Get Started

This starter kit is proposed by the IRT teams

The Get Started presentation is part of

IRT Saint Exupéry LIV-S085L01-001

IRT SystemX ISX-S2C-LIV-1285

Get Started

The « Get Started » purpose

Give you the keys to start modelling by yourself

The aim of this documents is to help you to start modelling on your own.

This presentation is not a user guide for a tool.

The example provided is done with Satodev Cecilia but the good practices are applicable for any other MBSA tool. In particular, the whole section “How to get started with MBSA” is applicable with Satodev Cecilia or with APSYS SimfiaNeo or any other tool using AltaRica language.

Table of contents

FRENCH INSTITUTES OF TECHNOLOGY

Table of contents

MBSA definition and studied perimeter

- MBSA stands for Model Based Safety Analysis

- It is a structural and behavioral organization to support abstraction of the system of interest regarding Safety assessment point of view.
- it models structure & behavior & failure injection into the system studied and allows:
  - quantitative assessment of associated failure conditions
  - qualitative assessment such as DAL assignation

- There are different kind of MBSA models and method such as Petri, Markov, …
- The current presentation focuses on AltaRica based model

FRENCH INSTITUTES OF TECHNOLOGY

AltaRica main principles

- AltaRica is a language used when doing MBSA

- AltaRica is a high-level formal language designed for the modelling of systems. A model describes a hierarchy of *nodes*; each component can embed several sub-nodes. These latter describe behaviors of components of the system.

- AltaRica permits many kind of expressions: mathematic, logic, boolean operators.

Table of contents

FRENCH INSTITUTES OF TECHNOLOGY

Questions to address before starting

- **What is the model for?**
  - Support a Trade Off ?
  - Capitalize data and information?
- **What are the analysis to be performed?**
  - Support classical analysis ?
  - FC quantification?
  - Qualitative analysis?
- **Which are the available information?**
  - Already existing artefacts or hypothesis at physical of functionnal level coming from other specialty domains?
  - What is the existing change process and change record versioning process?

- The answers will help to define:
  - **The information and the objects needed to build the model**
  - **The version and origin of information manipulated into the model to be manipulated in the model**
  - **The information to be observed: in general at safety level we assess Failure Conditions**
  - **The level and granularity to be achieved**
  - **Modelling approach and solutions choices**

FRENCH INSTITUTES OF TECHNOLOGY

Questions to address before starting

Related modelling activities before starting the model:

- **#MA1**: Define the perimeter of the system studied (its interfaces)
- **#MA2**: Raise a list of the main objects in the study perimeter: list of system components, list of failure conditions,…
- **#MA3**: Define the failure conditions in relationship with the model perimeter
- **#MA4**: Define the assumptions about propagation laws inside each modelling unit or node that results both:
  - From potential error and observable failure modes of all components/ functions
  - From safety functions performed in the nominal case

In order to satisfy its intended use, the model should:

- Fully cover the scope defined for the system studied
- Enable the observation and the analysis of a set of failure conditions

Go back to previous slide

Table of contents

FRENCH INSTITUTES OF TECHNOLOGY

The different steps to follow: details

- Definition of the Command / Monitoring (COM/MON) pattern example
- Detailed definition of the components
- Definition of the observer
- Step by step simulation

The COM/MON pattern example

- **Addressing the preliminary questions:**

- The proposed model will be used to compute the Failure Conditions CutSets and probabilities

- The observers (embedded tool bricks addressed in slide 23) will be the failure conditions

- The model will be close to the system representing F1, F2, Comp and Ct

- The granularity of the model needs to show the SFMEA failure modes and functional reconfigurations of the system

- The logics of the comparison needs to be written in a easy way to be validated by the system

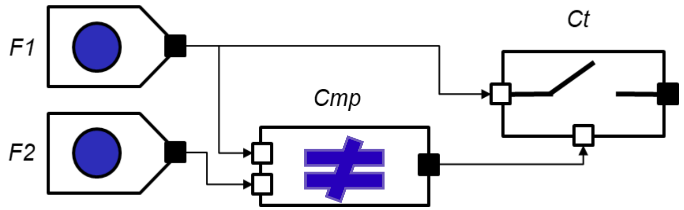
The COM/MON pattern example

System description

- Two input commands F1 and F2, one output command
- The system sends a consolidated command order F1
- The purpose of the system is to send a command order F1 consolidated from two input commands.
  - The system monitors the two orders F1 and F2.
  - When F1 and F2 are different, an opening command is sent to the Contactor, the Contactor opens and the command is lost.
  - When the Contactor does not receive the opening command, F1 is transmitted.

The analysis purpose is to assess two FCs:

- FC1: Erroneous output (CAT)
- FC2: Loss of output (MIN)



Command/monitoring description

The schematic represents the system architctecture that the SA has to assess
The system is composed of:

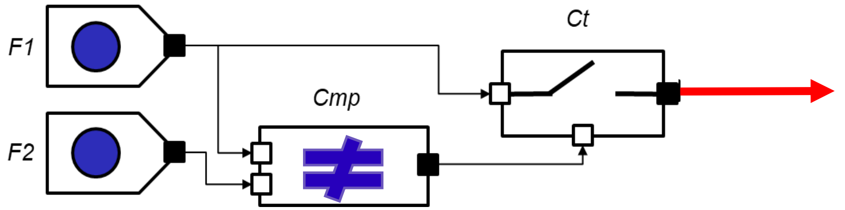
- Two Inputs or sources F1 and F2
- A comparator (Cmp)
- A contactor (Ct)

The COM/MON pattern example

The analysis purpose is to assess two FCs:

- FC1: Erroneous output (CAT) => the CMD at the system output is erroneous
- FC2: Loss of output (MIN) => the CMD at the system output is lost

System output flow variable
CMD
Domain of the flow
OK, ERR, LOST



System output is defined as per #MA1 and #MA2

Domain is defined as per #MA3

The elements (sources, comparator, contactor) are named as a component and not by their functions

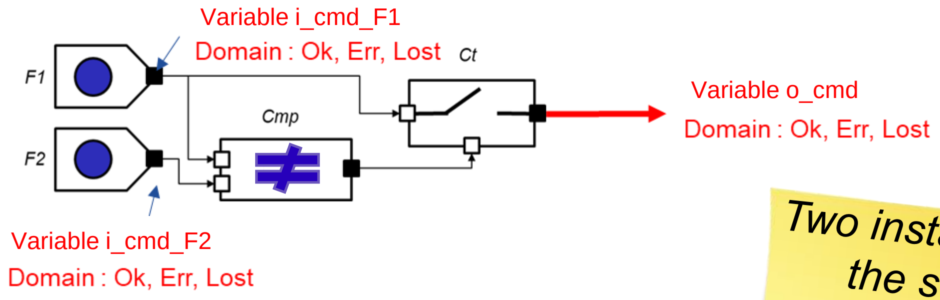
The different steps to follow: details

- Definition of the Command / Monitoring (COM/MON) example
- Local modelling: the modelling units
- Definition of the observer
- Step by step simulation

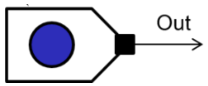
FRENCH INSTITUTES OF TECHNOLOGY

The source

Components F1 and F2 can be seen as two instances of the same component `Source`



The source



Two instances of the same modelling unit are used

The internal state of the Source depends on the failure modes

The domain of the State of the modelling unit/ physical component: State_OK, State_LOST, State_ERR

Existing failure modes

fail_LOSS
fail_ERR

In that case (no input) the output command only depends on the internal state of the component.

The associated transitions are:

State_OK |- fail_LOSS -> State_LOST
State_OK |- fail_ERR -> State_ERR

Even if it is possible to use same domain for State and Flow we advise to distinguish them to have more explicit names

The source

Internal State	o_cmd Type CMD Domain OK, ERR, LOST
State_OK	OK
State_LOST	LOST
State_ERR	ERR

trans
State=State_OK |- fail_LOSS -> State:= State_LOST;
State=State_OK |- fail_ERR -> State:= State_ERR;

assert
o_cmd = **case**{
(State=State_LOST): LOST,
(State=State_ERR): ERR,
else
OK
};

The assertion (*assert* in AltaRica) corresponds to the behavior logic . Here it means that when I am in state State_OK then when the trigger fail_LOSS appears, the output i_comd_F1 take the value Staet_LOST

The assumptions about propagation are defined as per #MA4 and #MA2

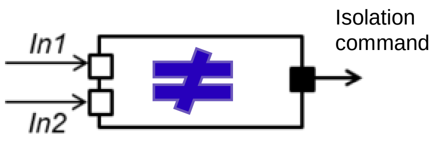
Domain is defined as per #MA3

FRENCH INSTITUTES OF TECHNOLOGY

The different steps to follow – Local modelling: the modelling units

The comparator

The comparator



Comparison between the two input values and isolation command when they are different
=> We choose to use a boolean output command

No HW failure modes

The inputs of the comparator are F1 and F2 => CMD (OK, ERR, LOST)

The output domain type is a boolean (true/false) to send or not the detection of the isolation cmd

true = isolation or opening command
false= « stay closed » command

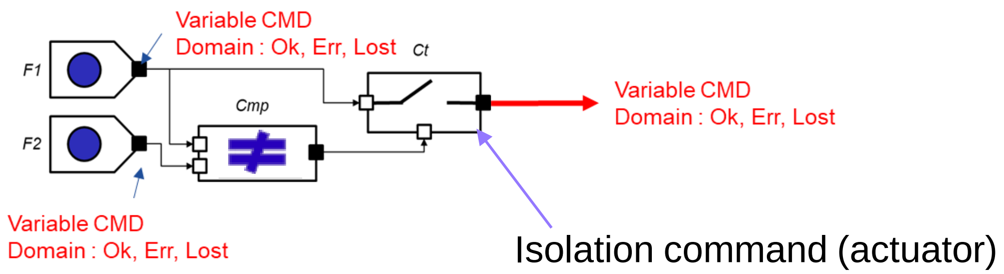
In this document we only consider comparison of exact variables – no variation between thresholds are considered

Definition of the local behavior

In1 Type CMD Domain OK, ERR, LOST	In2 Type CMD Domain OK, ERR, LOST	Out Type boolean Domain: true/false ⇒ Comparison ⇒ isolation
OK	OK	false
OK	LOST	true
OK	ERR	true
LOST	OK	true
LOST	LOST	false
LOST	ERR	true
ERR	OK	true
ERR	LOST	true
ERR	ERR	false

This step has to be done according to #MA4

The comparator: Isolation



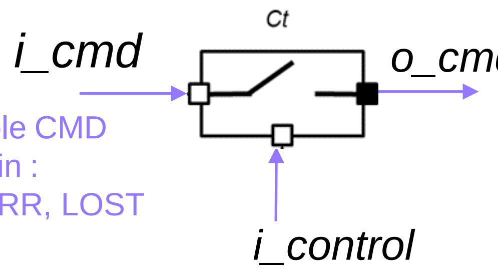
In1 Type CMD Domain OK, ERR, LOST	In2 Type CMD Domain OK, ERR, LOST	Out Type boolean Domain: true/false ⇒ Comparison ⇒ isolation
OK	OK	false
OK	LOST	true
OK	ERR	true
LOST	OK	true
LOST	LOST	false
LOST	ERR	true
ERR	OK	true
ERR	LOST	true
ERR	ERR	false

```
assert

// If Equal -> isolation

Out = case {
 (In1 = In2) :  false,
 else
 true
} ;
```

Comments :
After // for one line
Or between /* */ for several lines

This step has to be done according to #MA4

The contactor



The contactor is passive device that relies only on inptus . No power supply influences are considered (e.g.: too low voltage to close)

Modelling unit State Domain
(State_OK, State_Stuck_Open, State_Stuck_Closed)

Existing failure modes
fail_closed
fail_open

and associated transitions
State_OK |- failed_closed -> State_Stuck_Closed
State_OK |- failed_open -> State_Stuck_Open

Internal State OK, failed_closed failed_open failed_oscillatory	*i_cmd* Type CMD Domain OK, ERR, LOST	*i_control* Type isolation Domain true false	*o_cmd* Type CMD Domain OK, ERR, LOST
State_OK	OK	false	OK
State_OK	OK	true	LOST
State_OK	LOST	false	LOST
State_OK	LOST	true	LOST
State_OK	ERR	false	ERR
State_OK	ERR	true	LOST
State_Stuck_Open	ERR or LOST or OK	ERR or LOST or OK	LOST
State_Stuck_Closed	OK	true	OK
State_Stuck_Closed	OK	false	OK
State_Stuck_Closed	LOST	true	LOST
State_Stuck_Closed	LOST	false	LOST
State_Stuck_Closed	ERR	true	ERR
State_Stuck_Closed	ERR	false	ERR

fit FRENCH INSTITUTES OF TECHNOLOGY

The contactor

trans
(StateSwitch=OK) |-stuck_open ->
StateSwitch:=State_Stuck_Open;
(StateSwitch=OK) |-State_Stuck_Closed ->
StateSwitch:=State_Stuck_Closed;

assert
o_cmd = ***case*** {
(StateSwitch=State_Stuck_Closed): i_cmd,
(StateSwitch=State_Stuck_Open): LOST,
(i_control =true) and (StateSwitch= OK) : LOST,
else
true
};

The *case* allows to cover all possible « cases » to compute o_cmd

"As soon as one condition is reached, it is applied, the rest of the conditions are not evaluated" => priority

StateSwitch	*i_cmd* Type CMD Domain OK, ERR, LOST	*i_control* Type isolation Domain true false	*o_cmd* Type CMD Domain OK, ERR, LOST
State_OK	OK	false	OK
State_OK	OK	true	LOST
State_OK	LOST	false	LOST
State_OK	LOST	true	LOST
State_OK	ERR	false	ERR
State_OK	ERR	true	LOST
State_Stuck_Open	ERR or LOST or OK	ERR or LOST or OK	LOST
State_Stuck_Closed	OK	true	OK
State_Stuck_Closed	OK	false	OK
State_Stuck_Closed	LOST	true	LOST
State_Stuck_Closed	LOST	false	LOST
State_Stuck_Closed	ERR	true	ERR
State_Stuck_Closed	ERR	false	ERR

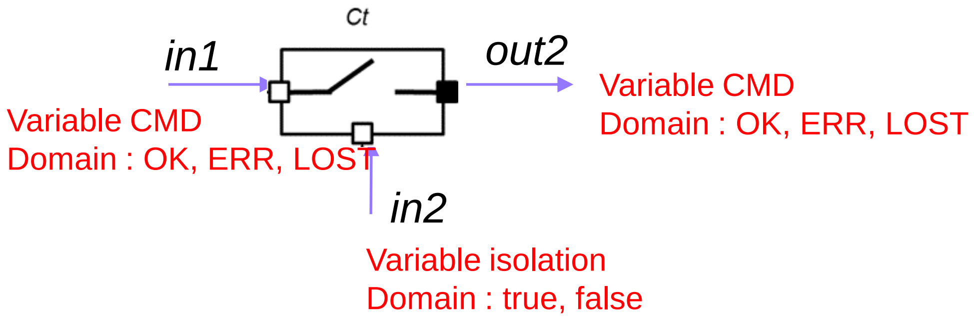
The contactor: oscillatory failure

- A failure can be oscillatory: what to do then?

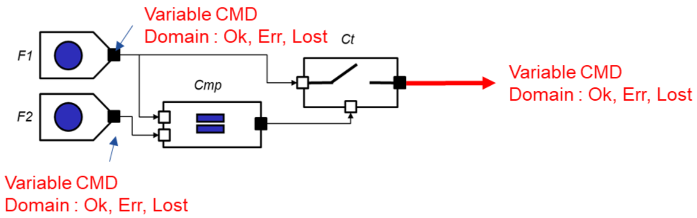
  - An oscillatory failure happens when the state is not constant. Then it is « oscillating » from true to false

  - The way to manage this kind of failure and the impact on the model is discussed on the next slide

The contactor: oscillatory failure



The aim of this slide is to introduce the impact of oscillatory failures on the model. A latch can be used to deal with this.



Modelling unit State Domain
(State_OK, State_LOST, State_ERR)

Existing failure modes
failed_closed
failed_open
failed_oscillatory

and associated transitions
- State_OK |- failed_closed -> State_LOST
- State_OK |- failed_open -> State_ERRoneous

State	In1 Type CMD Domain OK, ERR, LOST	In2 Type isolation Domain true false	Out Type CMD Domain OK, ERR, LOST
failed_oscillatory	OK	true	OK
failed_oscillatory	OK	false	LOST
failed_oscillatory	LOST	true	LOST
failed_oscillatory	LOST	false	LOST
failed_oscillatory	ERR	true	ERR
failed_oscillatory	ERR	false	LOST

The different steps to follow: details

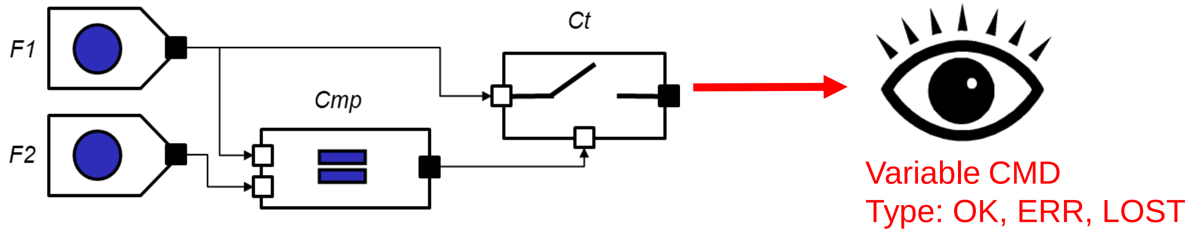
- Definition of the Command / Monitoring (COM/MON) example
- Detailed definition of the component
- Definition of the observer
- Step by step simulation

FRENCH INSTITUTES OF TECHNOLOGY

The different steps to follow – definition of the observers

Global modeling and observers

The observers are modeling artefacts used in the MBSA tools to calculate the Failure Conditions. It consists in a component that is observing the state of the variable associated to the failure condition. By calculating the probability of the different states of this observers, we can get the probability of the FC.



FC1: Erroneous output (CAT)
Observed value CMD=ERR

FC2 Loss of output (MIN)
Observed value CMD=LOST

The different steps to follow: details

- Definition of the Command / Monitoring (COM/MON) example
- Detailed definition of the component
- Definition of the observer
- Step by step simulation

FRENCH INSTITUTES OF TECHNOLOGY

Step by step simulation

- If you want to use the simulation in the best way (graphical options which can depend on the tool)
  - Define explicit icons for modeling units
  - Define colors for links
  - Examples:
    - OK: green, Erroneous: red, LOST: orange
    - Drift high: red, Drift low: blue
    - False: pink, True: blue green

Table of contents

FRENCH INSTITUTES OF TECHNOLOGY

Go through the tool

- You can open Cecilia using the Cecilia.bat
- Create a data base as described below:

clic right on the field "select database"

select Add database

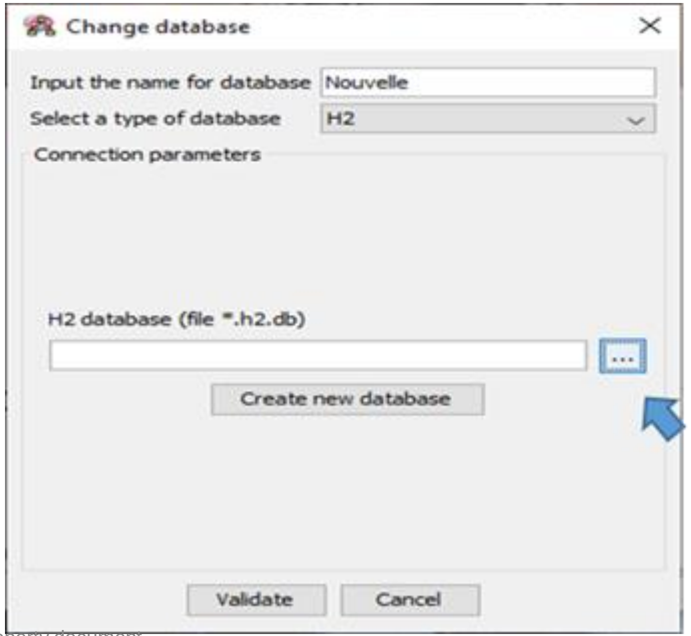
FRENCH INSTITUTES OF TECHNOLOGY

Table of contents

FRENCH INSTITUTES OF TECHNOLOGY

Go through the tool

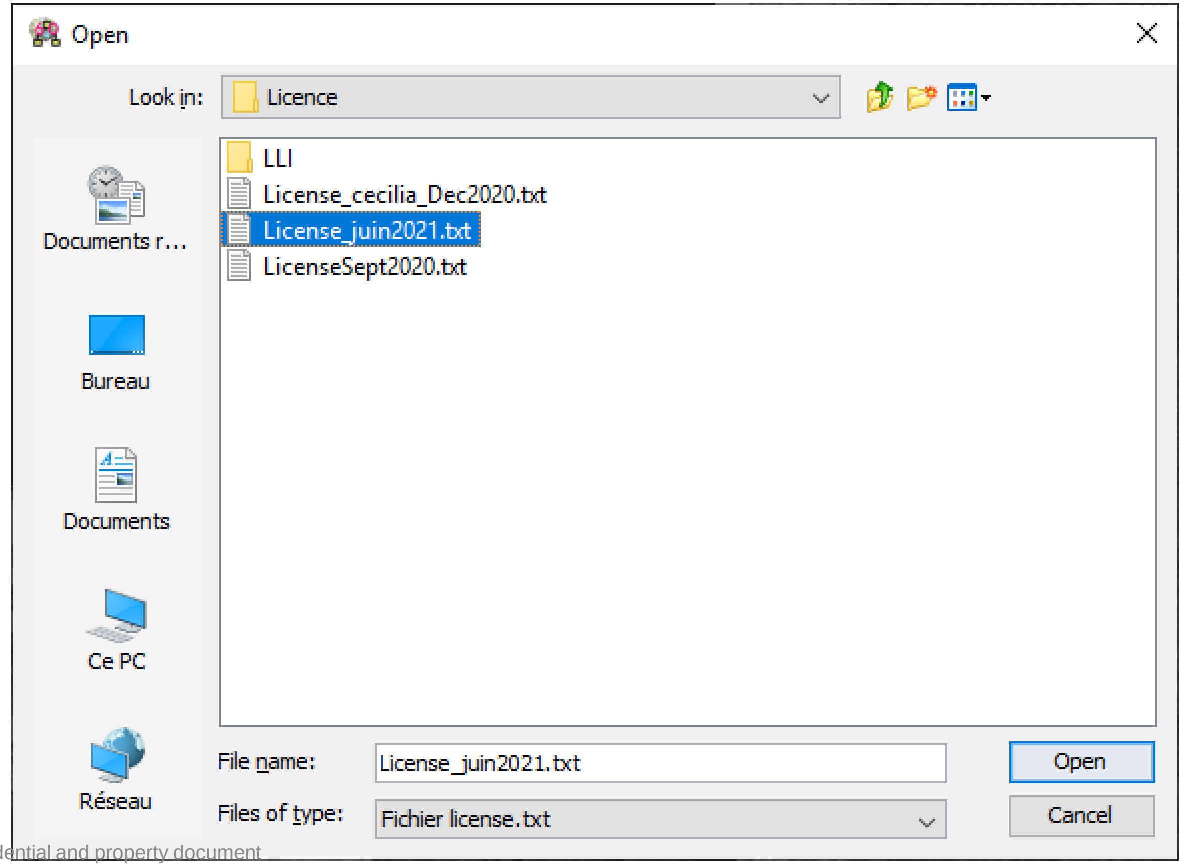
- Create a new data base => you select the location and name filling (the blank field in the image below or clicking on "Create new database")
- Write the name of your database in the field "Input the name for database"



FRENCH INSTITUTES OF TECHNOLOGY

Go through the tool

- The window Cecilia appears again with the name chosen for your data base. Select OK and you will be asked for the license " license is invalid". Enter the license path.

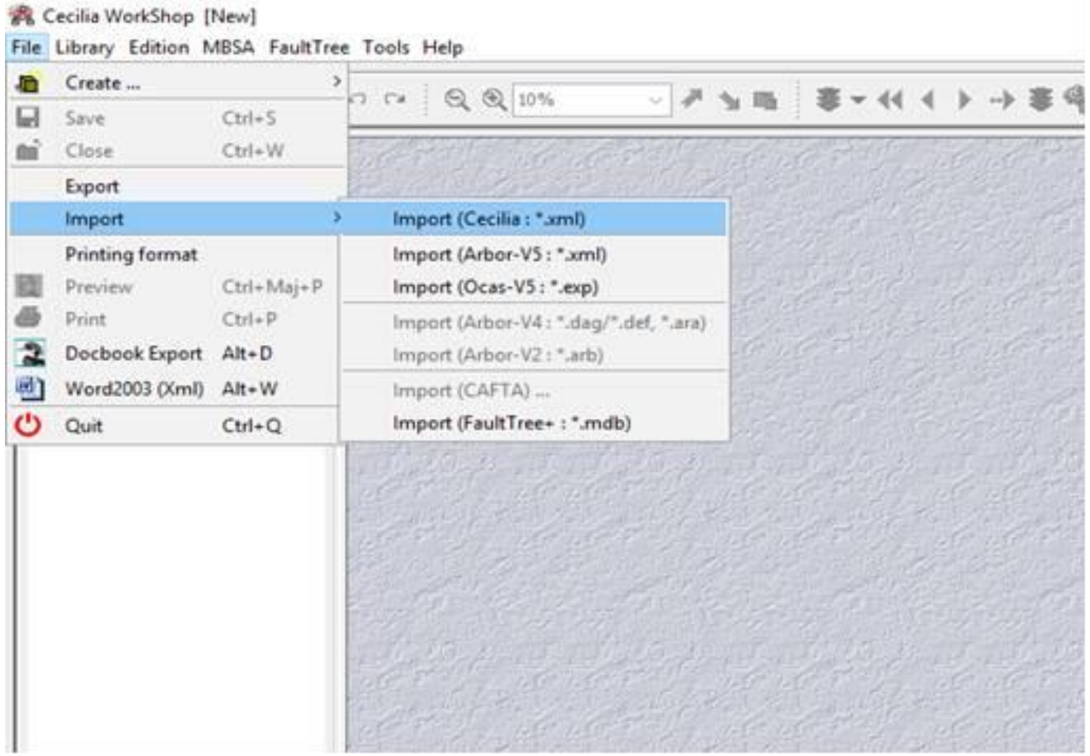


FRENCH INSTITUTES OF TECHNOLOGY

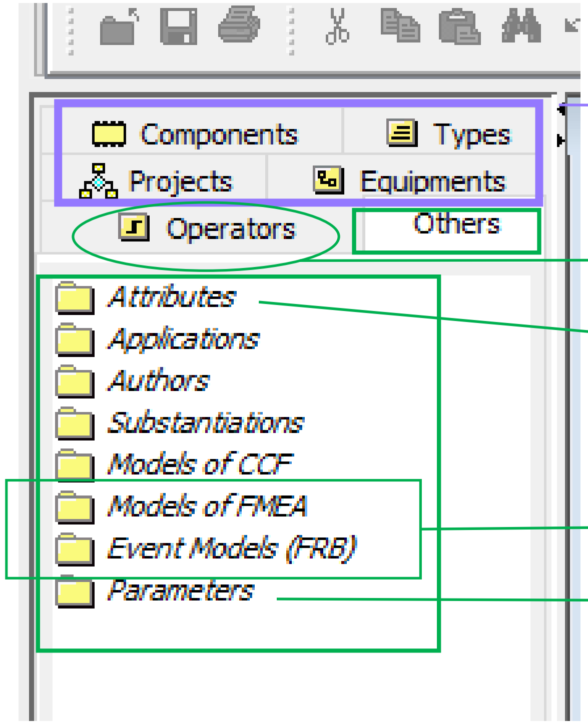
Go through the tool

- The default password is « admin »

- To import a file do: File> Create > Import> Import (Cecilia .xml) then select the » .xml » file to import.



Go through the tool: create and organize



Developped in the presentation

Definition of generic operators

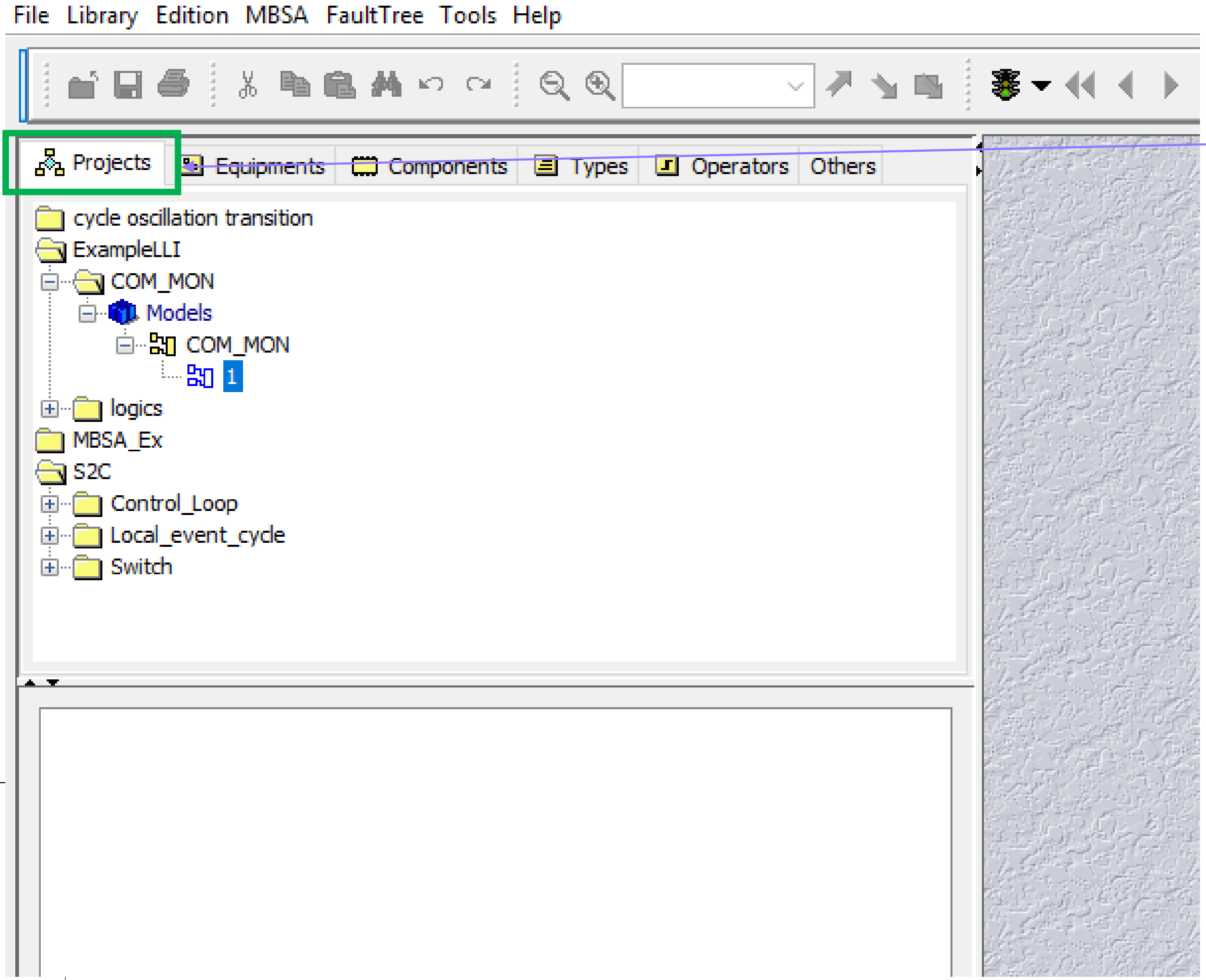
Possibly used in the event definition to support Attribute cutset computation

Possibly used in the event failure rate definition

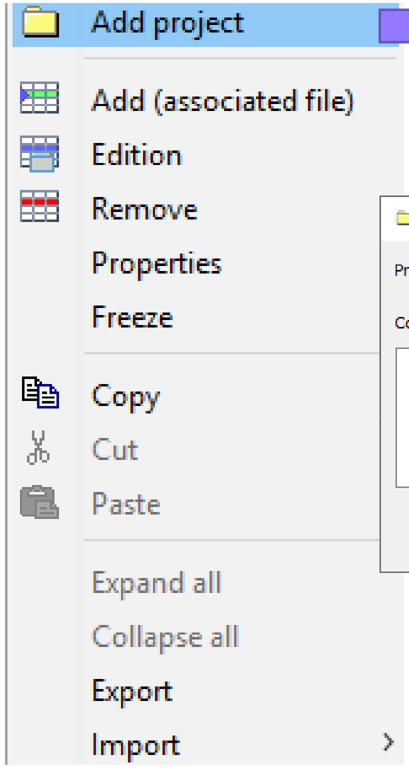
Definition of parameters such as Checks exposure time

The next three slides are dealing with the project creation and the organization of its content

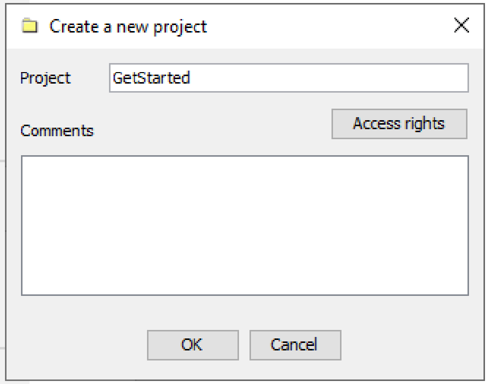
Go through the tool : create and organize



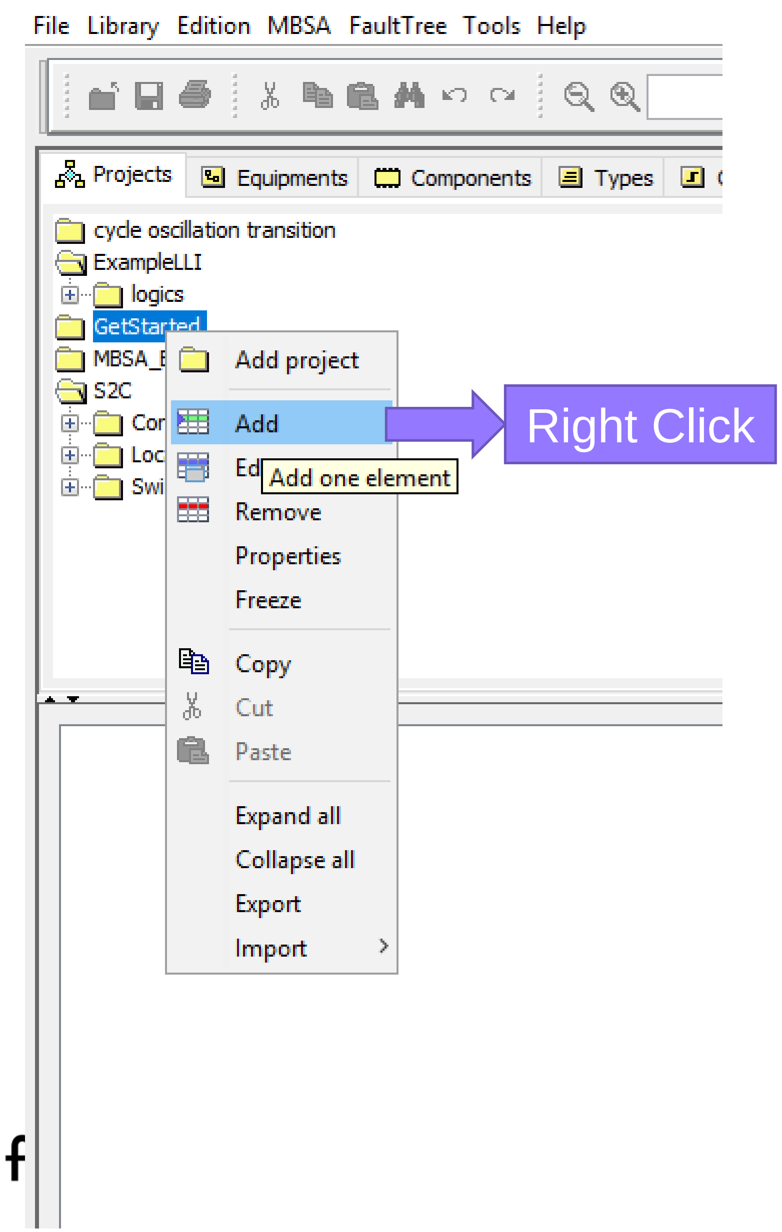
Project window: Click right

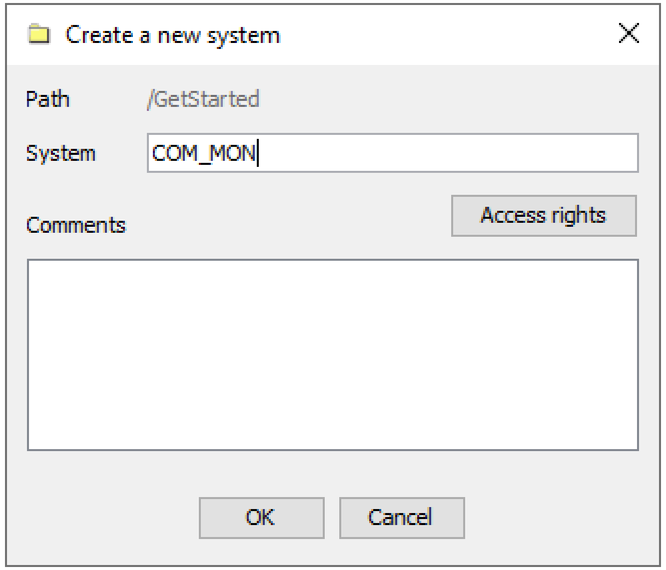


Click right

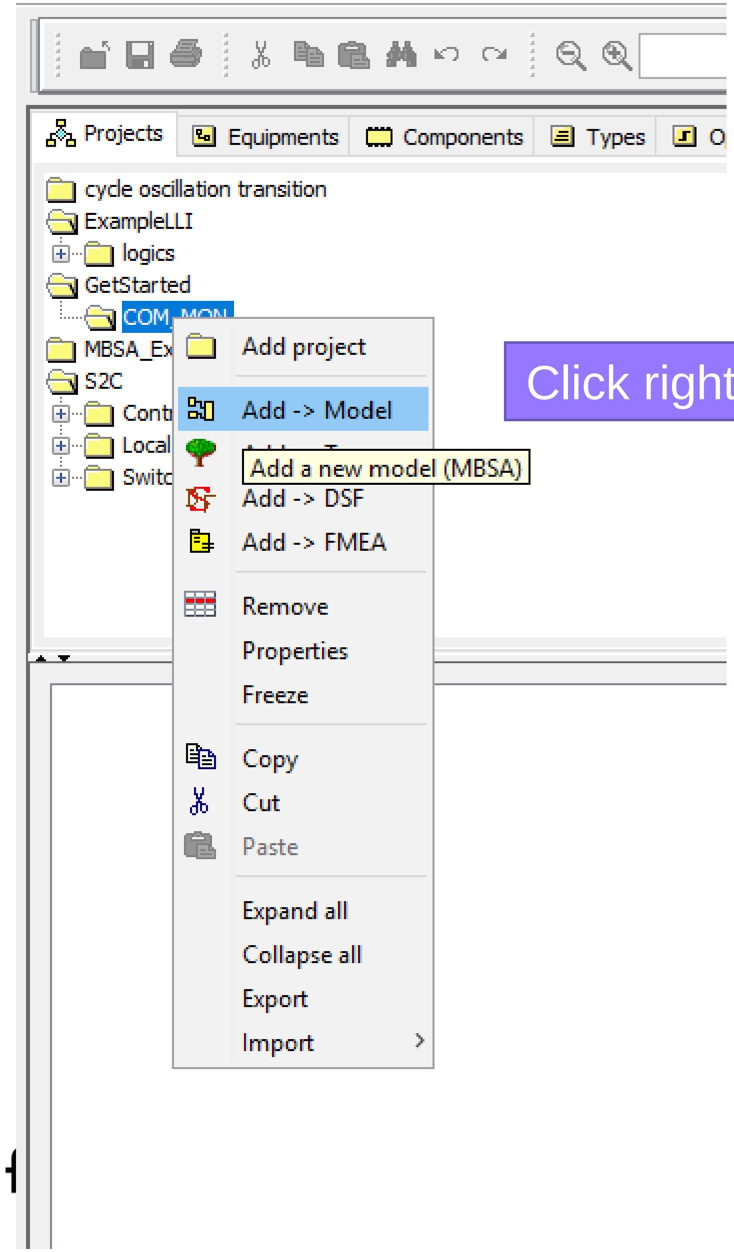


Go through the tool : create and organize

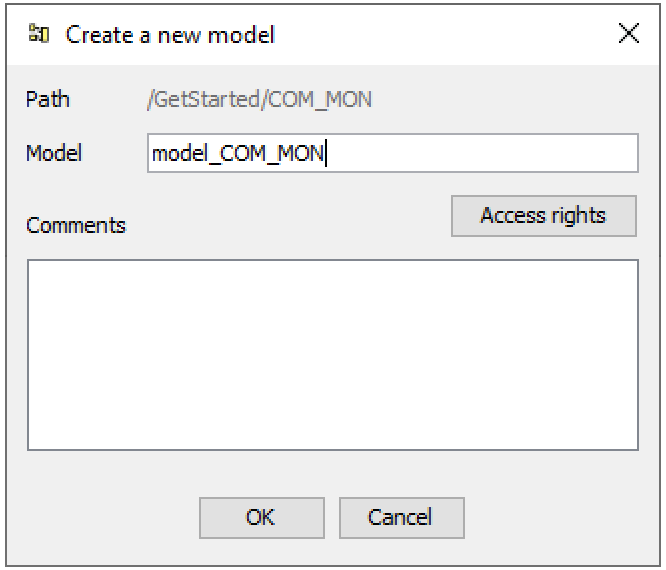




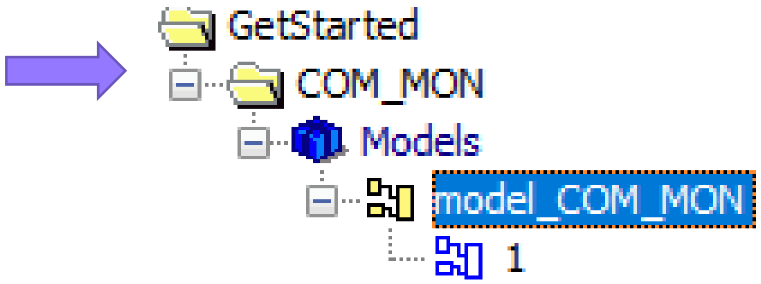
Go through the tool : create and organize



Click right



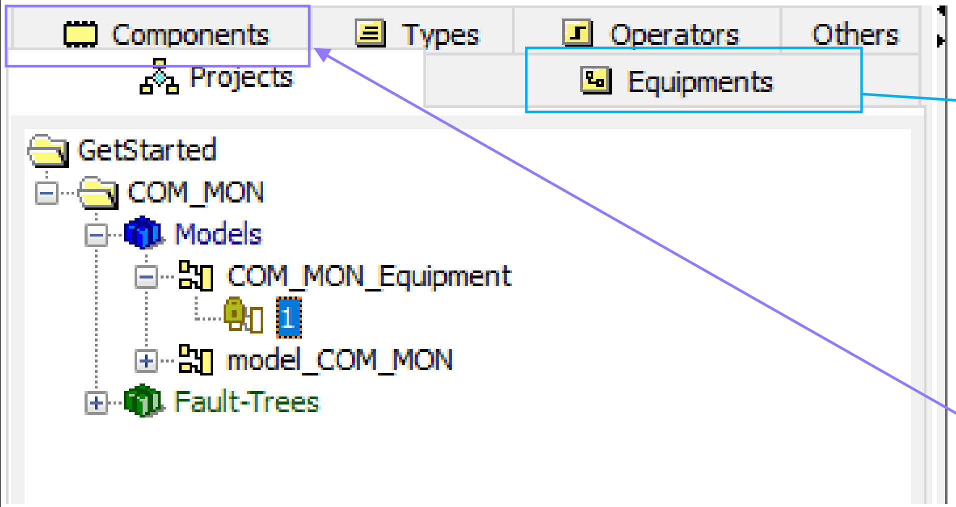
The new model appears in the browser

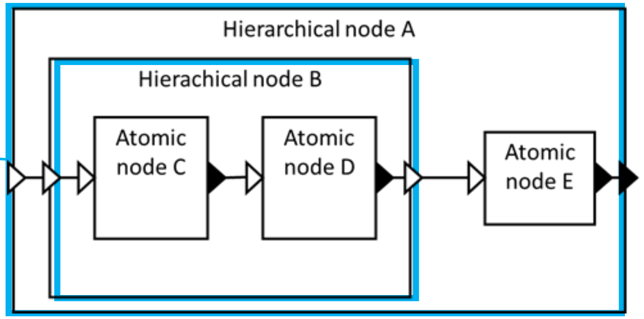


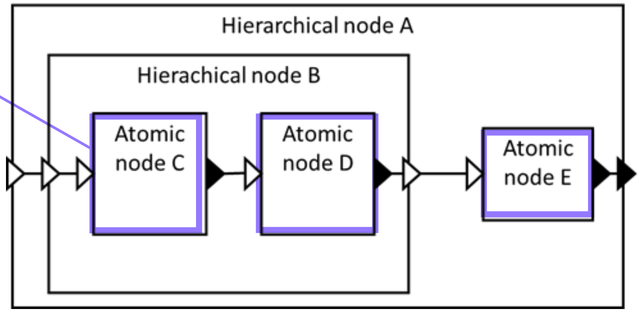
Double Click
To open it

Issue 1 of your model

Go through the tool : Component and Equipment







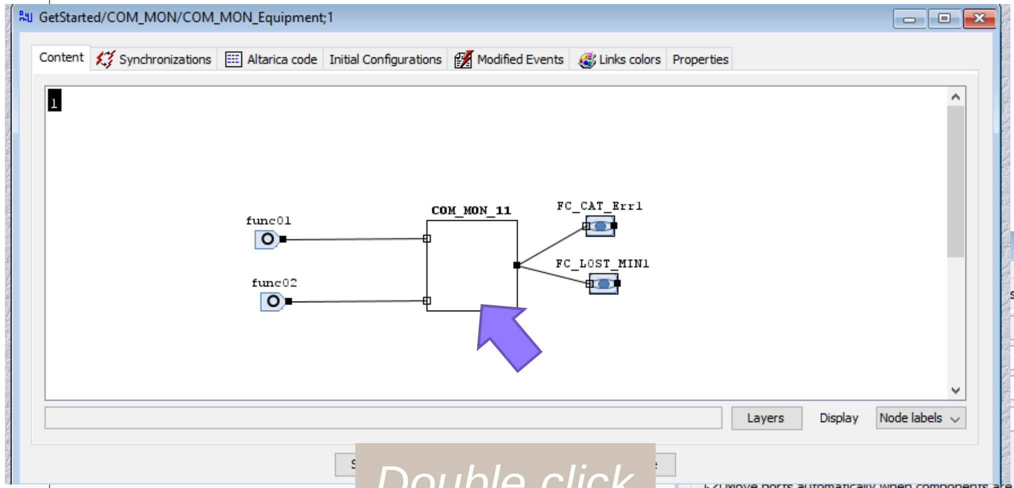
Hierarchy simplifies the reading of models

This activity has to be done according to #MA2

FRENCH INSTITUTES OF TECHNOLOGY

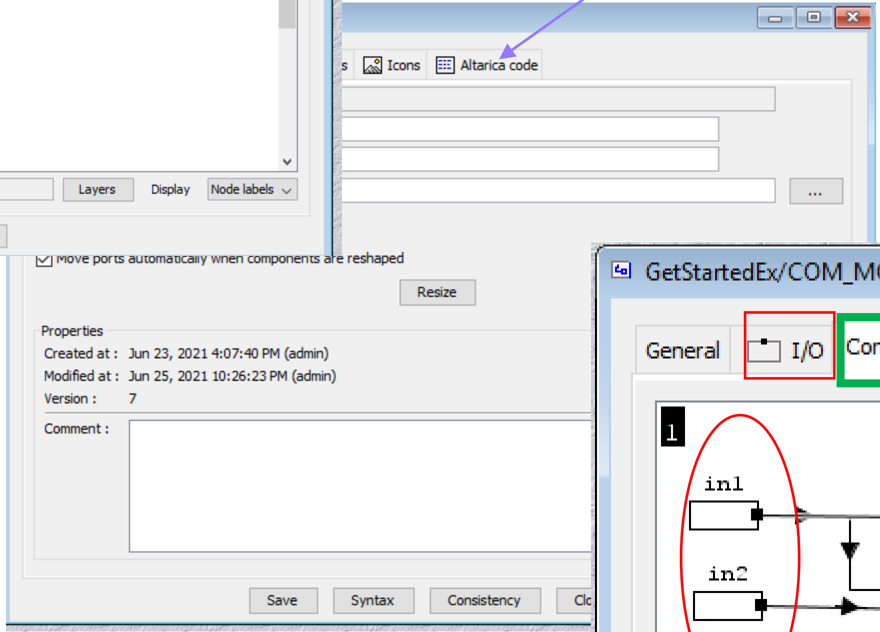
A quick look at equipment definition

To refer to variables of the components in the AltaRica code : Component_name.Variable



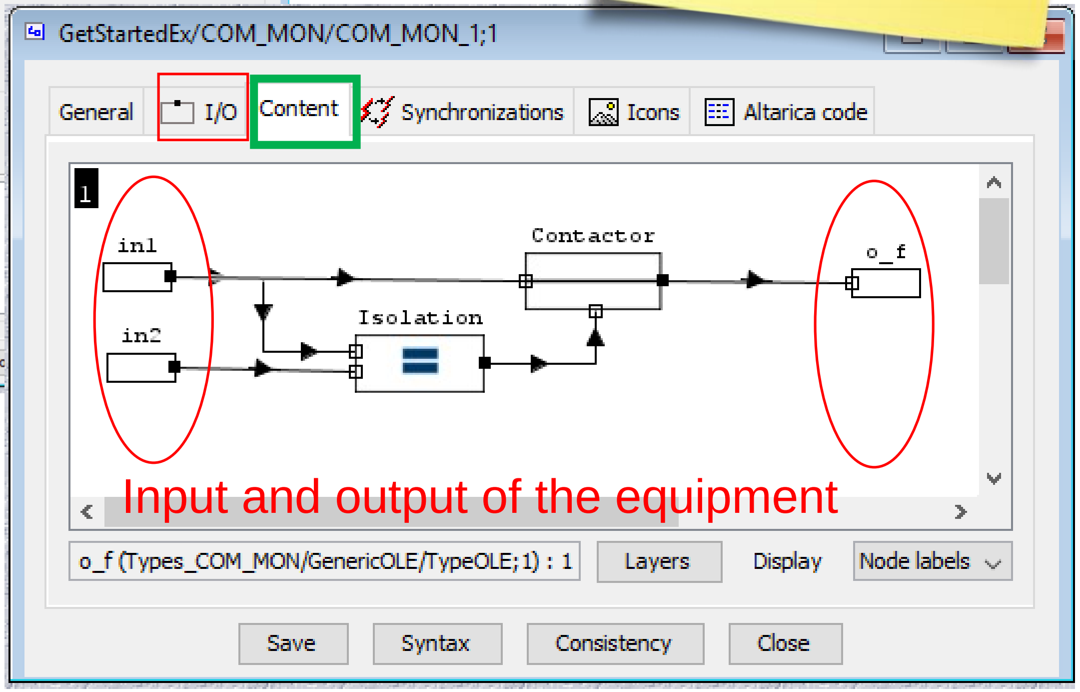
Double click

The model



The equipment definition

Use ^[space] to see available data

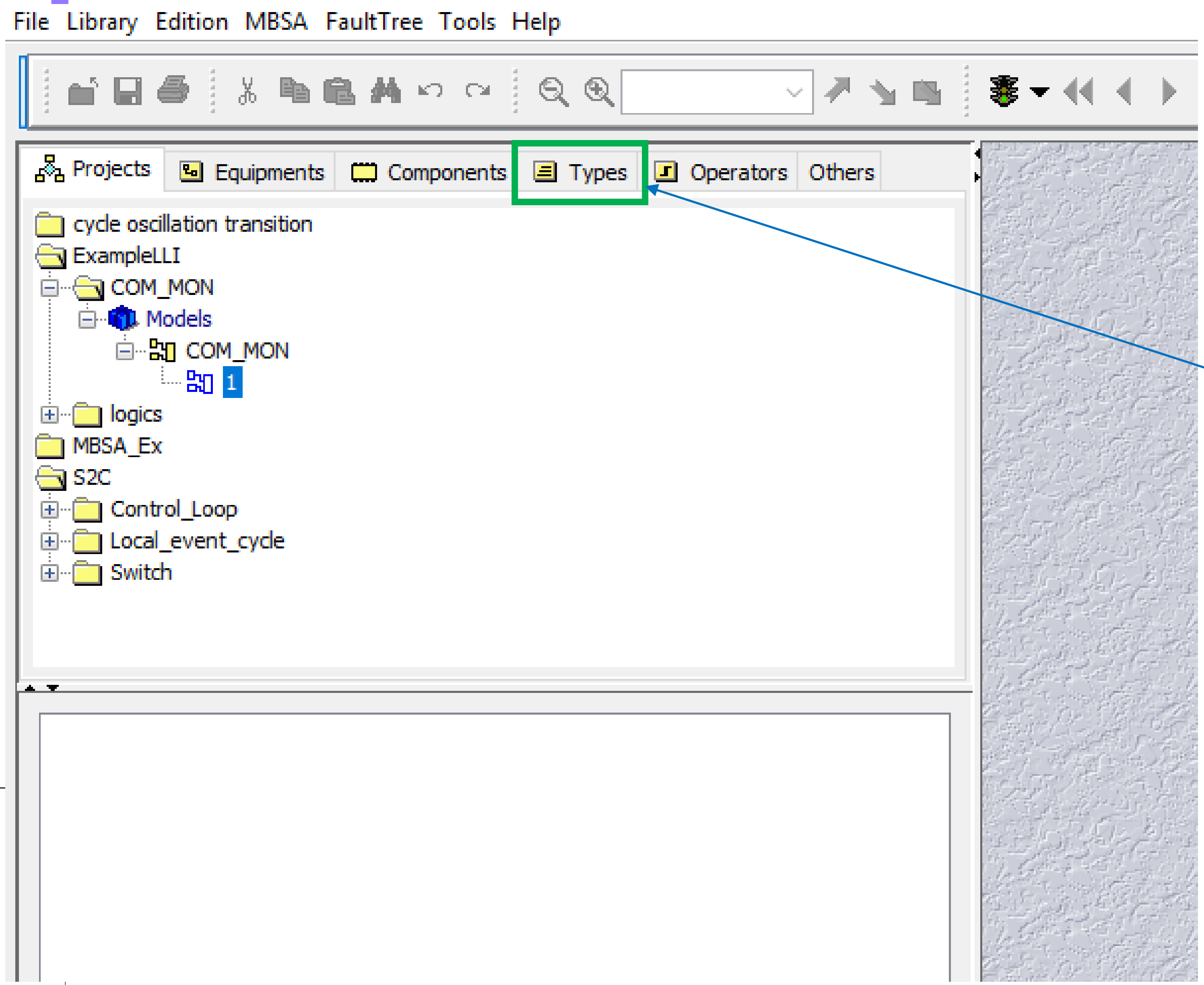


Equipment can be composed of equipment and modeling units called component in Cecilia

In the following we focus on the modeling units creation

FRENCH INSTITUTES OF TECHNOLOGY

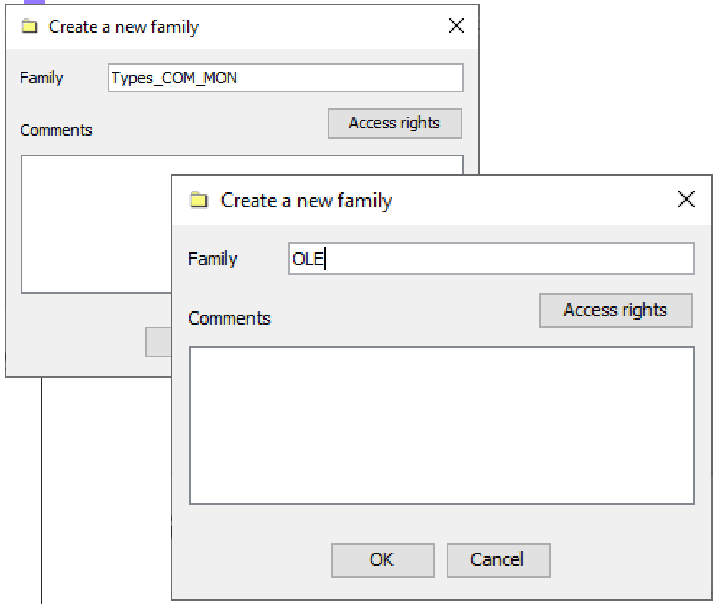
Types - Definition of the State Domain

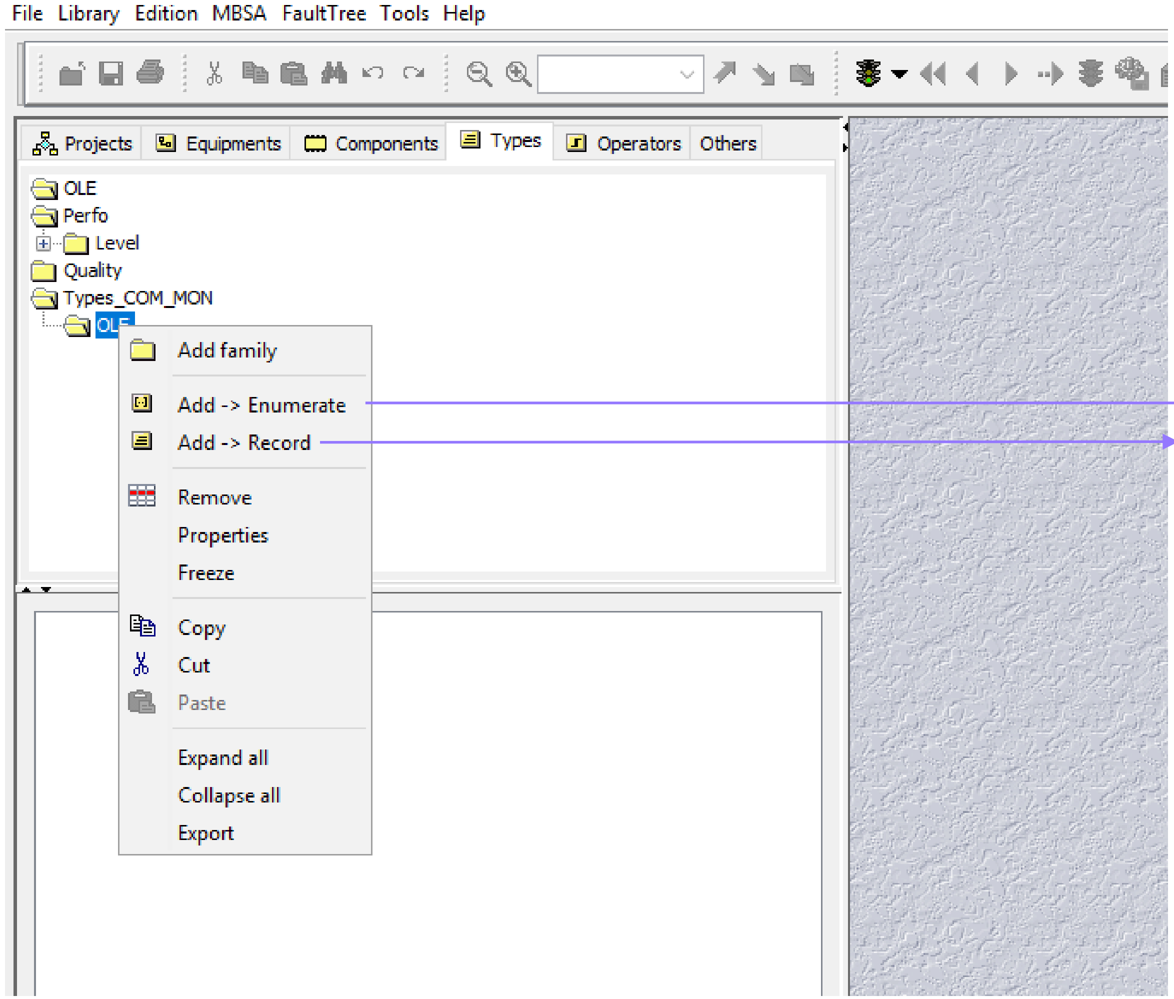


Same principle of creation than for the models

Types: define the domains, of the State and of the flow variable

Types - Definition of the State Domain





Enumerates

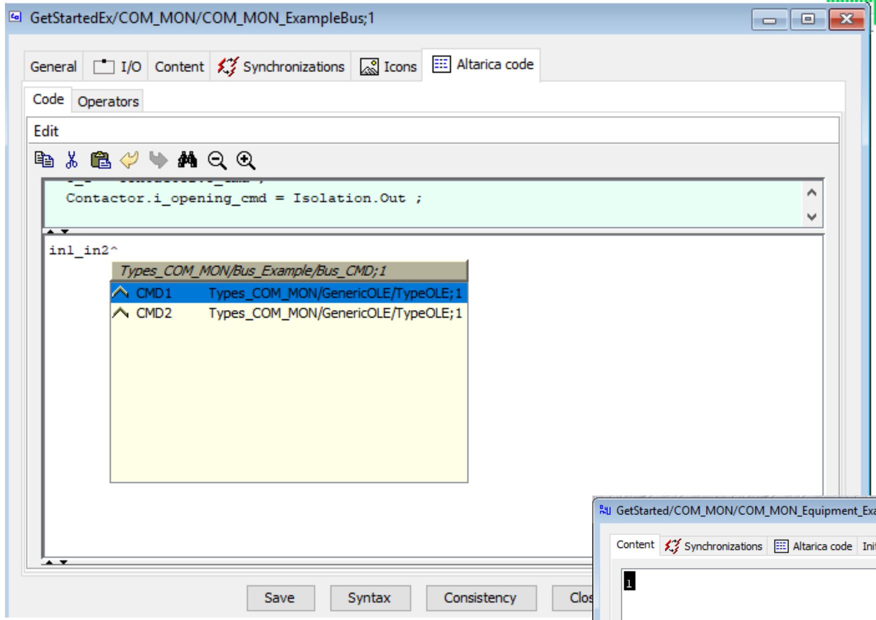
Bus
Composed type

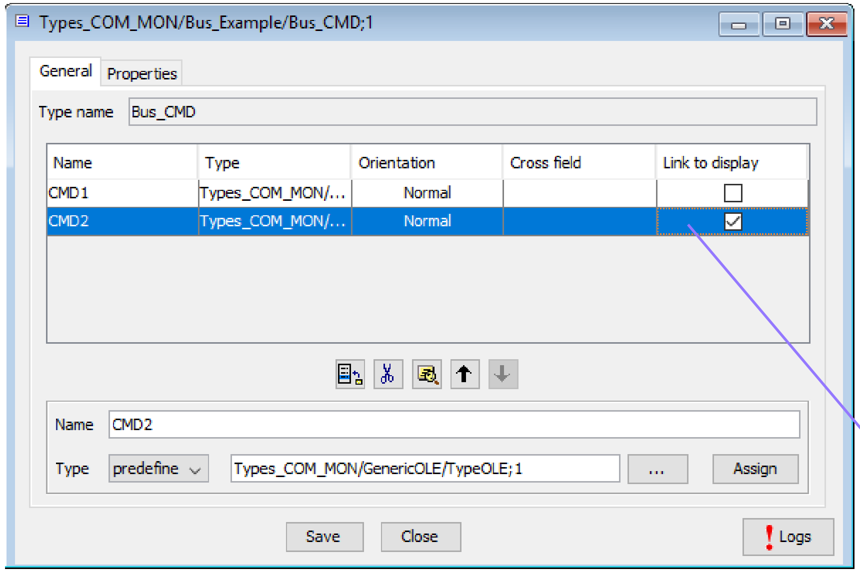
Record(Bus) are interesting features for complex systems : One bus can contain several flows
Bus.Flow

FRENCH INSTITUTES OF TECHNOLOGY

Types - Definition of the State Domain
The records

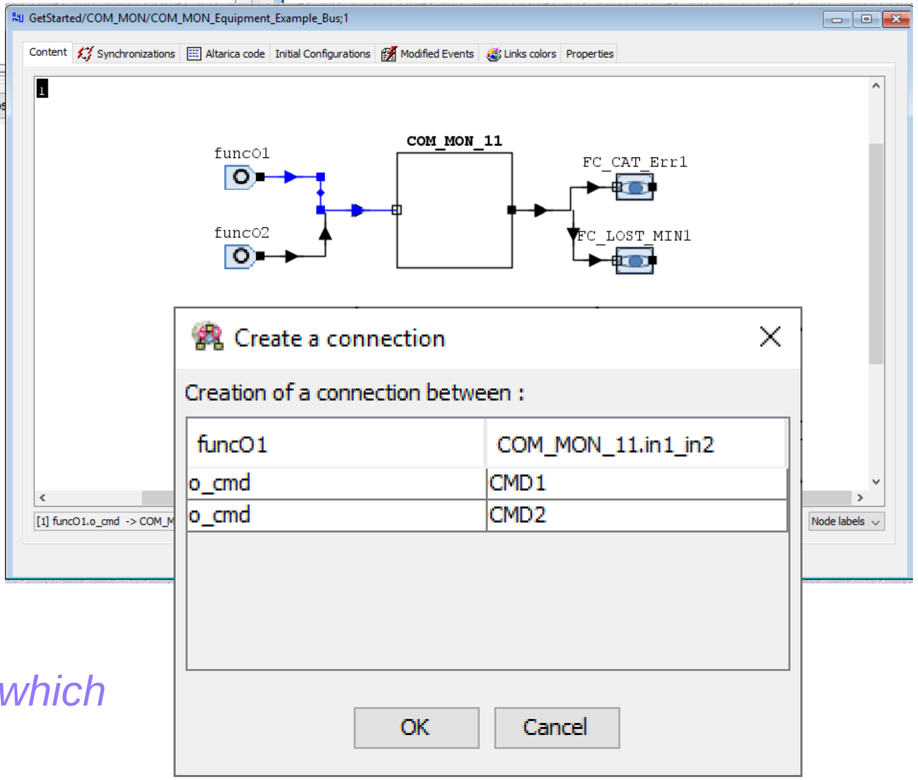
Use of record in AltaRica code « ^ »





Example of Record creation

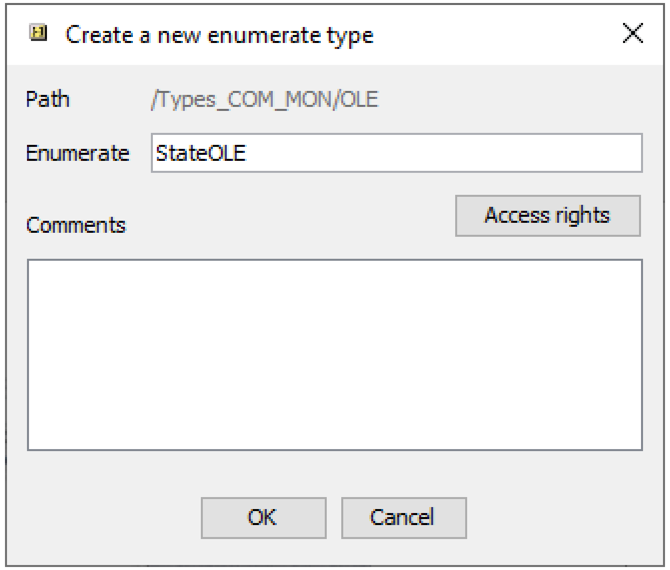
For the simulation choice of colour displayed

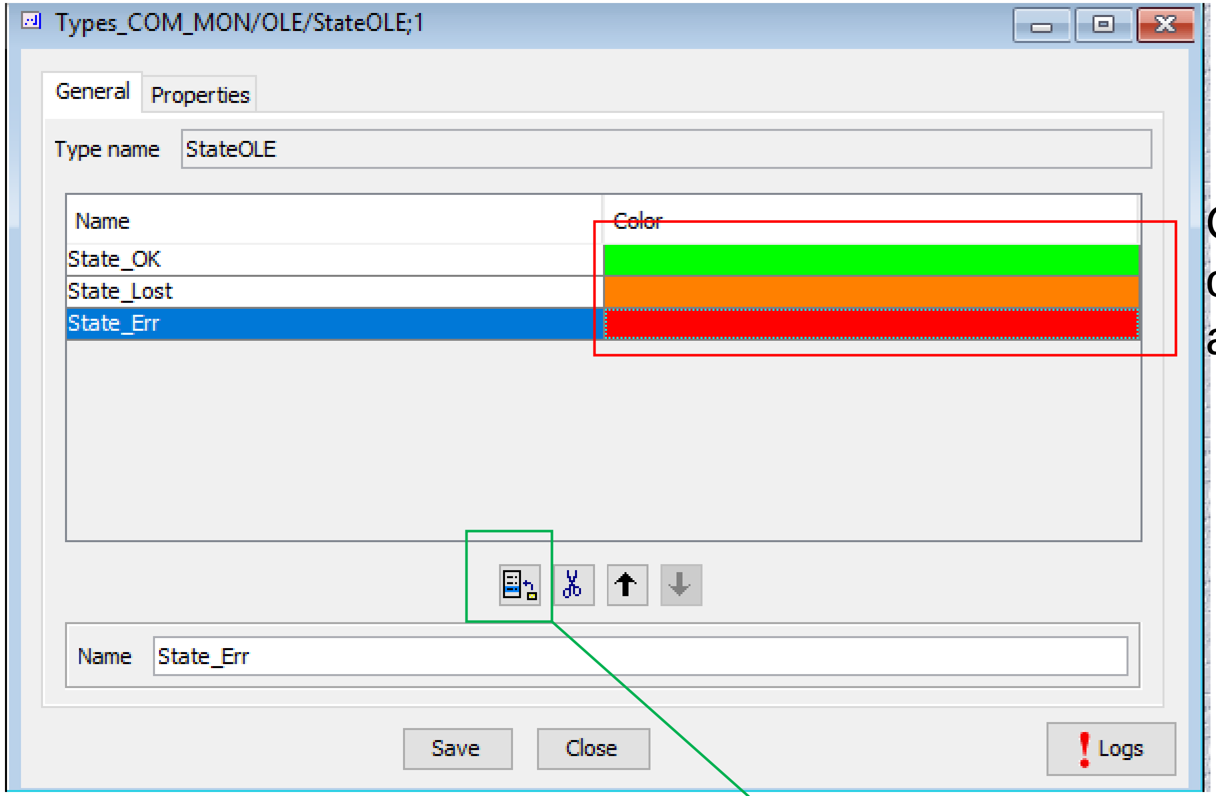


At the connexion you choose which data to connect

Types - Definition of the State Domain

(State_OK, State_LOST, State_ERR)



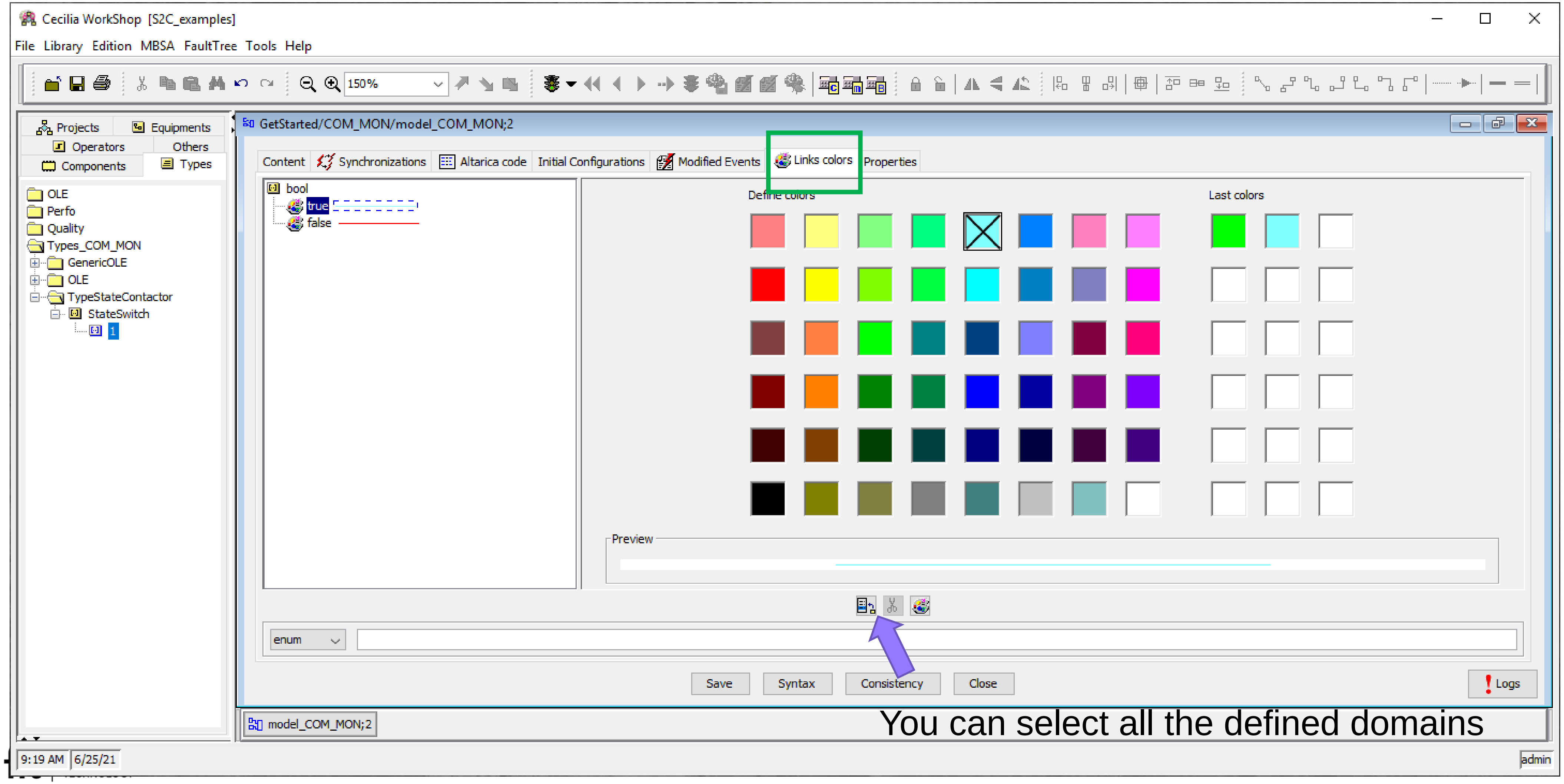


Choose the color of the flow when applicable

Add an enumerate

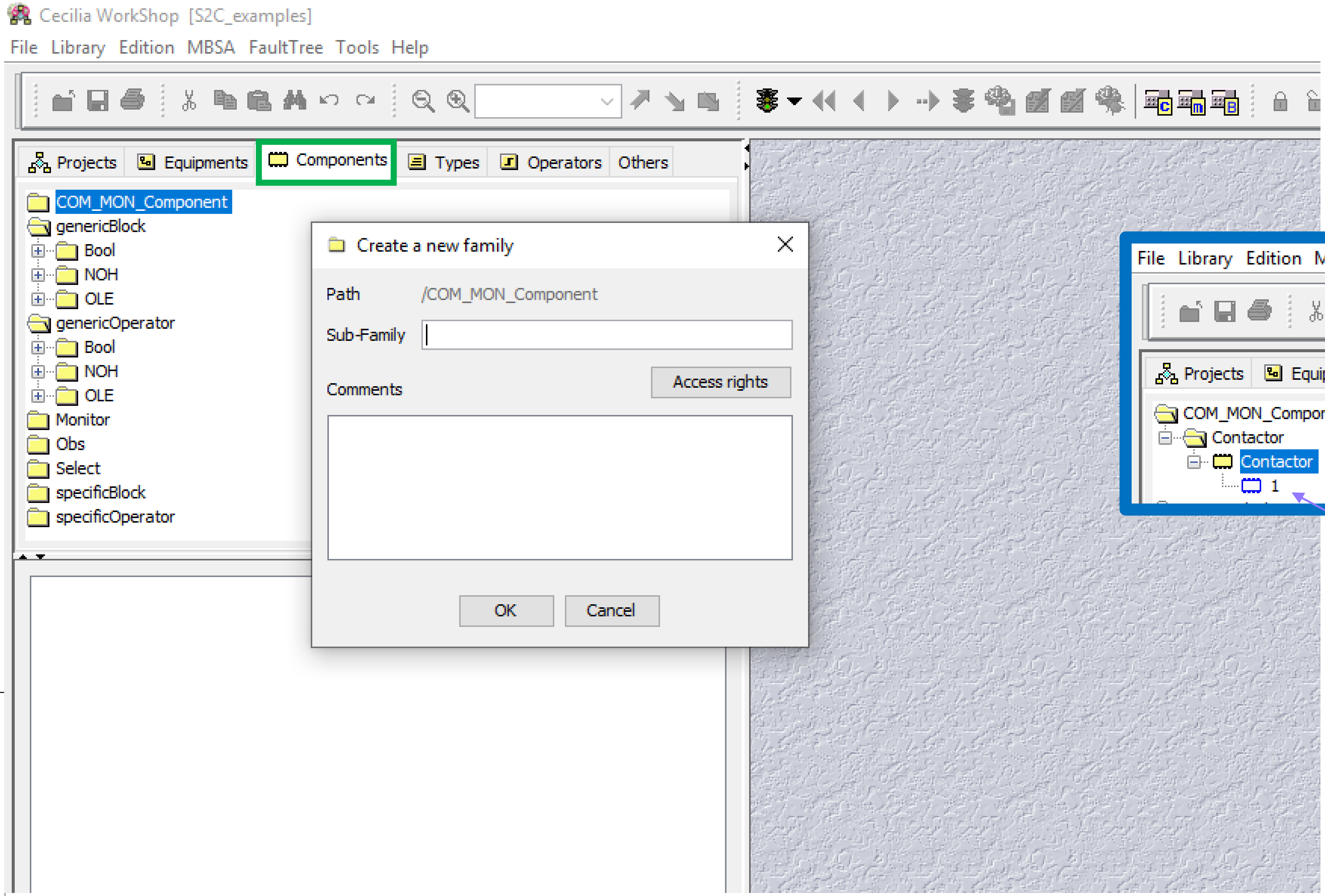
Note boolean (true/false) type are included in the language so do not need to be created (but possible)

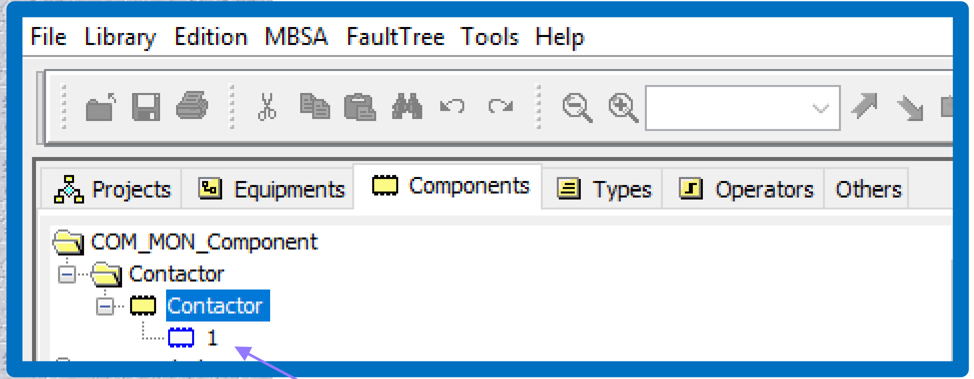
Access to the colors for a given model



You can select all the defined domains

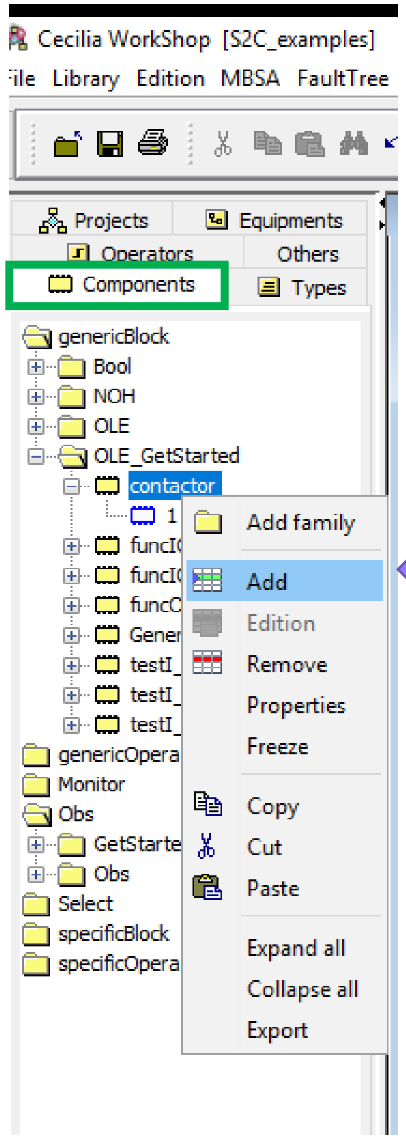
Modeling unit creation: the component
The contactor





To edit
Double click
Or right click
and choose
Edition

Go through the tool: components creation
The contactor

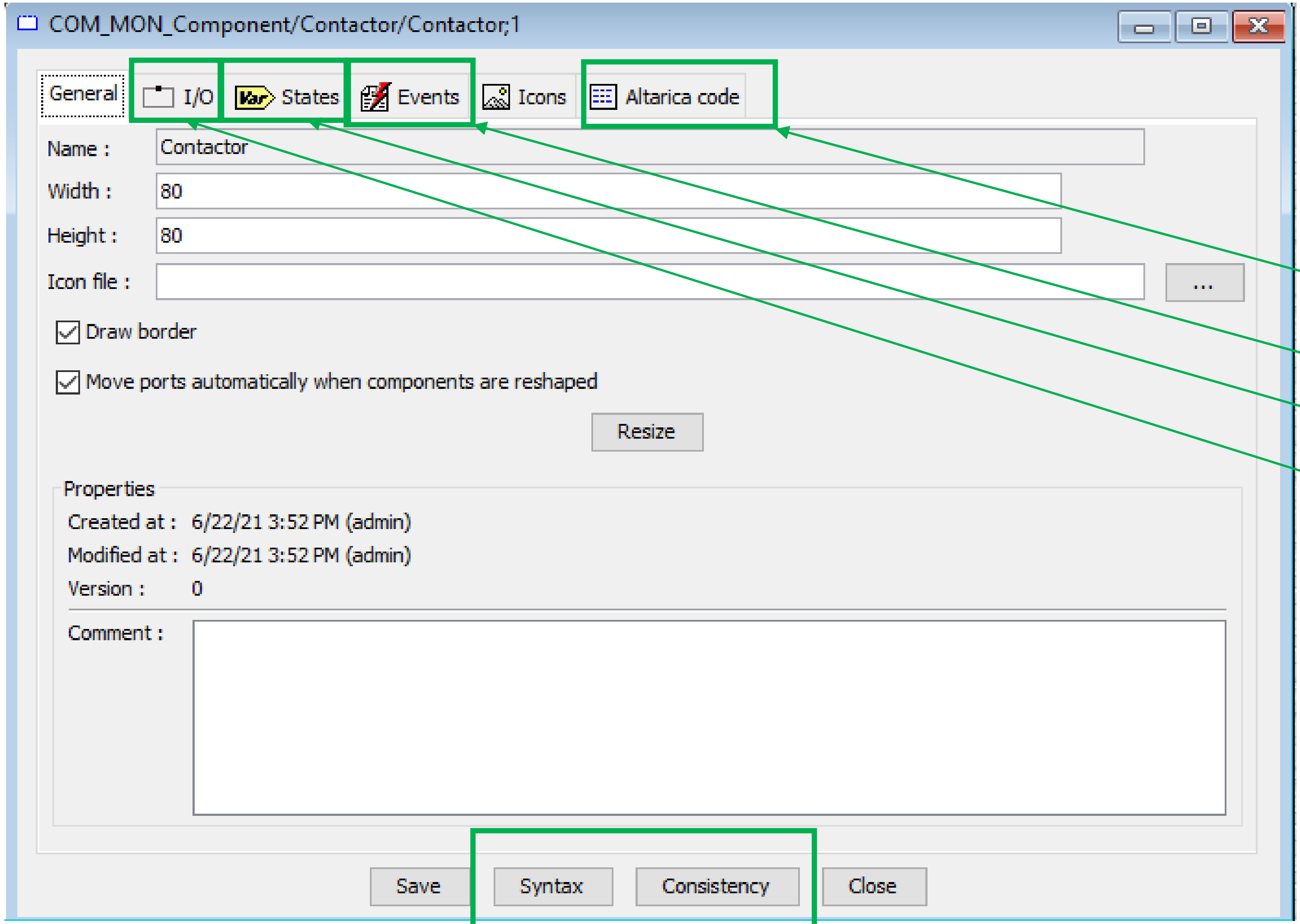


Add a new issue

- Possibility to copy paste and modify
- Create a new issue
- Replace

Modeling unit creation

Creation of a component: the Contactor



Definition of the transition and of the assertions

Definition of the events

Definition of the internal States

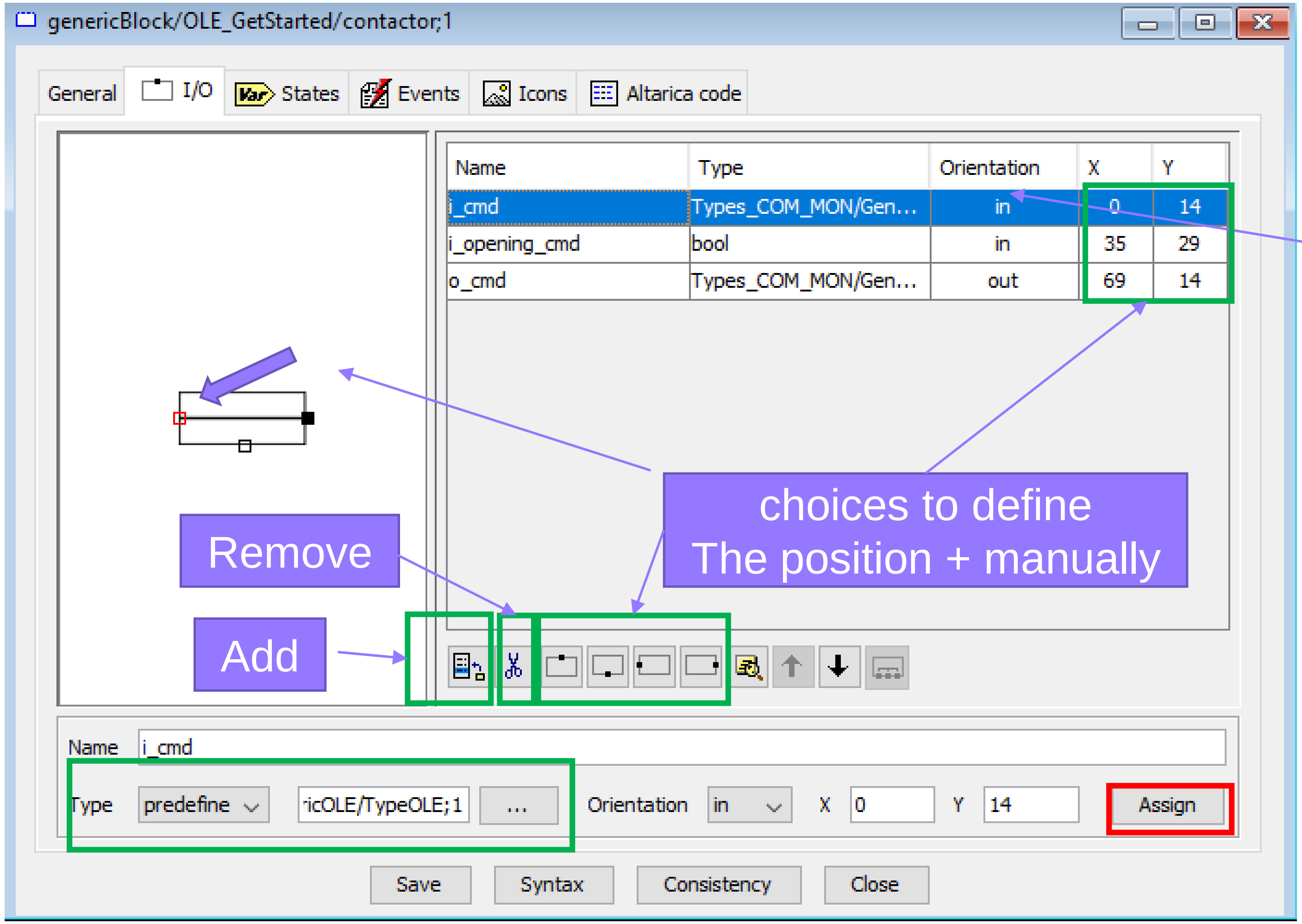
Definition of the I/O

! Tip! You can copy/paste any element and modify it

The checks

Modeling Unit

I/O



Flow orientation

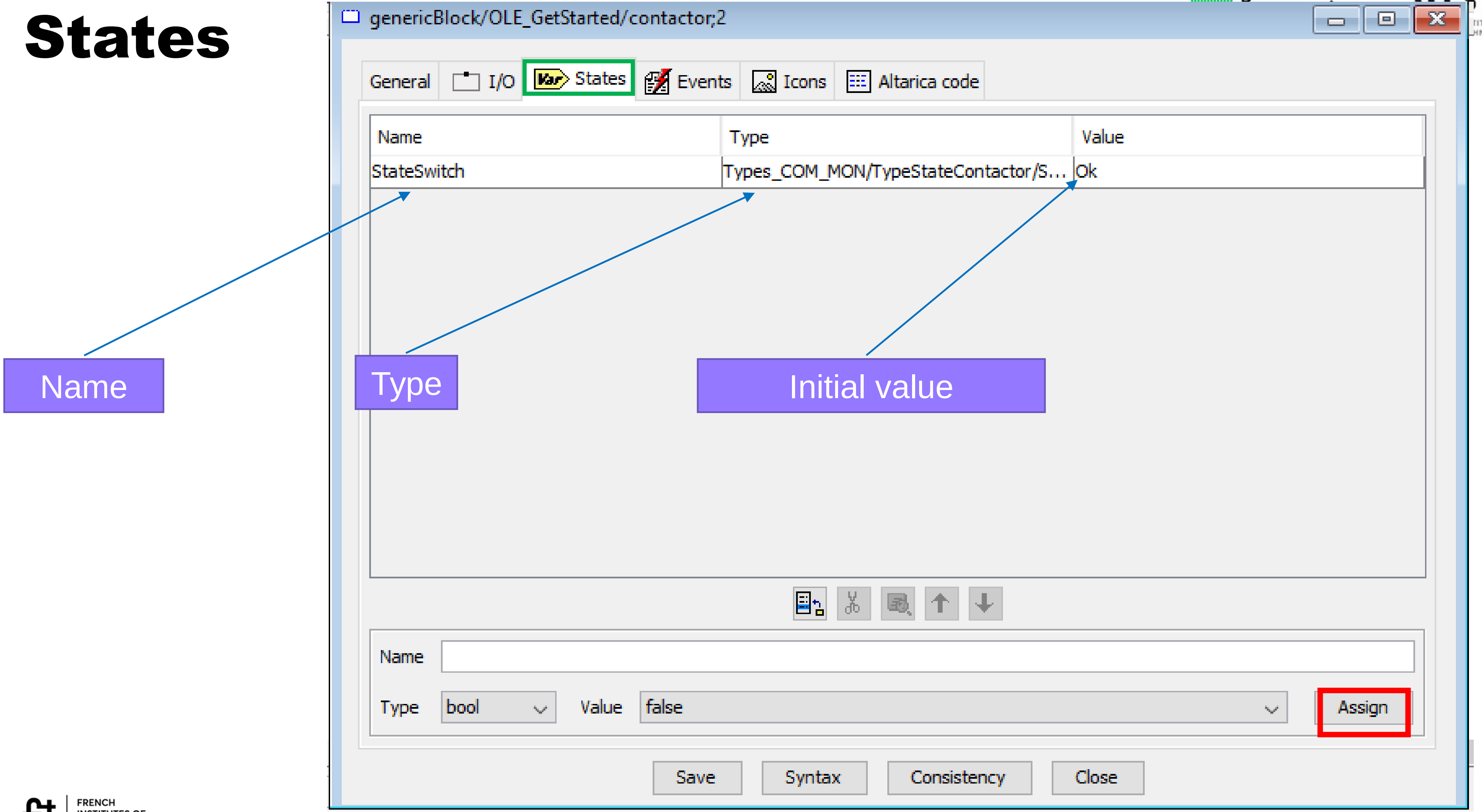
Only compatible connexion will be allowed

Local variable are allowed « local »

! When you modify an existing value: **Assign** to take the modification into account

Modeling Unit

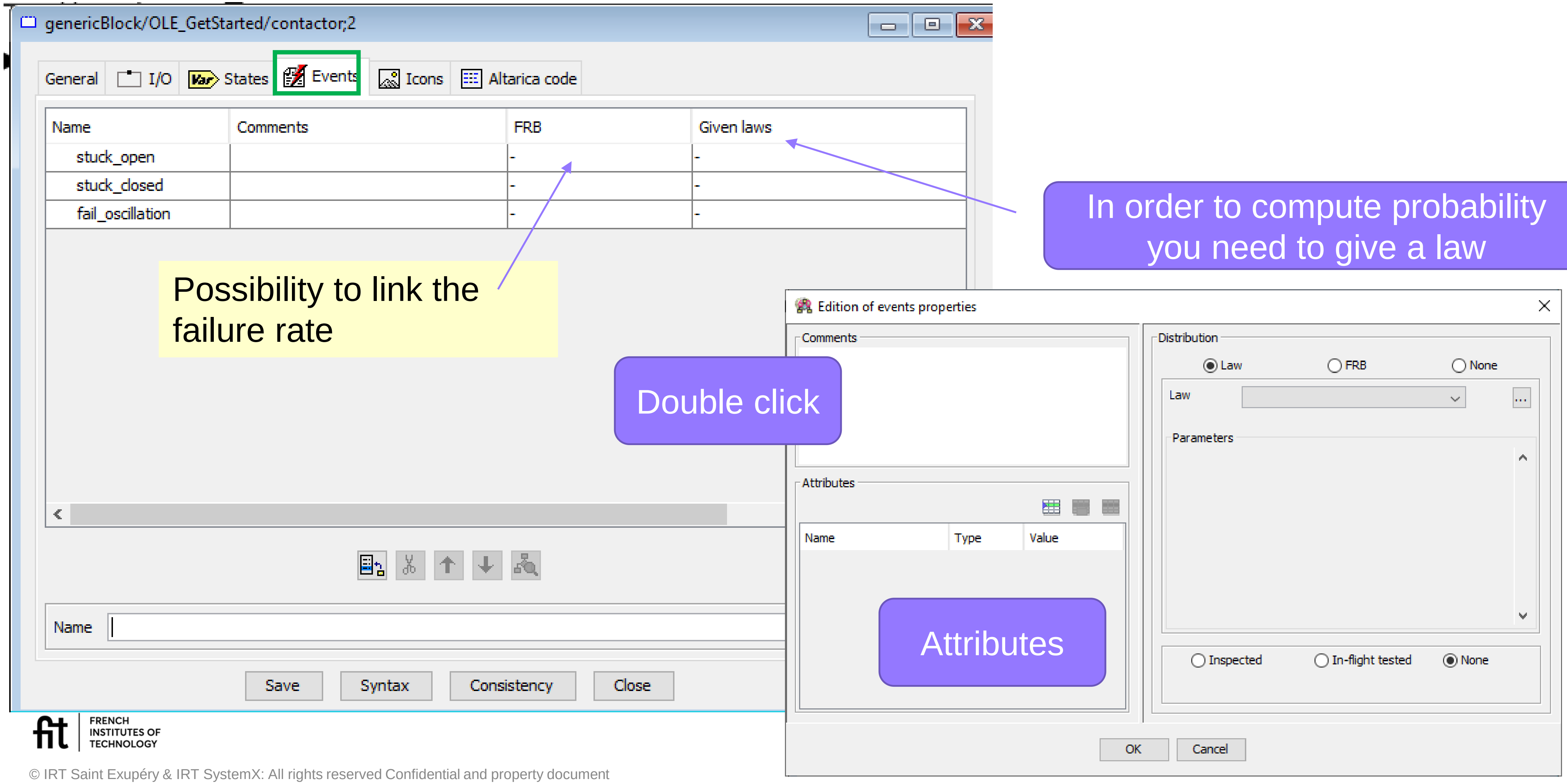
States



FRENCH INSTITUTES OF TECHNOLOGY

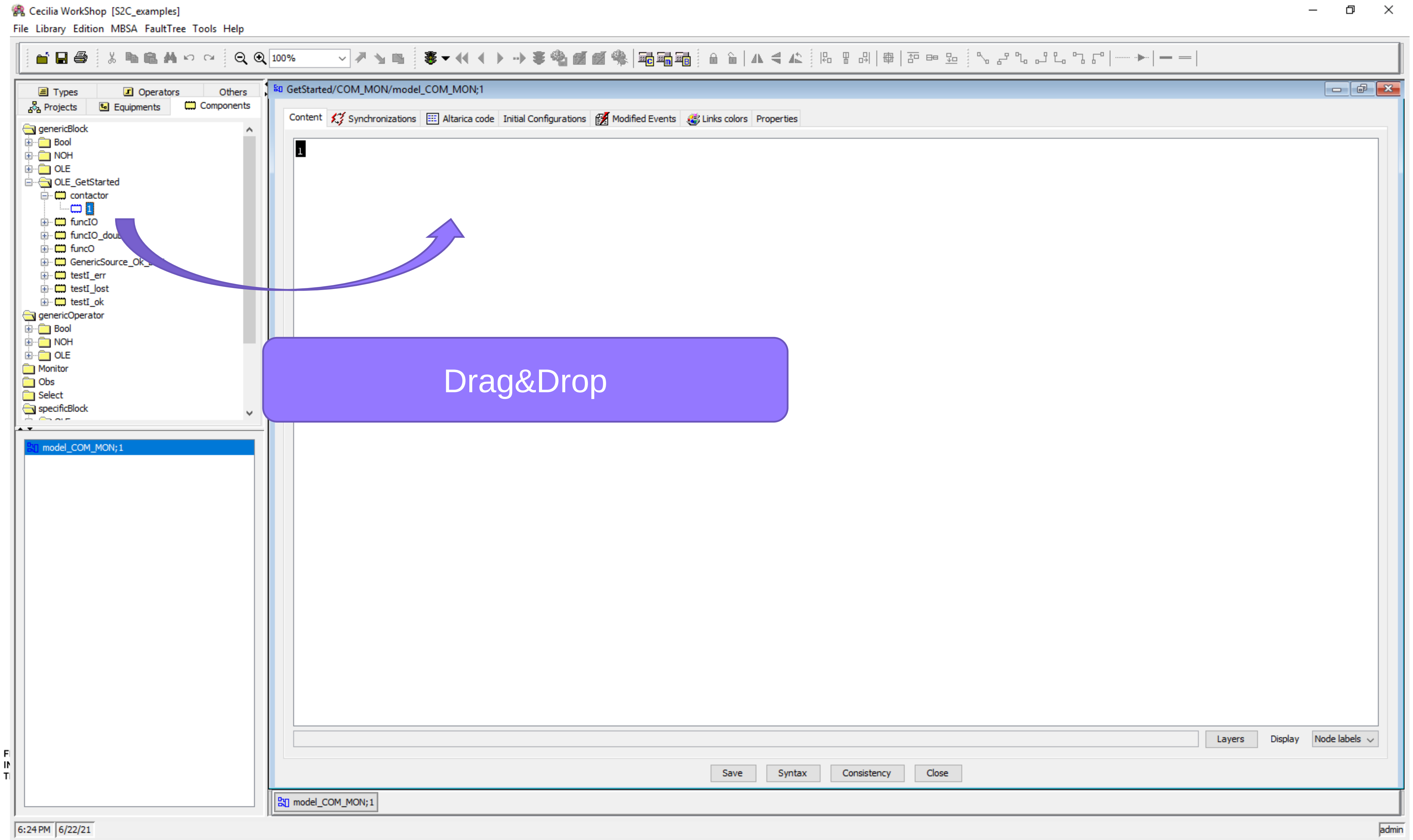
Modeling Unit

Events



Modelling unit validation

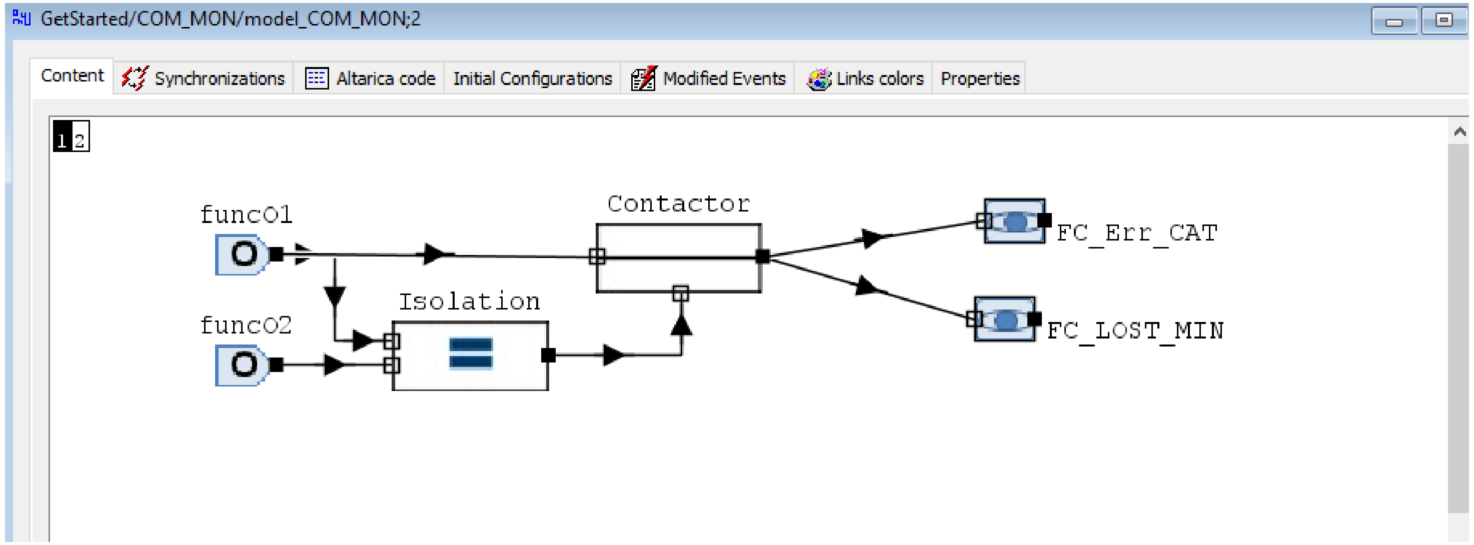
The Contractor in the global model



Global modelling

- Connexion of the modeling units, « components », « equipment » and observers
- If necessary set configuration (States initial values), otherwise set to defined default values
- Set synchronizations

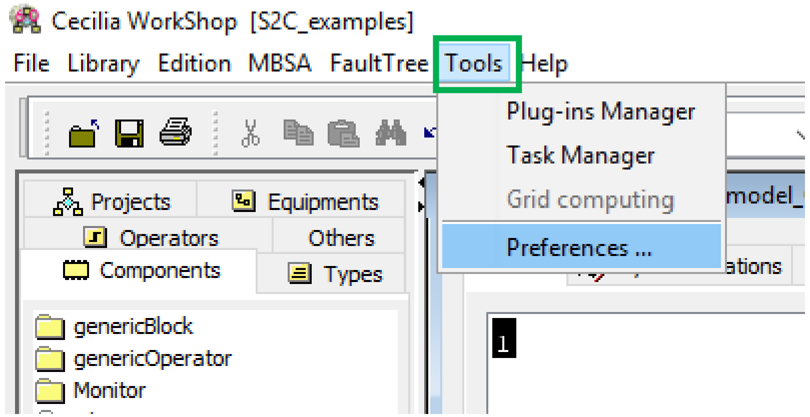
- Note: flows types and orientation need to be compatible

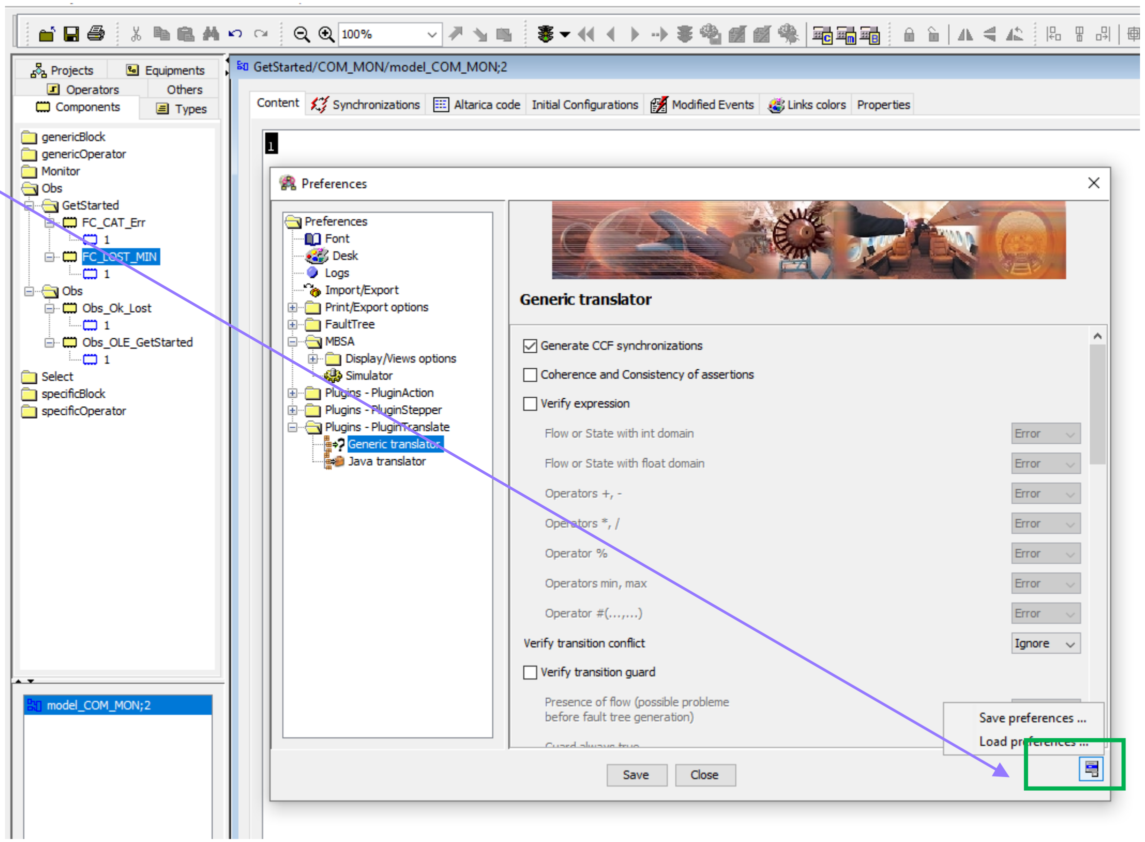


FRENCH INSTITUTES OF TECHNOLOGY

Global modeling

- Cecilia preferences : checks
    - File provided
    - High level advices
        - At the very beginning you can remove everything: checks are only checks
        - In a second time : select all checks and customize in order to understand what you remove
            - Example : do not check the events verification if you do not have events
        - The preferences can be saved or loaded





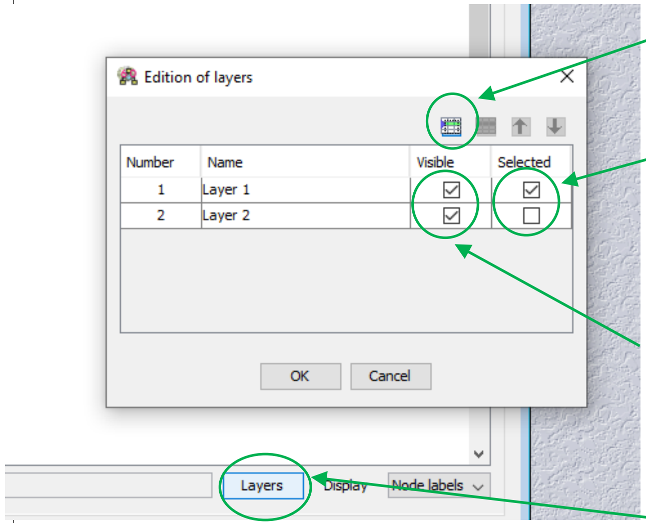
FRENCH INSTITUTES OF TECHNOLOGY

Best practices for modeling – episode 1

- The form matters
  - To understand the model, it is necessary to make it readable
    - Use harmonized connections when possible and hide them when they are unnecessary
    - Use the icons and colors and the convention associated to them
      - To ease the reading
      - To be in line with the referenced view points (System schematic or MBSE model)
    - Do not hesitate to add information or observers
    - Use the **layers**
    - Only one state variable per modeling unit and the convention associated to it
    - Prefer to use flows rather than state variable in dysfunctional modelling
      - Advantages: prefer to use double flows rahter than state Diracs (to ease cut sets / sequences exploitation)
      - Drawbacks : less readable in step by step simulation, functional modeling less easy to read

First very general modelling Advices

The Layers



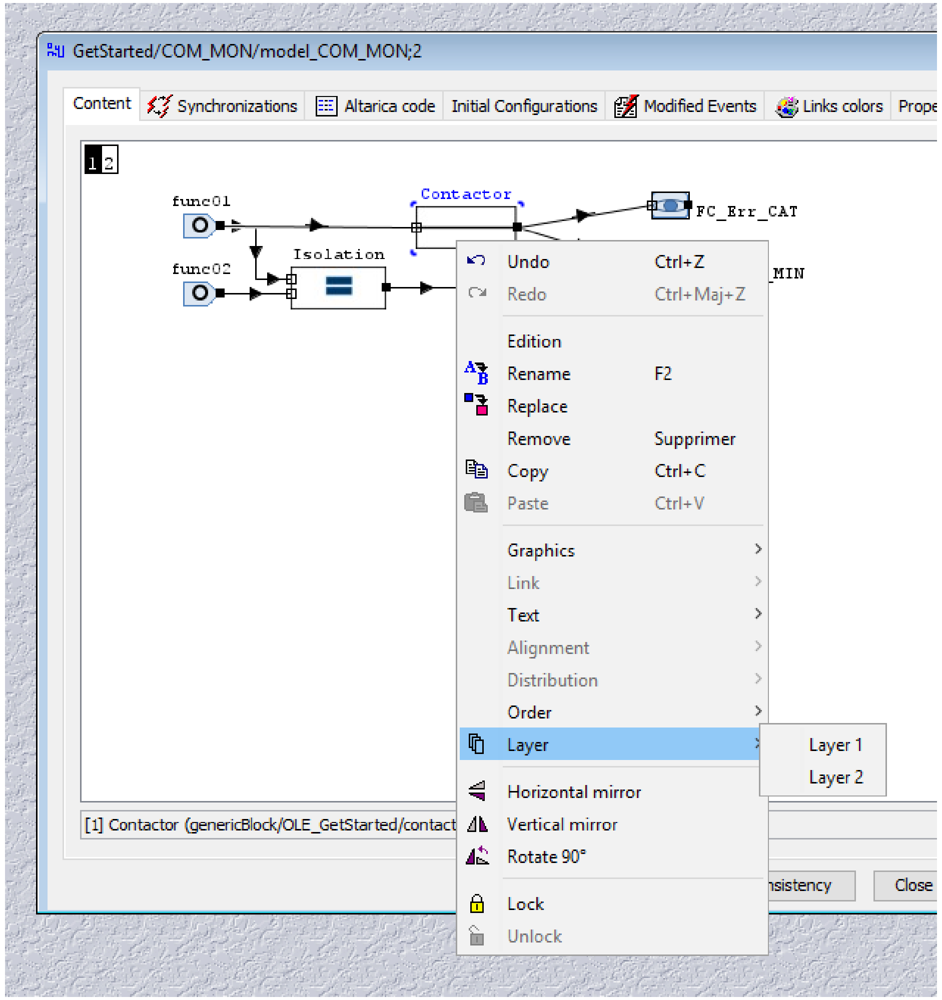
Add a layer

Selected layer for current modelling

Selected the visible layers

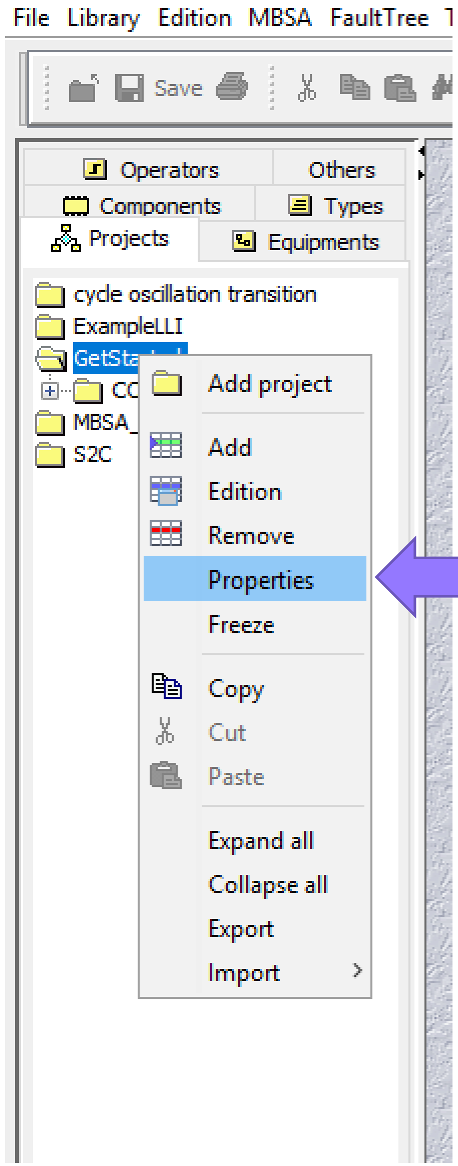
First : Open the « layer » window

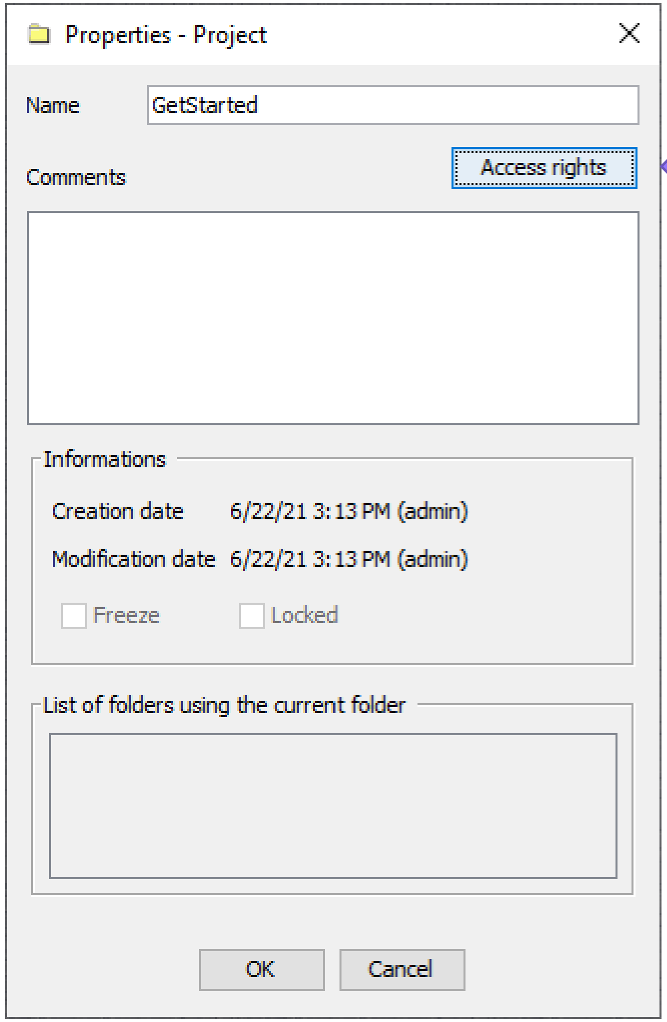
Possibility to click right on one or several selected objects to define the layer

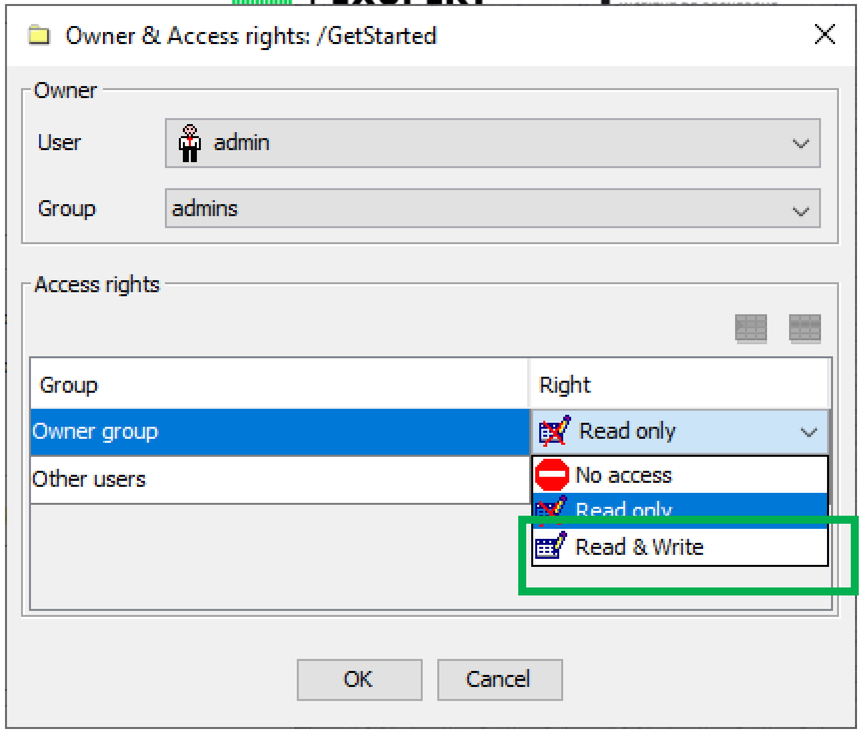


FRENCH INSTITUTES OF TECHNOLOGY

Access right



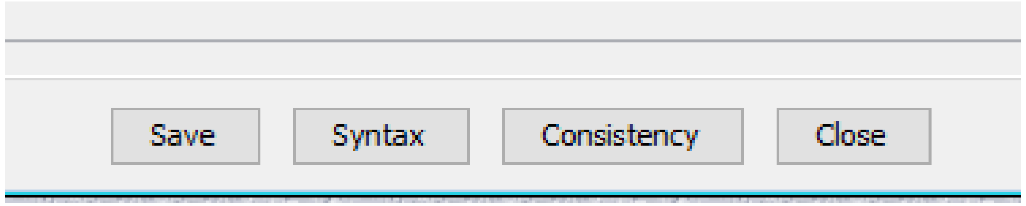




At every stage

Classical mistakes

- Be very carefull with the syntax including capitals
- Compatibility (Types, orientation,…)
- In the assertions you need to cover all possible cases in order to avoid computation problems
- If you use a Dirac law you need to handle the Dirac that may happen at the same « time »



There is syntax check and a consistency check

The consistency check is a global check

Classical Syntax rules in Satodev Cecilia

Refer to the MBSA modelling guide IRT Saint Exupéry LIV-S085L01-001 /IRT SystemX ISX-S2C-LIV-1001

```
trans

(DefinedStated=Value in State domain ) |-  event  -> DefinedStated := assigned value ;

assert

ValueOutput1 = case {

(Condition1 ) : Assigned value to ValueOutput ,

(Condition2) : Assigned value to ValueOutput ,

…

else

Value

};
```

Important notes regarding transitions
It is possible to have flows value and not only State in the transition => Announced by the checks
Possibility to introduce determinist events

Important notes regarding assertions
Conditions1 can involve States and Flows
The case allows to go through all possible conditions
As soon one condition is true the value is assigned and you get out of the « case. »

Best practice for modeling – episode 2

- Use the simulation capacities to validate the global behavior of your model

- Validate your model!
  - Check non regression of your previous results when you made a change in your model.
  - Validate your model « philosophy » in order to check your model will fulfil the needs: before being too far in the modelling validate your model answer the need and is readable and easy to validate

Cecilia WS

- Construction

  Use the bricks hierarchy in order to limit the complexity of a brick level

  Use incremental modeling

FRENCH INSTITUTES OF TECHNOLOGY

Table of contents

- How to get started with MBSA
  - Main principles
  - Questions to address before starting
  - The different steps to follow

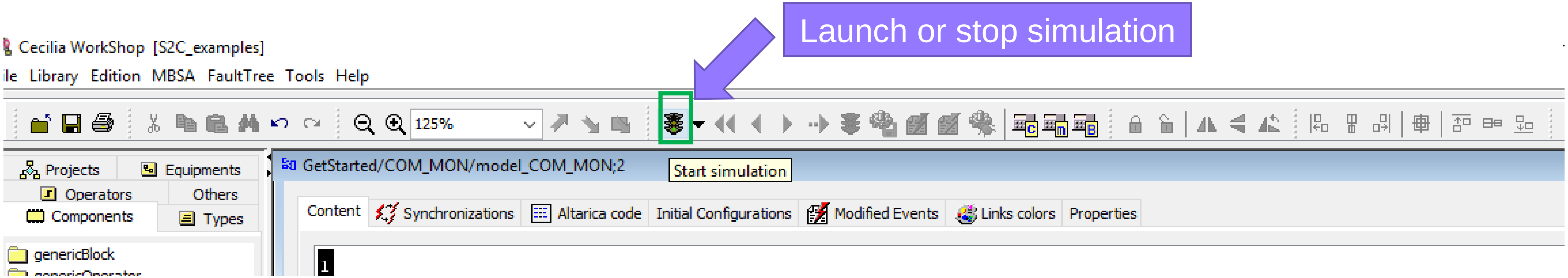
- Modelling the example
  - Go Through the tool – SATODEV Cecilia WS
  - How do I do in practice to model
  - How do I do in practice to simulate
  - How do I do in practice to compute

FIT FRENCH INSTITUTES OF TECHNOLOGY

Step by Step Simulation

If the Syntax and Consistency you can start your simulation

Note: it may not go if you still have mistakes in your model depending on the checks performed and of the kind of error you have made



FRENCH INSTITUTES OF TECHNOLOGY

Step by Step Simulation

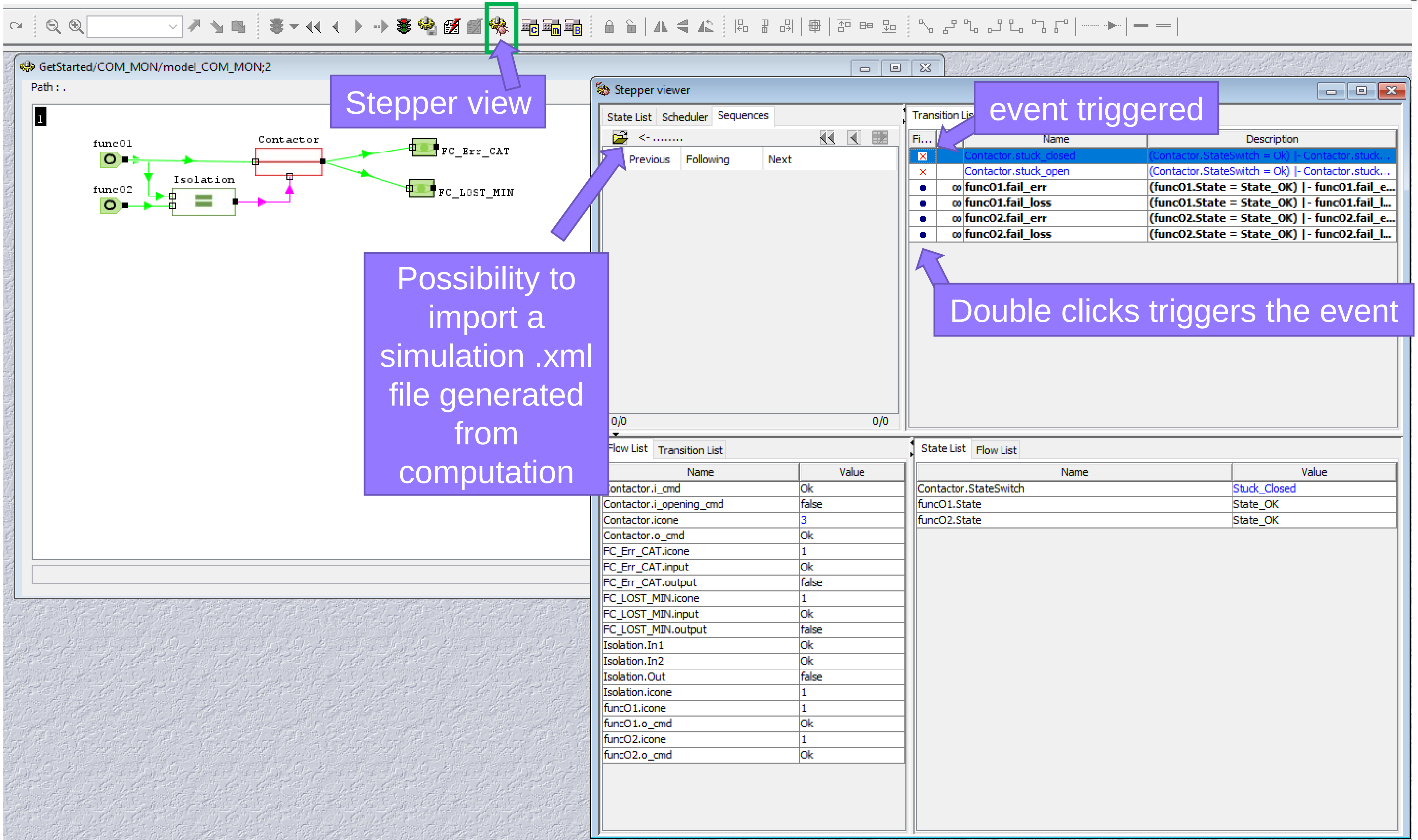


Table of contents

FRENCH INSTITUTES OF TECHNOLOGY

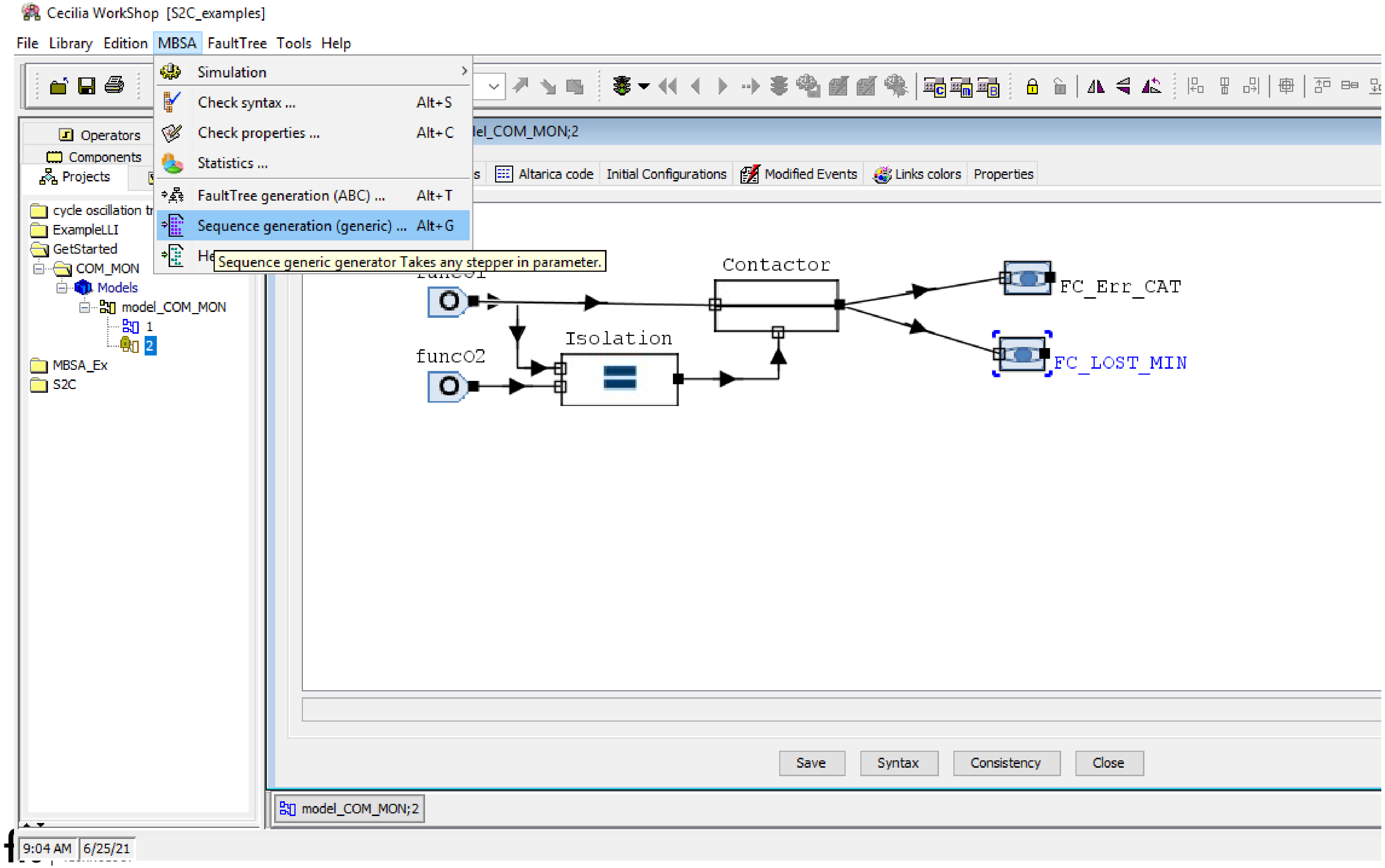
Table of contents

- How to get started with MBSA
  - Main principles
  - Questions to address before starting
  - The different steps to follow

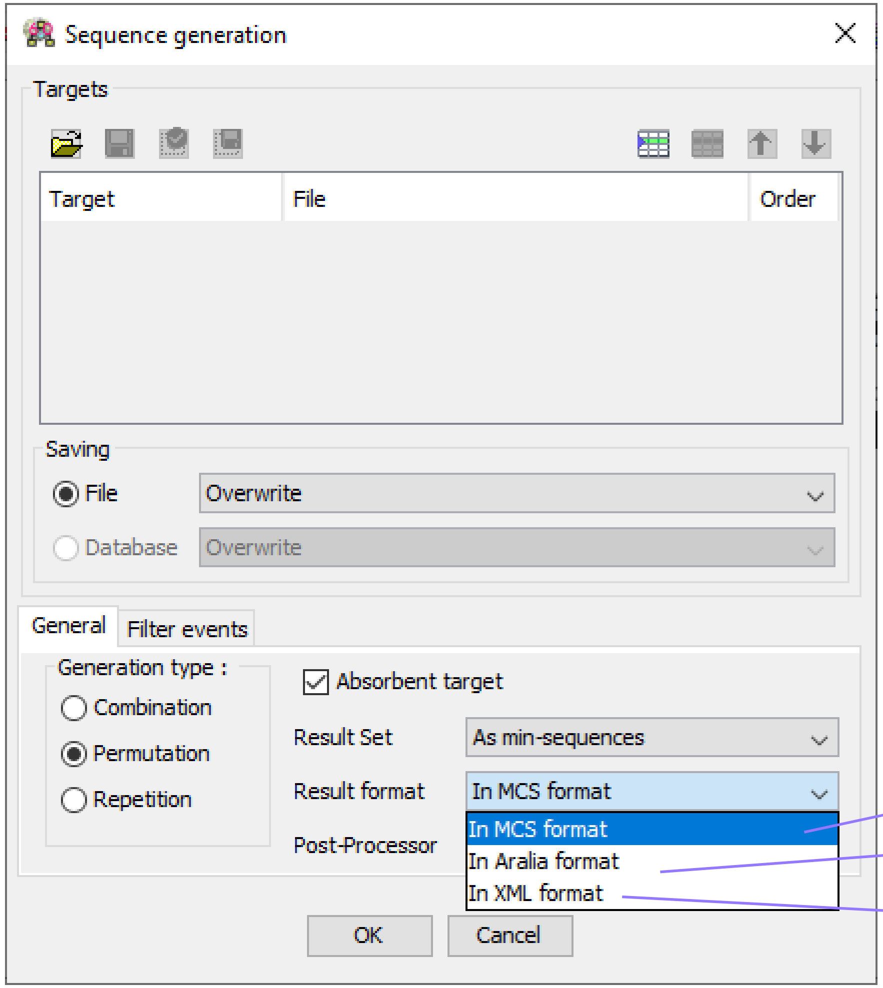
- Modelling the example
  - Go through the tool
  - To simulate
  - To compute

FRENCH INSTITUTES OF TECHNOLOGY

Compute


Cecilia WorkShop [S2C_examples]
File Library Edition MBSA FaultTree Tools Help
Simulation
Check syntax ... Alt+S
Check properties ... Alt+C
Statistics ...
FaultTree generation (ABC) ... Alt+T
Sequence generation (generic) ... Alt+G
Sequence generic generator Takes any stepper in parameter.
Operators
Components
Projects
cycle oscillation
ExampleLLI
GetStarted
COM_MON
Models
model_COM_MON
1
2
MBSA_Ex
S2C
Altarica code
Initial Configurations
Modified Events
Links colors
Properties
Contactor
Isolation
funcO2
FC_Err_CAT
FC_LOST_MIN
Save
Syntax
Consistency
Close
model_COM_MON;2
9:04 AM
6/25/21

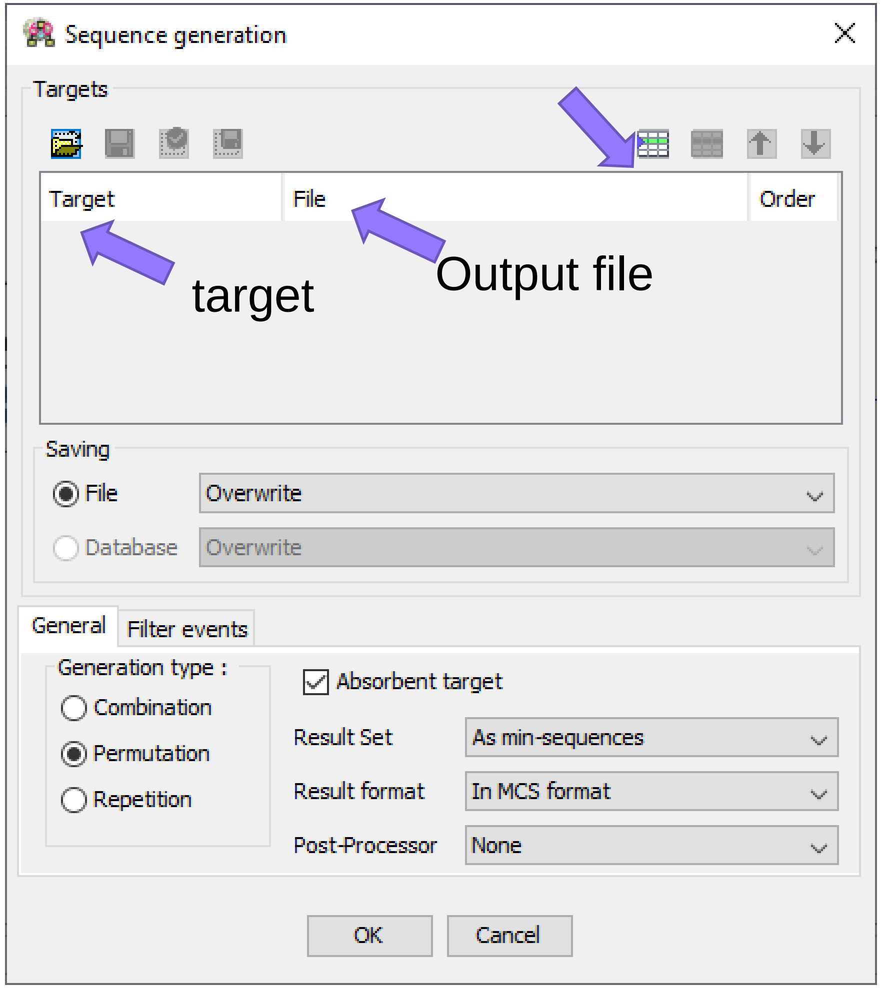
Compute output format

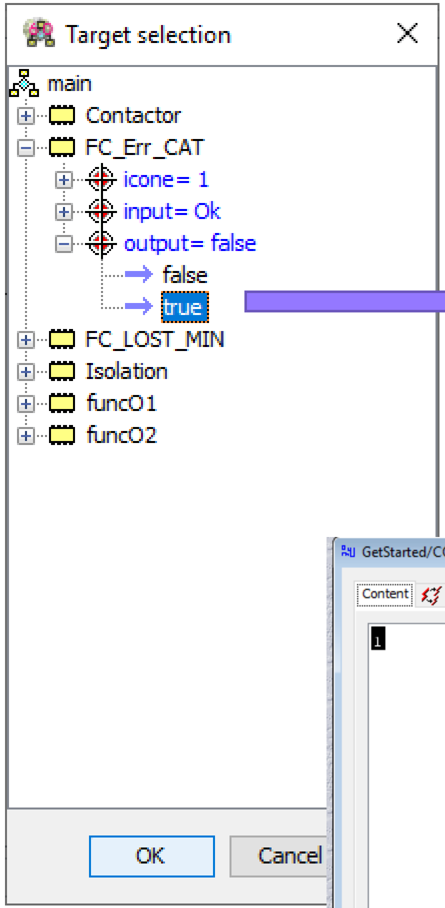


Minimum cutsets
Boolean equation (.xml format) to be imported
xml file to support simulation

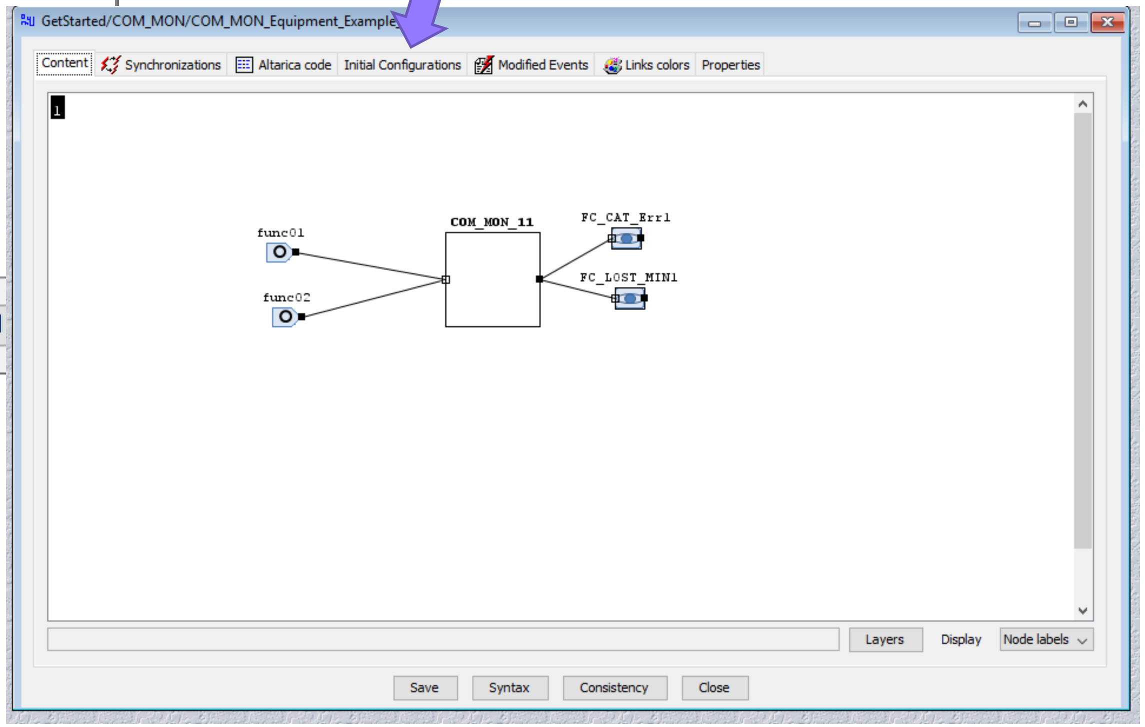
Compute

First step: Add one computation



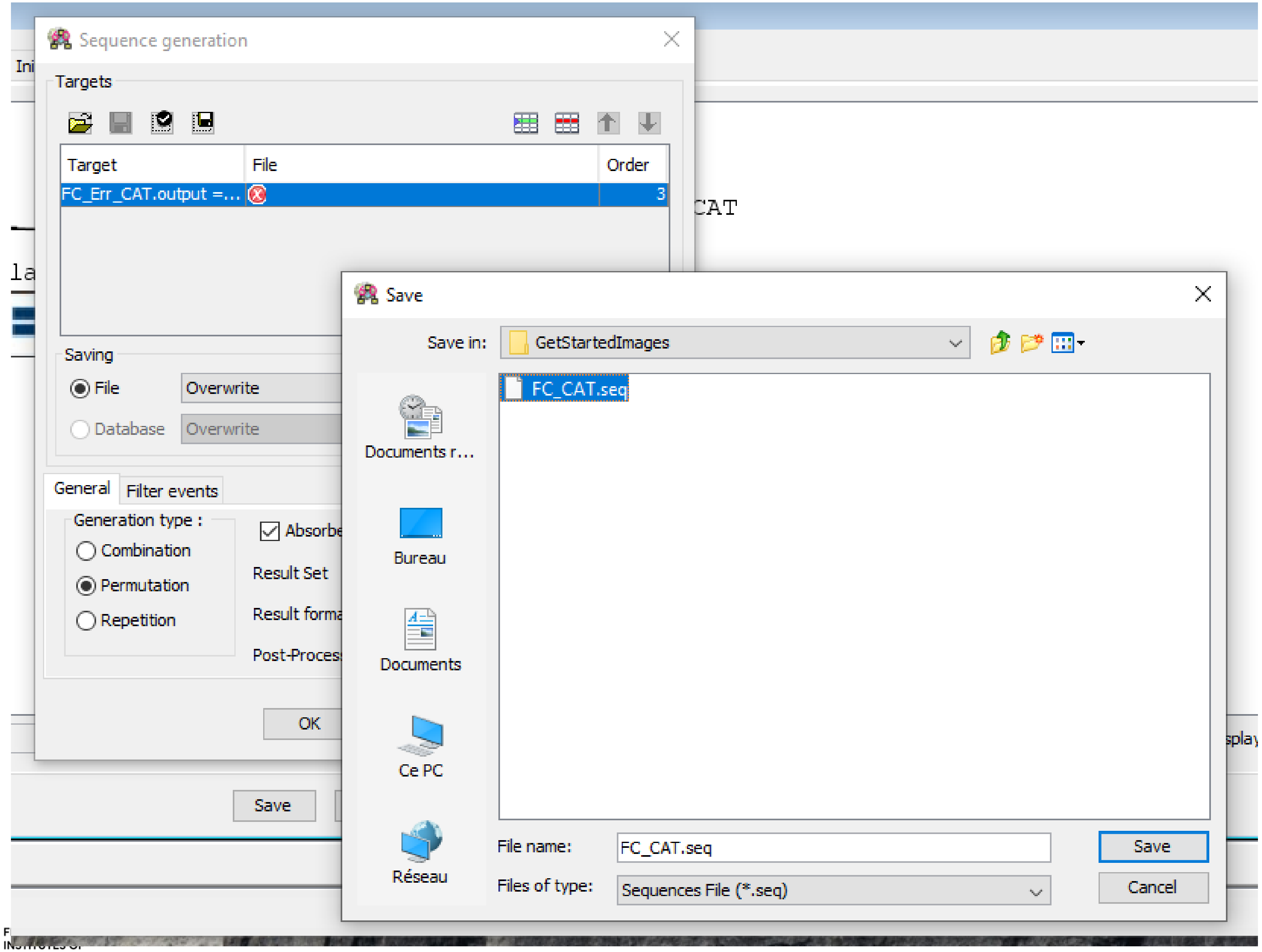


Cutsets of this value
Regarding chosen
Configuration (States)
Note: possibility to set the configuration during simulation

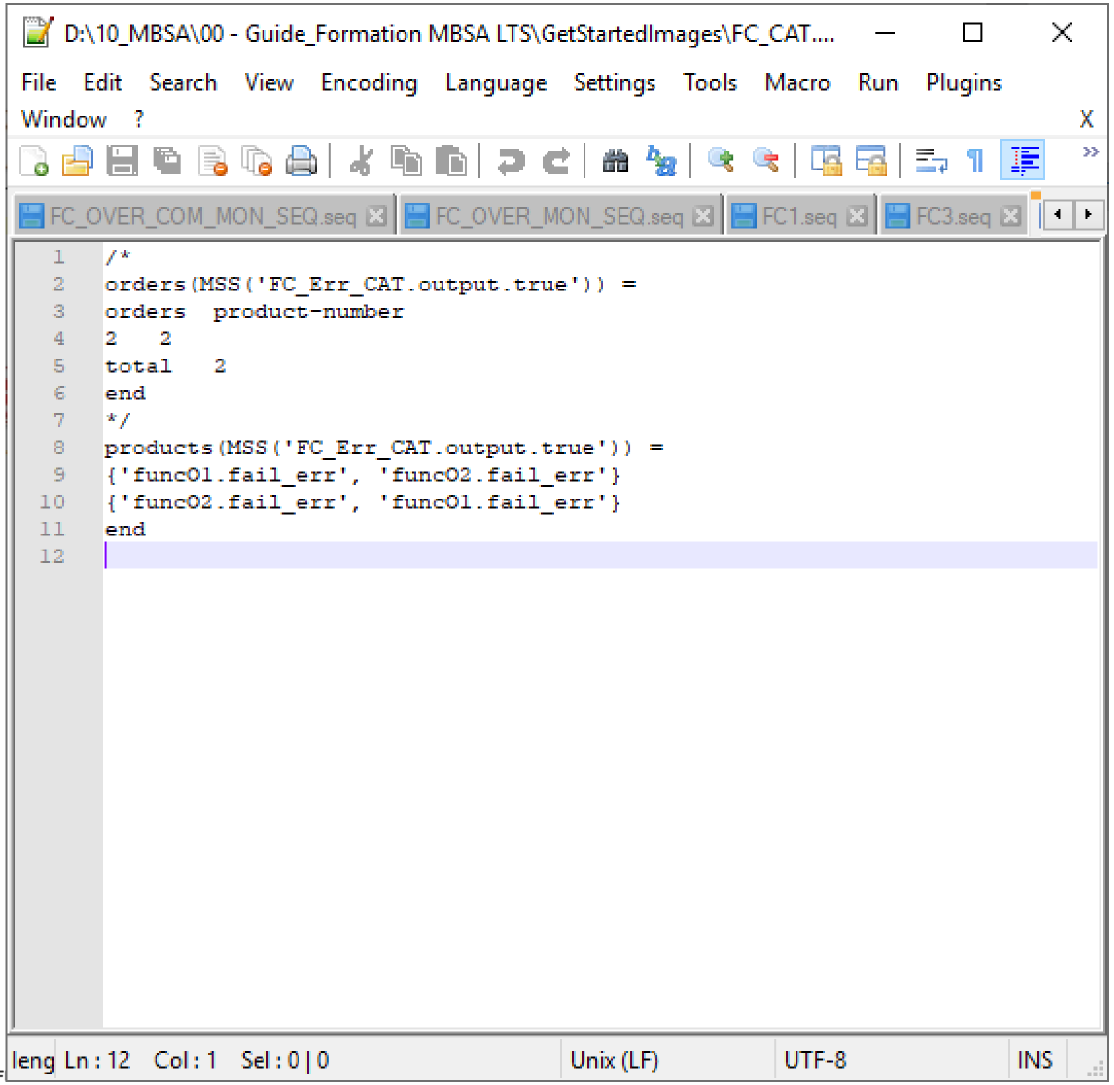


fit FRENCH INSTITUTES OF TECHNOLOGY

Compute: compute from the Aralia file



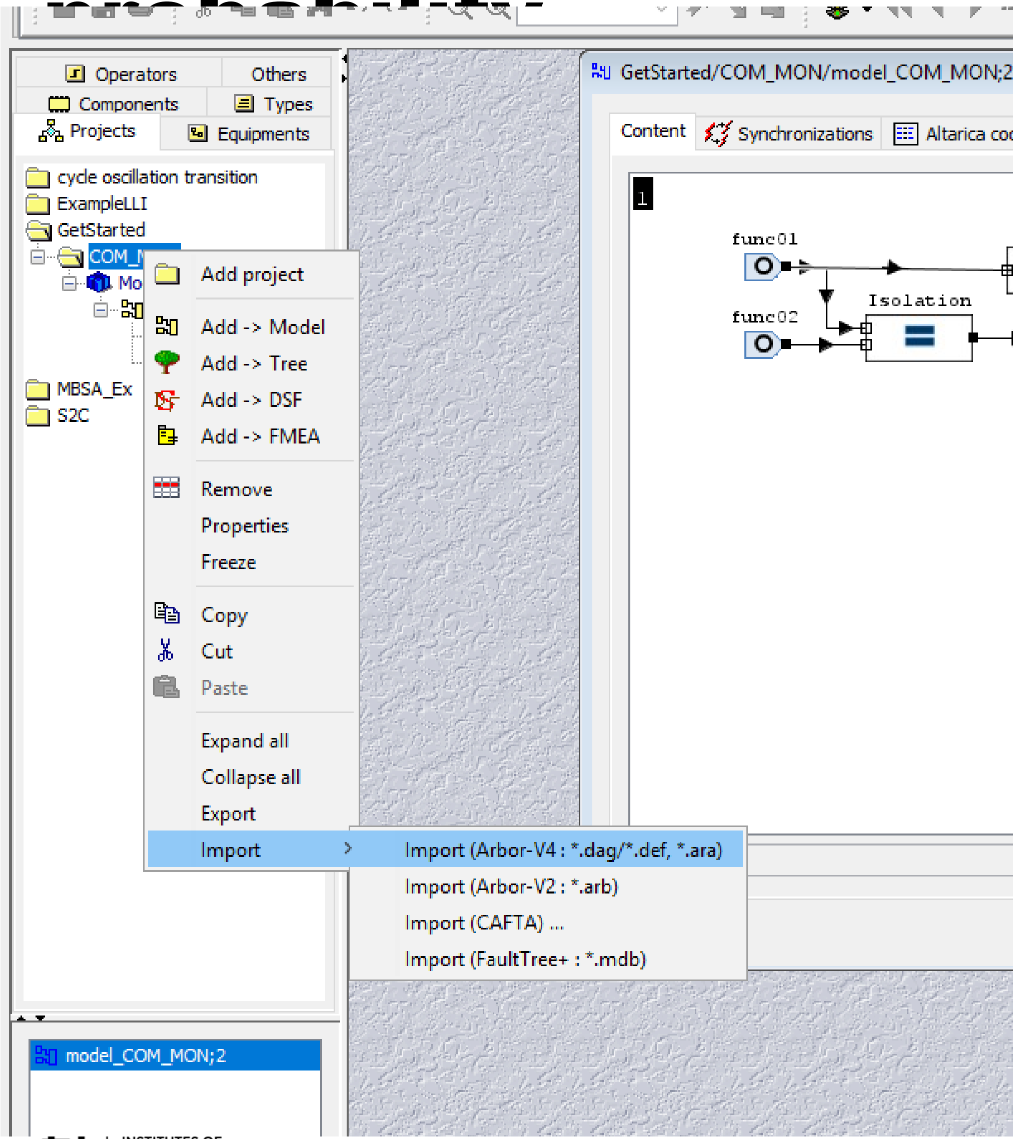
Compute: the results



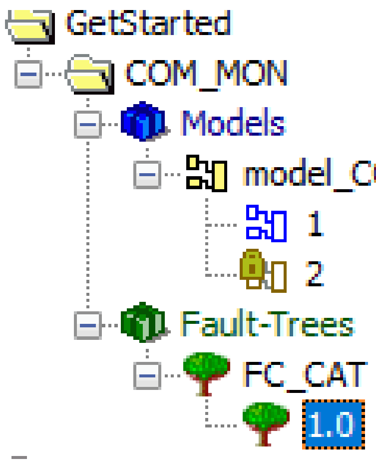
You can also generate an Aralia object and import it

You can generate an .xml to support the simulation

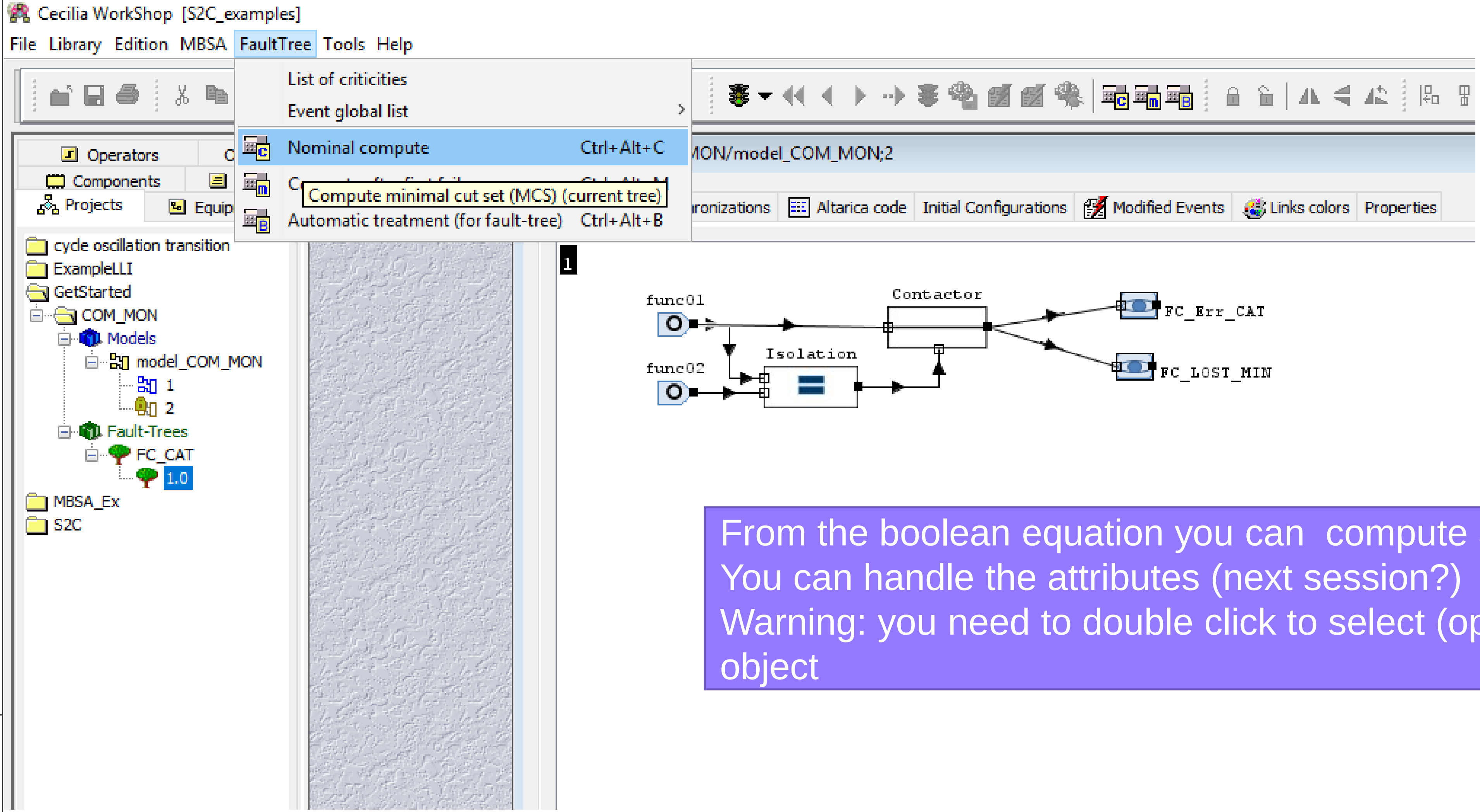
Compute: import Aralia file to compute



Import a boolean equation
Maybe a tree in the current version
But not necessarily



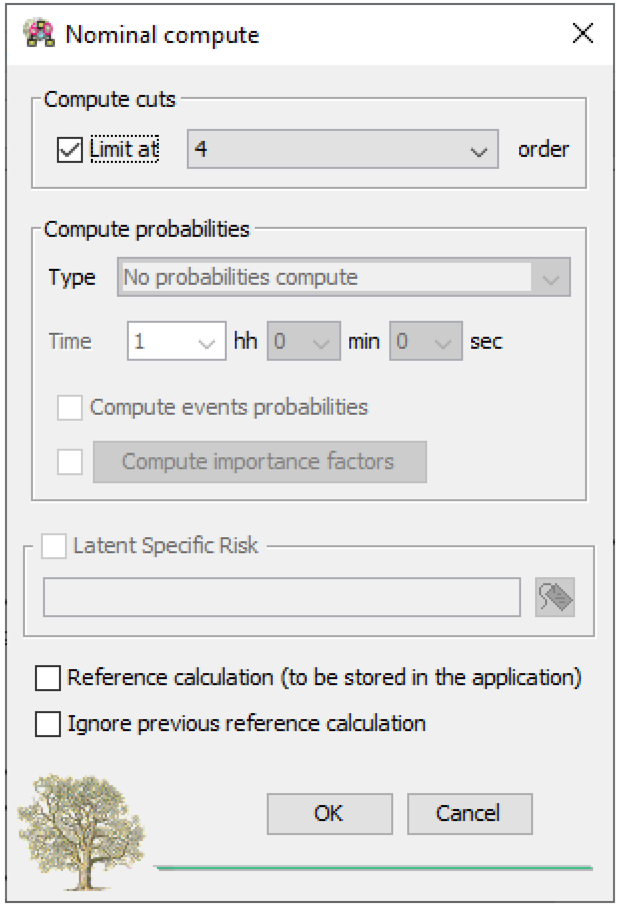
Compute: compute from the Aralia file

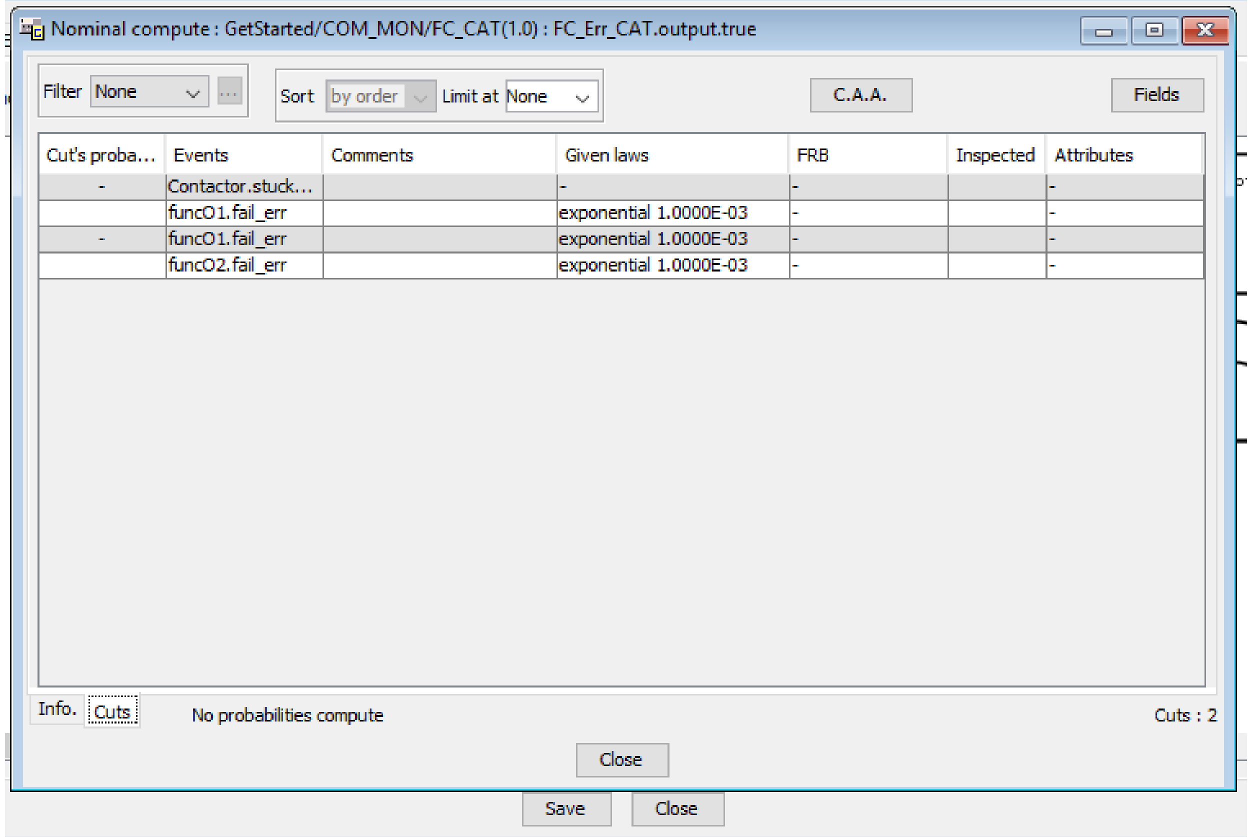


From the boolean equation you can compute as well
You can handle the attributes (next session?)
Warning: you need to double click to select (open) the object

FRENCH INSTITUTES OF TECHNOLOGY

Compute: The results





At this level if you define attributes you will have access to cut set by attributes

Table of contents

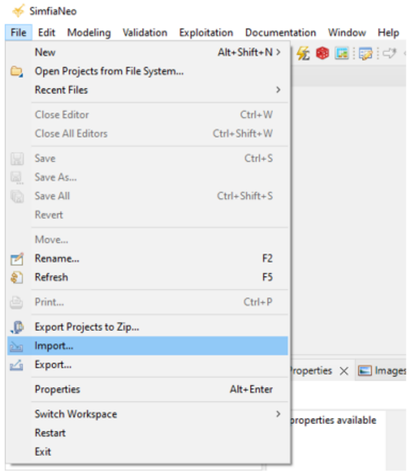
Table of contents

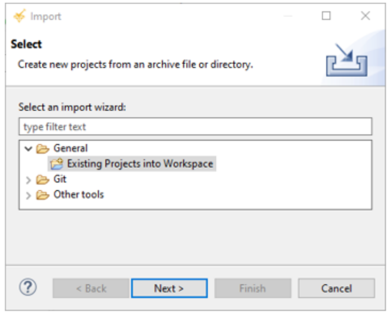
FRENCH INSTITUTES OF TECHNOLOGY

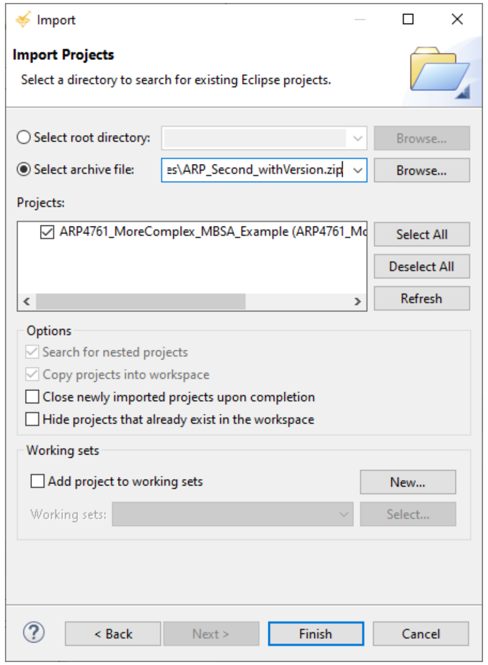
Go Through the tool – Airbus Protect SimfiaNeo

Opening a model

- *File> Import… then select General> Existing Projects into Workspace then Next>.*
- Next wizard enables to choose between:
- “Select root directory”: enables choosing a folder containing one or several unzipped SimfiaNeo projects
- “Select archive file”: enables choosing a zipped (.zip, .tar, .tar.gz, .tgz, .jar) file containing one or several SimfiaNeo projects







FRENCH INSTITUTES OF TECHNOLOGY

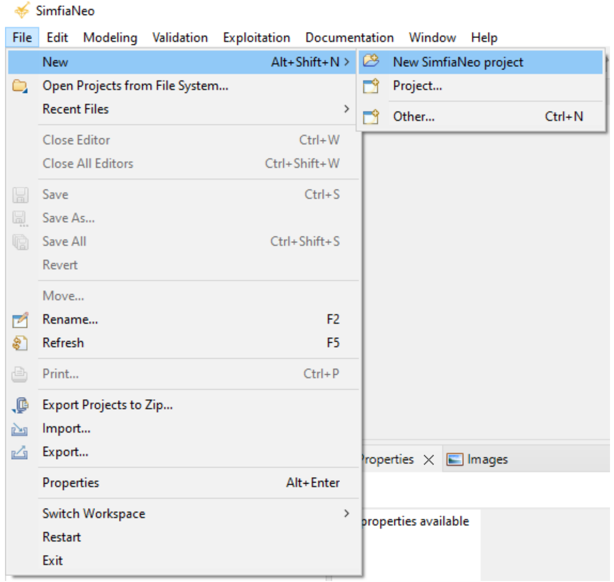
Go Through the tool – Airbus Protect SimfiaNeo

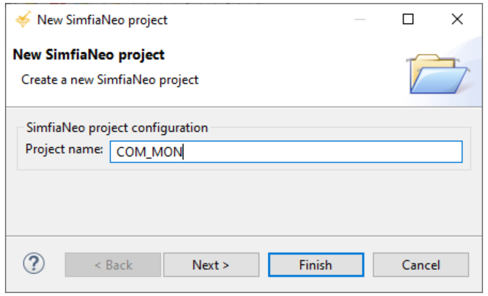
How to create a new project?

- Use the menu File> New> New SimfiaNeo project

- Input a name for your Project and select Finish.

- SimfiaNeo automatically creates a Project, System and Model with the same name. They can be seen in the Model Explorer.



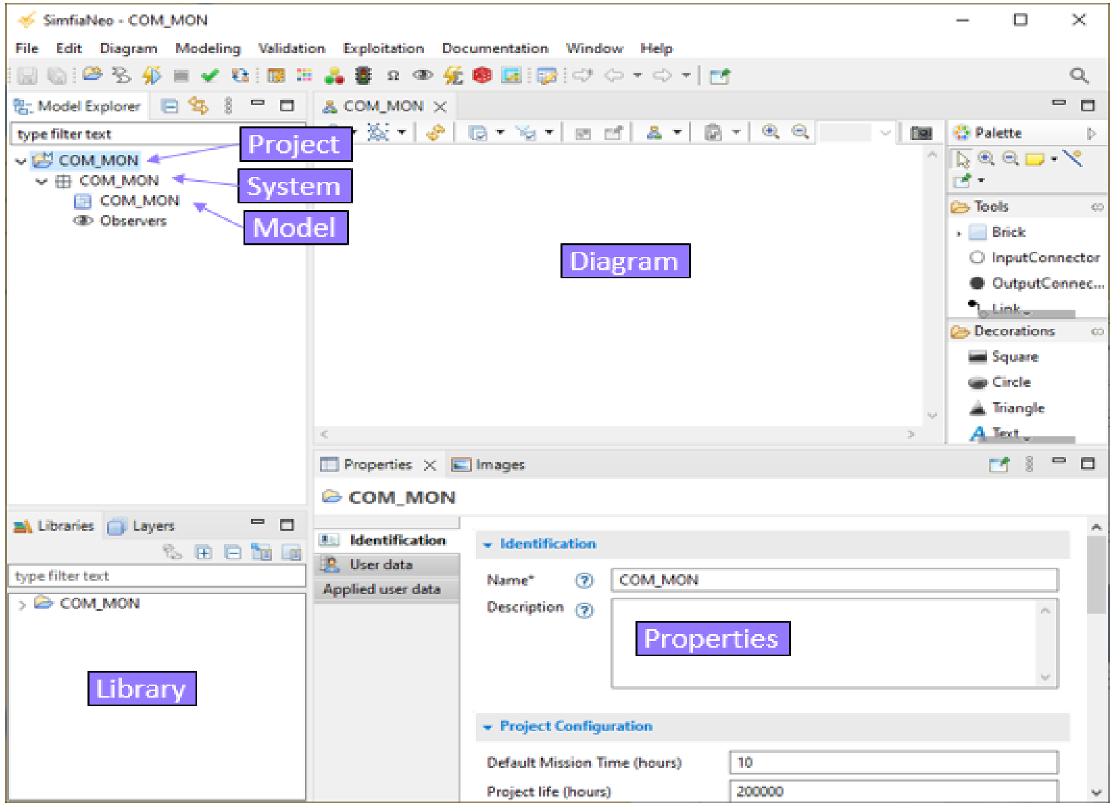


FRENCH INSTITUTES OF TECHNOLOGY

Go Through the tool – Airbus Protect SimfiaNeo

Project view in the tool

- Top-level diagram of the Model is also automatically opened. Properties view is automatically updated depending on currently selected element in Model Explorer, Diagram, Library…

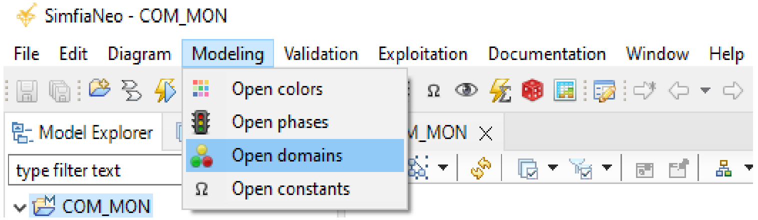


FRENCH INSTITUTES OF TECHNOLOGY

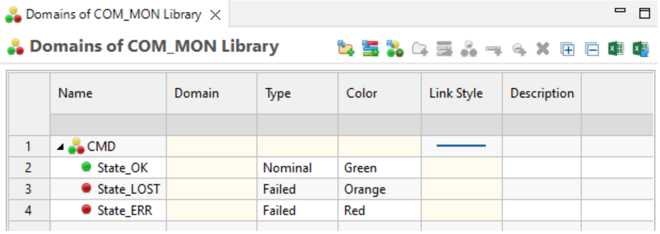
Go Through the tool – Airbus Protect SimfiaNeo

Create a domain

- Domains are managed through the Domains table. Open the table by selecting the menu Modeling> Open domains.



- Domains can be organized in folders, but this is not mandatory. Creation buttons are situated in the top-right corner of the table.

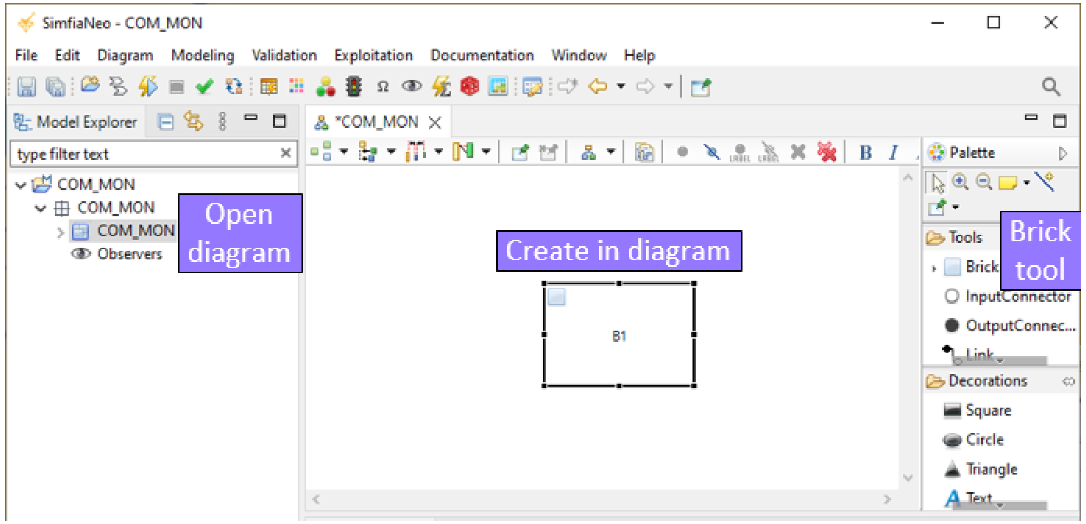


Names of domains and values can be customized either directly in the table or through the Properties view when the corresponding line/cell is selected in the table. Column named Type is used mainly for documentation purposes and can be ignored when getting started.
Structured domains (for connectors containing several variables) can also be defined in this table. Each variable in the structured domain is called a field.

Go Through the tool – Airbus Protect SimfiaNeo

Modelling unit creation in the tools; contactor example

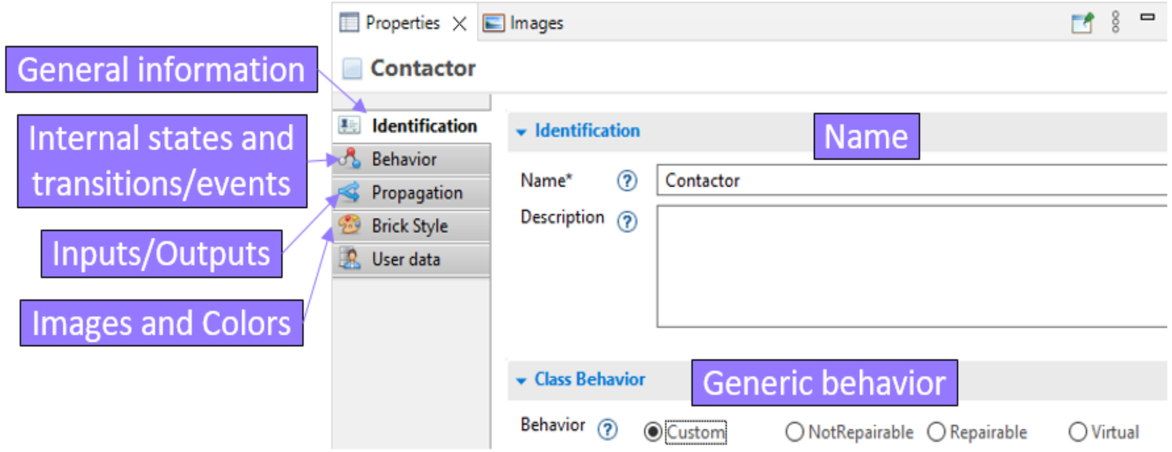
- This chapter deals with how to build the modelling unit with the example of the COM/MON contactor. It describes the GUI (Graphical User Interface).
- In SimfiaNeo, modelling units are called bricks, and are created directly in the model. They can then be added to the Project Library if deemed necessary for reuse.
- To create a new brick, open the Model Diagram (e.g. by double-clicking on the Model level in the Model Explorer on the left side), select the Brick tool in the Palette and click in the Diagram.



Go Through the tool – Airbus Protect SimfiaNeo

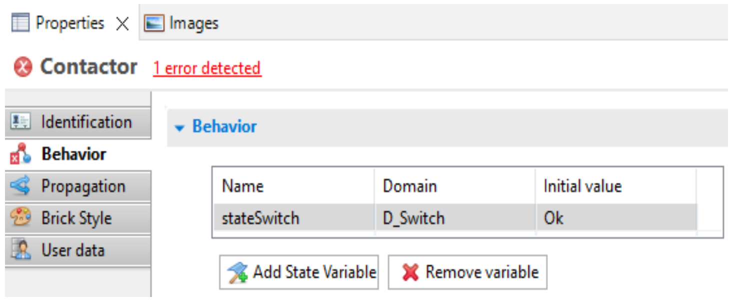
Modelling unit creation in the tools; contactor example

- Right after its creation, the new brick is automatically selected in the diagram, hence the Properties view at the bottom of the screen displays information on this brick. Modify its name to “Contactor” and its generic behavior to “Custom”.



Manage internal state variables

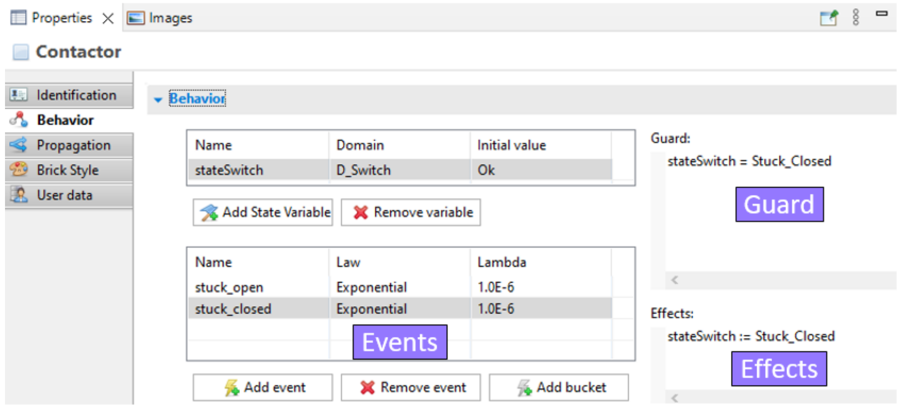
- State variables are managed in the Behavior tab of the Properties view. Dedicated table and buttons enable creating/deleting variables, renaming them, switching their domain, and setting their value at initial time.



Go Through the tool – Airbus Protect SimfiaNeo

Manage events and transitions

- Events and transitions are managed in the Behavior tab of the Properties view. Dedicated table and buttons enable creating/deleting events, renaming them, and customizing their probability/determinist law.



Selecting an event in the table on the left updates the contents in the fields Guard and Effects on the right. Guard field is used to input the Guard of the corresponding transition. Effects field is used to input the actions of the corresponding transition. In both fields, the shortcut Ctrl-Space enables using auto-completion feature.
Probability laws are filled directly in the Behavior tab (see above). Constants can be created in the Constants table (menu Modeling> Open constants) to have several laws share the same numerical parameters.

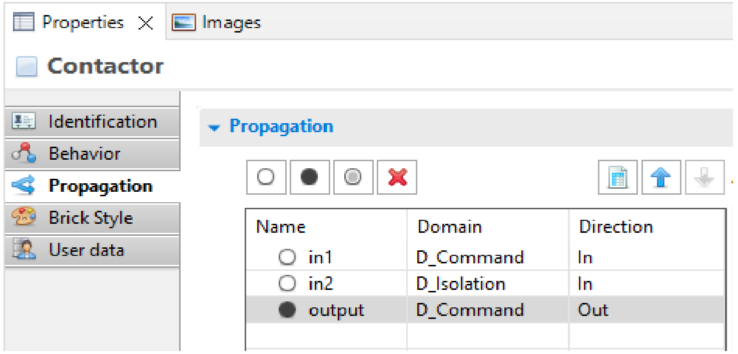
FRENCH INSTITUTES OF TECHNOLOGY

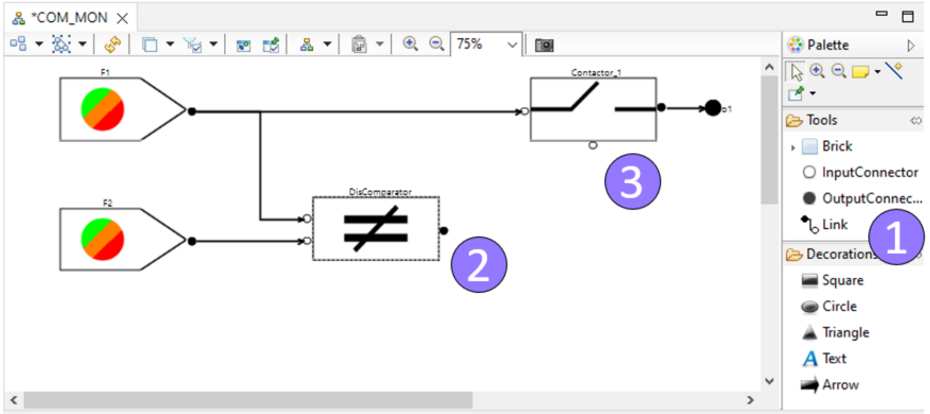
Go Through the tool – Airbus Protect SimfiaNeo

Create the connectors and links (flows)

- Prior to link creation, it is recommended to create connectors on the bricks. Connectors can be created by using the connector tools in the Palette of Diagrams, or by using the Propagation tab of the Properties view of a brick. In this tab, dedicated table and buttons enable to create or delete connectors, rename them and define their domain. White dots are input connectors while black dots are output connectors.

To create the links, go in a Diagram, select the Link tool in the Palette, click on an output connector (black dot), then click on an input connector (white dot). It is also possible to directly click on bricks instead of on connectors, in which case new connectors are created.



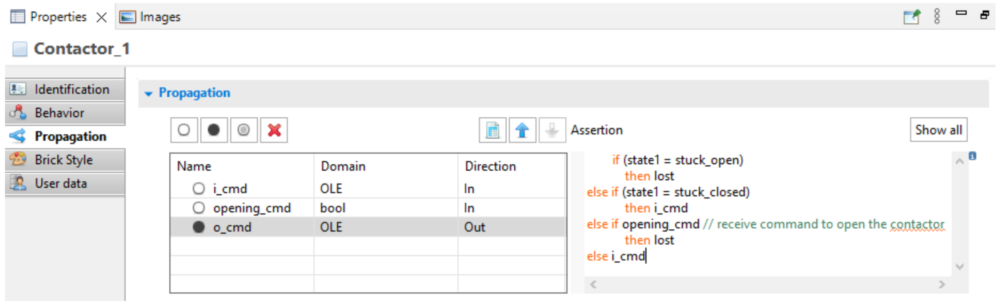


FRENCH INSTITUTES OF TECHNOLOGY

Go Through the tool – Airbus Protect SimfiaNeo

Create an assertion

- Assertions are filled in the Propagation tab of the Properties view of a brick. When selecting an output connector (black dot) in the table on the left side, the right side section is updated to display and edit the assertion.



In this field, the shortcut Ctrl-Space enables using auto-completion feature.

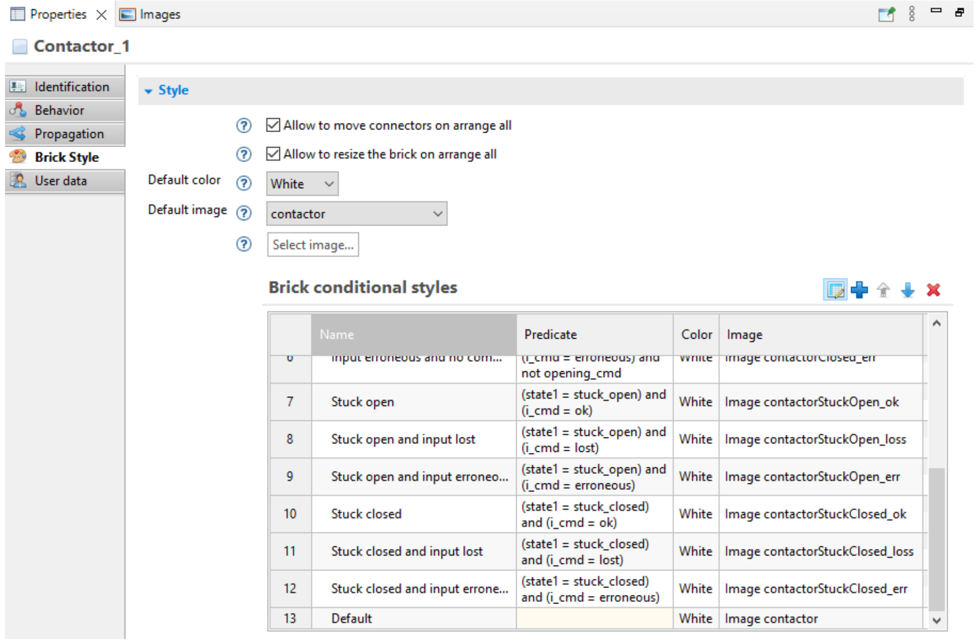
FRENCH INSTITUTES OF TECHNOLOGY

Go Through the tool – Airbus Protect SimfiaNeo

Create the conditional style

- Style of the brick is defined in the Brick Style tab of the Properties view of a brick. Default color and Default image drop-down menus enable defining style in edition mode.

- Default conditional style for simulation is based on internal state variable. It can be customized by activating the bottom table. Predicates are user-defined Boolean formula to determine the image and/or color of the brick. In this Predicate field, the shortcut Ctrl-Space enables using auto-completion feature.

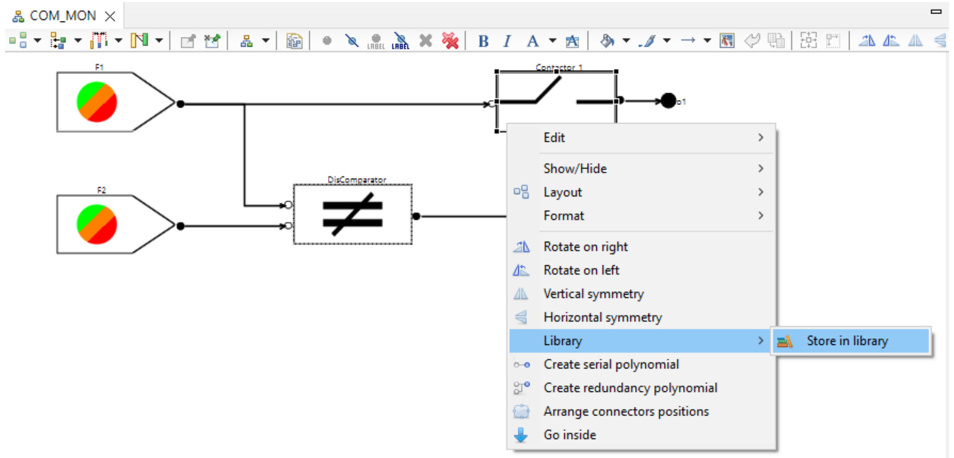


Go Through the tool – Airbus Protect SimfiaNeo

Put a brick in Library

- A brick that was created in a diagram can be added to the Library at any step. This is done with a right-click on a brick in a diagram and selecting Library> Store in library.

- Insert the name to use in the Library to finish. This brick is now displayed in the project Library available in the bottom-left corner.



FIT FRENCH INSTITUTES OF TECHNOLOGY

Go Through the tool – Airbus Protect SimfiaNeo

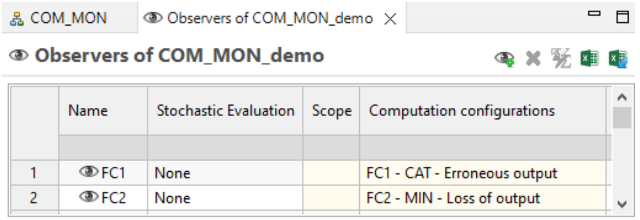
Instantiate a brick

- If a brick is already defined in Library, it can be instantiated in the model. This can be performed by following either of the following methods:
- Drag-and-drop the brick from the Library view (bottom-left corner) to the diagram
- Right-click in an empty space of the diagram and select Library> Instantiate existing class

Go Through the tool – Airbus Protect SimfiaNeo

Create the observers (link with FC)

- Observers are managed through the Observers table. Open the table by double-clicking on the Observers (eye icon) in the Model Explorer



	Name	Stochastic Evaluation	Scope	Computation configurations
1	FC1	None		FC1 - CAT - Erroneous output
2	FC2	None		FC2 - MIN - Loss of output

Creation button is situated in the top-right corner of the table. Name of the observer needs to follow AltaRica variables naming rules (mainly no spaces, and no digit as the first character). Stochastic Evaluation is kept to None (this option is linked to Monte-Carlo simulation). When selecting an observer, its Boolean expression can be customized in the Properties view. This expression takes the value *true* when the feared situation is reached. In this field, the shortcut Ctrl-Space enables using auto-completion feature.

FRENCH INSTITUTES OF TECHNOLOGY

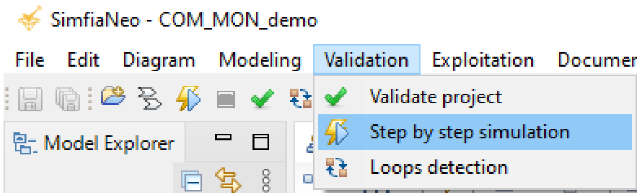
Table of contents

FRENCH INSTITUTES OF TECHNOLOGY

Go Through the tool – Airbus Protect SimfiaNeo

Launch a step-by-step simulation

- To launch the step-by-step simulation, use the menu Validation> Step by step simulation



- Events can be triggered by double-click on the left, or with a right-click on bricks in diagrams. Simulation is exited by using the Stop simulation (red square) button.

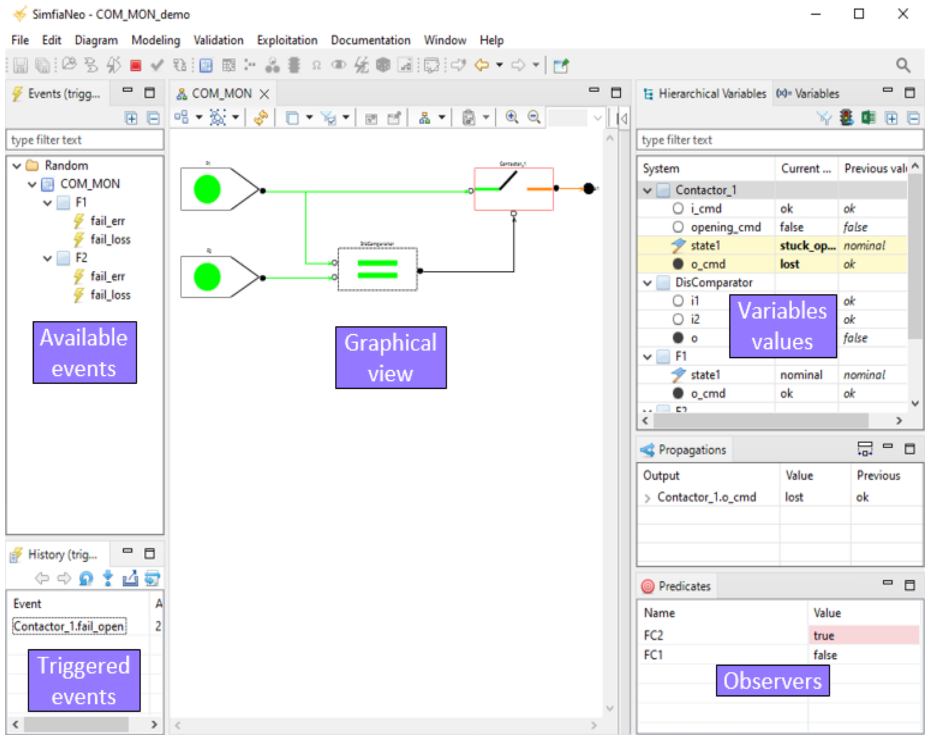


Table of contents

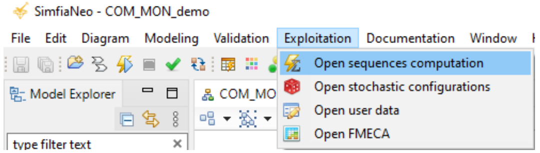
- How to get started with MBSA
  - Main principles
  - Questions to address before starting
  - The different steps to follow

- Modelling the example
  - Go Through the tool – Airbus Protect SimfiaNeo
  - How do I do in practice to model
  - How do I do in practice to simulate
  - How do I do in practice to compute

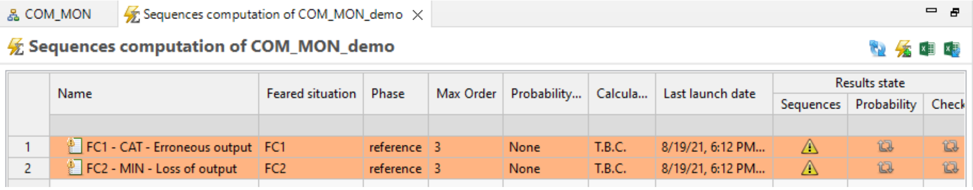
Go Through the tool – Airbus Protect SimfiaNeo

Calculate the Cut-Sets

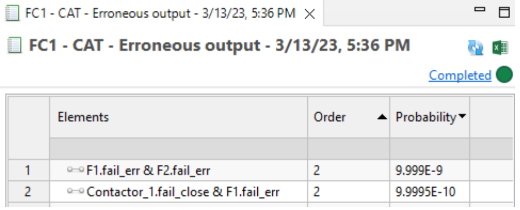
- To compute the Cut-Sets, the first step is to define the computation options. To open the corresponding table, use the menu Exploitation> Open sequences computation.



- Creation buttons are situated in the top-right corner of the table.



- After creating a computation config, go to the Options tab of the Properties view to customize the computation options. In particular, it is possible to define if you want only qualitative results or also would like the numerical probabilities. To launch the computation, right-click on the line in the table and select Execute.



- Results are stored in the project but can also be exported in Excel format.

Table of contents

- How to get started with MBSA
  - Main principles
  - Questions to address before starting
  - The different steps to follow

- Modelling the example
  - Go Through the tool – Open AltaRica
  - How do I do in practice to model
  - How do I do in practice to simulate
  - How do I do in practice to compute

FRENCH INSTITUTES OF TECHNOLOGY

COM/MON example in Open AR

- The model of the Com-Mon is divided in several parts.
  - Domain, that is used to type variables (state or flow variables).
  - Classes representing components,
  - the main block, corresponding to the entry point to the Com-Mon example.

FRENCH INSTITUTES OF TECHNOLOGY

Table of contents

FRENCH INSTITUTES OF TECHNOLOGY

COM/MON example in Open AR

Domain definition

```
domain FailureMode {OK, LOST, ERR}
// OK - normal behavior
// ERR - the sensor produces erroneous data
// LOST - the sensor produces no data
```

The domain is used to type variables (state or flow variables)

COM/MON example in Open AR

Class definition: the sensor

```
class Sensor
    // definition of the state variable
    FailureMode _mode (init = OK);
    // definition of the output flow variable
    FailureMode output (reset = LOST);
    // definition of events
    event failureLoss (delay = exponential(1.0E-4));
    event failureErr (delay = exponential(1.0E-5));
    // definition of transitions
    transition
      failureLoss:(_mode == OK) -> _mode := LOST;
      failureErr: (_mode == OK) -> _mode := ERR;
    // definition of the assertion
    assertion
        output := _mode;
end
```

The classes represent the components that would be instanciated in the main block.

COM/MON example in Open AR

Class definition: the contactor

```
class Contactor
    // definition of flow variables
    FailureMode input, output (reset = LOST);
    Boolean closeCondition (reset = false);
    // definition of the state variable
    Boolean _open (init = false);
    // definition of the event
    event openCT (delay = Dirac(0.0)) ;
    // definition of the transition
    transition
        openCT: not _open and not closeCondition -> _open := true;
    // definition of the assertion
    assertion
        output := switch {
                        case _open : LOST
                        default : input};
end
```

The classes represent the components that would be instanciated in the main block.

FRENCH INSTITUTES OF TECHNOLOGY

COM/MON example in Open AR

Class definition: the comparator

```
class Comparator
    // definition of flow variables
    FailureMode input1, input2 (reset = LOST);
    Boolean output (reset = false);
    // definition of the state variable
    Boolean _working (init = true);
    // definition of the event
    event failure (delay = exponential(1.0e-5));
    // definition of the transition
    transition
        failure: _working -> _working := false;
    // definition of the assertion
    assertion
        output := if _working then (input1 == input2) else true;
end
```

The classes represent the components that would be instanciated in the main block.

COM/MON example in Open AR

Block definition: the COM / MON

```
block ComMon
    // components of the Com-Mon
    Sensor F1, F2;
    Comparator Cmp;
    Contactor Ct;
    // definition of connections between components
    assertion
        Ct.input := F1.output;
        Ct.closeCondition := Cmp.output;
        Cmp.input1 := F1.output;
        Cmp.input2 := F2.output;
    // definition of failure conditions
    observer Boolean FC_B1 = (Ct.output == ERR);
    observer Boolean FC_B2 = (Ct.output == LOST);
end
```

The block is where the above classes are instanciated.

FRENCH INSTITUTES OF TECHNOLOGY

Table of contents

- How to get started with MBSA
  - Main principles
  - Questions to address before starting
  - The different steps to follow

- Modelling the example
  - Go Through the tool – Open AltaRica
  - How do I do in practice to model
  - How do I do in practice to simulate
  - How do I do in practice to compute

FRENCH INSTITUTES OF TECHNOLOGY

COM/MON example in Open AR: simulation and computation

Within the OpenAltaRica platform, this AltaRica 3.0 model of the Com-Mon example is assessed by using the generator of critical sequences. There are two parts to realize this assessment: a first one compiling the mode, and a second one realizing the generation of the critical sequences

Table of contents

FRENCH INSTITUTES OF TECHNOLOGY

COM/MON example in Open AR: compilation

```
domain FailureMode {OK, LOST, ERR}
 block ComMon
  Boolean Cmp._working (init = true);
  FailureMode Cmp.input1 (reset = LOST);
  FailureMode Cmp.input2 (reset = LOST);
  Boolean Cmp.output (reset = false);
  Boolean Ct._open (init = false);
  Boolean Ct.closeCondition (reset = false);
  FailureMode Ct.input (reset = LOST);
  FailureMode Ct.output (reset = LOST);
  FailureMode F1._mode (init = OK);
  FailureMode F1.output (reset = LOST);
  FailureMode F2._mode (init = OK);
  FailureMode F2.output (reset = LOST);
  event Cmp.failure (delay = exponential(1e-05));
  event Ct.openCT (delay = Dirac(0.0));
  event F1.failureErr (delay = exponential(1e-05));
  event F1.failureLoss (delay = exponential(0.0001));
  event F2.failureErr (delay = exponential(1e-05));
  event F2.failureLoss (delay = exponential(0.0001));
  observer Boolean FC_B1 = Ct.output == ERR;
  observer Boolean FC_B2 = Ct.output == LOST;
transition
    Cmp.failure: Cmp._working -> Cmp._working := false;
    Ct.openCT: not Ct._open and not Ct.closeCondition
              -> Ct._open := true;
    F1.failureLoss: F1._mode == OK -> F1._mode := LOST;
    F1.failureErr: F1._mode == OK -> F1._mode := ERR;
    F2.failureLoss: F2._mode == OK -> F2._mode := LOST;
    F2.failureErr: F2._mode == OK -> F2._mode := ERR;
  assertion
    Cmp.output :=  if Cmp._working then (Cmp.input1 == Cmp.input2)
                   else true;
    Ct.output :=  if Ct._open then LOST else Ct.input;
    F1.output := F1._mode;
    F2.output := F2._mode;
    Ct.input := F1.output;
    Ct.closeCondition := Cmp.output;
    Cmp.input1 := F1.output;
    Cmp.input2 := F2.output;
end
```

COM/MON example in Open AR: compilation

Generation of critical sequences

```
Cmp.failure F1.failureErr
Cmp.failure F2.failureLoss F1.failureErr
Cmp.failure F2.failureErr F1.failureErr

Cmp.failure F1.failureLoss
Cmp.failure F2.failureLoss F1.failureLoss
Cmp.failure F2.failureErr F1.failureLoss
F1.failureLoss Ct.openCT
F1.failureErr Ct.openCT
F2.failureLoss Ct.openCT
F2.failureErr Ct.openCT
```

Two computations are realized: a first one to get all sequences of events leading to the value ‘true’ of the observer ‘FC_B1’,, and a second one to get all sequences of events leading to the value ‘true’ of the observer ‘FC_B2’

- The result of the code provides the critical sequences for the COM / MON example

FRENCH INSTITUTES OF TECHNOLOGY