BPA SD9 Dysfunctional Analysis and Simulation

User Guide

BPA Dysfunctional Analysis and Simulation (SD9)
BPA Delivery 7 for V5R19 (V5.7)

DASSAULT SYSTEMES

Table of Contents

INTRODUCTION

This user's manual describes the workbench tool DAS which provides the main following features:

- Modelling functional and dysfunctional behaviour of systems by means of reusable components from library,
- Designing system architectures by reusing components and equipments from library,
- Interactive and graphical simulation of system architectures,
- Verification of the consistency of the behaviour,
- Automatic generation of fault trees by selecting an unexpected event from the model,
- Automatic generation of failure scenarios leading to a selected unexpected event for dynamic system models.

Appendix 1 describes the Syntax of Altarica Language.

Appendix 2 describes the Syntax of probability laws in Aralia format.

APPENDIX REFERENCES

- SD9_Appendix_ABC_R19_D7.pdf
- SD9_Appendix_SeqGen_R19_D7.pdf
- SD9_Appendix_Translator_R19_D7.pdf
- SD9_Appendix_Utility_R19_D7.pdf
- SD9_Appendix_Simulation_R19_D7.pdf
- SD9_Appendix_FRB_R19_D7.pdf
- SD9_Appendix_ Attributes_Named_Parameters_R19_D7.pdf

HARDWARE ARCHITECTURE

SYSTEM REQUIREMENTS

Minimal configuration:

- Any PC platform with, at least:
  - Processor: Pentium IV 2 Ghz
  - Hard disk: 50 MB free disk space,
  - RAM : 512 Mb Minimum (1 Gb recommended),
  - Graphics resolution: 1024 x 768 pixels, with a palette of 65536 colours or above,
  - Windows XP x32

INSTALLATION

Cf. Installation guide.

LAUNCHING DAS

Launch DAS from the shortcut added on the desktop after the installation, or from the shortcut in the start menu. The following windows are displayed:

- An initialization MS DOS (Microsoft Disk Operating System) DAS window (Figure 1).

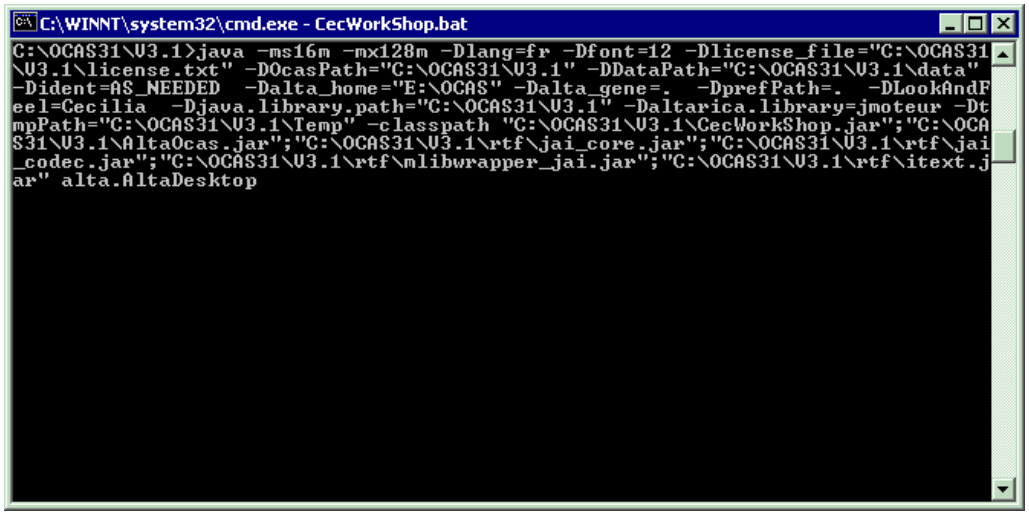


Figure 1 : Command window

- A launching window (Figure 2).

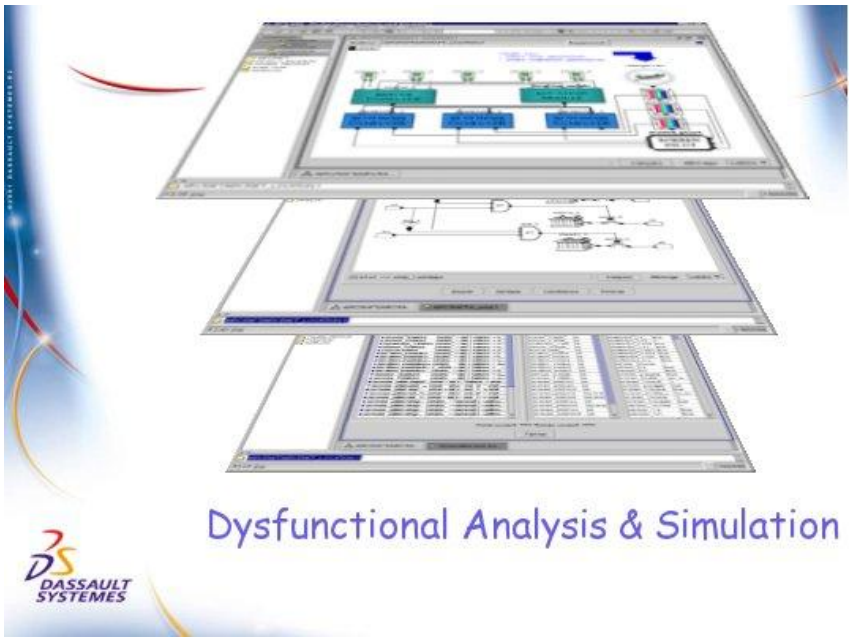


Figure 2 : launching window

Now, the user name and the password are required (Figure 3). The user rights depend on the selected database (Access © or Oracle ©).

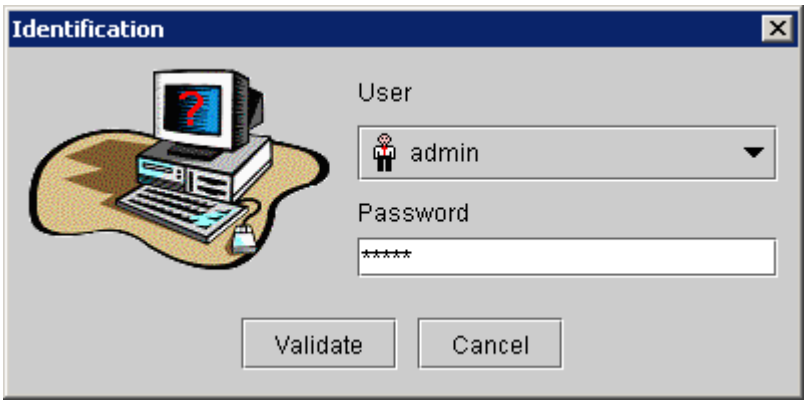


Figure 3 : Login window - user identification

Note: Each user must be registered by a user having administrator rights in the Access© or Oracle© databases.

USER MANAGEMENT

ADMINISTRATION WINDOW

The user management window is available only for the administrator user or any user having administrator rights.

Each administrator can manage access rights for DAS users:

- New user creation,
- Definition of user profile (user, administrator),
- Creation of User group.

Access on data (components, equipments, types, systems, operators) depends on user rights.

Select **Administration** command from the menu **File** (Figure 4) to open the administration console:

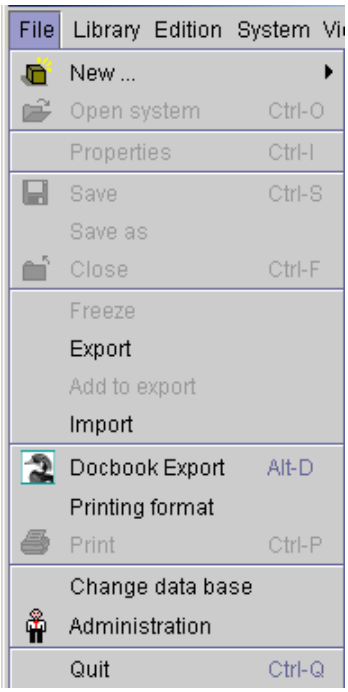


Figure 4 : Menu File

USER MANAGEMENT

User management is available through the tab **User** in ***User Administration*** window (Figure 5).

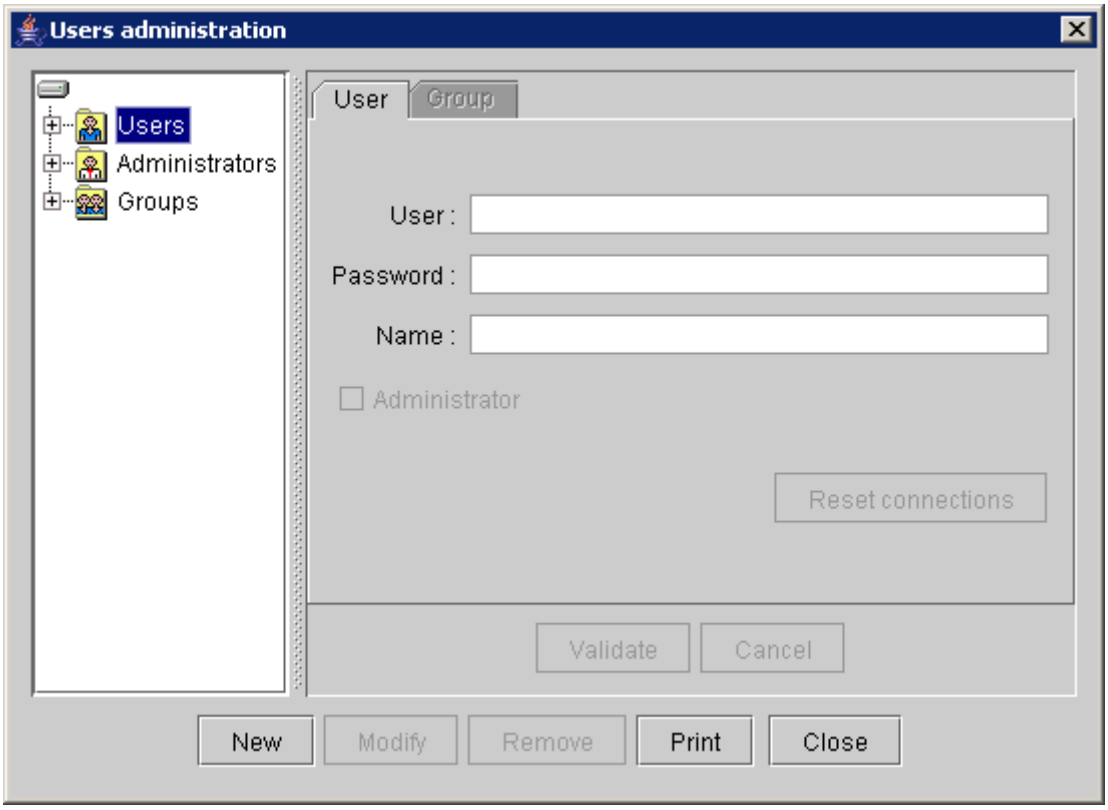


Figure 5 : Users Administration window – User tab

To create a new user :

- Select **Users** or **Administrators** in the left part of the window,
- Click on button **New**,
- Enter the logging name of the new user in the field **User**,
- Choose a password for the new user in the field **Password**,
- Enter a user name in the field **Name**,
- Tick the box **Administrator** to allow this user to access user administration functions (Administrator privileges).
- Click on button **Validate** to validate creation of the new user profile, or on button **Cancel** to cancel the creation of this profile.

When the user is created, he is inserted in the Administration tree structure, either in the ***Users*** *list* as a common user (Figure 6), or in ***Administrators*** list if he benefits of administrator privileges.

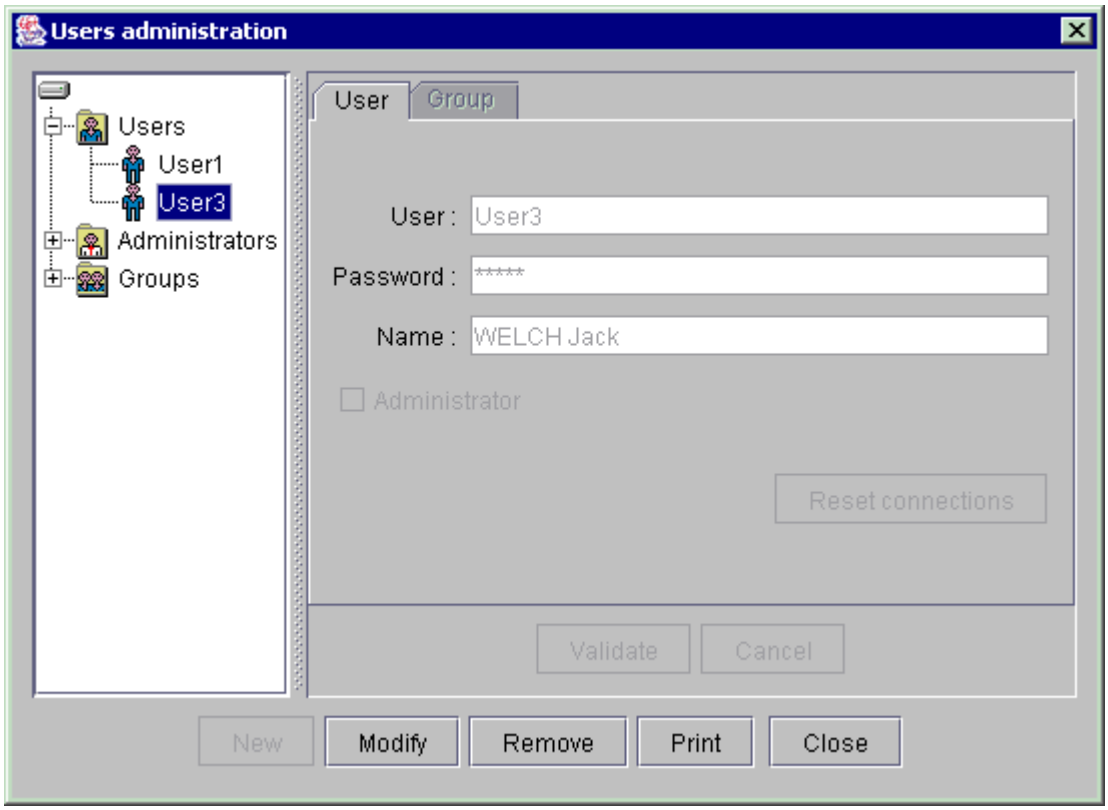


Figure 6 : Creation of a new user

To modify an existing user:

- Select (left click) in ***Users*** or ***Administrators*** list, the user to modify,
- Click on button **Modify**,
- Make all the necessary adjustments on user profile (User, password, name, rights),
- Click on button **Validate** to validate the modifications, or on button **Cancel** to cancel modifications on this profile.

To remove an existing user:

- Select (left click) user to be removed from the ***Users*** or ***Administrators*** list.
- Click on button **Remove**, the following window is displayed (Figure 7):

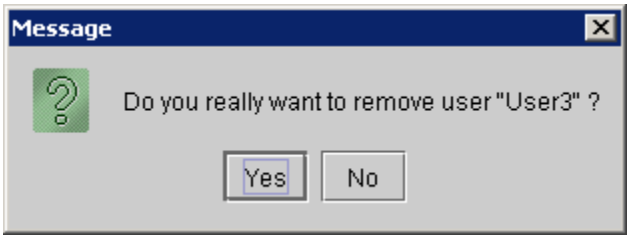


Figure 7 : Message for removing a user

- Click on button **Yes** to delete the user profile or on button **No** to cancel the destruction.
- If the removed user is the owner of a DAS Object (models, systems, etc …), the administrator must specify the user who will be the new owner of this object. During removal, a dialog box (Figure 8) asks Administrator for the new owner.

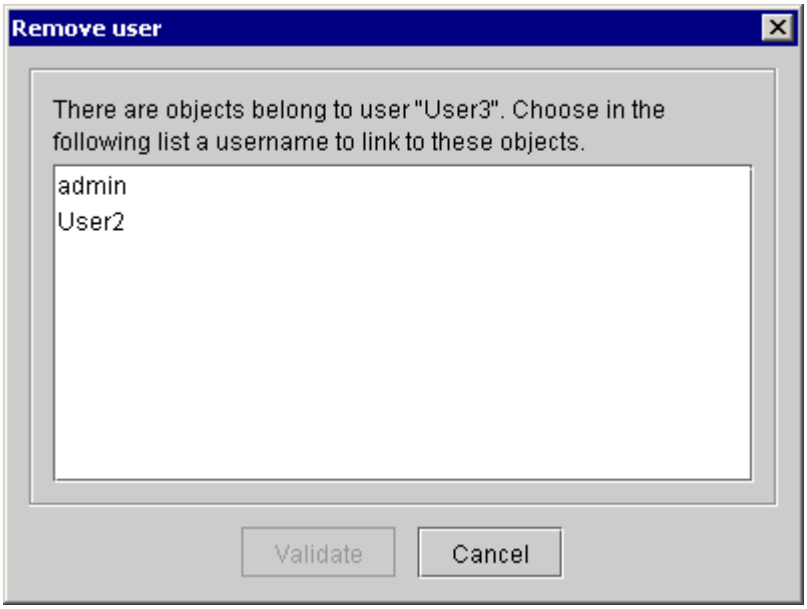


Figure 8 : New Owner

Note: Only the ***Master Administrator*** profile created during the initialization of DAS database (Access ©, Oracle ©) can not be deleted.

WORKGROUP MANAGEMENT

The user workgroup management is available through the tab **Groups** from the ***Users administration*** window (Figure 9).

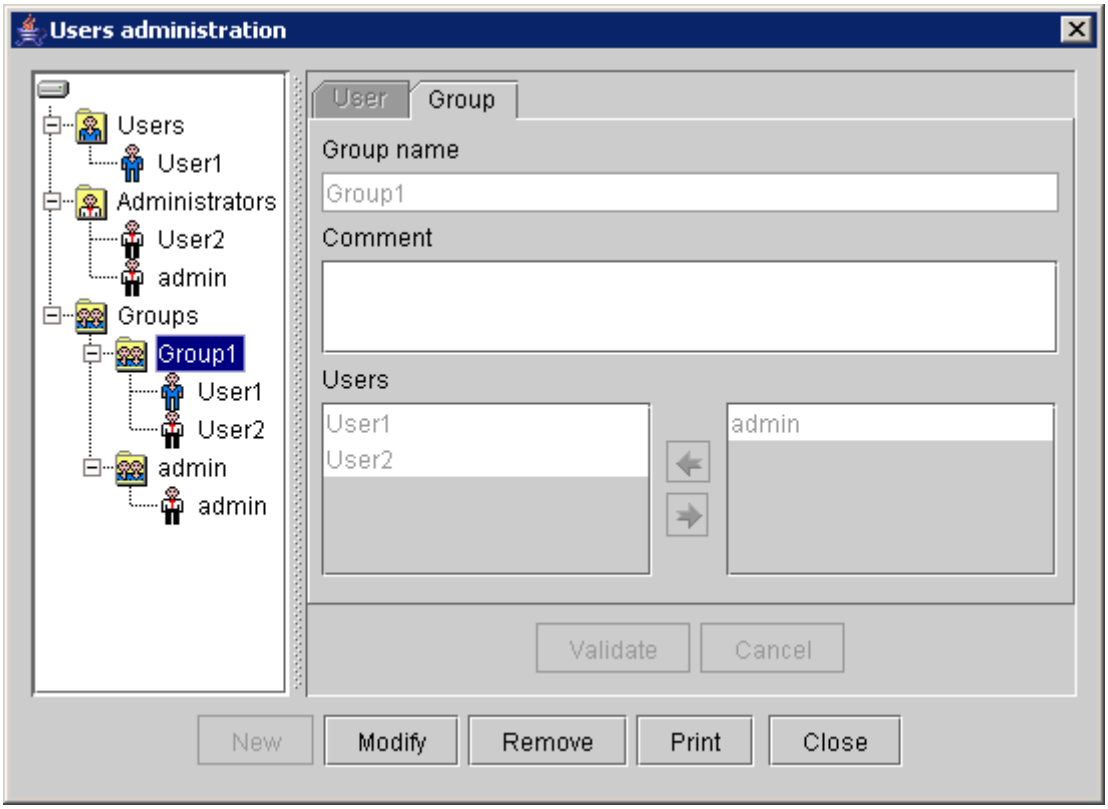


Figure 9 : Users administration window - Tab Groups

To create a new workgroup:

- Click on button **New**,
- Enter a group name in the field **Group name**,
- In the tab **Users**, select one by one users belonging to this group. Use the arrow ⬅ to move on the left list,
- Enter if required a comment for this group in the field *Comment*,
- Click on button **Validate** to validate the creation of this group, or on button **Cancel** to cancel the creation.
- When the group is created, it is displayed in the ***Groups*** list (Figure 10).

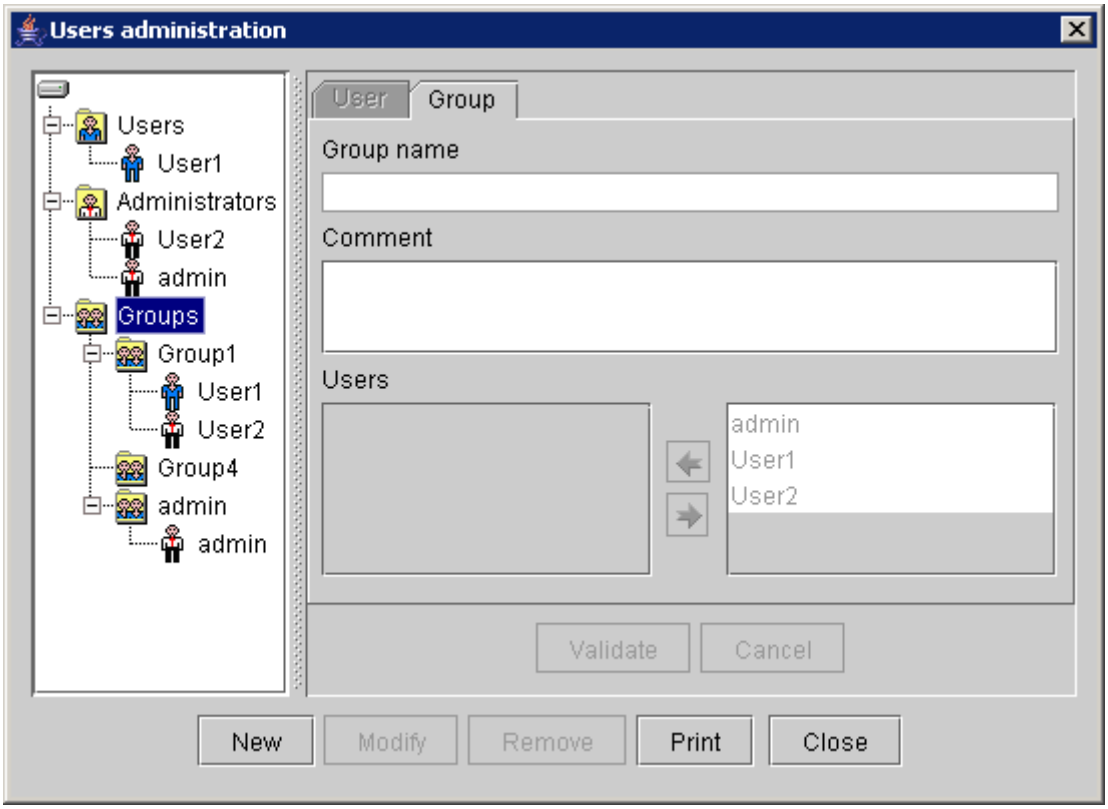


Figure 10 : Users group creation

To modify an existing group:

- Select the group to modify in the **Groups** list (left click),
- Click on button **Modify**,
- Make the required adjustments on this group. To remove a user, select in ***Users*** area the user in the left list and use the arrow ➡ to move it in right list. To add a user, proceed as described for the creation of a group,
- Click on button **Validate** to validate the modifications, or on button **Cancel** to cancel the modification on this group.

To delete an existing group:

- Select the group to be deleted in the **Groups** list (click left),
- Click on button **Remove**, the following window is displayed (Figure 11) :

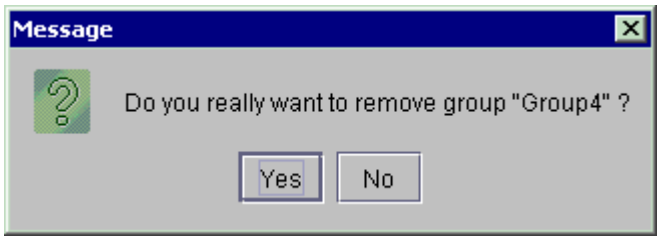


Figure 11 : message of the user group destruction

- Click on button **Yes** to valid, or on button **No** to cancel the deleting of this group.

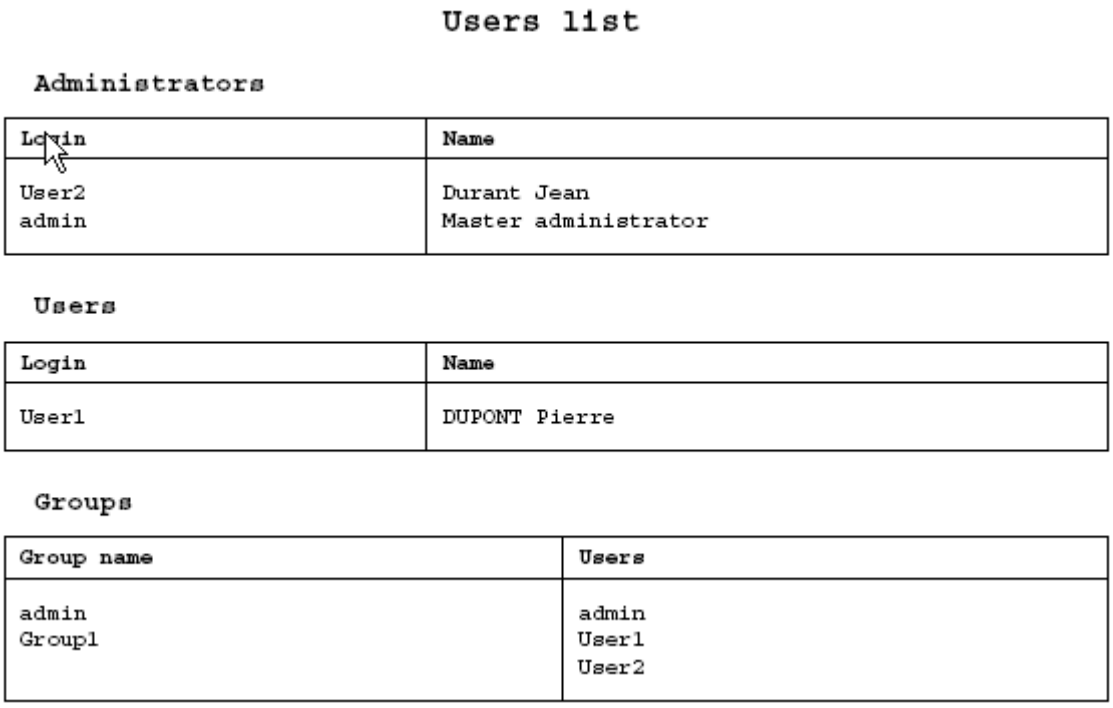
CAUTION: Only empty groups can be deleted.

Note: Only the ***Master Administrator*** profile created during the initialization of DAS database (Access ©, Oracle ©) can not be deleted.

PRINT ADMINISTRATION INFORMATION

The administrator can print information contained in the Administration database by clicking on button **Print** from the **User administration** window (Figure 12):

- The user list,
- The workgroups.



Users list

Administrators

Login	Name
User2 admin	Durant Jean Master administrator

Users

Login	Name
User1	DUPONT Pierre

Groups

Group name	Users
admin Group1	admin User1 User2

Figure 12 : Printing Administration information

MANAGEMENT OF DATA ACCESS RIGHT

DAS allows the access right management on data (generic models in library and System architectures) by dependencies and visibility attributes defined on each DAS object.

DEPENDANCIES

Any DAS object, generic model in library or system architecture, owns to the user who created it. If this user belongs to several Workgroups, this one must specify during the creation of a DAS object, the Workgroup to which this object must be linked.

Only the administrator of DAS Database or the owner of a given object can modify the dependencies by using the property window (Figure 12).

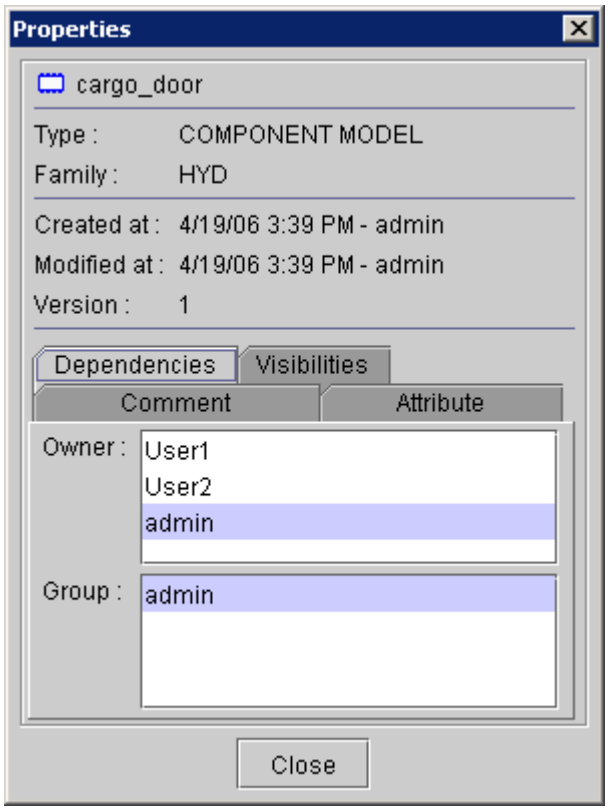


Figure 13 : Editor of Properties – Dependencies tab

To modify the owner of a DAS object:

- Select a DAS object (model or system) in the tree.
- Displays Property Editor with the File/Properties menu or with contextual menu.
- Click on the tab **Dependencies**,
- In the area ***Owner***, select the user who must be the new owner of this object.
- If the new owner of this object belongs to several Workgroups,

- In the ***Group*** area, select the workgroup to which this object must be linked
- Moreover, if the DAS object Is a project or a family, user can specify if the rights must be propagated to each included object with **Apply to included objects …**

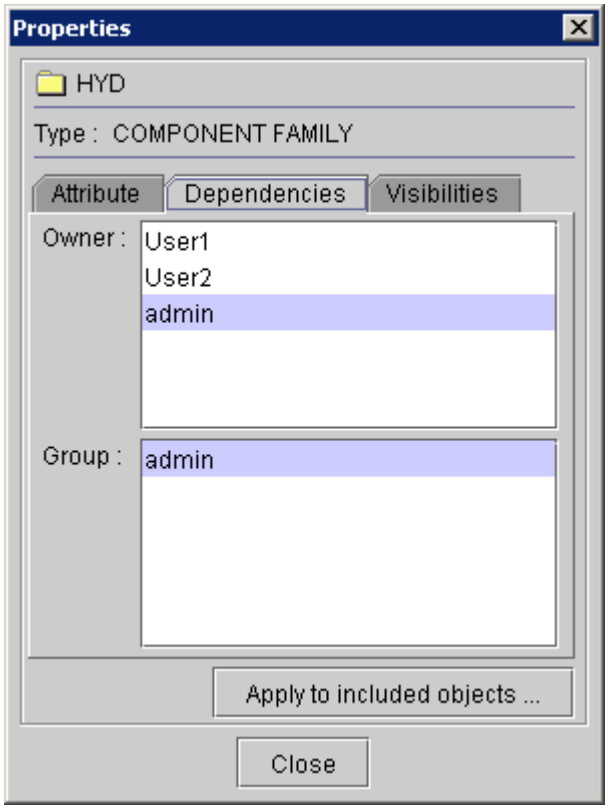


Figure 14 : Apply Dependencies to included objects

DATA ACCESS RIGHTS (ATTRIBUTES)

During the creation of a DAS object, the default access rights are:

- Read Access Right for users of the workgroup to which this object is linked (only the owner of this object has an Read/Write Access Right),
- No access for users of the other workgroups.

Only the administrator of DAS Database or the owner of a given object can modify its attributes of access rights by using the property editor (Figure 17).

- To modify the access rights to a DAS object for workgroups:
- Select the DAS Object (model or system architecture)
- Open the property editor by clicking on Properties in File Menu
- Click on the **Attribute** tab,
- In the ***Group access*** area, choose in the popup menu the new access rights for the workgroup to which this object is linked,
- In the ***Other access*** area, choose in the popup menu the new access rights for the other workgroups.

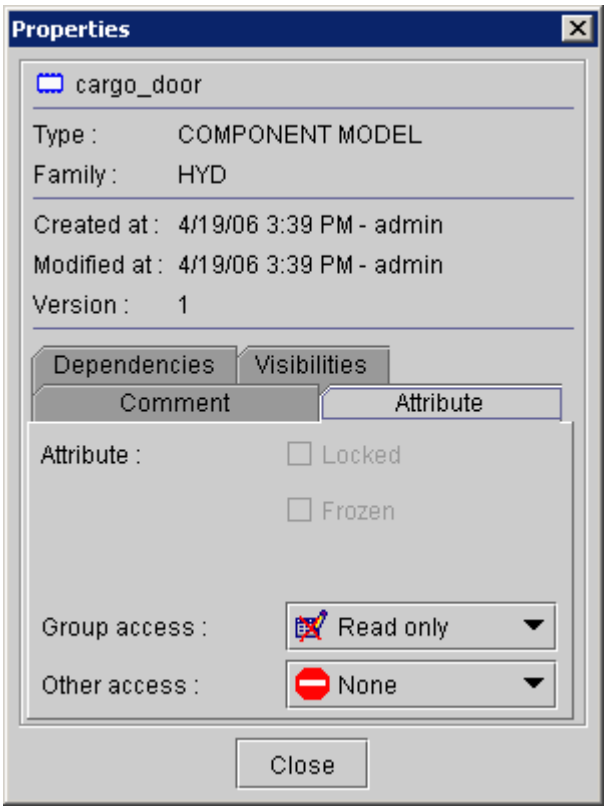


Figure 15 : Property edition - Attribute

- If you edit a Project or a Family of Models, you can specify if the rights must be applied to all included elements by clicking on Apply to included objects (Figure 16).

Figure 16 : Apply change to included Objects

VISIBILITY DEFINITION

The **Visibility** tab in **Properties** editor allows user to specify access rights to a DAS Object (generic models in library and system architectures) for each workgroup defined in the administration module (See § *Administration Window*). The access rights defined by default in the **Visibilities** tab are defined in the **Attribute/Dependencies** tabs.

Only the administrator of DAS database or the owner of a given object can modify attributes of access rights in the property editor (Figure 17).

To modify the access rights to a DAS object for a workgroup:

- Select the DAS Object (model or architecture)
- Launch the Editor of properties by clicking on Properties in File Menu
- Click on the **Visibilities** tab,
- In the Access Right area for this workgroup, select in the popup menu the new access rights for this workgroup (no access, read only, read/write).

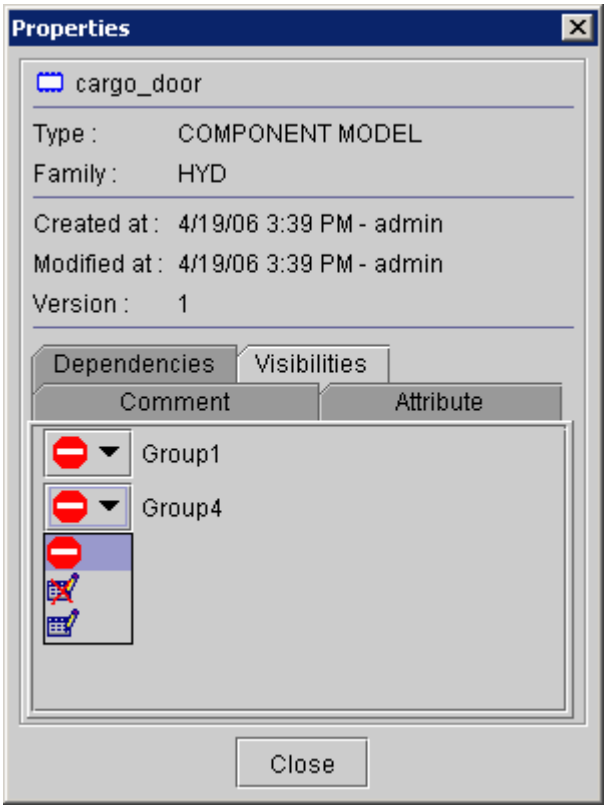
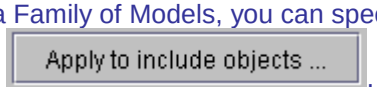


Figure 17 : Properties edition – Visibilities

- If you edit a Project or a Family of Models, you can specify if the visibilities must be applied to all included elements by clicking on 

.

CONFIGURATION EXAMPLE

The following diagram (Figure 18) displays:

- Users with their rights
- User Groups
- DAS model families and attributes (dependencies, rights, visibilities)

The definition of FAMILY_B attributes (Figure 18) associated with this configuration, are shown Figure 19.

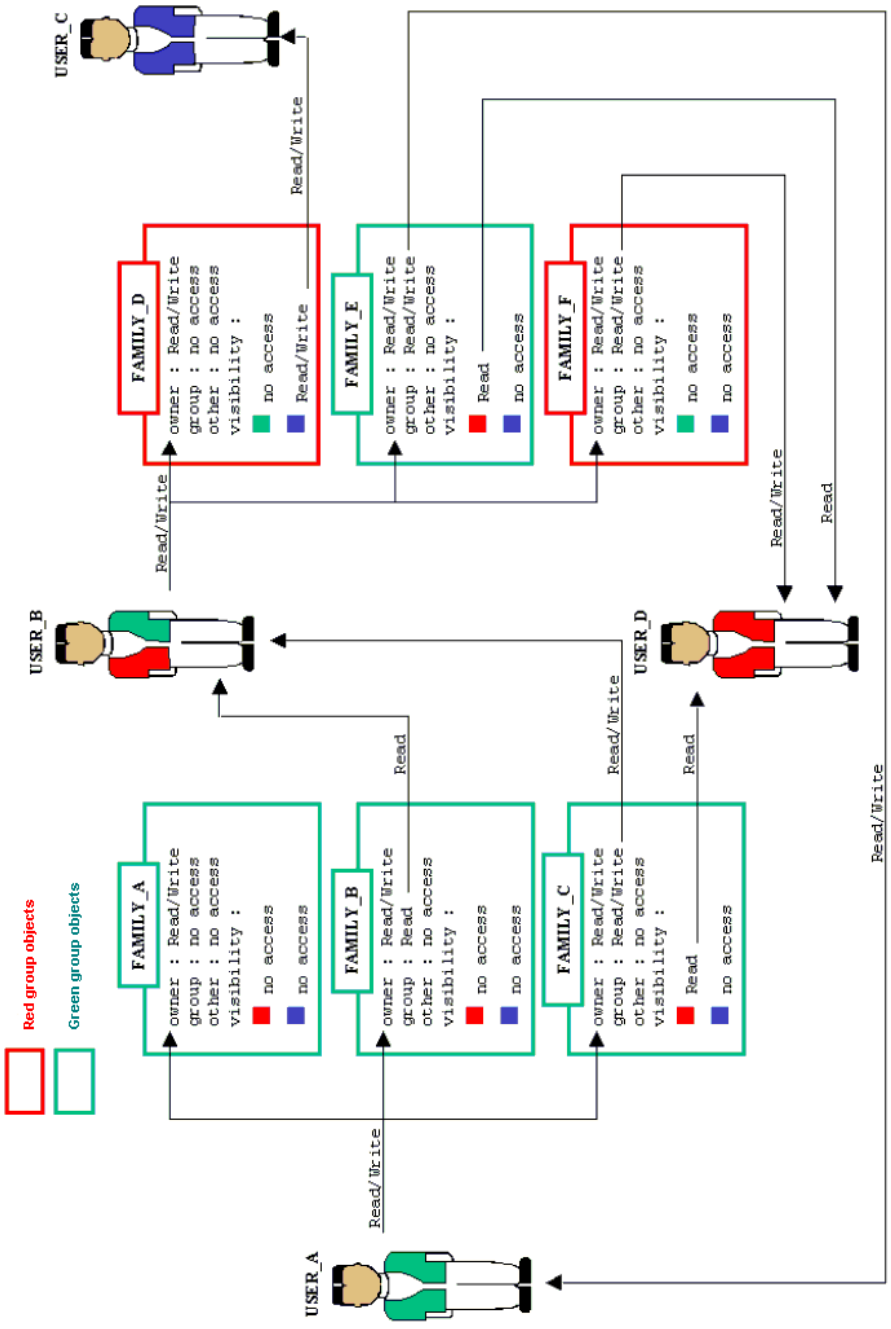


Figure 18 : Rights, Dependencies, Visibilities

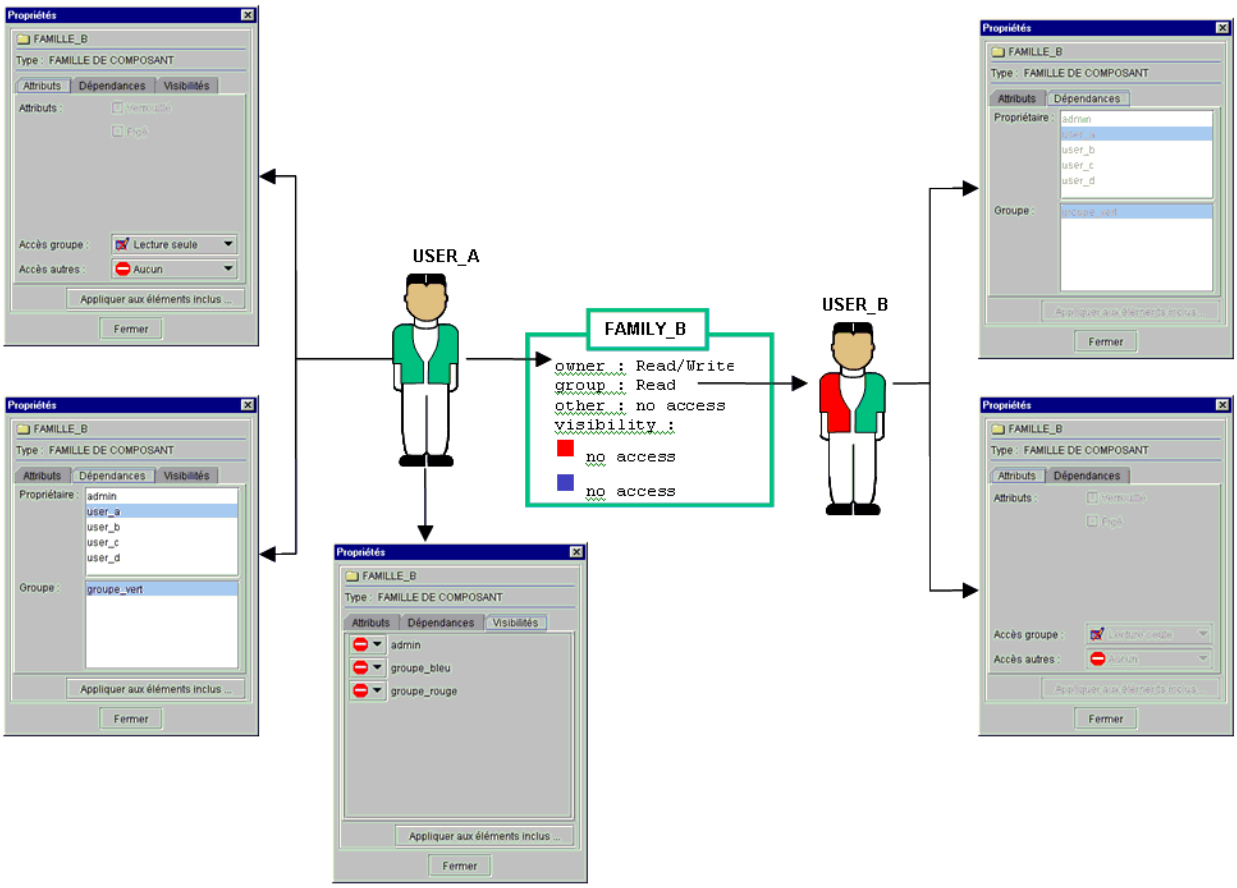


Figure 19 : Example: FAMILY_B

WORKBENCH PRESENTATION

The workbench of DAS is composed of seven parts:

- A title bar (1).
- A menu bar (2).
- A main tool bar (3) composed of the main tools.
- A work area (4) displaying the views and windows of opened models; the lower bar (5) indicates the opened objects in the work area. When the mouse pointer is above an object in this bar, an information bubble indicates the project or the model name.
- An information bar (6) indicating the current time and date.
- An area (7) displaying the list of opened systems and the decomposition; the dimensions of this area can be adjusted.
- A library (system, component, equipment, type, operators, FRB) area (8), allowing to manage objects and projects; the dimensions of this area can be adjusted.

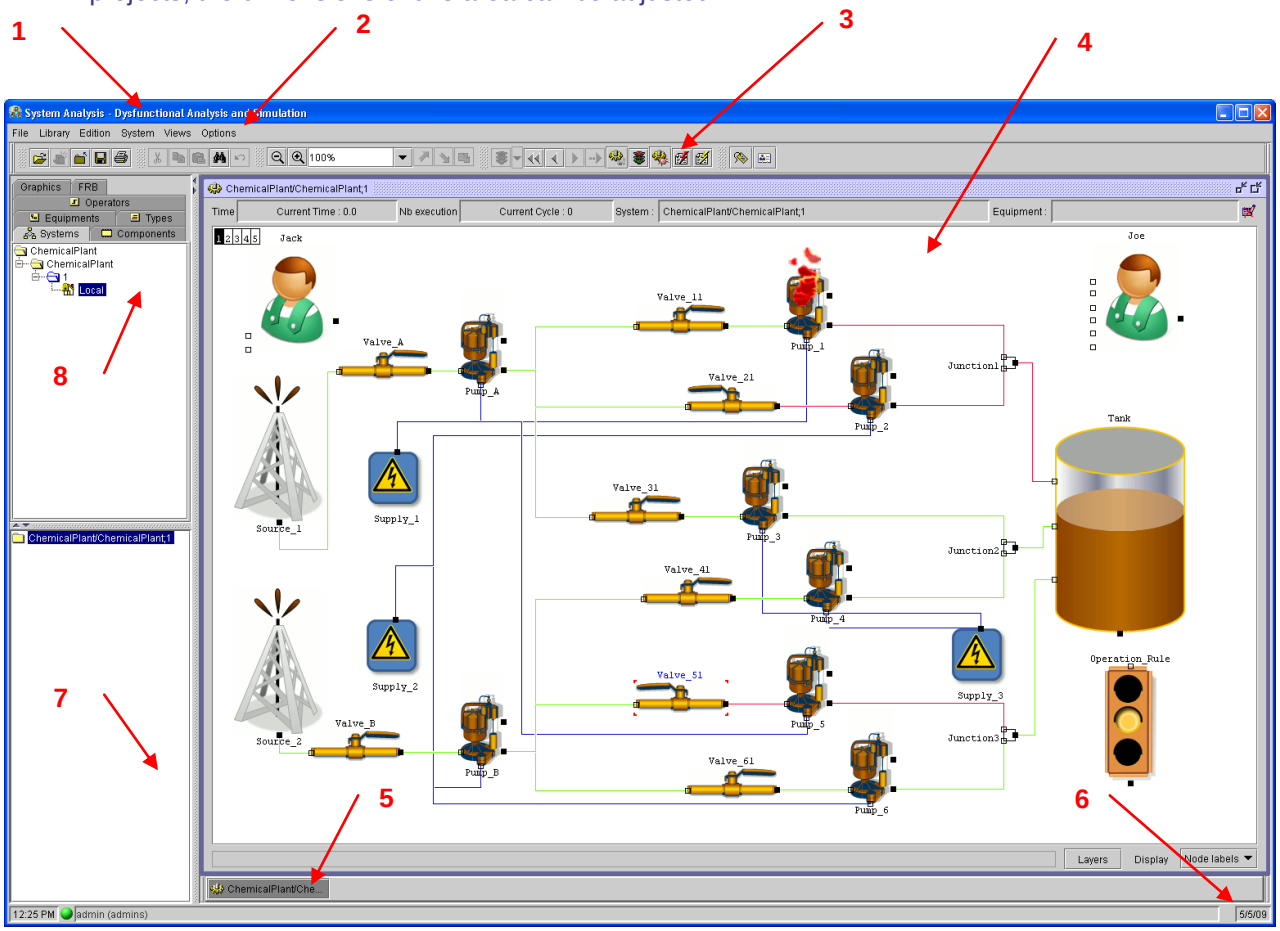


Figure 20 : Workbench

MENU BAR

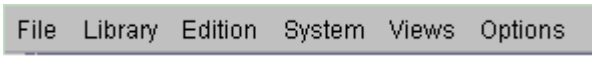


Figure 21 : Menu bar

The *menu bar* (Figure 21) on the top of the main window provides all DAS functionalities. It is composed of five menus:

- **File**: it contains all commands about projects or systems (New, Open, Save, Save as, Close, Freeze, Export, Import, Properties, Export Docbook, Printing format, Print, Change data base, Administration and Quit).

N.B: Freeze command can be applied on components, equipments and operators.

- **Library**: it contains all commands about families and models (Create, Remove, Duplicate, Rename, Delete, Edit, Add, Images).
- **Edition**: it contains all commands about object edition (Cut, Copy and Paste).
- **System**: it contains all commands about model processing (verification of property/syntax, simulation, sequence generation, fault tree generation, translate model…).
- **Views**: it contains all commands about the views (Architecture, Information, System links, Global synchronization and Colors).
- **Options**: it contains the software configuration (Preferences, Management of the license and About).

MAIN TOOL BAR

The main tool bar below the menu bar can be customized and is composed of seven command groups:

- Standard tool bar (Figure 22): it provides the following commands:
  * Open a system,
  * Create a new system,
  * Close a system,
  * Save a system,
  * Print (this command is not specific to a system architecture)

Figure 22 : Main tool bar

- Edition tool bar (Figure 23): it provides the following commands:
  * Cut,
  * Copy,
  * Paste,
  * Find a model or a component,
  * Cancel (undo).

Figure 23 : Edition tool bar

- Orientation tool bar (Figure 24): it provides the rotation commands:
  * Vertical mirror,
  * Horizontal mirror,
  * 90° rotation.

Figure 24 : Design tool bar

- Links tool bar (Figure 25) : it provides various kinds of graphical links between components or equipments:
  * Direct,
  * Right angle,
  * Right angle from right to bottom,
  * Right angle from left to down,
  * Right angle from right to top,
  * Right angle from left to top,
  * Possibility of specifying an orientation for the graphic link symbolizing the connection (an arrow is drawn to indicate the direction of the flow propagation),
  * Possibility to modify the body of the graphic link (thick line, double line)

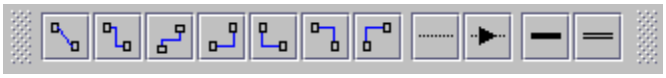


Figure 25 : Links tool bar

- Alignment tool bar (Figure 26) : it provides alignment commands of components or equipments in an architecture :
  * Left alignment,
  * Right alignment,
  * Top alignment,
  * Bottom alignment,
  * Centred alignment on a vertical axis,
  * Centred alignment on a horizontal axis,
  * Centre the selected elements.

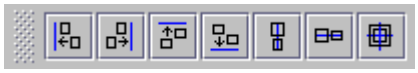


Figure 26 : Alignment tool bar

In all the cases the alignment is carried out the first selection element.

- Navigation tool bar (Figure 27) : it contains zoom (in and out) tools and the functionalities to navigate through the architecture hierarchy:
  * Zoom in,
  * Zoom out,
  * Zoom resolution,
  * Up,
  * Down,
  * Down in another view.

Figure 27 : Navigation tool bar

- The tool bar dedicated to the simulation of system model contains :
  - Start simulation,
  - Initialize,
  - Backward,
  - Forward,
  - Save current state as initial state,
  - Stop simulation,
  - Debug information window
  - Event selection
  - State value modification,

Figure 28 : Simulation tool bar

The main tool bar can be customized:

- Go to **Options – Preferences…** menu; the *Preferences* window is displayed (Figure 205).
- Select tab **Tool bars** and tick the tool bar which will be displayed on the main tool bar.

WORKAREA

The workarea is the main part of DAS GUI. It allows editing models and system architectures, or simulating systems graphically.

A taskbar below the workarea contains shortcuts of opened or minimized windows in the workarea (Figure 29). Click on a shortcut to restore on foreground the selected window.

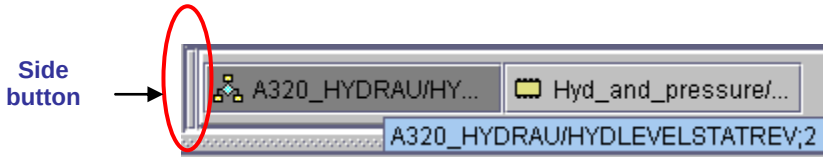


Figure 29 : Shortcuts of windows in taskbar

Note: click on buttons on taskbar sides to display hidden shortcuts if the taskbar is not large enough to display all of them.

Right click on a shortcut to select one of the following contextual commands: restore, reduce, extend or close window.

Set the mouse pointer above a shortcut to display an information bubble about the model.

INFORMATION BAR

The information bar below to the taskbar contains the following information: time, date, update status, user name, user group.

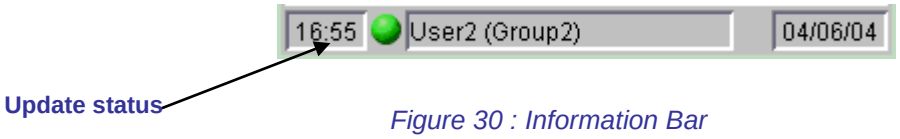


Figure 30 : Information Bar

The update status displays the validity of local data loaded in edition. The flag is green if local data of the model are identical to data from database.

When local data are different from database data, the flag becomes red.

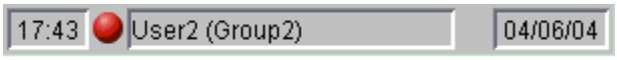


Figure 31 : Not Updated

To update the object:

- Left-click on red icon to launch the update. Object manager will take into account all modifications made by other users connected to database.

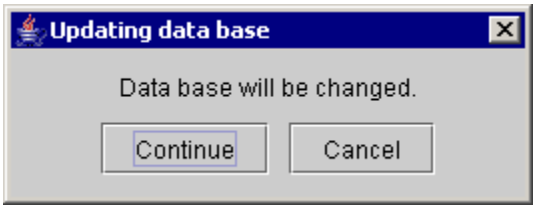


Figure 32 : Update object manager

- Click on Continue to accept the update.

MENU

MENU BAR

The menu bar contains the following menus:

- **File** (Figure 33):

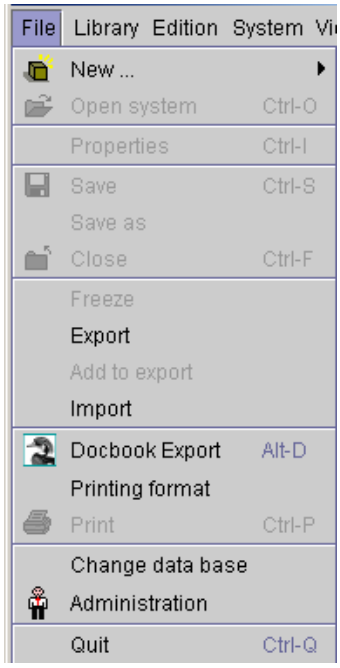


Figure 33 : menu File

- **Library** (Figure 34):

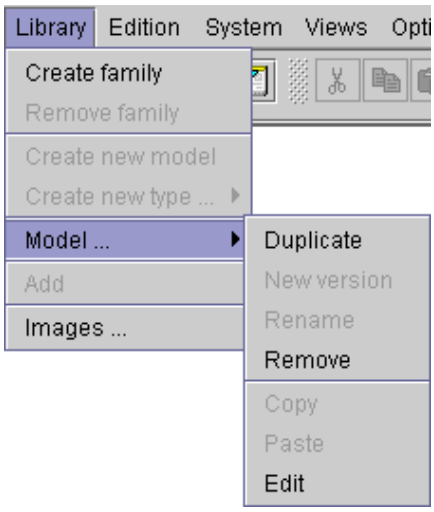


Figure 34 : menu Library

- **Edition** (Figure 35):

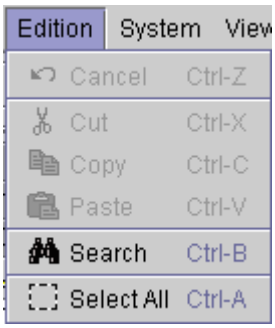


Figure 35 : menu Edition

- **System** (Figure 36):

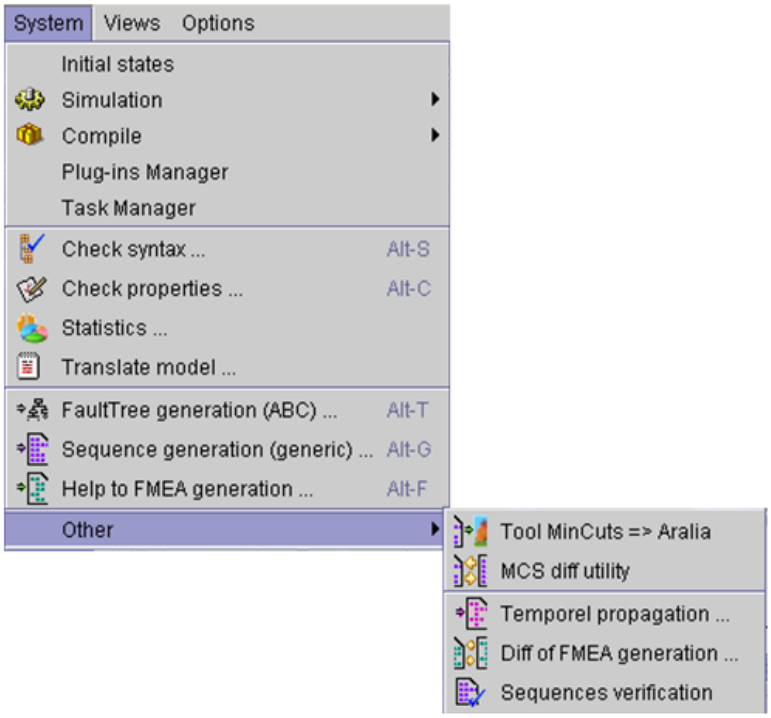


Figure 36 : System menu

- **Views** (Figure 37):

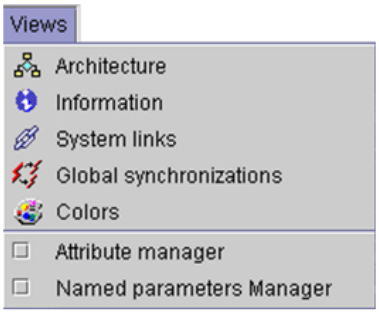


Figure 37 : menu Views

- **Options** (Figure 38):

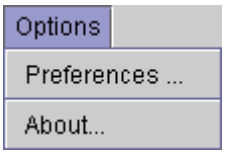


Figure 38 : menu Options

SUMMARY OF MENUS, SHORTCUTS AND ICONS

- File:
  - **New …** …… 
    - **Project**
    - **System**
  - **Open system** **CTRL + O** …… 
  - **Properties** **CTRL + I**
  - **Save** **CTRL + S** …… 
  - **Save as**
  - **Close** **CTRL + W** …… 
  - **Freeze**
  - **Export**
  - **Add to export**
  - **Import**
  - **Docbook Export** **CTRL+D**
  - **Printing format**
  - **Print** **CTRL + P**…… 
  - **Change data base**
  - **Administration** …… 
  - **Quit** **CTRL + Q**
- Library:
  - **Create family**
  - **Remove family**
  - **Create new model**
  - **Create new type …**
    - **Enumerate**
    - **Record**
  - **Model …**
    - **Duplicate**
    - **New version**
    - **Rename**
    - **Remove**
    - **Copy**
    - **Paste**
    - **Edit**
  - **Add**
  - **Images …**
- Edition :
  - **Cancel** **CTRL + Z**
  - **Cut** **CTRL + X**…… 
  - **Copy** **CTRL + C**…… 
  - **Paste** **CTRL + V**…… 
  - **Search** **CTRL + B**……
  - **Select All** **CTRL + A**…… 

- System:
  - **Initial states**
  - **Simulation** ..... 
    - **Start** ..... 
    - **Initialize** ..... 
    - **Go backward** ..... 
    - **Go forward** ..... 
    - **Go next** ..... 
    - **Save as initial state** ..... 
    - **Stop** ..... 
    - **Debug view** ..... 
    - **Events** ..... 
    - **State change** ..... 
  - **Compile** ..... 
    - **Byte Code Java Compiler [1] …**
  - **Plug-ins manager**
  - **Task manager**
  - **Check syntax…** **Alt + S**
  - **Check properties..** **Alt + C**
  - **Statistics…**
  - **Translate model …**
  - **FaultTree generation (ABC)** **Alt + T**
  - **Sequence generation (generic)** **Alt + G**
  - **Help to FMEA generation** **Alt + F**
  - **Other**
    - **Tool MinCuts => Aralia**
    - **MCS diff utility**
    - **Temporal propagation…**
    - **Diff of FMEA generation…**
    - **Sequences verification**

- Views:
  - **Architecture** ..... 
  - **Information** ..... 
  - **System links** ..... 
  - **Global synchronizations** ..... 
  - **Colors** ..... 
  - **Attribute manager**
  - **Named parameters manager**

- Options:
  - **Preferences …**
  - **About …**

DATA BASE MANAGEMENT

DAS allows to work with several databases (Access© and Oracle©) and to switch easily between these databases. User can:

- create several Access databases from an empty template,
- switch from an Access database to another Access database,
- switch from an Access database to an Oracle database,
- switch from an Oracle database to another Oracle database,
- switch from an Oracle database to an Access database.

Database management is available from the menu **File – Change database**:

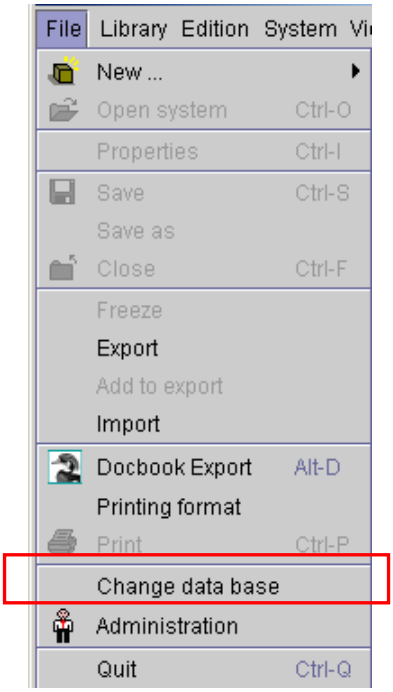


Figure 39: Menu File - Change database

CHANGE DATABASE

To change database, select **Change database** from the menu **File** as displayed on Figure 39. The following dialog box is displayed:

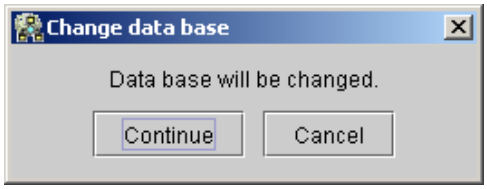


Figure 40: Dialog box for database change

Click on **Continue** to confirm the database change.

The popup menu allows user to choose the type of database (Figure 41):

- **Access** database
- **Oracle** database
- **ODBC** data source

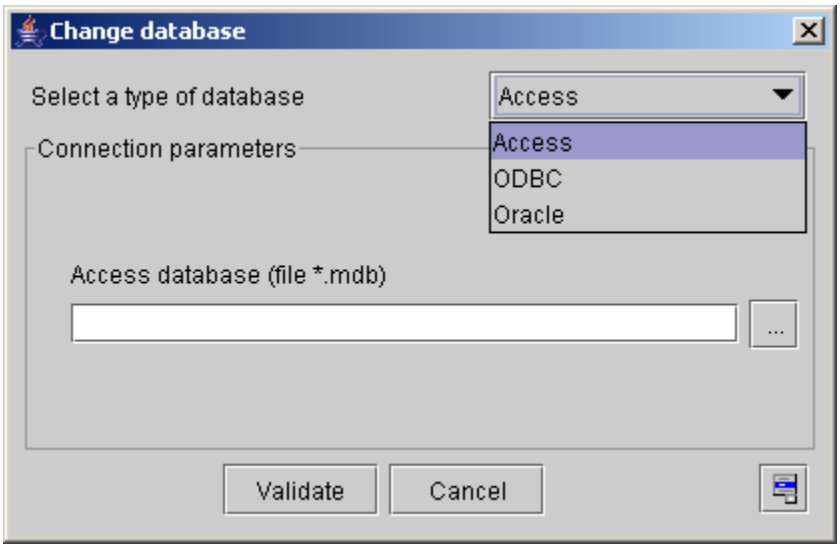


Figure 41: Selection of a database type

Selection of a database Access

Select type **Access** in the popup menu (Figure 41) to work in another database Access.

Enter path of the Access database file in the edition area or click on button Browse  to search it from an explorer.

Note: File extension of Access database is .mdb.

Then click on **Validate** to change database.

Selection of a database Oracle

Select type **Oracle** in the popup menu (Figure 41) to switch in a database Oracle.

The following configuration window is displayed:

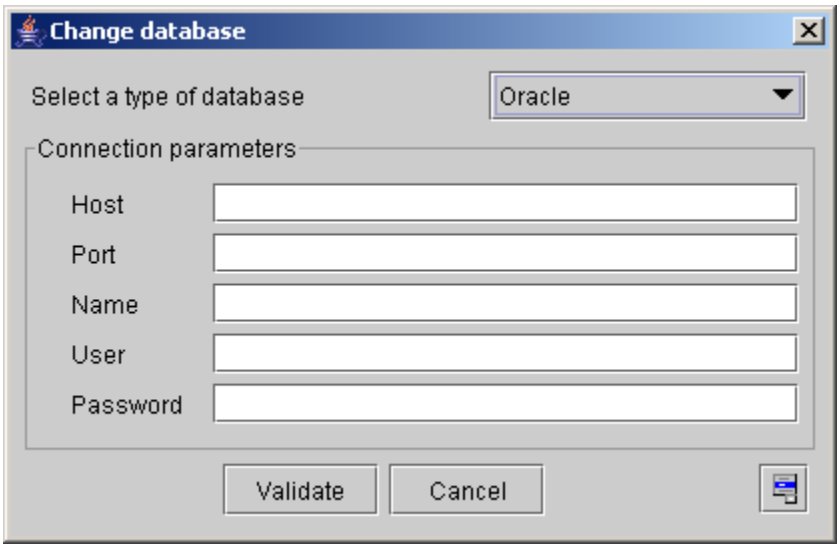


Figure 42: Selection of an Oracle database

Fill in the connection parameters to the Oracle database:

- Host
- Port
- Name
- User
- Password

Then click on **Validate** to change database.

Selection of a data source ODCB

Select type **ODBC** in the popup menu (Figure 41) to choose a data source ODBC.

Enter the name of the data source in the edition area (Figure 43).

Note: The creation of an ODBC data source is not explained in this manual.

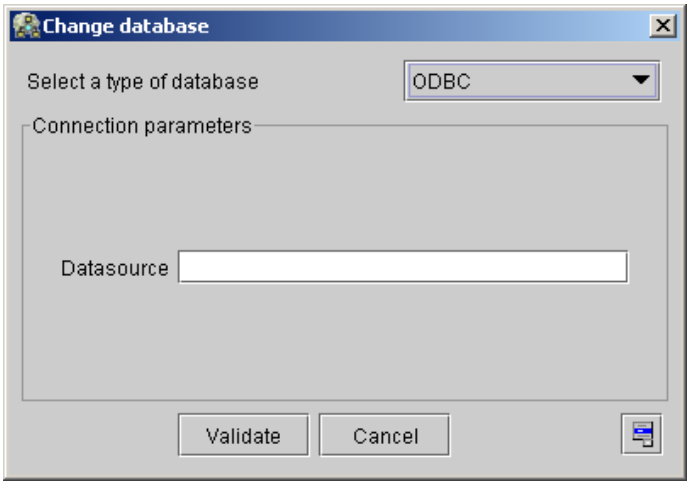


Figure 43: Selection of an ODBC data source

Then click on **Validate** to change database.

CREATE A NEW DATABASE

User can easily create a new empty Access database from the database management windows.

To do this, click on the button **Create new database** (Figure 44):

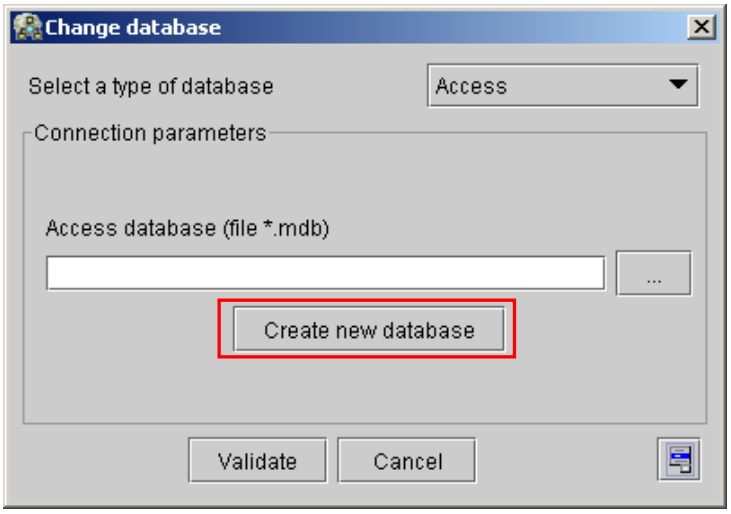


Figure 44: change database

Select '**Create new database**' in the displayed menu.

Note: Creation of a database is only available for type Access.

Choose a destination directory for the new database Access and a file name (extension .mdb) :

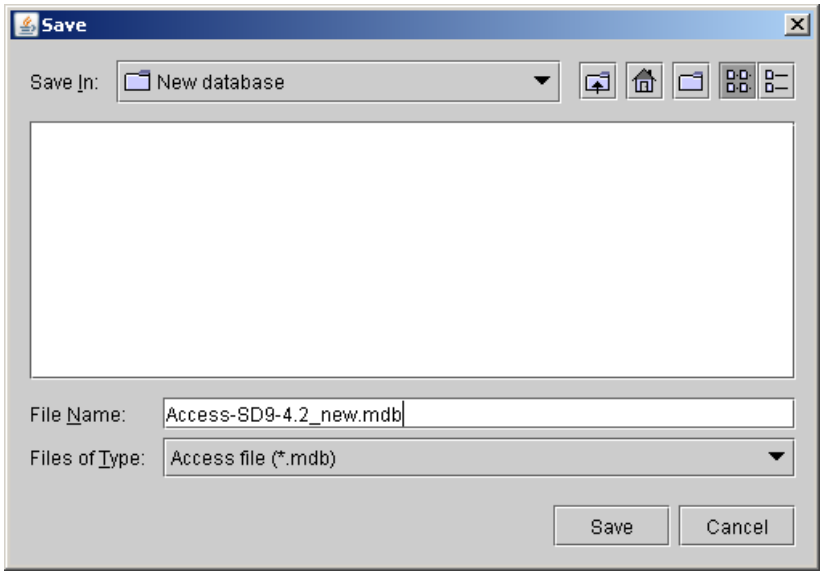


Figure 45: Selection of a new Access database

Click on **Save** to validate creation of the database.

Then, click on button Validate. :

After the database change is effective, user must logon in the new administration console.

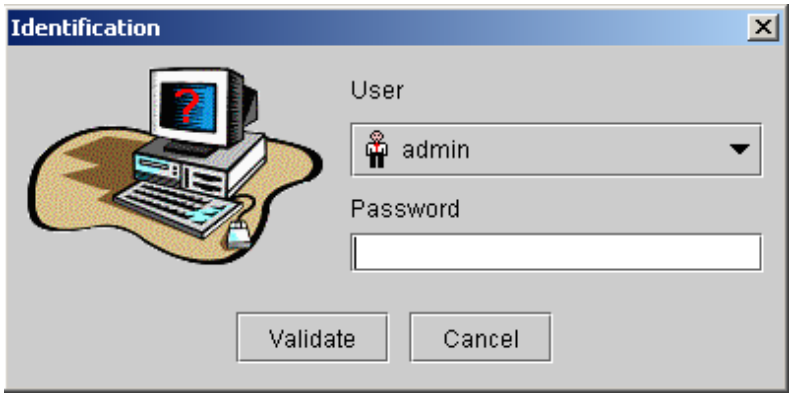


Figure 46: Identification console

LIBRARY OF MODELS

GENERAL

The library contains the main standard models in the following tabs:

- Types,
- Operators,
- Components,
- Equipments,
- FRB (Failure Rate Bank),
- Graphics
- Systems

These models are described below.

N.B : tab **Systems** contains the architectures made from generic models (instantiation of generic components and equipments). Systems are classified in **Projects**.

ORGANIZATION OF MODELS

Models can be organized in folders and sub-folders in the library. Two kinds of folders are available:

- **Family**: this kind of folder allows organizing models of components, equipments, operators, types.

  Models can be created anywhere in the folder tree. The following figure (Figure 47) presents an example of organization of components in folders and subfolders.

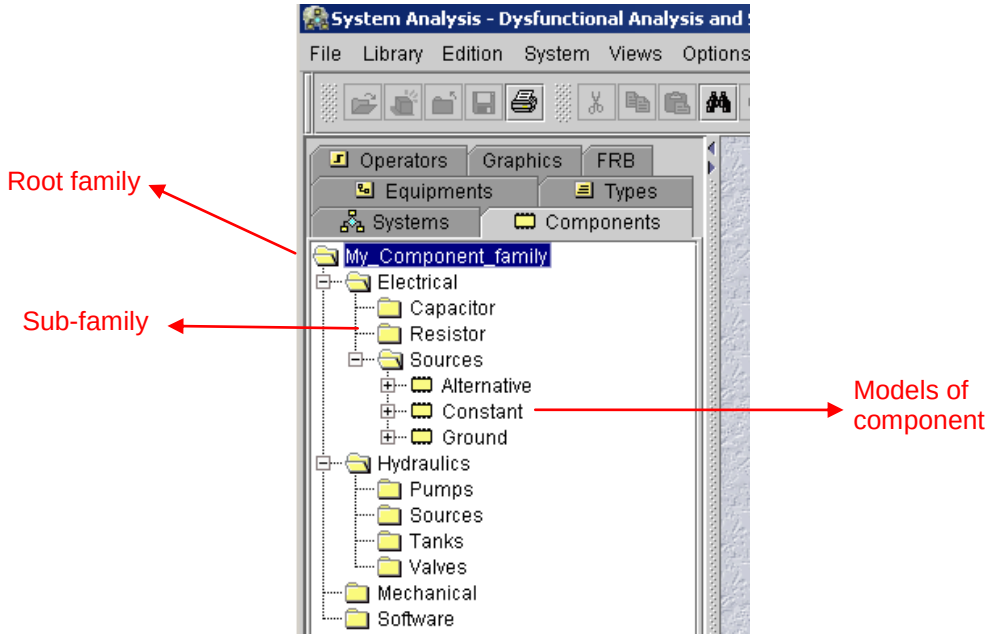


Figure 47: Example of component organization

- **Project**: this kind of folder allows to organize models of systems in a tree of projects and sub-projects.

  Models of systems can be created anywhere in the folder tree. The following figure (Figure 48) presents an example of organization of systems in folders and subfolders.

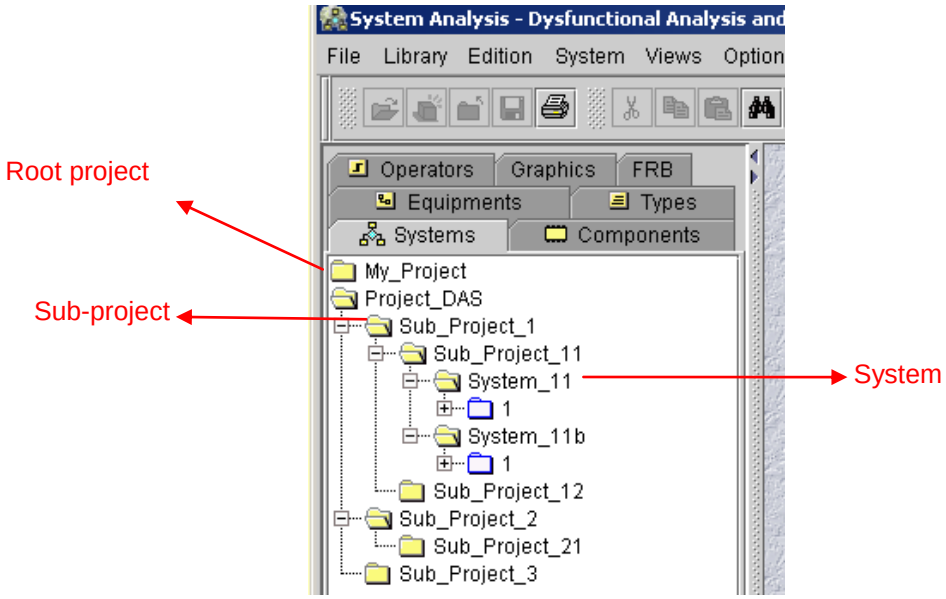


Figure 48: Example of System organization

Create a root family

To create a new root family of components, equipments, operators or types in the library:

- Select components, equipments, operators or types tab
- Right click in the empty area (no family must be selected) to display the contextual menu
- Select the command **Create family**

Note: Command for creation is also available from the menu **Library > Create family**.

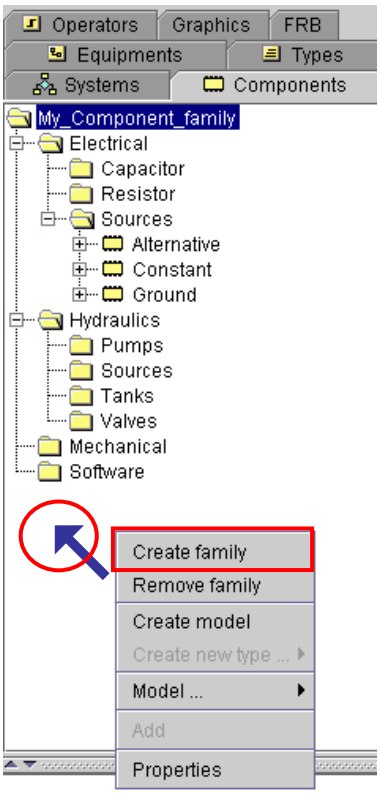


Figure 49: Creation of a root family

- Enter the name of the new family in the dialog box (Figure 50)

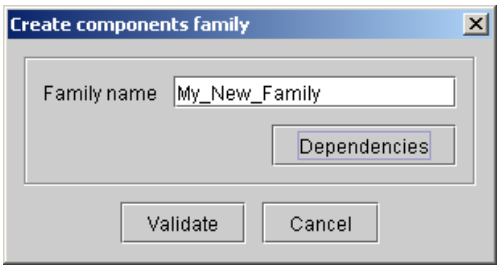


Figure 50: Family name and dependencies

- If the owner belongs to several workgroups, he has to specify the attachment group of the family. Click on the button **Dependencies** to select the group. Click on button **Close** to validate.

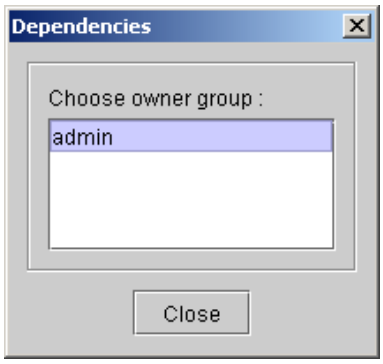


Figure 51 : Selection of dependencies

- Click on **Validate** to accept the creation of the new family.

The new root family is created in the library.

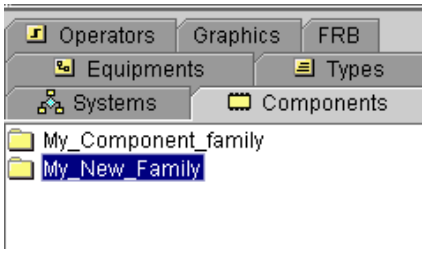


Figure 52: New root family created in the library

Create a sub-family

To create a new sub family in an existing family of models:

- Select components, equipments, operators or types tab
- Select the existing family in which the new sub-family must be created
- Right click on the existing family to display the contextual menu
- Select the command **Create family** (Figure 53)

Note: Command for creation is also available from the menu **Library > Create family**.

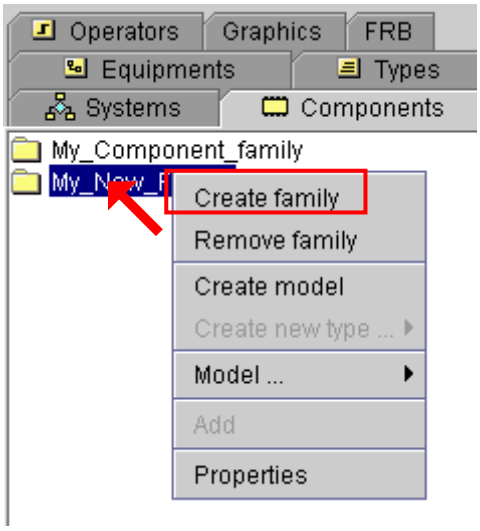


Figure 53: Creation of a sub-family

- Enter the name of the new sub-family in the dialog box (Figure 54)

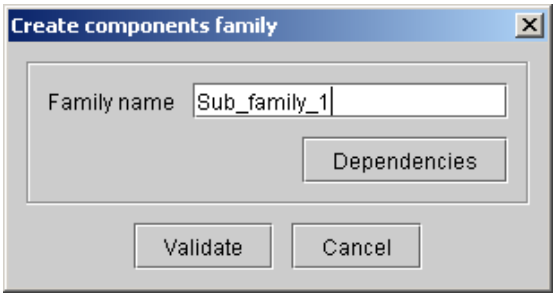


Figure 54: Name of the new sub-family

- If the owner belongs to several workgroups, he has to specify the attachment group of the family. Click on the button **Dependencies** to select the group. Click on button **Close** to validate.
- Click on **Validate** to accept the family creation.

The new sub-family is created and added under the selected existing family.

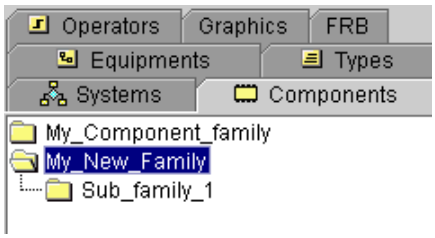


Figure 55: New sub-family created under the existing family

Create a project

Create a root project

To create a new root project in the library:

- Select **Systems** tab
- Right click in the empty area (no project must be selected) to display the contextual menu
- Select the command **Create… > Project**

Note: Command for creation is also available from the menu **File > New > Project**.

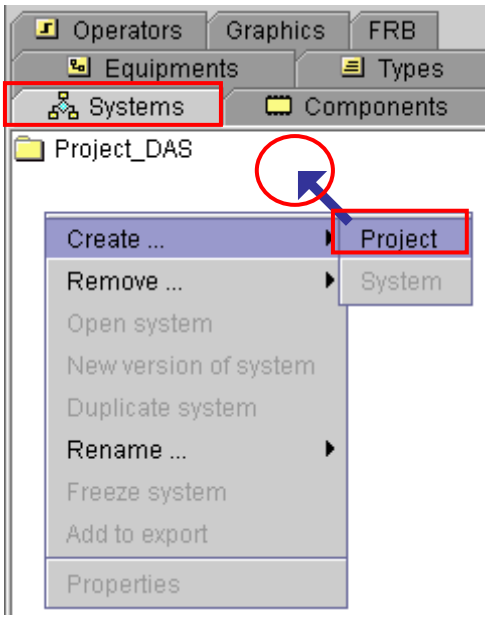


Figure 56: Creation of a new root project

- Enter the name of the new root project in the dialog box (Figure 57)

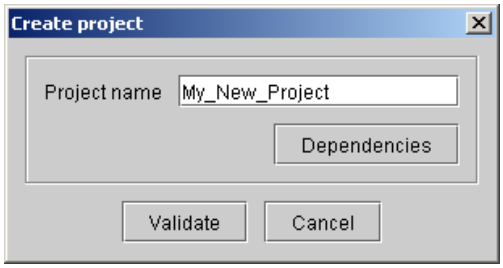


Figure 57: Name of the new project

- If the owner belongs to several workgroups, he has to specify the attachment group of the project. Click on the button **Dependencies** to select the group. Click on button **Close** to validate.

- Click on **Validate** to accept the project creation.

The new root project is created in the library:

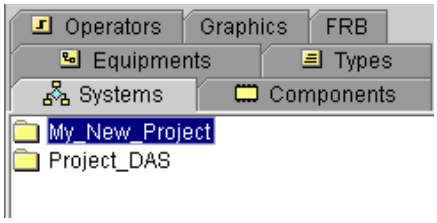


Figure 58: New root project created in library

A project can contain sub-projects and/or models of systems.

Create a sub-project

To create a new sub project in an existing project:

- Select **Systems** tab
- Select the existing project in which the new sub-project must be created
- Right click on the existing project to display the contextual menu
- Select the command Create… > Project (Figure 59)

Note: Command for creation is also available from the menu **File > New > Project**.

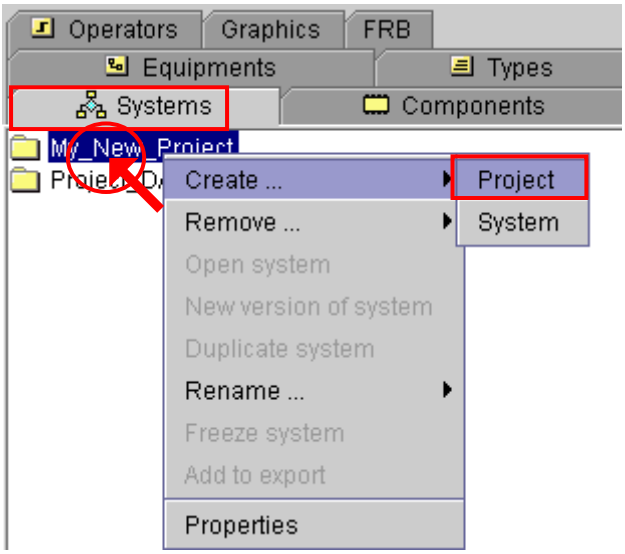


Figure 59: Create a new sub-folder

- Enter the name of the new sub-project in the dialog box

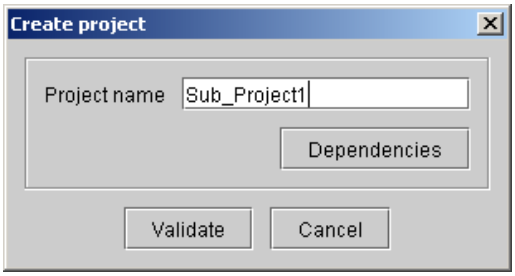


Figure 60: Name of the sub-project

- If the owner belongs to several workgroups, he has to specify the attachment group of the project. Click on the button **Dependencies** to select the group. Click on button **Close** to validate.
- Click on **Validate** to accept the project creation.

The new sub-project is created and added under the selected existing project.

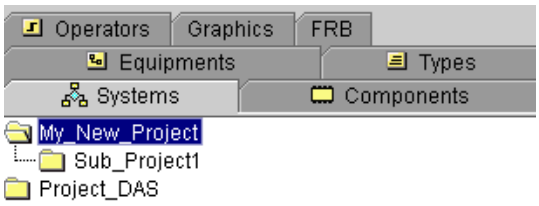


Figure 61: New sub-project created in the library

A sub-project can contain other sub-projects and/or models of systems.

Remove a family or a project

To remove a project of systems or a family of components, equipments, operators, types:

- Ensure that the family or the project to remove is empty, i.e. it doesn't contain any sub-family/sub-project or models
- Click on the family or the project to remove to select it in the library
- Right click to display the contextual menu and select **Remove family** or **Remove > Project (**Figure 62**)**

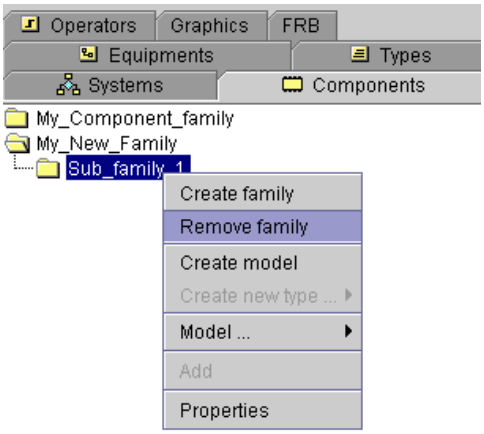


Figure 62: Remove a family or project from the library

- Click on **Continue** to remove the family/project (Figure 63)

Figure 63: Remove confirmation

The family or the project is removed from the library and the database.

Note: If the family/project is not empty, it is not possible to remove it. An error message is displayed:

Figure 64: Error message

User has to remove first all models and families/projects.

EDIT A COMPONENT

Create a new component model

To create a new component model:

- In the left part of the window, click on tab **Components**.
- Select the family in which the component will be created.
- Right click on the family to display the contextual menu.
- Click on command **Create new model**; the following bar graph is displayed (Figure 65).

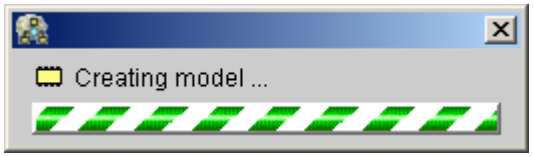


Figure 65 : Bar graph during the component creation

- The component editor is open (Figure 66).
- Complete information in the tabs

A component model can be considered as a flow processor which:

- has an internal state
- reacts to events
- receives and/or sends data through (**I/O**) interfaces which enable him to interact with other components (by flow propagation).

When an event occurs, the component checks the internal state transition induced and assesses the new flow values emitted on output ports.

The component editor allows specifying the features and the behaviour in the following tabs:

- The tab **General** to specify general information.
- The tab **I/O** to fill the set of flow variables emitted or received by the component.
- The tab **States** to fill the set of state variables characterizing the possible states of the component : functional and/or dysfunctional;
- The tab **Events** to specify the events on which reacts the component (occurrence of an event modifies the state of the component).
- The tab **Icons** to specify the graphic representations corresponding to the possible states of the component (used during the graphical simulation of the system).
- The tab **Altarica code** to specify the conditions of state changing (transitions) and the new value of output flows (assertions).

When the new model is saved, it is automatically stored in the library (alphabetically) with its version.

Remark: A new component model is created with version 1.

General tab

The **General** tab (Figure 66) contains the following information:

- **Name**: enter the model name,

  Note: This field is mandatory to save the model in library.
- **Width**: default value of icon width,
- **Height**: default value of icon height,
- **Icon file**: click on the browser button to choose a default icon for the model.

Remark: The possible file formats for icon are "***.gif**" or "***.jpg**".

- Check the box **Draw border** to display or a black border around the icon.
- Check the box Move ports automatically when components are reshaped.
- Click on **Resize** button to update, if necessary, the modifications of icon size.
- **Comment**: to add a comment for the model.

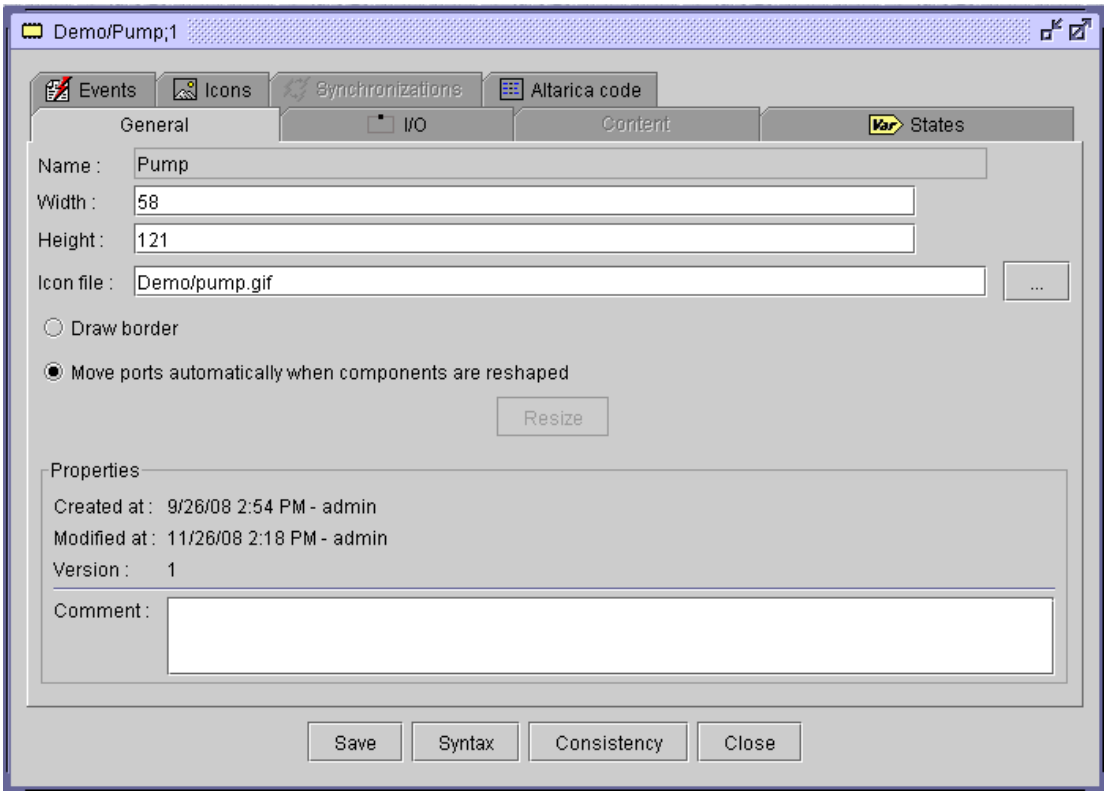


Figure 66 : Component – General tab

The component editor contains four buttons:

- **Save**: to save the model in the database,
- **Syntax**: to check syntax of Altarica code,
- **Consistency**: to check consistency of Altarica code,
- **Close**: to close the component editor.

Two shortcuts on the top right side allow modifying the window size:

- Reduce,
- Extend.

Remark: If the component is currently locked by another user, or if the component is simulated, it is not possible to save the modifications.

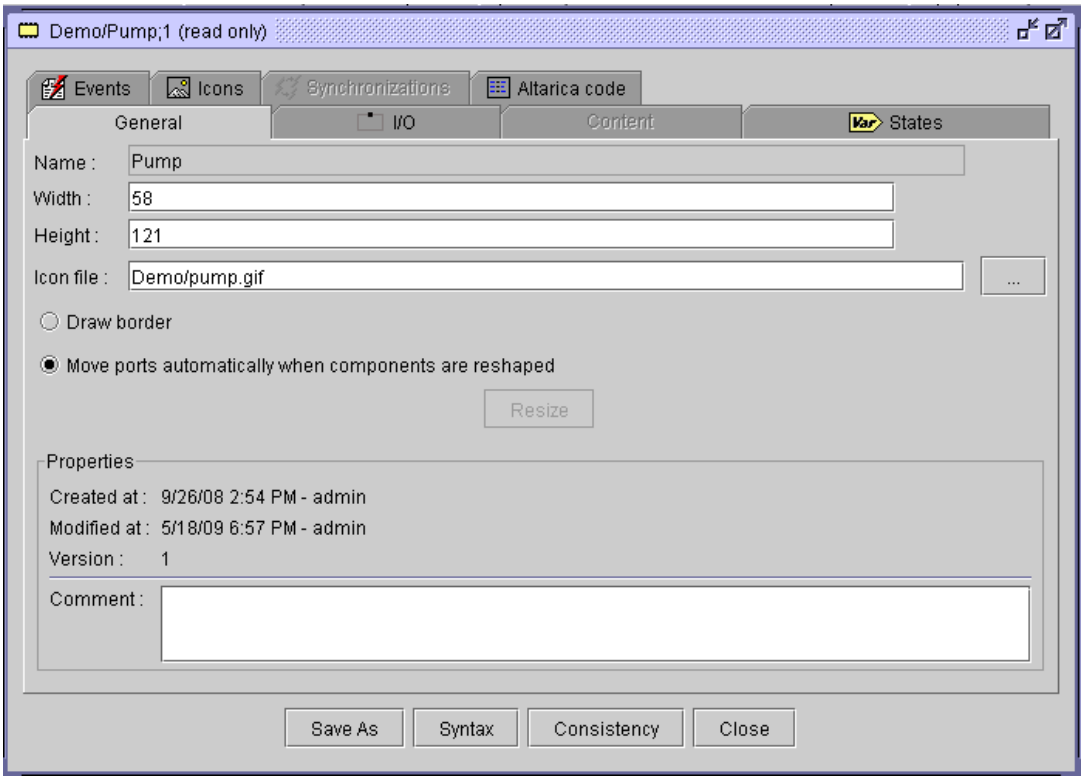


Figure 67 : Model modification during simulation

The button **Save** is replaced by the button **Save as** to save the modifications as a new model. Click on **Save as** to select the name of the new component and the family.

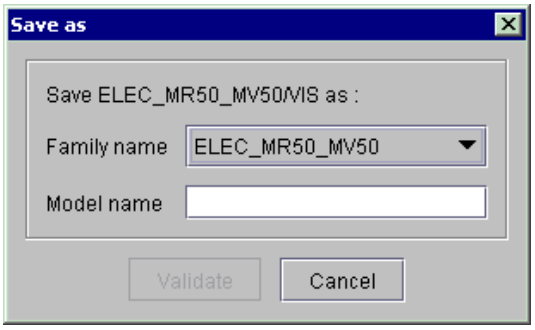


Figure 68: Model – Save As

The **Save As** button is available in component editor only when models can't be modified.

I/O *tab*

The **I/O** tab (Figure 69) allows to define and manage the input/output flow variables and to ports of the components.

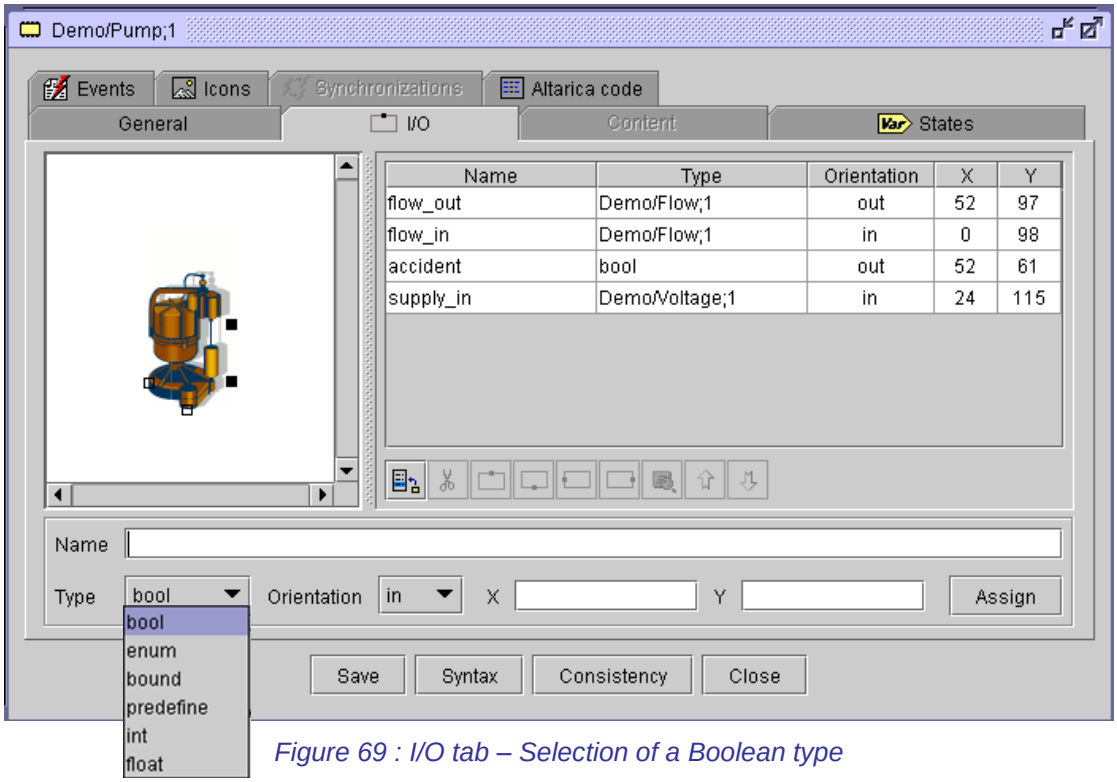


Figure 69 : I/O tab – Selection of a Boolean type

To create a new port on the component:

- In the field **Name**, enter the name of the flow to be created.

- Use the menu **Type** to select the type of the flow:
  - ***bool:*** *for* boolean type (Figure 69), the possible values are **true** and **false**
  - ***enum*****:** for enumerate type (Figure 74), values **must** be separated by commas without spaces, e.g. : *null,low,high*.
  - ***bound:*** *v*ariable is an integer having a value between a lower bound and a upper bound (Figure 75),
  - ***predefine:*** predefined type created in library (Figure 76)
  - ***int :*** integer type
  - ***float :*** float type
- In the menu **Orientation**, select the direction of the port: **in** (input), **out** (output) or **local**.
- **NB**: choose **local** if you want to create an internal variable (local flow) in the component. This type of variable allows to store complex assertions in the component. It is considered as a "memory" variable.
- 

Figure 70 : Port direction

- Enter the coordinates of the port in the fields X and Y.
- Click on icon or on key **Enter**; a new port is added in the list of I/O ports. The port is added graphically around the icon.

N.B: An input is identified by a white rectangle and an output by a black one.

-

NB 2: If an **I/O** is selected, the new **I/O** is added above the selected **I/O**. If no **I/O** is selected, the new **I/O** is added at the end of the list.

NB 3: User can re-order the list of the port. To move a port down or up in the list, click on the following tools:

- Move up ,
- Move down .

-

To modify the position of a port:

-
- Use the following tools to move the port automatically in the middle of a border:
  - ***Top:*** icon,
  - ***Down:*** icon,
  - ***Left:*** icon,
  - ***Right:*** icon.
- Or use the mouse to move the port I/O anywhere on the component border.

- Or double-click on the coordinates to edit the fields X and Y in the list of I/O.

To edit a port:

- Click on the port to select it in the list. The fields Name, Type, Orientation, and Coordinates are updated and can now be modified (Figure 71).

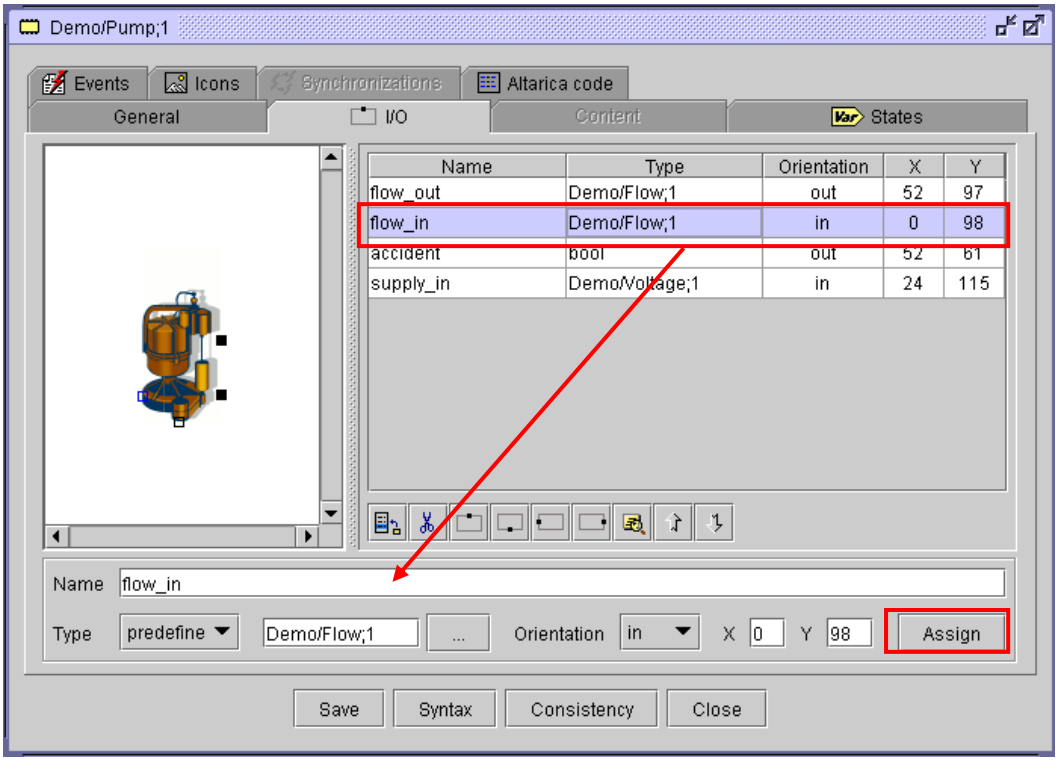


Figure 71: Modification of a port

- Modify the features of the port
- Click on **Assign** to update the port in the list.

To modify the name of a port

To modify the name of a port, double-click on the name of the port in the list. The name can be edited. Then, click on **Enter** to validate the new name.

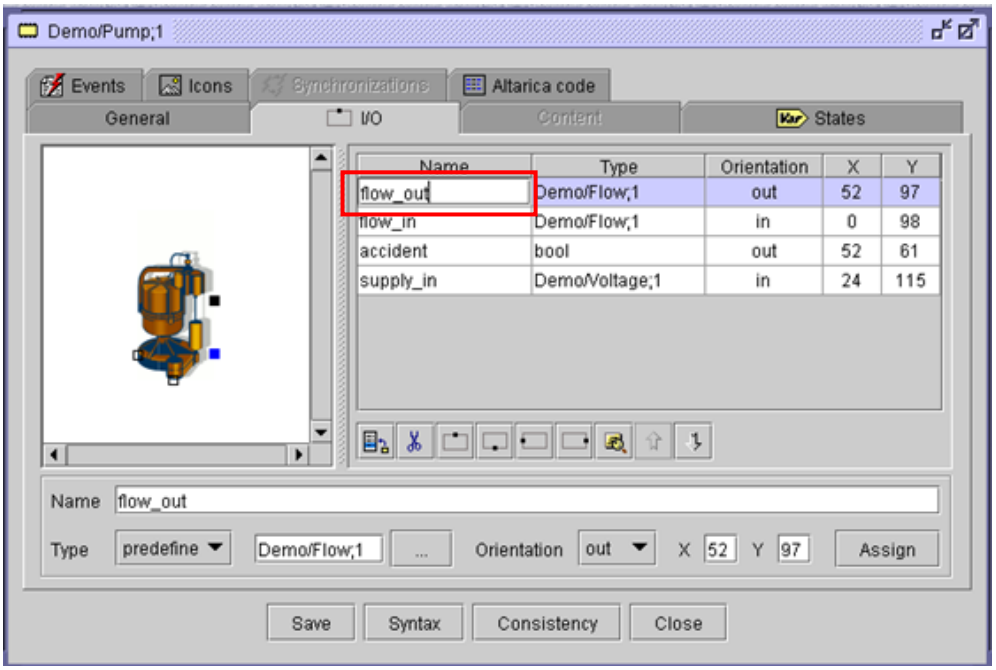


Figure 72: Modification of a port name

To modify a predefine type of a port:

User can create predefined types in the library. When a new port is added, it is possible to associate a predefined type from the library. The **I/O** tab of the editor allows to modify easily a predefined type:

- Right click on a port to display the contextual menu (Figure 73) and select **Edit Type**.

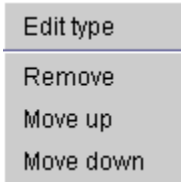


Figure 73 : Contextual menu

- Or click on the icon in the tool bar.
-
- Or double-click on the path name of the type in the list of ports.

To delete an existing port:

- Select the port in the list and click on icon to remove it from the component.
- Or right click on a port to display the contextual menu (Figure 73) and select **Remove.**
- Or click on key **SUPPR.**

Example of port types

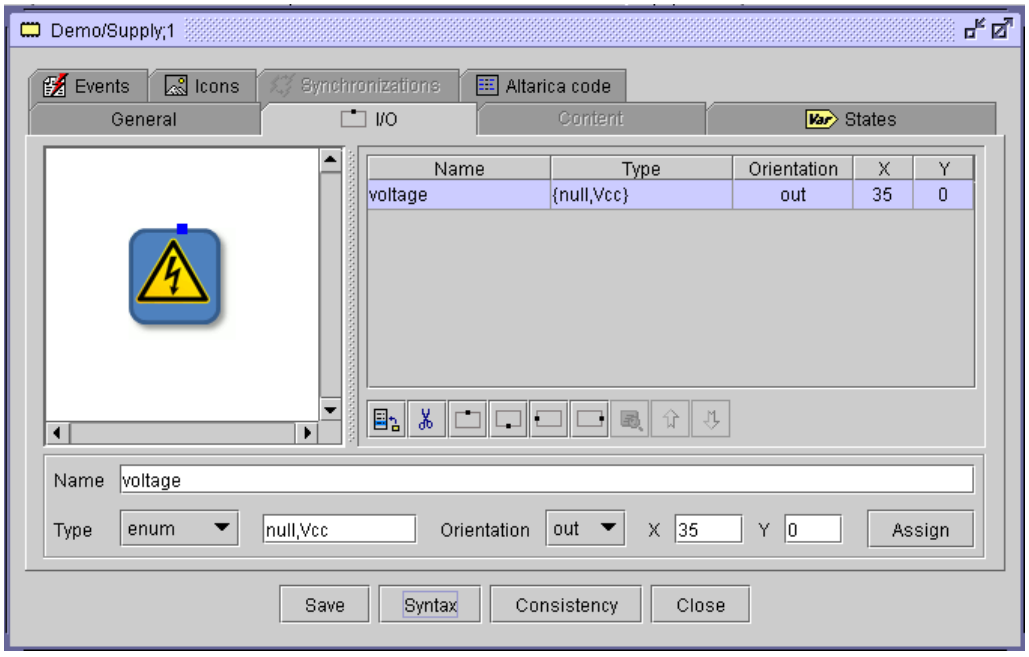


Figure 74 : I/O tab – Enumerated variable

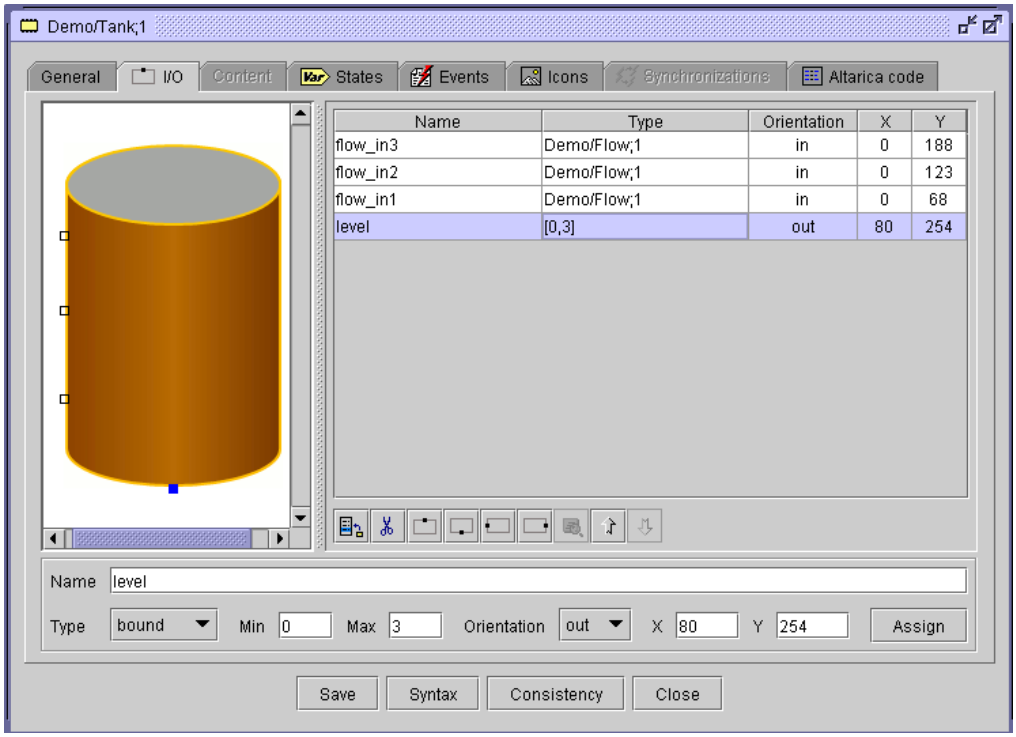


Figure 75 : I/O tab – Bound variable

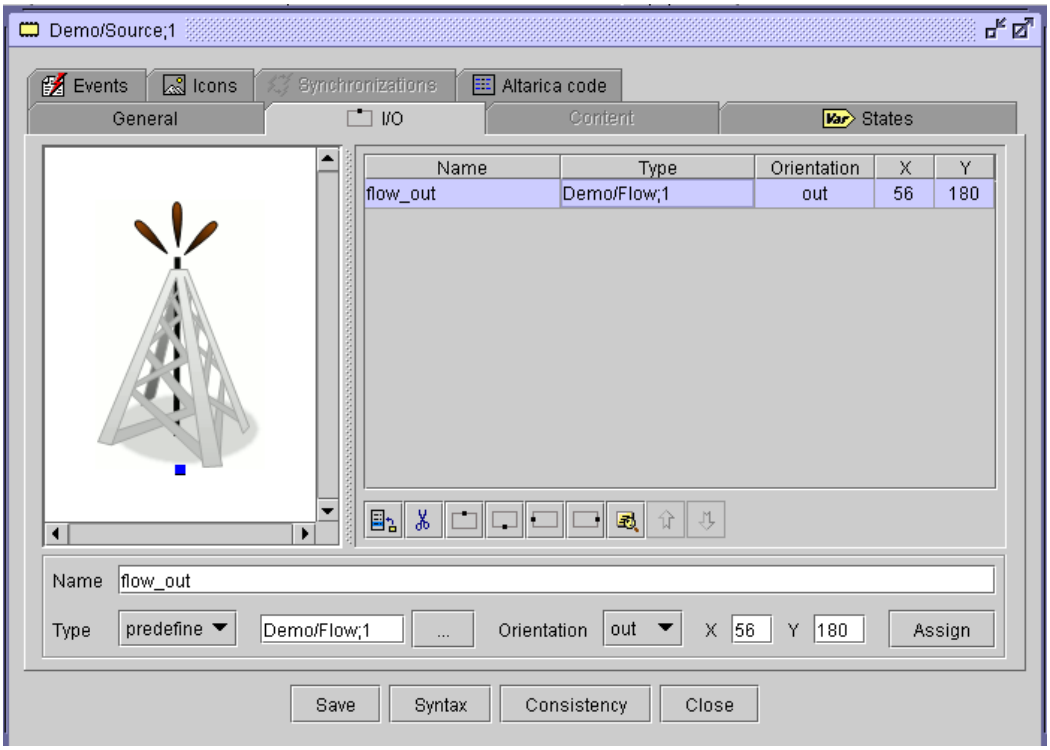


Figure 76 : I/O tab – Predefined variable

States tab

Click on tab **States** (Figure 77) to define and manage component states:

- In the field ***Name***, enter the name of the state variable.
- Use the popup menu **Type** and choose the type of state variable:
  - **bool**: boolean type,
  - **enum**: enumerated type, e.g. *ok,failure*,
  - **bound**: integer type with a value comprised between a lower bound and an upper bound,
  - **predefine**: predefined type in library (Figure 76).
- In the menu **Value**, choose the default value for the state. Default values are used to set the initial value of the component when a processing is launched (simulation, fault tree generation…).
- Click on icon or key **Enter** to accept the state creation.

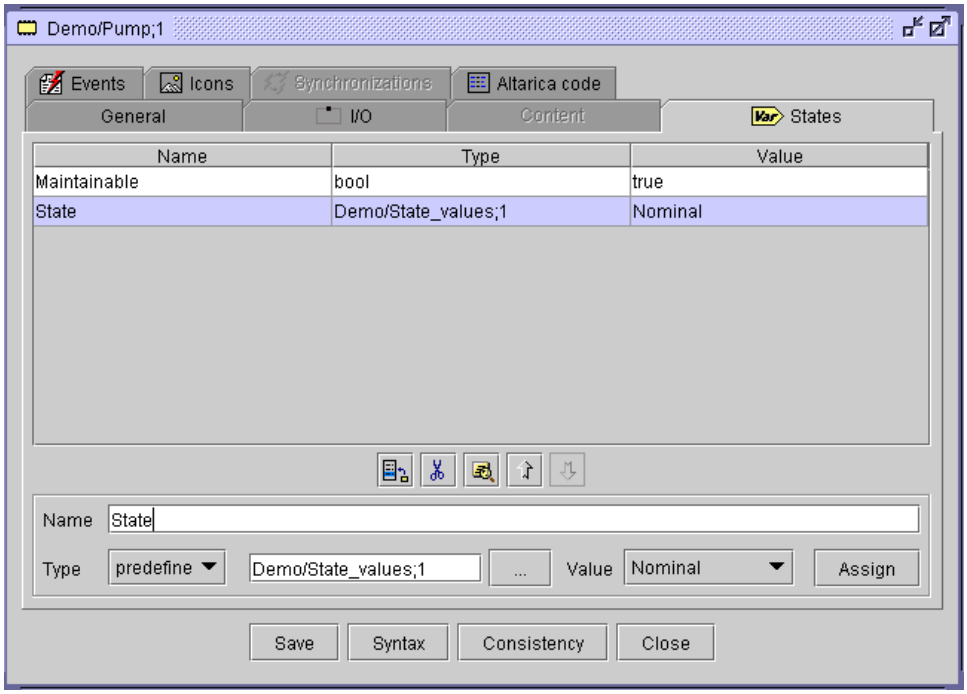


Figure 77 : Component – State tab

Remark: If a state variable is selected, the new state is added above the selected state. If no state is selected, the new one is added at the end of the list.

- To modify the type of an existing state variable, select it: choose a new type for this variable and click on **Assign** to update the state variable in the list.
- To modify the name of an existing state, click on the **Name** in the list; modify it and validate with **Enter** key.
- Click on **Value** field in the list to modify the default value if necessary.
- Three commands are available:
  - Edition of the predefined type of the state variable : click on this icon or double click on the predefined type to edit it
  - To move up a selected state in the list ,
  - To move down a selected state in the list .

Events tab

The tab **Events** (Figure 78) allows defining and managing events for the component model:

- In the field **Name**, enter the name of the event.
- Click on icon or key **Enter** to create the new event. It is added in the list.

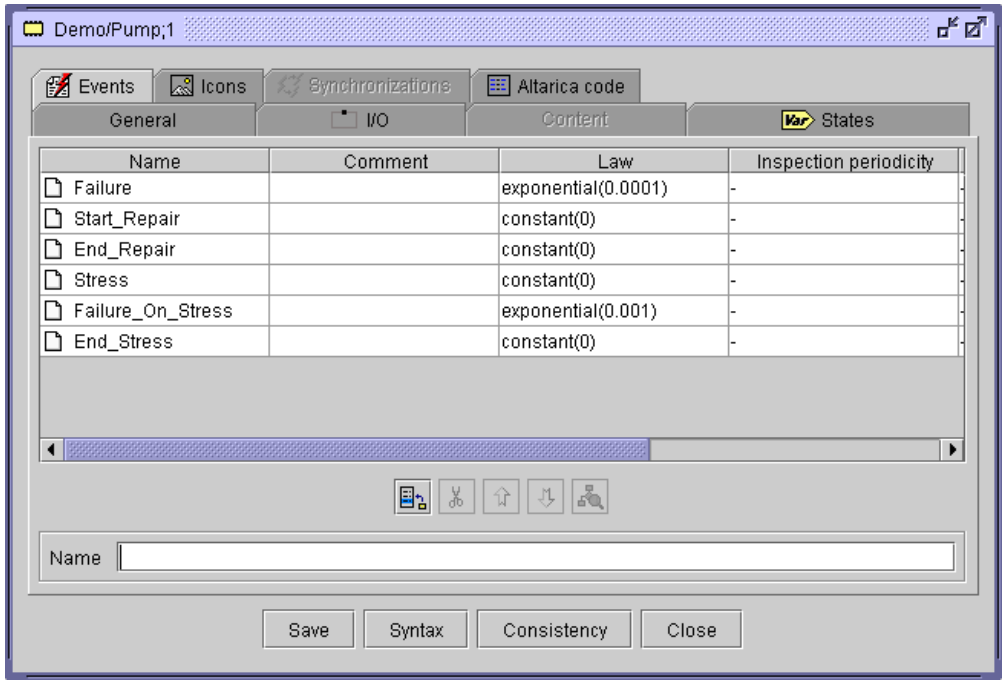


Figure 78 : Events tab

To modify the name of an existing event, double click on the **Name** field; modify it and validate the new value with **Enter**.

Remark: If an event is selected, the new event is added above the selected one. If no event is selected, the new is added at the end of the list.

- Click on icon to delete selected event.
- Click on icon to move up the selected line.
- Click on icon to move down the selected line.

Properties of an event

To edit the properties of an event, select the event and click on icon (Figure 80).
User can define:

- A comment
- A probability law
- Attribute(s)

Edition of a comment

Click on tab and enter free text (Figure 79).

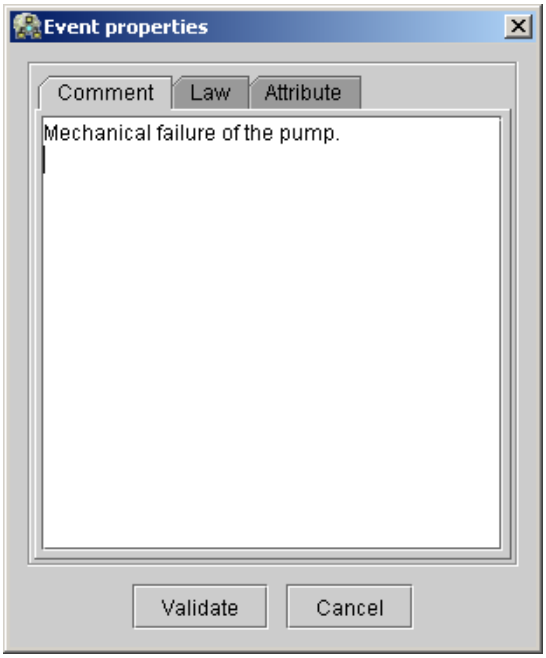


Figure 79: Comment of an event

Remark : The comment added on an event is included in fault tree file generated by DAS (Aralia file). An event comment becomes an attribute in Aralia format which can be used by fault tree editors.

Edition of a law

The field **Law** allows associating a probability law to an event. Laws are used to quantify probabilities on fault trees.

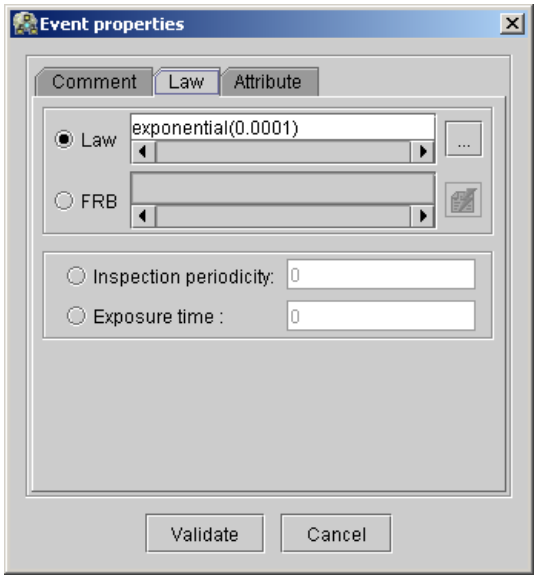


Figure 80 : Event law

To associate a probability law to an event, user can:

- Enter a link to the Failure Rate Bank. Check the box FRB and enter the link to the probability law stored in the library (FRB). For more information about Failure Rate Bank, please cf. to the appendix SD9_Appendix_FRB_R19_D7.pdf

- Or edit the law in the field **Law** according to the following syntax:

  <*Law_Name*> (<*Parameter_List*>) ; parameters are separated with comma

  The usual probability law are presented in the following table:

<Law_Name> (<Parameter_List>)	Example
constant (<probability>)	constant(0.5)
exponential(<lambda>)	exponential(1e-6)
GLM (<gamma> ,<lambda>, <mu>)	GLM(0.5,1e-6,1e-3)
GLM_asymptotic (<lambda>,<mu>)	GLM_asymptotic(1e-6,1e-3)
Weibull(<alpha>,<beta>,<t0>)	Weibull(0.5,1.5,0.5)
Periodic_test(<lambda>,<period>,<t0>)	periodic_test(1e-3,10,0)
Periodic_test_2(<lambda>,<lambda*>, <mu>, <tau>, <theta>, <gamma>, <pi>, <X>, <sigma>, <omega>)	periodic_test_2(1e-4, 1e-3, 0.125, 72, 72, 0,0, 1, 1, 0)
dormant (<lambda>, <MTTR>, <period>)	dormant (1e-3, 10, 25)
CMT(<lambda>, <duration>, <probability>)	CMT (1e-3, 72, 0)
NRD (<mu>, <delay>)	NRD(0.125, 10)

Table 1: Usual probability laws

- The button  of Figure 80 opens a law editor which enables easy law writing.

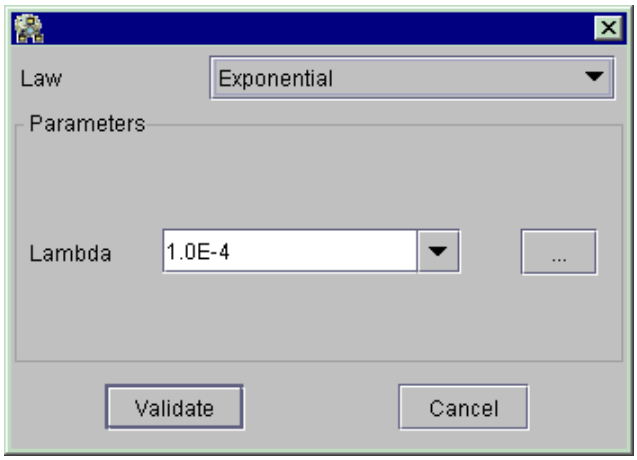


Figure 81 : Law editor

- Law editor is split in two parts; the first part contains a combo box to select the law. The second one contains as many fields as law parameters. Enter the value of parameter(s). The button  opens an editor to define an uncertainty law associated to the parameters:
- Lognormal
- Uniform
- Normal

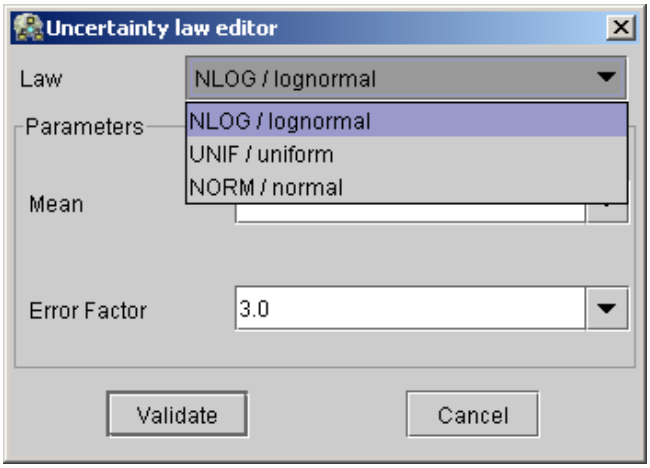


Figure 82: Uncertainty law editor

Edition of attribute(s)

- The tab **Attributes** allows the user to associate attributes to an event. The general syntax is the following:

  <Attribute_Name> <Attribute_Value> **Example:** ***' Type="Valve" '***

Figure 83: Event attributes

Attributes have to be created in the **Attributes manager**. For more information about the attribute creation, please refer to the appendix SD9_Appendix_ Attributes_Named_Parameters_R19_D7.pdf.

Icons tab

General

The tab **Icons** (Figure 84) allows selecting icon(s) to display graphically evolution of component behaviour during the simulation. To select icon(s):

- Click on tab **Icons**
- In the right part of the tab, select in the drop-down menu the **directory** where icons are stored.

Remark : the previous selected directory is memorized.

- Select the icon.
- Click on to add the selected icon to the icon list; a number is associated to each added icon.

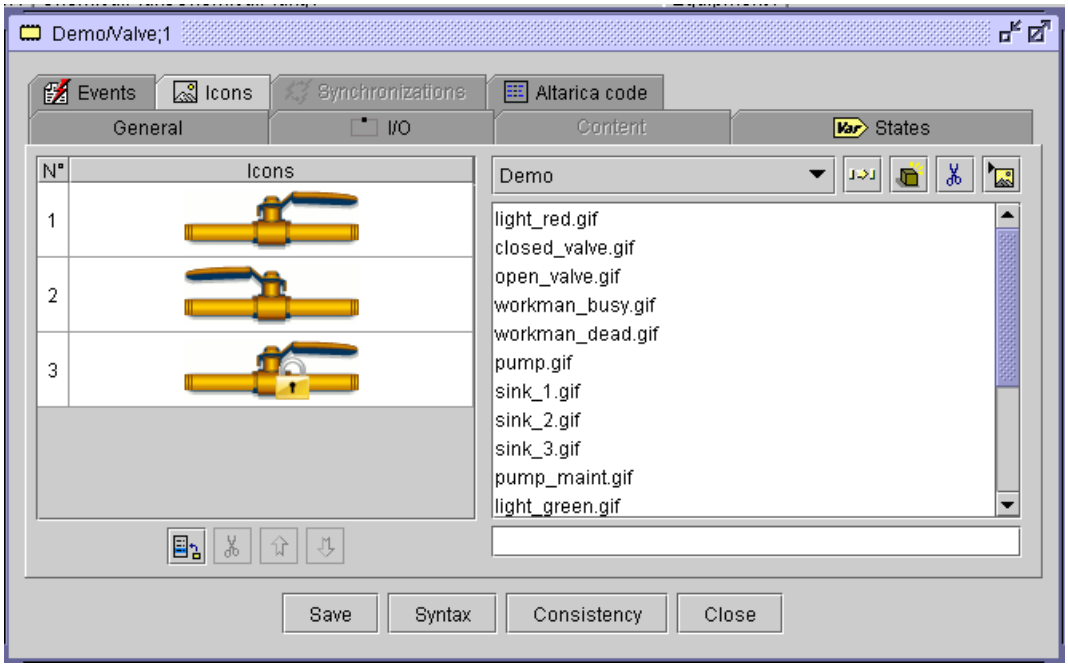


Figure 84 : Icons tab

- Click on , to remove the selected icon from the icon list.

  By using the two icons (and) you can move up or move down the selected icon in the list.

In Altarica code, the icon associated to a state is identified by a number. Thus, modifying the icon order in the list implies a change of icon in Altarica code.

When an icon is selected on the left part, the current directory is automatically updated in the drop-down menu.

When the component doesn't change graphical representation during the simulation, do not add icons in the list. The default icon is the icon selected in the **General** tab (default icon used for edition).

Remark : It is **recommended** that all icons of a component model have the same size.

Four commands are available beside the drop-down menu:

* Update:  ,
* New library:  ,
* Remove icon:  ,
* Import icons: 

Update

The button  allows updating the icon library when new icons are added.

Create a new icon library

Proceed as follows:

- Click on  .
- Enter the name of the new library, e.g. *"library_xxx".*
- Validate with the key **Enter**.
- Check that *"library_xxx"* has been created in the drop down menu.

Delete icons

Proceed as follows:

- Select icon to delete in the library,
- Click on  to delete the selected icon.

Import icons

Proceed as follows:

- Select a directory
- Click on  .
- Select the directory containing icons to be imported (Figure 85).

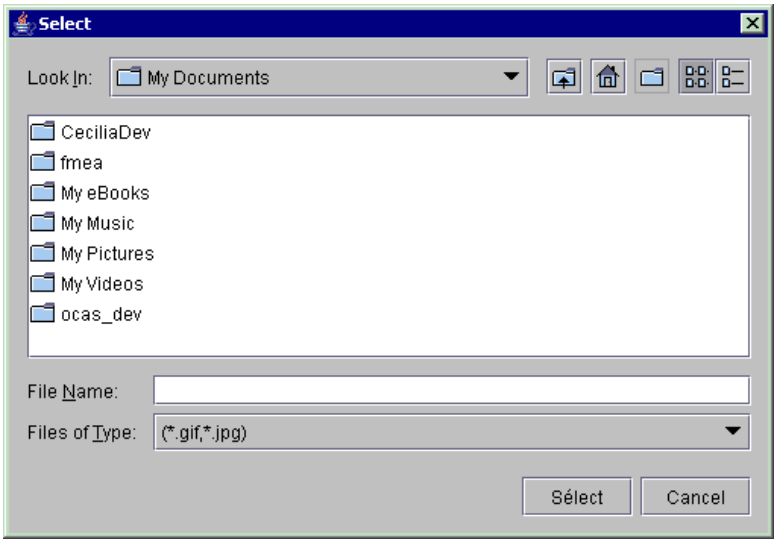


Figure 85 : File selection

Remark: ***Create new folder, List*** and ***Details*** are not usable from the window **Select**.

- Select the picture files ("*.gif" or "*.jpg").
- Click on button **Select**; the pictures are imported in the selected directory.

Note: The item **Images …** from the menu **Library** allows launching the icon manager (Figure 86) to manage, search and preview icons stored in library.

You can also use the following commands:

- Update : ,
- Create a new library : ,
- Delete icons :
- Import icons: .

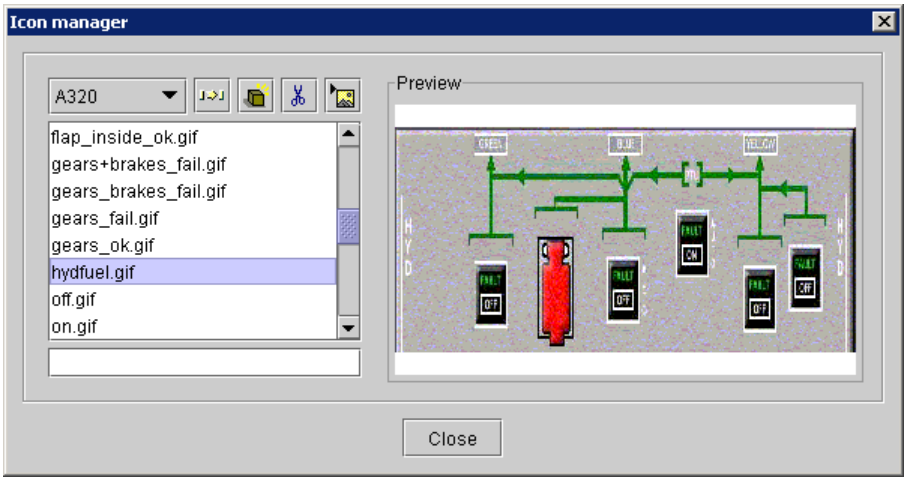


Figure 86 : Icon Manager

Altarica code tab

The **Altarica code** tab (Figure 87) allows defining the behavior for component models. The language dedicated to the functional/dysfunctional behavior and safety assessments is Altarica. Altarica is a state/transition language. For more information about the syntax of Altarica, please refer to the appendix 1: Syntax of Altarica code for component models.

The ***Altarica code*** tab contains two sub-tabs: **Code** and **Operators**.

Sub-tab Code

The sub-tab **Code** contains (Figure 87):

- A shortcut bar (Copy, Cut, Paste, Zoom in, Zoom out, Display initial states).
- An upper area describing a part of Altarica code. This code is generated automatically from the model declaration (node name) and the data defined in tab **States**, **I/O**, **Events** and **Icons**. This area contains the declaration of variables (states, flows, icons) and events. It is not possible to edit it directly.
- A lower area describing Altarica code composed with assertions (keyword *assert*) and transitions (keyword *trans*). Initially, this section is empty. It can be edited. User has to complete this part of Altarica code.
- A third area containg extern clauses can be displayed. It contains in particular the laws associated to the events. By default, this area is hidden. To display/hide it, use the tool 

NB: It is recommended to check often the syntax and consistency of Altarica code with the buttons **Syntax** and **Consistency**.

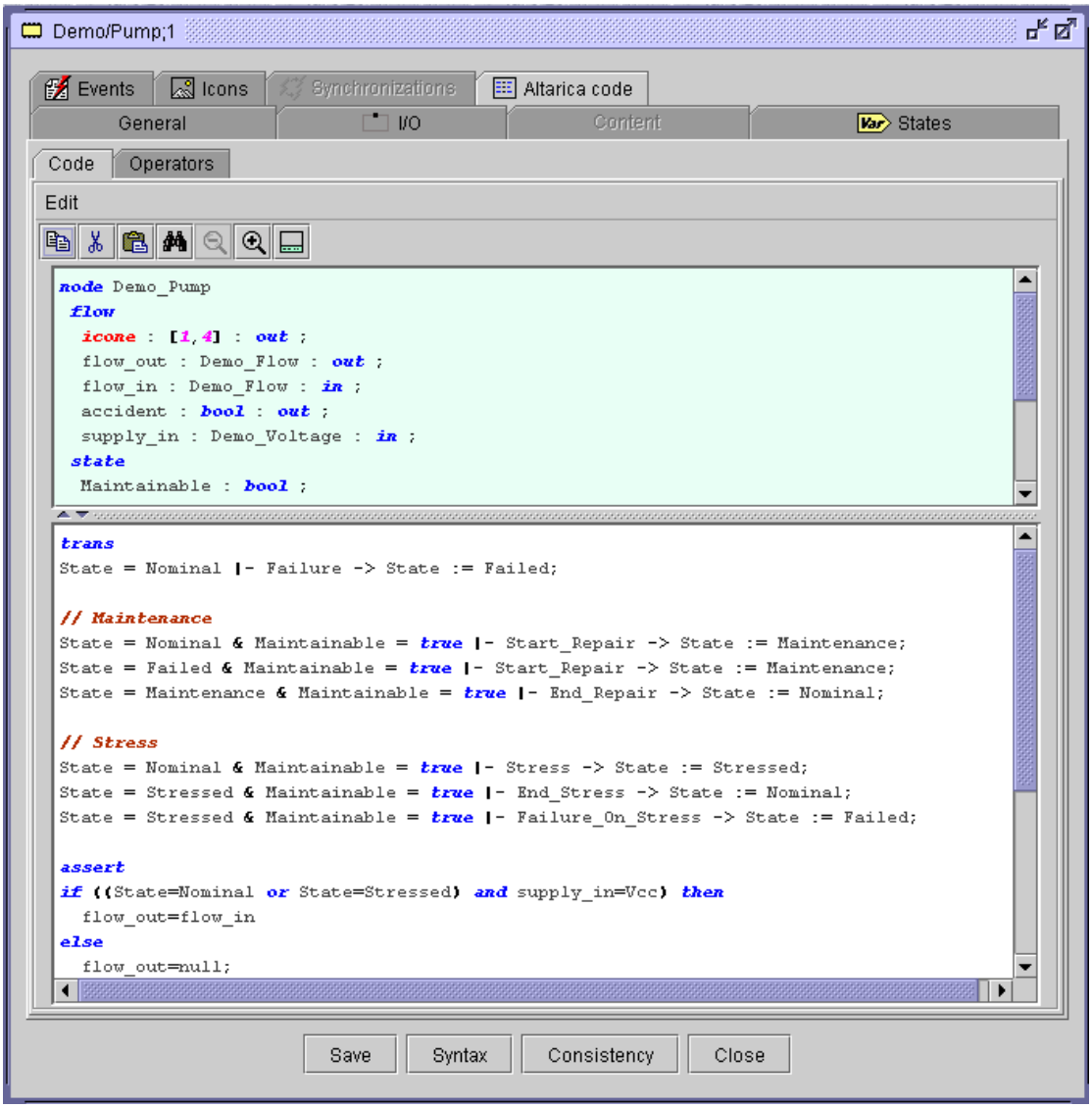


Figure 87 : Tab Altarica code– sub-tab Code

Sub-tab Operators

Operators can be created in library to improve the readability of the Altarica code. These operators can be used in Altarica code of component models. When an existing operator is used and recognized by the language parser, this operator is put in the list in sub-tab **Operators** (Figure 88).

Several different operators with a same name can coexist in the library:

- In different families
- In different versions

The sub-tab **Operators** allows selecting the suitable operator when several operators having a same name are available in the library.

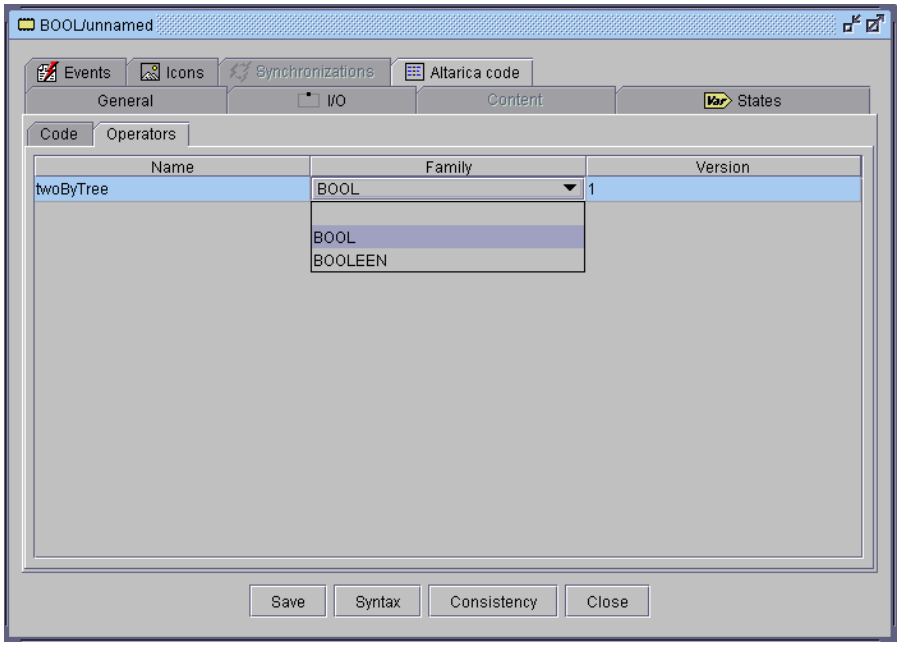


Figure 88 : Tab AtlaRica Code – Sub-tab Operators

NB: When an operator is used and recognized in Altarica code, it is displayed in green color.

Selecting variables

When user is currently editing Altarica code, it is possible to display and insert existing variables (states, flows, events, icons) declared in the model. These variables are displayed in a **Variable Selector**.

To open the Variable Selector:

- Use the shortcut **Ctrl+<SPACE>** during the edition of AltaRica code. The following window is open:

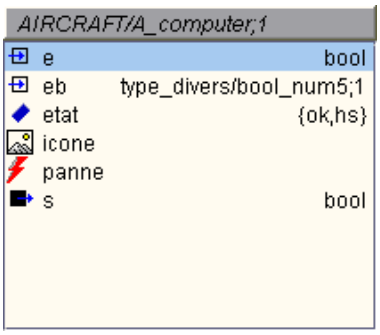


Figure 89 : Variable selector

To insert a variable, proceed as follows:

- Click in the Altarica edition area
- Use the shortcut **Ctrl+<SPACE>** to open the Variable Selector,
- Double-click on a variable to insert it in the code at the current cursor position**.**

For a variable having a **Record** type, a field can be inserted by this way:

- Type the symbol character "**^**" (**Alt gr**+<**^**>) after the variable name (var_name^), to display the list of fields:

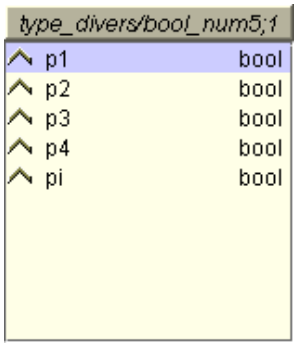


Figure 90: Selection of a field

- Double-click on a field to insert it after the record name (ex. **var_name^field**).

Default colours of Altarica keywords

.

To improve edition of Altarica code, keywords are displayed with specific colors when they are recognized by the language parser:

- Keyword (for example, trans, assert, and, or, if, then, else…) : blue,
- Numerical values: purple,
- Operator name: green,
- Icons: red,
- Punctuation: bold.

Checking syntax and consistency

Click on button **Syntax** (Figure 87), to check syntax of the Altarica code. If a syntax error is detected, an error window is displayed and describes precisely the kind of error and the line containing this error.

The most usual errors are:

- Syntax of a transition is not correct (operator **=** for condition, operator **:=** for assignment).
- An assertion is specifying an affectation between variables having a type not compatible (enumerated equal to a Boolean, …)
- The keywords « ***trans*** » or « ***assert*** » are omitted.
- The syntax of expressions between brackets is not correct. It must keep the following form:

```
(if expression1 then expression2 else expression3) ;
```

or :

```
(if expression1
        then expression2
        else (if expression3
                        then expression4
                        else expression5
              )
);
```

Note: "***if then else***" can be replaced by a "***case***" statement:

```
flow-var = (case { bool-exp1 : expression_1 ,
                           |
                   bool-expn : expression_k ,
                 else expression-else  } );
```

A good indentation of the code in the expressions ***if then else*** can generally avoid problems. Each opened bracket must correspond to a closed bracket.

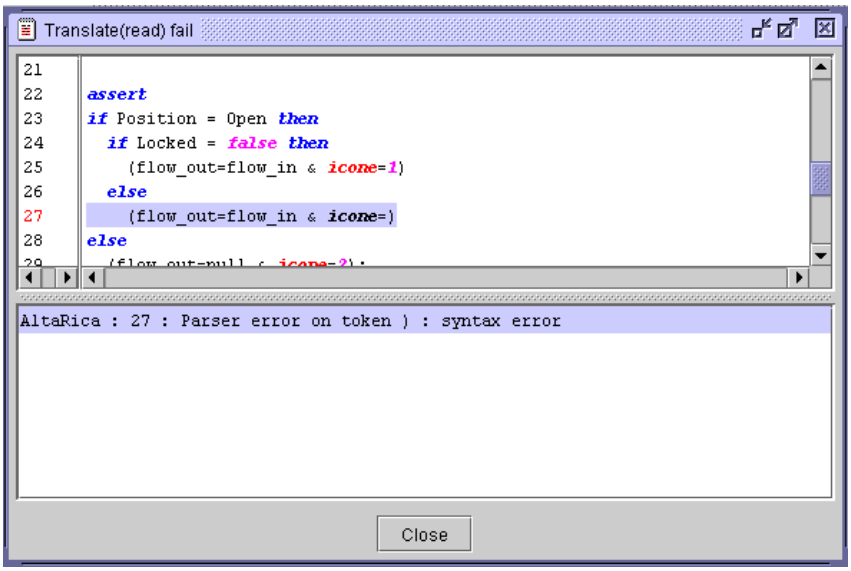


Figure 91: Example of error message

Click on button **Consistency** (Figure 87), to check the consistency of Altarica code. If a consistency error is detected, an error message is displayed and describes precisely the kind of consistency error and the line containing the error.

The most usual errors are:

- An output flow is not assigned for each state of the component.
- An output flow is assigned with two different values for a same state.

Edit a component model

To edit a component model proceed as follows:

- Expand the folder in the component library to display the family containing the model to edit
- Left click on the model to select it,
- Click on ⊞ to display the versions (Figure 92):

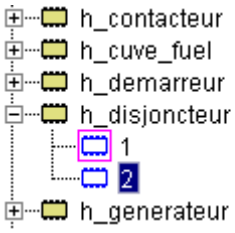


Figure 92 : Versions of a component model

- A component can be edited by two ways :
  - Double click on the version number
  - Select the command **Edit model** from the menu **Library** or use command **Edit model** in the contextual menu (right click).
- A bar graph ***Reading model…*** (Figure 93) is displayed:

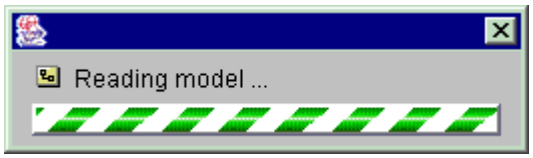


Figure 93 : bar graph during model Reading

When a model is edited by a user, this model becomes locked.
In multi user mode, this mechanism allows managing concurrent access. A locked model can be open in read only mode.
The symbol 2 in the tree structure means that the component model is currently edited. When another user tries to open a model in edition, the following message is displayed:

Figure 94: Component in edition

Remark: A model can remain locked after the session closing, for example if the OS crashes; in this case, it is possible to unlock manually the model by opening the property window (Figure 95). It is available with the command **Properties** from the contextual menu or from the menu **File**.

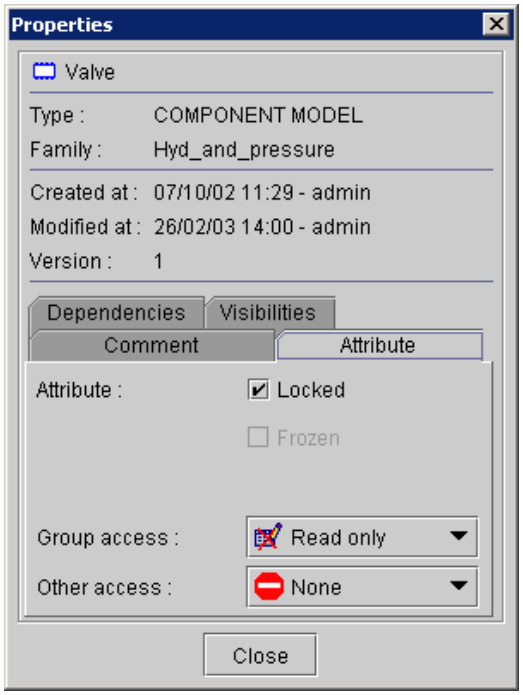


Figure 95 : Component properties – lock attribute

Rename a component model

To rename a component model:

- In the **Components** tab, click on the model to rename.
- In the contextual menu, select the command **Rename model**; the field **name** of the component becomes available.
- Enter the new model name and validate with **Enter**.

Remark: If the name already exists, an error message is displayed. Click on **OK** and enter a new name, then validate with **Enter**.

Warning: This command is **high-risk** and must be used with caution, because the renaming leads to a reference modification of the model in system architectures/equipments using this reference. To keep valid the links between the component reference and the component instantiations in equipments and systems, proceed as follows:

- Open all equipments and architectures containing the component **before** renaming it,
- Rename the component model,
- Save all equipments and architectures **after** renaming.

If a component is renamed when the architecture using it is closed, the component becomes an unknown model in this architecture. Cf. § *Removed links* and § Rename a component model.

Duplicate a component model

To duplicate a component model:

- Click on the model to duplicate in the **Component** tab,
- Right click on the model to open the contextual menu,
- Select the command **Model … → Duplicate**
- A bar graph (Figure 96) indicates that the component is currently duplicated; a copy of the model is then created with the extension *_copy*.

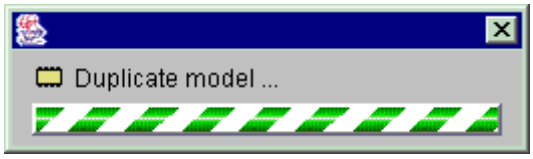


Figure 96 : Bar graph during duplication

- The default name can be changed.

Warning : If several model versions exist, only in the last version is duplicated. To duplicate another version of the component, user must select the right version in the component tree structure (Figure 92) before launching the command **Model … → Duplicate**.

Remark : To move a component model in another folder:

- Copy the component with the command **Model … → Copy**,
- Paste the component in the destination folder : command **Model … → Paste**,
- Restore the original name of the component (delete the extension « *_copy* ») in the destination folder.

Create a new version of a component model

To create a new version of a component model:

- Click on the model in the **Component** tab,
- Right click on the model to open the contextual menu,
- Select the command Model … → New version,
- A bar graph (Figure 97) indicates that the new version is currently creating.

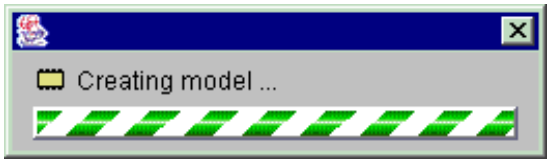


Figure 97 : Creation of a new version

When the new component version has been created, it is stored in the component library (Figure 92) below the previous model version.

Freeze a component model

Warning: This command is **high-risk** and must be used with caution, because it is irreversible: modifications are not possible on a frozen model.

To freeze a component model:

- Click on the model in the **Component** tab,
- Click on ⊞ to display the available versions,
- Left click on the version number to select it,
- Right click on the model to open the contextual menu,
- Select the command **Model …** → **Freeze**.

Remark: The frozen versions are represented by the symbol in the library. This ***Frozen*** attribute can be also visualized in the model properties (Figure 98); select command **Properties** from the contextual menu or from menu **File**.

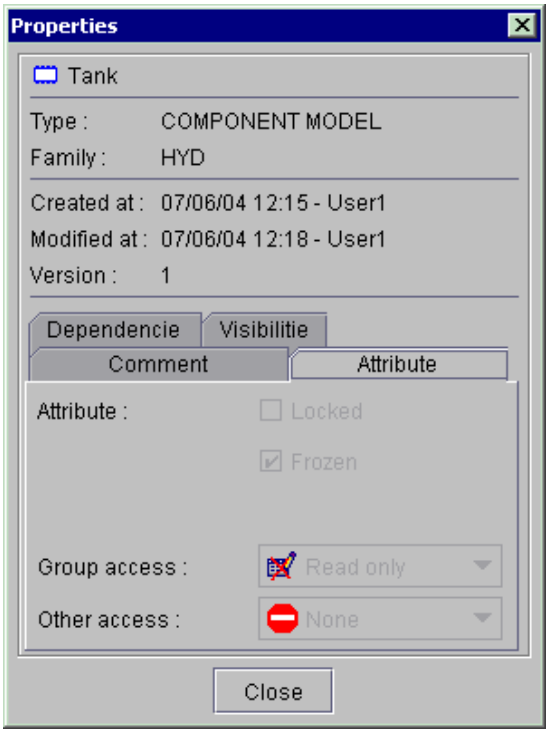


Figure 98 : Model properties

Delete a component model

Warning: This command is **high-risk** and must be used with caution, because it is irreversible. The links between the component and the other models are deleted. .

To delete a component model:

- Click on the model in the **Component** tab,
- Click on ⊞ to display the available versions,
- Left click on the version number to select it,
- Right click on the model to open the contextual menu,
- Select the command **Delete model**; a *dialog box is open:*

Figure 99: Dialog box for model removing

- Click on **Continue** to accept deletion, or on **Cancel** to cancel deletion.

EDIT AN EQUIPMENT

Create a new equipment model

To create a new equipment model:

- In the left part of the window, click on tab **Equipments,**
- Select the family in which the equipment will be created.
- Right click on the family to display the contextual menu.
- Click on command **Create new model**; the following bar graph is displayed (Figure 100).

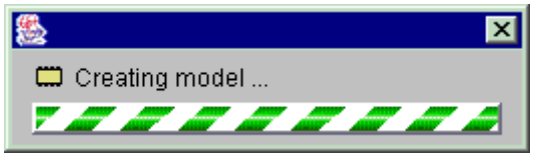


Figure 100 : Bar graph during the equipment creation

- The equipment editor (Figure 101) is open
- Complete information in the tabs.

The editor is composed with the following tabs:

- General,
- I/O,
- Content,
- Icons,
- Synchronizations,
- Altarica code.

When the new model is saved, it is automatically stored in the library (alphabetically) with its version.

Remark : The default version of a new equipment is 1.

General tab

The **General** tab (Figure 101) contains the following information:

- **Name**: enter the model name,

  Note: This field is mandatory to save the model in library.

- **Width**: default value of icon width,
- **Height**: default value of icon height,
- **Icon file**: click on the browser button to choose a default icon for the model.

Remark: The file format for icon is "***.gif**" or "***.jpg**".

- Check the box **Draw border** to display or a black border around the icon.
- Check the box Move ports automatically when components are reshaped.
- Click on **Resize** button to update, if necessary, the modifications of icon size.
- **Comment**: to add a comment to the model.

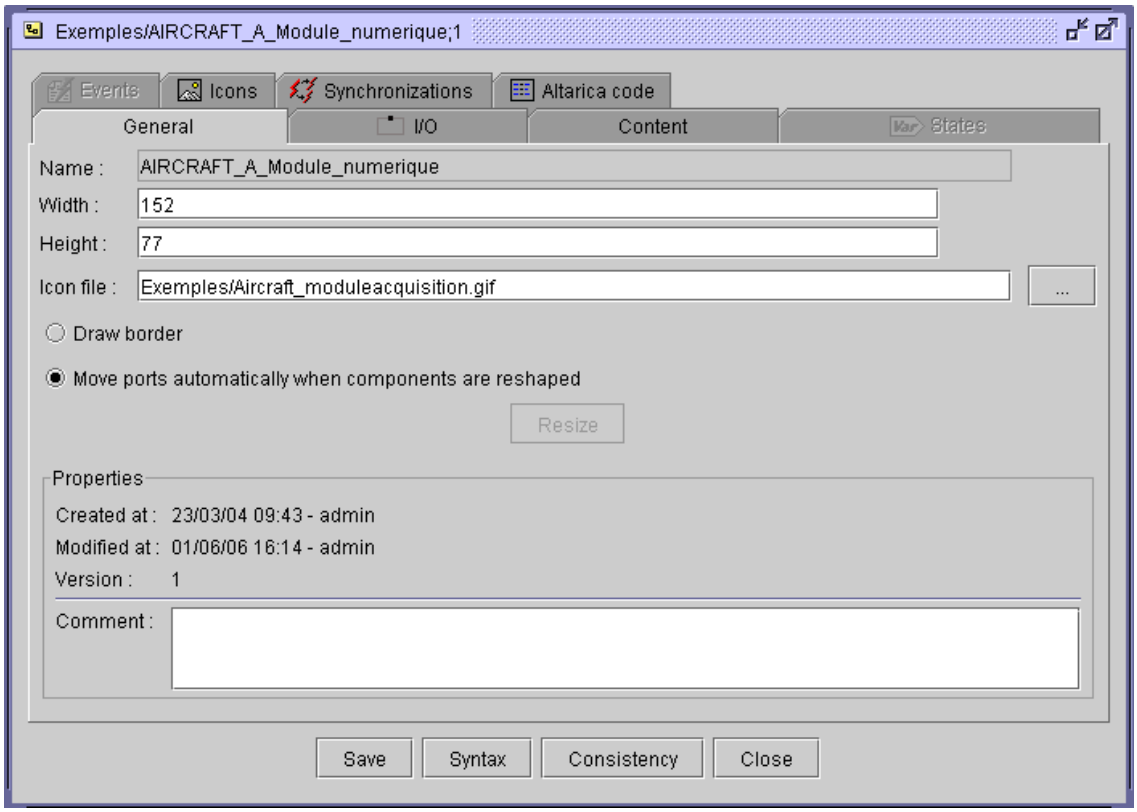


Figure 101 : Equipment - General tab

The equipment editor contains four buttons:

- **Save**: to save the model in the database,
- **Syntax**: to check syntax of Altarica code,
- **Consistency**: to check consistency of Altarica code,
- **Close**: to close the equipment editor.

Two shortcuts on the top right side allow modifying the window size:

- Reduce,

- Extend.

I/O tab

The **I/O** (Inputs/Outputs) tab allows to define and manage the input/output flow variables and to ports of the equipment.

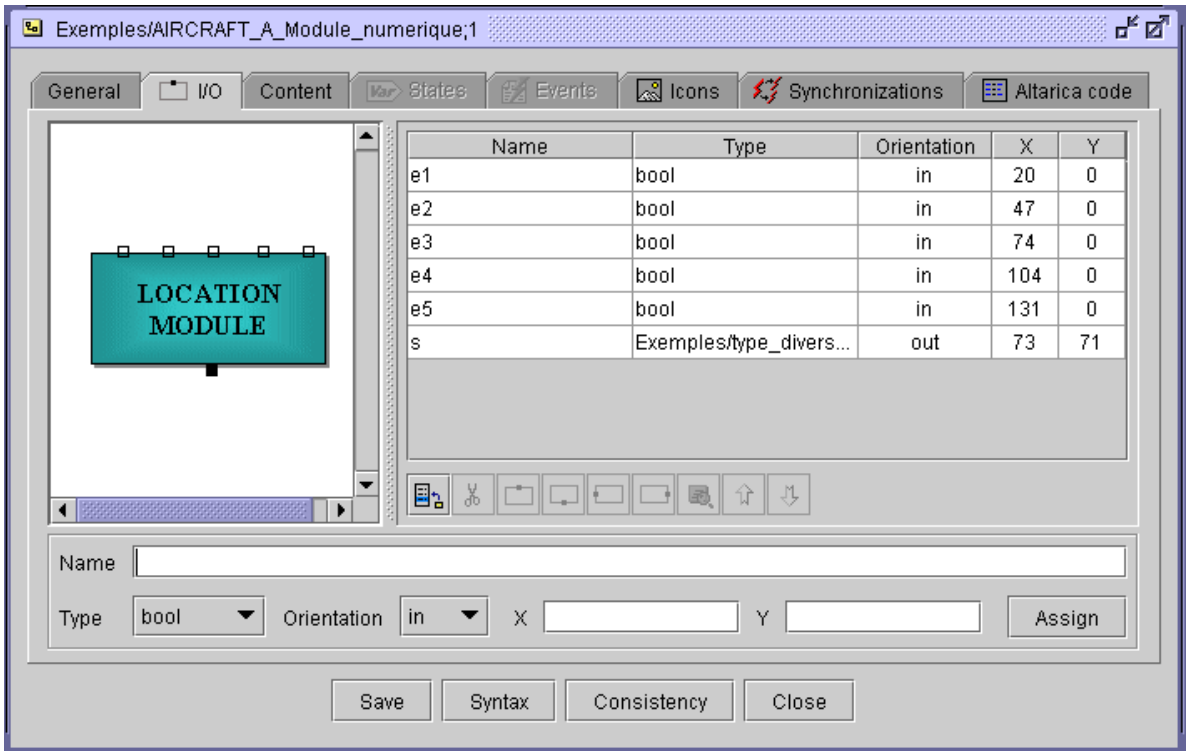


Figure 102 : Equipment - I/O tab

To create a new port on the equipment

- In the field **Name**, enter the name of the flow to be created
- Use the menu **Type** to select the type of the flow:
- ***bool:*** *for* boolean type, the possible values are **true** and **false**
- ***enum*****:** for enumerate type, values **must** be separated by commas without spaces, e.g. : *null,low,high*.
- ***bound:*** *v*ariable is an integer having a value between a lower bound and a upper bound,
- ***predefine:*** predefined type created in library
- ***int :*** integer type
- ***float :*** float type

N.B: The “space” character is forbidden between the *enumerated* values.

- In the menu **Orientation**, select the direction of the port: **in** (input), **out** (output) or **local**.
- Enter the coordinates of the port in the fields X and Y.
- Click on icon  or on key **Enter**; a new port is added in the list of I/O ports. The port is added graphically around the icon.

N.B: An input is identified by a white rectangle and an output by a black one.

-

NB 2: If an **I/O** is selected, the new **I/O** is added above the selected **I/O**. If no **I/O** is selected, the new **I/O** is added at the end of the list.

NB 3: User can re-order the list of the port. To move a port down or up in the list, click on the following tools:

* Move up  ,
* Move down  .

To modify the position of a port:

-
- Use the following tools to move the port automatically in the middle of a border:
  * ***Top:***  icon,
  * ***Down:***  icon,
  * ***Left:***  icon,
  * ***Right:***  icon.

- Or use the mouse to move the port I/O anywhere on the equipment border.
- Or double-click on the coordinates to edit the fields X and Y in the list of I/O.

To edit a port:

- Click on the port to select it in the list. The fields Name, Type, Orientation, and Coordinates are updated and can be modified.
- Modify the features of the port
- Click on **Assign** to update the port in the list.

To modify the name of a port

To modify the name of a port, double-click on the name of the port in the list. The name can be edited.
Then, click on **Enter** to validate the new name.

To modify a predefine type of a port:

User can create predefined types in the library. When a new port is added, it is possible to associate a predefined type from the library. The **I/O** tab of the editor allows to modify easily a predefined type:

- Right click on a port to display the contextual menu (Figure 103) and select **Edit Type**.

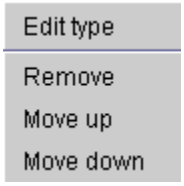


Figure 103 : Contextual menu

- Or click on the icon in the tool bar.
- Or double-click on the path name of the type in the list of ports.

To delete an existing port:

- Select the port in the list and click on icon to remove it from the equipment.
- Or right click on a port to display the contextual menu (Figure 103) and select **Remove.**
- Or click on key **SUPPR.**

Content tab

N.B: When equipment is created, it contains the Input/Output ports added in the **I/O** tab.

The **Content** tab (Figure 104) contains the equipment architecture composed with components and/or other equipments; modeling an equipment and a system architecture are similar.

Use Drag & Drop to insert models of components and equipments in the architecture content of the equipment from the library. Connect graphically the models by selecting input port with output port if they have a compatible flow (same type).

NB: it is possible to create not graphical connections by defining assertions in tab **Altarica code**.

The Content tab has two buttons:

- The **Layers** button allows users to work on several layers. Thus, some parts of the architecture can be defined of different layers. Then, it is possible to hide or display them.
- The **Display** button allows users to display or hide some information:
  - *none* : model names are hidden
  - *Node labels*: name of models (components and equipments of the content) and name of the equipment ports are displayed.
  - *Link labels*: names of types are displayed on the graphical connections.
  - All labels: all names are displayed (models, equipment ports and types).

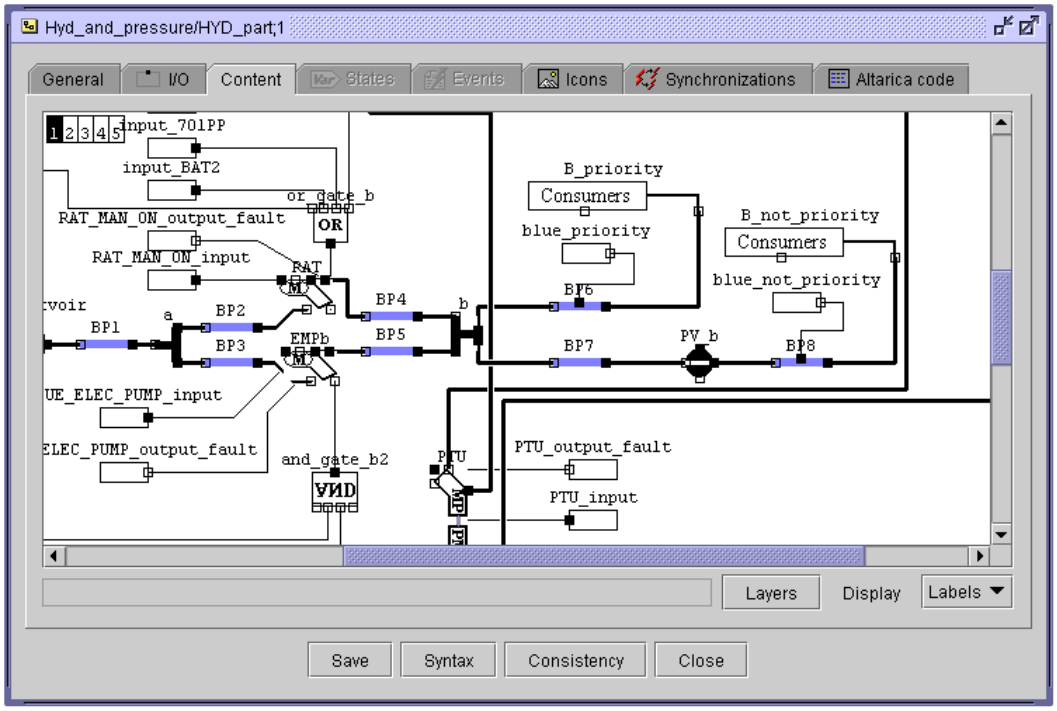


Figure 104 : Equipment – Content tab

Synchronization tab

The tab **Synchronization** (Figure 105) allows creating and managing synchronizations. The occurrence of a synchronization event corresponds to the simultaneous occurrence of several basic events defined in the equipment architecture (**Content** tab).

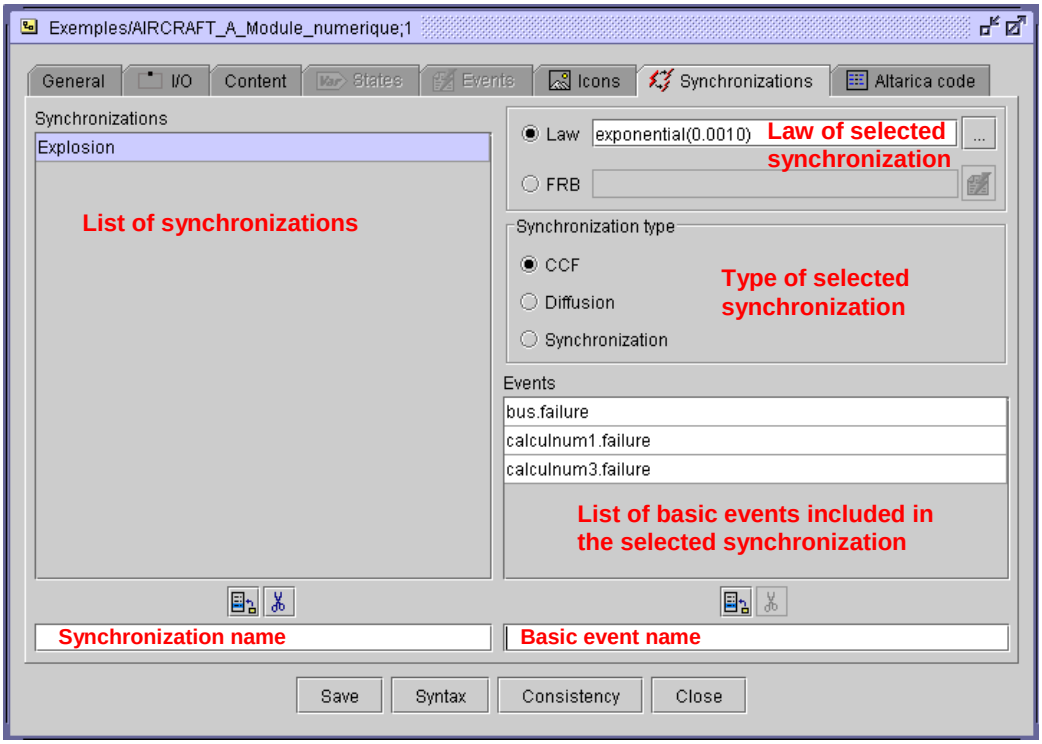


Figure 105 : Synchronization tab

To create a synchronization event:

- Enter the name of the new synchronization event in the left edition field,
- Click on the left command , to add the synchronization to the list,
- Define a probability law associated to the synchronization:
- Check box **Law** to define a law manually,
- Check box FRB to associate an existing law from the library (Failure Rate Bank). For more information about FRB, please refer to the appendix SD9_Appendix_FRB_R19_D7.pdf

- Select the type of synchronization:
- CCF (Common Cause Failure)
- Diffusion
- Synchronization (“Rendez-Vous”)
- Enter the name of the basic event with its hierarchy in the equipment. Click on the right command , to add the basic event to the current synchronization. Repeat this step for each basic event to include in the synchronization.

NB: To make the selection of a basic event with its hierarchy easier, you can use the shortcut **Ctrl+<SPACE>** to display the list of models contained in the equipment and then the list of basic events for a selected model.

To select a basic event:

- Click in the right edition field,

- Use the shortcut **Ctrl+<SPACE>** to display the models of the equipment content (Figure 106)
-

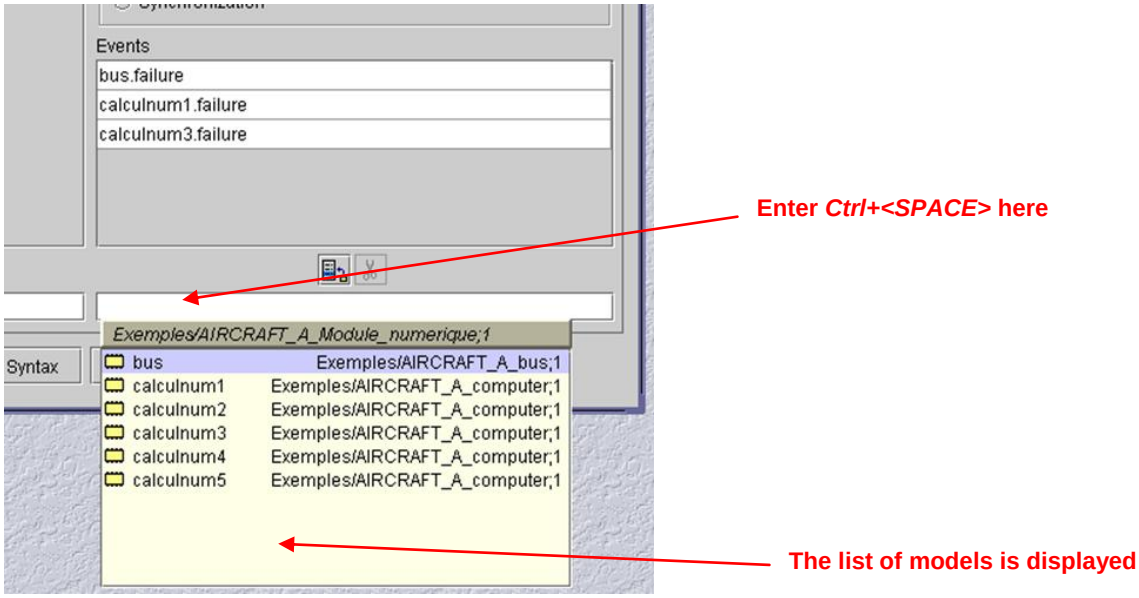


Figure 106: Selecting a model with Ctrl+<SPACE>

- Select the component or equipment containing the event to include in the synchronization,
- Double-click on this model, its name **<Model_Name>** is then inserted in the edition field,
- Type the character **<.>** (point) after the model name **<Model_Name>.** to display the list of events if the model is a component (Figure 107), or the list of sub-models if the model is an equipment.
- Repeat the previous step until an event is selected.
- Click on the right command , to add the event to the current synchronization.

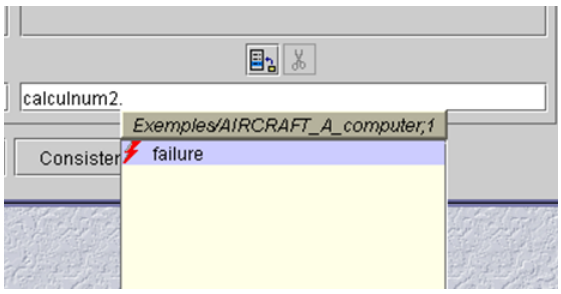


Figure 107: Selecting a basic event

NB: During a simulation, user can inject synchronization in the model, like basic event.

NB2: Probability laws are used to quantify fault trees. The general syntax of a law is:
- *<Law_Name>* (*<Parameter_List>*) ; parameters are separated with comma.

NB3: The usual probability laws are described in the following table:

<Law_Name> (<Parameter_List>)	Example
constant (<probability>)	constant(0.5)
exponential(<lambda>)	exponential(1e-6)
GLM (<gamma> ,<lambda>, <mu>)	GLM(0.5,1e-6,1e-3)
GLM_asymptotic (<lambda>,<mu>)	GLM_asymptotic(1e-6,1e-3)
Weibull(<alpha>,<beta>,<t0>)	Weibull(0.5,1.5,0.5)
Periodic_test(<lambda>,<period>,<t0>)	periodic_test(1e-3,10,0)
Periodic_test_2(<lambda>,<lambda*>, <mu>, <tau>, <theta>, <gamma>, <pi>, <X>, <sigma>, <omega>)	periodic_test_2(1e-4, 1e-3, 0.125, 72, 72, 0,0, 1, 1, 0)
dormant (<lambda>, <MTTR>, <period>)	dormant (1e-3, 10, 25)
CMT(<lambda>, <duration>, <probability>)	CMT (1e-3, 72, 0)
NRD (<mu>, <delay>)	NRD(0.125, 10)

Table 2: Usual probability laws

Altarica code tab

Altarica code in equipment (Figure 108) allows describing equipment assertions. Contrary to components, transitions are not allowed in equipment code.

Tab **Altarica code** contains two sub-tabs: **Code** and **Operators**.

Sub-tab **Operators** describes the list of Operators used in sub-tab **Code**. User can refer to the presentation of sub-tab **Operators** of components (§. Sub-tab Operators).

Sub-tab **Code** contains:

- A shortcut bar (Copy, Cut, Paste, Zoom in, Zoom out, Display external clauses).
- An upper area describing a part of Altarica code. This code is generated automatically from the model declaration (node name) and the data defined in tab **States**, **I/O**, **Synchronizations** and **Icons**. This area contains the declaration of variables (flows, icons) and synchronizations. It is not possible to edit it directly.
- A lower area describing Altarica code composed with assertions (keyword *assert*). Initially, this section is empty. It can be edited. User has to complete this part of Altarica code with *assertions*.
- A third area containg extern clauses can be displayed. It contains in particular the laws associated to the synchronizations. By default, this area is hidden. To display/hide it, use the tool 

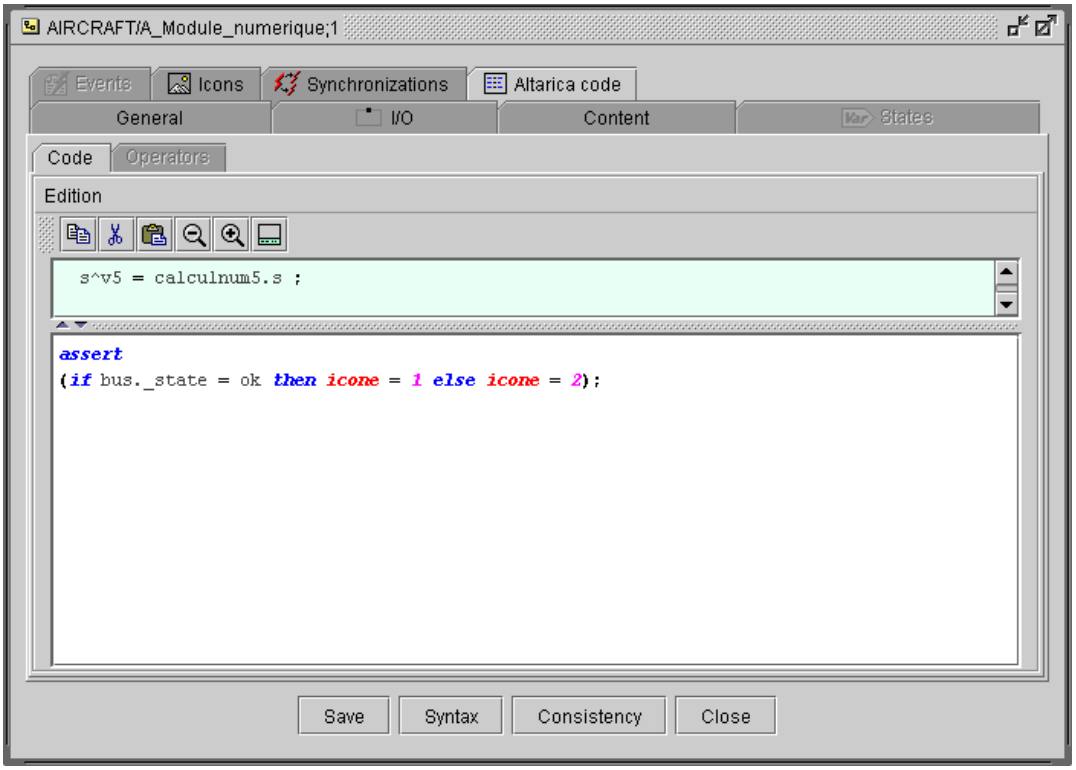


Figure 108 : Equipment - Code Altarica code

Assertions

Assertions described in tab **Altarica code** are used:

- to define the value of icons (loaded in tab **Icons**) according to value of variables of the equipment architecture. These variables can be states, flows of components included in the equipment content. Icon values are assigned by using assertions on model variables. Icons are used during the simulation.
- To define links (connections) between models (components, equipments) contained in the equipment (tab **Content**). When assertions are used to define links, these connections have not a graphical representation. This feature can be very useful to hide connections which have not a physical meaning, for example to connect an Observer to a physical architecture. An assertion can connect two models which are instantiated in different hierarchical level.

Use the edition area in tab **Altarica code** to define assertions.
The declaration of assertions must begin with the keyword **assert**.

NB: During the edition of Altarica code, it is possible to display and insert variables of the equipment content in the code. These variables are defined in the models of the equipment architecture (flows, states). To display the variables (ordered according the architecture structure), use the shortcut **Ctrl+<SPACE>.** To display and insert a variable, proceed as follows:

- Click in the Altarica edition area
- Use the shortcut **Ctrl+<SPACE>** to open the Variable Selector (Figure 109),

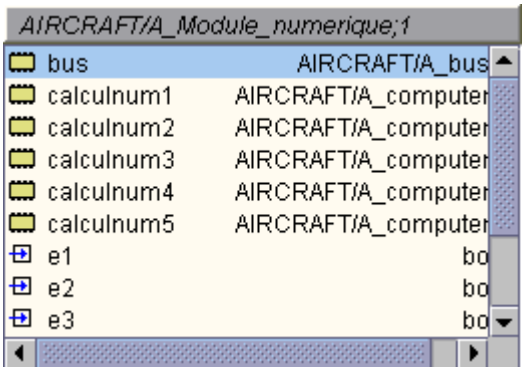


Figure 109 : Altarica Code – Variables selector

- Double-click on a variable to insert it in the code at the current cursor position
- Enter the character <.> (point) to insert the rest of the model and variable path.
-
- For a variable having a **Record** type, a field can be inserted by this way:
- Type the symbol character "^" (**Alt gr**+<^>) after the variable name (var_name^), to display the list of fields
- Double-click on a field to insert it after the record name (ex. **var_name^field**).

Syntax of assertions:

* Simple assignment between “flow in” and “flow out” having a same type.

<component_i>.<In_Name> = <component j>.<OutName>;

- Simple assignment between “flow in” and a value:

  <component_i>.<In_Name> = true ; ***(Boolean)***

  <component_i>.<In_Name> = low ; ***(enumerated)***

- Assignment using logical expressions or operators:

  <component_i>.<In_Name> = (<component j>.<OutName> or
  <component k>.<OutName>);

  <component_i>.<In_Name> = my_operator (<component _j>.< OutName >,
  < component _k>.<OutName S>);

For more information about Altarica syntax and assertions, please refer to APPENDIX 1: SYNTAX OF Altarica.

Don’t forget to check the assertions by clicking on button **Syntax** and **Consistency**. If an error is detected, an error message is displayed and describes precisely the kind of error and the line containing it.

Edit an equipment model

To edit an equipment model proceed as follows:

- Expand the folder in the equipment library to display the family containing the model to edit
- Left click on the model to select it,
- Click on ⊞ to display the versions (Figure 110),

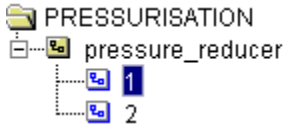
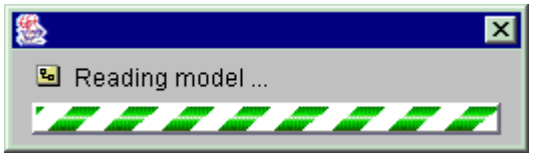


Figure 110 : Versions of an equipment

An equipment can be edited by two ways :

- Double click on the version number
- Select the command **Edit** model from the menu **Library** or use command **Edit model** in the contextual menu (right click).

- A bar graph ***Reading model…*** is displayed:



When a model is edited by a user, this model becomes locked.
In multi user mode, this mechanism allows managing concurrent access. A locked model can be open in read only mode.

The symbol  in the library means that the equipment is currently edited. When another user tries to open a model in edition, the following message is displayed:

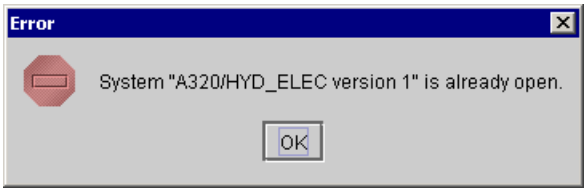


Figure 111: Equipment in edition

Remark: A model can remain locked after the session closing, for example if the OS crashes; in this case, it is possible to unlock manually the model by opening the property window (Figure 112). It is available with the command **Properties** from the contextual menu or from the menu **File**.

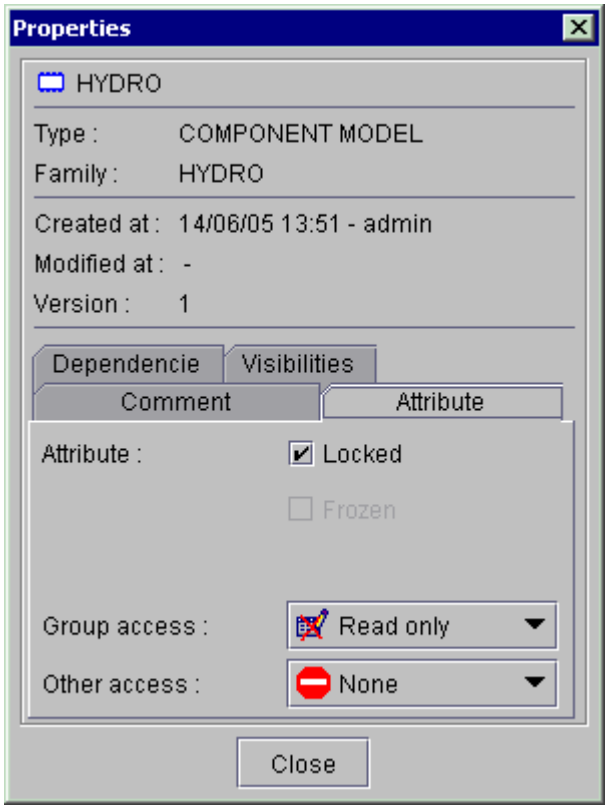


Figure 112 : Equipment properties - Lock attribute

Rename an equipment model

To rename an equipment model:

- In the **Equipments** tab, click on the model to rename.
- From the contextual menu, select the command **Rename model**; the field **name** of the equipment becomes available.

- Enter the new model name and validate with **Enter**.

Remark: If the name already exists, an error message is displayed. Click on **OK** and enter a new name, then validate with **Enter**.

Warning: This command is **high-risk** and must be used with caution, because the renaming leads to a reference modification of the model in system architectures/equipments using this reference. To keep valid the links between the equipment reference and the equipment instantiations, proceed as follows:

- Open all equipments and architectures containing the equipment reference **before** renaming it,
- Rename the equipment model,
- Save all other equipments and architectures **after** renaming.

If an equipment is renamed when the architecture using it is closed, the equipment becomes an unknown model in this architecture. Cf. § *Removed links*.

Duplicate an equipment model

To duplicate a component model:

- Click on the model to duplicate in the **Equipment** tab,
- Right click on the model to open the contextual menu,
- Select the command **Model … → Duplicate**
- A bar graph (Figure 113) indicates that the equipment is currently duplicated; a copy of the model is then created with the extension *_copy*.

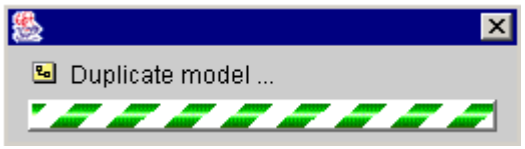


Figure 113 : Bar graph during duplication

- The default name can be changed.

Warning : If several model versions exist, only in the last version is duplicated. To duplicate another version of the equipment, user must select the right version in the equipment tree structure (Figure 110) before launching the command **Model … → Duplicate**.

Remark : To move an equipment model in another folder:

- Copy the equipment with the command **Model … → Copy**,
- Paste the equipment in the destination folder : command **Model … → Paste**,
- Restore the original name of the equipment (delete the extension « *_copy* ») in the destination folder.

Create a new version of an equipment model

To create a new version of an equipment model:

- Click on the model in the **Equipments** tab,
- Right click on the model to open the contextual menu,
- Select the command **Model … → New version**,
- A bar graph (Figure 114) indicates that the new version is currently creating.

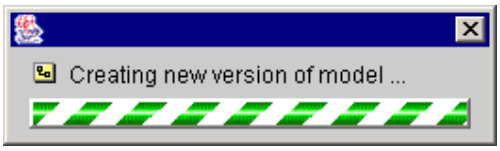


Figure 114 : Bar graph during the new version creation

When the new equipment version has been created, it is stored in the equipment library (Figure 92) below the previous model version.

Freeze an equipment model

Warning: This command is **high-risk** and must be used with caution, because it is irreversible: modifications are not possible on a frozen model.

To freeze an equipment model:

- Click on the model in the **Equipment** tab,
- Click on to display the available versions,
- Left click on the version number to select it,
- Right click on the model to open the contextual menu,
- Select the command **Model … → Freeze**.

Remark: The frozen versions are represented by the symbol in the library. This ***Frozen*** attribute can be also visualized in the model properties (Figure 115); select command **Properties** from the contextual menu or from menu **File**.

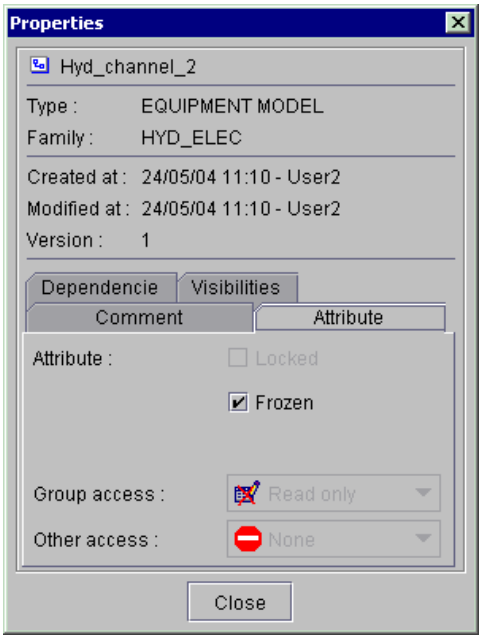


Figure 115 : Model properties

Delete an equipment model

Warning: This command is **high-risk** and must be used with caution, because it is irreversible. The links between the equipment and the other models are deleted.

To delete an equipment model:

- Click on the model in the **Equipment** tab,
- Click on ⊞ to display the available versions,
- Left click on the version number to select it,
- Right click on the model to open the contextual menu,
- Select the command **Delete model**; a *dialog box is open:*

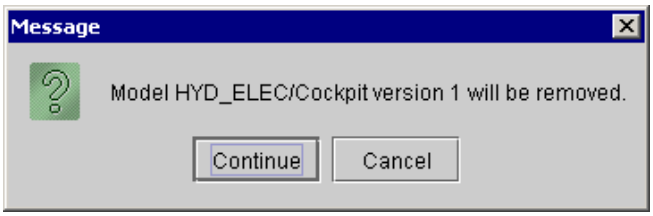


Figure 116 : Dialog box for model removing

- Click on **Continue** to accept deletion, or on **Cancel** to cancel deletion.

EDIT AN OPERATOR

Create a new operator model

Definition

Operators are used to simply the writing of assertions in Altarica code. An operator corresponds to a function between output flow and input flow(s). Contrary to a component model, an operator doesn't contain transitions.

Remark: the operators can be used only in component (or equipment) assertions. Using operators in transitions (Altarica code) is not allowed.

To create a new operator model:

- In the left part of the window, click on tab **Operators,**
- Select the family (folder) in which the equipment will be created.
- Right click on the family to display the contextual menu.
- Click on command **Create new model**; the following bar graph is displayed (Figure 117) :

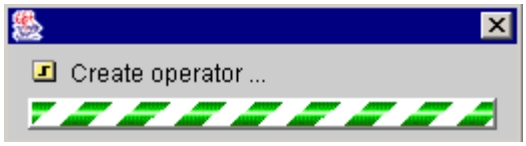


Figure 117 : Bar graph during the operator creation

- The *Operator editor* (Figure 118) is open; it contains three tabs :
  - **General**,
  - **Properties**,
  - **Altarica code**.
- Complete information in the tabs.

When the new model is saved, it is automatically stored in the library (alphabetically) with its version.

Remark : The default version of a new operator is 1.

An operator is defined by a relation between an output flow and one or several input flow(s). The input flow(s) can be seen as arguments of the function *operator*. These arguments are called *Operands*. The relation between the operator output and the operands is expressed with assertions. When user creates a new function operator, he has to complete the following information:

- Name of the *Operator*: this name is used to define the operator and the output result
- Name of the *Operand(s):* it corresponds to the input flow(s)
- *Relation* between the Operand(s) and the Operator (result): in Altarica code (assertions)

General tab

The **General** tab (Figure 118) contains the following information:

- **Name** (in the top edition field): enter the operator name, e.g. *o2by3*.
- The list of operands with their type. The first line is allocated to the operator result and is displayed in red. This line is mandatory and can't be removed or modified.

Remark: The name of the variable "operator result" is identical to the operator function.

- **Name** (in the bottom edition field): enter the name of the operand.
- Use ***Type*** menu to define the operand type.
- Click on command  or valid with **Enter** to add the operand in the current list.
- To modify the type of an existing operand, select it in the list, modify the type, and click on **Assign** to update the type.

Remark: Type of an operator/operands can be a record, but the orientation must be "normal" and without crossing (cf. EDIT A TYPE).

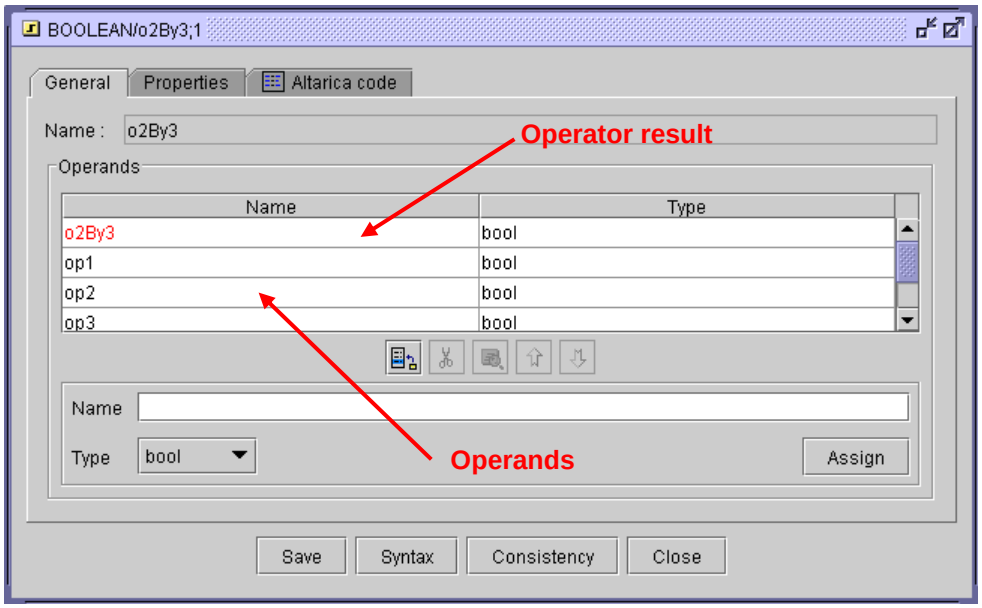


Figure 118 : Operator editor - General tab

Properties tab

- The Properties tab (Figure 119) contains the following information:
  - Creation date,
  - Last modification date,
  - Version.
- The **Comment** tab is used to add a possible comment to the operator.

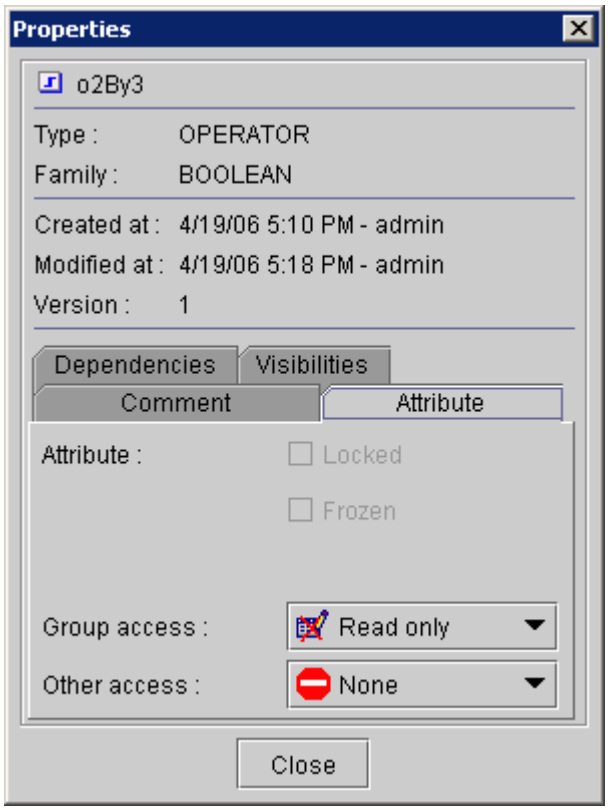


Figure 119 : Operator – Properties tab

Altarica code tab

The sub-tab **Code** contains:

- A shortcut bar (Copy, Cut, Paste, Zoom in, Zoom out, Display external clauses).
- An upper area describing a part of Altarica code. This code is generated automatically from the model declaration (operator name with the keyword **func**) and the data defined in tab **General**. This area contains the declaration of variables (operator, operands with their type). It is not possible to edit it directly.

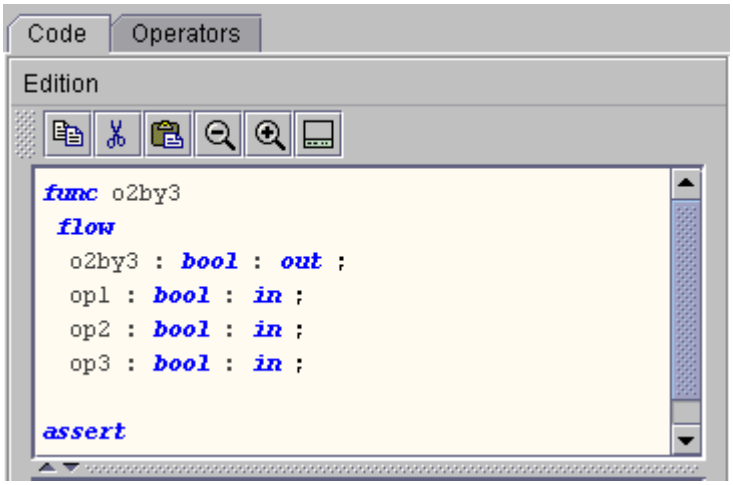


Figure 120 : Operator – Altarica code – Operands

- A lower area describing Altarica code composed with assertions. The keyword *assert* is inserted automatically. Initially, this section is empty. It can be edited. User has to complete this part of Altarica code with *assertions* to define the relation between the operator and the operands (cf. Annexe 1: Syntax of Altarica code):

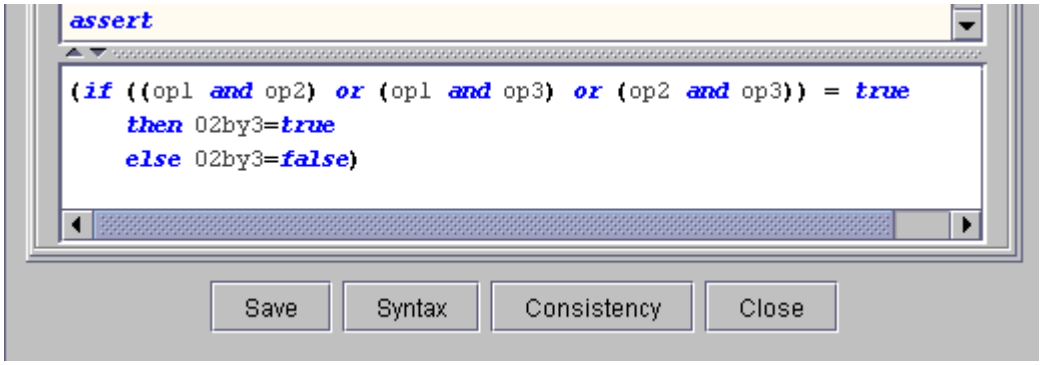


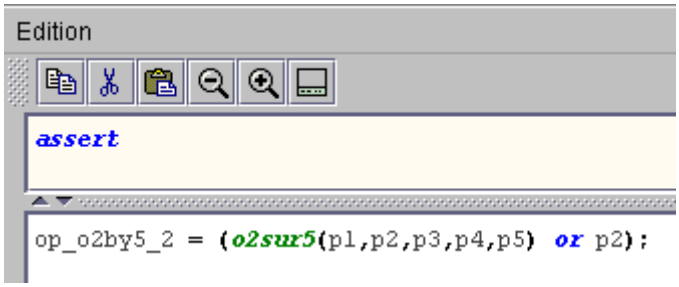
Figure 121 : Operator editor - Altarica code tab

- A third area containg extern clauses can be displayed. It contains in particular the laws associated to the synchronizations. By default, this area is hidden. To display/hide it, use the tool 

Assertions describe the value of the output flow (identified by the operator name: O2by3) according to the input flows (operands: op1, op2, op3).

Remark 1: in the previous example, the assertion can be defined by the following expression: ***O2by3***=((op1 ***and*** op2) ***or*** (op1 ***or*** op3) (op2 ***or*** op3)).

Remark 2: Operators can be used to define other operators:



When an existing operator is used and recognized by the language parser, this operator is put in the list in sub-tab **Operators** (Figure 122). The sub-tab **Operators** allows selecting the suitable operator when several operators having a same name are available in the library.

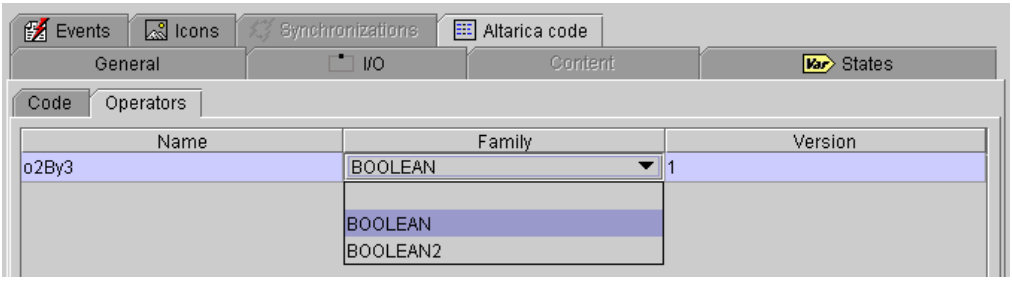


Figure 122 : AtlaRica Code - Operators Tab

NB: When an operator is used and recognized in Altarica code, it is displayed in green color.

Check syntax and consistency of an operator:

Click on button **Syntax** (Figure 121), to check syntax of Altarica code. . If a syntax error is detected, an error window is displayed and describes precisely the kind of error and the line containing this error.

The most usual errors are:

- An assertion is specifying an affectation between variables having a type not compatible (enumerated equal to a Boolean, …)
- The syntax of expressions between brackets is not correct.

Click on button **Consistency** (Figure 108) to check consistency of Altarica code. If a consistency error is detected, an error message is displayed and describes precisely the kind of consistency error and the line containing the error.

The most often consistency errors detected are:

- The assignation of the operator result is not complete for a combination of operand values. It means that the parser has identified a combination of input values for which the output value is not defined. .
- Existence of a combination of operand values (input flow variables) for which the operator result (output flow variable) is assigned with two different values.

Edit an operator model

To edit an operator model proceed as follows:

- Expand the folder in the operator library to display the family containing the model to edit
- Left click on the model to select it,
- Click on ⊞ to display the versions (Figure 123),

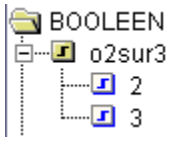


Figure 123 : Versions of an operator model

- An operator can be edited by two ways :
  - Double click on the version number
  - Select the command **Edit model** from the menu **Library** or use command **Edit model** from the contextual menu (right click).
- A bar graph ***Reading model*** (Figure 124) is displayed:

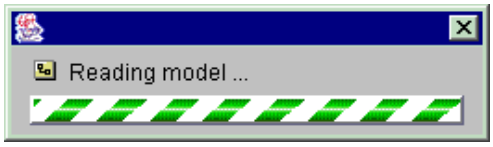


Figure 124 : Bar graph during model reading

- The operator editor is open (Figure 118).

Rename an operator

To rename an operator model:

- In the **Operators** tab, click on the model to rename.
- In the contextual menu, select the command **Rename model**; the field **name** of the operator becomes available.
- Enter the new model name and validate with **Enter**.

Warning: When an operator is renamed, the Altarica code must be modified because the output result is identified by the operator name.
Moreover, modifying name of an operator can have an impact on component models using this operator.

Remark: If the name already exists, an error message is displayed. Click on **OK** and enter a new name, then validate with **Enter**.

Duplicate an operator

To duplicate an operator model:

- Click on the model to duplicate in the **Operators** tab,
- Right click on the model to open the contextual menu,
- Select the command **Model … → Duplicate**
- A bar graph (Figure 125) indicates that the operator is currently duplicated; a copy of the model is then created with the extension *_copy*.

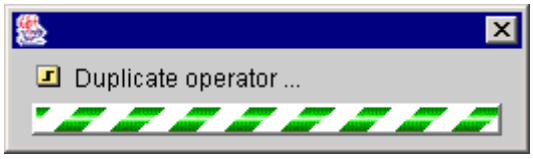


Figure 125 : Bar graph during duplication

- The default name can be changed

Warning : If several model versions exist, only in the last version is duplicated. To duplicate another version of the operator, user must select the right version in the operator tree structure (Figure 123) before launching the command **Model … → Duplicate**.

Remark : To move an operator model in another folder:

- Copy the operator with the command **Model … → Copy**,
- Paste the operator in the destination folder : command **Model … → Paste**,
- Restore the original name of the operator (delete the extension « *_copy* ») in the destination folder.

Create a new version of an operator

To create a new version of an operator model:

- Click on the model in the **Operators** tab,
- Right click on the model to open the contextual menu,
- Select the command Model … → New version,
- A bar graph (Figure 114) indicates that the new version is currently creating.

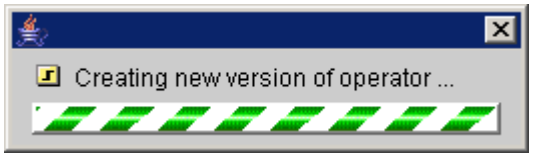


Figure 126 : Creation of a new version

When the new operator version has been created, it is stored in the operator library (Figure 110) below the previous model version.

Freeze a version of an operator

Warning: This command is **high-risk** and must be used with caution, because it is irreversible: modifications are not possible on a frozen model.

To freeze an operator model:

- Click on the model in the **Operators** tab,
- Click on ⊞ to display the available versions,
- Left click on the version number to select it,
- Right click on the model to open the contextual menu,
- Select the command **Model … → Freeze**.

Remark: The frozen versions are represented by the symbol: in the library. This ***Frozen*** attribute can be also visualized in the model properties (Figure 127); select command **Properties** from the contextual menu or from menu **File**.

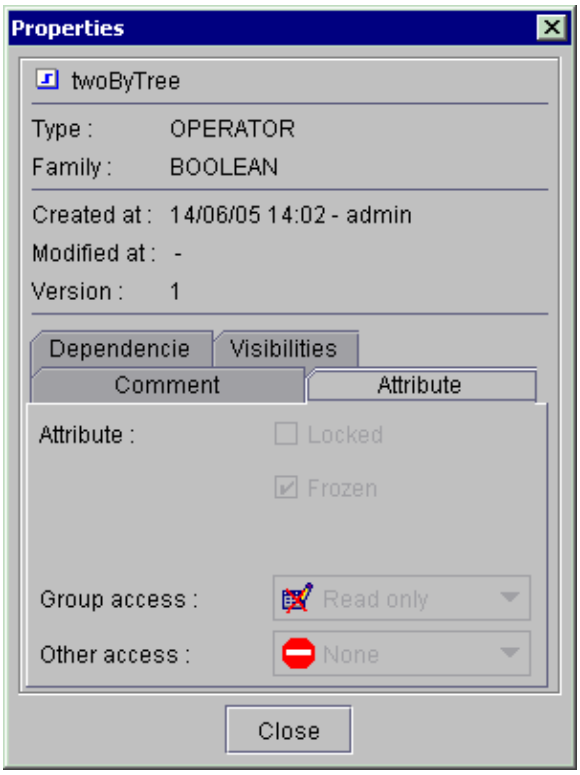


Figure 127 : Operator - Model properties

Delete an operator version

Warning: This command is **high-risk** and must be used with caution, because it is irreversible.

To delete an operator model:

- Click on the model in the **Operators** tab,
- Click on ⊞ to display the available versions,
- Left click on the version number to select it,
- Right click on the model to open the contextual menu,
- Select the command **Delete model**; a *dialog box is open*:

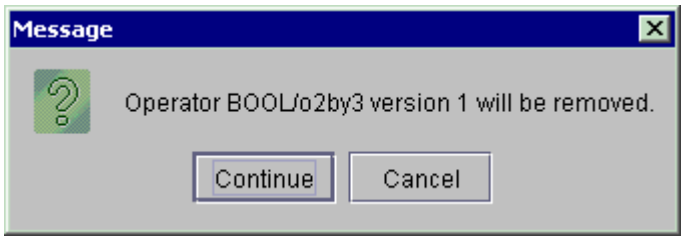


Figure 128 : Dialog box for model removing

- Click on **Continue** to accept deletion, or on **Cancel** to cancel deletion.

EDIT A TYPE

Create a new type

Create a new enumerated type

Definition of types in library allows re-using them several times in model of components, equipments or operators.
An enumerated type (also called enumeration or enum) is a data type consisting of a set of named values called elements or enumerators of the type. The enumerator names are identifiers that behave as constants in AltaRica. A variable that has been declared as having an enumerated type can be assigned any of the enumerators as a value.

To create a new enumerated type:

- In the left part of the window, click on tab **Types,**
- Select the family in which the type will be created.
- Right click on the family to display the contextual menu.
- Select command **Create new type… > Enumerate**.
- Enter a name for the new type in the field **Type name** (Figure 129)
- In the field **Name**, enter the name of the first enumerator (string value).
- Click on command  or type on **Enter**, the enumerator is added in the list.
- Add the possible enumerators in the list.
- Click on buttons **Save** and **Close**.

Remark: The default version of a new type is 1.

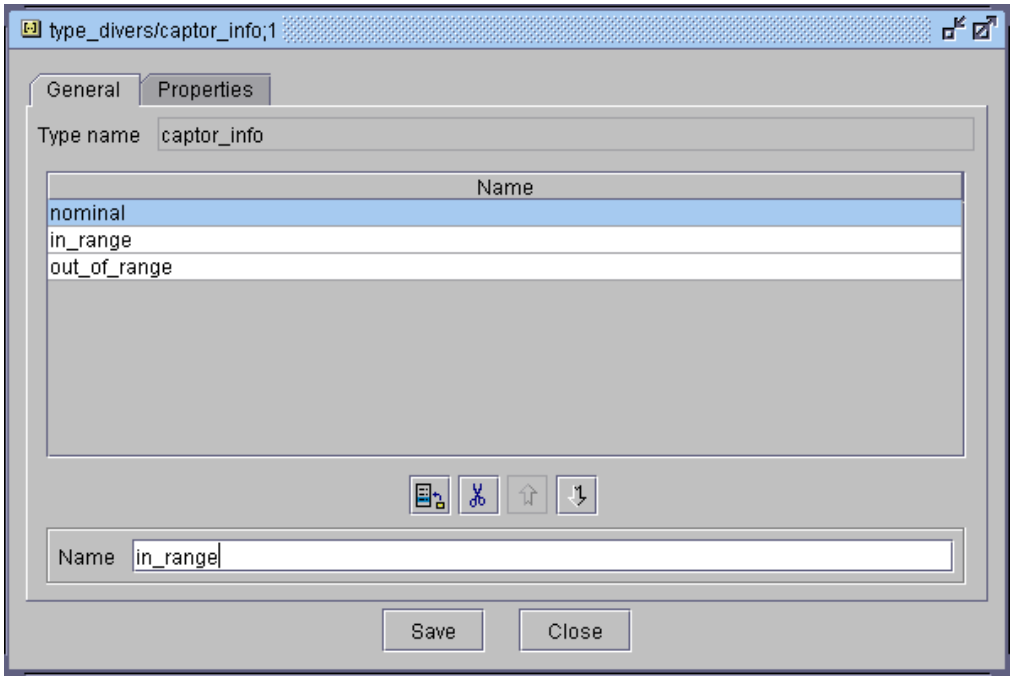


Figure 129 : Create enumerated type

Remark : In the tab **Types**, the enumerated type is represented by the symbol .

Create a new record type

A **record** (also called tuple or struct) consists of two or more variables stored in a data structure. The variables are flows. They are called **Fields**.
Each flow variable of a record can be defined with a different type (enumerated, Boolean, integer,…).
A record is used to include several data (fields) in a same variable identifier. In an Altarica model, the enumerated type is a list of possible values for a variable (state or flow) whereas a record type is a list of flow variables.

A record can be used to define a logical bus between two models, allowing the reduction of the number of graphic link. Only flow variables (`in` or `out`) can be declared with a record type

To create a new record type:

- In the left part of the window, click on tab **Types,**
- Select the family in which the type will be created.
- Right click on the family to display the contextual menu.
- Select command **Create new type… > Record**..
- Enter a name for the new type in the field **Type name**.
- In the field **Name**, enter the name of the first record field.
- In the menu **Type**, select a type.
- Click on command or type on **Enter**, the field and its type are added in the list.

- Click on the button **Assign** to modify the type, if necessary; the type is displayed in the column **Type**.
- Click on the column **Orientation**; select the orientation: **normal** or **inverse**.
- Click on the column **Crossfield** to define a flow crossing (Figure 131)
- Enter the cross field and validate with **Enter**.

Remark: The default version of a new type is 1.

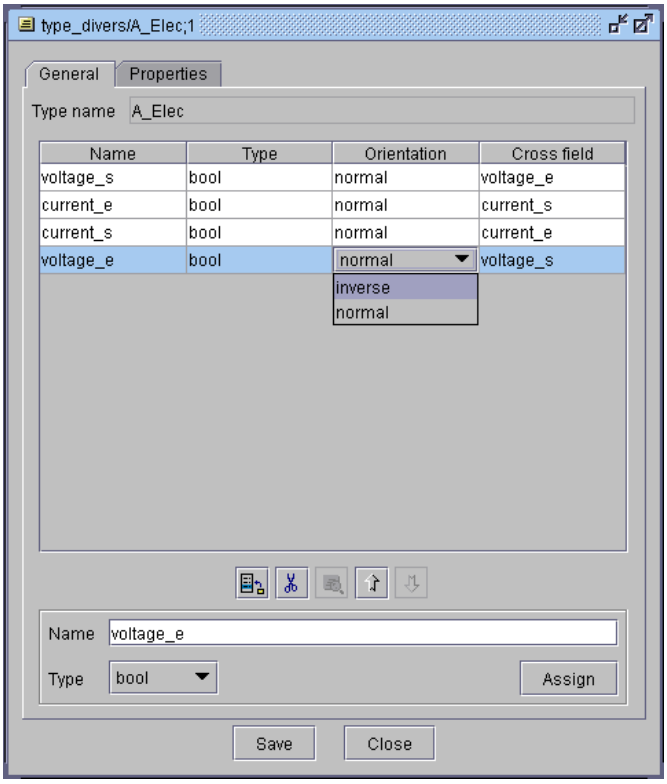


Figure 130 : Create a record type

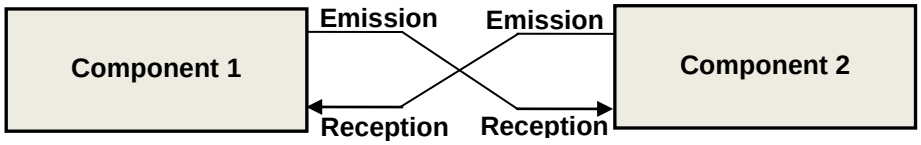


Figure 131 : Cross field for Inputs/Outputs

Transcription in Altarica code :

Creation of a record flow generates two connection studs:

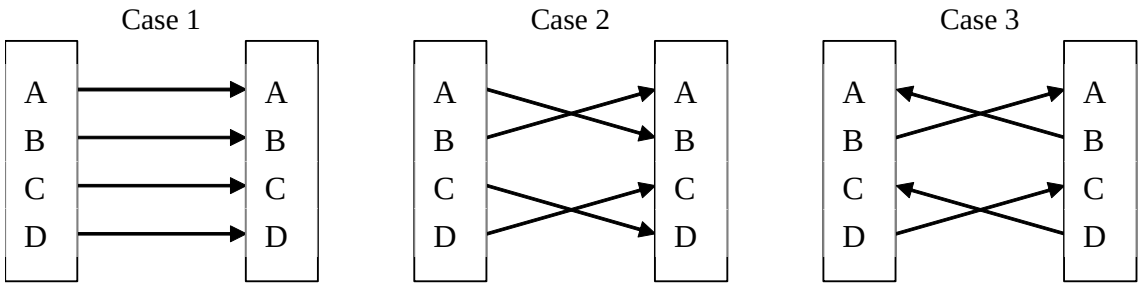
1. a port `in`,
2. a port `out`.

Each connection stud is composed of n fields (flow parameters=field).

In general the field has the same orientation as its reference stud. The clause **inverse** is used to invert orientation of the fields of the stud **in** or stud **out**.

The figure below presents three possible cases of records and connections between the flow fields (A, B, C, D).

Case 1 : A data structure with direct assignments,
Case 2 : A data structure with cross field assignments,
Case 3 : A data structure with inverted and cross field assignments.



In order to simplify the Altarica code interpretation presented below, the equality relations are oriented and thus defined using the assignments.

The character is '**^**' is used to read the field from the record identifier.

```
link
  flow A,B,C,D : int ;
  assert
    in^A := out^A;
    in^B := out^B;
    in^C := out^C;
    in^D := out^D;
knil
```

```
link
  flow A,B,C,D : int ;
  assert
    in^A := out^B;
    in^B := out^A;
    in^C := out^D;
    in^D := out^C;
knil
```

```
link
  flow A,B,C,D : int ;
  inverse out^A; out^C;
          in^B;  in^D;
  assert
    in^A := out^B;
    out^A := in^B;
    in^C := out^D;
    out^C := in^D;
knil
```

Use the following commands to manage the fields from the list:

- : to delete an Input/Output field,
- : to edit a field if it is predefined (or double click on **Type**),
- : to move up the selected line in the list of Inputs/Outputs fields,
- : to move down the selected line in the list of Inputs/Outputs fields,

These commands are also available from the contextual menu with a right click.

Remark 1: In the tab **Types** in the left part of the screen, the record type is represented by the symbol .

Remark 2: In the list of “predefined” types, only the list of enumerated types is displayed (the list of record types is not displayed) because a record type cannot include a field with a record type.

Edit a type

To edit a type model:

- Expand the folder in the type library to display the family containing the model to edit
- Left click on the model to select it,
- Click on ⊞ to display the versions (Figure 132),

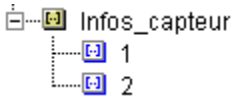


Figure 132 : Versions of a type

A typecan be edited by two ways :

- Double click on the version number
- Select the command **Edit** model from the menu **Library** or use command **Edit model** in the contextual menu (right click).

A bar graph ***Reading model…*** is displayed (Figure 133) :

Figure 133 : Type reading bar graph

The type edition window is displayed (Figure 129).

Rename a type

To rename a type model:

- In the **Types** tab, click on the model to rename.
- From the contextual menu, select the command **Rename model**; the field **name** of the type becomes available.
- Enter the new model name and validate with **Enter**.

Remark: If the name already exists, an error message is displayed. Click on **OK** and enter a new name, then validate with **Enter**.

Warning: This command is **high-risk** and must be used with caution, because the renaming leads to a reference modification of the model in components/equipments/operators/types using this reference.

Duplicate a type

To duplicate a type model:

- Click on the model to duplicate in the **Types** tab,
- Right click on the model to open the contextual menu,
- Select the command **Model … → Duplicate**
- A bar graph indicates that the type is currently duplicated; a copy of the model is then created with the extension *_copy*.

Figure 134 : Duplicate type bar graph

- The default name can be changed.

Warning : If several model versions exist, only in the last version is duplicated. To duplicate another version of the type, user must select the right version in the type tree structure (Figure 132) before launching the command **Model … → Duplicate**.

Remark : To move a type model in another folder:

- Copy the type with the command **Model … → Copy**,
- Paste the type in the destination folder : command **Model … → Paste**,
- Restore the original name of the type (delete the extension « *_copy* ») in the destination folder.

Create a new version of a type

To create a new version of an equipment model:

- Click on the model in the **Types** tab,
- Right click on the model to open the contextual menu,
- Select the command **Model … → New version**,
- A bar graph (Figure 135) indicates that the new version is currently creating.

Figure 135 : New version creation bar graph

When the new type version has been created, it is stored in the equipment library (Figure 132) below the previous model version.

Copy a type

To copy a type:

- Click on the model to copy in the **Types** tab,
- Right click on the model to open the contextual menu,
- Select the command **Model … → Copy**
- A bar graph indicates that the type is currently copying;
- Select the new target family for the copy.
- Select the command **Paste**.

Remark : Only for the last version of a type can be copied.

Freeze a type version

Warning: This command is **high-risk** and must be used with caution, because it is irreversible: modifications are not possible on a frozen model.

To freeze a type model:

- Click on the model in the **Types** tab,
- Click on  to display the available versions,
- Left click on the version number to select it,
- Right click on the model to open the contextual menu,
- Select the command **Model … → Freeze**..

Remark: The frozen versions are represented by the symbol:  in the library. This ***Frozen*** attribute can be also visualized in the property editor: select command **Properties** from the contextual menu or from menu **File**.

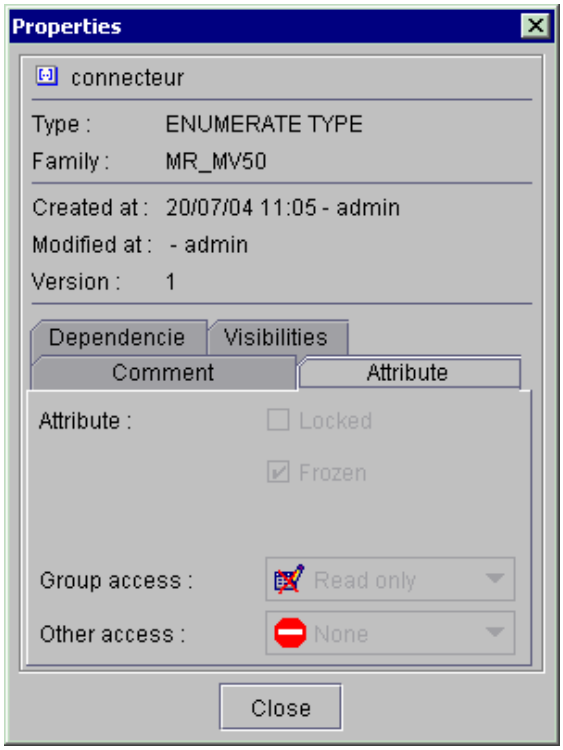


Figure 136 : Type – Model properties

Delete a type

Warning: This command is **high-risk** and must be used with caution, because it is irreversible.

To delete a type model:

- Click on the model in the **Types** tab,
- Click on ⊞ to display the available versions,
- Left click on the version number to select it,
- Right click on the model to open the contextual menu,
- Select the command **Delete model**; a *dialog box is open* (Figure 137):

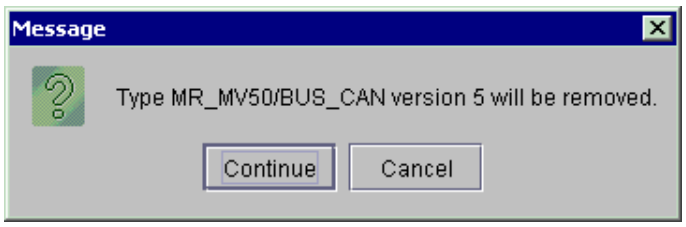


Figure 137 : Delete type message

- Click on **Continue** to accept deletion, or on **Cancel** to cancel deletion.

FIND A MODEL

To find a model:

- Select **Search** from the menu **Edition** or the shortcut **CTRL + B** or the command from the main tool bar; the window **Search** is displayed (Figure 138).
- Click on tab **Model** and enter the model name in field **Filter** (use the character "*" enter an incomplete model name).

Remark: the character « * » can be used to decrease the search space (example: **H*/val*)** or to complete a partially-known-name (examples: **HYD*/val***, ***/valve**) .

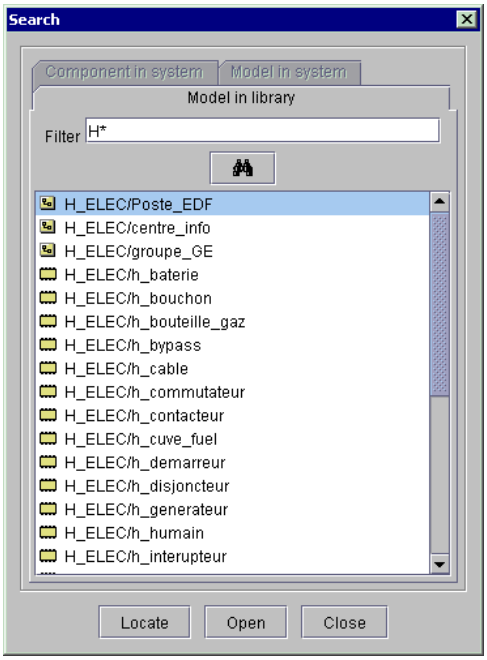


Figure 138 : Models search

- Click on the command ; the model found (or the models having a name corresponding to the defined filter) is displayed in the lower part of the window.
- Click on button **Locate**
- SD9 selects the tab of the library, opens the family and displays the model.
- If necessary, click on button **Open** to edit the model or double click on the selected model in the **search** window.

EDIT A SYSTEM

Create a new system

To create a new system:

- Click on the tab **Systems**.
- Select the project folder in which the new system will be created.
- Create the system in the selected folder:
  - Select the command **Create…> System** from the contextual menu or
  - Select the command **New… - System** from the menu **File** or
  - Click on the command  from the main toolbar
- the dialog box (Figure 139) is displayed.

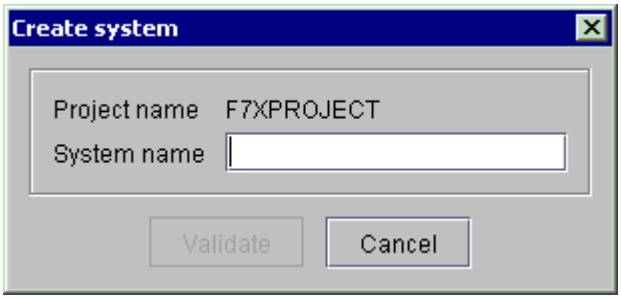


Figure 139 : Create system

- Enter the name of the system.
- Click on **Validate** to accept the creation, otherwise click on **Cancel**.
- When the new system is created, it is automatically inserted in the library by alphabetical order (Figure 140), and it is automatically edited.

Figure 140 : New system created in the library

Remark : New system is created with version 1

Open a system

To open an existing system:

- Click on tab **Systems** to display the list of project folders and systems,
- Expand the folders and sub-folders to display the system to open,
- Double click on the system name to display the available versions,

- Select the version and open the system:
  - Select the command **Open system** from the menu File or
  - Use the shortcut **CTRL + O** from the keyboard or
  - Select the command  from the main toolbar or
  - Select the command **Open system** from the contextual menu.
- If the system is already opened, the error message *The system is already open* is displayed (Figure 107); click on the **OK** button.

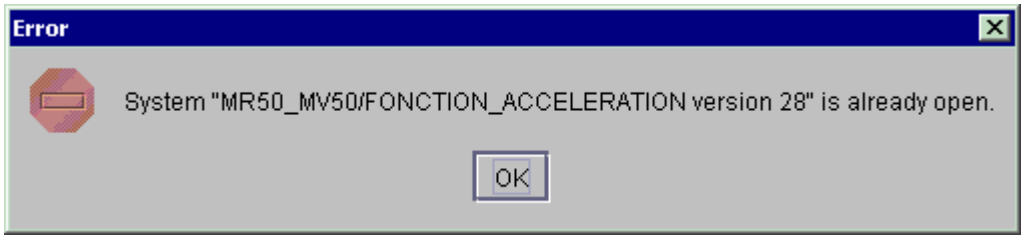


Figure 141 : System already open

- If the system is locked, it can be open in the read only mode. A warning message is displayed (Figure 142). A system can be locked:
  - If another user is working on the current system (in edition mode by another user)
  - The application has closed abnormally, for example when the OS crashed.

Figure 142: Message when a system is locked

When the system is open, it is displayed in the left area of the window (Figure 143).

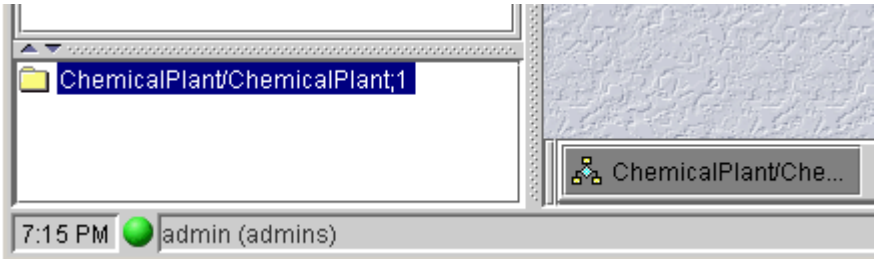


Figure 143 : Systems edition management

- Right click on the system to display the contextual menu (Figure 144):

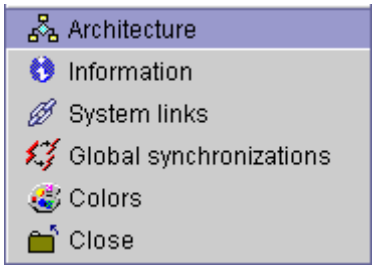


Figure 144 : contextual menu of a system

- This menu is also available from the menu **Views**.

N.B: To launch the simulation, the system must be open in edition.

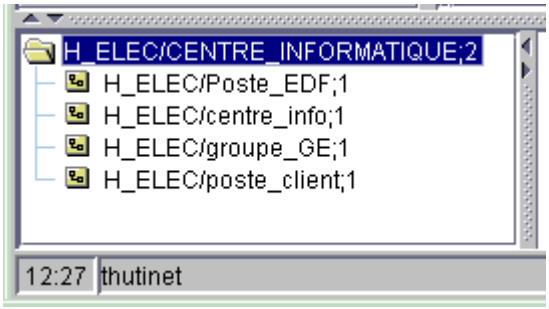


Figure 145 : System equipments

Unlock a system

N.B: A system is automatically locked when it is opened to avoid that two users modify it simultaneously. It is unlocked when the user closes a system or quits DAS. If the computer is stopped in an untimely mode, it may remain locked. The system **must** be unlocked before renaming or duplicating it.

The icon 

indicates that the system is locked. When a system is renamed or duplicated, an *Error* message (Figure 146) is displayed if the system is locked.

Figure 146 : Locked system message

If the session of the system edition is stopped abnormally (after a system crash for example), the system can remain locked. The system must be unlock before to edit it, rename it or duplicate it again.

To unlock the system, open the window **Properties** from the contextual menu or from the menu **File** (Figure 147). Uncheck the box "**Locked**" to unlock the system.

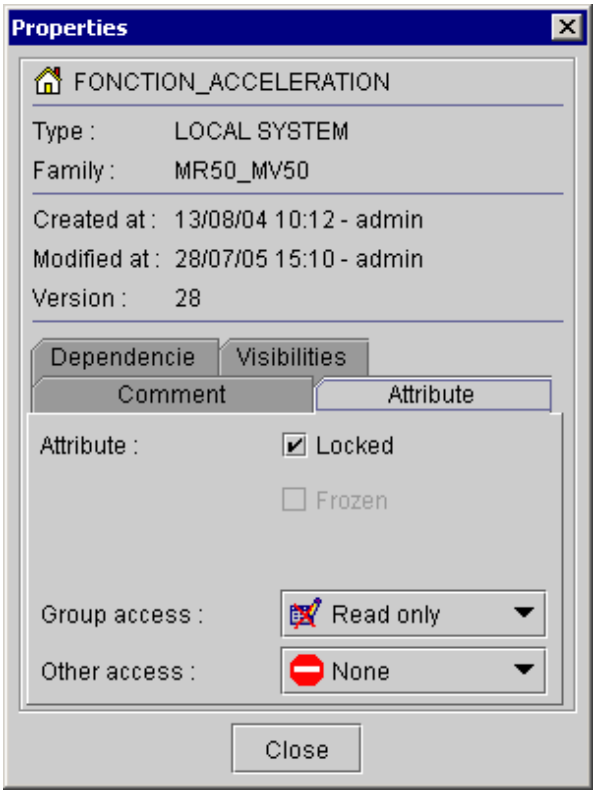


Figure 147 : Properties edition – Locked attribute

Close a system

To close a system:

- Select the command **Close** from the menu File or
- Use the shortcut **CTRL + F** from the keyboard or
- click on the command from the toolbar or
- Select the command **Close** from the contextual menu.

If the system has not been modified, a *Closing system* bar graph (Figure 148) is displayed:

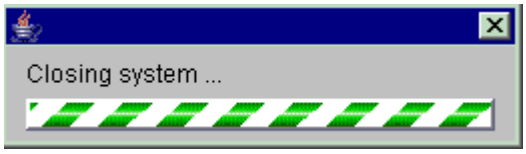


Figure 148 : Closing system bar graph

If the system has been modified, a dialog box (Figure 149) indicates that it has been modified.

Figure 149 : Save system message

Click on the button **Yes** to save the modifications or **No** to lose the modifications. Otherwise click on the button **Cancel**.

Rename a system

To rename a system:

- Right click on the system from the library to display the contextual menu,
- Select the command **Rename system**.

A system must be unlocked to be renamed.

Duplicate a system

To duplicate a system:

- Right click on the system from the library to display the contextual menu,
- Select the command **Duplicate system**.
- A *Duplicate system* bar graph (Figure 150) indicates that the system is being duplicated; the name is copied with the extension *_copy* in the same project folder.

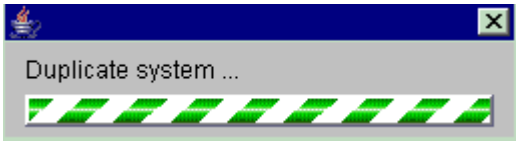


Figure 150 : Duplicate system bar graph

- The default name can be changed.

Warning : If several versions exist, only in the last version is duplicated. To duplicate another version of the system, user must select the right version in the system tree (Figure 132) before launching the command **Duplicate system.**

Create a new version of a system

To create a new version of a system:

- Select the system in the library
- Right click to display the contextual menu
- Select the command **New version of system**.
- A bar graph is displayed:

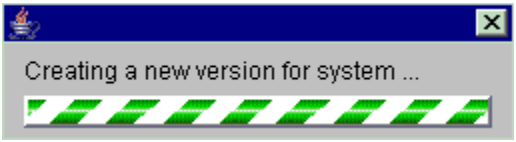


Figure 151 : Creation of a new version of a system

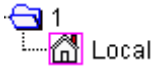
When the new version has been created, it is stored in the library (Figure 132) below the previous system version..

Freeze a system

Warning: This command is **high-risk** and must be used with caution, because it is irreversible.

To Freeze a system, proceed as follows:

- Click on the system in the **Systems** tab,
- Click on ⊞ to display the available versions,
- Left click on the version number to select it,
- Right click on the system to open the contextual menu,
- Select the command **System Freeze**.

Remark: The frozen versions are represented by the symbol 

in the library. This "Frozen" attribute can be also visualized in the system **Properties**. select command **Properties** from the contextual menu or from menu **File.**

Remove a system

To remove a system, proceed as follows:

- Click on the system in the **Systems** tab,
- Click on ⊞ to display the available versions,
- Left click on the version number to select it,
- Right click on the system to open the contextual menu.

- Select the command **Remove… → System**, a *Message* window is displayed indicating that the selected version system will be removed (Figure 152).

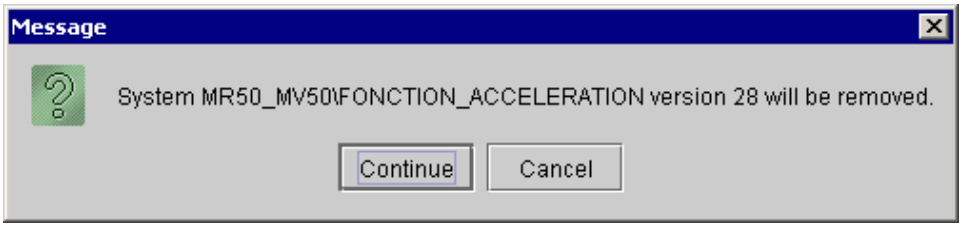


Figure 152 : System removal message

- Click on **Continue** to accept deletion, or on **Cancel** to cancel deletion.

Remark: A system is definitely removed from the library when all its versions are removed.

MODELING A SYSTEM ARCHITECTURE

To create a system architecture by re-using components and equipments from library, proceed as follows:

- In tab **Systems**, create a project folder and create a system inside this project folder.
- If the system is not visible, use the command **Architecture** from the menu **Views** or from the contextual menu or click on icon ; an empty system is then open in the workarea.

Insert a component in the architecture

To insert a component in the architecture, proceed as follows:

- Click on the tab **Components** and select the version number of the component to insert in the current architecture (Figure 153).

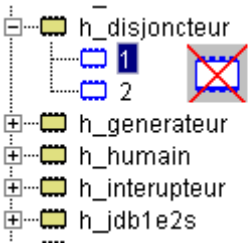


Figure 153 : Component selection

- To insert the component in the current architecture :
  - Select the component version and drag and drop it in the architecture. The symbol means that a component is selected and moved. When the symbol changes into , it indicates that the mouse button can be released and the component can be inserted in the architecture. A default name is assigned to the component instance.
  - From the contextual menu, select the command **Add** (the model instance is inserted at the top left of the architecture).

Remark: The procedure is identical to insert equipments in the current architecture.

When they are inserted, components and equipments are created with a default name:

- ***Component**_xx*: for a component name, where xx is an increasing number,
- ***Equipment**_yy:* for an equipment name, where yy is an increasing number.

To modify the name of a model instance in the architecture:

- Left click on the component or equipment name (Figure 119). A new name can be edited.
- Right click on the component or equipment model to display the contextual menu (Figure 154). Select command **Rename**:

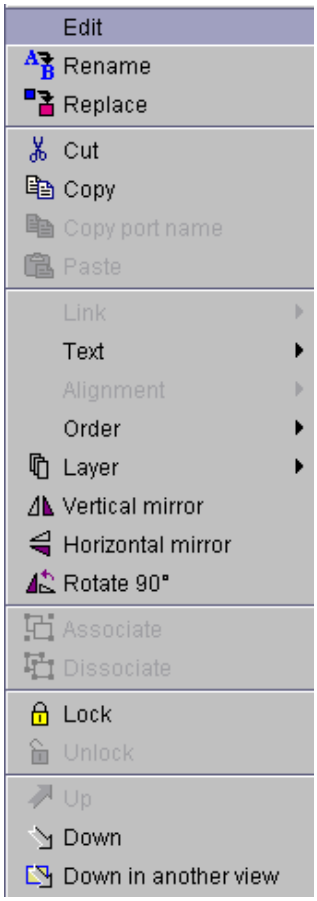


Figure 154 : Architecture – Contextual menu

Remark: To rename a model, the name of the instances must be visible. Ensure that the option **Node labels** is set to **Node labels** or **All labels** (Figure 155):

* *None* : all labels are hidden,
* *Node labels*: name of components, equipments and equipment ports are displayed,
* *Link labels*: names of the connections (type of the flows) are disaplyed
* *All labels*: all names are displayed.

Figure 155 : Architecture – Display menu

- Use the command **Replace** ( ) from the contextual menu to replace the current model by another one from the library

Remark: If the model to be replaced and the new model have the same interface (port names and flow types), the existing graphical links are kept, otherwise they are removed.

- Use the command **Cut**, **Copy** and **Paste** on equipment and component instances to cut, copy and paste them in the current architecture in edition.
- The command **Lock** from the contextual menu is used to prevent the user from accidental moving. Graphical moving of instances is forbidden on locked models.
- In the current architecture, the label name is displayed in bold style. The following commands are available for equipment models:
  - : move up in an architecture to display a higher level of the architecture,
  - : move down in an architecture to display a lower level of the architecture,
  - : the equipment content is open in a new window.

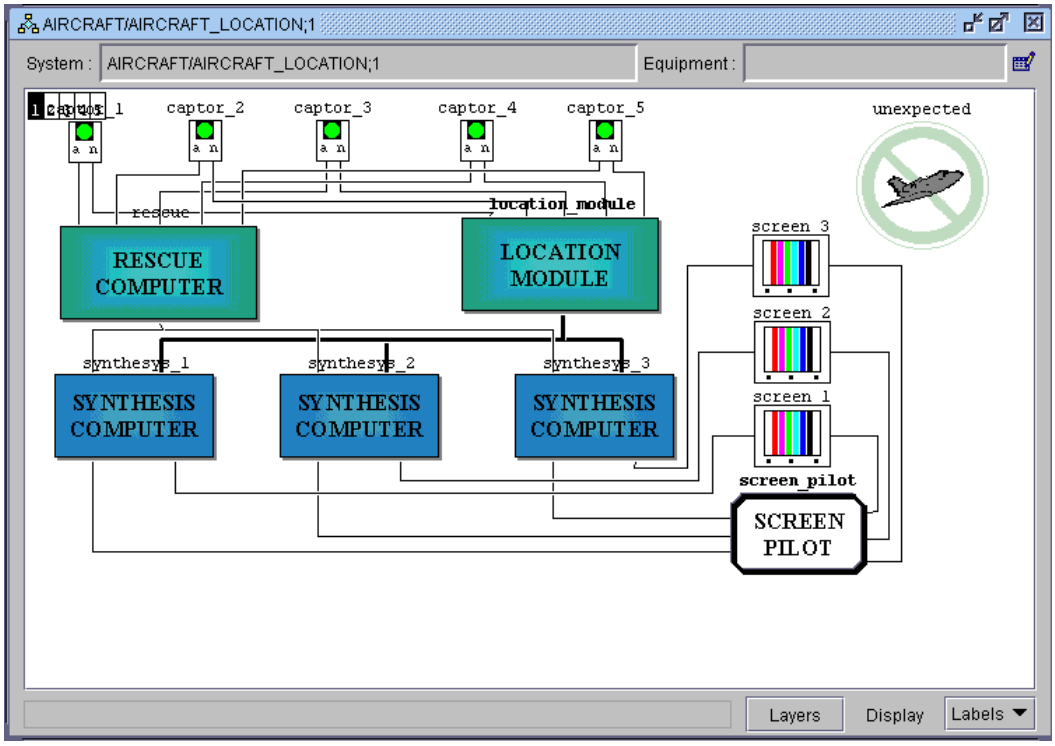


Figure 156 : Architecture in edition mode

- Use the following commands from the contextual menu or from the shortcuts in the design toolbar to optimize the location of components or equipments in current view:
  - Vertical mirror: ,
  - Horizontal mirror: ,
  - 90° rotation: .
- To move a model instance in the current view (if the model is not graphically locked):
  - Drag and drop the model in the architecture or,
  - Select the model instance and use the keyboard arrows ←↑↓→.

Link two components

To create a connection between two component ports: :

- Use the mouse to draw a graphical link between an input port and an output port. When the link is created, user can insert breakpoint with the mouse to customize the link shape (Figure 157).
- When a link is created, a verification is performed on the compatibility of the flows (direction: input/ouput, type: inverse/normal/cross field).

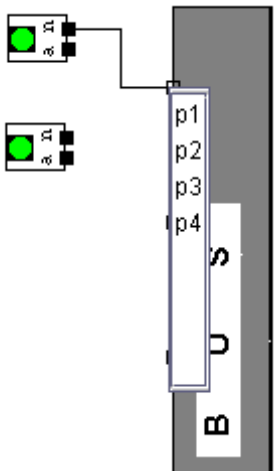


Figure 157 : Creating a link between a single type and a record type flow

- Click on a link and use the command **Link** from the contextual menu, or the commands from the toolbar **Link**, to optimize the shape of the links:
  - Direct :
  - Right angle :
  - Right angle from right to bottom :
  - Right angle from left to bottom :
  - Right angle from right to top :
  - Right angle from left to top :
  - To display an arrow on links (flow direction):

Remark: When a link has two right angles, move the straight line between these two right angles as follows: hold **SHIFT** key from the keyboard and drag and drop the line to move it the architecture. No breakpoint is created and the right angles are kept.

- To align automatically a set of models, select them with the mouse, and use the following commands from the contextual menu:
  - Left :,
  - Right :,
  - Top :,
  - Down :,
  - Vertical : ,
  - Horizontal :.

Example of system graphically optimized (Figure 158):

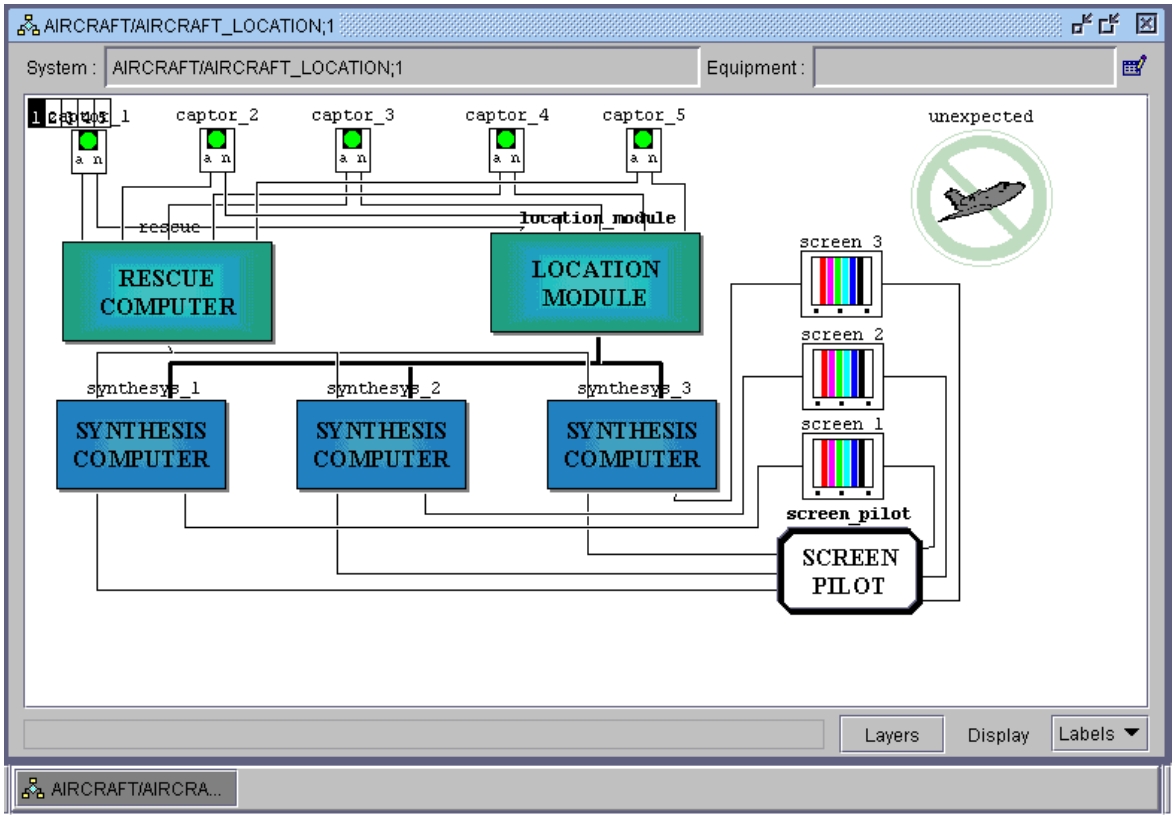


Figure 158 : Architecture example with links

- To display information about ports:
  - Move the mouse pointer over the graphical port
  - Information is displayed in the status bar at the bottom of the edition window
    - Model name: synthesys_3,
    - Port name,
    - Flow type.

- To display information about graphical links:
  - Move the mouse pointer over the graphical link
  - Information is displayed in the status bar at the bottom of the edition window
    - *<Model_name_1>* **.** *<flow_1>* **<->** *<Model_name_2>* **.** *<flow_2>*

- To display information about models:
  - Move the mouse pointer over the model
  - Information is displayed in the status bar at the bottom of the edition window
    - Name of the model in library and name of the instance

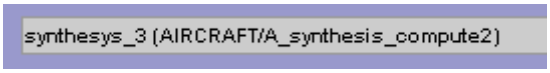


Figure 159: Example of information displayed in the status bar

Remark: When modifications are performed on a model in the library, these modifications are also carried on the instances in system architecture. A consistency checking is performed on the architectures. If a consistency error is detected on an existing link (incompatible input flow and output flow), a dialog box propose to remove the incoherent links when the model is saved (Figure 160):

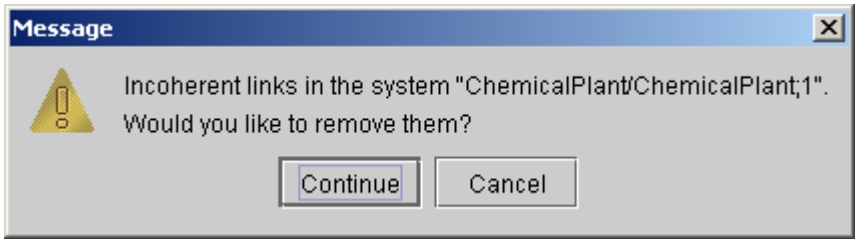


Figure 160 : Link deletion message

- To remove the incoherent links in the current architecture, click on button **Continue**.
- To keep the incoherent links in the architecture, click on button **Cancel**. The incoherent links are displayed in **red** color:

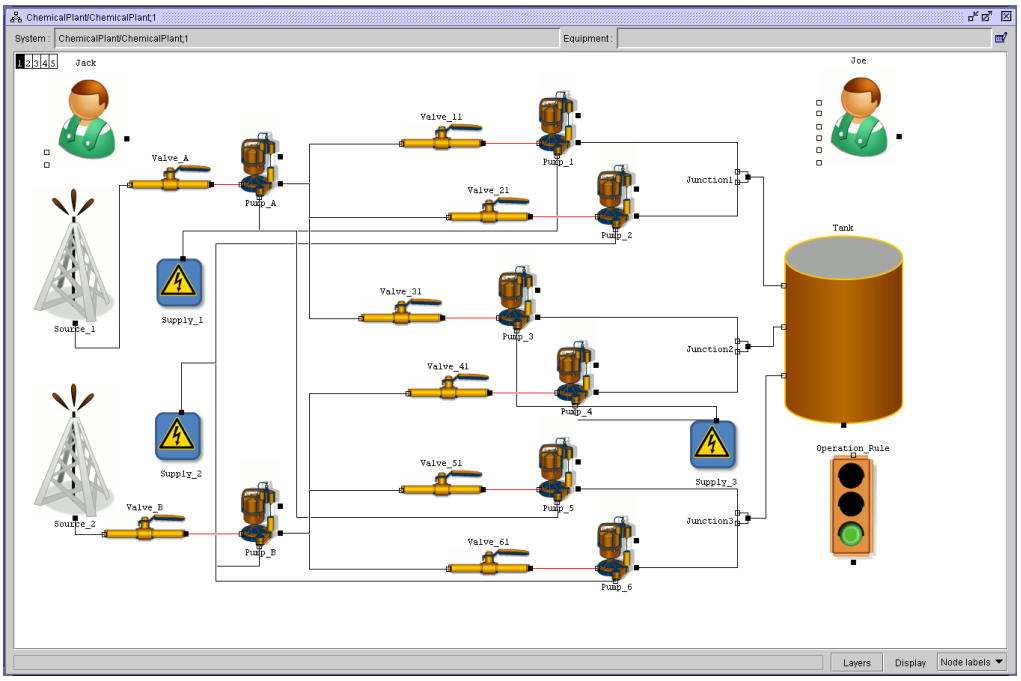


Figure 161 : Incoherent links in the current architecture

Automatic Connections

The user can automatically connect ports of components or equipments **based on the name of the port**.

This functionality is available through the contextual menu by right clicking on the current work area with no-object selected (see Figure 162):

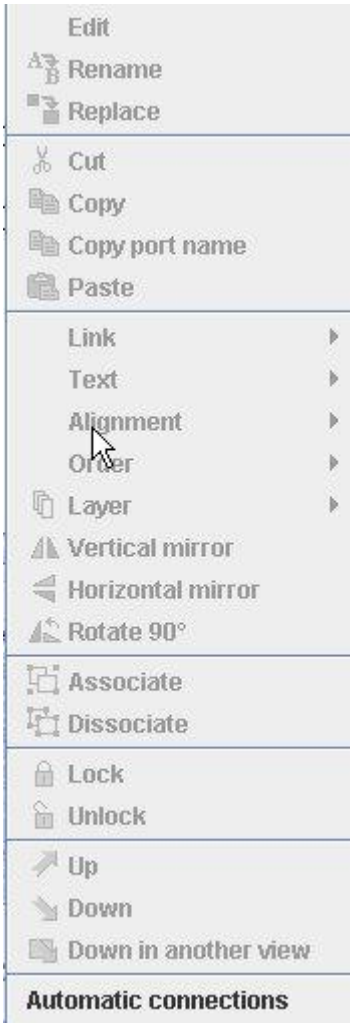


Figure 162: Contextual menu for automatic connections

This functionality asks first for the kind of link to use, then reports the non-ambiguous connections and after validation by user do the non ambiguous connections. It reports also the following lists (with the possibility to locate the communication port involved):

1. List of connections with problem about type
2. List of outputs connected to the same input (fan-in problem)
3. Inputs not connected without any related output found
4. List of non-connected outputs which can connect to input already connected

To end, the user can interactively resolve non-connected inputs for the three first items listed above (we list outputs with the same port name for the second item and outputs with the same port type for the third item).

Thus, the functionality ‘automatic connections’ processes in four steps:

Step 1:

A first panel describes the main rules about the port connections. Use can also select the link type which will be used for automatic connections:

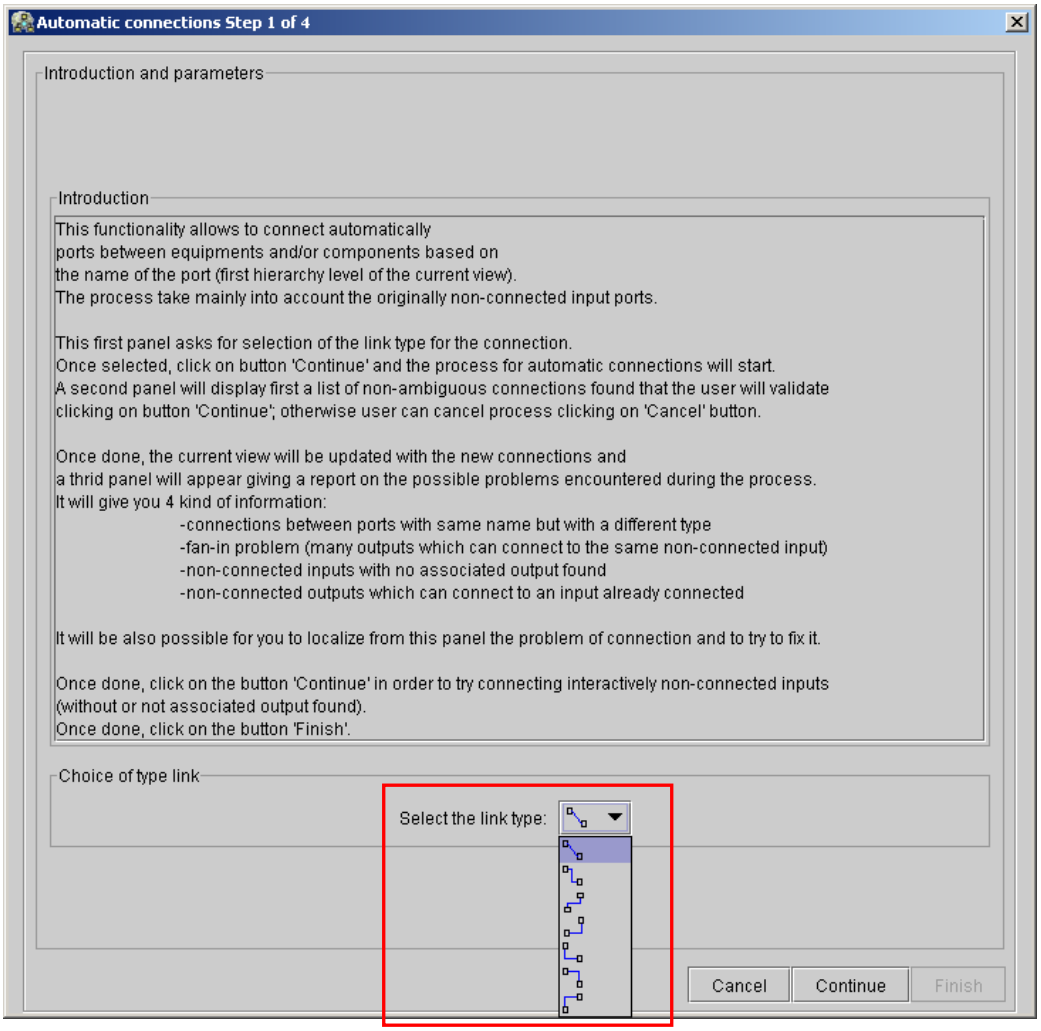


Figure 163: Automatic connections: step 1 of 4

Step 2:

The second panel displays the list of non-ambiguous links (same name and same type):

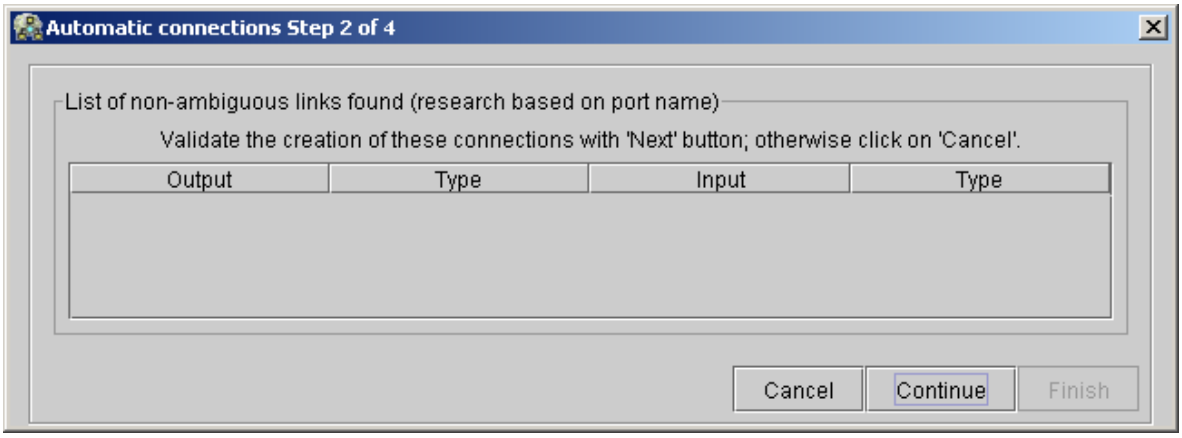


Figure 164: Automatic connections: step 2 of 4

Step 3:

The third panel displays a report about potential problems on automatic connections:

- List of connections having a same port name, but a different type,
- Multiple outputs which can be connected (same name) to a same non-connected input,
- List of Non-connected inputs because no output was found with a same port name,
- List of Non-connected outputs which can connect to an input already connected.

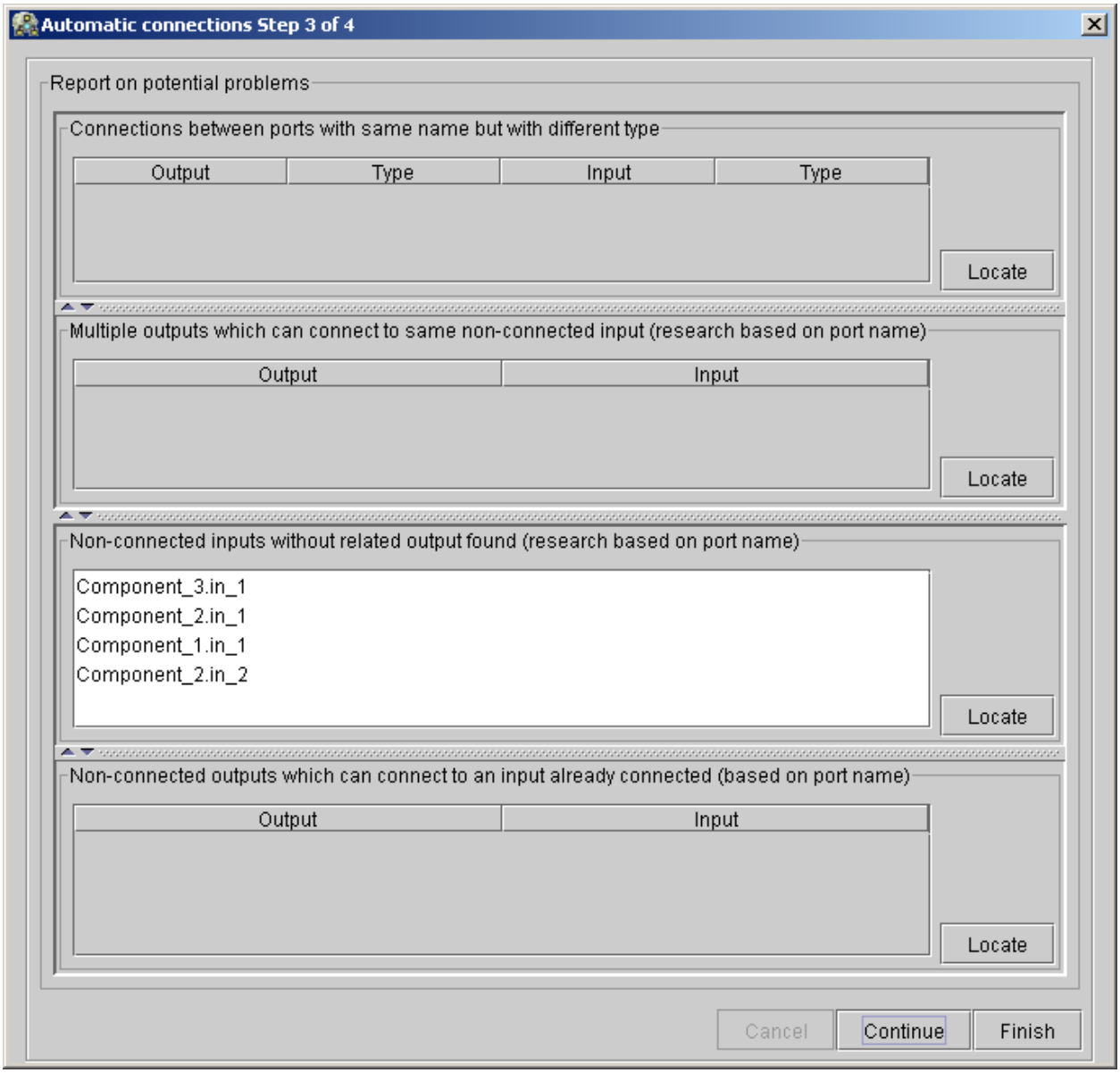


Figure 165: Automatic connections: step 3 of 4

You can select a port name in a list and click on button **Locate** to identify graphically the model in the current view.

Step 4:
The last step consists in proposing user to connect interactively the remaining ports identified in step 3.

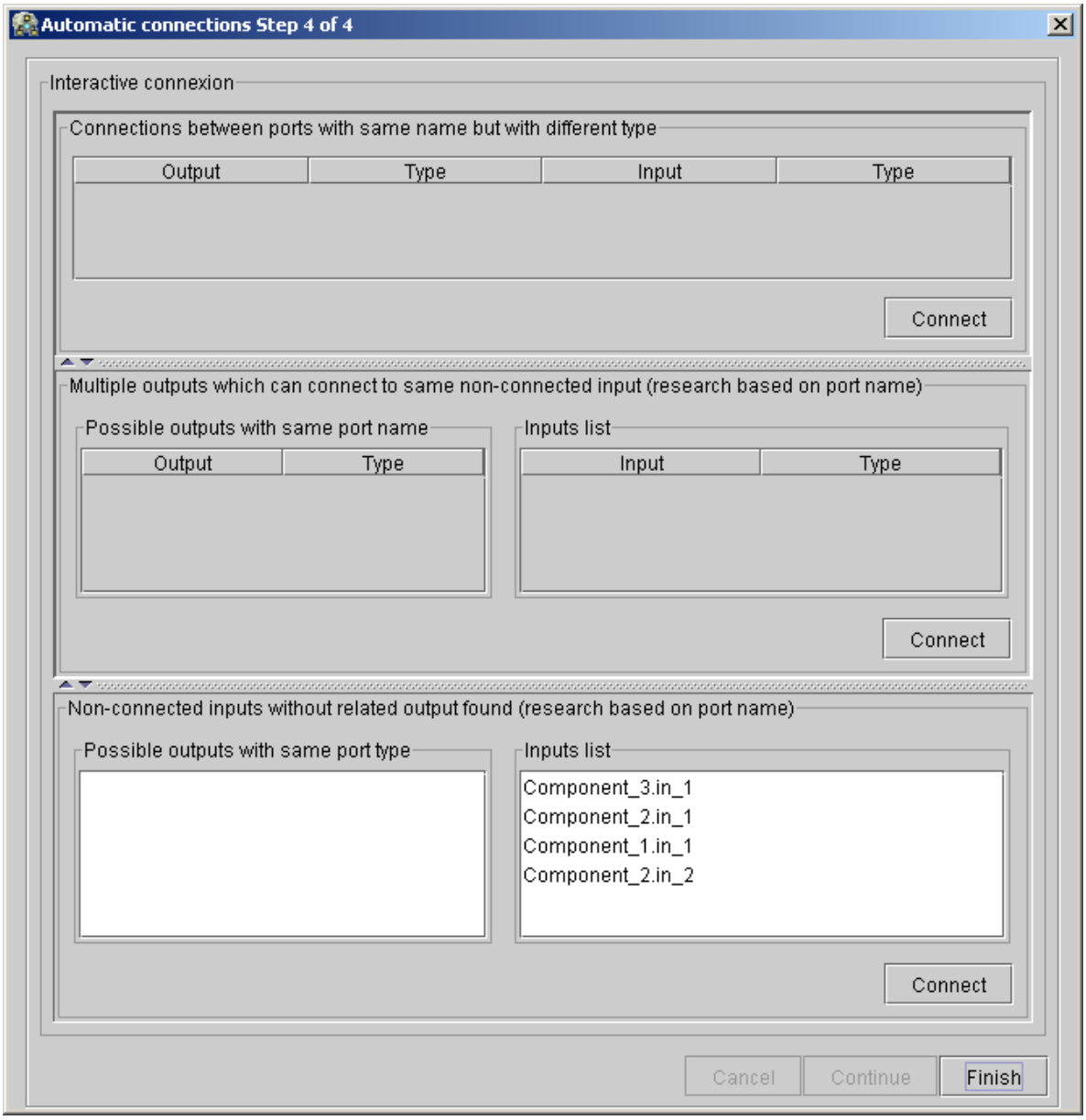


Figure 166: Automatic connections: step 4 of 4

Select ports to connect, and click on button **Connect** to create the link.

Layers

An architecture view can be composed with five display layers. Graphical objects representing components, equipments, links can be shared out in the layers.

To put graphical elements in the different layers:

- Select the graphical element (component, equipment, link) (Figure 167),
- In contextual menu (right click) select the layer on which the element must be belong.

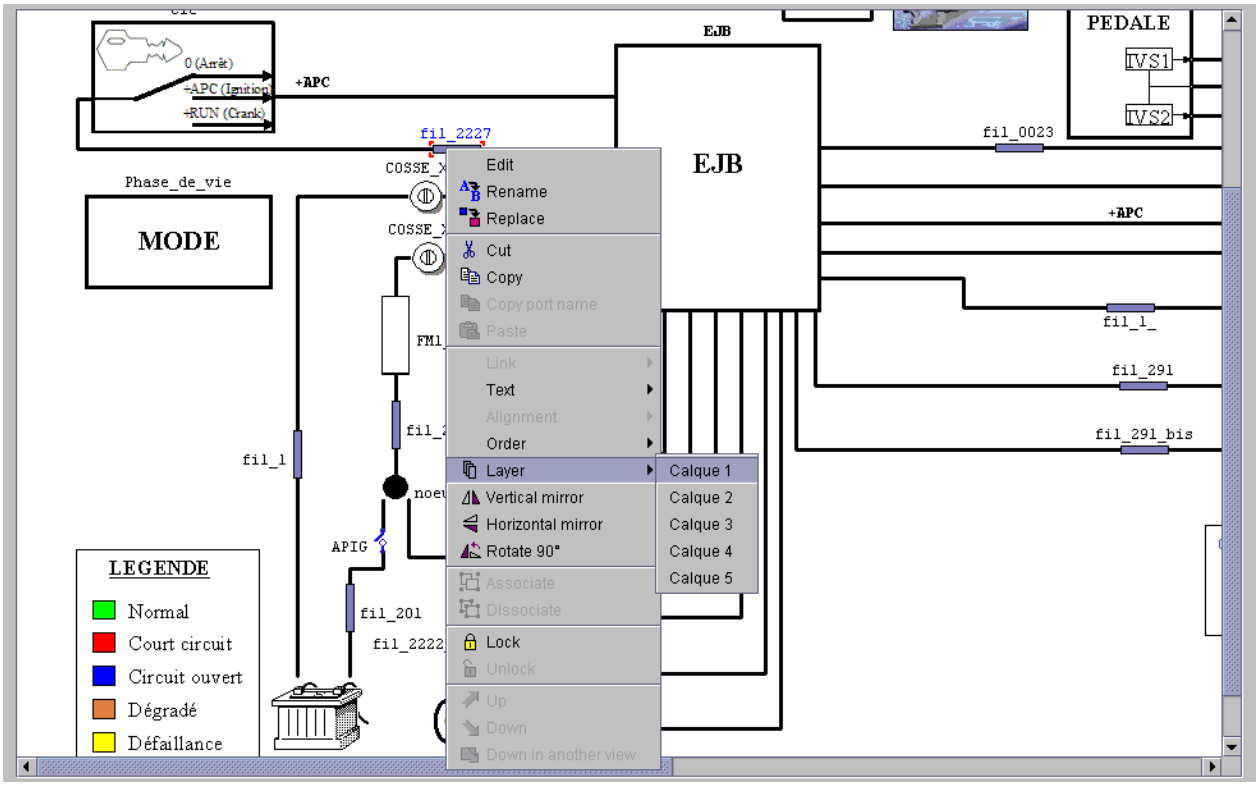


Figure 167 : To put an element on a layer

To check the element displayed, proceed as follows:

- Click on button **Layers** (Figure 168) to open the layer manager (Figure 169).

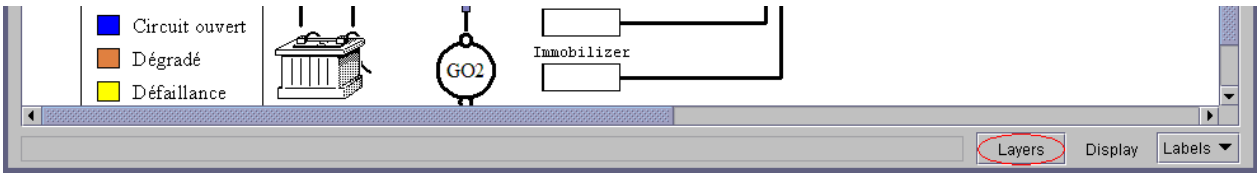


Figure 168 : Layer edition

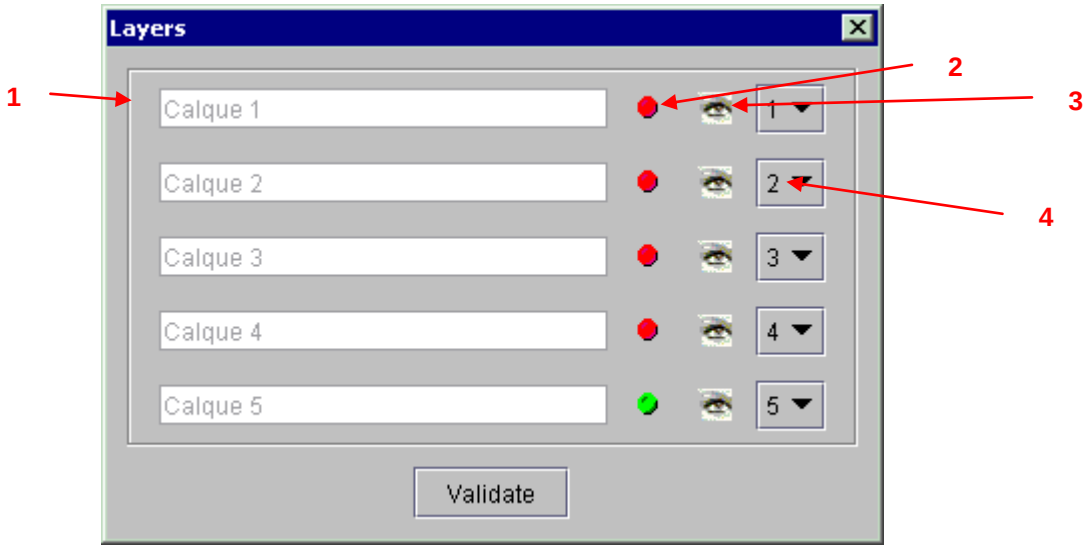


Figure 169 : Layer manager

The layer manager includes: :

- Name of the layer (1): double-click on the field to change the layer name; the default name is “Calque_n”.
- Current active layer (2): The green light indicates the current active layer. Models are inserted in active layer of the architecture. To change the current active layer, click on a red light.
- Visible layers (3): user can make a layer visible or hidden. Click on the icon to switch from the visible mode to the hidden mode.
- Layers orders (4): the user can define the priority of each layer. The priority correspond to the graphical order (1 : foreground layer, to 5 : background layer).

In the following example, the layer 2 “Calque 2” is hidden and new instance models are inserted in layer 1.

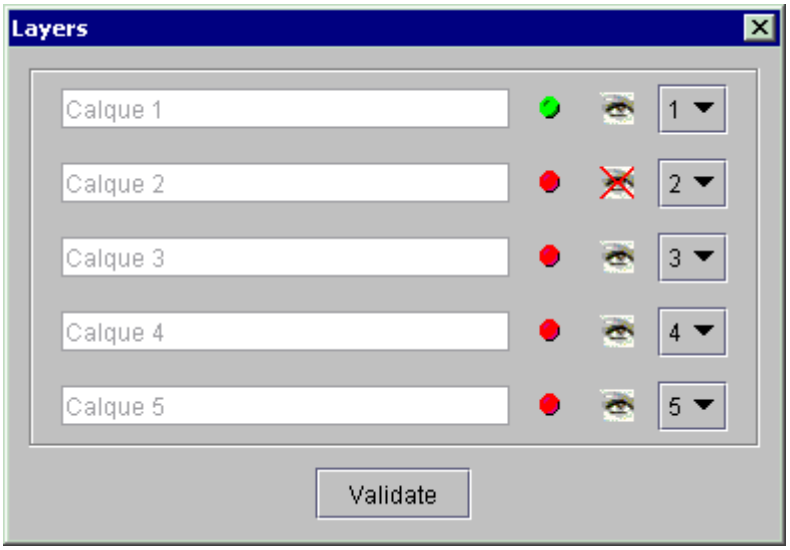


Figure 170: Example of layer configuration

SEARCH A COMPONENT MODEL IN A SYSTEM

To search a component in the current architecture:

- Select the command **Search** from the menu **Edition** or use the shortcut **CTRL + B** from the keyboard or use the command from the *Standard* toolbar; the ***Search*** window (Figure 171) is open.
- Click on tab **Component** and enter the component name in the field *Filter*, as follows : *<Family_Name>/<Model_Name>*

  NB: the character « * » can be used to reduce the search space (example: **H*/val*)** or to define a partially-known-name (examples: **HYD*/val***, ***/valve**).
- Click on the button , the components which fulfil the filter request are displayed in the middle area of the window

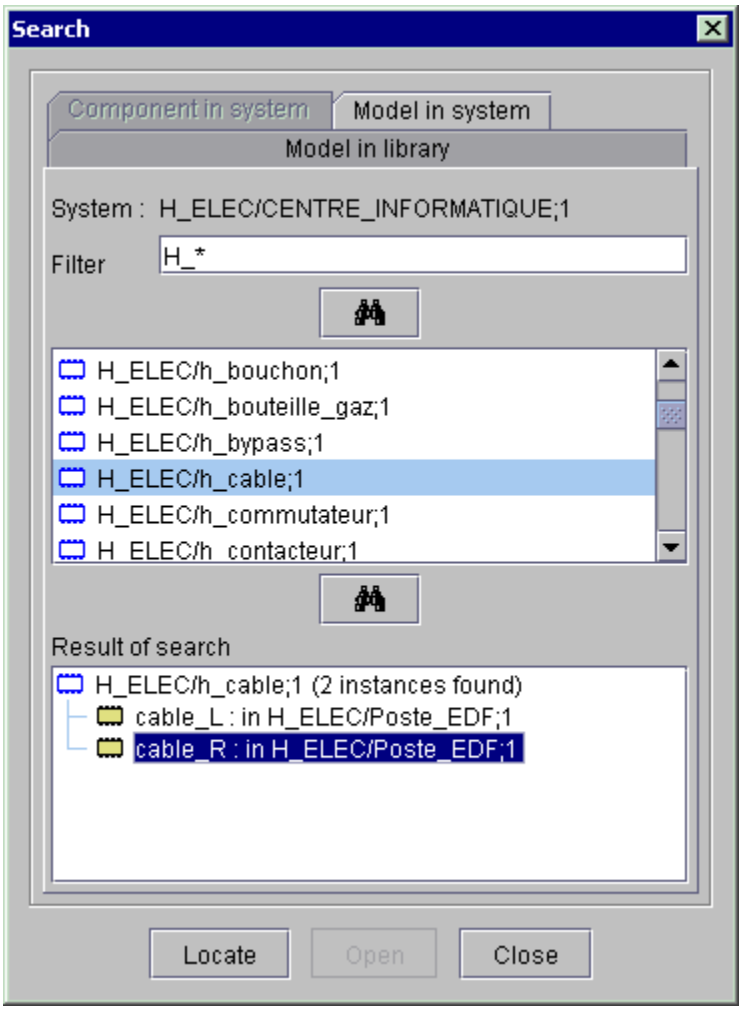


Figure 171 : Search window

- Select the model, click on the second button , identified instances are displayed in the lower part of the window.

- Double click on the model to expand the list of model instances existing in the system. It provides information about their path (in H_ELEC/Poste_EDF; 1).
- Select an instance, click on the button **Locate** to highlight the instance in the current architecture.

INFORMATION

Remark: The tool **Information** is used when a consistency problem message was detected. Information window displays:

- The unknown models (when models are renamed or removed from the library)
- The removed links

Tool **Information** is available from the menu **Views**, or from the contextual menu. An architecture must be in edition. Information window contains six tabs (Figure 173):

- Properties
- Unknown models,
- Suppressed links,
- Invalid links,
- Renamed components,
- Changed models.

Access commands

Click on the **Properties** tab (Figure 172) the window displays the following information:

- System Creation date
- Last modification date
- Version number,
- A comment on the system, it can be modified by any user with write privileges.

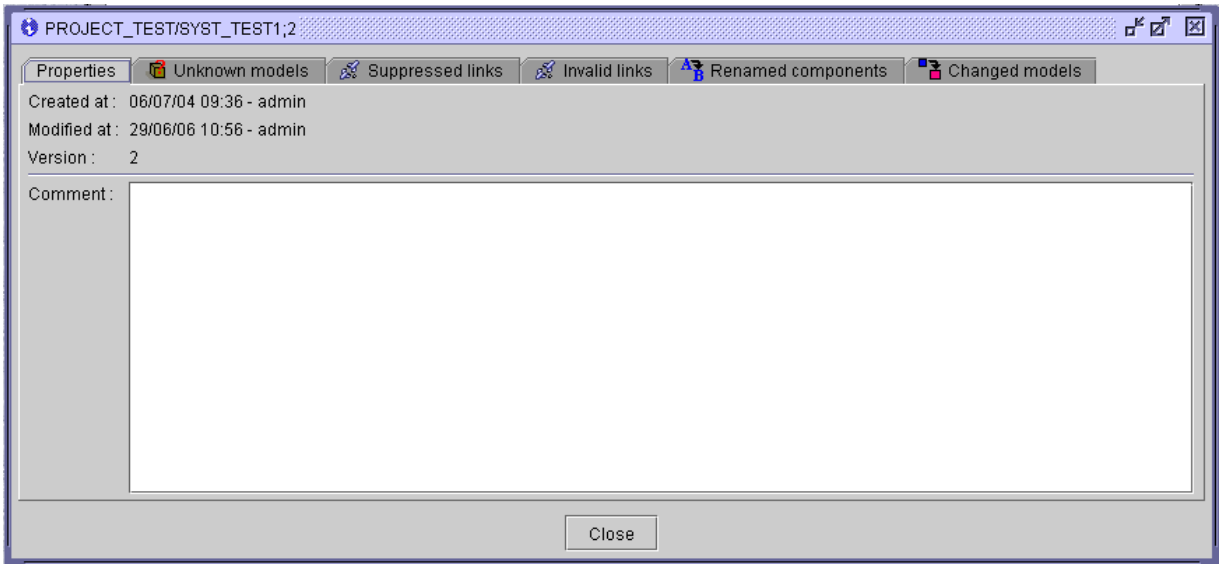


Figure 172 : Information - Properties

Remark: This information can be accessed by **Properties** command in **File** Menu.

Unknown models tab

Click on the **Unknown models** tab (Figure 173) the window displays the list of unknown models.

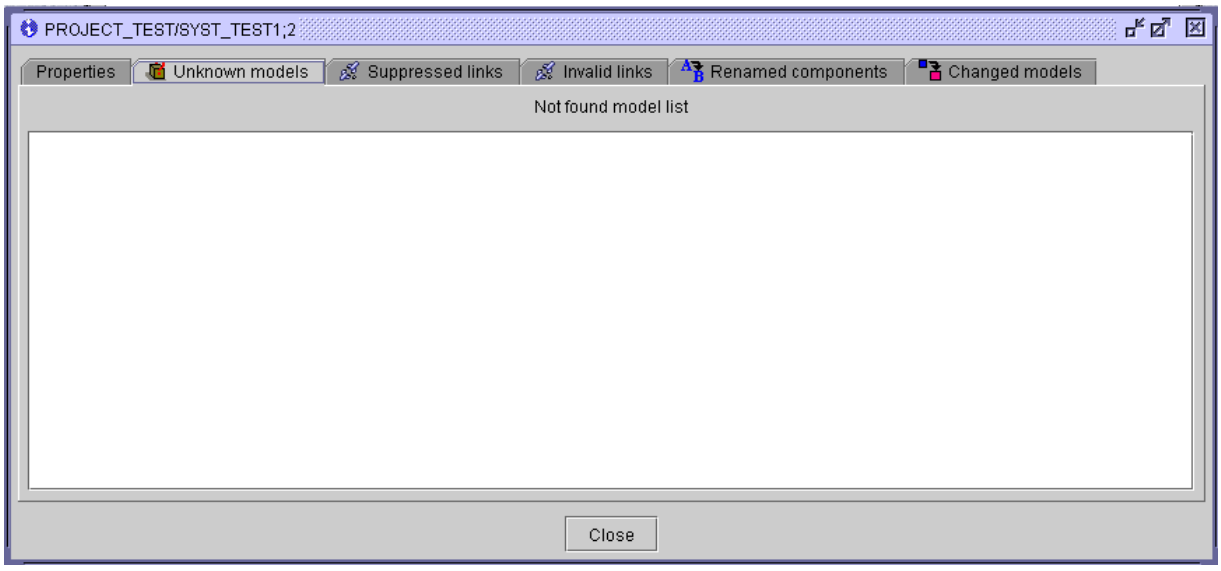


Figure 173 : Information –Unknown models tab

Suppressed links tab

Click on the **Suppressed links** tab (Figure 174); it indicates information for each of the following elements: component, component model, port, port type and field of type.

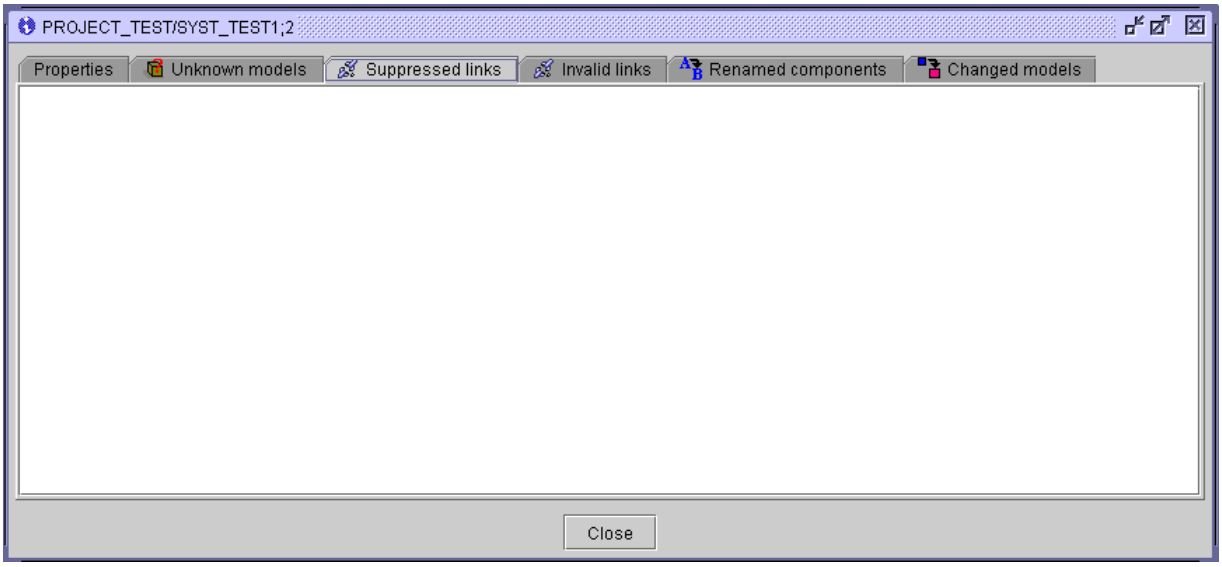


Figure 174 : Information – Suppressed links tab

Invalid links tab

Click on the **Invalid links** tab (Figure 174); it indicates information for each of the following elements: component, component model, port, port type and field of type.

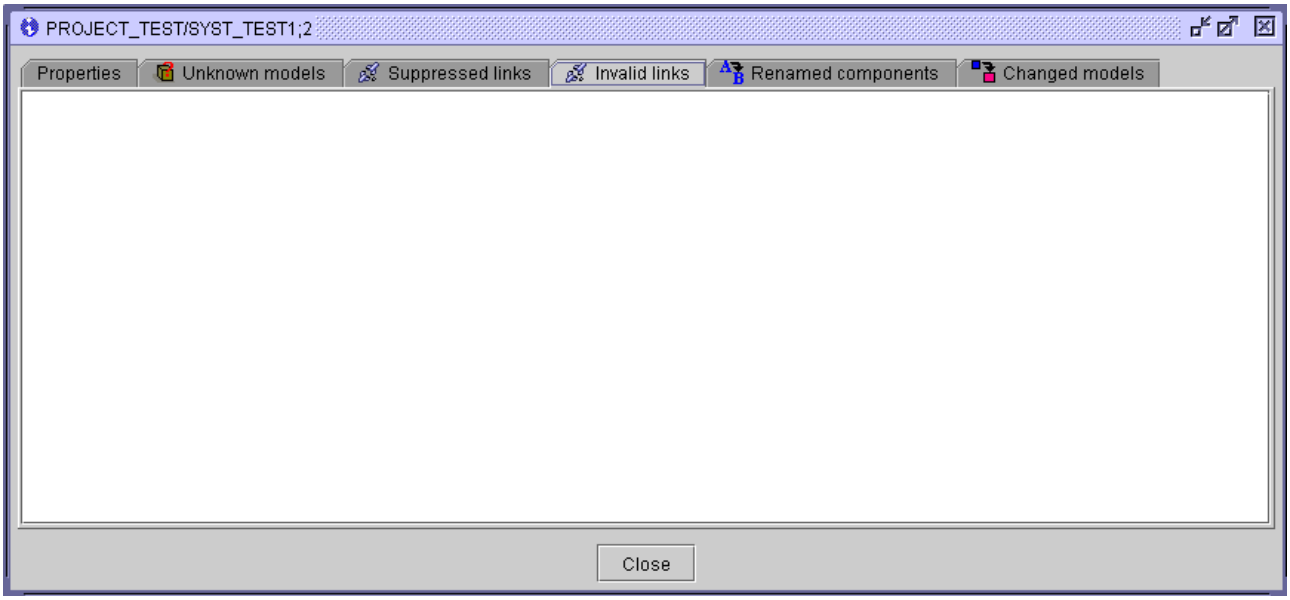


Figure 175 : Information – Invalid links tab

Renamed components tab

Click on the **Renamed components** tab (Figure 176).

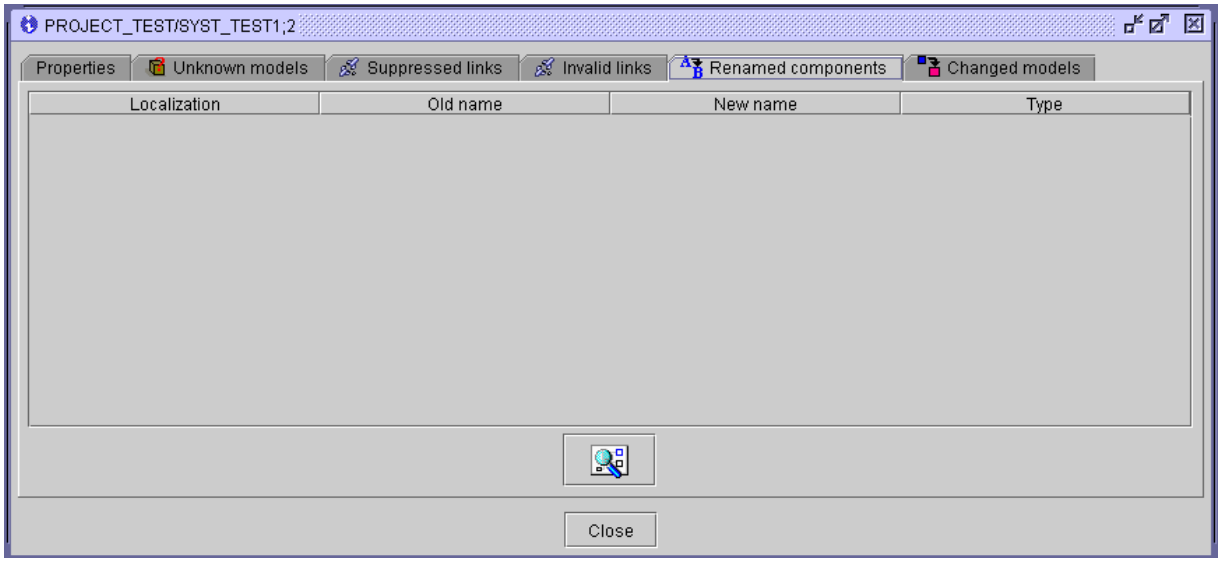


Figure 176 : Information – Renamed components tab

The window displays the following information concerning components which have been renamed in an architecture:

- Localization,
- Old name,
- New name,
- Type.

Changed models tab

Click on the **Changed models** tab; the window (Figure 177) indicates the models, the ports and the states.

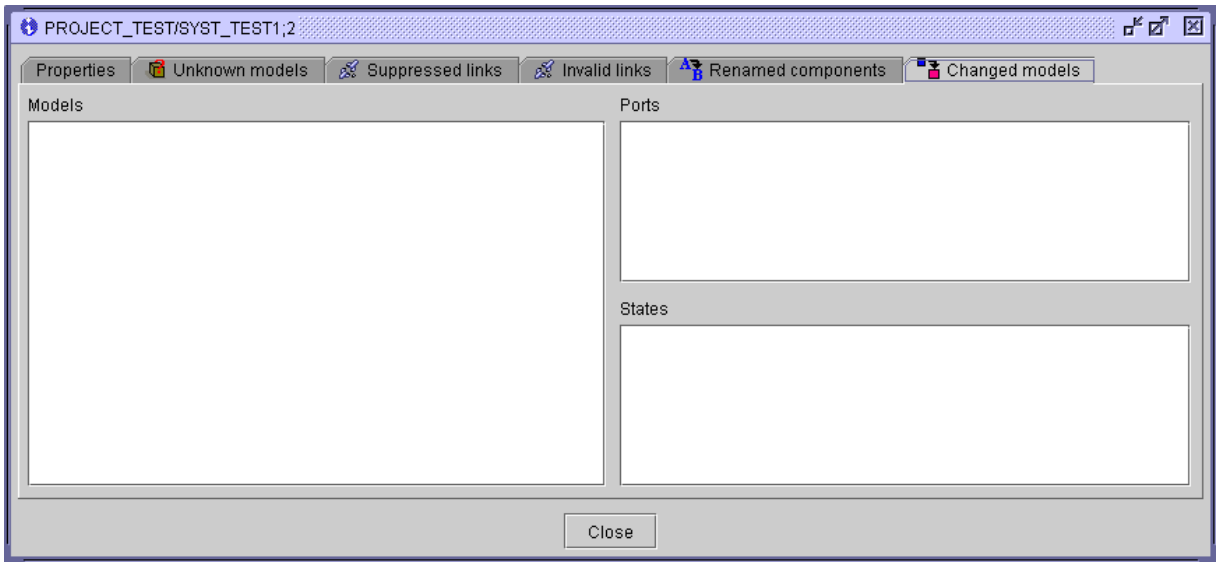


Figure 177 : Information – Changed Models tab

SYSTEM LINKS

Commands

- Use the **Views – System links**, or in the system contextual menu, click on the **System links** command to display the corresponding information.
- The *System links* window is displayed; it contains two tabs:
  - System assertions,
  - Links.

System assertions tab

A system assertion is a link which has no graphical representation.

- Click on the **System assertions** tab; the following window (Figure 178) is displayed.
- In the left part of the window, click with the mouse left button on the “main“ hierarchical level, then double click to display the hierarchical levels; select a hierarchical level.
- The *Edition* area displays the variable assignment in the **System assertions**.

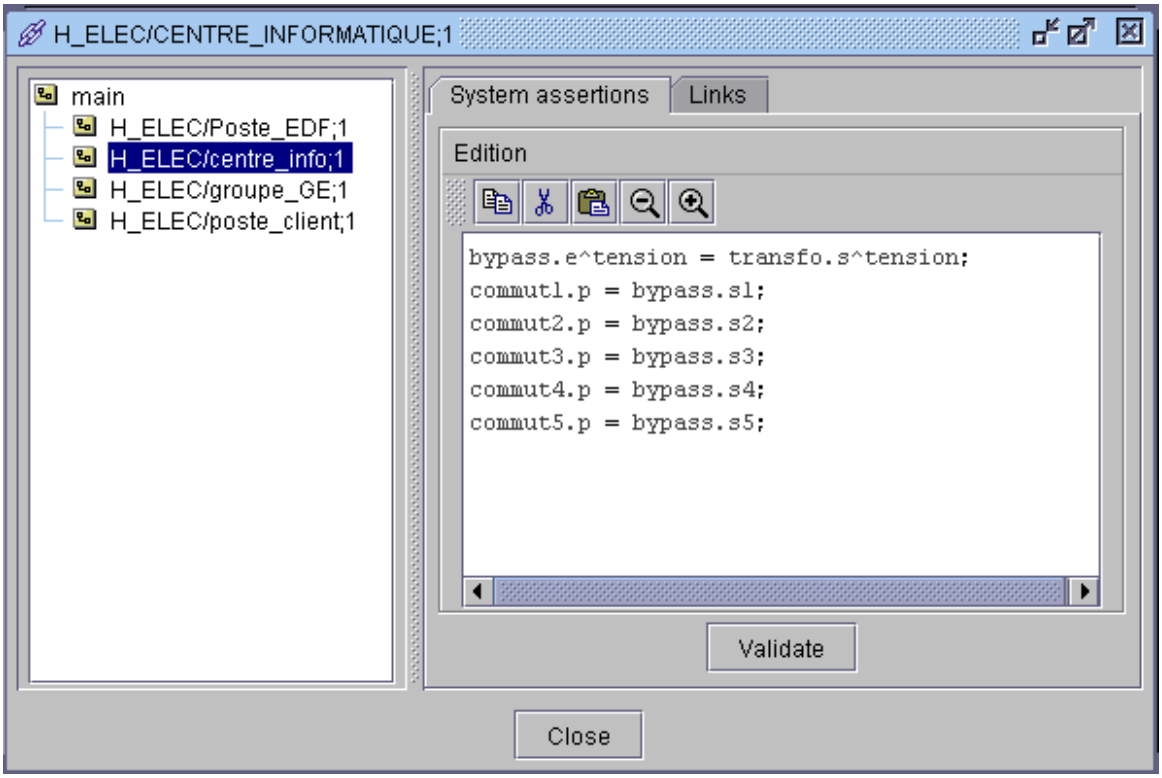


Figure 178 : System assertions tab

Remark: System assertions are links which exist but are not displayed in the graphical view; they concern only flow assignments.

- The syntax for the system assertions is:

  - Simple affectation between “flow in” and “flow out” with same type.

    <component_i>.<In_Name> = < component j>.<OutName>;

  - Simple affectation between “flow in” and a specific value defined with a same type.

    <component_i>.<In_Name> = true ;

    <component_i>.<In_Name> = low ;

  - Affectation using operators:

    <component_i>.<In_Name> = (<component j>.<OutName> or
    <component k>.<OutName>);

    <component_i>.<In_Name> = my_operator (<component _j>.< OutName >,
    < component _k>.<OutName S>);

- It is possible to get the port name in the clipboard by selecting a port in the system model: right click to display the contextual menu and choose **Copy port name**. The name of this port can be pasted in the equipment Altarica code.

Five icons are available to edit system assertions:

- : Copy,
- : Cut,
- : Paste,
- : Backward zoom,
- : Forward zoom.

Remark: the validity of these system assertions will be checked by the system consistency control (activated with the command **Check properties** from the **System** menu).

Caution: In system architecture, assertions can only be written at the system level (**main** node). For hierarchical level associated to equipments, only a viewing of assertions defined in library is possible (in equipment model editor).

Links tab

A link defines an interconnection between two objects in architecture.

- Click on the **Links** tab; the following window (Figure 179) displays an exhaustive list of all the links which have a graphical representation in a given hierarchical level.
- In the left part of the window, click with the mouse left button to display the “main” hierarchical level, and then double click to display the hierarchical levels; select a hierarchical level.

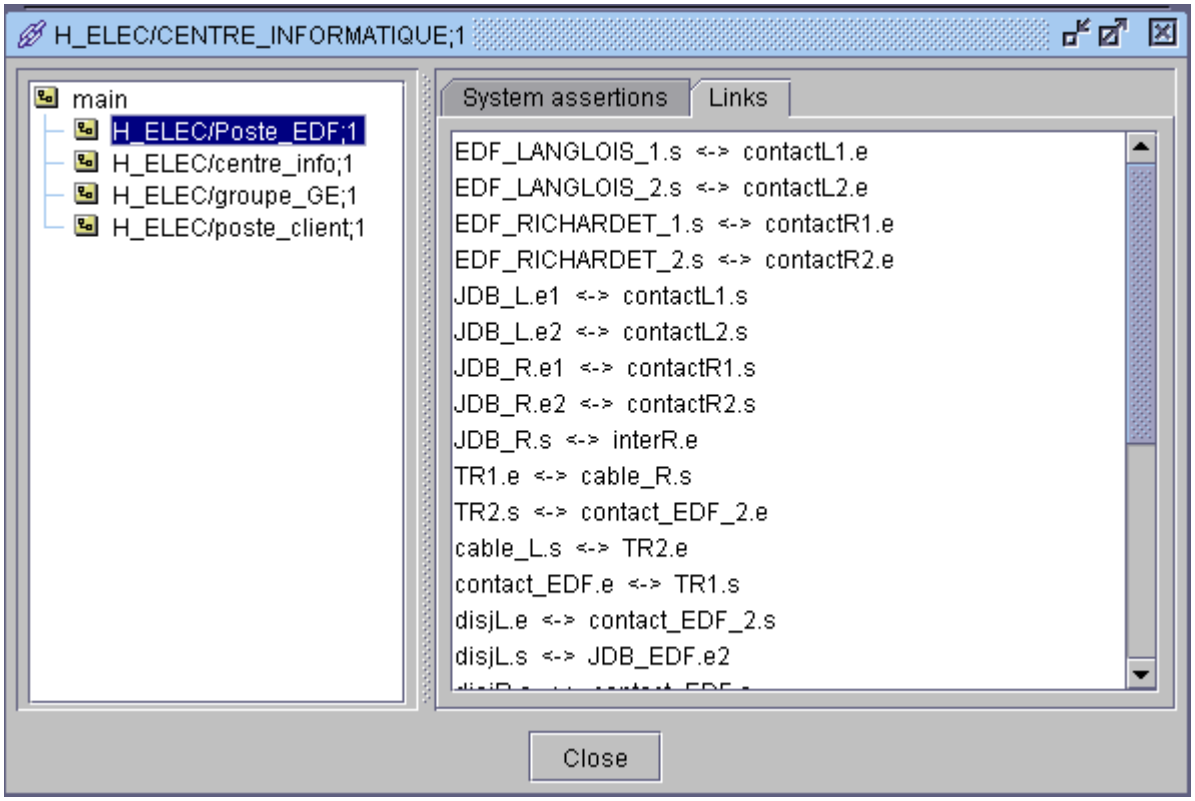


Figure 179 : System assertions – Links tab

LINK COLOR

- Open a system architecture.
- Use the **Views – Colours** menu to define the links colour in a system during simulation.
- The Links colours window (Figure 180) is displayed: it contains in its right part a colour palette, and in the left part, an area displaying the link colours according to their type.
- When a new architecture is created, the boolean type is the default type with two colours (Figure 180):
  - False: red,
  - True: green

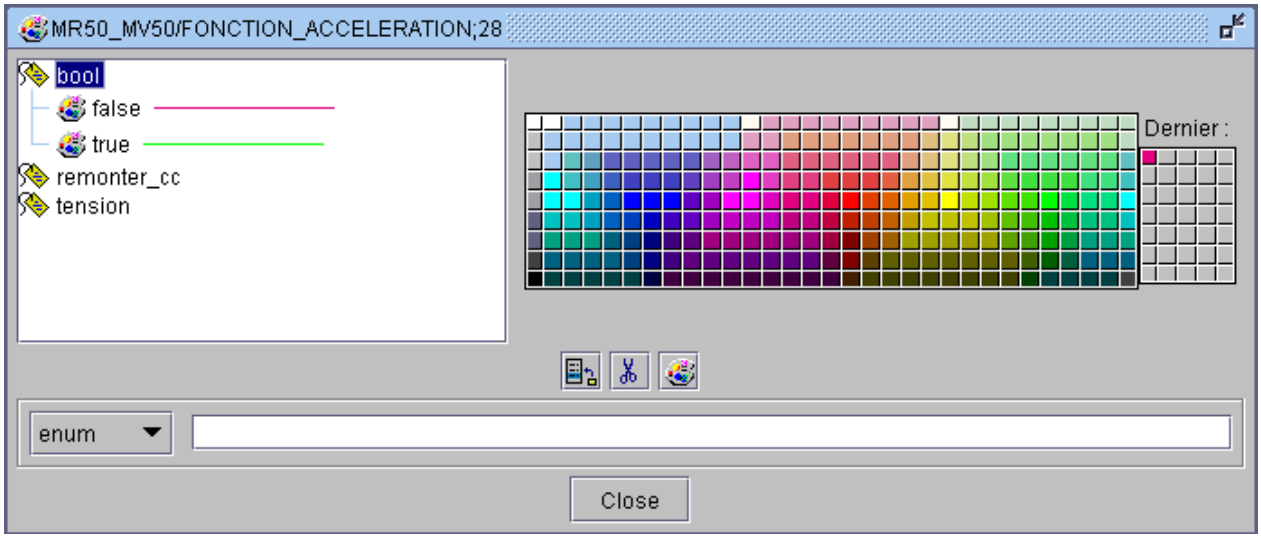


Figure 180 : Boolean variable colours

- If a new architecture contains links (non-boolean) which must be coloured, follow the procedure:
  - Enter a type in the text edit zone (enum, bound or predefined).
  - Click on the icon or use the **Enter** key to add the type in the list.
  - E.g.: To select a predefined type variable, enter *pressure* in the **enum** field; the following window is displayed (Figure 181) :

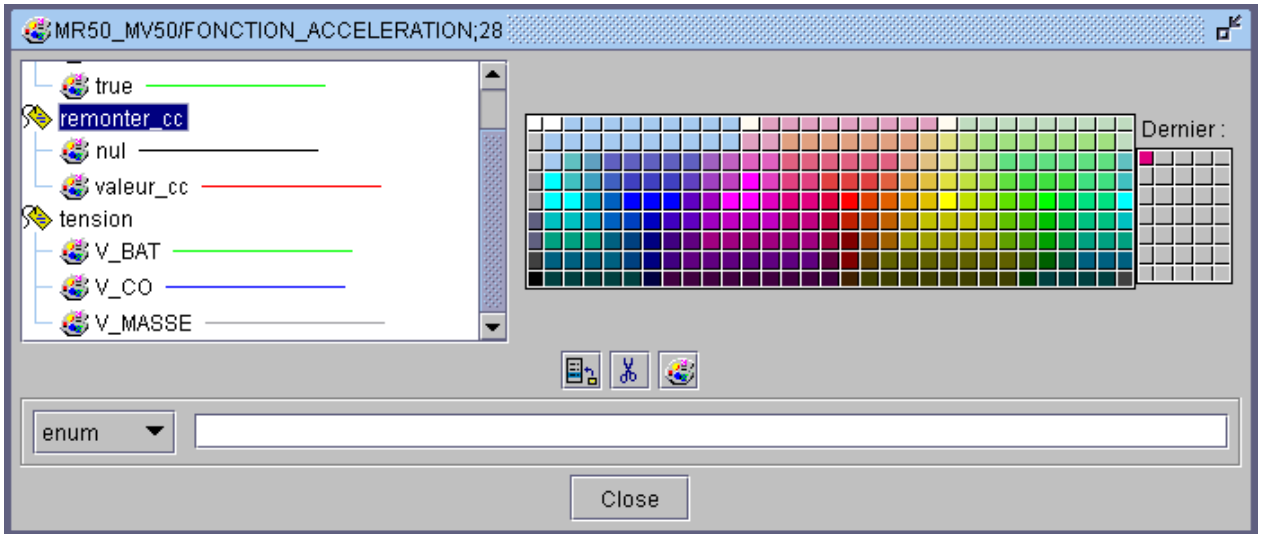


Figure 181 : Links colour

  - Double click on the type in the list to display the possible values, e.g., *pressure*; the four possible values for pressure are displayed:
    - High,
    - Low,
    - Normal,
    - Null,
  - Select a value, e.g. *Normal*.
  - In the colour palette, select a colour; the grid on the right indicates the last selected colours.

- Click on the  icon to assign the color to the value;
- Repeat the same steps to define the colours for the three other values.

N.B: The color assignment is specific for a system; the colors are saved with the system.

Remark: An existing enumerated type can be deleted by clicking on the corresponding icon.

Remark: When a link is of the record type, the colors displayed during the simulation correspond to the first flow field.

PRINTING

Prepare the printing of a model or system or of the *System links* window as follows:

- Select the **File – Printing** format menu.
- The **Page Setup** window (Figure 182) is displayed; in the ***Paper*** area, select:
  - The size (A4 or A3),
  - The source (manual feeding or automatic selection).
- In the ***Orientation*** area, tick the *Portrait* or *Landscape* mode.
- Each format or orientation modification automatically updates the scale factor of the page displayed.
- In the ***Margin*** area, enter the margins (left, right, up or down).
- Click on the **OK** button to save the new printing format, or click on the **Cancel** button to return to the previous format.

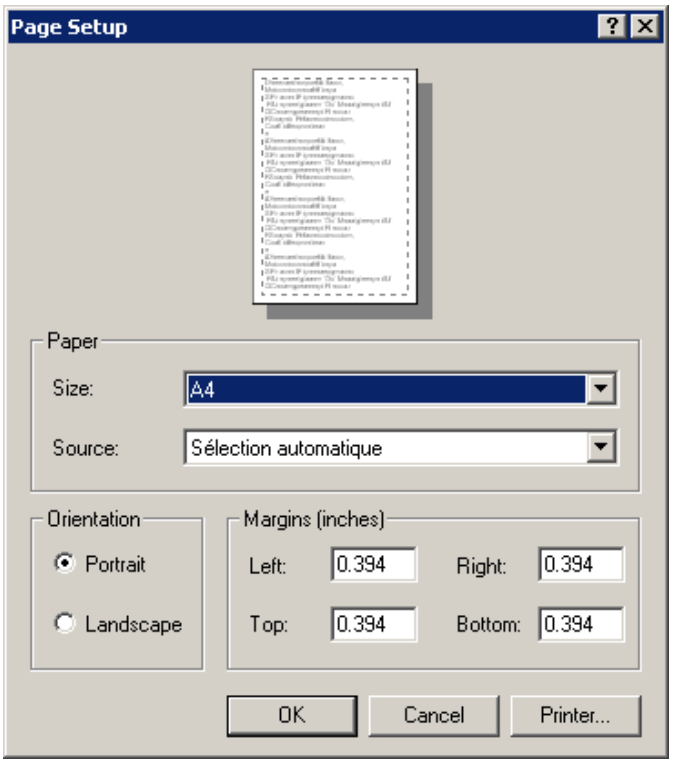
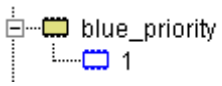


Figure 182 : Page Setup

Print a model as follows:

- Select the model version you want to print in the library:



- Edit model by right-clicking on the version, select **Model – Edit** in contextual menu
- Select the **File – Print** menu or click on icon .

Print a system as follows:

- Select the system version you want to print.
- Open **System** (double-click)
- Select the **File – Print** menu or click on icon to print selected system (Figure 183);
- Select the pages to be printed.

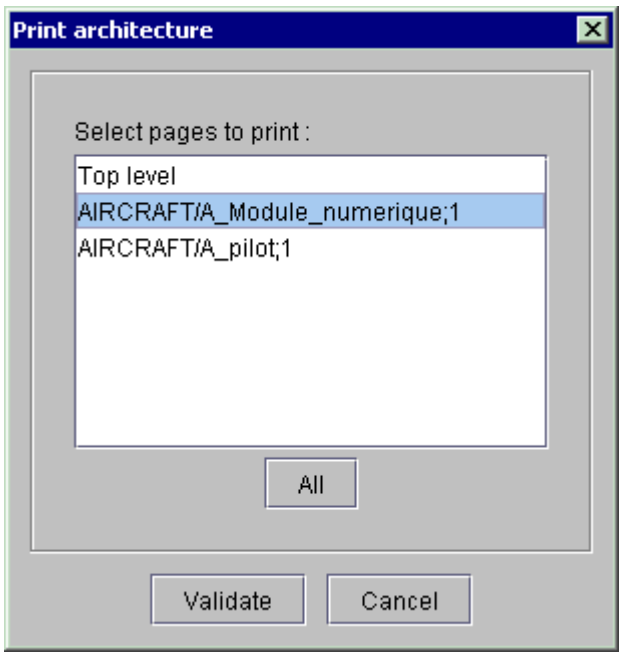


Figure 183 : Print architecture window

- Click on the **Validate** button, or click on the **Cancel** button to abort the printing.
- The *Printing* window is displayed (Figure 184).
- In the *Printing area,* select the area to be printed *(*all, pages *n* to *m* or selection*).*

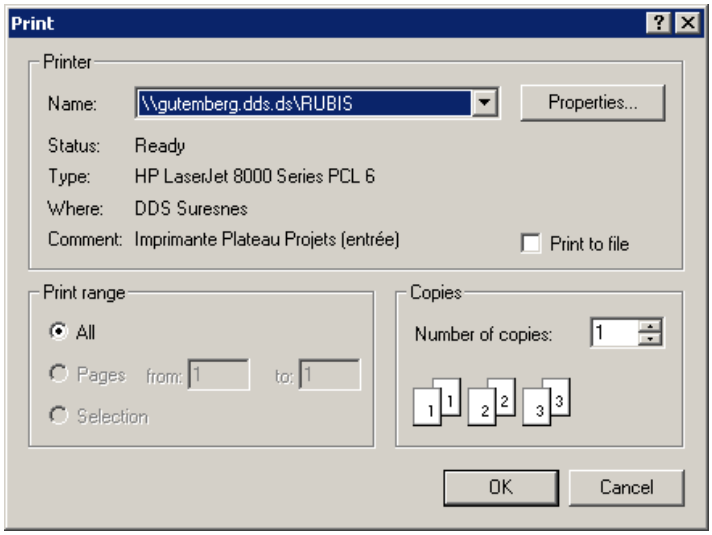


Figure 184 : Printing window

Remark: In the **Preferences** window – **Options** tab, it is possible to select a file containing a logo which will be added.

EXPORT/IMPORT DATA INTO XML FORMAT

In order to facilitate data exchanges, DAS allows export and import of DAS object (family, project, models, and systems) in XML Format.

EXPORT IN XML FORMAT

To export data, proceed as follows:

- Activate the export setup window (Figure 185) with the **Export** command in **File** menu.

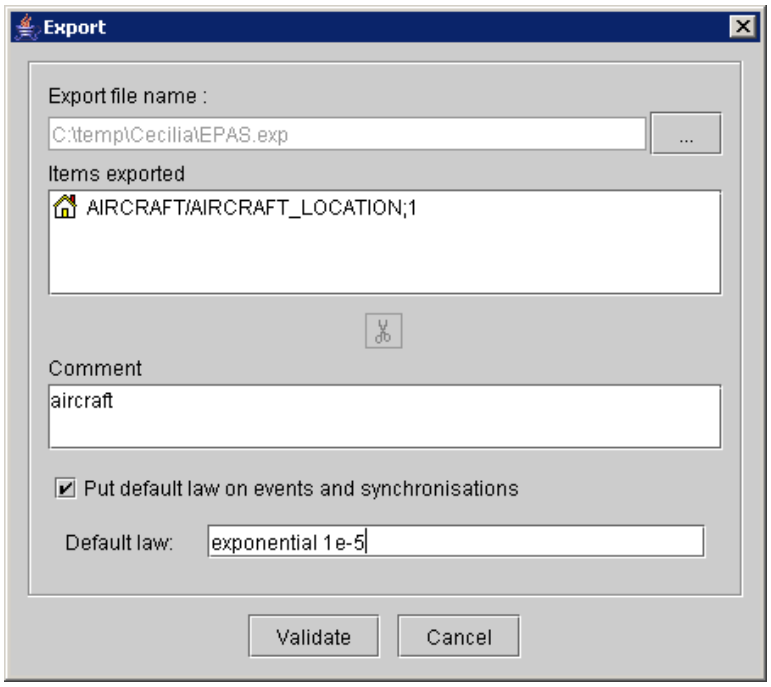


Figure 185 : XML export setup

- Click on ... button in **Export file name** area,
- Chose the export file name and the directory,
- add elements to export. Select an element in the library and use the **File/Add to Export** menu, or drag and drop the object in **Items exported** area.
- The **Comment** area allows to add a description.
- Tick the box **Put a default law on events and synchronizations** to add a default law on events and synchronizations that have no law. Use the **Default law** area to specify the default law
- Click on **Validate** button to export or on **Cancel** to abort exportation.
-

WARNING : Frozen and locked models can't be exported.

IMPORT IN XML FORMAT

To import SD9 data from an XML import file (.exp extension) generated by SD9, proceed as follows:

- Display the import window with the **Import** command of **File** menu.

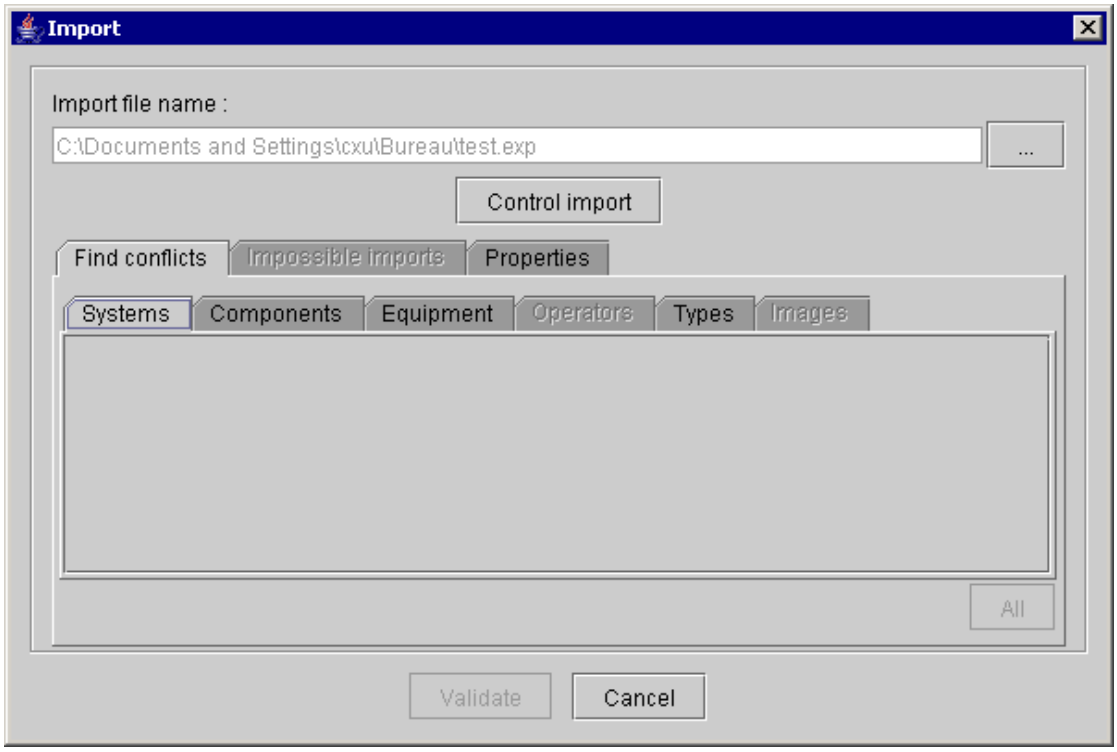


Figure 186 : XML Data Import

- Click on the browser button [...] in **Import file name** area,
- Choose the export file (extension .exp) containing DAS data. The **Properties** tab display properties associated to the imported data (creation date, comment, and owner).
- Click on **Control Import** to launch validity check. This check do the following operations:
  - *XML Format validity Check.* Detected errors are displayed as "impossible imports.

    Remark: only ascendant compatibility is ensured.

  - *Detection of conflict between imported data and data in database*.
    Remark: This detection is done essentially by comparing the last modification date of each imported element with the model in database.

    Detected conflicts are displayed in the following tabs: **Systems**, **Components**, **Equipments**, **Operators**, **Types**, **and Images.** By default, data in database are not updated during data importation.

    Nevertheless, the user can specify, for each conflict, a possible update of the database by clicking on  (Figure 187). The **All** button allows updating (importing) all models in conflict displayed in the tab.

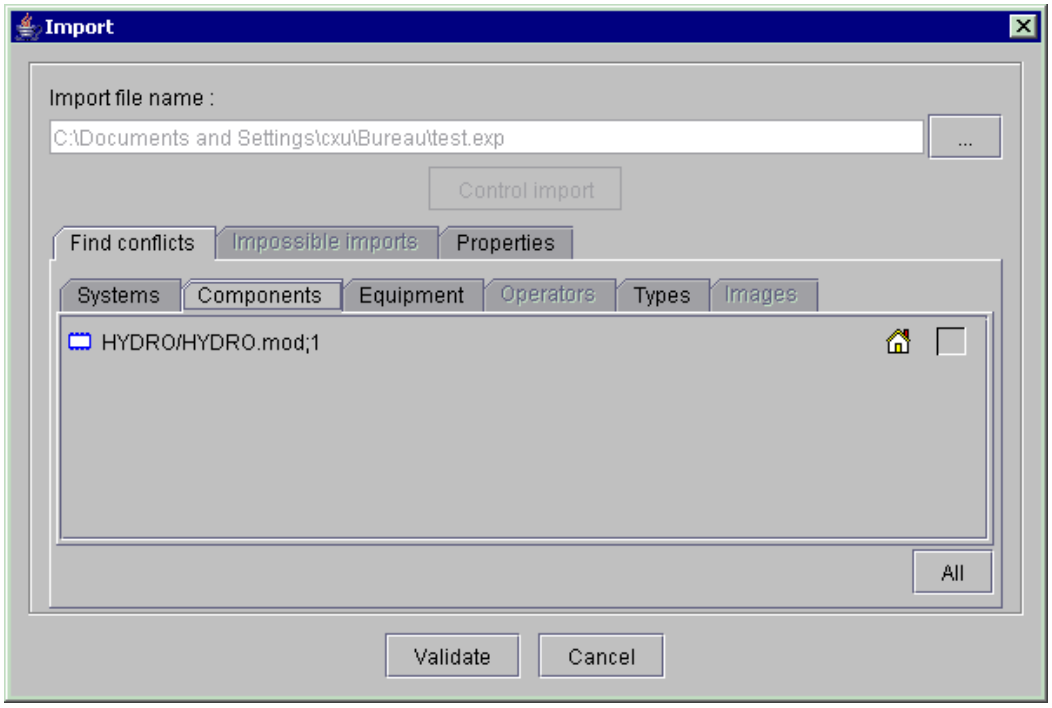


Figure 187 : XML Import – Detected conflict

- Click on **Validate** to import or **Cancel** to abort importation.

THE PLUGIN MANAGER

INTRODUCTION

The plugin manager allows the user to manage the compute plugins. The compute plugins are processing units taking in input a model generated in Altarica. According to the treatment made, the user will obtain on output event minimal cuts, event sequences, a step by step simulator by inference rules or a Java step by step simulator.

A user can create new plugins and add them inside SD9 by using the plugin manager. This chapter does not present the plugin creation process. This section is developed in details in another document.

From their use point of view, the user sees the plugins like treatment units contained in JAR files (in Java). A JAR file can contain several plugins. These files are read by the plugin manager in order to load the plugins into the application during its launch.

GENERAL DESCRIPTION OF THE PLUGIN MANAGER PANEL

The plugin manager window contains three tabs which are: **Plugins**, **Actions** and **Items** (Figure 188):

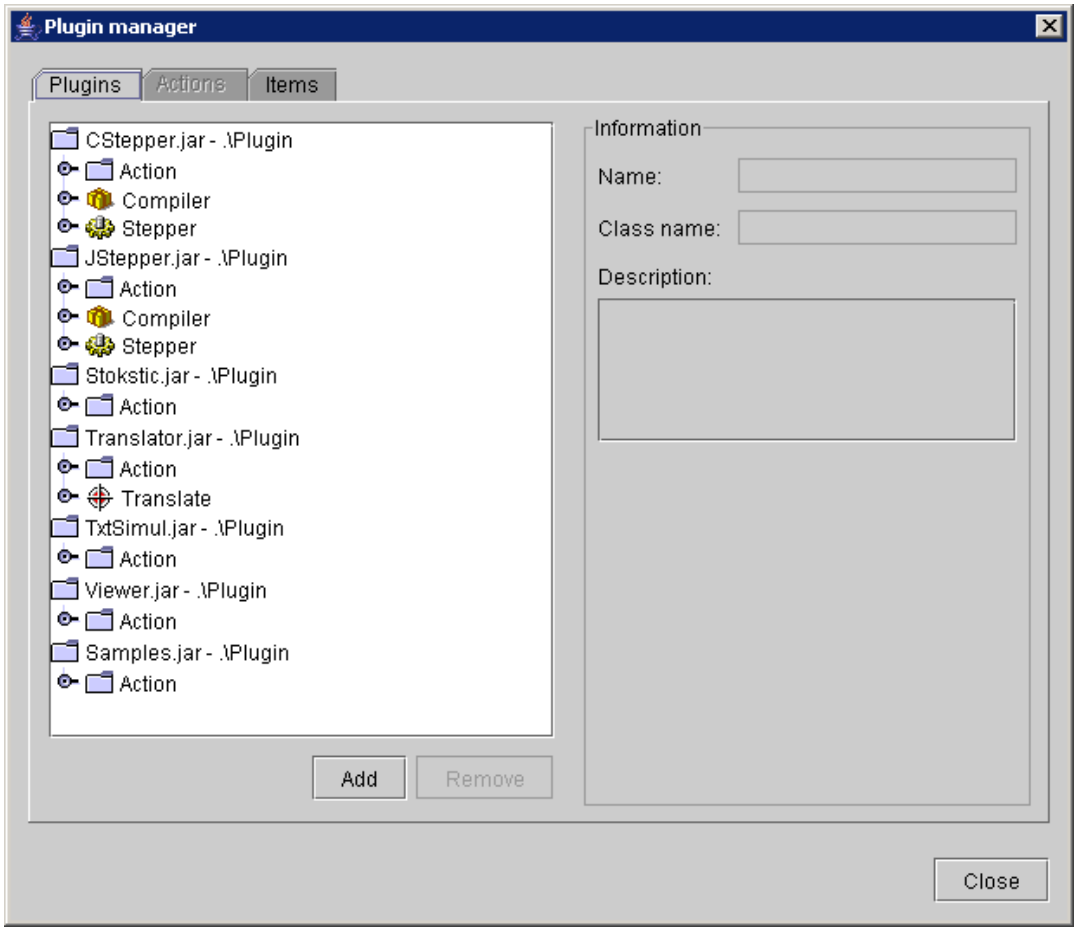


Figure 188 : Plugin manager

The **Plugins** tab allows the user to manage the plugins. The **Actions** tab allows adding actions associated to the plugins and the **Items** tab allows inserting inside the application user interface, plugin linked action launch point.

TAB PLUGINS

The **Plugins** tab allows the plugin management inside SD9. Through this tab the user can add, remove plugin to/from the application and also view the plugin properties.

Plugin organization

Plugins are organized in a tree structure. The first level shows the different JAR files from which the plugins are loaded into the application. The plugin paths are expressed in relative from the location of the file named “plugins.xml” (Figure 189):

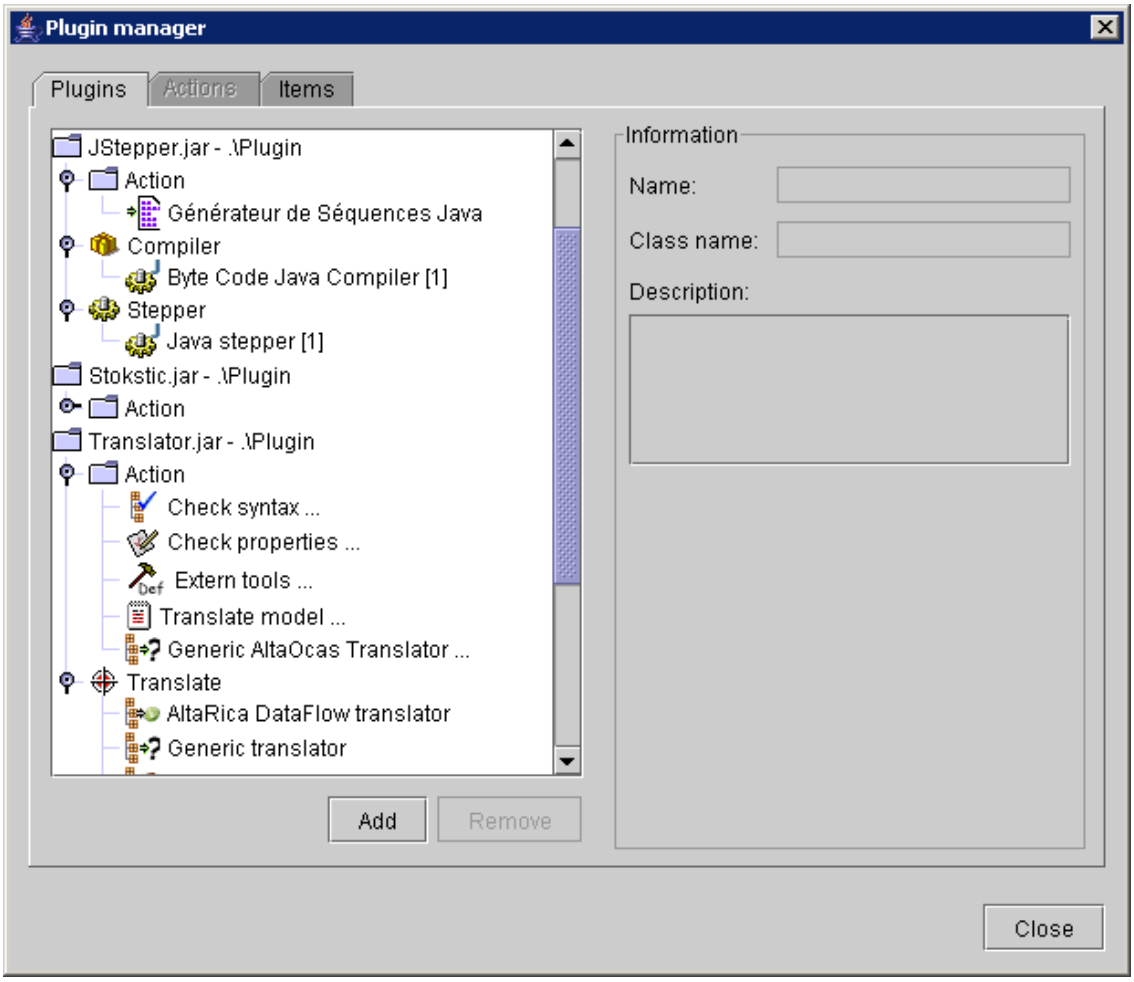


Figure 189 : Plugin manager – plugin tree structure

The second level contains **Action, Translate, Compiler** and/or **Stepper** nodes. The nodes allow classifying plugins into categories.

The **Action** nodes gather plugins that can be directly executed by the user if associated to an action and a launch point in the application user interface.

The **Translate** nodes gather plugins used in treatment flows and are dedicated to the translation of some languages into other languages.

The **Compiler** nodes gather plugins allowing compilation of models which accelerate its use by the simulation engine.

The **Stepper** nodes gather the plugins that implement simulation engines.

View plugin informations

In order to visualize plugin related information, the user selects a plugin in the tree structure. The information then appears in the right part of the tab (Figure 190):

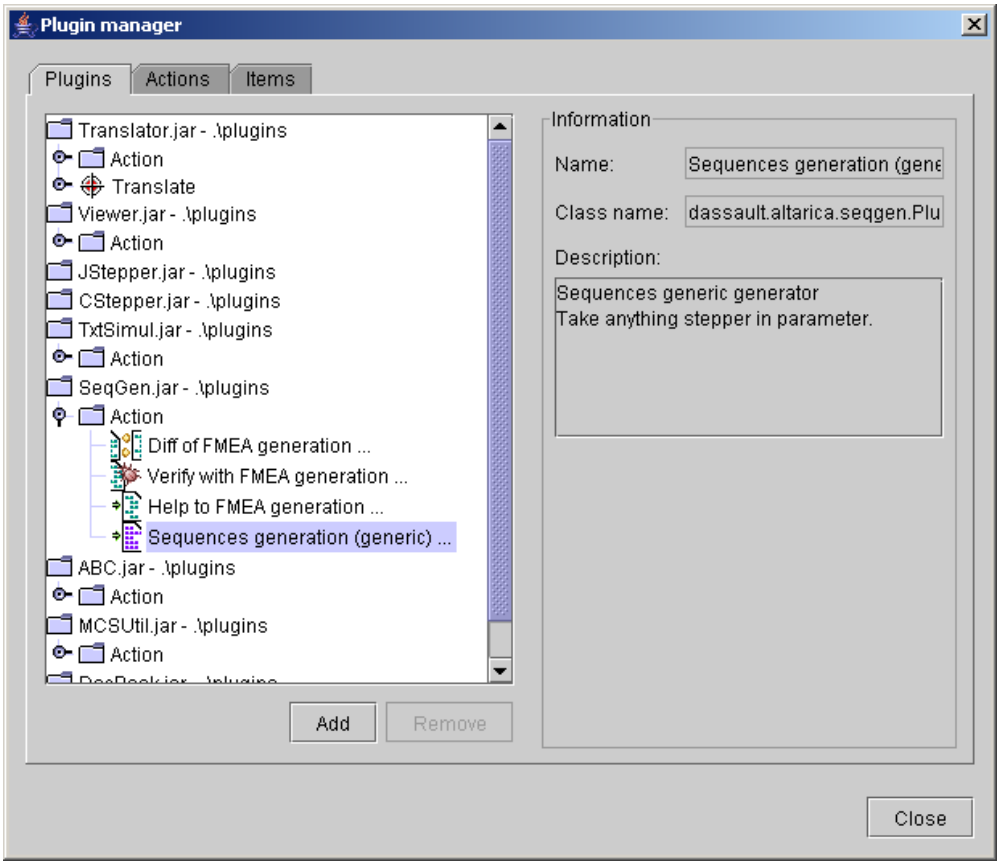


Figure 190 : Plugin manager – Plugin information area

A plugin has a name, a class name and a description. This information is accessible to the user but is not editable for write because it is to the plugin designer responsibility.

The class name is the complete name of the implementation Java class. It is the plugin entry point.

Add JAR files

Plugins are only accessible throughout JAR files. The user load JAR files containing plugins and not plugins directly.

To add JAR files containing plugins into the application, the user must proceed as follows:

- Click on the **Add** button; the following window then appears (Figure 191):

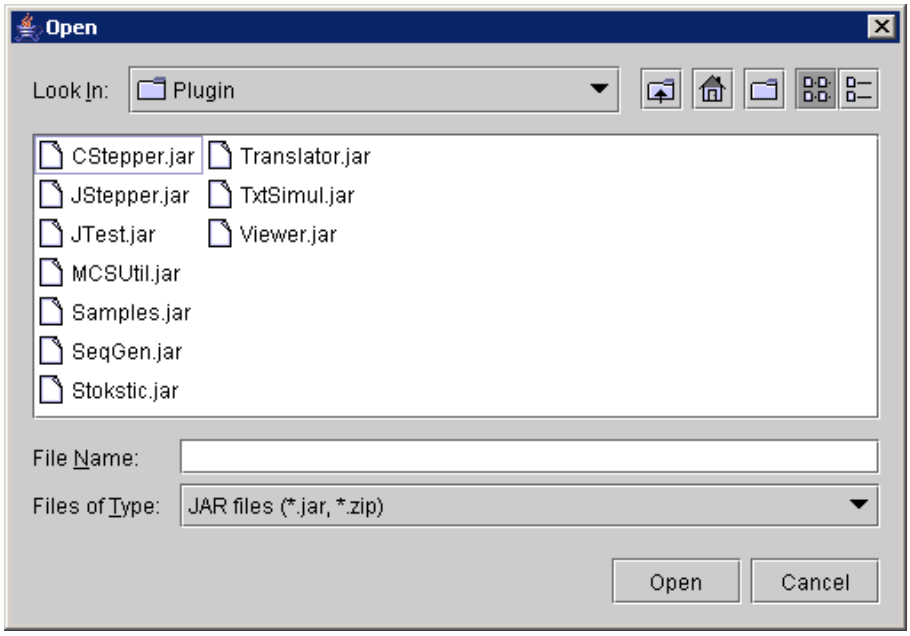


Figure 191 : Add plugins contained in JAR files

- Select the JAR files containing the plugins that are to be integrated into the application,
- Click on the **Open** button.

The plugins are registered in the application and then can be referenced by plugin actions.

Remove JAR files

To remove JAR files from the application, the user proceed as follows:

- Select JAR files nodes that are to be deleted,
- Click on the **Remove** button; the following confirmation message then is shown (Figure 192):
-

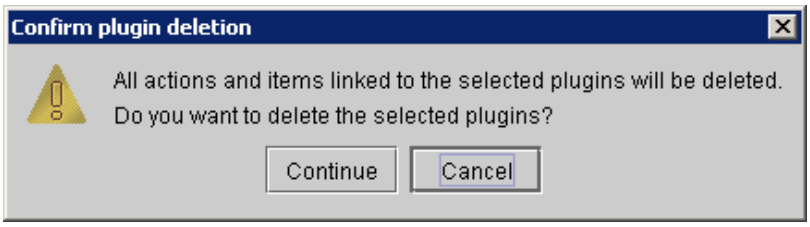


Figure 192 : JAR files deletion confirmation message

- Click on the **Continue** button to confirm the JAR files deletion or click on the **Cancel** button to cancel the action.

If the user clicks on the **Continue** button, the selected JAR files nodes and all their content are removed from the tree structure.

TAB ACTIONS

The actions are plugin instantiations in the sense that they allow using the plugins with distinct parameters. The actions only apply to the **Action** plugins and not to the others categories. When the user select a plugin of **Translate, Compiler** or **Stepper** type, the **Action** tab become inaccessible.

The **Action** tab appears as follows (Figure 193):

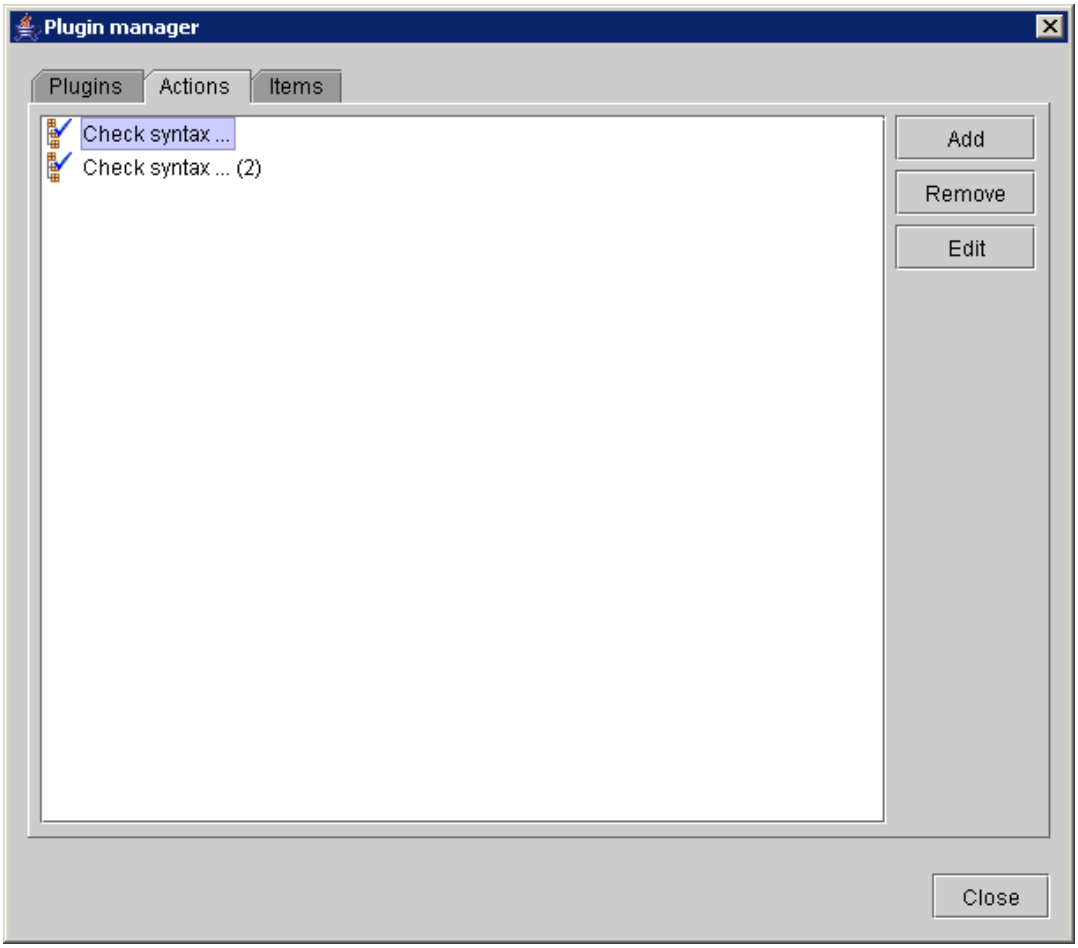


Figure 193 : Plugin manager –Actions tab

Add plugin action

To add a plugin action, the user proceeds as follows:

- Select if necessary the **Plugin** tab and then, in the tree structure, the plugin of **Action** type on which the new action is to be created,
- Select the **Action** tab,
- Click on the **Add** button; the following window then appears (Figure 194):

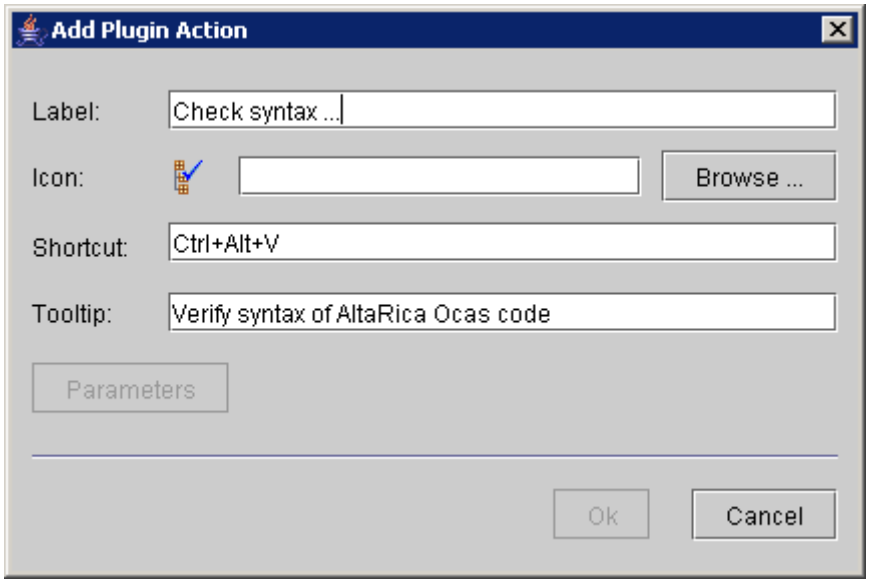


Figure 194 : Add a plugin action

- Fill the dialog box fields in,
- Click on the **Ok** button to confirm the plugin action creation or on the **Cancel** button to do nothing.

If the user clicks on the **Ok** button, the new action is added to the already existing actions list.

N.B.: A default label fill the corresponding field when the **Add a plugin action** dialog box is shown to the user. This label is provided by the plugin. It is the same for the **Icon, Shortcut** and **Tooltip text** fields.

Two different actions associated to the same plugin can not have the same label. When an already existing label is entered in the corresponding field, the **Ok** button is then disabled.

The **Parameters** button give write access to the plugin parameter list when the plugin has parameters. These parameters and their viewing are provided by the plugin designer.

Remove plugin action

To remove plugin actions, the user proceed as follows:

- Select the **Actions** tab if needed,
- Select the action nodes ,
- Click on the **Remove** button; the following action deletion confirmation message then appears (Figure 195):

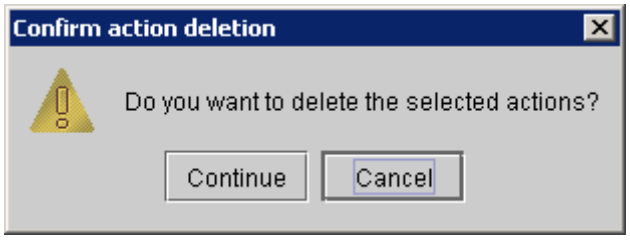


Figure 195 : Plugin deletion confirmation

- Click on the **Continue** button to confirm the actions deletion or on the **Cancel** button to do nothing.

If the user clicks on the **Continue** button, the selected actions then disappear from the action list.

Edit plugin action

To edit a plugin action, the user proceeds as follows:

- Select if necessary the **Action** tab and the action to edit,
- Click on the **Edit** button; the following window then appears (Figure 196):

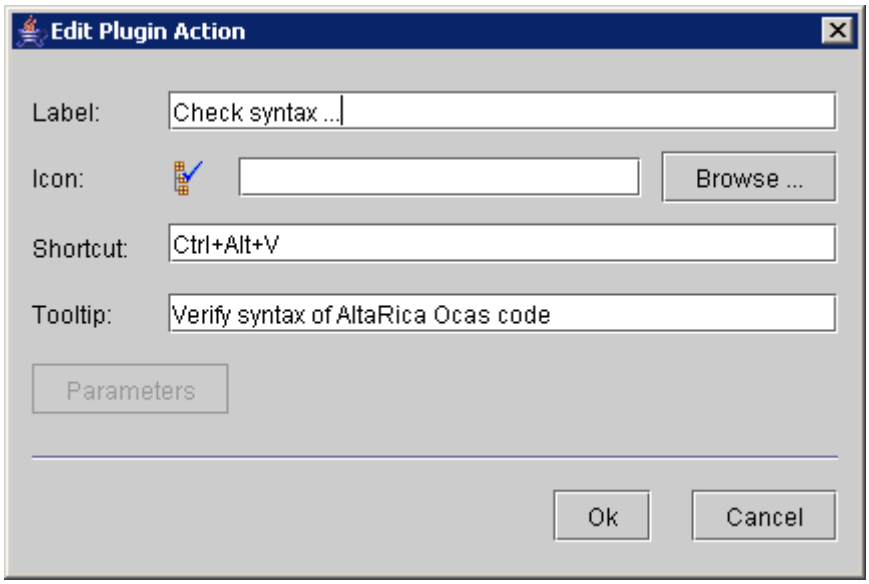


Figure 196 : Edit a plugin action

- Modify the fields in the dialog box,
- Click on the **Ok** button to confirm the modification or on the **Cancel** button to cancel the operation.

TAB PLUGIN ITEMS

Plugin items overview

The **Items** tab allows the plugin item management. Note that the plugin items are user interface graphical objects like buttons or menu items that provide a launch point for the **Action** typed plugins. By the other, a plugin item action is an action defined or predefined in the **Action** tab. When created on a tool bar, the plugin item is a button and created on a menu bar, the plugin item is a menu item.

The Items tab is shown bellow (Figure 197). The **Items** tab is divided in two parts: a first part that shows all the application menu items and toolbar buttons in a tree structure view and a second one that gather the commands accessible on the plugin items.

By navigating in the tree structure representation, the user can for instance reach a sub-menu in order to add a plugin launch point. The user has also the possibility to create other type of objects like separators or new menu in existing ones. All the supplementary graphic objects appear in the tree structure with bold text font.

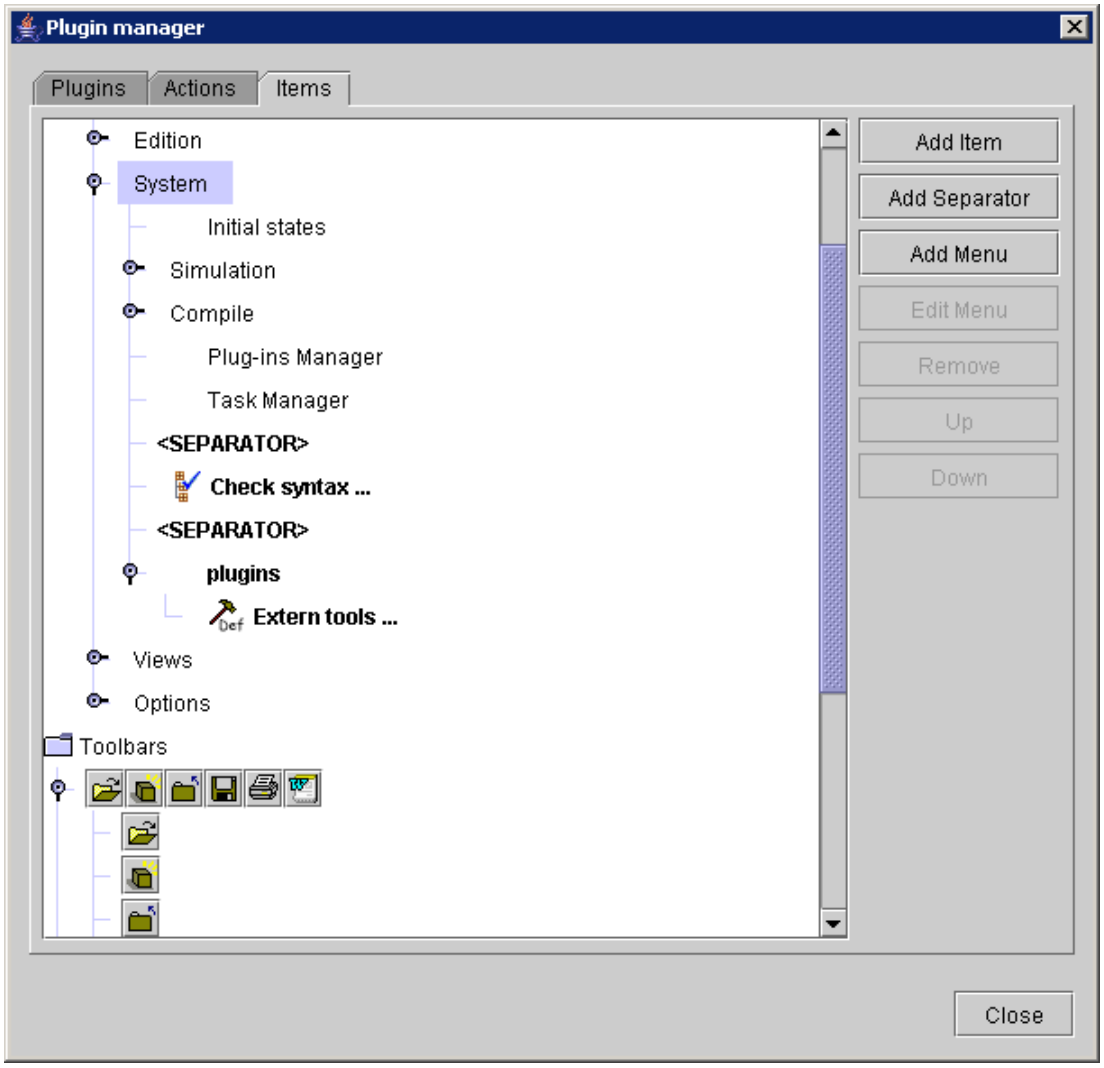


Figure 197 : Plugin manager –Items tab

Add plugin item

To add a plugin item, the user proceeds as follows:

- Select the **Plugins** tab then a plugin of Action type if necessary,
- Select the **Action** tab then an action in the list if necessary,
- Select the **Items** tab if necessary,
- Navigate in the tree structure until the insertion location,
- Click on the **Add Item** button.

The plugin item is then created and added to the tree structure and in the application menubar or toolbars. When the selected object for the item insertion is a menu, the item is inserted at the last position in that menu, when it is a menu item the insertion is done at the preceding position of that menu item.

Add a separator

To add a separator, the user proceeds as follows:

- Select the **Items** tab if necessary,
- Navigate in the tree structure until the insertion location,
- Click on the **Add Separator** button.

The separator is then created and inserted in the selected location. When the selected object for the insertion is a menu, the separator is inserted at the last position in that menu, when it is a menu item the insertion is done at the preceding position of that menu item. Separators created by the user have all the same name:

<SEPARATOR>

Add customized menu

To add a menu, the user proceeds as follows:

- Select the **Items** tab if needed,
- Navigate in the tree structure until the insertion location,
- Click on the **Add Menu** button, the following dialog box then appears (Figure 198):
-

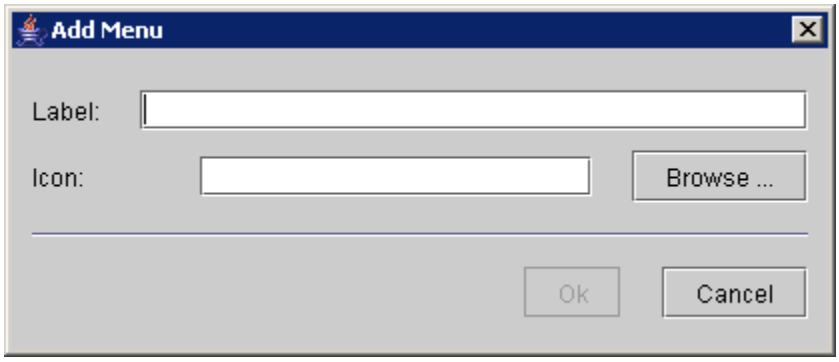


Figure 198 : Add menu

- Fill the label field in and if needed an icon path in the corresponding field,
- Click on the **Ok** button to validate the creation or on the **Cancel** button to cancel the operation.

If the user clicks on the **Ok** button, the newly created menu is added at the selected location. When the selected object for the insertion is a menu, the newly created menu is inserted at the last position in that menu, when it is a menu item the insertion is done at the preceding position of that menu item.

Edit customized menu

To edit a menu, the user proceeds as follows:

- Select the **Items** tab if needed,
- Select the menu to edit in the tree structure,
- Click on the **Edit Menu** button, the following dialog box then appears (Figure 199):

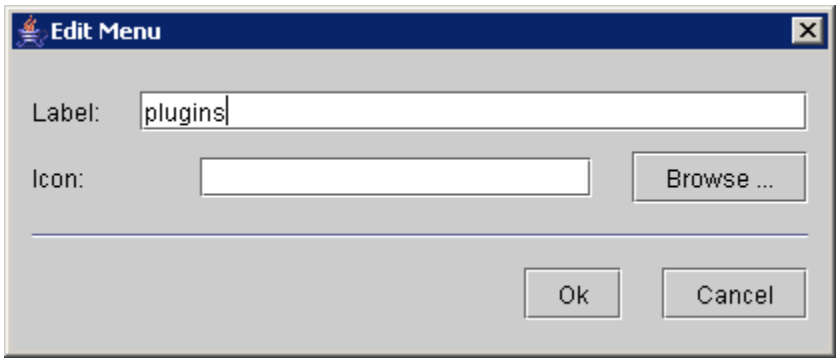


Figure 199 : Edit menu

- Modify the fields,
- Click on the **Ok** button to validate the modification or on the **Cancel** button to cancel the operation.

If the user clicks on the **Ok** button, the menu is then modified.

Remove graphic item

Only the graphic elements added by the user are removable. It can be plugin item, supplementary separator or supplementary menus.

Be careful: when the user removes a menu, all the graphic elements under this menu are also removed.

To remove graphic elements, the user proceeds as follows:

- Select the **Items** tab if needed,
- Select the graphic elements in the tree structure,
- Click on the **Remove** button; the following graphic element deletion confirmation message then appears (Figure 200):

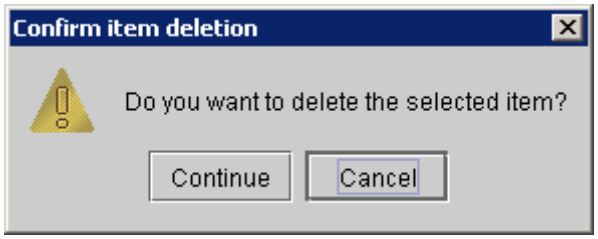


Figure 200 : Graphic element deletion confirmation

- Click on the **Continue** button to confirm the graphic element deletion or on the **Cancel** button to do nothing.

If the user clicks on the **Continue** button, the selected graphic elements then disappear from the tree structure and from the application user interface (menu bar and toolbar).

PREFERENCE CUSTOMIZATION

GENERAL

Each user can configure work environment. Thus, DAS provides a window for configuring the preferences which are available via menu **Options – Preferences**.

This menu gives access to a window containing eight rubrics:

- Environment,
- Desktop,
- Tools bars,
- Edition,
- Input/Output,
- Simulator,
- Fault Tree generation,
- Sequence generation,

ENVIRONMENT RUBRIC

N.B: Sub-rubric **Memory** (Figure 201) is for maintenance only.

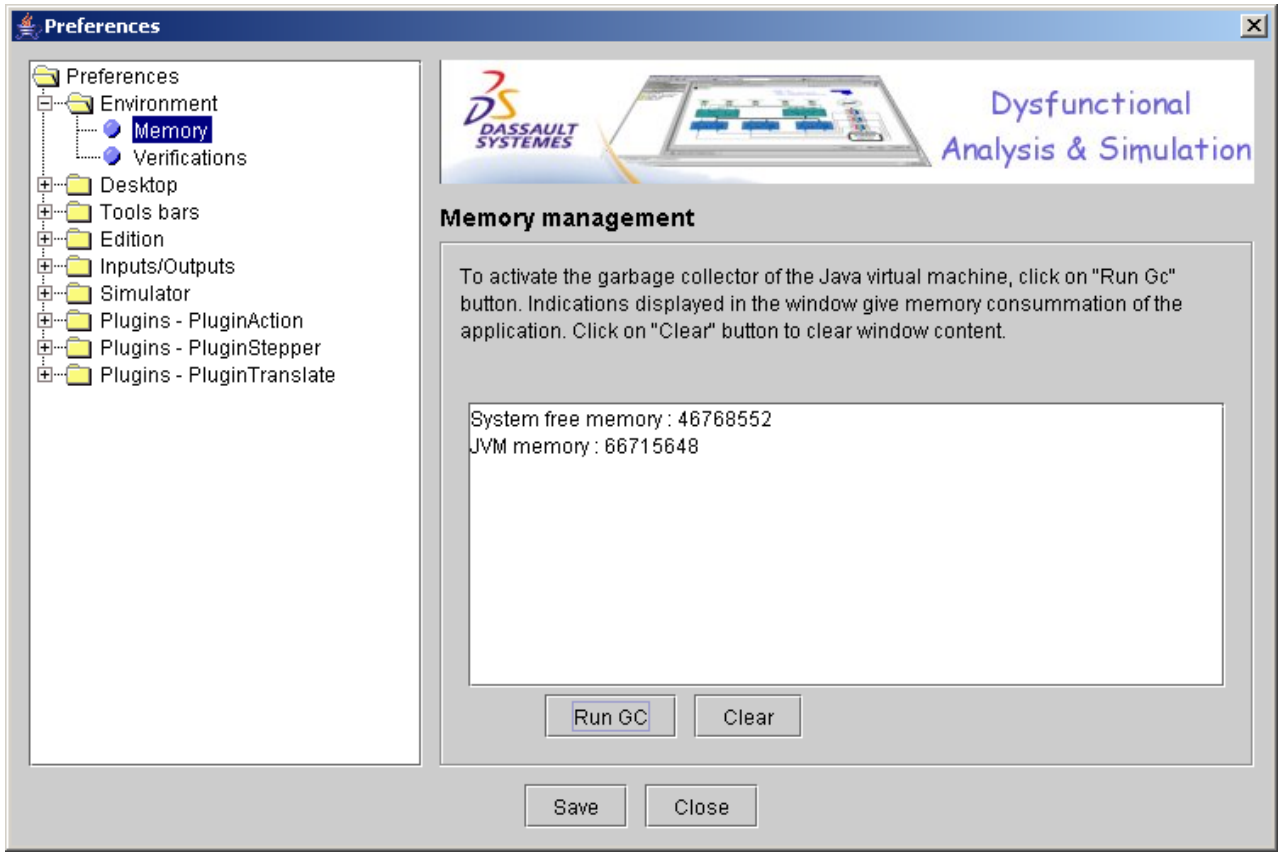


Figure 201 : Memory management

- Click on **Run GC** to run the Garbage Collector of Java Virtual Machine. Otherwise click on **Clear**.
- Click on Close to close this window.

Remark: The **Save** button as no effect in this rubric.

The Verification sub-rubric allows the user to choose the syntactic and consistency checking (Figure 202).

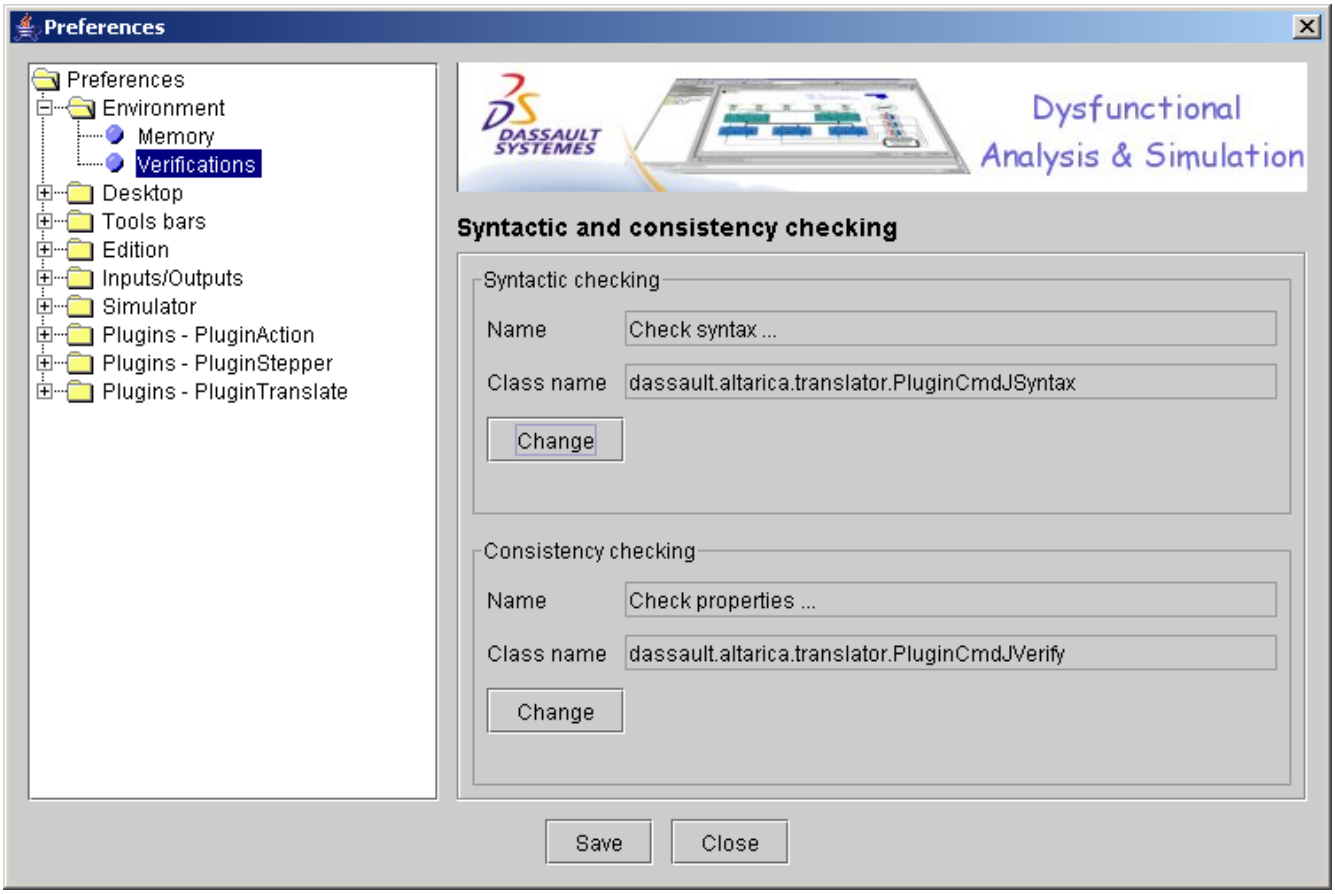


Figure 202 : Syntactic and consistency checking

DESKTOP RUBRIC

The **Screen background** sub-rubric (Figure 203) allows background choice:

- Selection of Background characteristics:
  - *Colour* (of desktop): this colour corresponds to the background colour of the work area if the *Image* field is empty (Red, Green and Blue ratios),
  - *Image*: use the selector to choose the type (gouttes, cuir, crépi, nuages, and bleu). The pathname for a "*.gif" or "*.jpg" file containing another pattern can also be indicated. If the *Image* field is not empty, this selection overrides the *Colour* field selection.
- Tick Mosaïc (for a mosaic display).
- Click on the Apply button.
- Click on the Save and Close buttons.

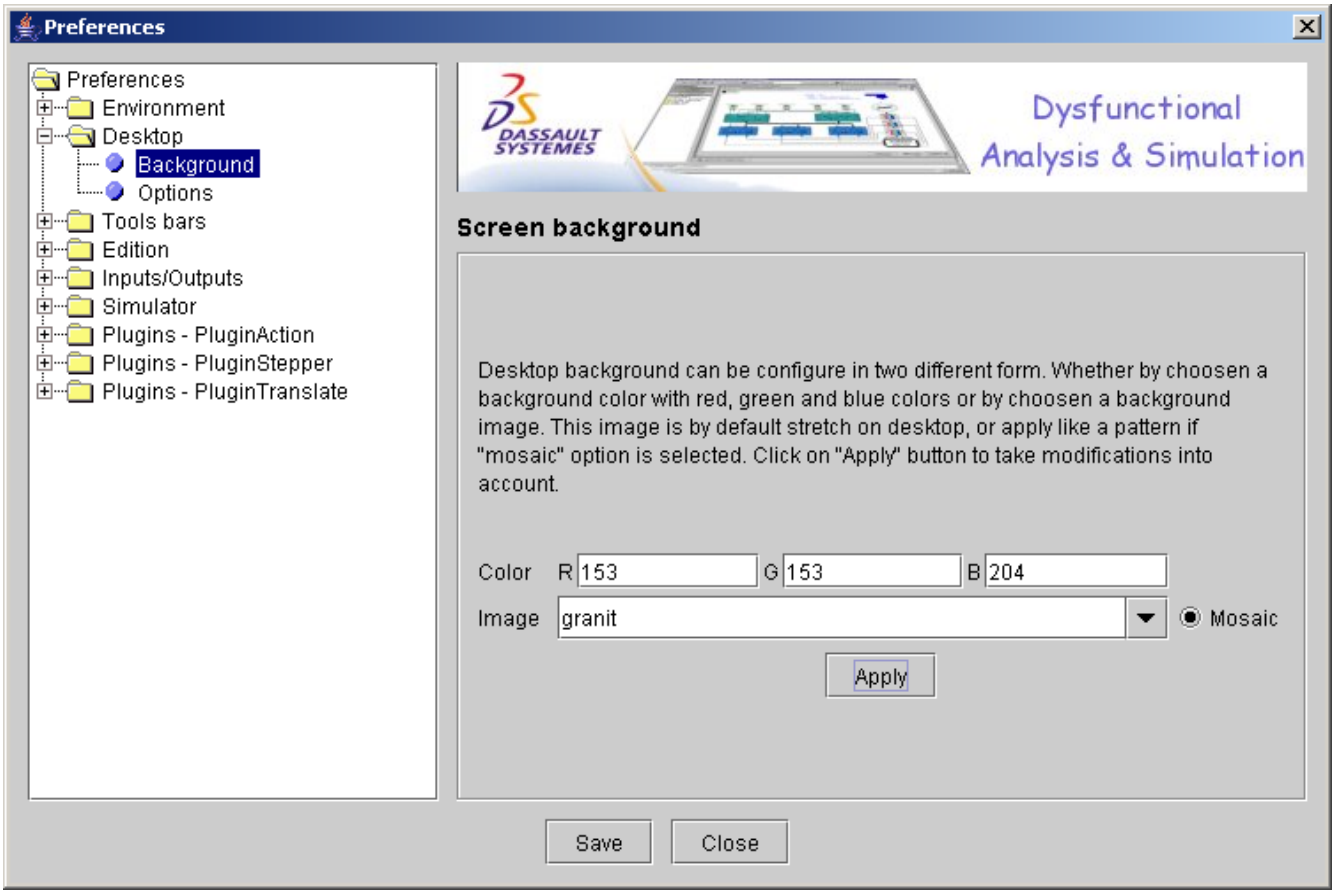


Figure 203 : Screen background

The **Desktop Options** sub-rubric (Figure 204) allows background choice:

- Display time and date,
- Display tools bars,
- Display task bar,

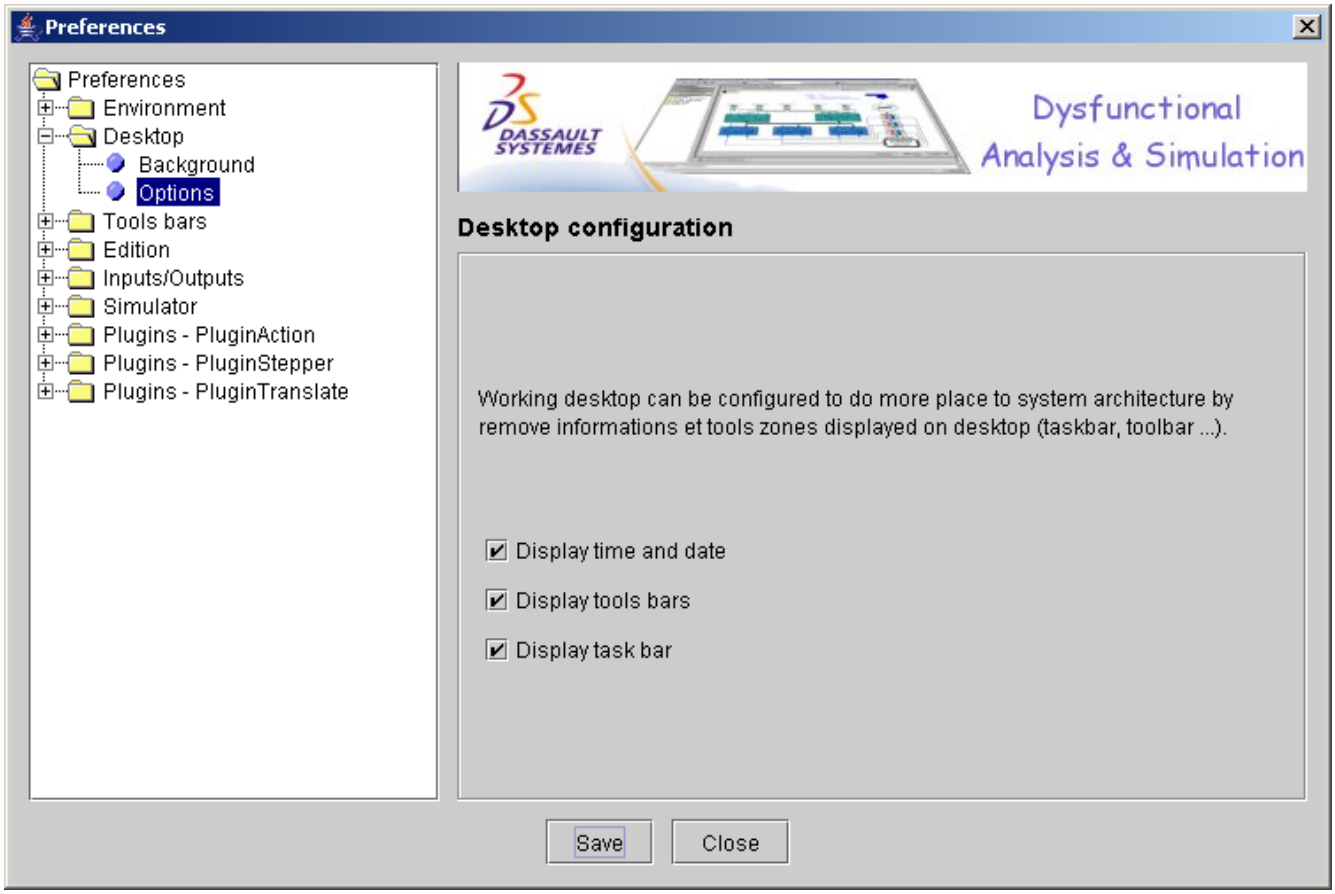


Figure 204 : Desktop options tab

TOOLS BARS RUBRIC

The **Tools bars** tab (Figure 205) allows customization of the tools bars with the following bars:

- Standard,
- Edition,
- Design,
- Links,
- Alignment,
- Navigation,
- Simulation.

Validate by clicking on **Save** and then **Close**.

Remark 2: If the number of tools bars exceeds the window width, buttons located at both ends enable access to the tools bars which are not displayed on the screen.

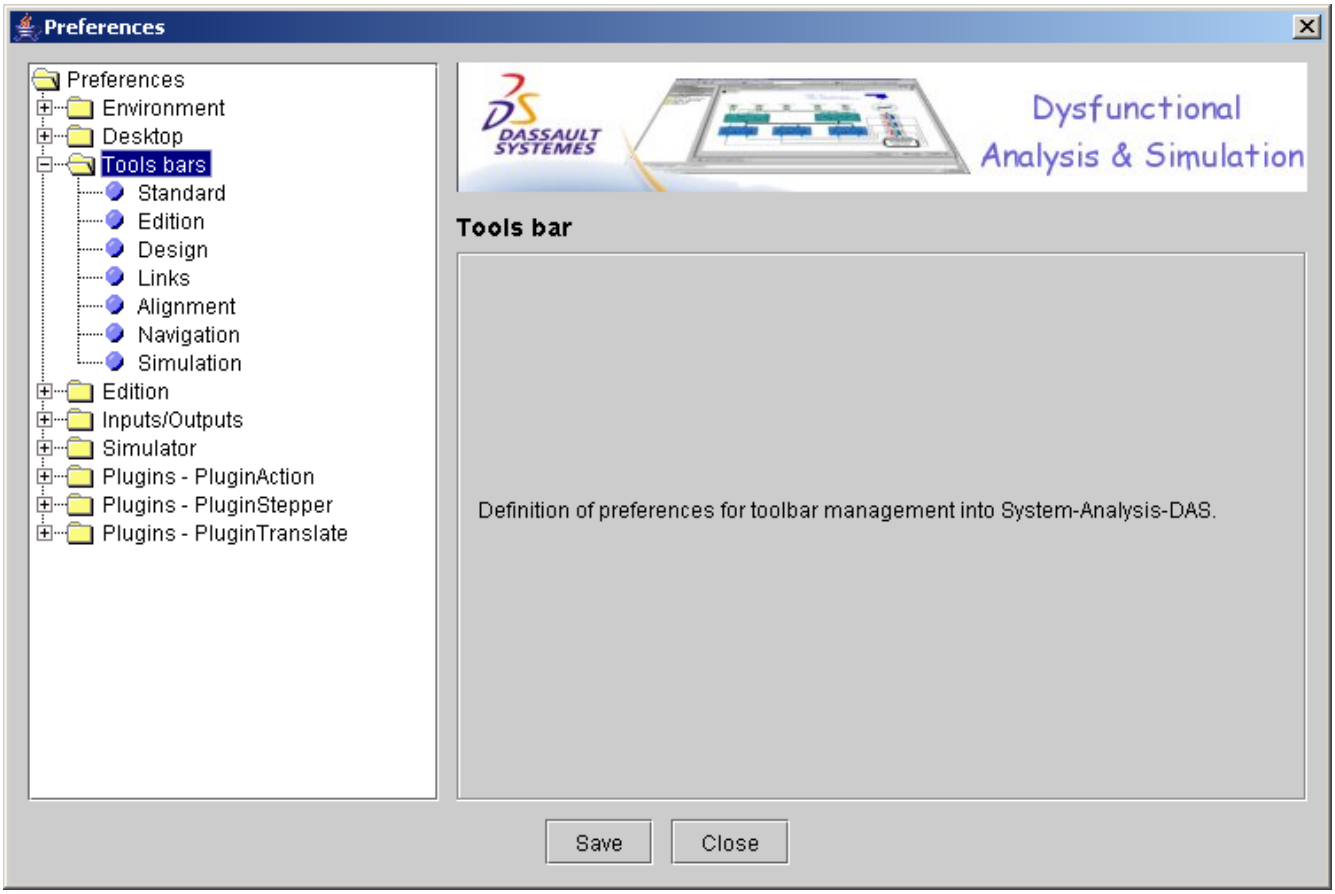


Figure 205 : Tools bars tab

EDITION RUBRIC

Font sub-rubric

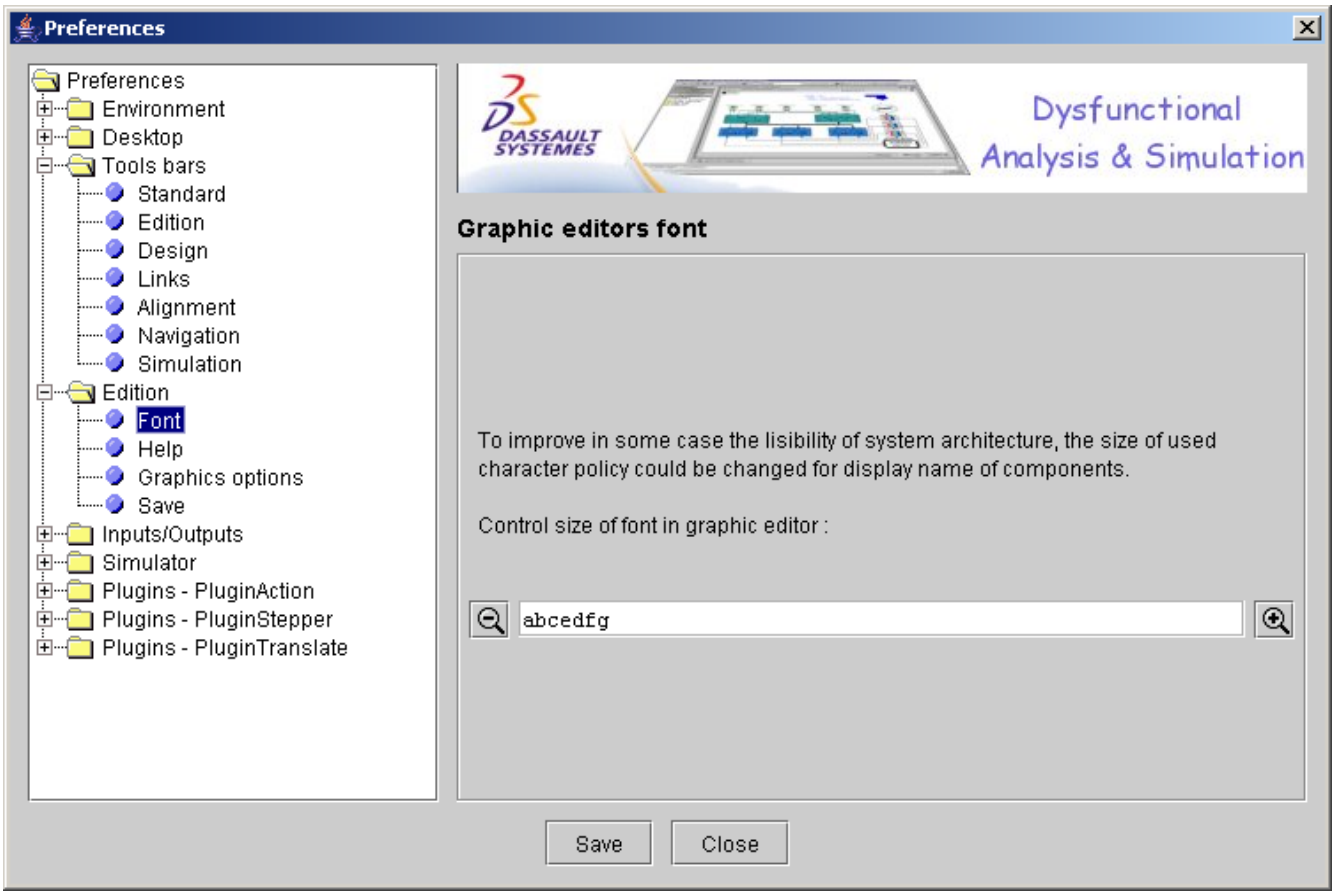


Figure 206 : Graphic editors font

In the Adjustment of font size in architecture view field, type any font size and adjust this size by means of the and icons.

Help sub-rubric

DAS has an edition help for writing Altarica code by given, during data acquisition, the list of component which are present at corresponding level.

Check “activate edition helper” to activate it.

Graphic options sub-rubric

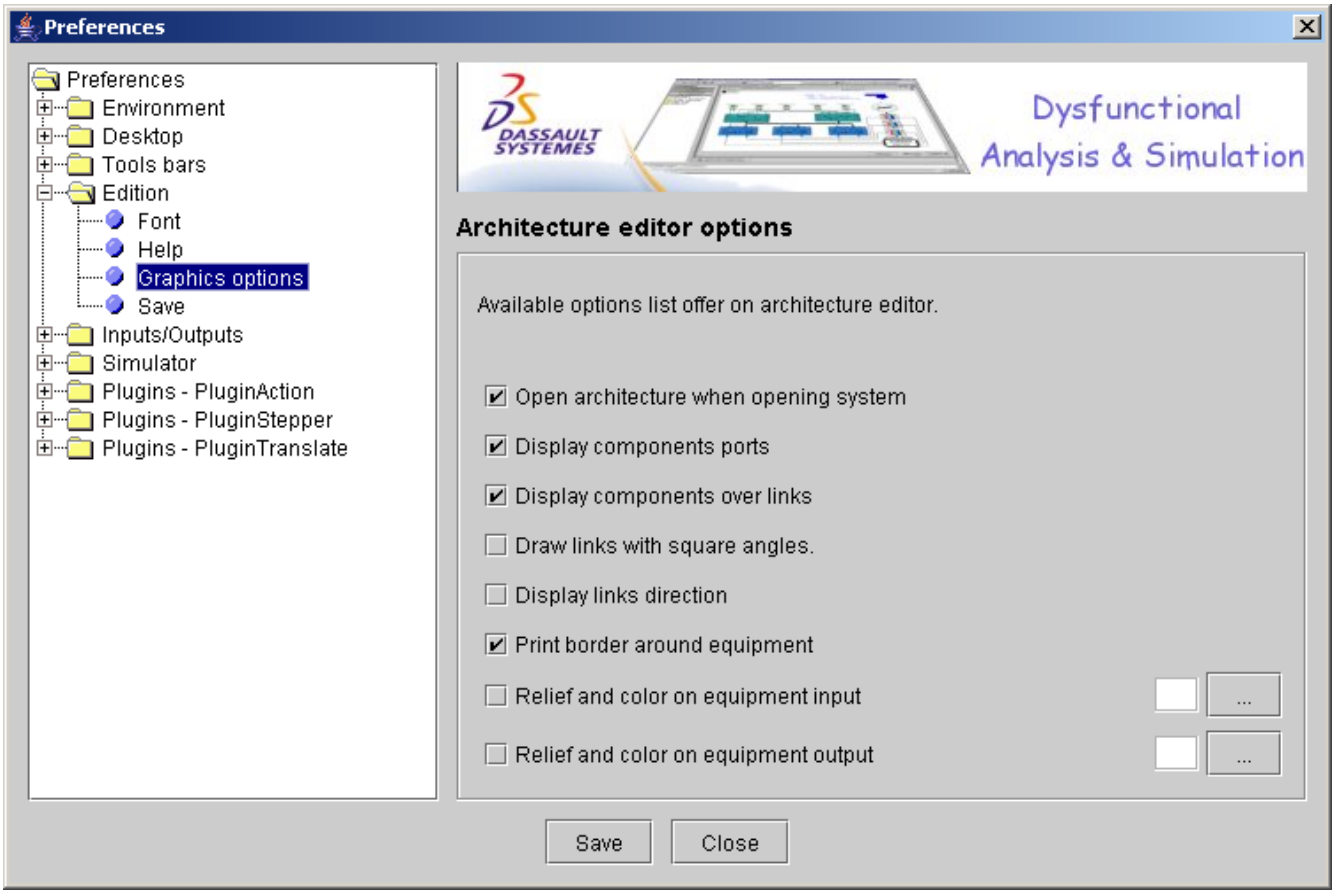


Figure 207: Graphics options sub-rubric

The **Graphics options** sub-rubric (Figure 207) allows customization of the DAS tool with the following options:

- Open architecture when opening project.
- Display components ports
- Display components over links
- Draw links with square angles
- Display links direction
- Print border around equipment.
- Relief and color on equipment input
- Relief and color on equipment output

Save sub-rubric

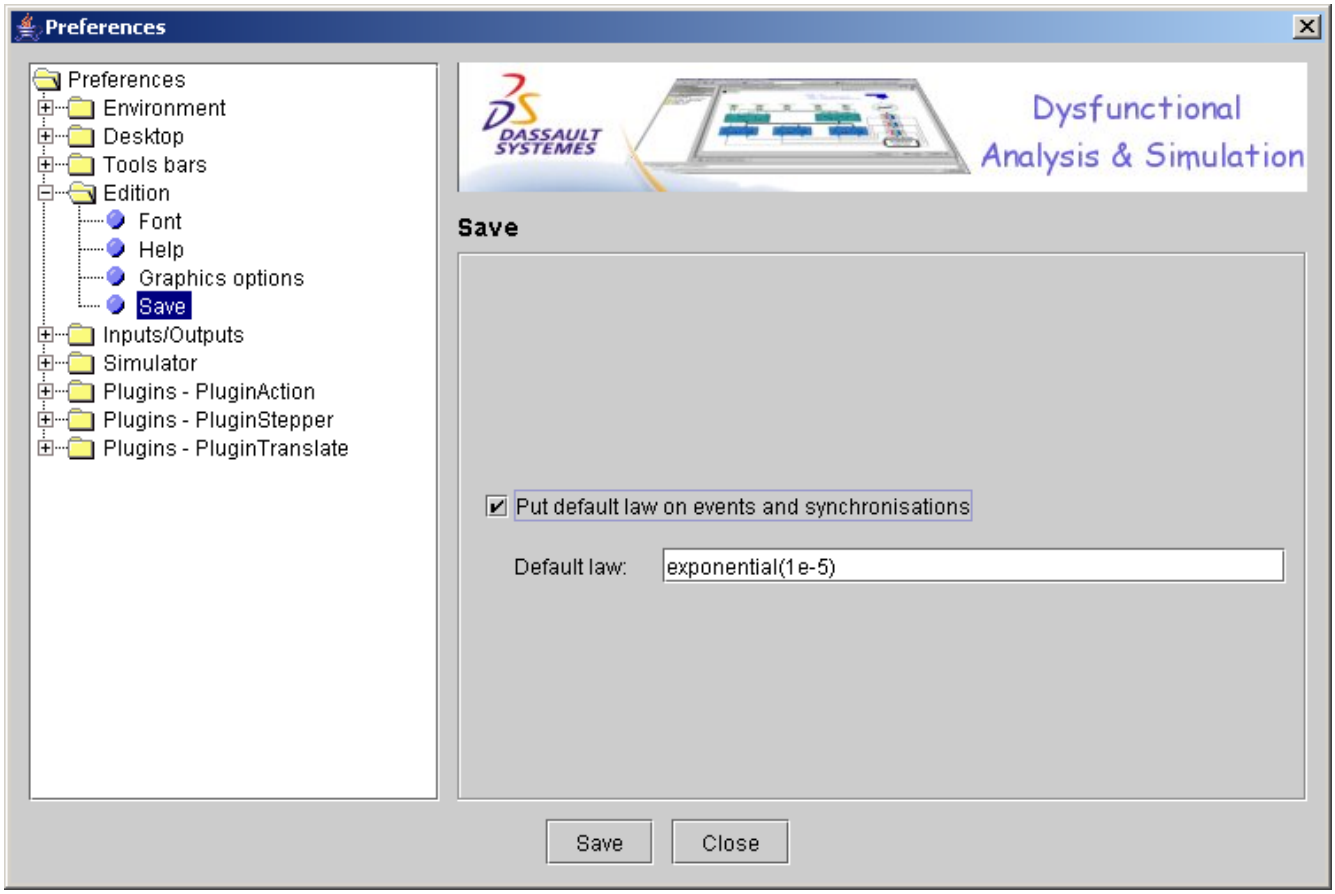


Figure 208 : Save

Check the box “Put default law on events and synchronizations” to save events and synchronizations with a default law. The default law is only put on events and synchronizations that have no law. Use the “default law” area to define this law.

INPUTS/OUPUTS RUBRIC

The Inputs/Outputs rubric contains one sub-rubric: **Printing logo** (Figure 209)

In the File name field (Figure 209), type the pathname to a logo which will be printed in the page top left corner (the file format containing the logo is of the "*.gif" or "*.jpg" type) or use the Select button to indicate the file containing the logo.

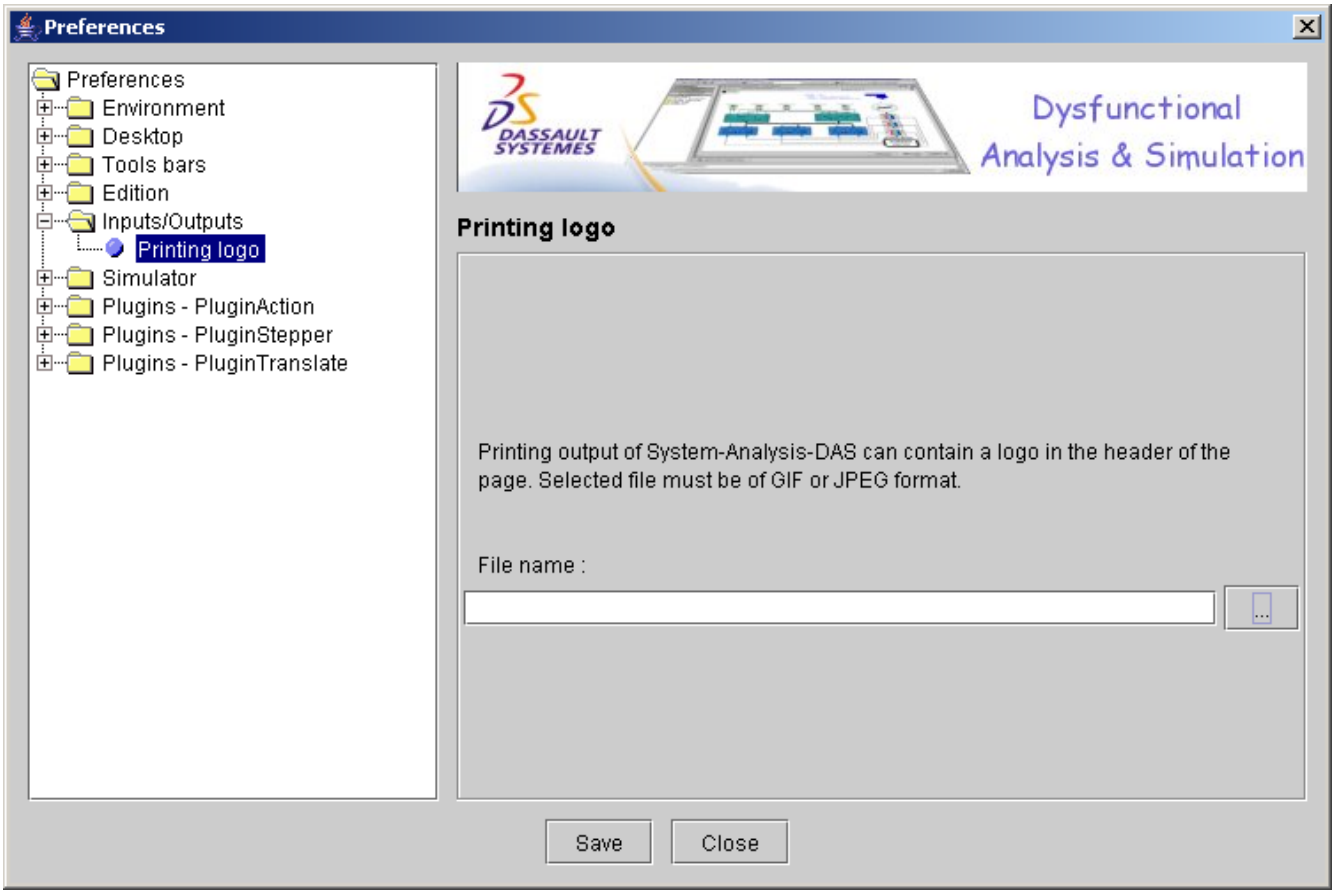


Figure 209 : Printing Logo

Validate by clicking on the **Save** and **Close** buttons.

Remark: The **Save** button must be used only if the modifications are to be saved in the "preferences.dat" file.

SIMULATOR RUBRIC

The Simulator rubric allows choosing simulator options.

Simulator behaviour depends on options that can be activated or not (Figure 210):

- Trigger Automatically event and change state off
- Number of automatic execution: to avoid endless simulation due to automatic triggering of instantaneous transition, user must specify the maximum number of automatic.

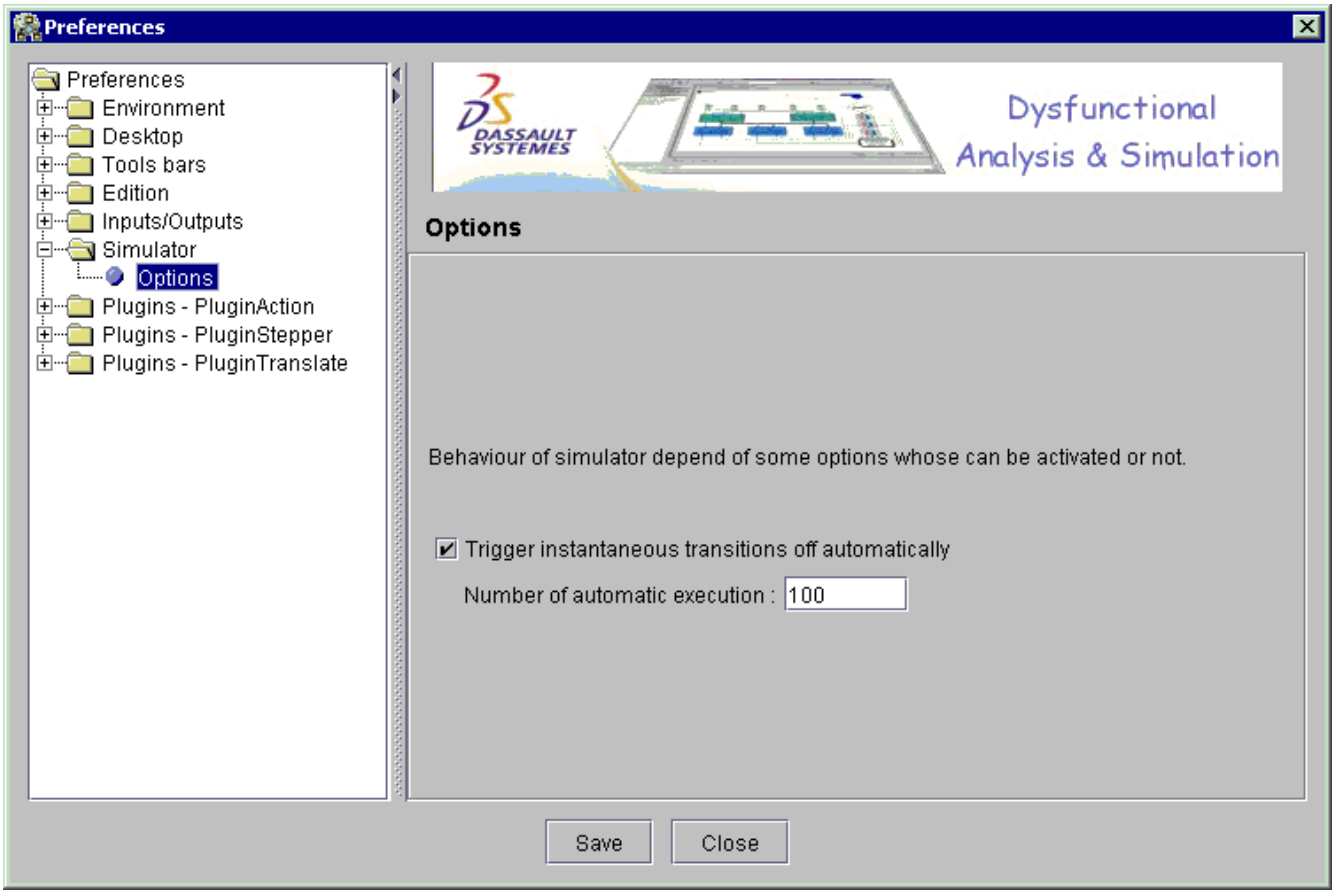


Figure 210 : Simulator Options

PLUGIN OPTIONS

The **Plugins – PluginAction** and **Plugins - PluginTranslate** rubrics (Figure 211) allows to choose options concerning the plugins.

For detailed explanations about these options, please see the plugin document annexes.

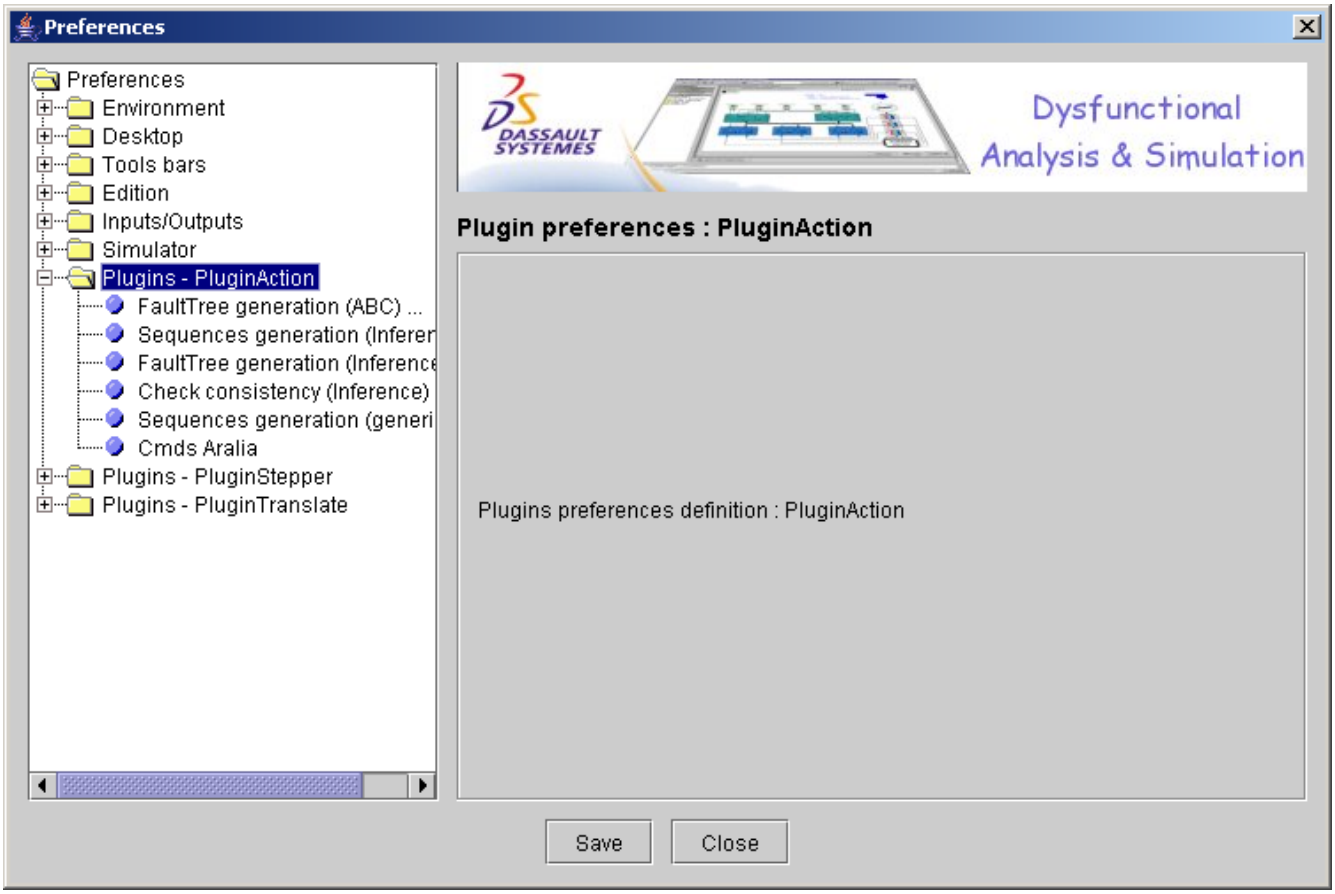


Figure 211 : Plugin option rubrics

QUIT DAS

To quit DAS, proceed as follows:

- Select **Quit** in menu **File**.
- Use shortcut **CTRL+Q** on keyboard.

If any system has been modified, the following window **Quit** (Figure 212) is displayed.

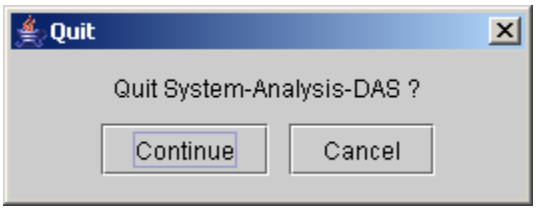


Figure 212 : Quit DAS

- Click on **Continue**, or on **Cancel** button.
- The **MSDOS JAVA** window is closed.
- If one (or several) model(s) has (have) been modified, the following window is displayed (Figure 213):

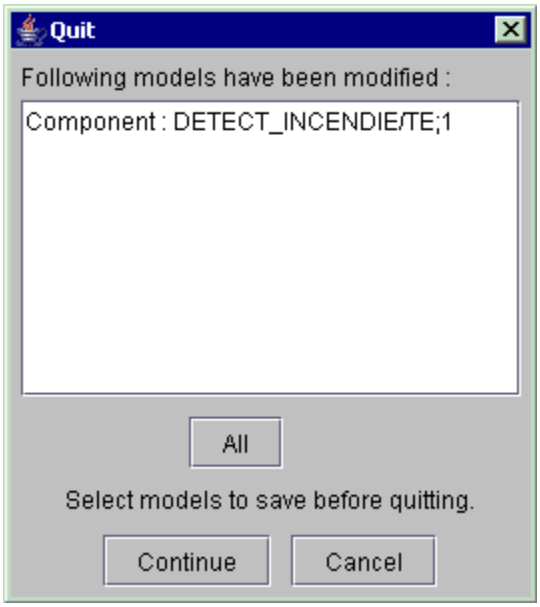


Figure 213 : Saving of modified model (s)

- Click on button **All**; all modified systems in the list appear in grey colour or select manually system(s) to be saved.
- Click on **Continue**, or on **Cancel** button to remain in DAS environment.
- **MSDOS JAVA** window is closed.

GLOSSARY

Assertion (**assert**)	Altarica code allowing variation of the output flow variables of components
Base	It is made up of three directories (icons, models and project) and contains all the systems, equipment, components, operators and types of the DAS tool library. **N.B**: There are systems which do not belong to the library but which are built up with the use of the library.
Boolean constant	It is one of the two following keywords: true or false.
Component	It is an elementary behavioural object which can be represented graphically. It is a component type model.
Enumerate constant	It is an identifier previously declared in an enumeration field.
Enumerate type	It consists of a single variable which can take several values, e.g. power supply: {null, nominal}
Equipment	It is a hierarchical behavioural object which can be represented graphically. It is an equipment type model.
Event	It is an external action which conditions a component state variable.
Family	It enables classification of models in the library.
Identifier	String of characters containing letters (uppercase or lowercase), numbers or underscore character (_). The string must start with a letter.
Integer constant	It is either zero or a string of numbers which do not start with zero.
Keyword	It is an identifier reserved for exclusive use in the Altarica language, e.g. if, then, etc.
Lexems	They are identifiers, keywords, integer constants, operators and separators in the Altarica code.
Link	It defines a link between two components or equipment in an architecture.
Model	It is a generic term which describes an object in the library.

Object	It is a generic term which denotes a component, an equipment, an operator or a type.
Operator	It is a generic elementary behavioural object without graphical representation which can be used to modelize a component.
Project	It is a generic term representing a set of systems.
System	It is a component of a project which is represented by an architecture.
System assertions	System assertions are links which are not displayed in the hardware architecture; they represent only flow assignments and do not use operators (e.g., cst1.e = true).
Transition (**trans**)	Altarica code which enables evolution of the state variables of components.
State variable	It is a variable identifying a component internal state (e.g. a variable with the following values “open/blocked”, “ok/hs”…).
Structured Type	It is constituted of many variables, and each variable can have many values, for example voltage :{ null, nominal} and short-cut :{ yes, no}
Flow variable	It is a component input or output variable.

APPENDIXES

APPENDIX 1: SYNTAX OF Altarica

General

Typing of the behavior of component models comprises:

- The lexical conventions.
- The assertions.
- The transitions.
- The comments.
- The identifiers.
- The keywords.
- The constants.
- Assertion examples.
- Transition examples.

The lexical conventions

The Altarica language lexems are the identifiers, keywords, integer constants, operators and separators.

Spaces, tabulations, end of line characters as well as comments are ignored except when they separate lexems.

The transitions

Transitions allow evolution of the state variables of components.

Transition model:

transition condition | – event –> transition assignments

The declaration of a model transitions starts with the **trans** keyword. This keyword is followed by a non-empty list of transition specifications separated by semi-colons. The transitions declaration may eventually end by a semi-colon.

transitions-definitions
trans transition-list

transition-list
transition ; transition-list

A transition specification always starts with a boolean expression called the transition *condition*. The variables which appear in these expressions are the model flow variables or state variables.

transition
boolean-expression transition-target-list

boolean-expression
expression

The transition condition is then followed by a non-empty list of actions associated to events.

transition-target-list
transition-target; transition-target-list
transition-target

An action starts by the separator | – followed by a non-empty list of events previously declared in the model.

The list ends with the –> separator followed by an assignment list for the variables.

transition-target
| – event-name-list –> assignment-list

Assignments of variables are separated by commas (,).

assignment-list
assignment , assignments-list

An assignment is composed of the identifier of the model state variable followed by the := separator and ends with an expression of the same type as the state variable. The expression assigned to the state variable can contain the model flow variables or state variables

assignment
variable-name := expression

variable-name
IDENTIFIER

Transition example

The assertions

Assertions allow modification of the output state variables of components.

The assertions declaration for a component start with the **assert** keyword. This keyword is followed by a non-empty list of boolean expressions separated by semi-colons (;). The declaration may eventually end with one semi-colon.

assertions-definition
assert assertion-list

assertion-list
assertion ; assertion-list

assertion
boolean-expression

The variables which appear in an assertion are the flow variable or state variables of the model in which the assertion is defined.

The boolean expressions authorized at the first level in an assertion are the following:

- The parenthezed expressions such as if then else
- The assignments of the variables:

An assignment is composed of the identifier of an output flow variable (out) for the model followed by the separator = and ends with an expression of the same type as the flow variable. The expression assigned to the flow variable can contain constants and input flow variables (in) or state variables for the model.

Assertion examples

Expression

This section presents the syntax of Altarica expression which may appear in the assertions or the transitions condition.

The expressions below are presented in the decreasing priority order (The logic OR is the lowest order priority).

- The parenthezed expressions.
- The variables.
- The atomic expressions.
- The unary operators.
- The relational operators.
- The equality operators.
- The logic AND operator.
- The logic OR operator.

The parentheses expressions

In order to facilitate interpretation of the expressions If-Then-Else and If-Then, the latter must be mandatory between parentheses.

parenthezed-expression
(expression *)*
(expression *?* expression *:* expression *)*
(if expression *then* expression *else* expression *)*
(if expression *then* expression *)*

For an If-Then-Else expression, the first sub-expression is mandatory boolean; it is called the operation condition. The two next sub-expressions make up the parts Then and Else of If-Then-Else.

Remark 1*:* The expressions authorized in the parts Then and Else are the same as those

authorized at the first level of an assertion.

Remark 2: (if *expression* then *expression* else *expression*) and (*expression* ? *expression* : *expression*) are equivalent.

The variables

The variables of a description can be specified by a path in the hierarchy of Altarica models. This path is a string of identifiers separated by points (.).

hierarchy-path
identifier . *hierarchy-path*
identifier

The atomic expressions

Atomic expressions are either pathnames to variables or constants or parenthezed expressions.

Depending on the context, a path can be interpreted as a variable identifier or an enumerate constant identifier or also a declared constant identifier. In these conditions, a same identifier cannot be used for a declared constant, a variable or an enumerate constant.

atomic-expression
parenthezed-expression
hierarchy-path
true
false

The unary operators

The unary operations are evaluated from right to left.

unary-expression
operator-not unary-expression
atomic-expression

operator-not
not
~

The NOT operator has two identifiers, the not word or the tilde ~. The NOT operand must be of the boolean type and the result of the expression is the inverted value of this operand.

The relational operators

The relational operators are evaluated from left to right. Their operands must be of an arithmetic type because, unlike some programming languages, neither the enumerate constants nor the boolean variables, are assimilated to integers.

The operators < (less than), > (greater than), <= (less or equal to) and >= (greater or equal to) give the value false if the given relation is not satisfied by the operands; otherwise, the value of the expression is true.

relational-expression
relational-expression < additive-expression
relational-expression > additive-expression
relational-expression <= additive-expression
relational-expression >= additive-expression
additive-expression

The equality operators

These operators are = (equal to) and != (not equal to).

The operands of the = and != operators are boolean, arithmetic or enumerate. The value of these expressions is true if the expressed relation is satisfied by the values of the operands and false in the contrary. In the boolean case, the equality corresponds to the equivalence between the operands and the difference with the Exclusive or (XOR).

equality-expression
relational-expression = relational-expression
relational-expression != relational-expression
relational-expression

The logic AND operator

The logic AND operator has two identifiers: *&* and *and.* Its operands must be of the boolean type. It is evaluated from left to right. It is true if its two operands are true and otherwise it is false.

and-expression
and-expression operator-and equality_expression
equality-expression

operator-and
and
&

The logic OR operator

The logic OR operator has two identifiers: | or *or.* Its operands must be of the boolean type. It is evaluated from left to right. It is false if its two operands are false and otherwise it is true.

or-expression
or-expression operator-or and-expression
and-expression

operator-or
or
|

The comments

Two types of comments are authorized:

The /* characters indicate the start of a comment on several lines and which must end by the */ characters. The comments must not be overlap.

Example:
```
/*
* This is a comment on several lines.
*/
```

The // characters indicate the start of a comment on one line. The comment ends either by carriage return character or by the end of a data flow.

Example:
```
// This is a comment on one line.
// This is another comment.
```

The identifiers

An *identifier* is a sequence of letters, of numbers or of '_'. A single identifier always starts with a letter and can be of any length. Uppercase and lowercase letters are different.
Example : Motor, Motor_9, motor_9 (the last two are different identifiers).
Identifiers must belong to the language defined by the following regular expression:
[a-zA-Z]([a-zA-Z0-9_]*)

The keywords

The following identifiers are reserved for use as keywords and cannot be used for other purposes:

and	else	imply	max	sub
assert	event	in	min	sync
bool	extern	init	node	term
case	false	int	not	then
cnuf	float	inverse	or	trans
const	flow	knil	out	true
domain	func	link	private	#
edon	if	local	state	

The constants

There are several types of constants, each one having a particular type: integer constants, enumerated constants and boolean constants.

An *integer constant* is:
Either 0,
Or a sequence of numbers not starting with the 0 character.
Integer constants must belong to the language defined by the following regular expression:
0 | ([1-9][0-9]*)
A *boolean constant* is one of the two following keywords: true or false.
An *enumerated constant* is an identifier previously defined in a field of enumerations.

Assertion examples

Example 1

```
assert
(if etat=ok and
(true = (e_v1 and e_v2) or (e_v1 and e_v3) or (e_v1 and e_v4) or (e_v1 and e_v5) or
(e_v2 and e_v3) or (e_v2 and e_v4) or (e_v2 and e_v5) or
(e_v3 and e_v4) or (e_v3 and e_v5) or (e_v4 and e_v5))
then s = true , validite = true);
```

Example 2

```
assert
(if etat = non_repartie or etat = repartie then
(if e1 = normal or e2 = normal then
s = normal
else
(if (e1 = intempestif) or (e2 = intempestif) then
s = intempestif
else
s = absent
)
)
);
```

Example 3

```
assert
(if (panne_moteur_droit and (~auto_drapeau_droit or ~suralimenter_gauche)) or
(panne_moteur_gauche and (~auto_drapeau_gauche or ~suralimenter_droit))
then
s = true , icon = 2
else
s = false , icon = 1);
(if etat =hs then s = false , validite = false);
(if etat=ok then icon=1 else icon=2);
```

Example 4

```
s1 = e1 or e2 ;
s2 = e3 ;
```

Transition examples

Example 1

```
trans
etat=ok |- panne -> etat := hs;
```

Example 2

```
trans
etat = non_repartie |- def_local_panne -> etat := hs;
etat = non_repartie|- def_local_intempestif -> etat := instable;
```

Example 3

```
trans
etat = ouvert and blocage = non |- fermeture_intemp -> etat := ferme , blocage := oui;
etat = ferme and blocage = non |- ouverture_intemp -> etat := ouvert , blocage := oui;
```

PAGE LEFT BLANK INTENTIONALLY

APPENDIX 2: PROBABILITY LAWS IN ARALIA FORMAT

Probability laws for Basic Events				
Law	**Expression**	**Par.**	**Definition**	**Limits**
constant	$A(t) = q$	q	Failure probability	$0 \leq$ Probability ≤ 1
exponential	$A(t) = 1\text{-}\exp^{-\lambda.t}$	λ	Failure rate	Rate ≥ 0
GLM	$A(t) = \gamma \exp^{-(\lambda+\mu).t} + \lambda/(\lambda+\mu) \cdot \exp^{-(\lambda+\mu).t}$	γ	Probability of initial start-up refusal	$0 \leq$ Probability ≤ 1
		λ	Failure rate	Rate ≥ 0
		μ	Repair rate	Rate ≥ 0
GLM-asymptotic	$A(t) = \lambda/(\lambda+\mu)$	λ	Failure rate	Rate ≥ 0
		μ	Repair rate	Rate ≥ 0
Weibull	$A(t) = 1\text{-}\exp^{-\phi(t)}$ avec $\phi(t) = ((t\text{-}\tau)/\alpha)^{\beta}$	α	Scale parameter	Factor> 0
		β	Form parameter	Factor> 0
		$\tau 0$	Location parameter (α<=0 as t-α must always be greater than 0)	Time> 0
periodic-test	$A(t) = \begin{cases} 1-\exp^{-\lambda.t} & \text{si } t \leq \tau \\ 1-\exp^{-\lambda.((t-\tau)\%\theta)} & \text{si } t > \tau \end{cases}$	λ	Failure rate of the component in operation or on stand-by	Rate≥ 0
		τ		Time > 0
		θ	First interval between tests	Time> 0
periodic-test-2	*Numeric integration*	λ	Functioning failure rate	rate ≥ 0
		$\lambda*$	Failure rate during test	rate ≥ 0
		μ	Repair rate (once failure is detected)	rate ≥ 0
		τ	First interval between tests	Time ≥ 0
		θ	Interval between two tests	Time ≥ 0
		γ	Failure probability because of test (0 if test can't create failure)	$0 \leq$ Proba ≤ 1
		π	Test duration	Time ≥ 0
		X	Availability during test (= 0, if unavailable; = 1, if available)	0 or 1
		σ	Test coverage rate (the probability that test detect failure)	$0 \leq$ Proba ≤ 1
		ω	reSetUp forgetfulness probability (after test and after repair)	$0 \leq$ Proba ≤ 1
dormant		λ	Failure rate	rate ≥ 0
		MTTR	Medium time to repair	Time ≥ 0
		T	interval between two consecutive tests	Time ≥ 0

Probability laws for Basic Events				
Law	**Expression**	**Par.**	**Definition**	**Limits**
CMT	$A(t) = Q + 1 - \exp^{-\lambda.T}$	λ T Q	Failure rate Mission duration Probability	rate ≥ 0 Time ≥ 0 $0 \leq$ Proba ≤ 1
NRD	$A(t) = \exp^{-\mu.d}$	μ d	Repair rate Delay	rate ≥ 0 Time ≥ 0