

# **TASTE Editors 2.0**

## **User Manual**

*reference:*

ESA/ESTEC frame contract n°4000104809 – 29 November 2011

Call-Off Order 005 – 02 December 2014

*issue:*

release 1.0 – 31 December 2015

*Authors:* Pierre Dissaux (Ellidiss Technologies)

EUROPEAN SPACE AGENCY

CONTRACTUAL REPORT

The work described in this report was done under ESA contract

Responsibility for the contents resides in the author or organisation that prepared it

## Table of Contents

|      |                                                  |    |
|------|--------------------------------------------------|----|
| 1.   | Introduction.....                                | 3  |
| 2.   | Installation.....                                | 4  |
| 3.   | Overview of the TASTE graphical editor .....     | 5  |
| 4.   | About TASTE AADL models .....                    | 6  |
| 5.   | Quick start tutorial .....                       | 7  |
| 5.1. | Load a DataView.....                             | 7  |
| 5.2. | Build an Interface View .....                    | 8  |
| 5.3. | Build a Deployment View.....                     | 11 |
| 5.4. | Build and analyse the Concurrency View .....     | 13 |
| 5.5. | Display and verify the generated AADL code ..... | 15 |

## 1. Introduction

The TASTE 2.0 editor is a new graphical editor that replaces the three pre-existing ones in the TASTE tool-chain. The concerned tools are the Interface View editor (IV), the Deployment View editor (DV) and the Concurrency View editor (CV).

The new tool provides the same functionalities as the three previous ones while providing new improvements for the TASTE modelling activities:

- Full integration of the IV-DV activities within a single tool
- Enhanced support for reuse of IV models and components
- Updated real-time analysis tools for the CV, including the most recent version of the Cheddar scheduling analysis framework and of the Marzhin AADL simulator.

The TASTE modelling workflow is thus significantly simplified as shown in Figure 1.

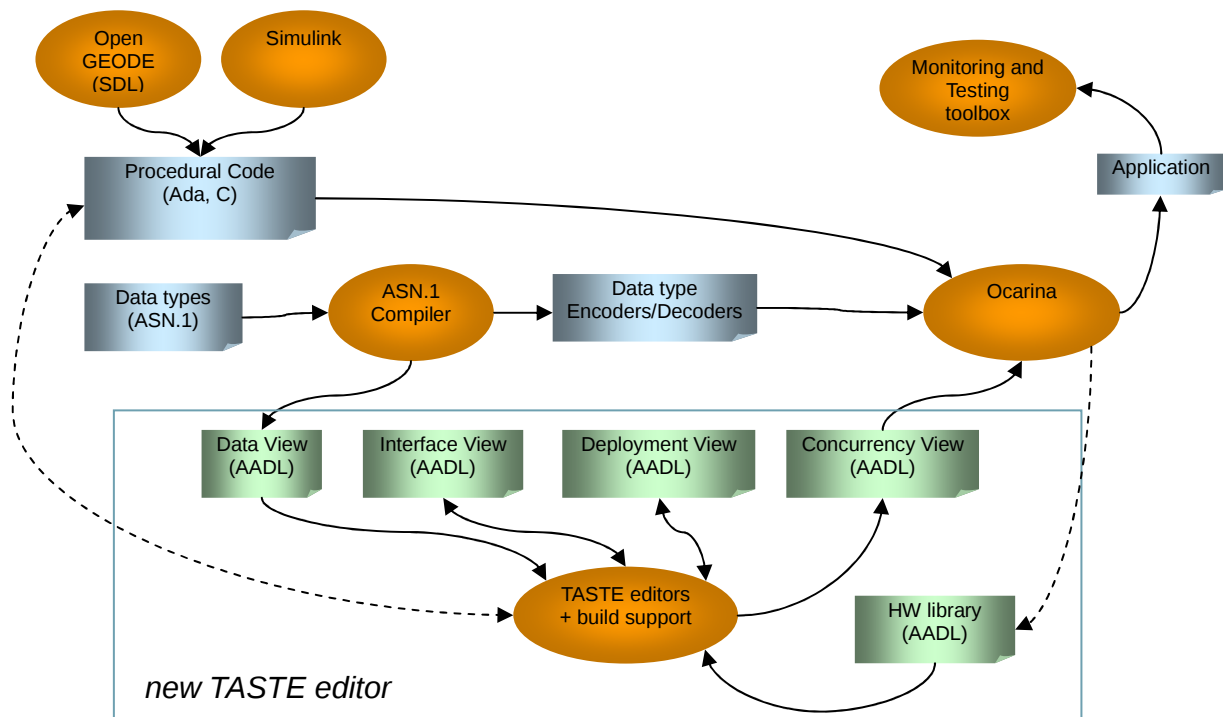


Figure 1: The TASTE workflow

The TASTE editor uses various background technologies:

- GMP (Graphic Model Processing): generic graphical framework developed by Ellidiss Technologies.
- LMP (Logic Model Processing): generic model transformation and AADL toolbox developed by Ellidiss Technologies.
- Cheddar (<http://beru.univ-brest.fr/~singhoff/cheddar>): real-time scheduling analysis tool developed by the University of Brest and Ellidiss Technologies.
- Marzhin : AADL run-time simulator developed by Virtualys and Ellidiss Technologies.
- Ocarina (<http://www.openaadl.org/ocarina.html>): AADL compiler and code generator developed by ISAE.

## 2. Installation

The new TASTE editor is embedded into the standard TASTE tool-chain distribution. The current TASTE tool-chain distribution is available at the following address:

<http://download.tuxfamily.org/taste/taste-vm.tar.gz>

It consists of a LinuxVMWare image that can be run with a VMWare player. There is no specific installation procedure regarding the TASTE editor.

It is also possible to run the TASTE editor in a standalone mode on a Windows or Linux platform, in which case some features of the TASTE- toolchain will not be available, such as the code generation. Standalone distributions are available at the following address:

<http://www.ellidiss.com/downloads.asp>

For Linux platforms, unzip and untar the downloaded archive into a local workspace and run the TASTE executable file located in the `bin` directory. For Windows platforms, unzip the downloaded archive into a local workspace and run the executable file: `TASTE.exe` that is also located in the `bin` directory.

Various customizations are available for the TASTE editors. In particular, the `IVConfig.ini`, `DVConfig.ini`, `AIConfig.ini` and `TasteConfig.ini` files within the `config` directory may be used to apply a few user configurations.

Please note that the use of the TASTE editors is subject to the General Clauses and Conditions (GCC) applying to contracts placed by the European Space Agency (ESA). Dedicated distribution and end-user licences are also applicable for the TASTE editor and the use of the corresponding background technologies. Refer to the `License.pdf` file located in the `doc` directory of the TASTE editor distribution.

For any technical or commercial question related to the TASTE editors that are described in this manual, please contact the Ellidiss support team:

<mailto:taste@ellidiss.fr>

### 3. Overview of the TASTE graphical editor

The new TASTE graphical editor now consists of a single window that encompasses:

- A main menu and button bar. The button bar is updated upon the current modelling or verification activity, as defined by the selection tab.
- A models browser where the overall project hierarchy and organization is displayed in a deployable tree structure. This browser may be hidden from the View main menu.
- A set of selection tabs to enable one of the proposed TASTE modelling or verification activity, and update the content of the working area.
- A working area, showing either:
  - o A textual editor where the ASN.1 and ACN representations of a TASTE DataView model can be edited, or
  - o A box-arrow editor where a graphical representation of a TASTE Interface View or Deployment View model can be edited, or
  - o A read-only area showing the results of the timing analysis of a TASTE Concurrency View model, or
  - o A read-only textual editor that displays the generated textual AADL code
- A log view where main editing operations are logged. This log view is hidden by default and can be made visible from the View main menu.
- A status bar showing system information, warning and error messages

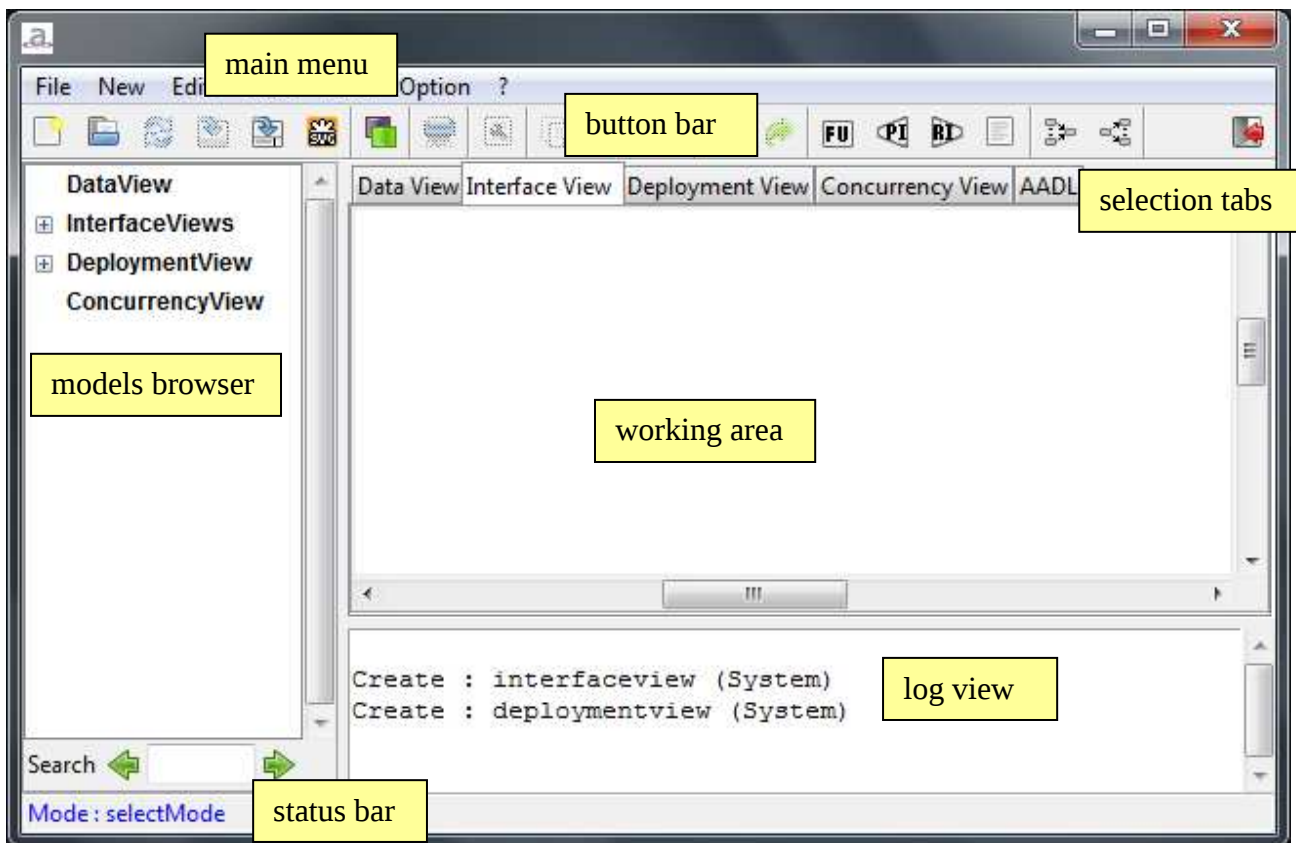


Figure 2: The combined TASTE graphical editor

## 4. About TASTE AADL models

The TASTE models that can be loaded into and saved from the TASTE editor are serialised in AADL format. The AADL language is a SAE international standard that is used to model and analyse the architecture of safety critical real time systems. More information about AADL can be found at:

<http://www.openaadl.org/>

For the purpose of TASTE models serialisation, only a subset of the AADL language is used, and the TASTE entities are associated with specific AADL constructs that have been defined to ensure a correct semantic mapping. A TASTE project makes use of several kinds of models. Each of them is associated with a dedicated AADL subset. These models are:

- **DataView:** automatically generated from ASN.1 source code.
- **Interface View:** automatically generated from the Interface View diagram.
- **Hardware Library:** provided by the Ocarina environment.
- **Deployment View:** automatically generated from the Deployment View diagram.
- **Concurrency View:** automatically generated by the TASTE Build Support utility.
- **Customised Properties:** specified by the `TASTE_IV_Properties` and `TASTE_DV_Properties` files located in the `config` directory of the TASTE editor distribution.

All the AADL files that are involved in a TASTE project are either provided or automatically generated. It is thus not necessary – and even not recommended - for a TASTE end-user to edit the AADL models. Directly editing the provided or generated AADL files may lead to load errors and model corruption.

Within the TASTE editor, the AADL files can be loaded either in a generic way from the File main menu (Load), or for a particular kind of model from the contextual menus of the models browser (Load DataView, Load IV, Load DV, Load CV). The generic load process only works for AADL files that have been produced by version 2.0 or greater of the TASTE editor. Older files can be made compatible by adding a comment line at the beginning of the AADL file to specify which kind of TASTE model is concerned:

```
-- type dataview
-- type interfaceview
-- type deploymentview
-- type concurrencyview
-- type hwlibrary
```

Note that only one model of each kind can be loaded at a time. A new load action will replace the previously loaded model. When it is required to have access to several models of the same kind, the import menu must be used instead. This is in particular the case for reusing components from another Interface View model.

Also note that loading an IV model automatically loads the referenced DataView. Similarly, loading a DV model automatically loads the referenced HW Library and IV, as well as the DataView referenced by this IV. When a new DataView is explicitly or implicitly loaded, it is merged with the already loaded ones. This action may raise an error when the ASN.1 toolbox is not available (i.e. outside the TASTE VM), but does not prevent the DataView to be used for modelling activities.

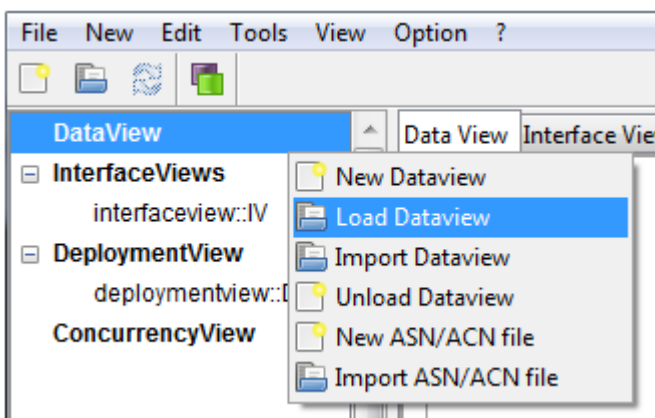
## 5. Quick start tutorial

This section gives an example of use of the main modelling and verification features of the TASTE editor. It does not address the target code generation activity that requires the use of other components that are made available in the TASTE VM distribution.

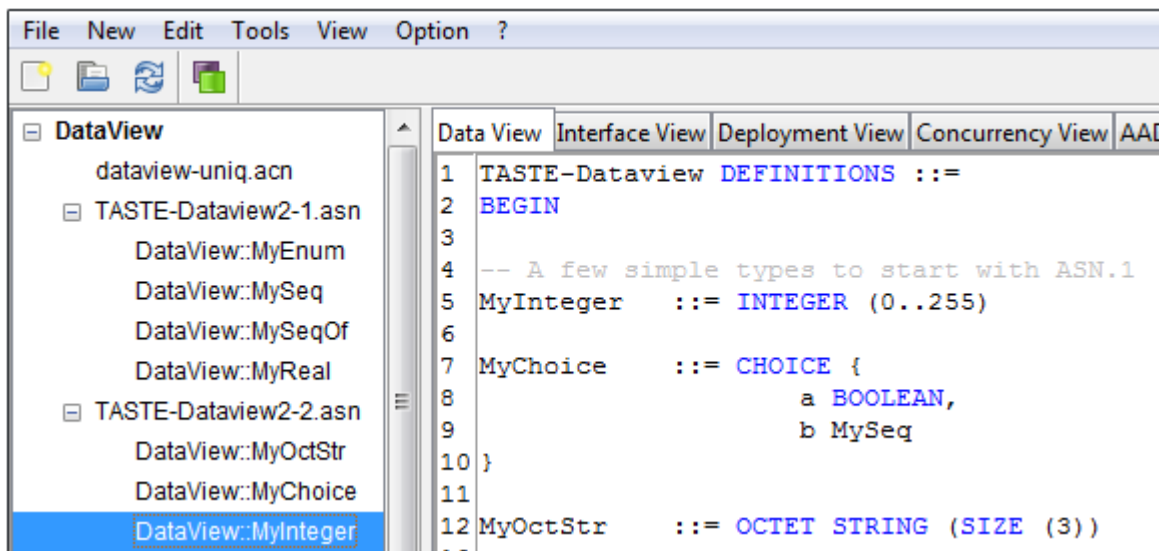
After the TASTE editor has been launched in standalone mode, it looks like the Figure 2 above (except that the Log View is not shown) where default empty Interface View and Deployment View models are initiated and the Interface View tab is selected. It is thus possible to directly start the creation of the IV model, however it is recommended to load a DataView first to be able to use data types while editing Interface and Context Parameters.

### 5.1. Load a DataView

Select the *Data View* tab and the *DataView* heading in the models browser, then the *Load Dataview* contextual menu:



Choose an AADL file containing a TASTE DataView model. An example is provided in the Workspace directory of the distribution.

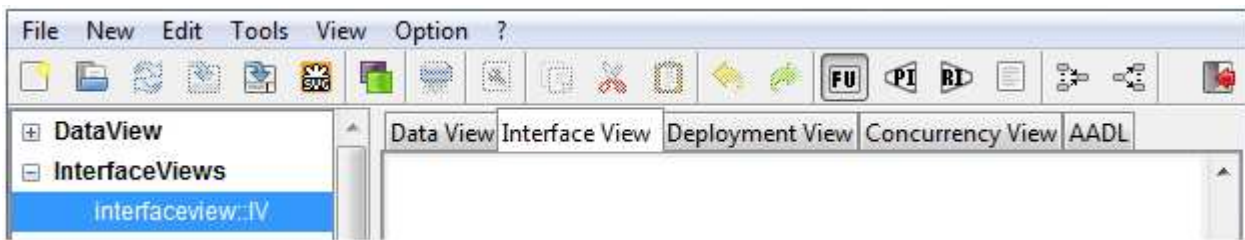


After the file has been loaded, the Browser shows the various ACN and ASN source entries as well as the list of available data types. Double clicking on one of these items in the browser displays the corresponding source contents.

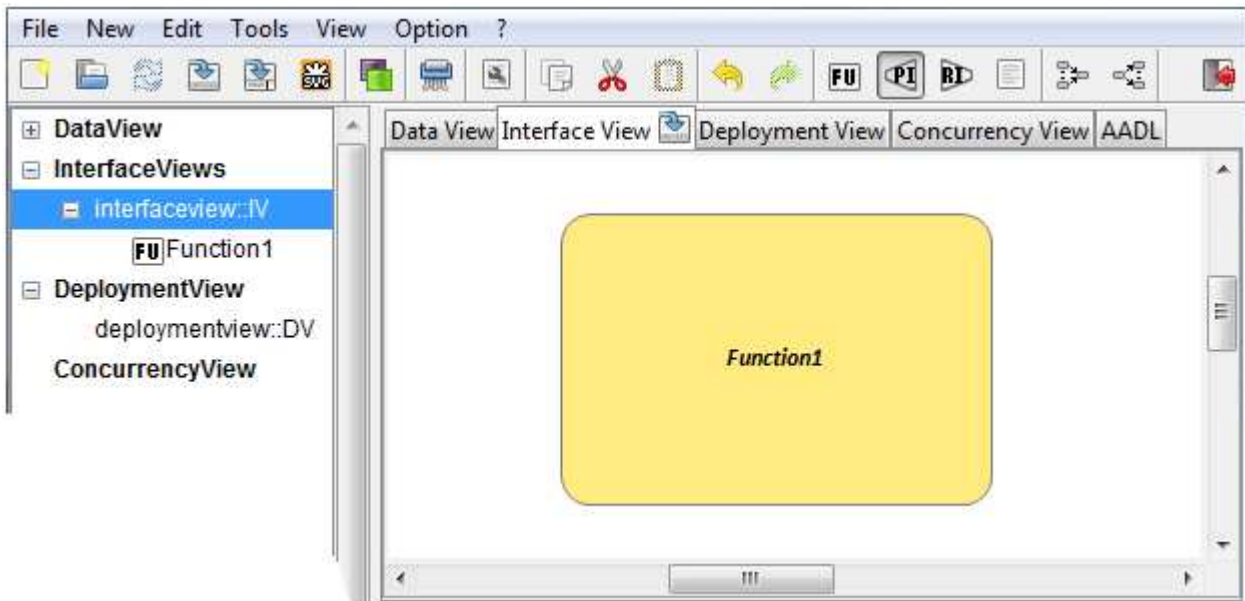
Within the TASTE Linux only, these source contents may be modified and the composite AADL file is updated and re-loaded. Outside the TASTE VM, a message “Error in dataview generation” appears in the status bar.

## 5.2. Build an Interface View

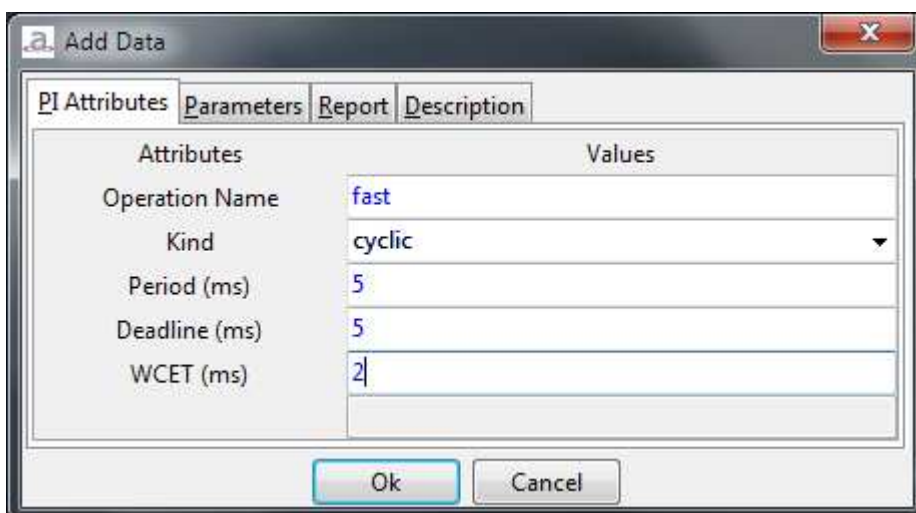
Select the *Interface View* tab and press the *New Function* button:



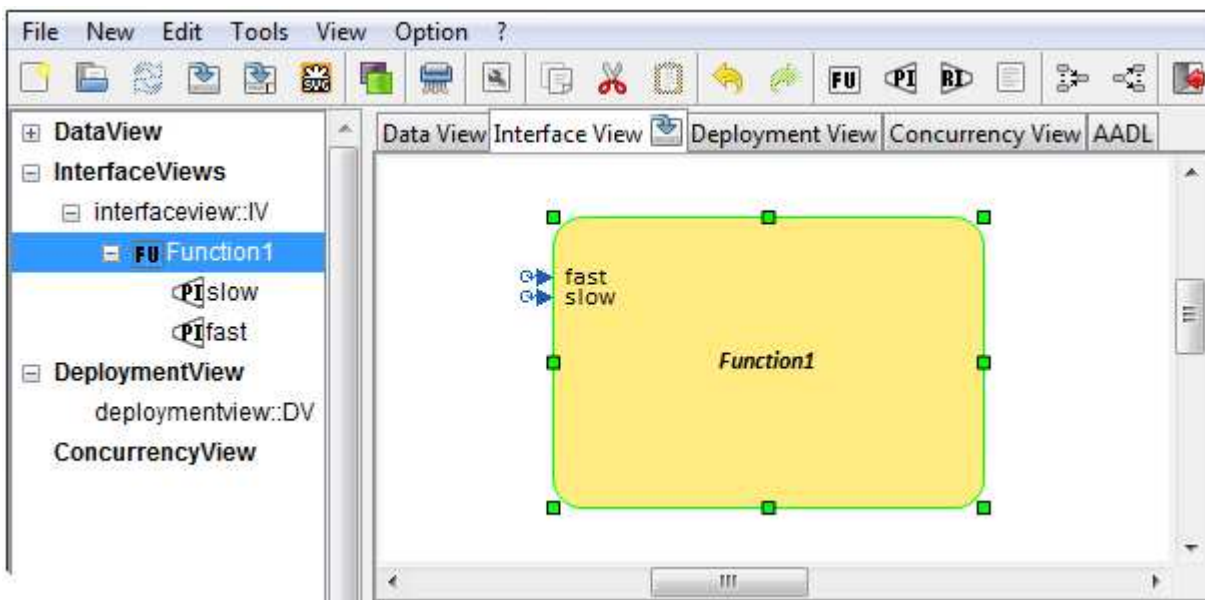
Draw the Function box in the working area and then select the *New PI* button to create a Provided Interface:



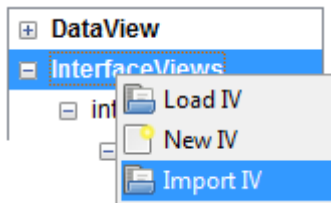
While clicking on a border of the Function, a dialog box automatically opens that can be filled in for instance as follows:



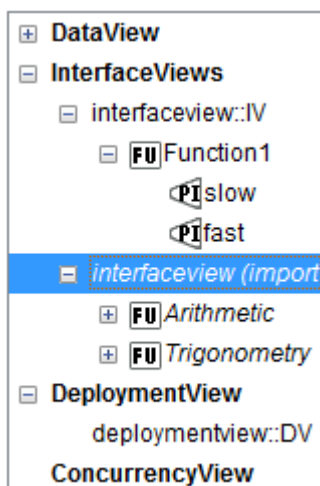
We can then create another Provided Interface with the following attributes: slow, cyclic, 10, 10, 5. Our Interface View now looks like the following:



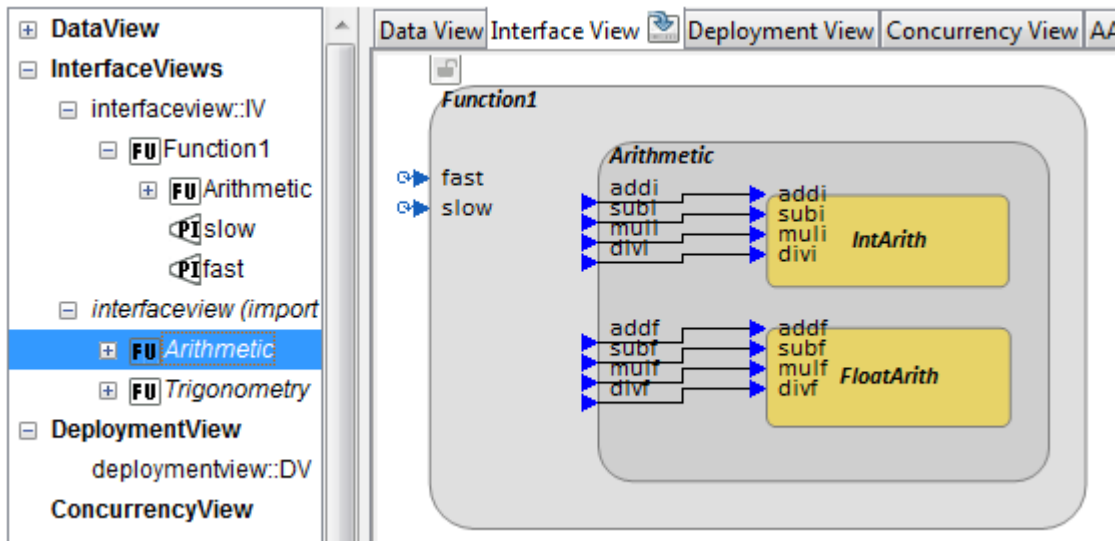
We may also wish to reuse a Function that was defined in another Interface View model. To do so, we will import this other IV model thanks to the *Import IV* contextual menu of the browser:



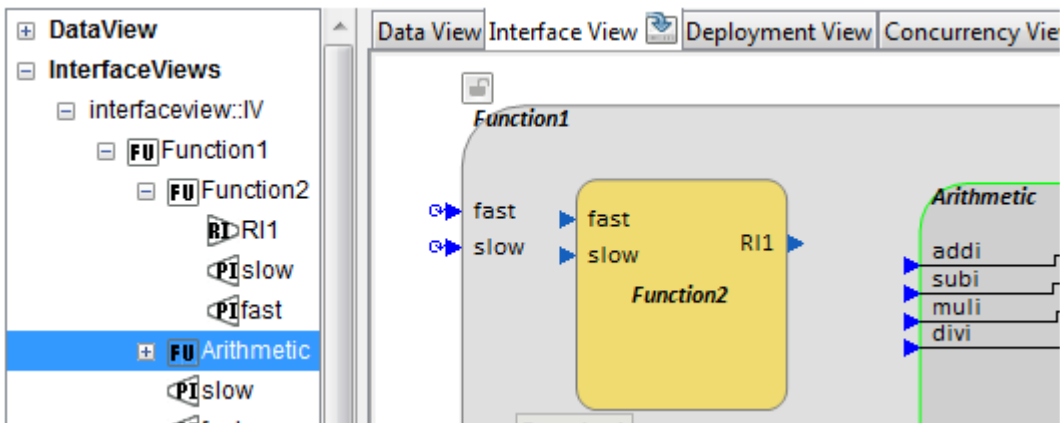
Choose the MathLib.aadl file that is provided in the Workspace directory. The browser now shows the new imported Functions Arithmetic and Trigonometry.



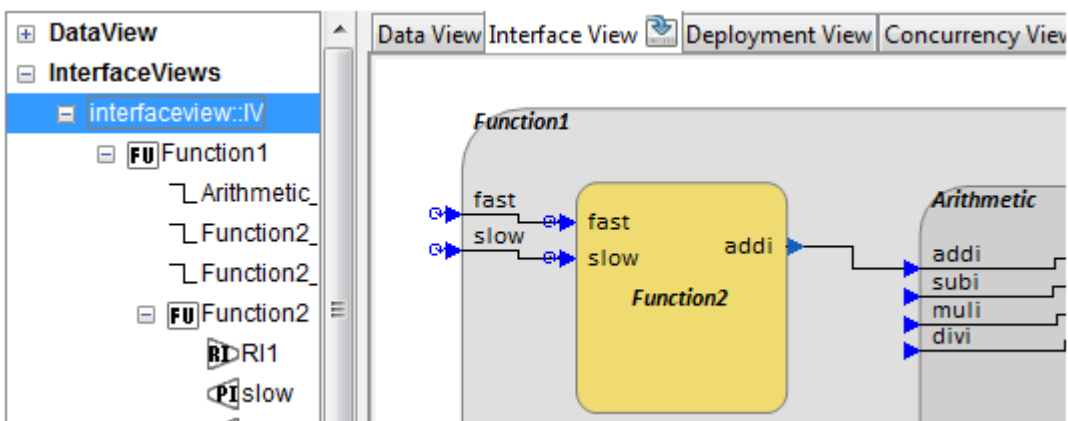
After having zoomed in a little the Function1 in the working area, drag and drop the imported Arithmetic Function inside it. This creates a local copy of the imported Function inside the current Interface View.



We can now complete the Interface View model by adding a child Function with two PIs and a RI. To create a RI, use the *New RI* button.



The missing connections can be drawn by clicking on the first PI or RI and dragging the mouse until it reaches the second PI or RI. Note that by default a RI inherits the attributes of the connected PI and for PI-PI or RI-RI connections the second end inherits from the first one.

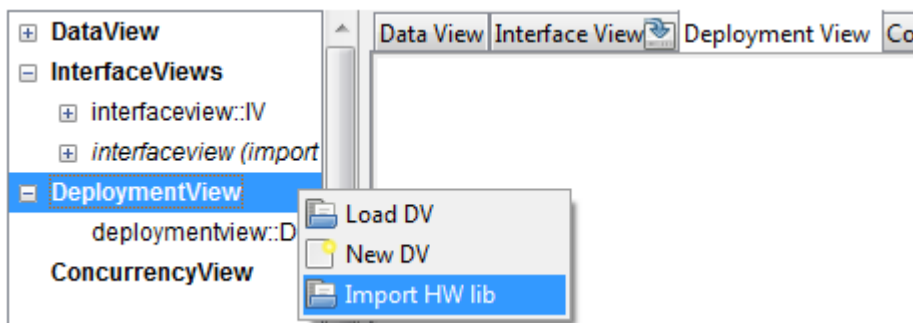


Note that Function attributes can be edited thanks to the Edit Properties button or contextual menu, or by using the ctrl-e keyboard shortcut.

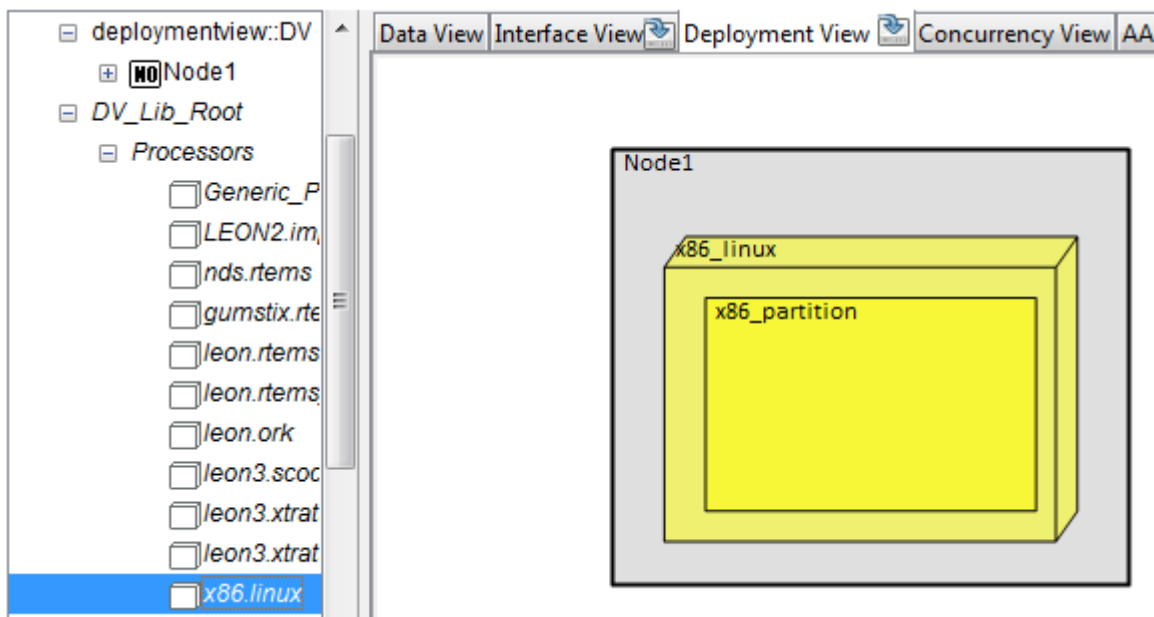
### 5.3. Build a Deployment View

The Deployment View describes the hardware environment on which the software application defined by the Interface View can run. Only those hardware components that are supported by the Ocarina code generator can be used in a TASTE Deployment View. That's why the first step consists in loading the predefined Hardware Library. A template of this library is provided in the Workspace directory to be used when the editor is operated outside the TASTE VM.

To load the hardware library, select the *Deployment View* tab and the *DeploymentView* heading in the browser, then use the *Import HW lib* contextual menu. Then choose the *TestLibHW.aadl* file in the Workspace directory.



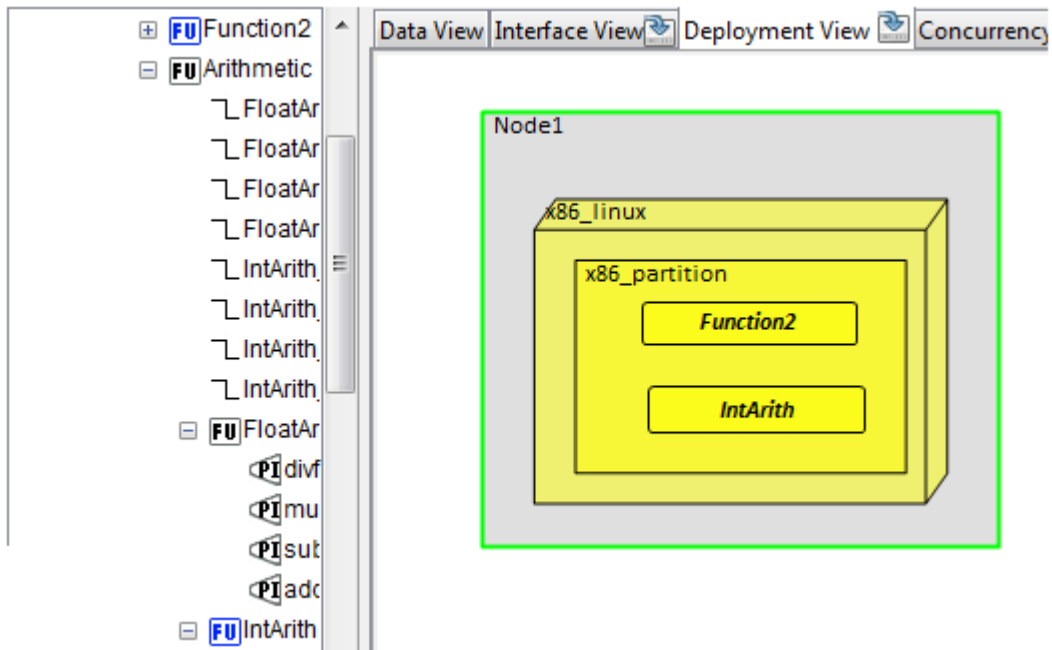
This action adds a set of new items in the browser. These items are organised in three categories: Processors, Devices and Buses. In this short tutorial, we will only use a Processor. Select *x86.linux* in the list of Processors and drag & drop it onto the working area to create a new deployment node.



We now need to allocate the Interface View Functions to the Partition contained by this Node. To do so, while remaining on the Deployment View working area, it is necessary to deploy the *InterfaceView* sub-tree in the browser to show the available Functions. Then drag & drop the two used terminal Functions onto the Partition.

Note that as opposed to the reuse of Functions in the Interface View, the reuse of hardware components corresponds to a reference and not to a copy.

Also note that non terminal Functions of the IV cannot be bound.



Note that the IV Functions that have been bound to a deployment Partition are displayed in blue. We can now analyse the timing behaviour of our application using the Concurrency View.

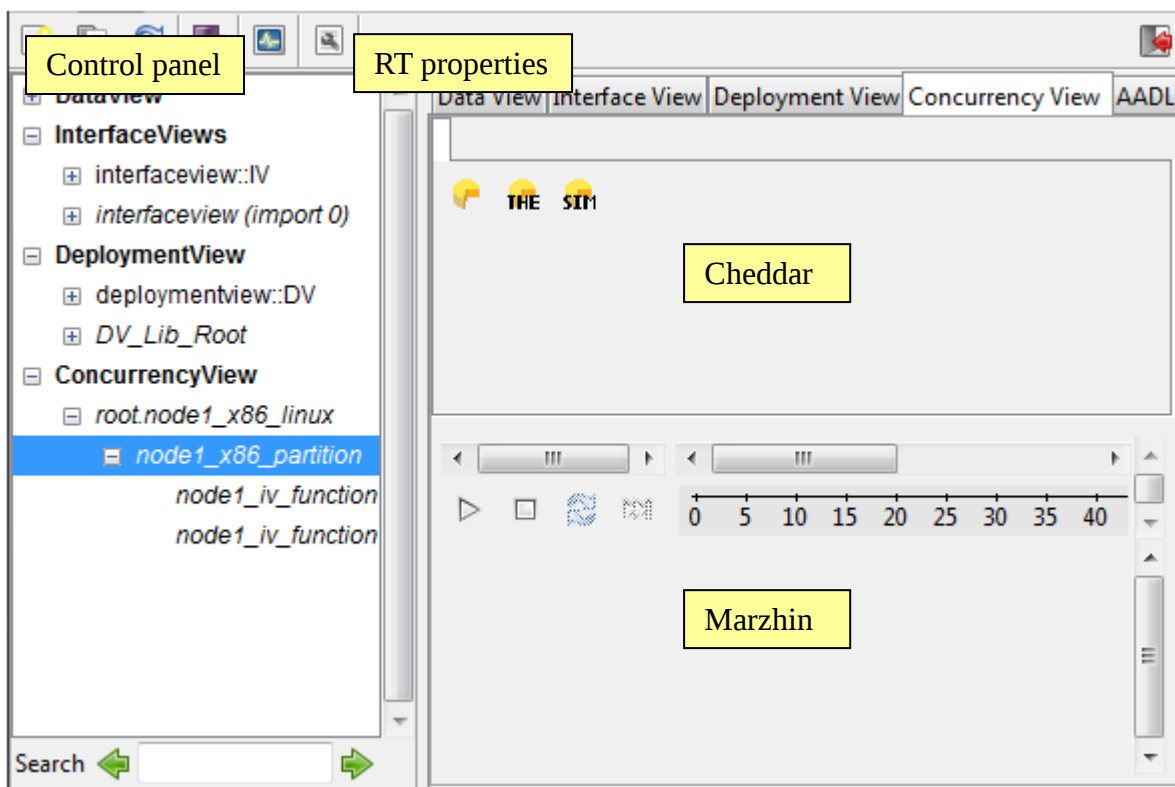
## 5.4. Build and analyse the Concurrency View

The TASTE Concurrency View is the result of a model transformation from the DataView, the Interface View, the Deployment View and the Hardware Library. It describes the architectural model of the future software and can thus be analysed to perform early verifications before code generation.

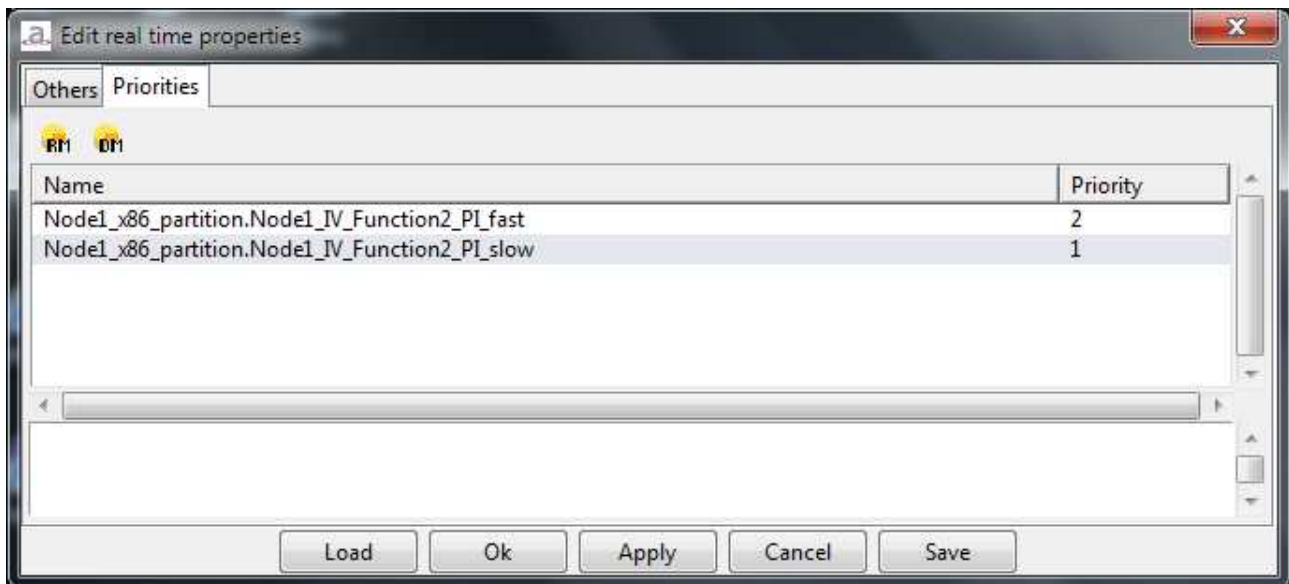
To build the Concurrency View, use the menu **Tools/Build Concurrency View**. Within the TASTE VM, this activates a service of the Build Support Utility. For standalone use of the editor outside the VM, a simplified transformation has been implemented. The results may thus differ in the two cases. Once the Concurrency View has been created, it appears in the browser.

Note that both the Interface View and the Deployment View must have been saved before building the Concurrency View.

To perform the timing analysis on the Concurrency View, select the *Concurrency View* tab. The working area now contains two parts: the upper pane is used to display the results of the scheduling analysis tools provided by Cheddar. These tools can be activated by the three yellow buttons (schedule table, theoretical tests and simulation tests). The lower pane is dedicated to the display of the Marzhin event based simulator traces. The main button bar has two specific buttons: one to open the simulator control panel and another one to edit the main real-time properties of the Concurrency View.



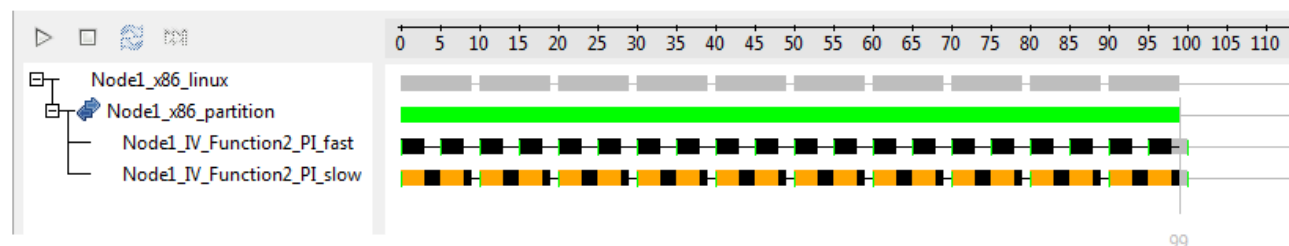
Press the *Edit properties* button to show the current real time properties that have been deduced from the Interface View. This action opens a dialog box where information about the two concurrent periodic threads of the application is displayed. Select the *Priorities* tab and use the *RM* button to assign a priority to each thread according to the Rate Monotonic law, and then close the dialog box by pressing the *Ok* button.



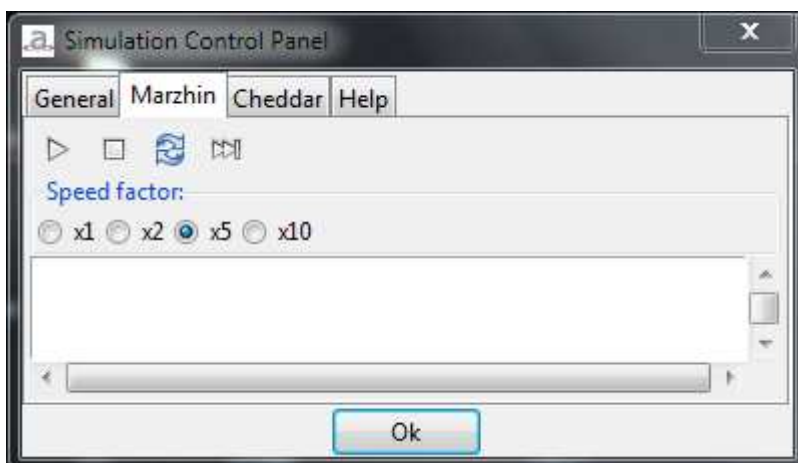
Now, press the *SIM* button on the Cheddar upper pane, the result of the schedulability tests performed by static simulation over the hyper period is displayed:

| test                                                                 | entity                        | result                                                                      |
|----------------------------------------------------------------------|-------------------------------|-----------------------------------------------------------------------------|
| <input checked="" type="checkbox"/> Task response time computed from | Node1_x86_linux               | No deadline missed in the computed scheduling : the task set is schedulable |
| Number of preemptions                                                | Node1_x86_linux               | 1                                                                           |
| Number of context switches                                           | Node1_x86_linux               | 3                                                                           |
| Task response time computed from                                     | Node1_x86_linux.Node1_x86_par | worst = 3, best = 3 and average = 3.00000                                   |
| Task response time computed from                                     | Node1_x86_linux.Node1_x86_par | worst = 9, best = 9 and average = 9.00000                                   |

Press the *Start Simulator* button on the Marzhin lower pane, the simulation trace is then displayed tick after tick:

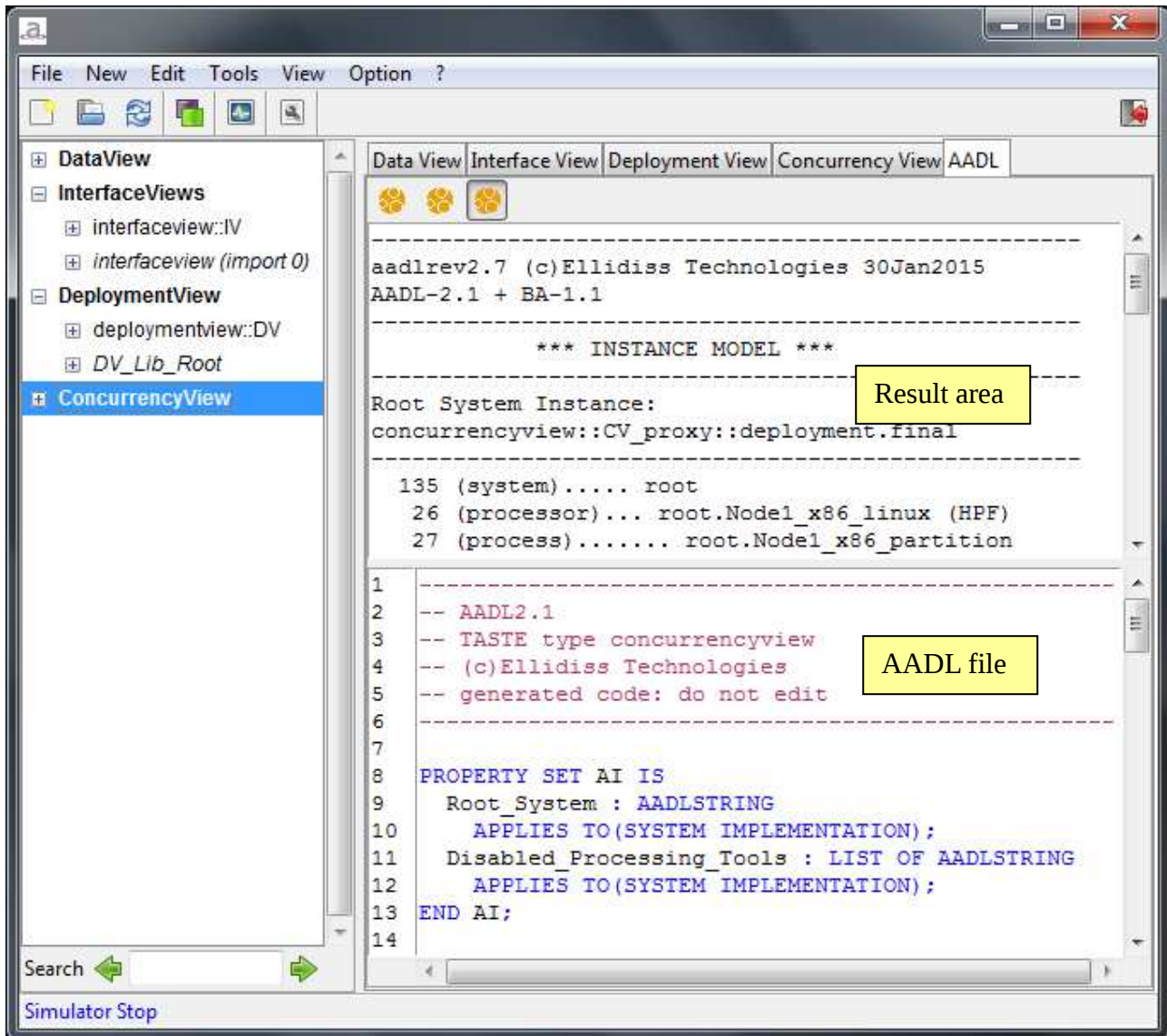


Note that the speed of the simulator can be controlled thanks to the simulation control panel that can be opened thanks to the corresponding button in the main button bar.



## 5.5. Display and verify the generated AADL code

Most of the times, it is not necessary for the TASTE user to look at the generated AADL files. However, the TASTE editor offers several tools to perform verifications at that level if required. To access these tools, select the *AADL* tab.



The working area is separated in two parts. The lower part shows the contents of the AADL file corresponding to the TASTE model selected in the browser. When the Concurrency View is selected, the upper part can show the result of the three AADL verification tools that are proposed:

- The *parse model* button indicates if the Concurrency View can be parsed by the Ocarina tool
- The *instantiate model* button indicates if the Concurrency View can be instantiated by Ocarina.
- The *Metrics* button gives the result of the analysis of the Concurrency View by the LMP AADL toolbox.