



# SATODEV

SAFETY TOOLS DEVELOPMENT

## BOOLEAN LIBRARY [BL 1.N]

## MODELING RULES

## CECILIA 6.2.X

Document Version BL\_1\_6.2-A

(issue A)



## TABLE OF CONTENTS

---

<b>1. SUBJECT</b> .....	<b>3</b>
<b>2. REVISIONS</b> .....	<b>3</b>
<b>3. FIELD OF APPLICATION</b> .....	<b>3</b>
<b>4. GENERAL</b> .....	<b>3</b>
3.1 NAMING .....	3
3.2 COLORS .....	4
3.3 ICONS.....	4
3.4 LAYERS .....	5
<b>4. TYPES</b> .....	<b>5</b>
4.1 DIRECTORIES.....	5
4.2 NAMING .....	5
4.3 SPECIFICATION .....	6
4.4 CHECKLIST .....	6
<b>5. OPERATORS</b> .....	<b>7</b>
5.1 DIRECTORIES.....	7
5.2 NAMING .....	7
5.3 SPECIFICATION .....	7
5.4 CHECKLIST .....	7
<b>6. COMPONENTS</b> .....	<b>8</b>
6.1 DIRECTORIES.....	8
6.2 NAMING .....	8
6.3 SPECIFICATION .....	8
6.4 CHECKLIST .....	9
<b>7. EQUIPMENT</b> .....	<b>9</b>
7.1 DIRECTORIES.....	9
7.2 NAMING .....	10
7.3 SPECIFICATION .....	10
7.4 CHECKLIST .....	10
<b>8. PROJECTS MODELS</b> .....	<b>11</b>
8.1 DIRECTORIES.....	11
8.2 SPECIFICATION .....	11
8.3 EXAMPLE .....	11



## 1. SUBJECT

---

This document gathers a set of modeling rules that have been followed when building the Boolean library [BL 1.n] within Cecilia Workshop version 6.2.x.

These rules cover the organization of directories, naming, colors, etc., for the various library elements, as well as how to specify and verify them.

The aim of applying these rules is not only to enable collaborative work on the same project, to enrich the components library, but also to be able to maintain models over time.

## 2. REVISIONS

---

Version	Date	Updates
BL_1_6.2-A	22/04/2024	Initial version

## 3. FIELD OF APPLICATION

---

These modeling rules apply to version 6.2.x of the Dassault Aviation tool, Cecilia Workshop, whatever "x". They concern [BL 1.n] Boolean libraries, whatever "n".

## 4. GENERAL

---

### 3.1 Naming

Naming is an important part of the modeling process. Explicitly naming the various items used has several advantages:

- quickly identify the content or the function of an item
- make it easy the reading of the results
- enable others to understand the model more quickly

Allowed characters: lowercase letters, uppercase letters, underscore "\_", numbers (not allowed at the beginning of a word). Spaces, special characters and accents are not allowed, and avoid too long names!

Names should be as readable as possible:





Avoid:           communicationsystem

Prefer:          CommunicationSystem or Communication\_System



## 3.2 Colors



Here is the list of the colors used:

Color	RGB	HTML code	
Green	0, 255, 0	# 00FF00	
Red	255, 0, 0	# FF0000	
Orange	255, 153, 0	# FF9900	
Grey	204, 204, 204	# CCCC00	

Other colors can be used as needed.

### 3.2.1 Boolean type colors (Flows)





Default colors for Boolean flows are the following:

"Boolean" flows	
true	
false	

These colors can be overloaded in each simulable model (Models in the "Projects" tab), in the "Links colors" sub-tab.

### 3.2.2 Components and equipment colors

For components and equipment, the color reflects their intrinsic integrity.

States	Component	Equipment	
Nominal	nominal behavior*	all components: Nominal	
Failure	no longer works or misleading behavior*	all components: Failure	
Degraded	-	at least one component: not Nominal	
Off	-	no power supply	

\* due to a failure mode (random failure or development error) or the power supply

## 3.3 Icons

The size of the icons determines the size of the "simulation" mode windows. Thus, special attention must be paid to them.

Taking into account the size and the orientation of the displays, the following distribution of inputs/outputs is recommended (left-right reading):

- Inputs on the left
- Outputs on the right
- ElecPower input at bottom left (close to the small lightning)
- State output at top right

The name of the item may be above, below or even inside.



The following naming is recommended:

*Edition mode:*

- ExplicitName\_0 no color

*Simulation mode:*

- ExplicitName\_1 green
- ExplicitName\_2 red
- ExplicitName\_3 orange
- ExplicitName\_4 grey

### 3.4 Layers

To facilitate the presentation of models, items can be assigned to different layers, for example:

Layer 1:	Architecture	items and logical links
Layer 2:	ElecPower	electrical power supply links
Layer 3:	HydrauPower	hydraulic power supply links
Layer 4:	Functional	logics and links between organic view and functional view
Layer 5:	Init	initialization artefacts and links
Layer 6:	States	state artefacts and links
Layer 7:	Graphics	graphic items (lines, shapes, images, text, etc.)

## 4. TYPES

---

There are 2 categories of types: enumerate and record. The “enumerate” types represent various flows that will link the different items. A “record” type gathers several “enumerate” types.

### 4.1 Directories

Types are distributed in families and sub-families as follows:

#### **Frmk\_Boolean**

<b>Bus</b>	record of Boolean flows and states
<b>States</b>	states of different items (components, equipment...)

### 4.2 Naming

Types globally follow the hereafter naming:

<b>Bus:</b>	<i>Bus<sub>nn</sub></i>	nn: the number of fields
	<i>Example:</i>	Bus_05 is a bus of 5 Boolean flows
<b>States:</b>	<i>NameState</i>	Name: the type of item
	<i>Example:</i>	ComponentState, EquipmentState



### 4.3 Specification

The specification of each type is defined in the “Properties” tab. The template is the following:

#### Enumerate type:

[description]

What the type represents.

\*\*\*\*\*

[domain]

Value1: definition

Value2: definition

Value3: definition

\*\*\*\*\*

Copyright if any.

\*\*\*\*\*

#### Record type:

[description]

What the type represents.

\*\*\*\*\*

Copyright if any.

\*\*\*\*\*

Even if their definition varies from one flow to the other, the following domains are mainly used:

#### Frmk\_Boolean/States/EquipmentState

Nominal	:	all components are in a Nominal state
Failure	:	all components are in a Failure state
Degraded	:	at least one component is not in a Nominal state
Off	:	no electrical power supply (or switched off)

### 4.4 Checklist

Here is a checklist of the various steps involved in creating a "Type" item:

<b>Enumerate type</b>	Does the item already exist?
	Create the item in the right directory
	<i>Enumerate choice</i>
	Name the item
	General tab: name the values of the domain
	General tab: define the default colors for each value
	Properties tab: write the specification of the item
	Save



<b>Record type</b>	Does the item already exist?
	Create the item in the right directory
	<i>Record</i> choice
	Name the item
	General tab: define the fields (name, type, orientation)
	General tab: define the default link to display
	Properties tab: write the specification of the item
	Save

## 5. OPERATORS

---

Operators are logical functions (in the mathematical/computer sense) that manipulate the values of different flows.

### 5.1 Directories

Operators are distributed in families and sub-families as follows:

**Frmk\_Boolean**  
**Icon**

### 5.2 Naming

The prefix "op\_" is used for the operators, followed by an explicit description, as far as possible.  
*Example:* op\_Equipment\_Icon

### 5.3 Specification

The specification of operators is defined in the "Properties" tab. The template is the following:

[description]

What is the global function of the operator.

\*\*\*\*\*

[logic]

What logic is implemented, in literary format.

\*\*\*\*\*

Copyright if any.

\*\*\*\*\*

### 5.4 Checklist

Here is a checklist of the various steps involved in creating an "Operator" item:

<b>Operator</b>	Does the item already exist?
	Create the item in the right directory
	Name the item
	General tab: choose the type of the operator
	General tab: name the operands and their types
	Properties tab: write the specification of the item
	Altarica code tab: write and comment the code in AltaRica language
	Syntax and Consistency checks
	Save



## 6. COMPONENTS

---

There are 2 categories components:

- basic blocks which can embed states, events to switch from one state to the other (that can represent failure modes, functional actions...) and icons (to reflect the states in simulation mode). These blocks can be used to create equipment.
- graphical operators, which do not embed any event but only logical behaviors

The components are the only items that can embed events.

### 6.1 Directories

Components are distributed in families and sub-families as follows:

#### Frmk\_Boolean

<b>DynamicGates</b>	logical gates for which depending on events sequences
<b>Items</b>	basic blocks related to the organic items
<b>LogicalGates</b>	classical logical gates (and, or, not, xor...)
<b>Misc</b>	anything else

### 6.2 Naming

The components globally follow the hereafter naming:

Name\_NumberOfOperands

*Example:* OR\_2, AND\_3, K2\_out\_of\_M3...

Inputs / Outputs / Local variables / Events:

For components inputs, use the “i” prefix and for outputs, use the “o” prefix. The rest of the word may be generic or refer to the item from which it comes / to which it goes.

*Examples:* iElecPower, i1, oData, oState...

For local variables, use the “l” prefix, and for events, use the “e” prefix.

*Examples:* lState, eFailure...

### 6.3 Specification

The specification of components is defined in the “General” tab, Properties/Comment part. For the basic blocks components, the template is the following:

[description]

What does the component represent.

\*\*\*\*\*

[behavior]

What is the behavior of the component, depending on its events (failure modes, functional action, zonal threats...), in literary format.

\*\*\*\*\*

[assumption]

Assumptions that have been made.

\*\*\*\*\*





[remark]

For which purpose can it be used.

\*\*\*\*\*

Copyright if any

\*\*\*\*\*

For the graphical operators, the template is close to the template of operators:

[description]

What is the global function of the graphical operator.

\*\*\*\*\*

[logic]

What logic is implemented, in literary format.

The truth table can be entered here.

\*\*\*\*\*

[remark]

For which purpose can it be used.

\*\*\*\*\*

Copyright if any.

\*\*\*\*\*

## 6.4 Checklist

Here is a checklist of the various steps involved in creating a "Component" item:

<b>Component</b>	Does the item already exist?
	Create the item in the right directory
	Name the item
	General tab: choose the icon in edition mode and define its size
	General tab, Properties part: write the specification of the item
	I/O tab: define inputs/outputs/local variables (name, type, position)
	States tab: define the different states, their type and the default value
	Events tab: define the failure modes, functional actions, zonal threats...
	Icons: define the icons in simulation mode (pay attention to their order) The colors are defined in §3.2.2.
	Altarica code tab: write and comment the code in AltaRica language
	Syntax and Consistency checks
	Save

## 7. EQUIPMENT

The equipment are, according to the Cecilia's definition, a set of components and/or equipment. Therefore, they can either represent a part of a real equipment, a real equipment, a sub-system, a system, the integration of several systems or a view.

### 7.1 Directories

Equipment are distributed in families and sub-families as follows:

#### Frmk\_Boolean

**Common**

equipment shared by some domains

**Communication, Avionics...**

equipment specific to certain domains



## 7.2 Naming

The naming of the equipment should be as explicit as possible. The suffixes “system” and “views” are used when necessary.

*Examples:* Display, Calculator, ControlPanel, Battery, ElectricalSystem, OrganicView...

The naming of inputs, outputs and local variables is identical to that of components (§6.2). The naming of layers is defined in §3.4.

## 7.3 Specification

The specification of equipment is defined in the “General” tab, Properties/Comment part. The template is the following:

[description]

What does the equipment represent.

\*\*\*\*\*

[behavior]

What is the high-level behavior of the equipment, depending on its components and on zonal threats.

\*\*\*\*\*

[assumption]

Assumptions that have been made.

\*\*\*\*\*

[remark]

For which purpose can it be used.

\*\*\*\*\*

Copyright if any.

\*\*\*\*\*

## 7.4 Checklist

Here is a checklist of the various steps involved in creating an "Equipment" item:

<b>Equipment</b>	Does the item already exist?
	Create the item in the right directory
	Name the item
	General tab: choose the icon in edition mode and define its size
	General tab, Properties part: write the specification of the item
	I/O tab: define inputs/outputs/local variables (name, type, position)
	Content tab: drag & drop the components and/or equipment, name and link them
	Synchronizations tab: define the synchronizations, if any
	Icons: define the icons in simulation mode (pay attention to their order) The colors are defined in §3.2.2.
	Altarica code tab: write and comment the code in AltaRica language
	Syntax and Consistency checks
	Save



## 8. PROJECTS MODELS

---

The “projects models” are the only items that can be simulated. It can be either to validate components/equipment or to simulate a complete modeling (several views and/or systems).

The computations (Boolean equations generation or sequence generation) can only be made on these projects models.

### 8.1 Directories

In the hierarchy, the first level is called Project, the second level is called System. Different items can then created (Model, Tree, DSF and FMEA). The following deals only with Models.

Two projects are present by default, the **Bench** project to test and validate the components/equipment and the **Boolean** project that an example.

### 8.2 Specification

The specification of a project model is defined in the “Properties” tab. The template is the following:

[description]

What does the model represent, what is the context.

\*\*\*\*\*

[library version]

Version of the library that has been used.

\*\*\*\*\*

[model version]

Version of the model (number, date...) and differences with the reference version.

\*\*\*\*\*

[model specification]

Reference of the excel file which embeds the model specification.

\*\*\*\*\*

[to-do list]

List of ongoing activities or still to be done.

\*\*\*\*\*

[remark]

Any other information.

\*\*\*\*\*

Copyright if any.

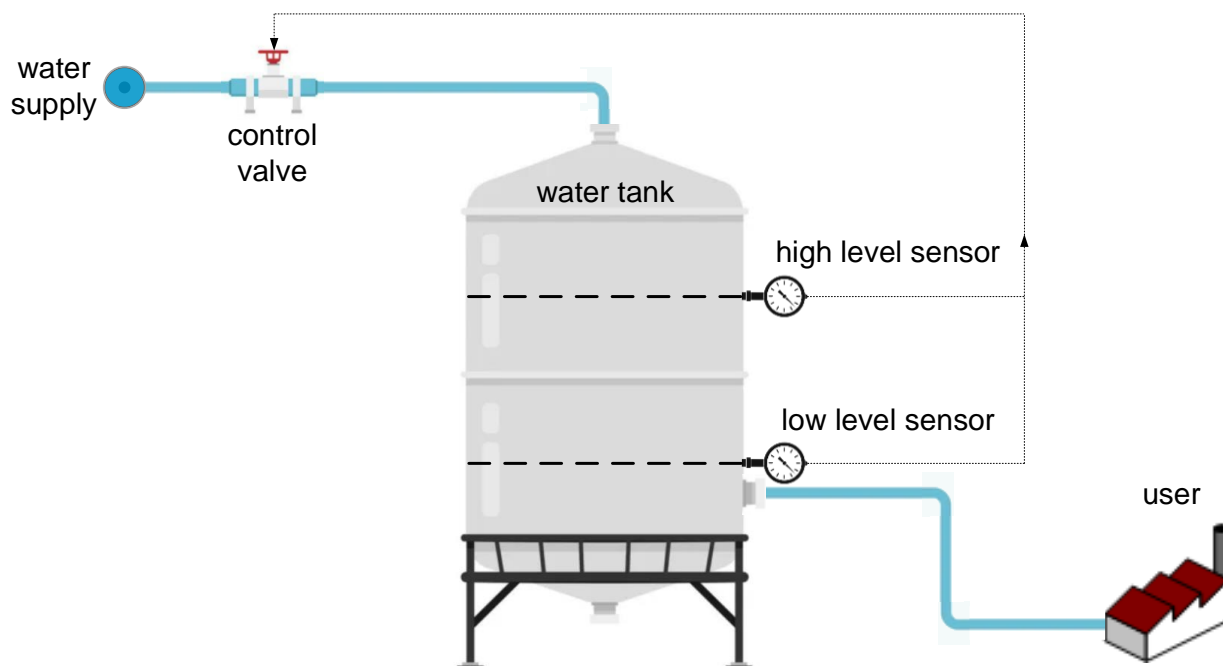
\*\*\*\*\*

### 8.3 Example

The example deals with a water tank which distributes water from a water supply to users. The feared events are:

- Tank overflow: critical  
*People safety concerns and significant damages to installations arounds the tank*
- Water service interruption: major  
*Economic consequences*





Water tank example

The system is composed of 2 gauges (low level and high level sensors) that command a control valve.

When the water reaches the high level, a closing command is sent to the control valve. Underneath, an opening command is sent.

When the water is under the low level, an opening command is sent to the control valve. Overhead, a closing command is sent. Tank

The 2 sensors shall send the same command to the control valve, otherwise there is no effect.

The water supply, the water tank and the user do not have failure modes.  
Control valve has 2 failure modes: *stuck\_open* and *stuck\_closed*.  
Level sensors have 1 failure mode: *bad\_detection*.

