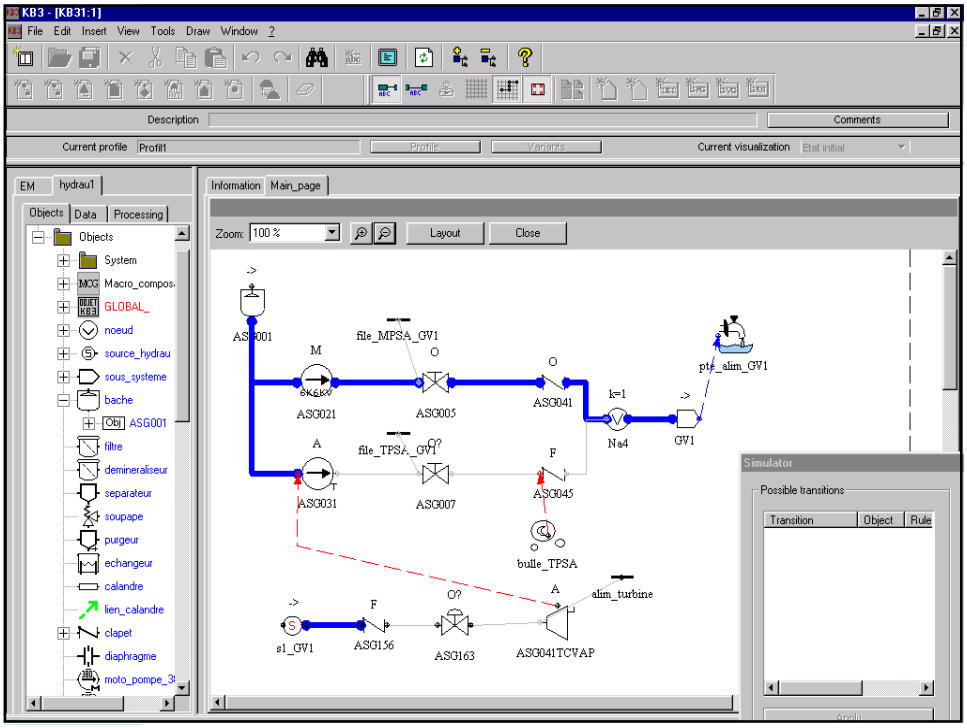
44

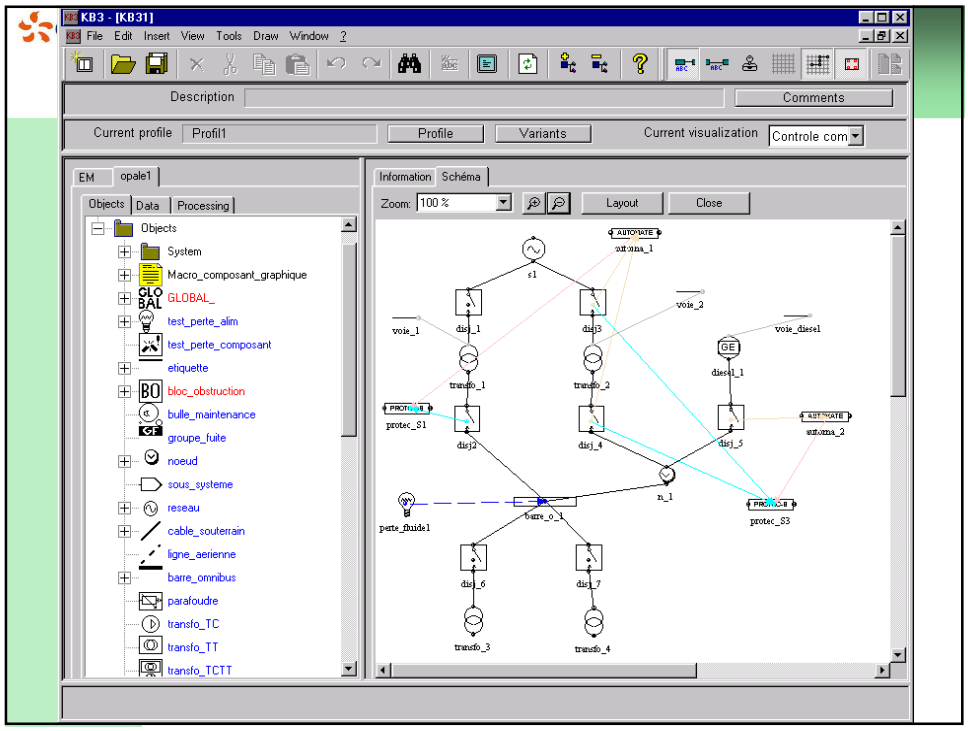
A knowledge base = a graphic language to build models

A KB is defined by two main files (+ icons)

- Generic behavior models of the components of the KB in FIGARO language : defines the semantics of the graphic language
- Parameter settings of the graphic interface
  - Define the "grammar": what is allowed / forbidden in terms of connections
  - Improve the user-friendliness by the definition of changes of colors, texts and icons in interactive simulation

45





Decomposition of a NPP

NPP → Function : produce electricity

System → Functions of the system
Needs several components
- ex: cool the condenser

Important component or set of components
- ex : pump, regulation valve, heat exchanger

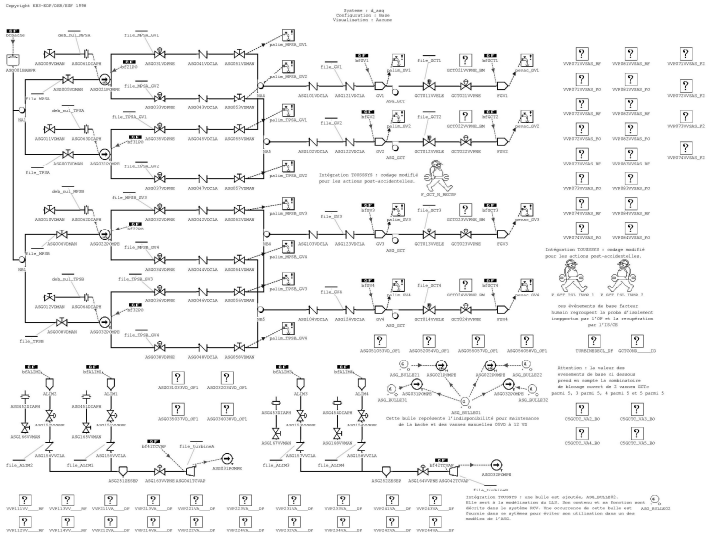
Sub-component
- ex : pump, motor,…

Subset
- ex : contactor

Item
- ex : bearing

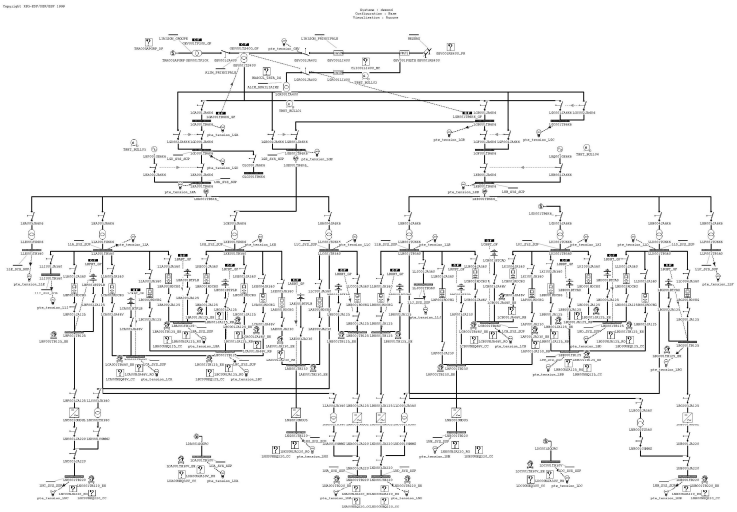
48

KB3 model of Emergency Feedwater System



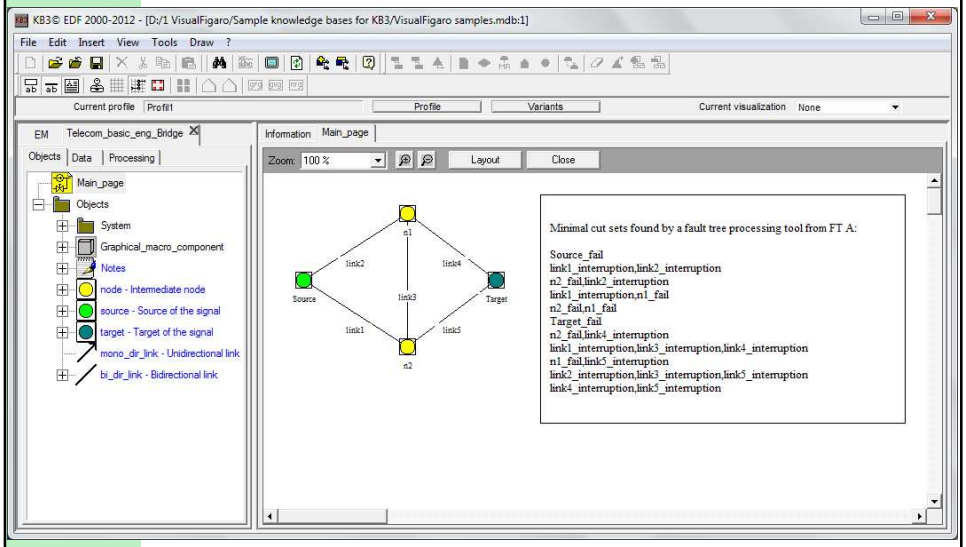
49

KB3 model of electrical supplies



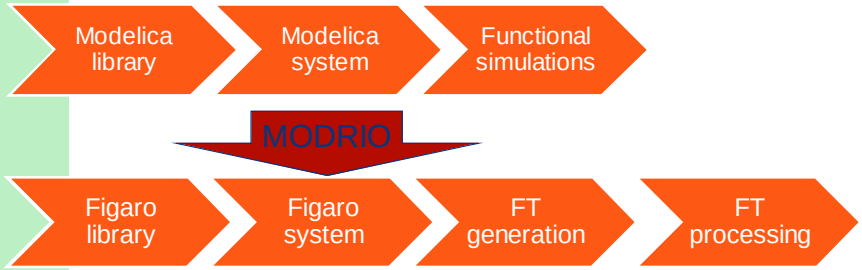
50

A simple telecom network



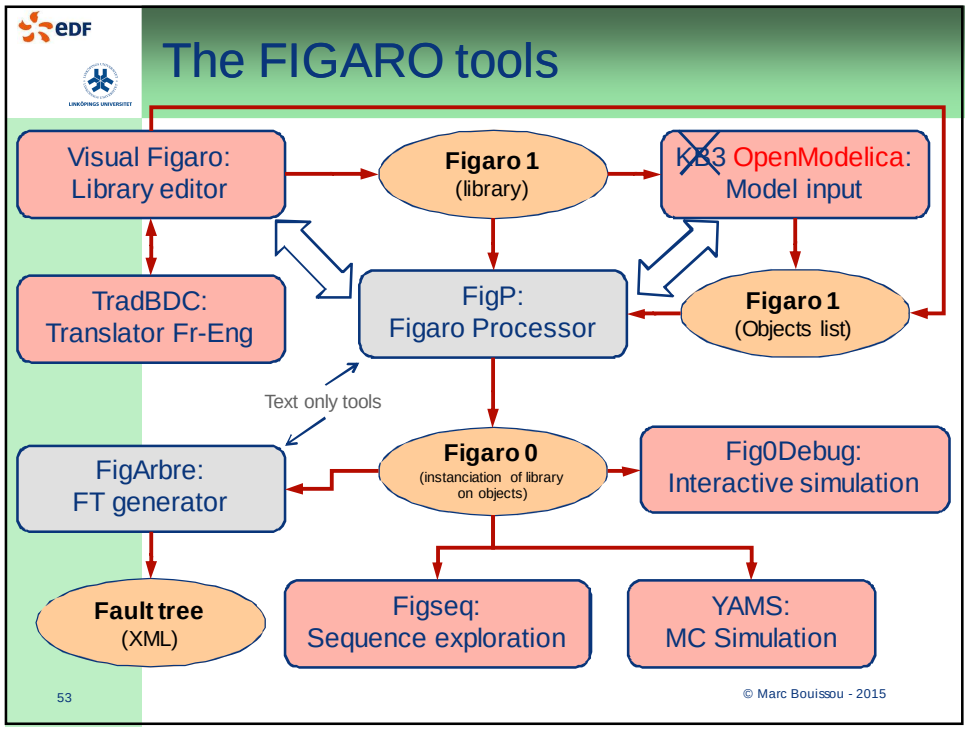
51

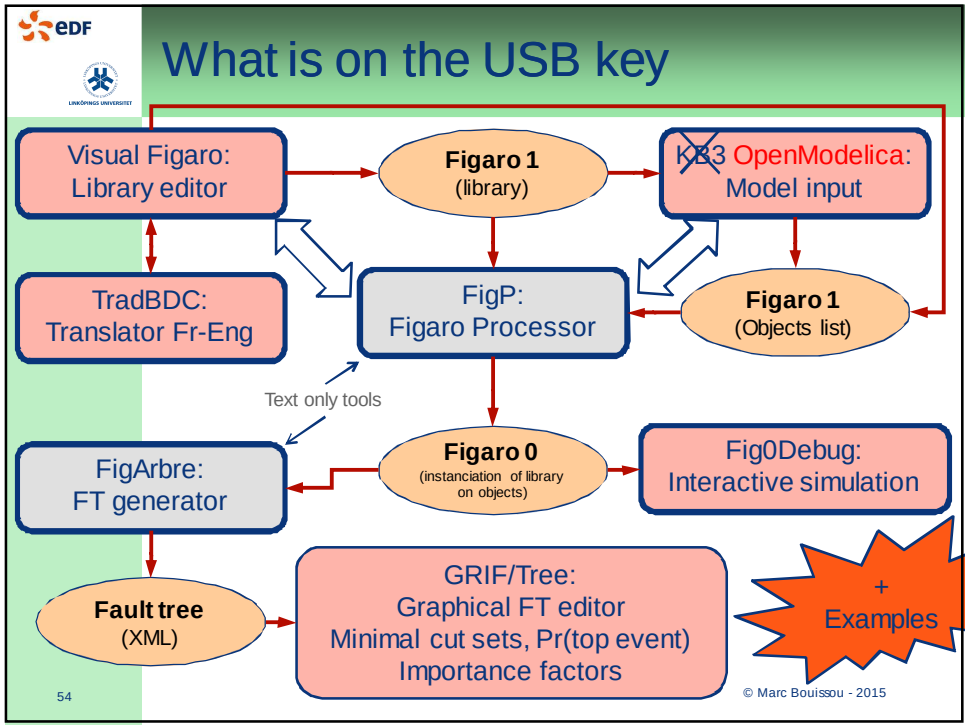
Engineering workflows



- Visual Figaro
- Modelica tool
- Figaro processor
- GRIF/Tree, Risk Spectrum…

52


The FIGARO tools
Visual Figaro: Library editor
Figaro 1 (library)
KB3 OpenModelica: Model input
TradBDC: Translator Fr-Eng
FigP: Figaro Processor
Figaro 1 (Objects list)
Text only tools
FigArbre: FT generator
Figaro 0 (instanciation of library on objects)
Fig0Debug: Interactive simulation
Fault tree (XML)
Figseq: Sequence exploration
YAMS: MC Simulation
53
© Marc Bouissou - 2015


What is on the USB key
Visual Figaro: Library editor
Figaro 1 (library)
KB3 OpenModelica: Model input
TradBDC: Translator Fr-Eng
FigP: Figaro Processor
Figaro 1 (Objects list)
Text only tools
FigArbre: FT generator
Figaro 0 (instanciation of library on objects)
Fig0Debug: Interactive simulation
Fault tree (XML)
GRIF/Tree: Graphical FT editor Minimal cut sets, Pr(top event) Importance factors
+ Examples
54
© Marc Bouissou - 2015

RELIABILITY ANALYSIS WITH MODELICA AND FIGARO

55

Modelica and Figaro: two DSLs

- Modelica: system design and functional validation
  - Deterministic physical models: algebraic and differential equations (DAE)
  - Object oriented
  - Declarative and procedural parts
  - Supported by a large number of tools (open source or commercial)
- Figaro: system dependability analysis
  - Discrete stochastic models: states and stochastic transitions
  - Object oriented
  - Declarative (based on occurrence and interaction rules)
  - Supported by the KB3 platform (partly free. Should be open source soon)

56

Why interface Modelica with Figaro?

- Modelica models used in dynamic model verification of the functional behavior
- Figaro can be used for static or dynamic reliability analysis

These two ways to model

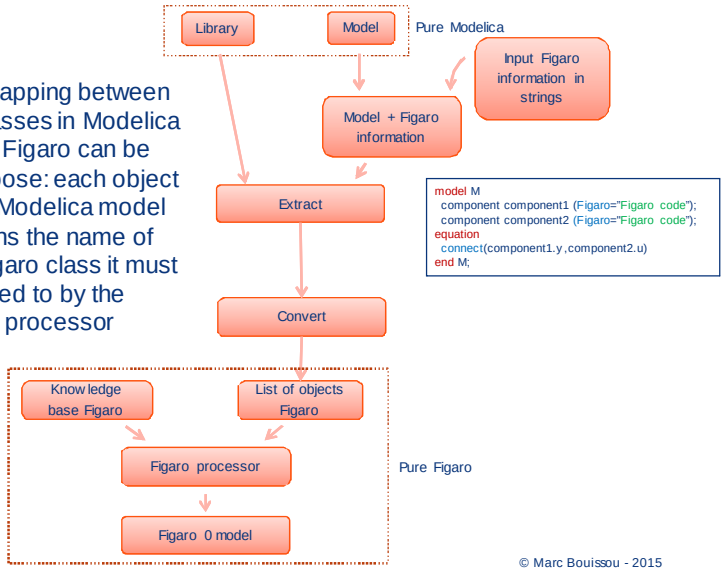
systems are complementary !

Advantages

- Modelica models allow a graphic representation of the system structure
- Automatic generation:
  - Reduces risk of introducing errors
  - Improves usability
  - Simplifies model maintenance

Using Modelica in conjunction with the Figaro tools (the binding principle)

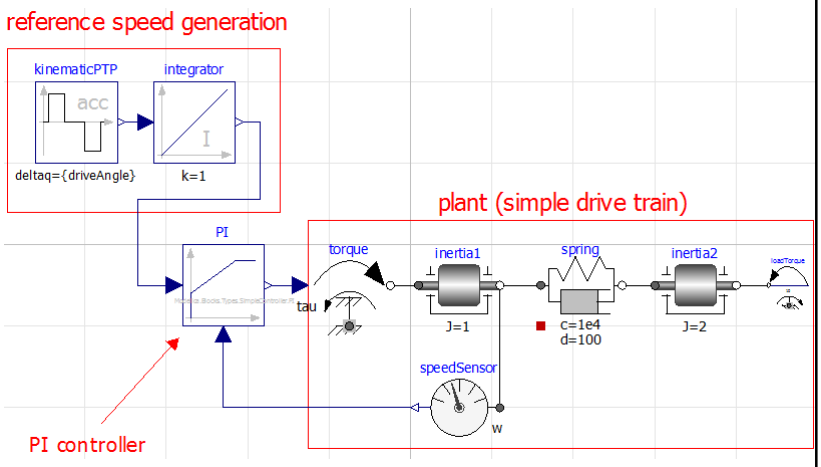
The mapping between the classes in Modelica and in Figaro can be very loose: each object in the Modelica model contains the name of the Figaro class it must be linked to by the Figaro processor



59

First example of the tutorial

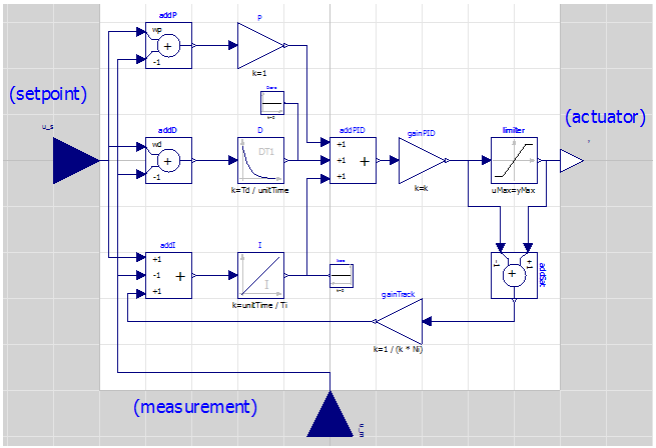
In this system taken from the SML…



What failures of the PI controller components can induce a bad control?

60

System to be studied: PID regulator from the SML

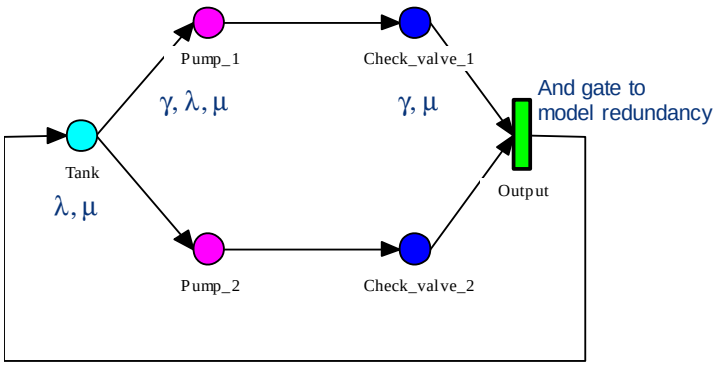


Closed loop control system

Purpose : find the failure combinations that may cause a bad output

Figaro library to be used: Digraph

- A digraph is a simple Boolean model allowing loops
- Basic elements: nodes and logical gates
- Each node can have one or two failure modes



62

Steps needed to get a Figaro model starting from a pure Modelica model

- Load the FIGARO generic classes in OMEdit (M2F_Tutorial\4_tutorial_examples\FIGARO.mo)
- Clone the PID controller example (from SML) in OMEdit
- Adapt it to add the FIGARO information
- Configure FigaroExport
- Generate the Fault Tree

63

Mapping Modelica to Figaro (1/2)

The reliability information in the Modelica model is expressed in parameters inherited from two abstract classes:

```
model Figaro_Object
  parameter String fullClassName;
  parameter String codeInstanceFigaro;
end Figaro_Object;
```

For this example, we will need only components « lm » with parameters λ, μ

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

Cf. Telecom example to see a case where connectors are needed

Mapping Figaro to Modelica (2/2)

```
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;
```

Through the use of inheritance the default values for these parameters can be specified once for a set of components.

Create new classes by multiple inheritance

on FMI Export Tools Help

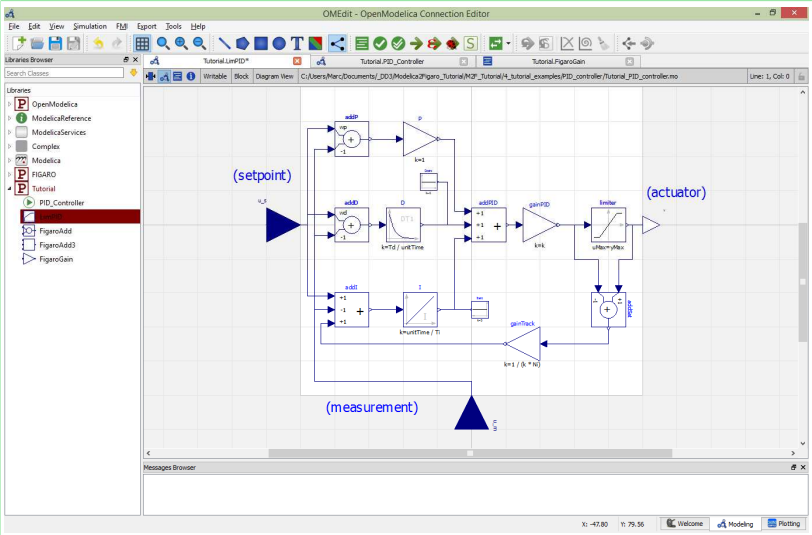
Tutorial.LimPID | Tu

Writable | Model | Text View | //psf/Home/Work/Modeli

```
model FigaroAdd3
  extends Modelica.Blocks.Math.Add3;
  extends FIGARO.Figaro_Object(fullClassName = "lm");
end FigaroAdd3;
```

The string fullClassName allows to map the classes of the original Modelica model to the Figaro classes

Use these new classes to rebuild the PID controller

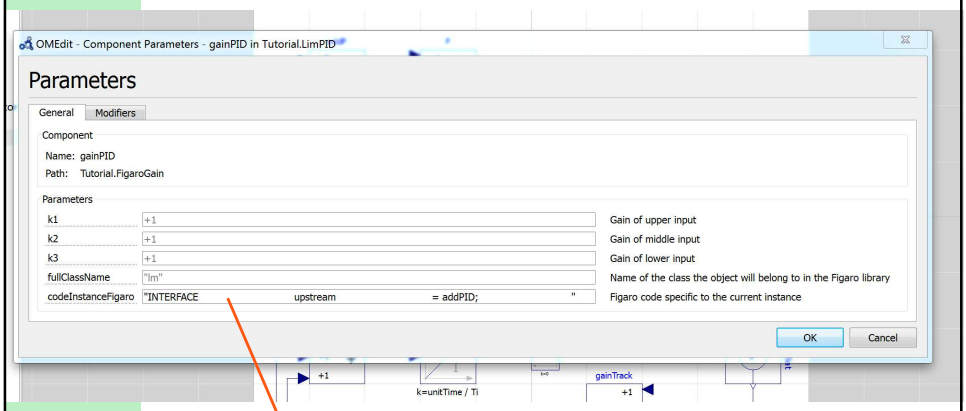


The new model looks exactly like the original one, has the same behavior in simulation…

67

Input the Figaro topology in the Modelica model

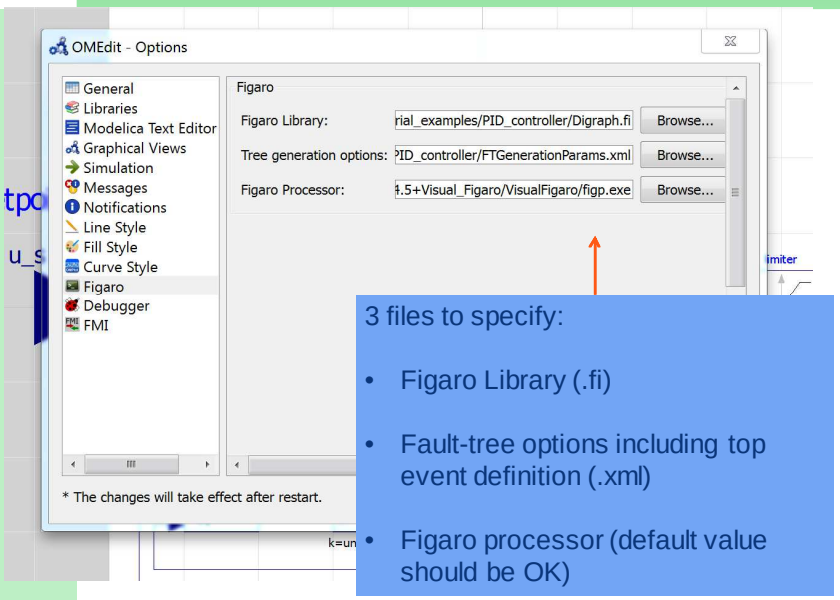
… but allows to input Figaro information



INTERFACE upstream = addPID;

68

Configuring Figaro generation in OMEdit Options



3 files to specify:

- Figaro Library (.fi)
- Fault-tree options including top event definition (.xml)
- Figaro processor (default value should be OK)

The fault tree generation options

```
<?xml version="1.0" encoding="UTF-8"?>
<GEN_TREE_OPTIONS>
  <FORMAT>KB3</FORMAT>
  <SIMPLIFICATION>TOTALE</SIMPLIFICATION>
  <COHERENCY>PARTIELLE</COHERENCY>
  <UNLOOP>VRAI</UNLOOP>
  <TREE_NAME>Test_generation</TREE_NAME>
  <PREFIX>__ARBRE__</PREFIX>
  <TEST_VARIABLE>
    <OBJECT>gainPID</OBJECT>
    <VARIABLE>loss</VARIABLE>
  </TEST_VARIABLE>
  <MAX_SONS_GATE>100</MAX_SONS_GATE>
  <LIST_GROUPS>
    <GROUP>common_group</GROUP>
    <GROUP>SANS_NOM</GROUP>
  </LIST_GROUPS>
  <COMMENT>Comment</COMMENT>
</GEN_TREE_OPTIONS>
```

Keep all the blue text without changes

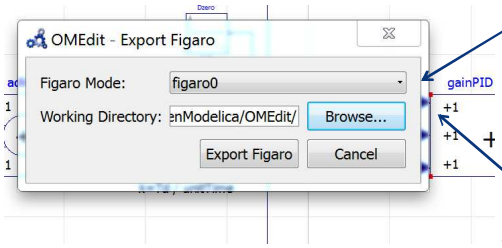
Top event definition

Names of the groups of rules needed for FT generation. Always keep “SANS_NOM”

70

Launch the Figaro export

- **Save the model**
- Right click on the system class in the library tree
- Select Export to Figaro



Select Figaro0 or Fault tree

Make sure you select the directory containing the model

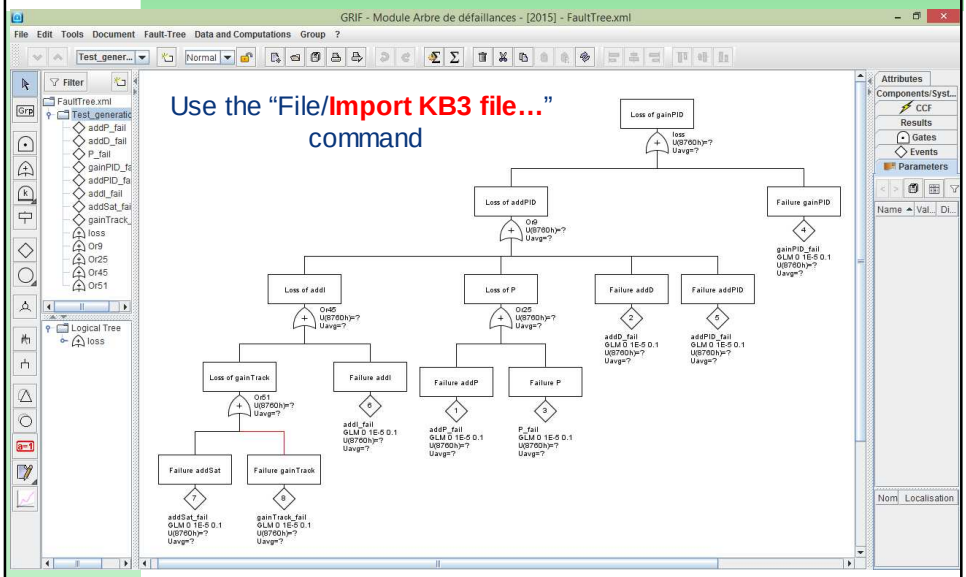
Generation of the Figaro model in OpenModelica: how it works

- To generate the Figaro model from the augmented Modelica model:
  - the abstract syntax tree representation of the model is parsed
  - the Figaro specific properties and Figaro object instances are exported and saved as a Figaro file containing objects: FigaroObjects.fi
  - The Figaro library and FigaroObjects.fi are processed jointly by the Figaro processor in order to produce a Figaro 0 file or a fault tree
  - This latter operation can be done from VisualFigaro, with a more complete GUI for choosing options

OpenModelica scripting API:

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

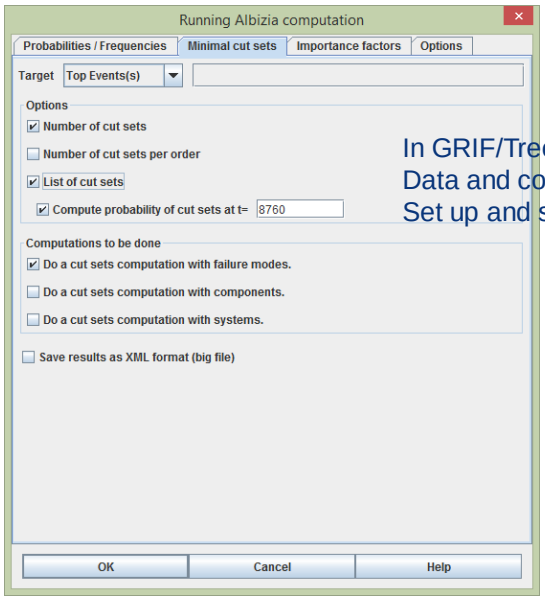
The generated fault tree in GRIF/Tree



74

© Marc Bouissou - 2015

Launch calculations on the FT



In GRIF/Tree, choose the comman
Data and computations/
Set up and start computation

75

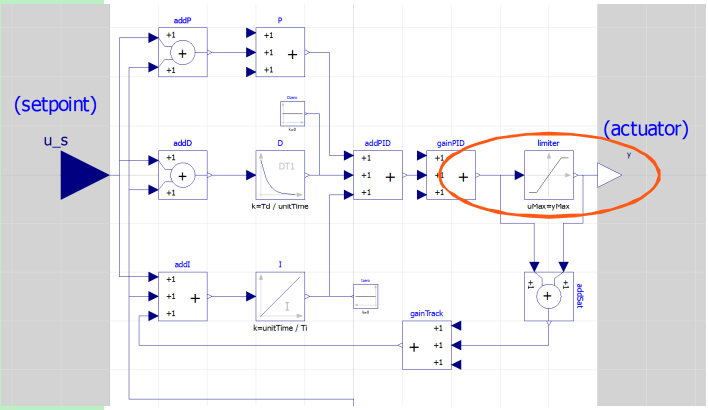
Exercise

- Open the Digraph knowledge base in VisualFigaro.
- Look at the different types of probability models you can have in the system
- Change the (Figaro) node type from lm to cs for a component in the Modelica model. Also add the information "CONSTANT mu=0;" in the string codeInstanceFigaro to specify that this component is not repairable
- Regenerate a Fault tree and see the differences in the probabilities of cut sets calculated with GRIF/Tree

76

Exercise

- Extend the reliability model to include the limiter. You will need to extend the limiter class, add the reliability information on the system and redefine the top event.



77

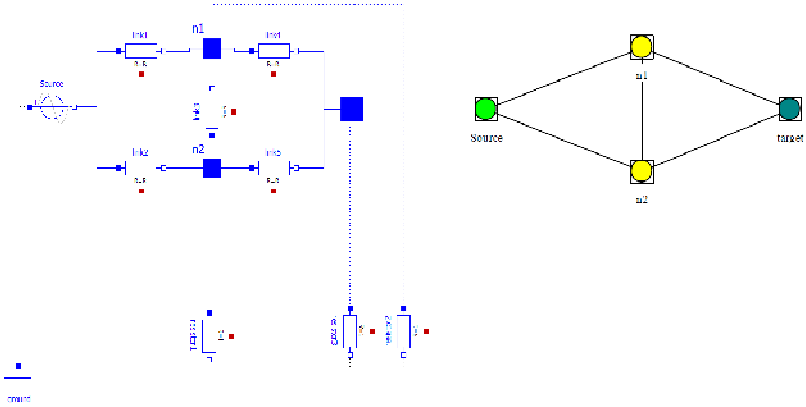
Extra Exercise : Hierarchy levels

- Look at the PID controller model, which includes the LimPID model with reliability information
- Extend the PID controller with reliability information, by mapping each component to an lm node.
- Generate the flat reliability model and the fault tree.

78

Second example of the tutorial

- How to deal with non oriented interactions, redundancies.
- Modeling connections with failures



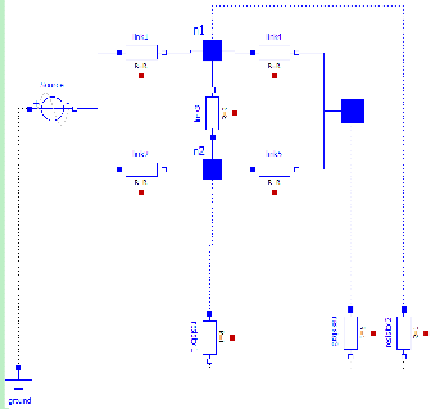
What are all the combinations of resistor failures that prevent the signal emitted by the source to reach the target?

Exercise : Telecom network

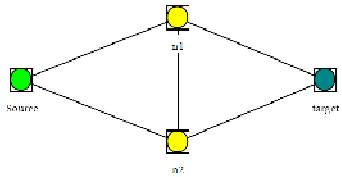
- Read explanations on the system in the next slides
- Generate the fault tree with top event corresponding to the fact that the pin “Target” is not connected to the Source.
- Find the minimal cut sets using the Grif/Tree tool
- In OpenModelica: simulate a failure in the telecom model by increasing the resistance in one of the links
- Generate a Figaro0 model
- Use Fig0debug to simulate an equivalent failure in the Figaro model

Mapping Figaro to Modelica

The topology in Figaro is much simpler



Modelica model



Figaro model

Simple telecom model

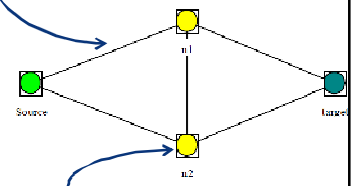
Mapping Figaro to Modelica – how?

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

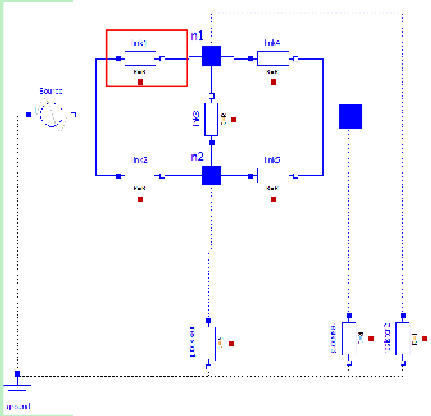
```
model Figaro_Object
  parameter String fullClassName;
  parameter String codeInstanceFigaro;
end Figaro_Object;

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

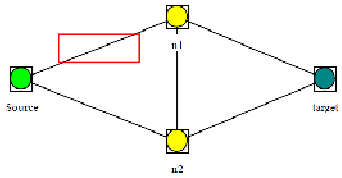
Links are full fledged objects in Figaro!



Mapping Figaro Links to Modelica



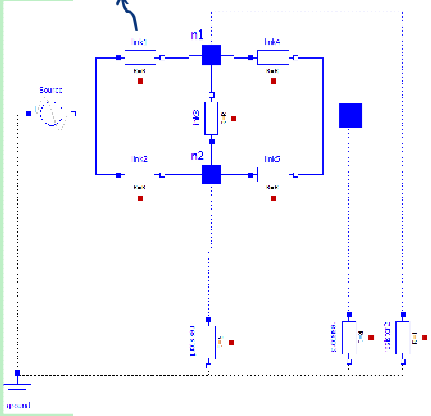
Modelica model



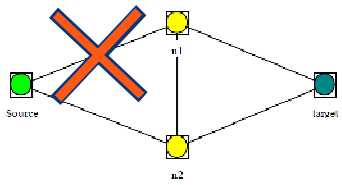
Figaro model

Simulation of a link failure

R = 100000



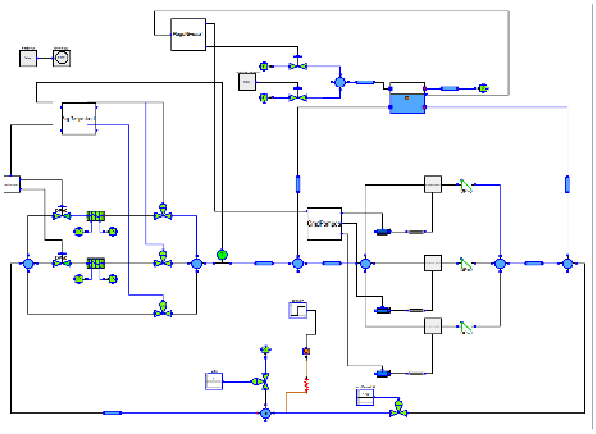
Modelica model



Figaro model: use Fig0debug to simulate the failure

Third example of the tutorial: SRI model

Mapping a thermohydraulic model from Modelica to Figaro: use case with complex libraries



Exercise : SRI model

- Go to the SRI folder in the exercises
- Load **first** the Thermosyspro library **then** the SRI model
- Configure the Figaro plugin
- Generate the fault tree
- Open the fault tree in the graphical editor GRIF/Tree
- Find the minimal cut sets

For this example, the two files of the Figaro Skelbo library (.fi and .bdc) are required (the Figaro export automatically loads both of them). See explanations in the documentation of the Skelbo knowledge base.

86

THANK YOU!

Where to find more information on tools:

http://openmodelica.org download OpenModelica with Figaro text-only tools

http://sourceforge.net/projects/visualfigaro/ download KB3 and other Figaro based tools

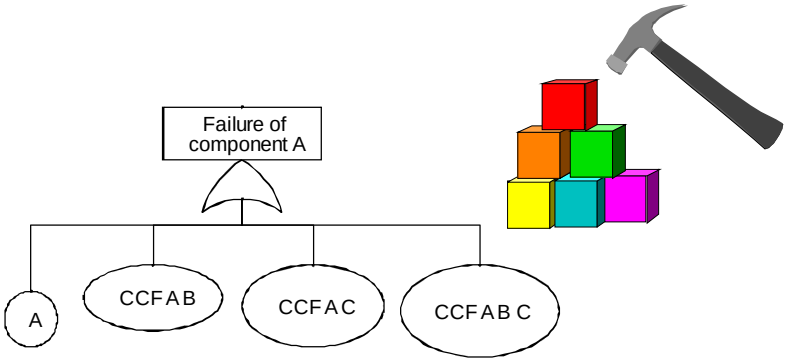
http://marc.bouissou.free.fr/ papers on the principles of Figaro tools

http://rdsoft.edf.fr then link to KB3: general information on KB3

Appendix

88

Modeling common cause failures



A belongs to a group of 3 identical components A, B, C, susceptible to CCFs

89

Advantages of using the Figaro language and processor

- Figaro: the first domain specific language for reliability (1990)
- Basis of the KB3 workbench: the reference tool at EDF
- Generic KB thanks to quantifiers
- Two ways to build FT
  - The CAT algorithm (Salem 1976) like most nowadays tools
  - **With macro components** => legible and structured FT
- Options to process negations (non coherent FT)
- Ability to produce correct FT for looped systems (cf. telecom example)

S. Salem, G.E. Apostolakis, D. Okrent : "*A computer oriented approach to fault tree construction*" EPRI-NP-288. 1976

90

Limitations of the CAT algorithm

3. CRITICISM OF THE CAT ALGORITHM

Let us consider a simple but useful example for the following discussions. Let us analyze a circuit composed of a series of three resistors (Fig. 2)

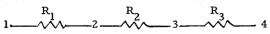


Figure 2

In particular we will analyze those failures causing the current loss at point 3. CAT models as presented in [1] will be used. Since this problem is essentially related to the existence of a signal in a particular point of the circuit, the choice will be of "current type".

Therefore, as the current flows from node 1 to node 4, the i-th resistor simplified model will be as follows:

I_i	SR_i	I_{i+1}
0	-	0
-	1	0
1	-	1

where - = don't care state
I_i = current at i-th node (0 = no current)
SR_i = state of i-th resistor
SR_i = 0 resistor good / 1 resistor failed

With orientation from left to right, in the generated FT for I_3=0, R_3 does not appear

With the macro component approach of KB3, the model states:
- failure of R_i => failure of MC
- failure of MC => I_i = 0

G. Squellati "*Critical review of the CAT algorithms for automated fault tree construction*". JRC technical note N° 1.06.01.80.84. October 1980

91

Model Translation Process to Hybrid DAE to Code