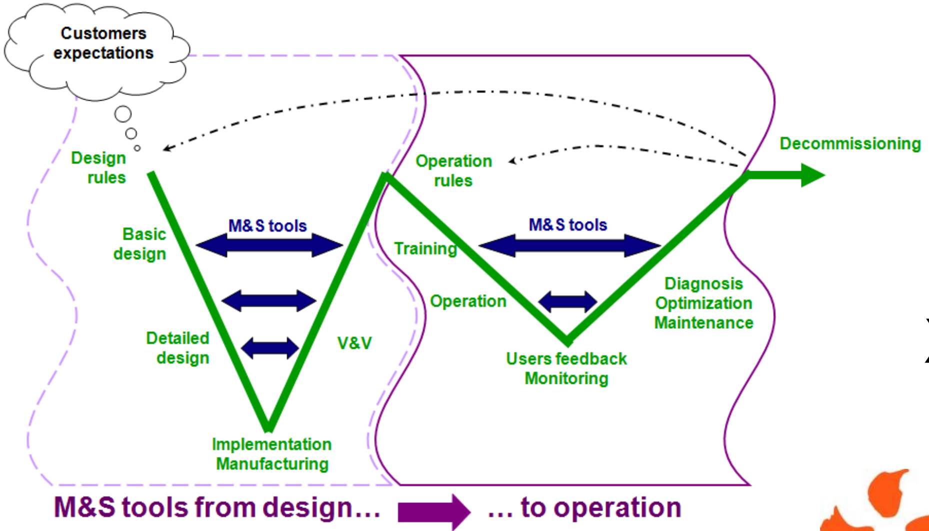
ITEA 2

INFORMATION TECHNOLOGY FOR EUROPEAN ADVANCEMENT



From Modelica models to dependability analysis

Marc Bouissou (EDF)
Xavier de Bossoreille (DLR)

eDF

DLR Robotics and Mechatronics Center

MODRIO

LMCS 2015

Industrial challenge

- Separation of design and dependability analysis workflows
- Consequences:
  - The same information is input twice
  - Consistence is not guaranteed
  - Long delay between the two kinds of studies => no possibility of feedback, except for very serious issues

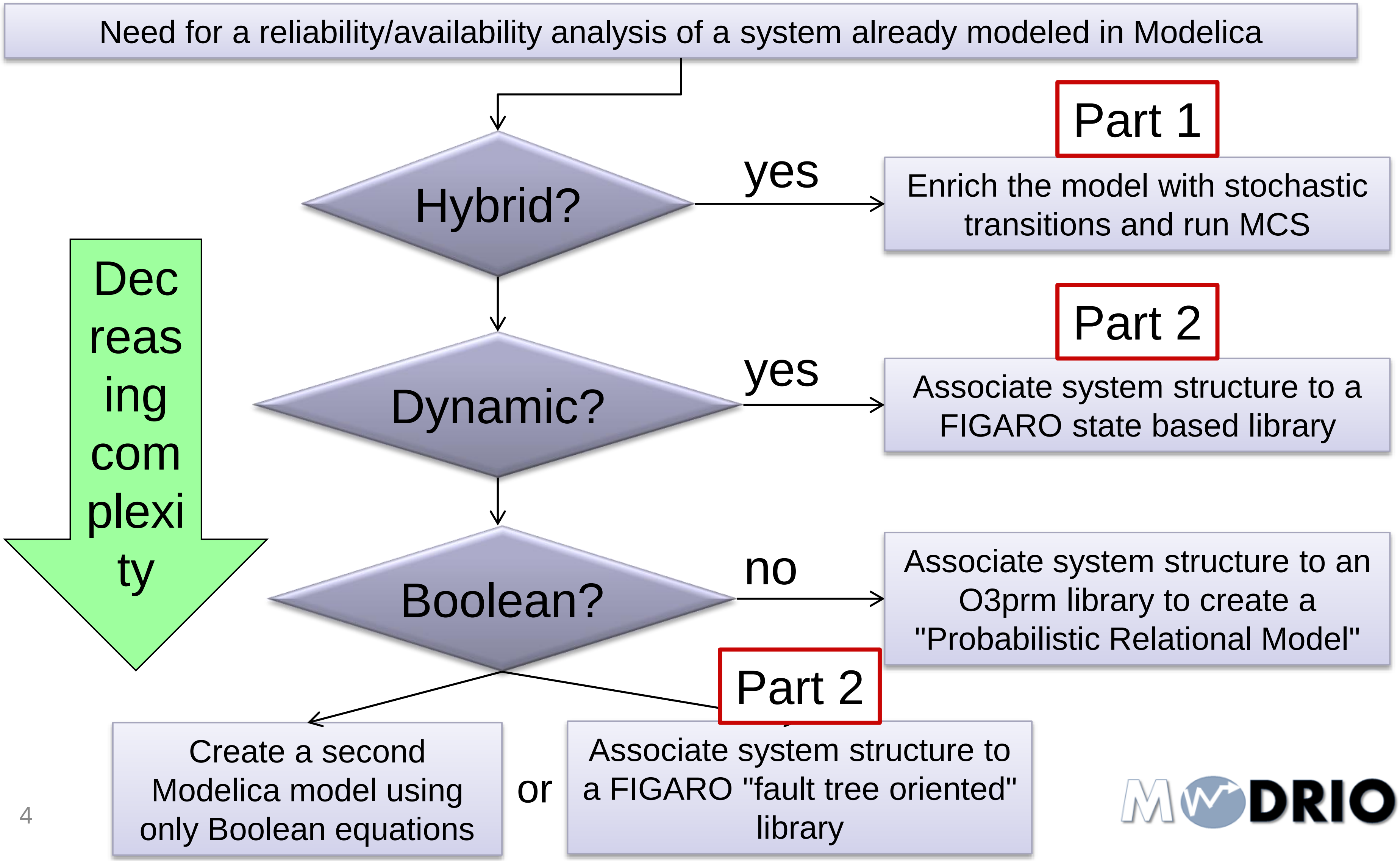
- Existing attempts to link design models to dependability:
  - Limited to fault tree production (+ FMEA)
  - Rely on a simple algorithm (assembly of FT parts, not a true *generation*), works for control systems but not for physical systems
  - No storage of *generic* reliability models

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)

Various ways to derive stochastic models from Modelica models



Need for a reliability/availability analysis of a system already modeled in Modelica

Hybrid?

yes

Part 1

Enrich the model with stochastic transitions and run MCS

Dynamic?

yes

Part 2

Associate system structure to a FIGARO state based library

Boolean?

no

Associate system structure to an O3prm library to create a "Probabilistic Relational Model"

Decreasing complexity

Create a second Modelica model using only Boolean equations

or

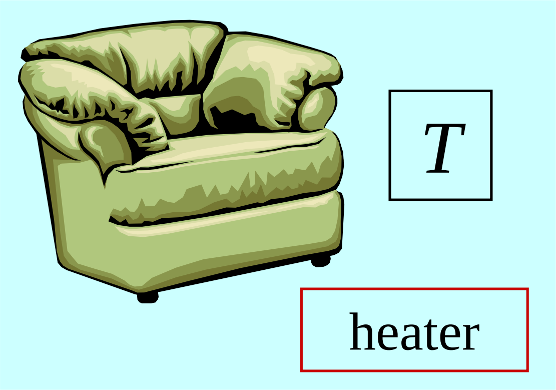
Part 2

Associate system structure to a FIGARO "fault tree oriented" library

Outline of part 1

- Introduction of stochastic transitions in Modelica
  - A simple test case
  - Modeling it using state machines
- Two more complex examples
- Perspectives

The "Heated room" test case

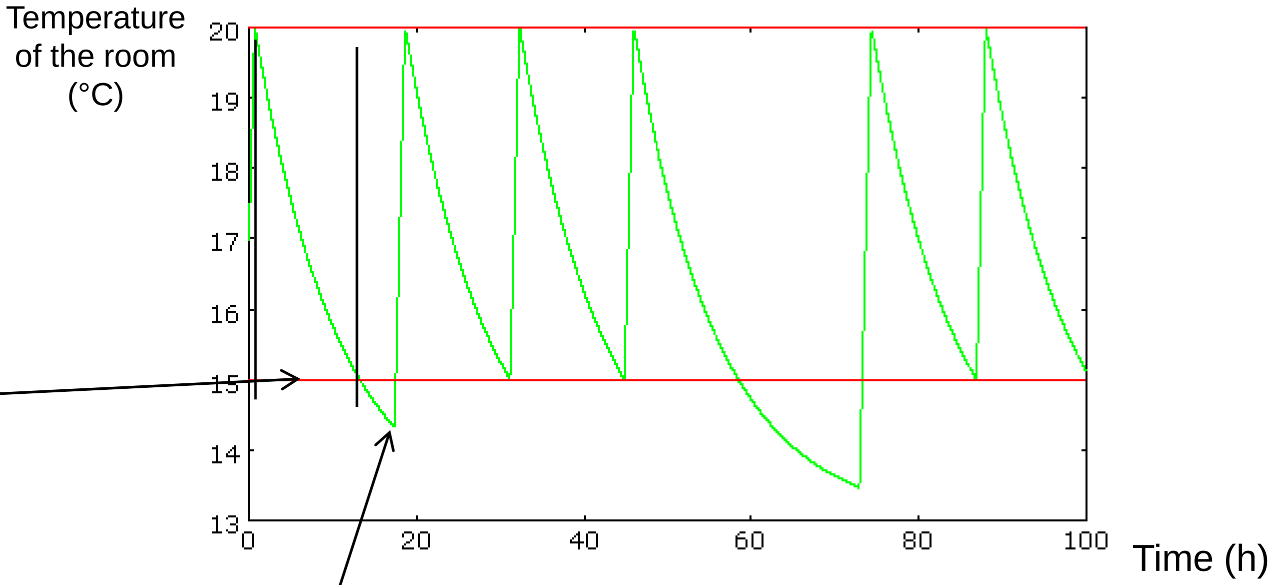


T_E External temperature

Heater:

- on at Tmin, off at Tmax
- subject to random failures and repairs
- exponential distributions for times to failure and times to repair

An example of single (random) trajectory of T

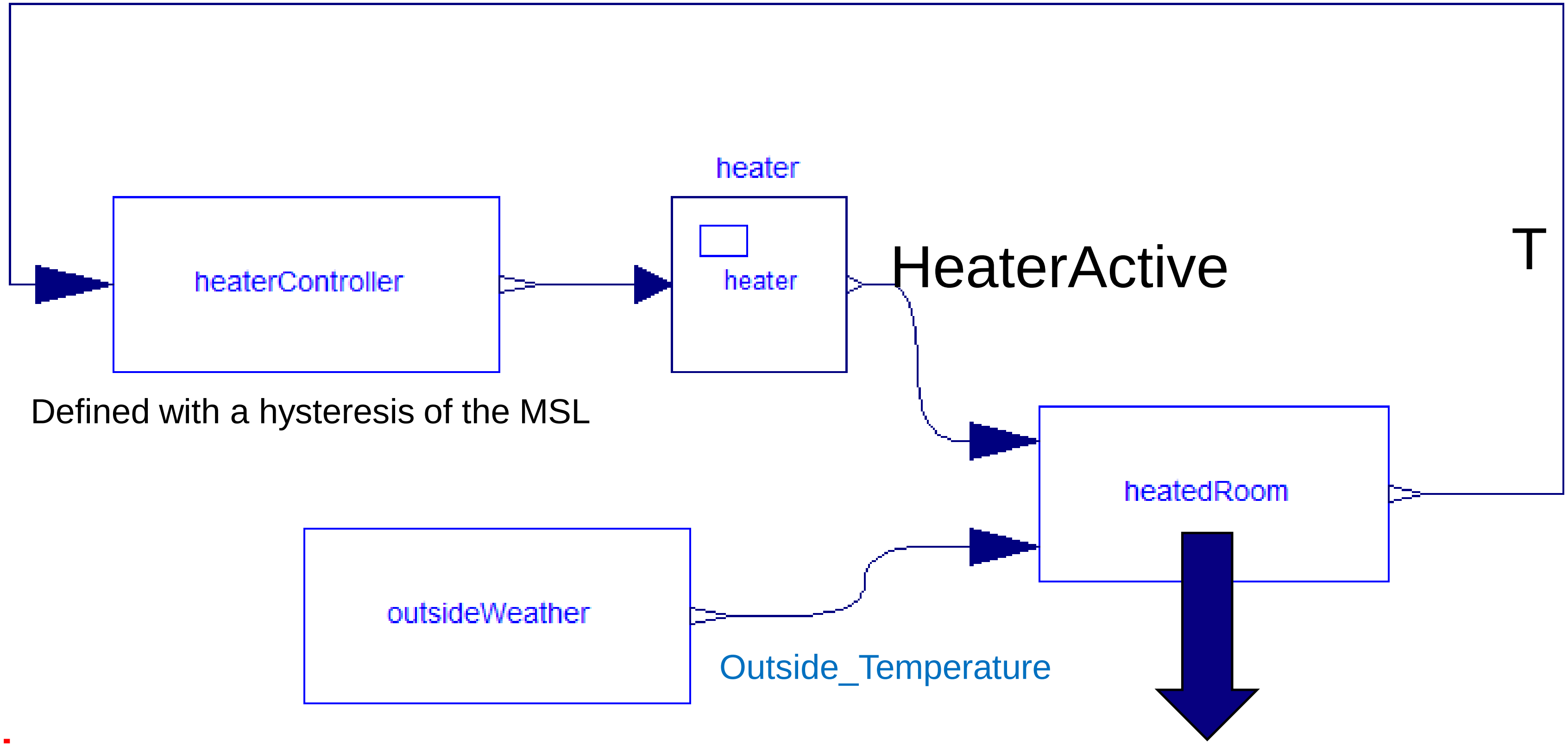


A failure of the heater occurred somewhere here

And was repaired at that time

This is a PDMP:
Piecewise Deterministic Markov Process

Modelica model: structure



Defined with a hysteresis of the MSL

equation

der(T)= CoeffHeaterController*HeaterActive + CoeffOutsideTemperature*(Outside_Temperature - T) ;

The heater: a first model

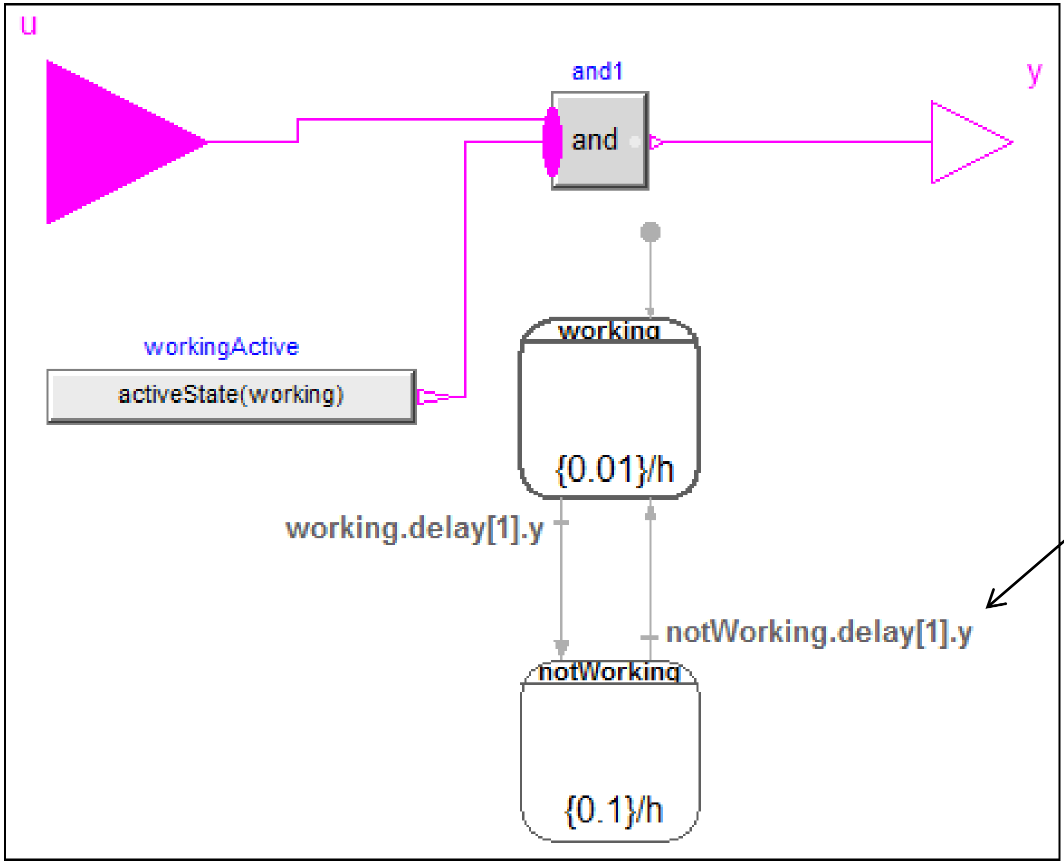
```
algorithm
when initial() then
  F := seed; //each calculation of F will yield a pseudo random number in [0,1]
end when;
// Attention: the two following rules must not be merged in a single one!
when initial() then   //calculating the first random working time
    F := mod(a*F+c, m);
    x := F / m;
    X:= (-log(1-x))/lambda;
end when;
when  working then //random draw of the next working time
    F := mod(a*F+c, m);
    x := F / m;
    X:= (-log(1-x))/lambda;
end when;
// X is the working time
when working and (time - starttime_working) > X then
    working := false;
    starttime_notworking := time;
end when;

…. Similar instructions for repairs
```

```
// Input-output relation
equation
if working then
  y =  u;
 else
    y =  0.;
end if;
```

Very cumbersome and error prone!

The heater (in Modelica 3.3)



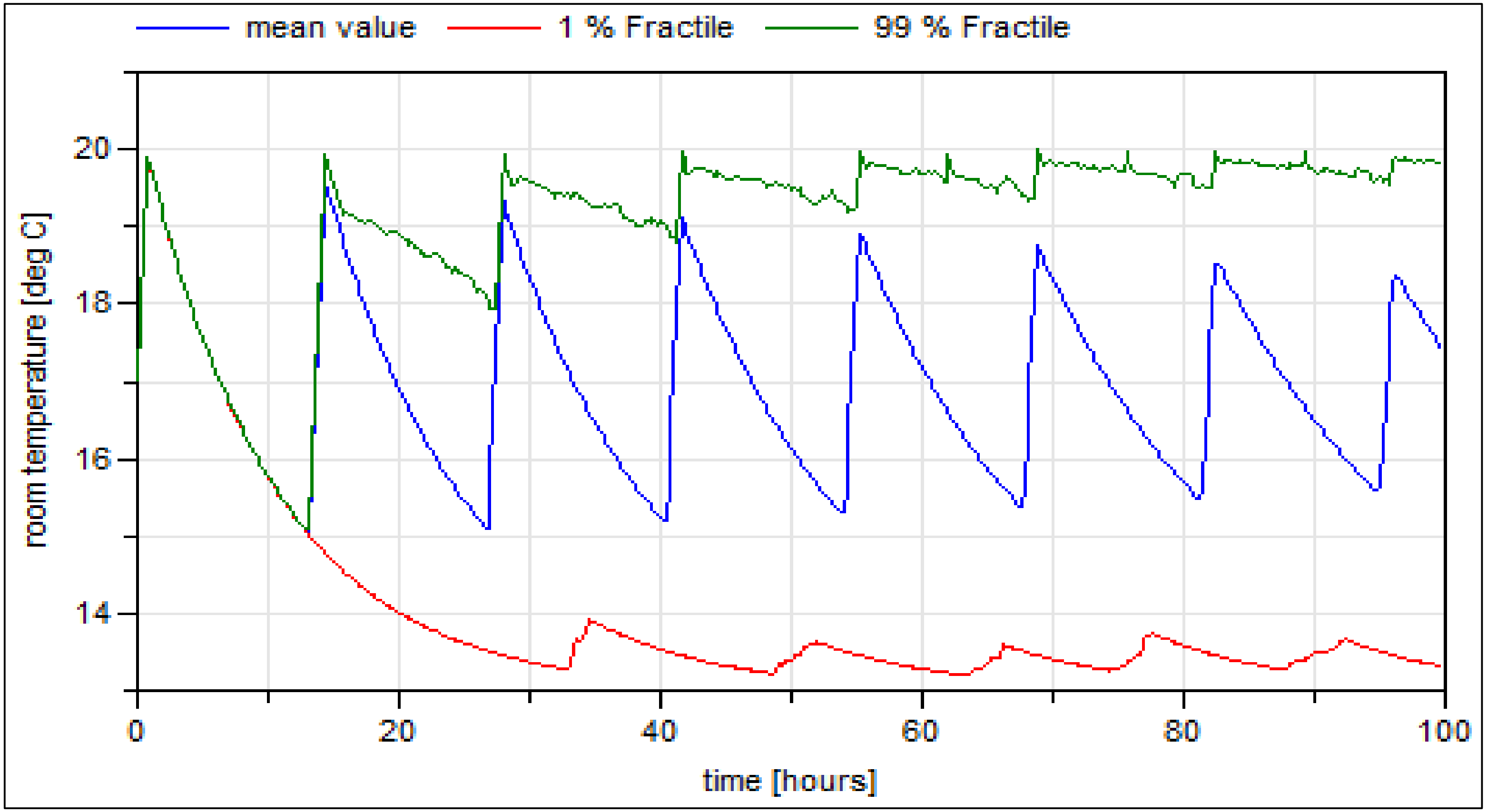
Essentially the same principles as what was shown on slide "Heater: a first model", but taking advantage of abstraction mechanisms provided by Modelica.

The user does not have to write any code

We use a continuous time state machine to describe the random behavior of the heater (only available in Dymola, cf. *Modelica extensions for multi-mode DAE systems*, Elmqvist, Mattsson and Otter, 2014)

Results: Temperature
(10000 random trajectories)

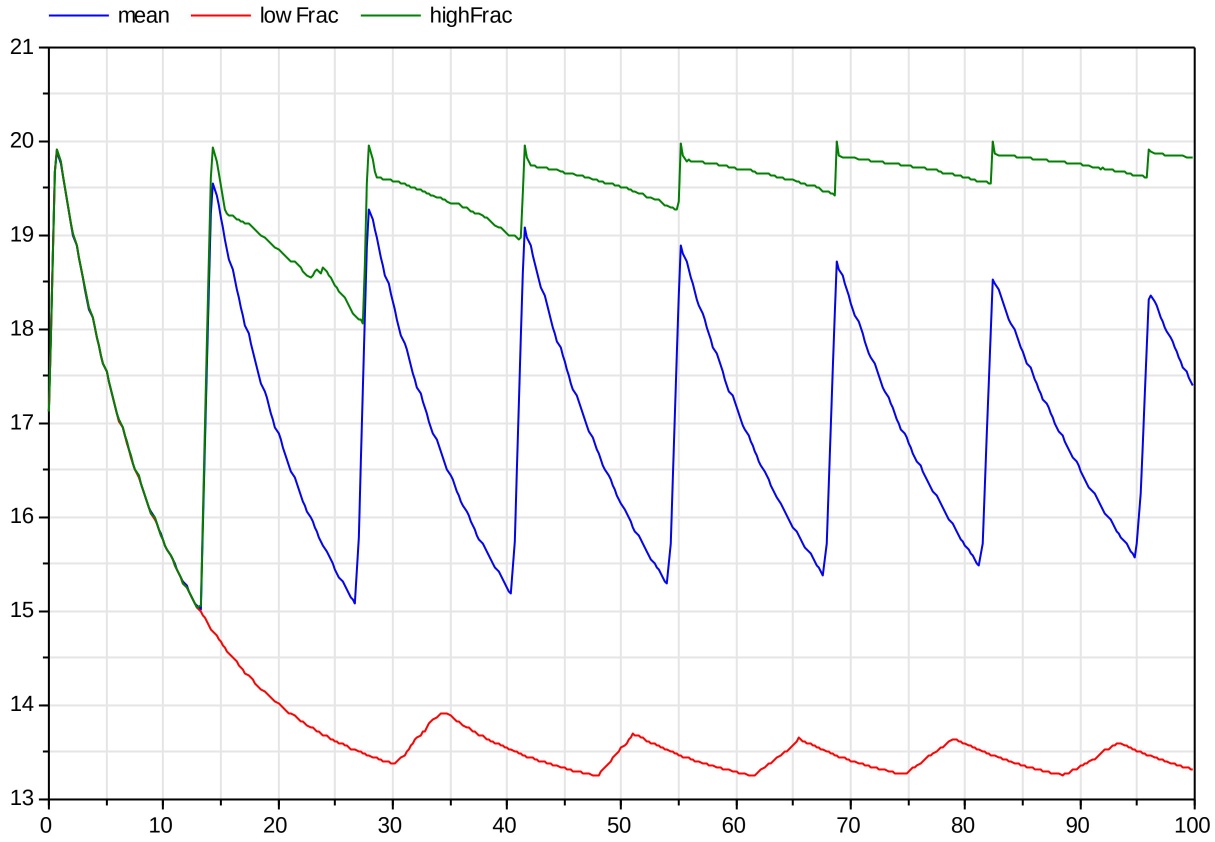
T(°C)



time (h)

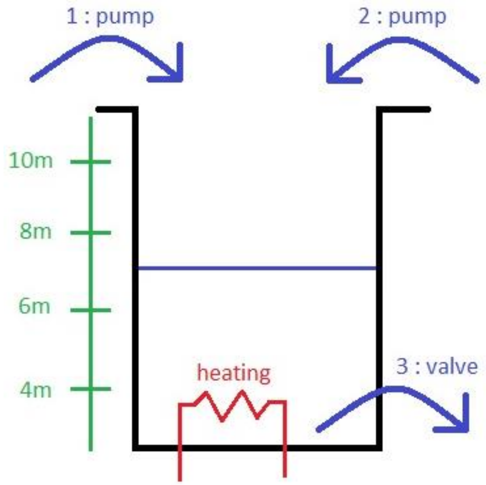
Results

- Heated-room
- 1 000 000 sim
- 1h30



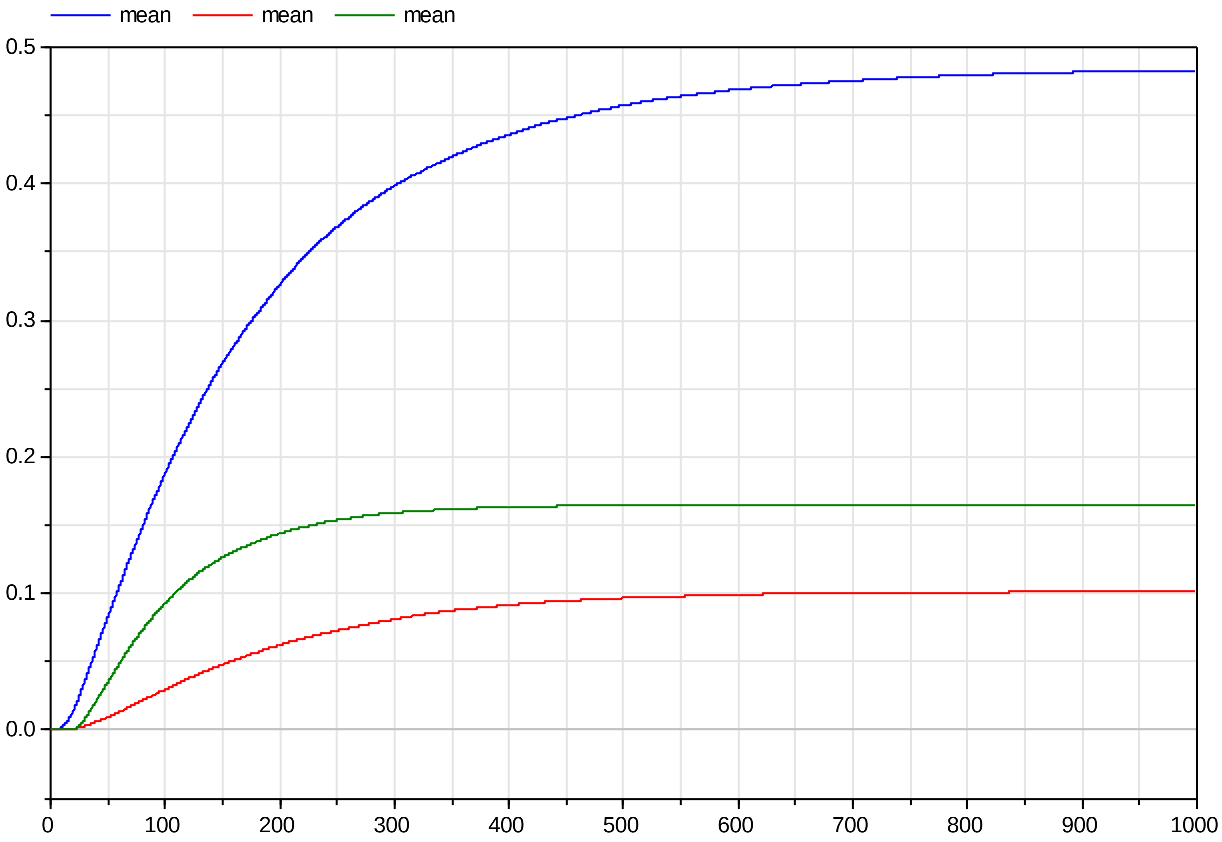
Ex 2 : heated-tank

- Tank with 2 pumps and 1 valve
- Failure rates dependent on temperature
- 2 tied continuous variables : temperature and level
- A test case solved with dozens of methods in the literature!



Results

- Heated-tank
- 100 000 sim
- 2h15
- Zhang et al. :
  1 min 30s
  or
  23h



Conclusion

- 3 complementary test cases were solved
- Random state machines work… in Dymola
- Monte Carlo works and is fairly efficient
- Anyway, faster than some other general* methods

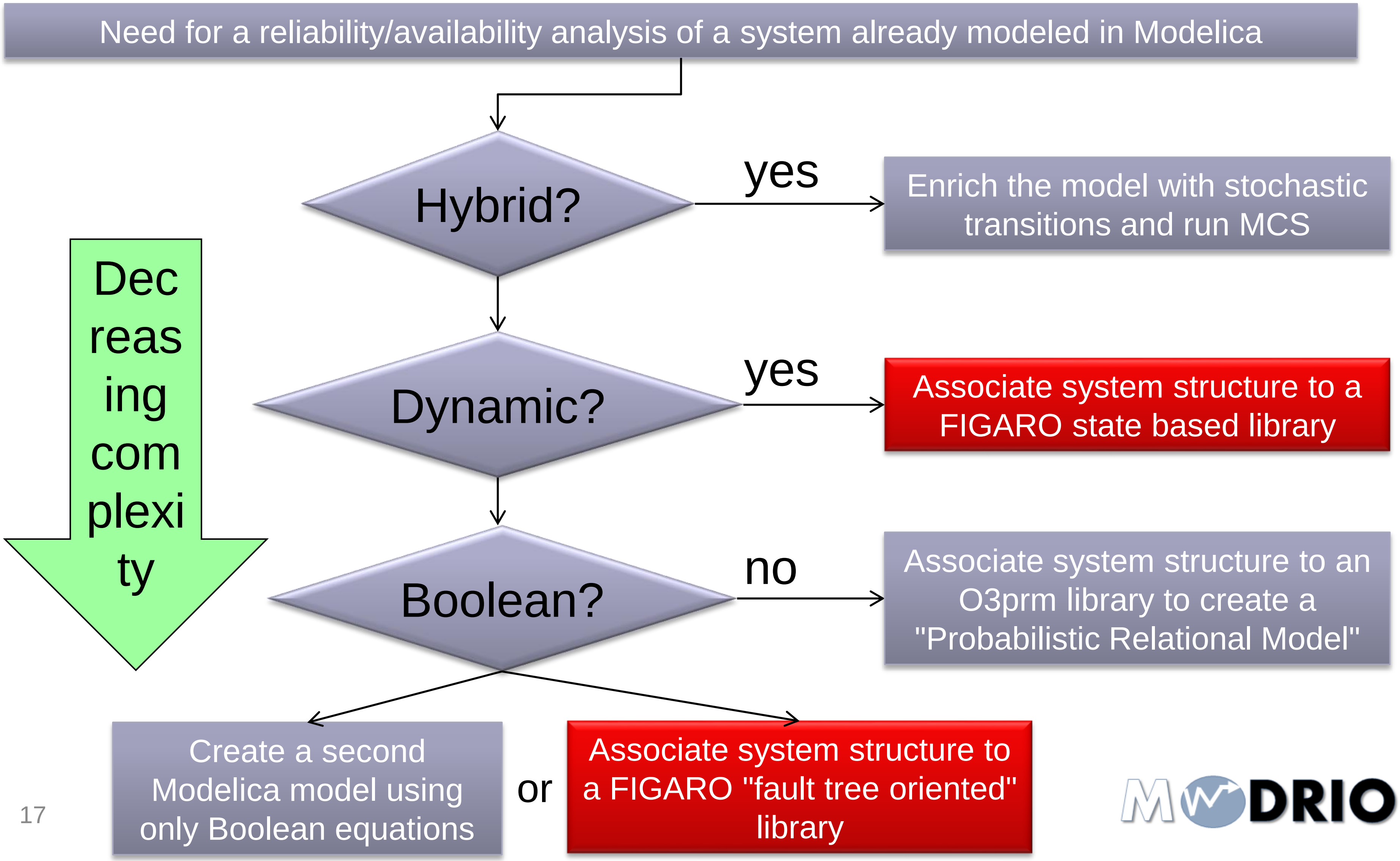
*General = that do not use analytical solutions for DAE

Perspectives

- Apply those principles to a complex system: a data center with electrical supplies, thermohydraulic cooling system
- Requires the extension of components models with stochastic state machines to represent failures, repairs, reconfigurations…
- Refine the Monte Carlo function

- Convince the Modelica association to introduce "native" stochastic transitions in Modelica!

Various ways to derive stochastic models from Modelica models



Need for a reliability/availability analysis of a system already modeled in Modelica

Hybrid?

yes

Enrich the model with stochastic transitions and run MCS

Dynamic?

yes

Associate system structure to a FIGARO state based library

Boolean?

no

Associate system structure to an O3prm library to create a "Probabilistic Relational Model"

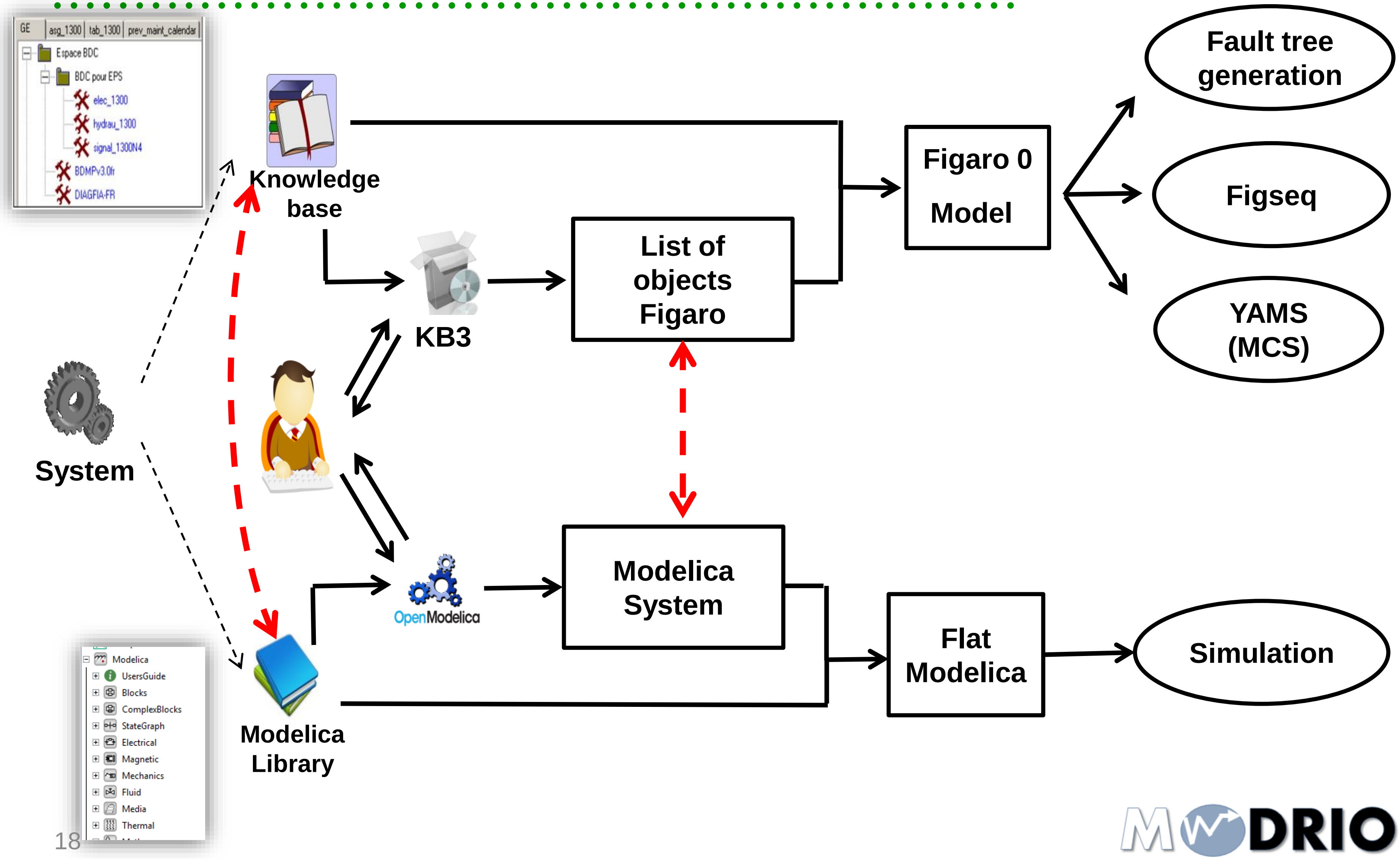
Create a second Modelica model using only Boolean equations

or

Associate system structure to a FIGARO "fault tree oriented" library

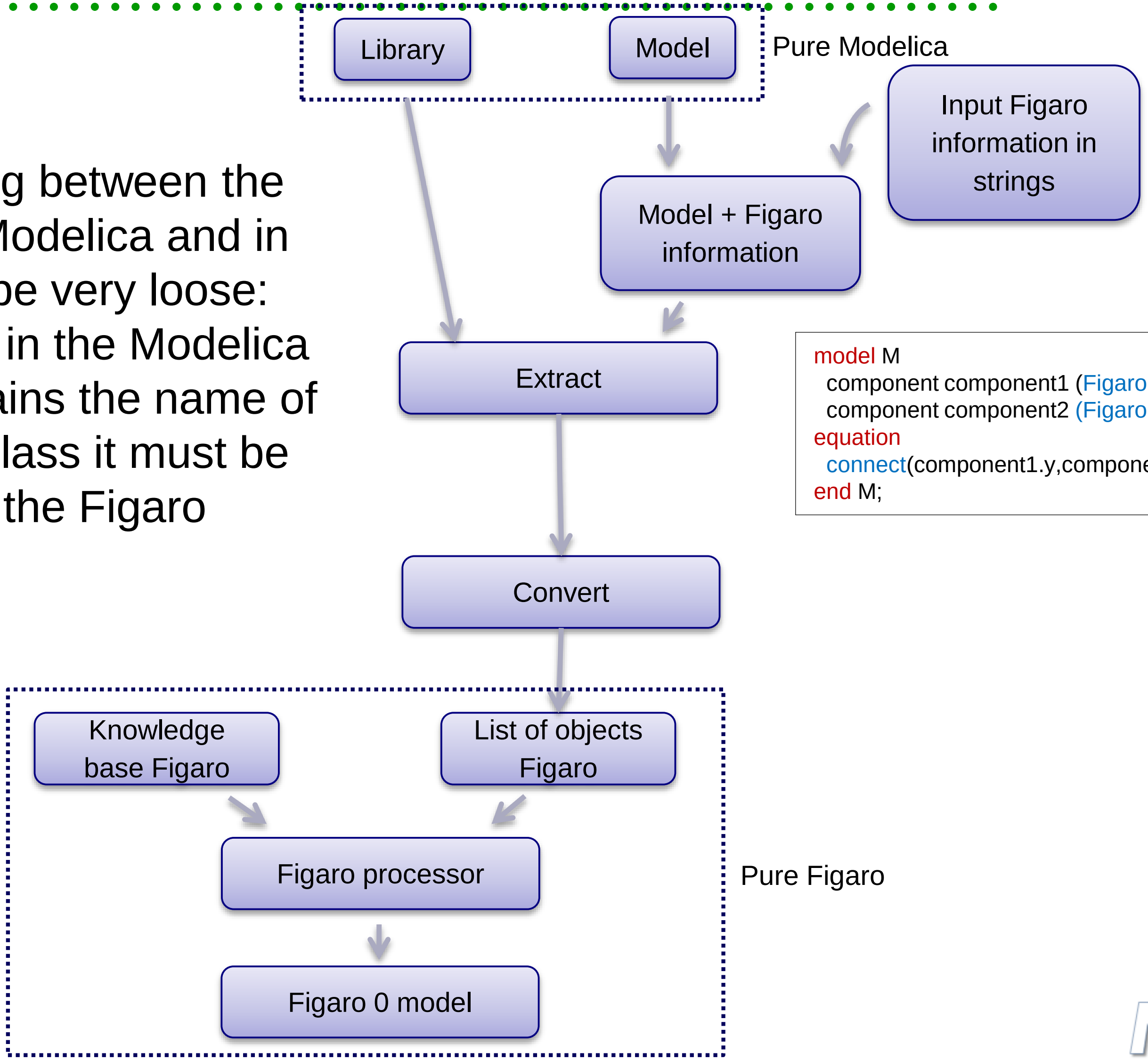
Decreasing complexity

The Modelica tools and KB3 work in similar ways



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



```
model M
  component component1 (Figaro=”Figaro code”);
  component component2 (Figaro="Figaro code”);
equation
  connect(component1.y,component2.u)
end M;
```

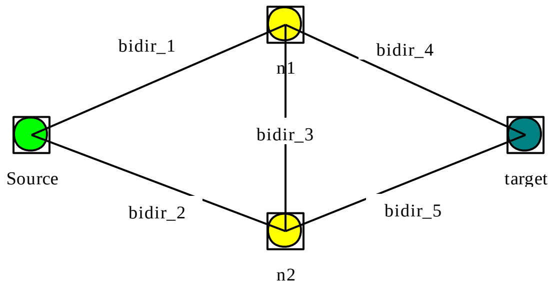
Advantages of using the Figaro language and processor

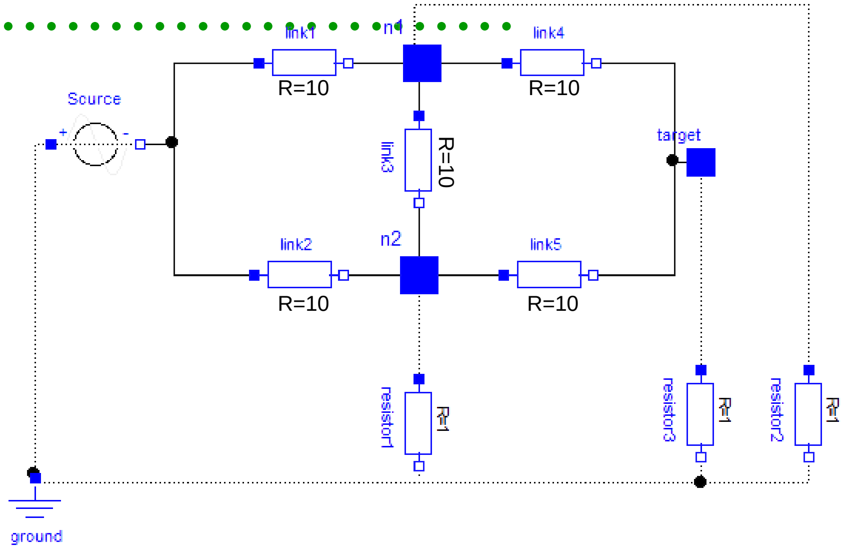
- Figaro: the first domain specific language for reliability (1990)
- Basis of the KB3 workbench: the reference tool at EDF
- Generic KB thanks to quantifiers
- Two ways to build FT
  - The CAT algorithm (Salem 1976) like most nowadays tools
  - **With macro components** => legible and structured FT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

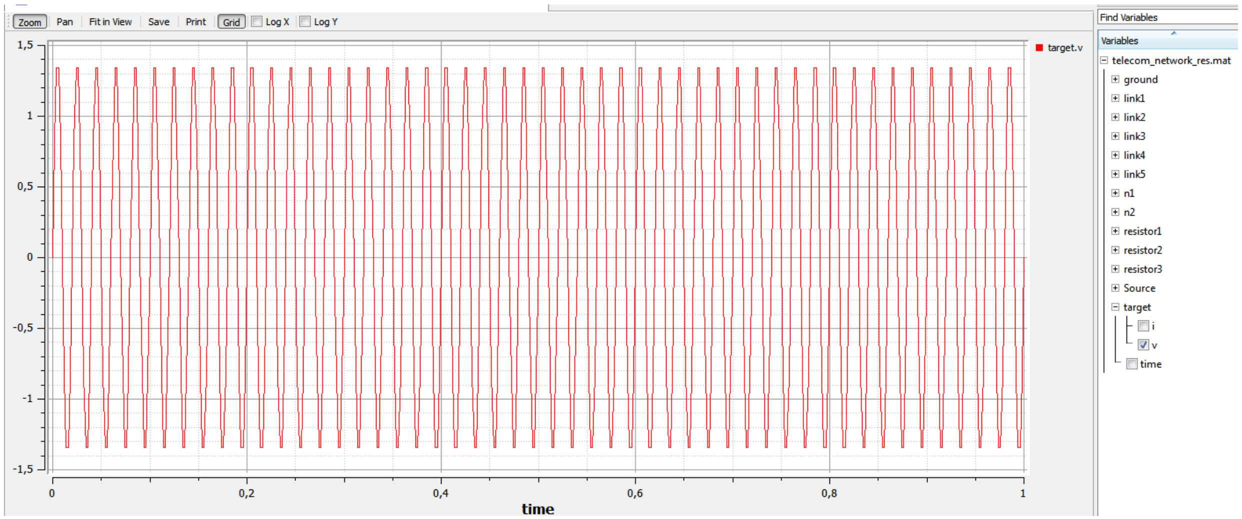
S. Salem, G.E. Apostolakis, D. Okrent : "A computer oriented approach to fault tree construction" EPRI-NP-288. 1976

Example: telecom network

Nominal mode

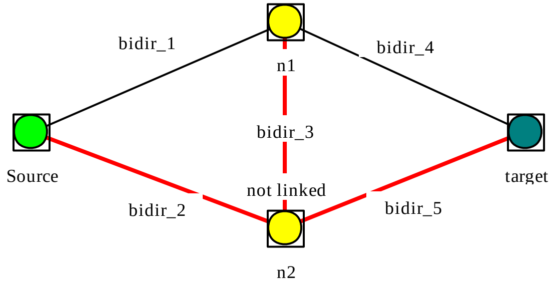


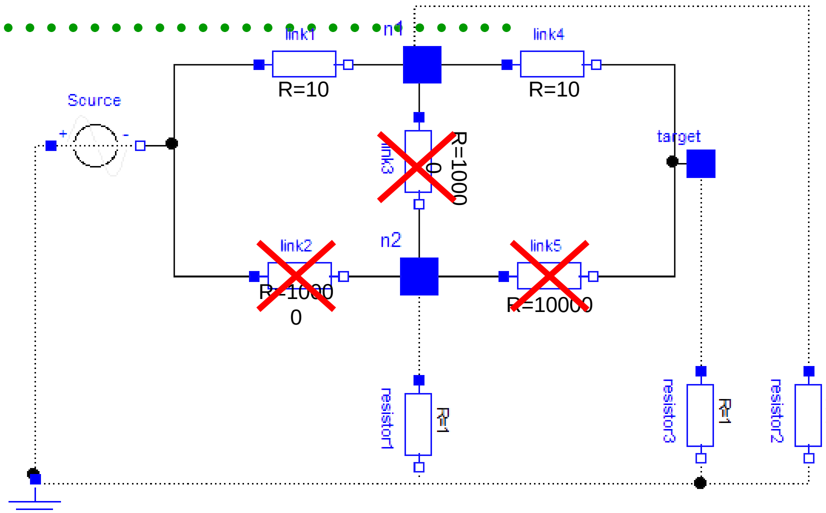




Example: telecom network

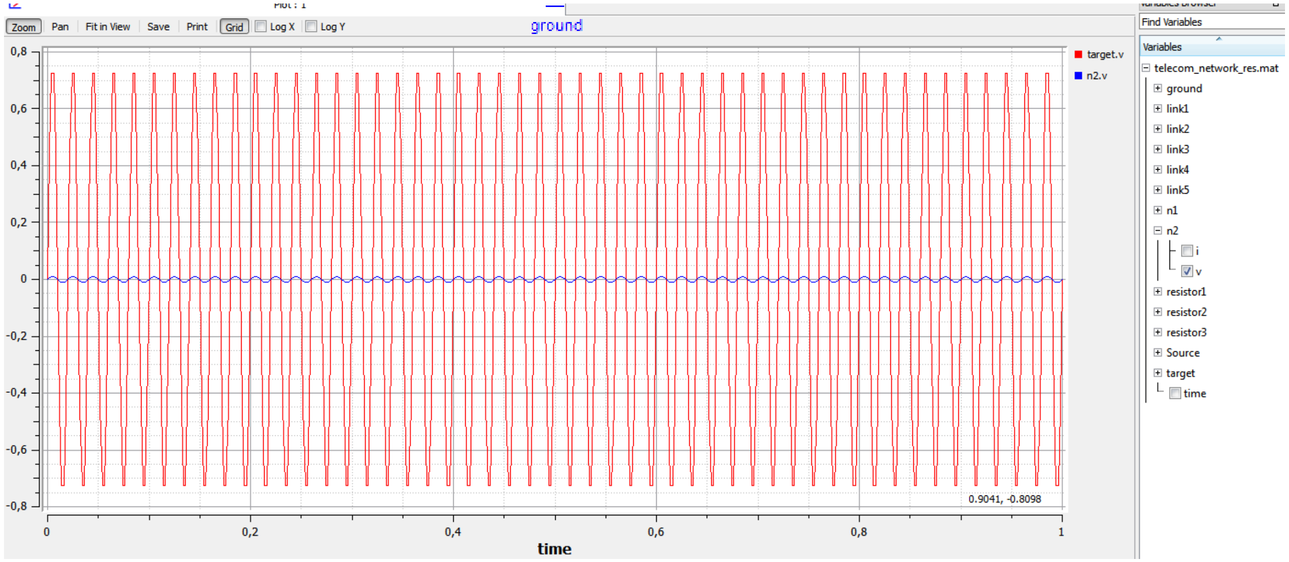
Degraded mode





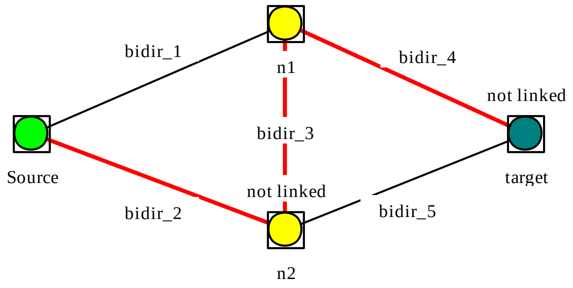
target.v

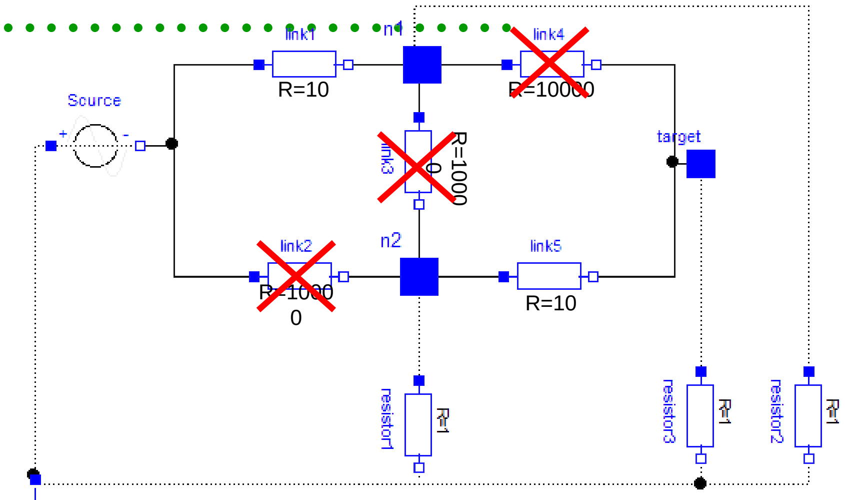
n2.v



Example: telecom network

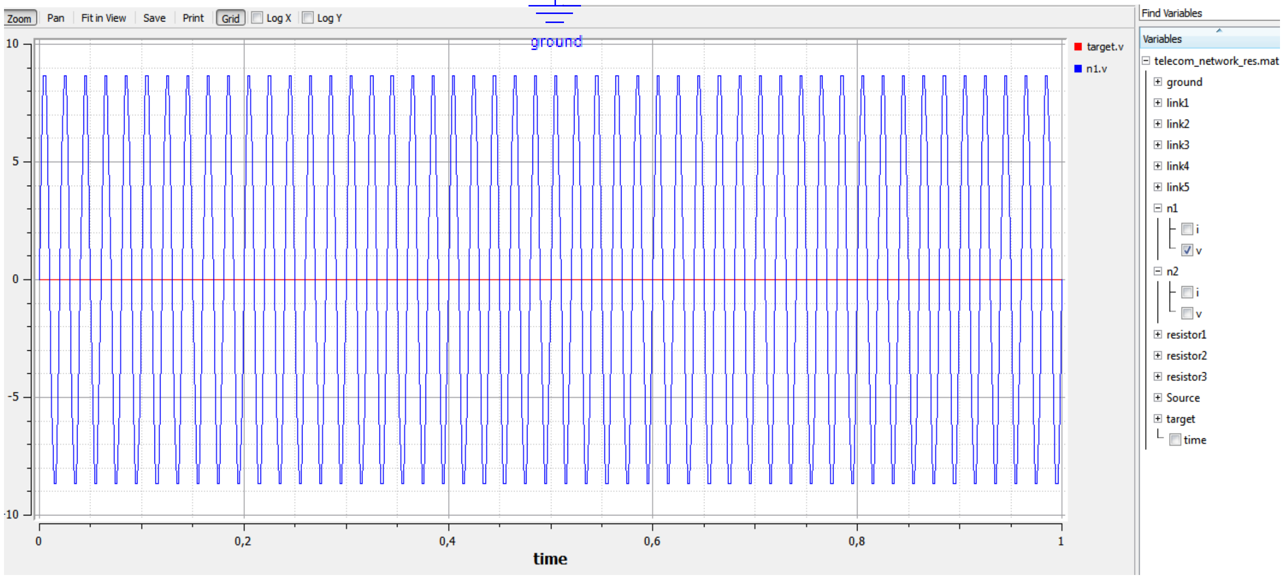
Failure of the mission





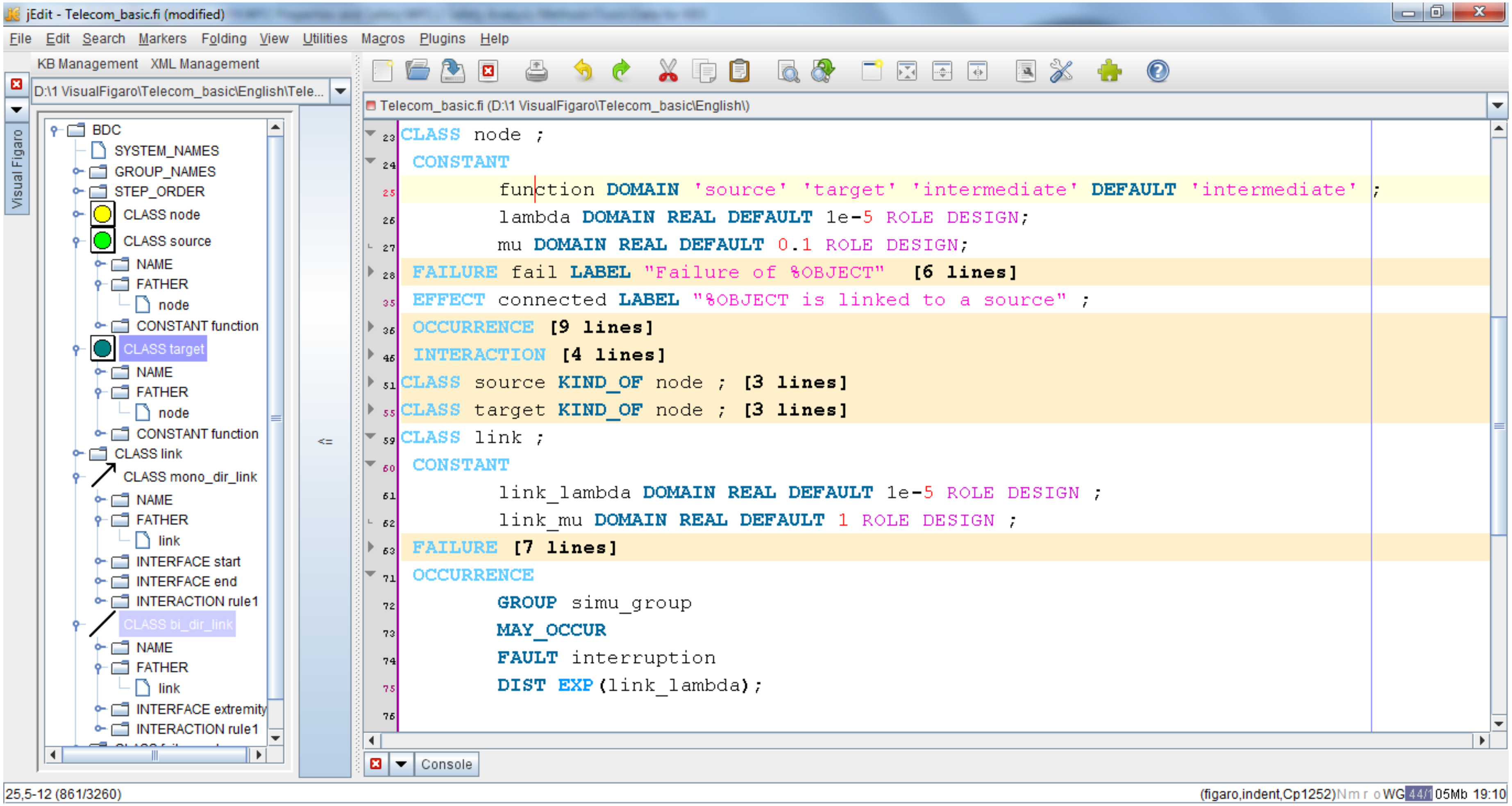
n1.v

target.v



We want to get the exhaustive list of minimal cut sets, without trying all combinations of failures in Modelica!

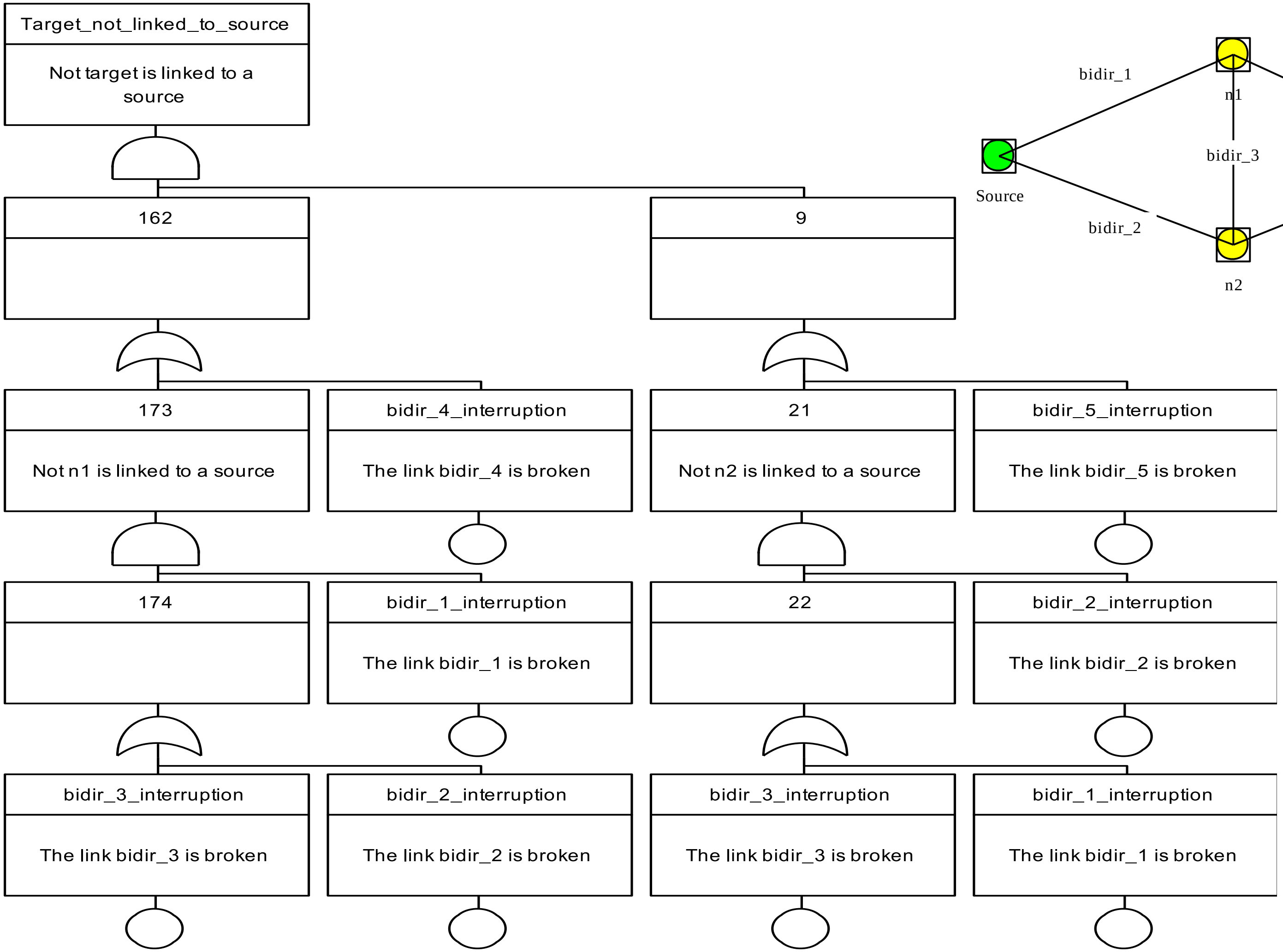
Telecom network: the FIGARO library



```
CLASS node ;
 CONSTANT
        function DOMAIN 'source' 'target' 'intermediate' DEFAULT 'intermediate' ;
        lambda DOMAIN REAL DEFAULT 1e-5 ROLE DESIGN;
        mu DOMAIN REAL DEFAULT 0.1 ROLE DESIGN;
 FAILURE fail LABEL "Failure of %OBJECT"   [6 lines]
 EFFECT connected LABEL "%OBJECT is linked to a source" ;
 OCCURRENCE [9 lines]
 INTERACTION [4 lines]
CLASS source KIND_OF node ; [3 lines]
CLASS target KIND_OF node ; [3 lines]
CLASS link ;
 CONSTANT
        link_lambda DOMAIN REAL DEFAULT 1e-5 ROLE DESIGN ;
        link_mu DOMAIN REAL DEFAULT 1 ROLE DESIGN ;
 FAILURE [7 lines]
 OCCURRENCE
        GROUP simu_group
        MAY_OCCUR
        FAULT interruption
        DIST EXP(link_lambda);
```

The library is edited in the Visual Figaro editor
Total length: 115 lines

Example: telecom network
Fault tree generated by the FIGARO processor



Target_not_linked_to_source

Not target is linked to a source

162

9

173

Not n1 is linked to a source

bidir_4_interruption

The link bidir_4 is broken

21

Not n2 is linked to a source

bidir_5_interruption

The link bidir_5 is broken

174

bidir_1_interruption

The link bidir_1 is broken

22

bidir_2_interruption

The link bidir_2 is broken

bidir_3_interruption

The link bidir_3 is broken

bidir_2_interruption

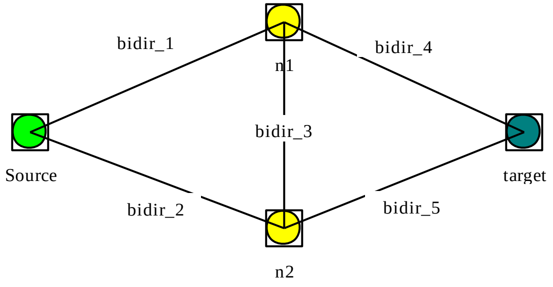
The link bidir_2 is broken

bidir_3_interruption

The link bidir_3 is broken

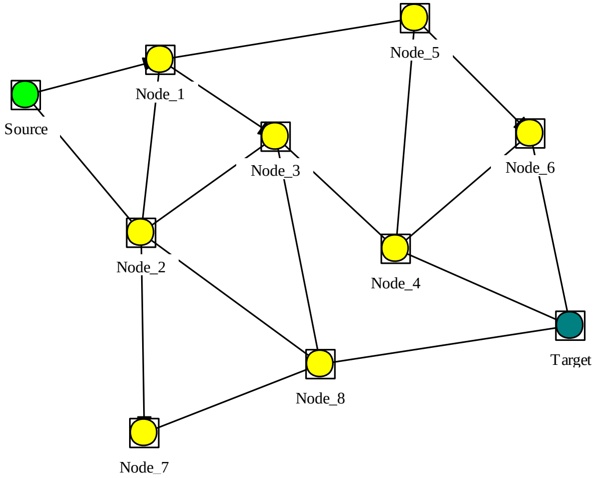
bidir_1_interruption

The link bidir_1 is broken

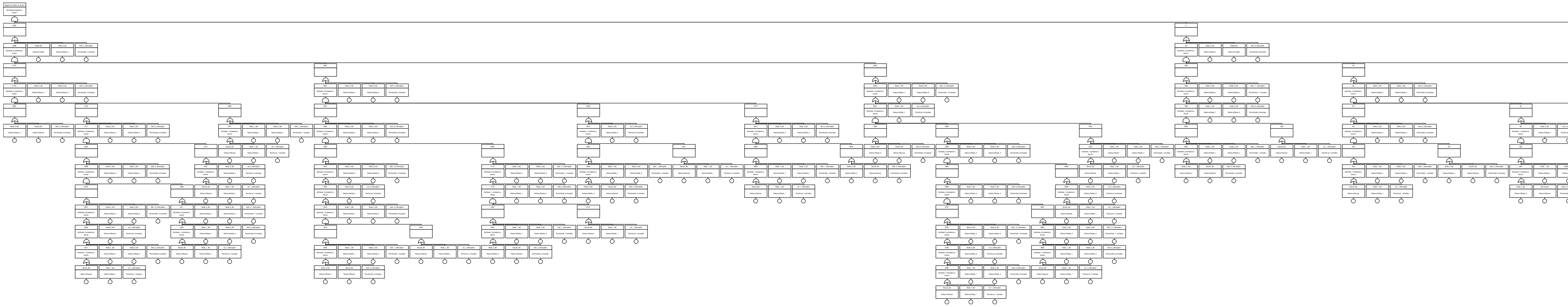


(considering nodes as perfect to get a small fault tree)

It could also work for a more complicated case…



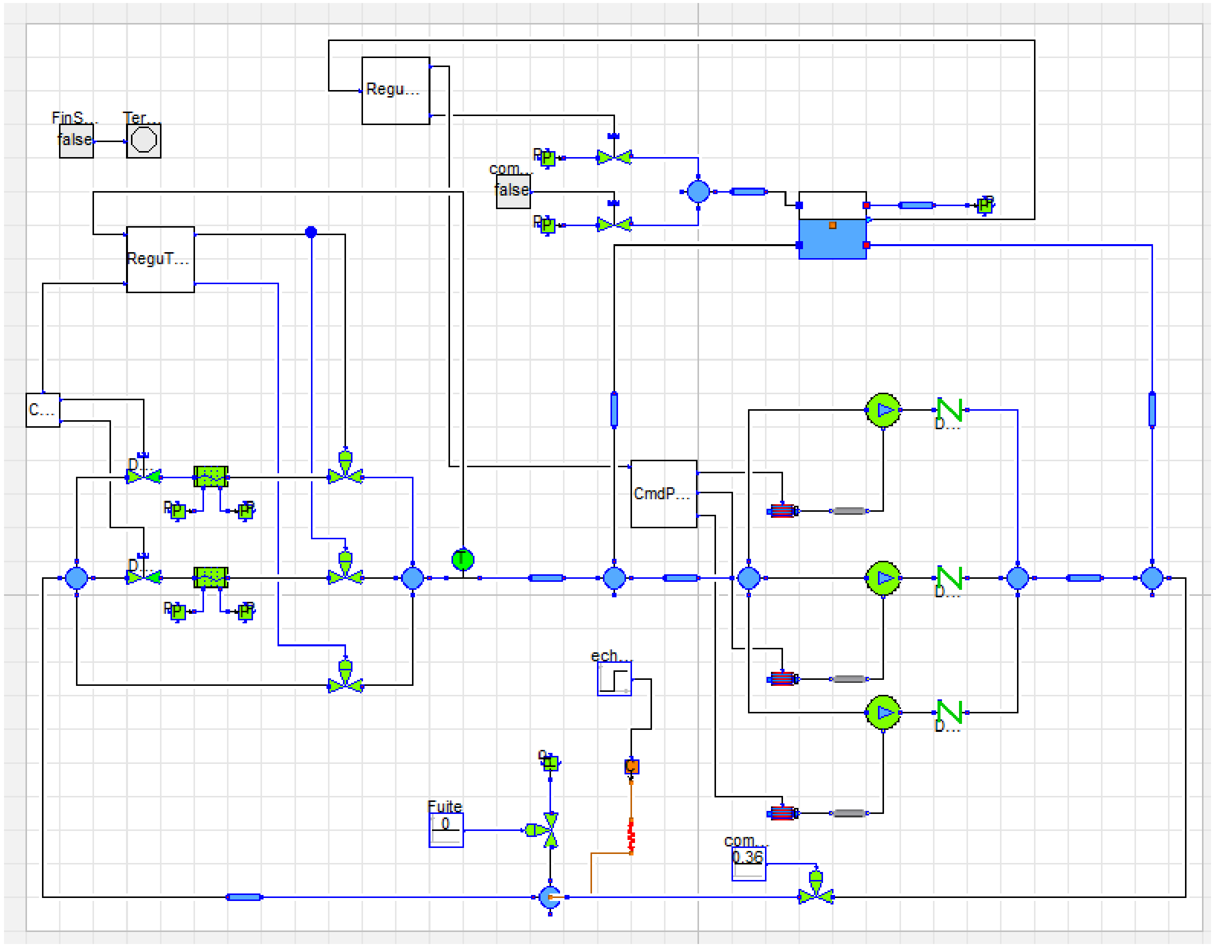
Generation of a fault tree eliminating loops by KB3 (failures of nodes and links taken into account)



First third of the fault tree…

Example 2: thermohydraulic system (SRI)

- Real use case of MODRIO
- Cooling system in a nuclear power plant



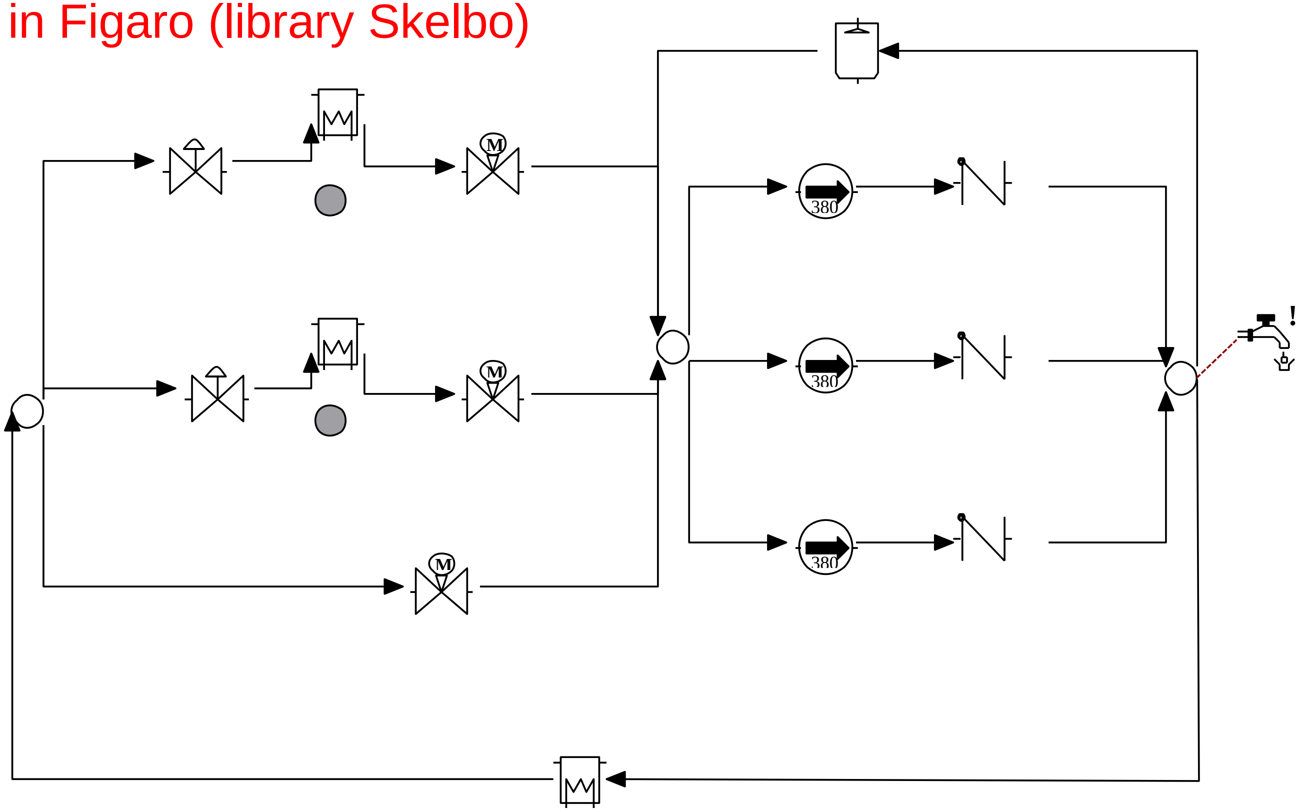
SRI in Modelica
(library ThermosysPro)

The SRI as it would be input in KB3

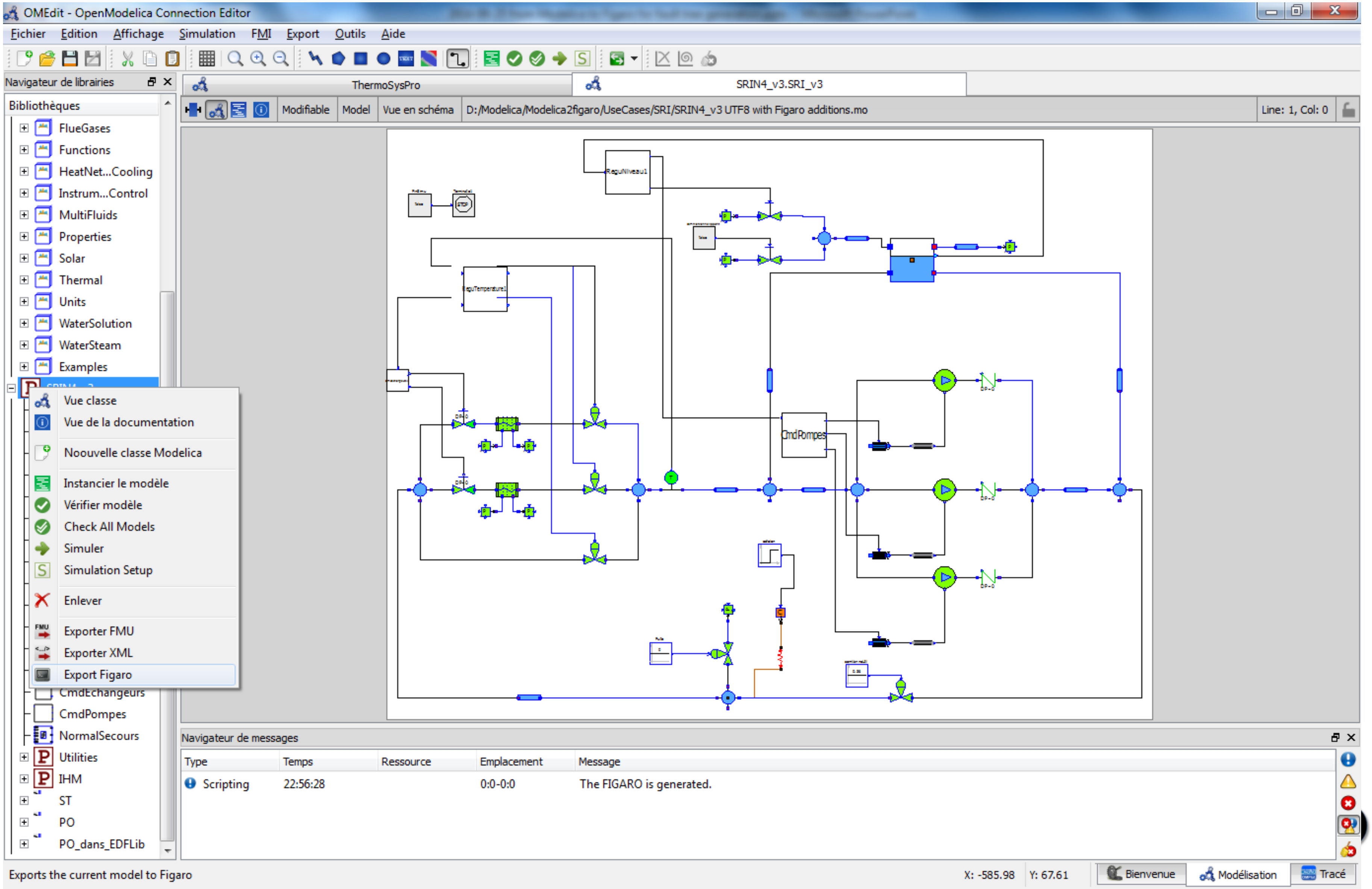
- The Figaro library (Skelbo) was developed long ago, independently from the SRI model in Modelica

SRI in Figaro (library Skelbo)

In fact, the same topological information is input in the Modelica model



Demonstration: OpenModelica



D:/Modelica/Modelica2figaro/UseCases/SRI/SRIN4_v3 UTF8 with Figaro additions.mo

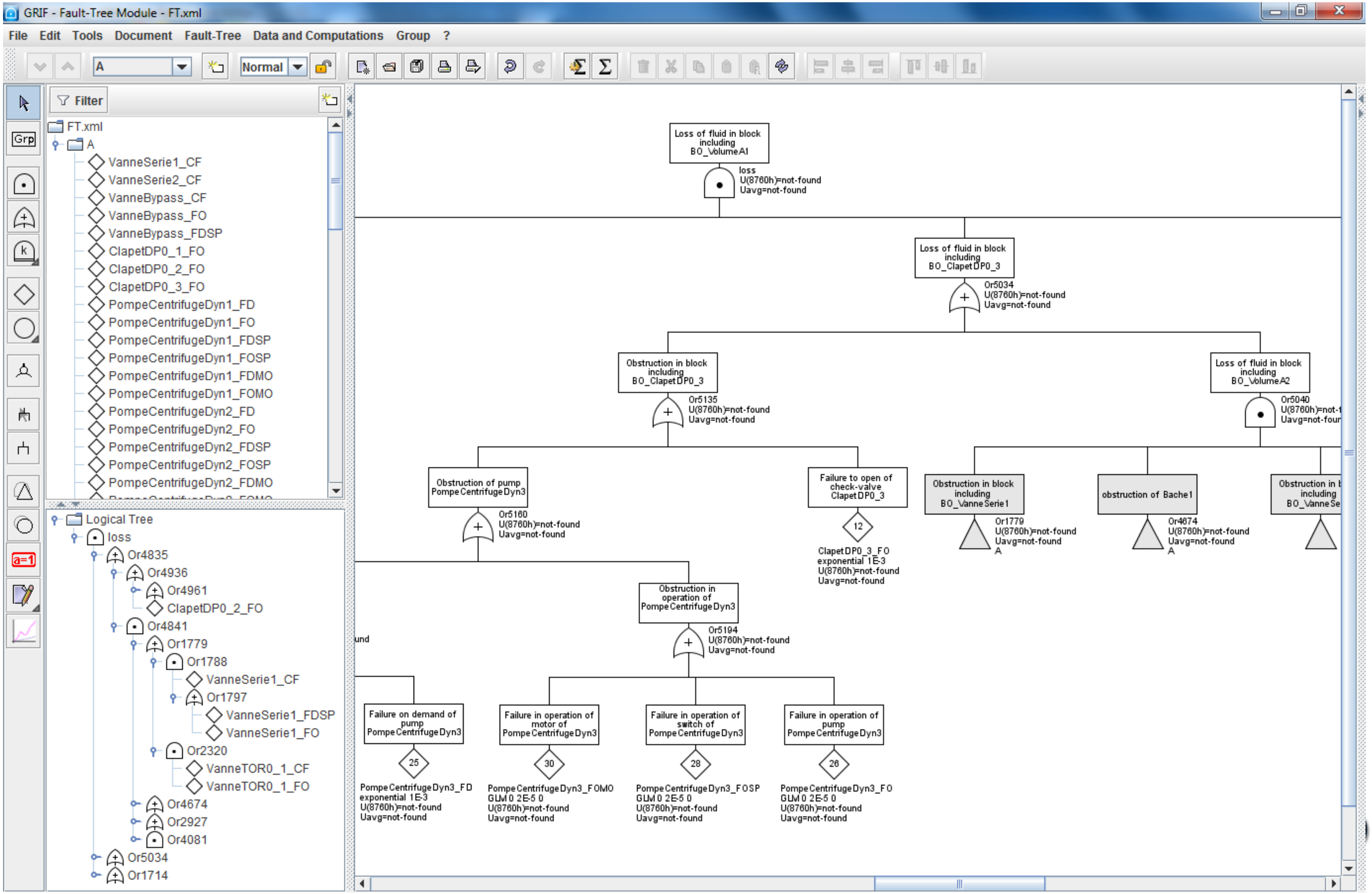
Type	Temps	Ressource	Emplacement	Message
Scripting	22:56:28		0:0-0:0	The FIGARO is generated.

Exports the current model to Figaro

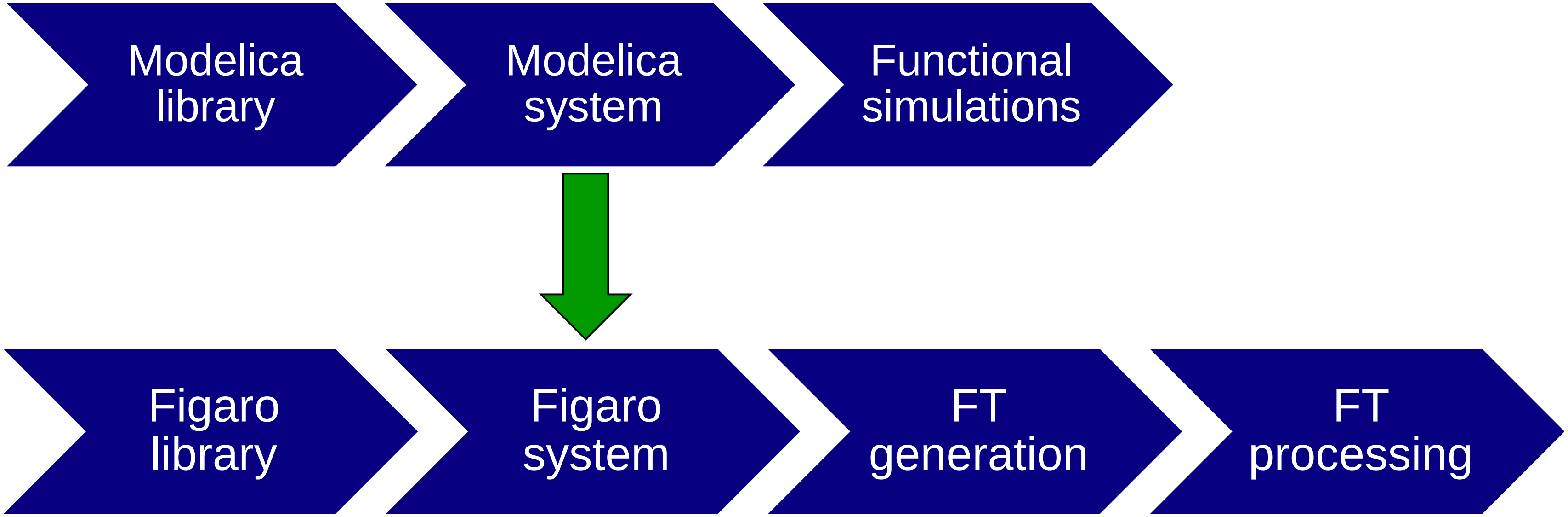
X: -585.98 Y: 67.61

The generated fault tree in the GRIF editor

Presentation at DCDS 2015



Summary: engineering workflow

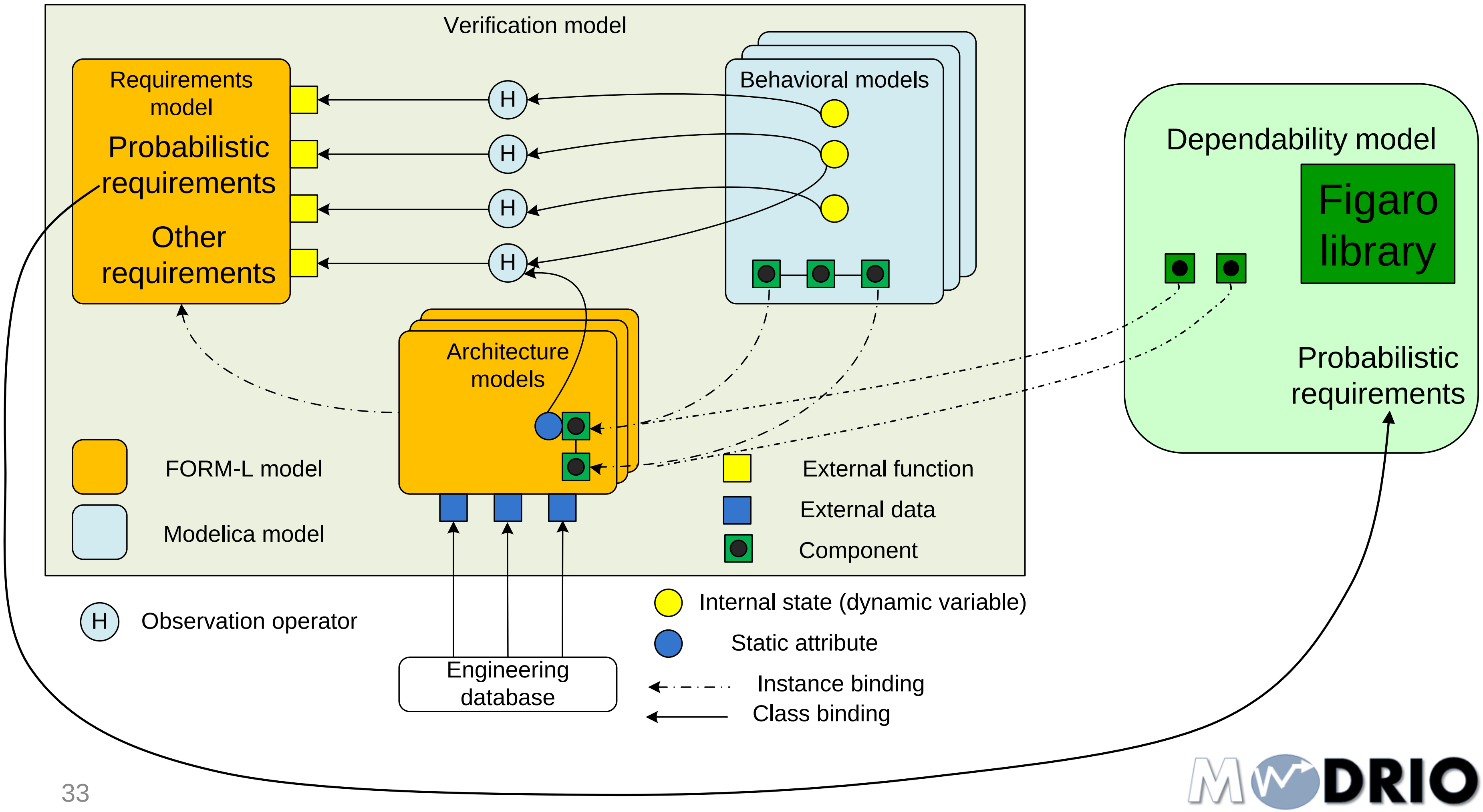


- Visual Figaro
- Modelica tool
- Figaro processor
- GRIF, Risk Spectrum…

Conclusion of Part 2

- Main advantage of this approach: relies on mature tools and formalisms
- Smart fault tree generator
  - Can use automatically generated macro components
  - Handles looped systems properly
- No need to change the Modelica model, nor the Figaro library
- The Figaro and Modelica parts are loosely coupled: no need to have a bijection between the elements of the two models
- It will be possible to propose an enhanced GUI to help the user input correct Figaro additions (with pull down menus…)

Perspective: binding the Figaro model to Form-L models



THANK YOU!

Where to find more information on tools:

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

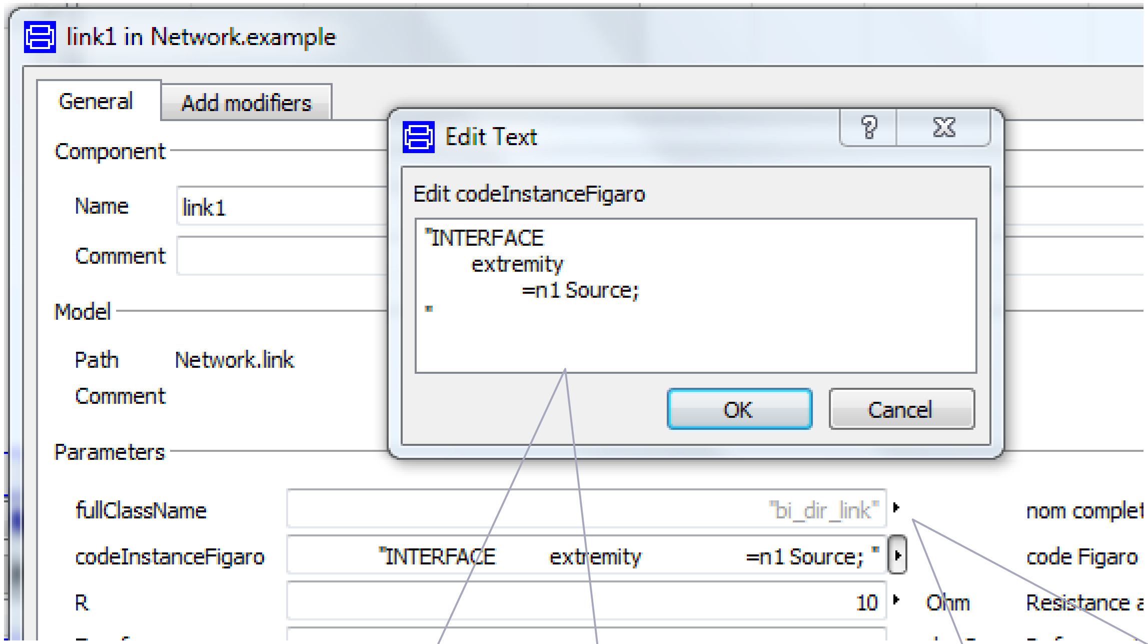
http://sourceforge.net/projects/visualfigaro/ download KB3 and other Figaro based tools

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

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

COMPLEMENTS

Telecom network: the FIGARO information that must be added in the Modelica model



Edit codeInstanceFigaro

```
"INTERFACE
     extremity
          =n1 Source;
"
```

Component

Name: link1

Comment:

Model

Path: Network.link

Comment:

Parameters

fullClassName: "bi_dir_link" — nom complet

codeInstanceFigaro: "INTERFACE   extremity   =n1 Source; " — code Figaro

R: 10 Ohm — Resistance a

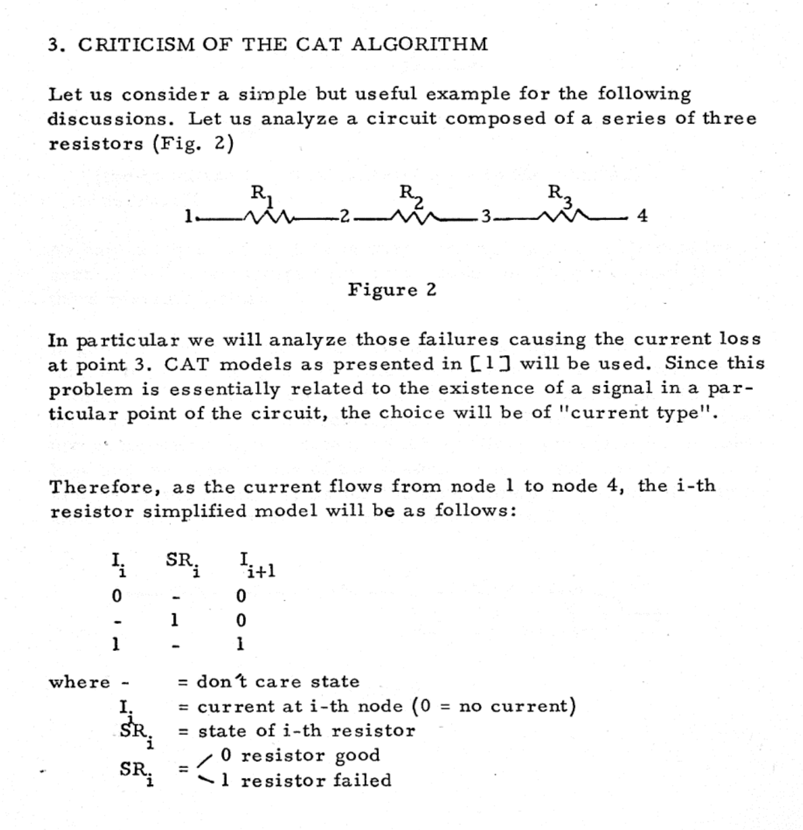
In the Modelica model, the class "link" is a subclass of:

- The electrical component resistor
- The generic class FIGARO that contains only two string parameters A and B

A: The only Figaro code the user has to input
(for every link)

B: The name of the corresponding class in the FIGARO library is defined in the class "link"

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)

R_1 R_2 R_3

1 2 3 4

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