TASTE Tutorial

Julien Delange <julien.delange@esa.int>

01/02/2012

Table of Contents

Prerequisites

- Having the Virtualbox image from the following location : http://download.tuxfamily.org/taste/aadl-tutorial-vm.tgz
- A functional installation of Virtualbox on your personal computer. Virtualbox is an open-source software that can be get on https://www.virtualbox.org/

Import the virtual machine

- Uncompress the virtual machine image you download from the TATE download website
- Start VirtualBox and import the Virtual Machine by selecting the `File` → `Import Appliance` option. A window (like in Illustration 2 would appear and let you select the Virtual Machine Archive file)
- Select the `.ova` file that was created when uncompressing the virtual machine image
- Proceed and check the virtual machine settings. Please note that the MAC address of the virtual machine has to be reinitialized, especially if several users are suppose to use the same virtual machine in the same location. The Virtual Machine settings shall look like the one of Illustration 1
- Please wait until the completion of the process (import process looks like the screen of Illustration 3). Once complete, the new Virtual Machine would appear in Virtual Box, as illustrated in Illustration 4.

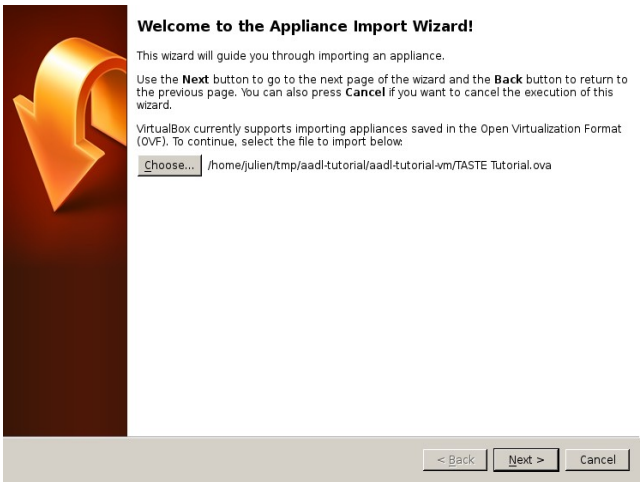

Illustration 2: Import Appliance main screen

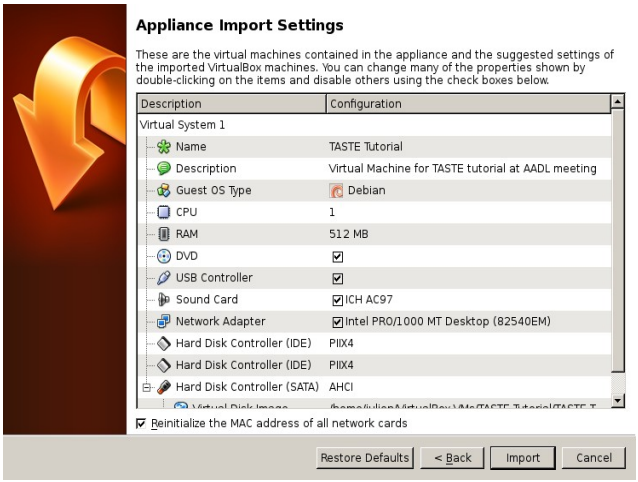

Illustration 1: Virtual Machine settings

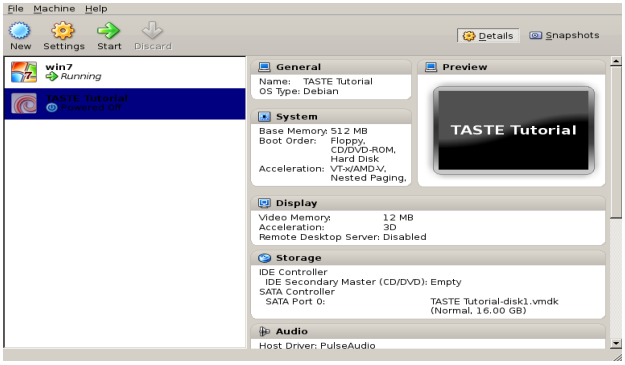

Illustration 3: Main Window of Virtualbox

Illustration 4: Import virtual machine screen

Start the virtual machine

- Choose the virtual machine in Virtualbox (main window Illustration 4) and select `Start`. Then, the virtual machine will start with a graphical environment, as illustrated in Illustration 5.
- You will then get a graphical environment with the following icons:
  - FAQ.txt: Frequently Asked Questions about the Virtual Machine, the user/password required to execute the Virtual Machine, etc.
  - File Manager: a explorer.exe clone to explore the virtual machine filesystems
  - Update-TASTE: a dedicated program to update the TASTE tools to the latest version
  - LXTerminal: a lightweight terminal
  - TASTE documentation: shortcut to the main TASTE documentation, explaining and detailing TASTE concepts and some specific aspects (device drivers configuration, import of code from Simulink/RTDS, etc.)
  - TASTE-GUI: shortcut to the TASTE graphical interface tool that provide guidance for the use of each toolchain program.
  - TASTE Quick Reference: a short notice about basic use of TASTE. This is a short version of the TASTE documentation
  - Change-Resolution: a program that automatically change screen resolution if you experience any problem with the initial configuration.
  - To start the TASTE graphical tools, double-click on the TASTE-GUI icon. Then, the TASTE graphical tool would appear, as in the Illustration 6.

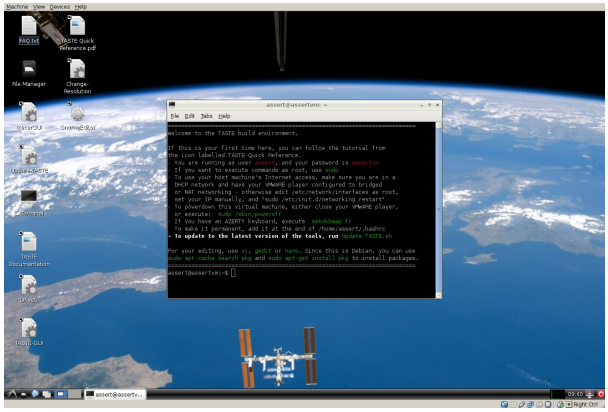

Illustration 5: Virtual Machine initial interface

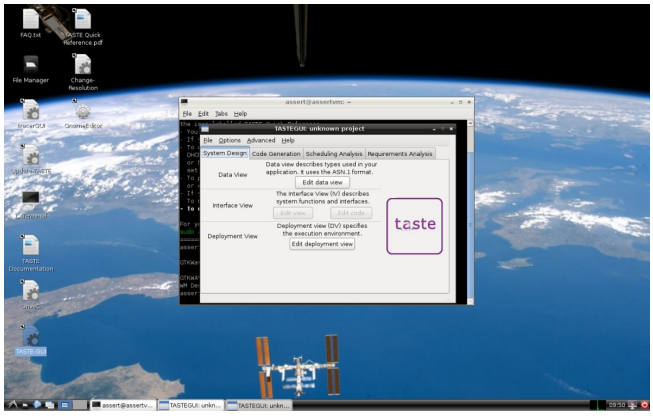

Illustration 6: TASTE GUI in the Virtual Machine

Building your first system

Data View

1. Define the system data view. To do so, click on "`Edit Data View`" in the main TASTE interface and put the code shown in Text 1.
2. Click on Save.

```
DataView DEFINITIONS AUTOMATIC TAGS ::=
BEGIN
My-Integer ::= INTEGER (0 .. 65535)
END
```

Text 1: Data View for the first system

Interface View

1. Click on "`Edit View`" on TASTE main interface. The main TASTE Interface View editor will start and provides you the ability to define system functions and interfaces.
2. Add a first **C-fuction** called "`producer`". To do so, click on the `FU` button in the TASTE-IV toolbar and draw the function in the graphical area. The label and instance name are "`producer`". The implementation language (`Language` property of the function) will be C.
3. Add a **Cyclic Provided Interface (PI)** to the `producer` function called activator. To do so, issue a right click on the producer function and choose "New PI". Then, fill the dialog box with the following requirements:
    - **Operation name**: activator
    - **Kind**: cyclic
    - **Period**: 15
    - **Deadline**: 15
    - **WCET**: 4
    - **Unit**: ms
4. Add a second C-function called "`consumer`". To do so, proceed as for the "`producer`" function.
5. Add a **Sporadic Provided Interface** in the new "consumer" function. To do so, right-click on the "`consumer`" function and click on the "`Add PI`" menu item. Add the Provided Interface with the following requirements:
    - **Operation name**: receiveint
    - **Kind**: sporadic
    - **Minimum inter-arrival Time**: 10
    - **Deadline**: 10
    - **WCET**: 3
    - **Unit**: ms
    - **Queue size**: 1
6. Add also **one parameter** to the `receiveint` interface. To do so, click on the parameters submenu in the interface window and finally on the + button. The parameter shall have the following requirements:
    - **Name**: val
    - **Type**: My_Integer
    - **Encoding Protocol**: Native
    - **Direction**: in
7. Then, you have to connect both function and introduce a link between the producer and the consumer functions. To do so, right-click on the producer function and choose to add a RI. Then, a new arrow will be added on the "`producer`" function. You have to connection this new **Required Interface (RI)** from the `producer` function with the **Provided Interface (PI)** `receiveint` from the `consumer` function.
8. Once you finish to edit your interface view, your virtual machine environment would look like Illustration 7.

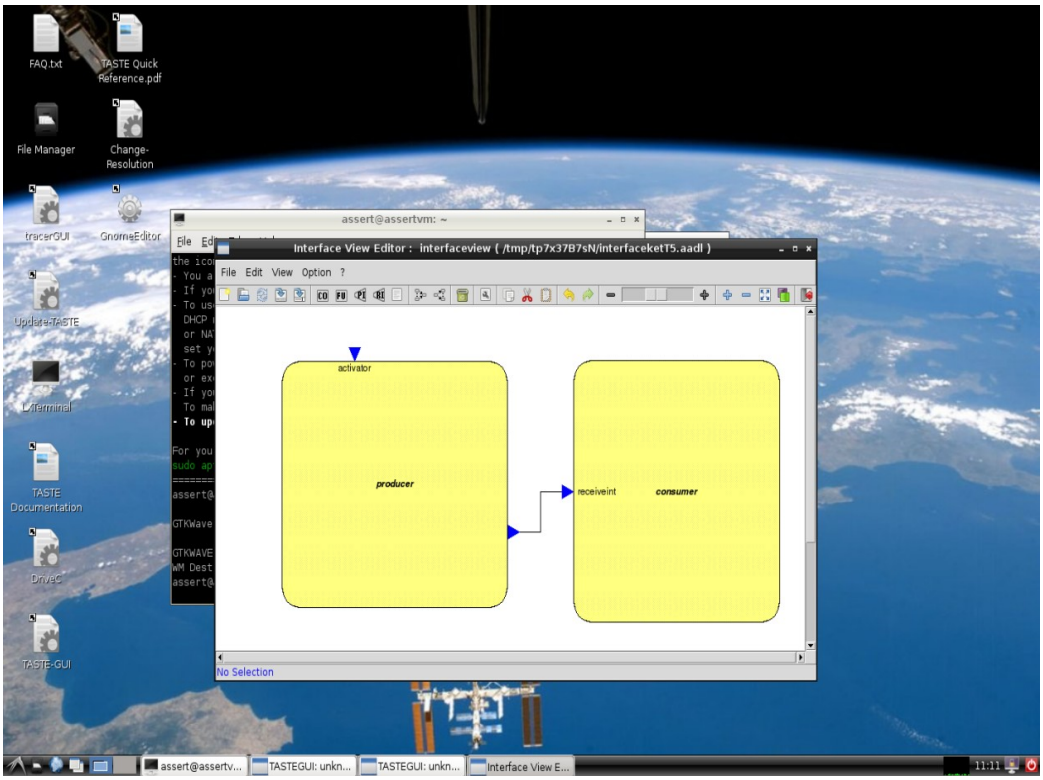

Illustration 7: Virtual Machine with the complete Interface View

Write interface code

- Click on `Edit Code` in the main TASTE GUI interface
  - Choose the `producer` function
    - Complete the code like the one in Text 2
    - Save the file by clicking on `File` → `Save`
- Click on `Edit Code` in the main TASTE GUI interface
  - Choose the `consumer` function
    - Complete the code like the one in Text 3
    - Save the file by clicking on `File` → `Save`

```
#include "producer.h"
#include <stdio.h>

void producer_startup()
{
        printf ("Start the producer\n");
}

void producer_PI_activator()
{
        static asn1SccMy_Integer myval = 0;
        printf ("Send value %lld\n", myval);
        producer_RI_receiveint (&myval);
        myval++;
        fflush (stdout);
}
```

Text 2: C code for the producer function

```
#include "consumer.h"
#include <stdio.h>

void consumer_startup()
{

}

void consumer_PI_receiveint(const asn1SccMy_Integer *IN_val)
{
        printf ("Receive %lld\n" , *IN_val);
        fflush (stdout);
}
```

Text 3: C code for the receiver function

Deployment View

Once functional code is written, deployment consideration must be specified. To do so:

1. Click on `Edit deployment` view in the main TASTE window. It will open the tool TASTE-DV, the deployment view editor.
2. Add a **processor board** by clicking on the **PB** icon in the toolbar.
3. Double-click on the `processor` and edit its properties. Set its values to the following:
    - **Name**: `tutorialcpu`
    - **Classifier**: `ocarina_processors_x86::x86.linux32`
4. Add the `producer` function to the partition by clicking on the **FU** button in the toolbar and drawing the function in the partition of the processor board.
5. Add the `consumer` function to the partition by proceeding as previously but with the consumer function.
6. Save the project and close the program

Once the deployment view is complete, your workspace would look like Illustration 8.

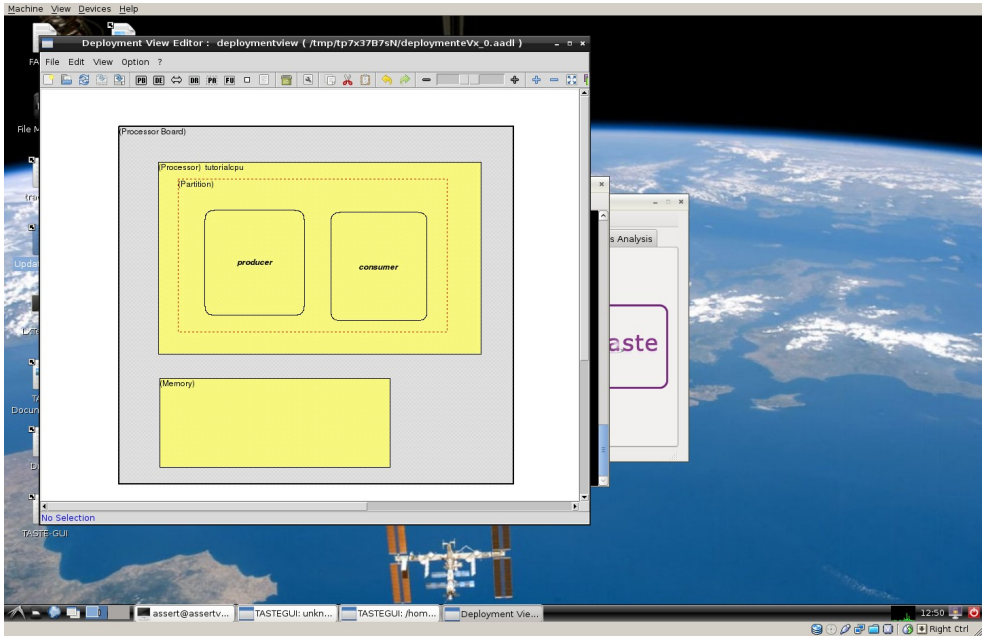

Illustration 8: Workspace with the complete deployment view

Choose the C runtime, compile and configure system execution

First, **you must choose a runtime to support the execution of the generated code**. You must change the default runtime (the Ada runtime) and use the C runtime. To do so, click on `Options` → `PolyORB-HI-C`.

Then, to compile your system, click on the `Code Generation` menu in the main TASTE GUI window and click on the `Compile` button. A new window would appear, indicating the status of the process. If all steps were correctly done, the build would be successful and you should have a screen similar to Illustration 9.

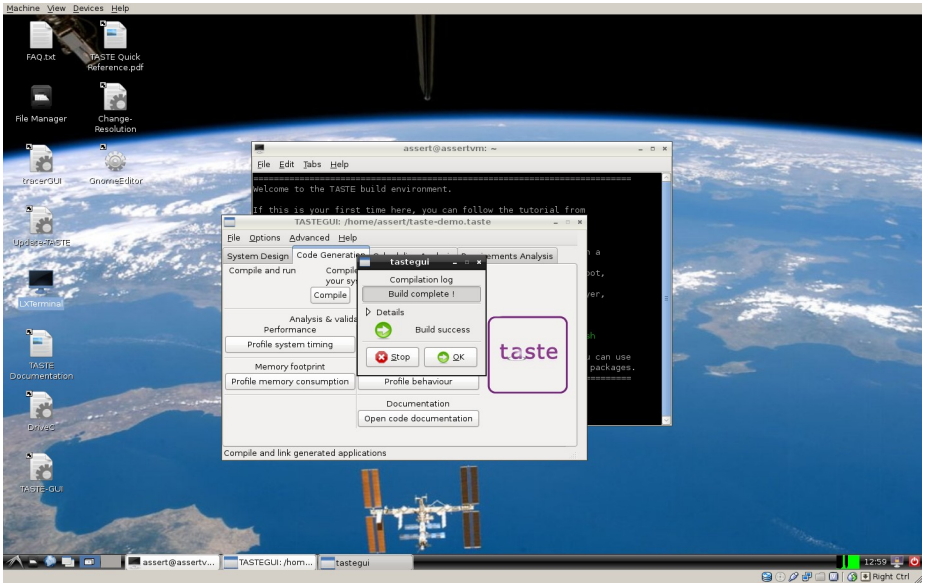

Illustration 9: Workspace with successful compilation

Finally, you have to describe how your system is executed. As we deploy it for the native platform, it will run within the virtual machine. To specify the deployment and execution aspects:

1. Click on the `Configure` button of the `Code Generation` menu
2. Select the node to be generated/executed
3. Choose the **native execution** method (as in Illustration 10)
4. Click on **OK**

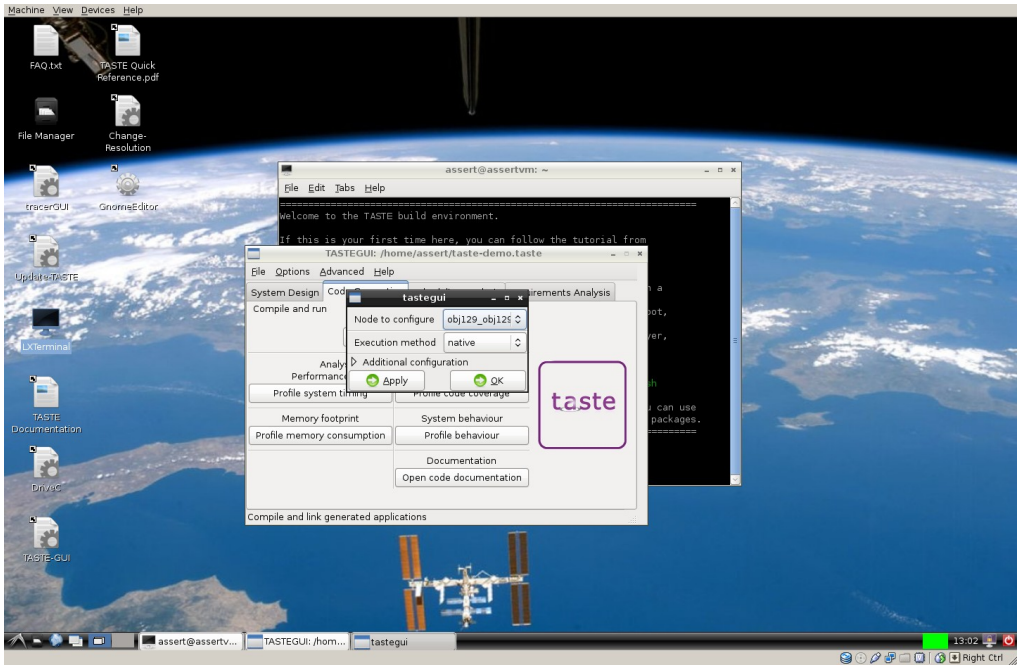

Illustration 10: Deployment configuration for the system

Execute your system

Once everything was correctly specified, system can then be executed. To do so, click on the `Execute` button on the `Code Generation` menu. Execution would then be traced in a dedicated window, as in Illustration 11.

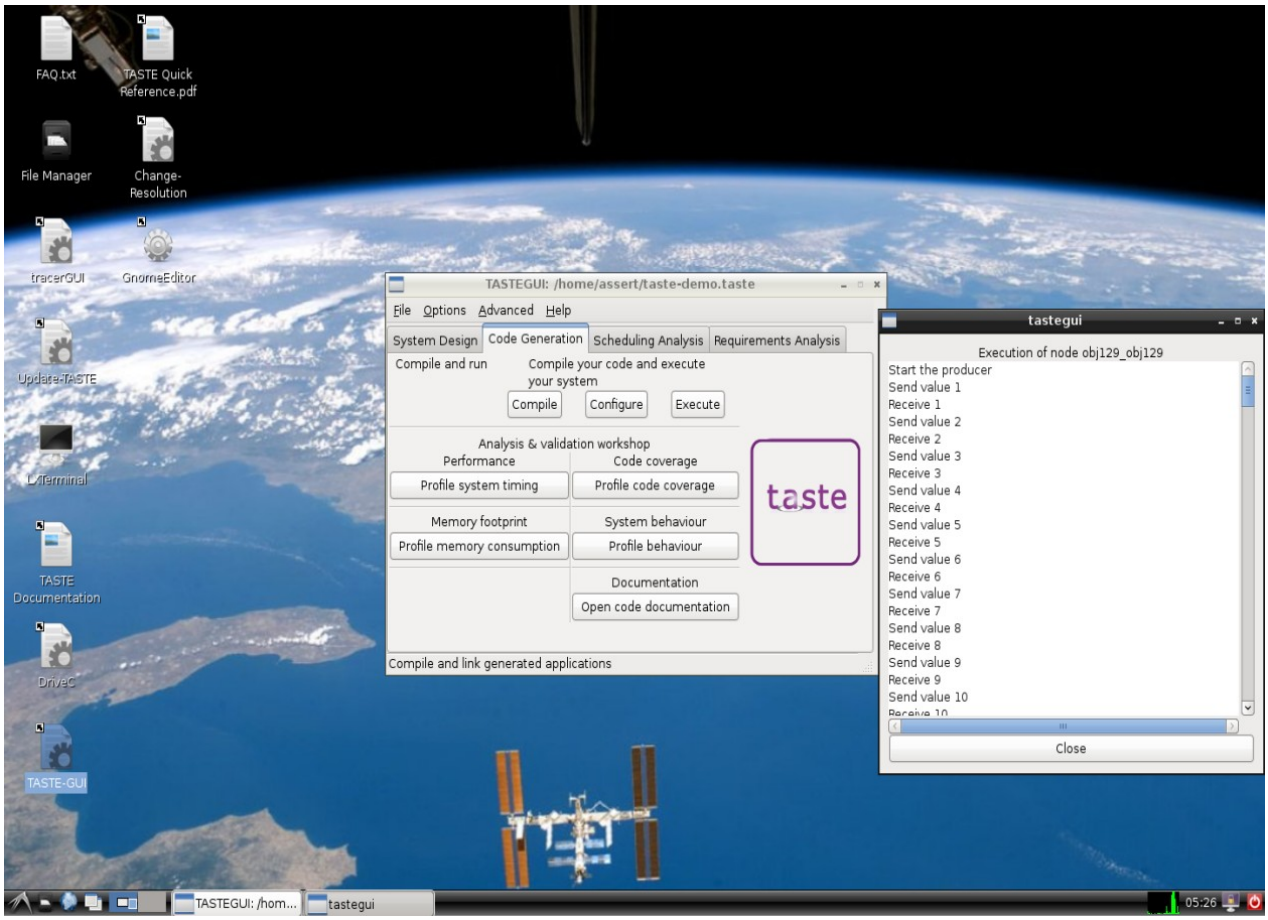

Illustration 11: Workspace with the execution of the system

Inspecting AADL generated code with full execution semantics

The AADL interface view and deployment view represent hardware and software system specifications at a high-level. Even if they describe system requirements, they do not bind both aspects (how the software is executed, which resources are used, etc.). Combining these two aspects is done with a specific tool (`buildsupport`) that transforms the AADL interface and deployment view into a concurrency view that describes resources usage and software/hardware association.

To see the concurrency view, you can do the following:

1. Click on `Scheduling Analysis` menu
2. Click on `Launch TASTE-CV` button
3. The TASTE concurrency view editor opens and shows the AADL concurrency view on the left

Scheduling analysis with TASTE-CV

The same tool (TASTE-CV) used to inspect the concurrency view can be used to perform scheduling analysis. TASTE-CV provides one scheduling analysis function and one scheduling simulation function.

Scheduling analysis with Cheddar

TASTE-CV provides function to make schedulability analysis using the Concurrency View specifications. To use this functionality:

1. Start TASTE-CV by clicking on the Scheduling Analysis menu of the TASTE GUI tool
2. Click on Launch TASTE-CV button
3. In TASTE-CV, click on the cheese button on the upper-right part of the window
4. Result of Cheddar analysis will then appear

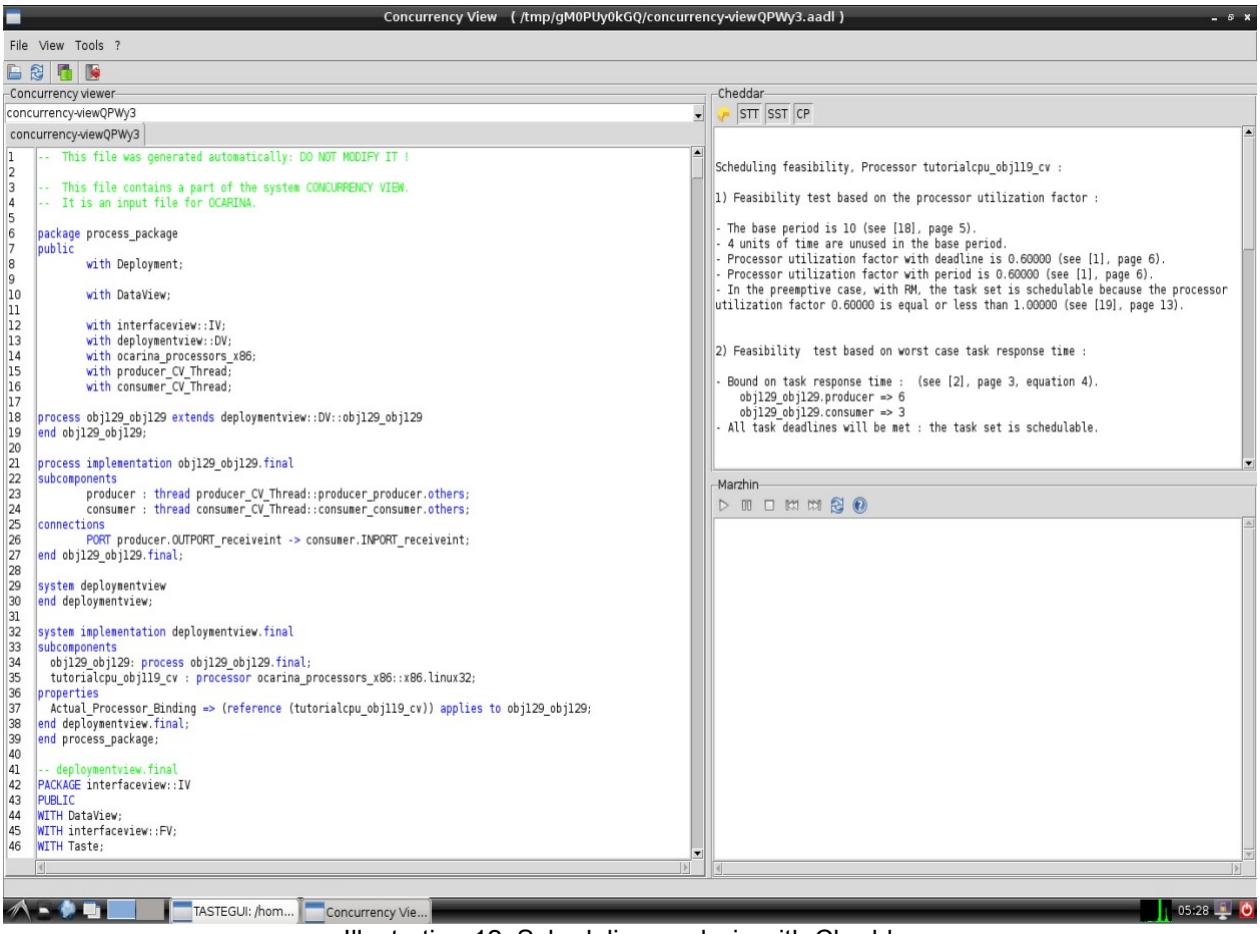

Illustration 12: Scheduling analysis with Cheddar

Scheduling simulation with Marzhin

TASTE-CV provides the ability to simulate system execution, showing task activity and data contained in AADL data ports or AADL event data ports. This is done with Marzhin. To start Marzhin and simulation facilities:

1. Start TASTE-CV by clicking on the Scheduling Analysis menu of the TASTE GUI tool
2. Click on Launch TASTE-CV button
3. In TASTE-CV, click on the play (▸) button on the lower-right part of the window
4. Simulation of the system then start, your workspace shall look like Illustration 13.

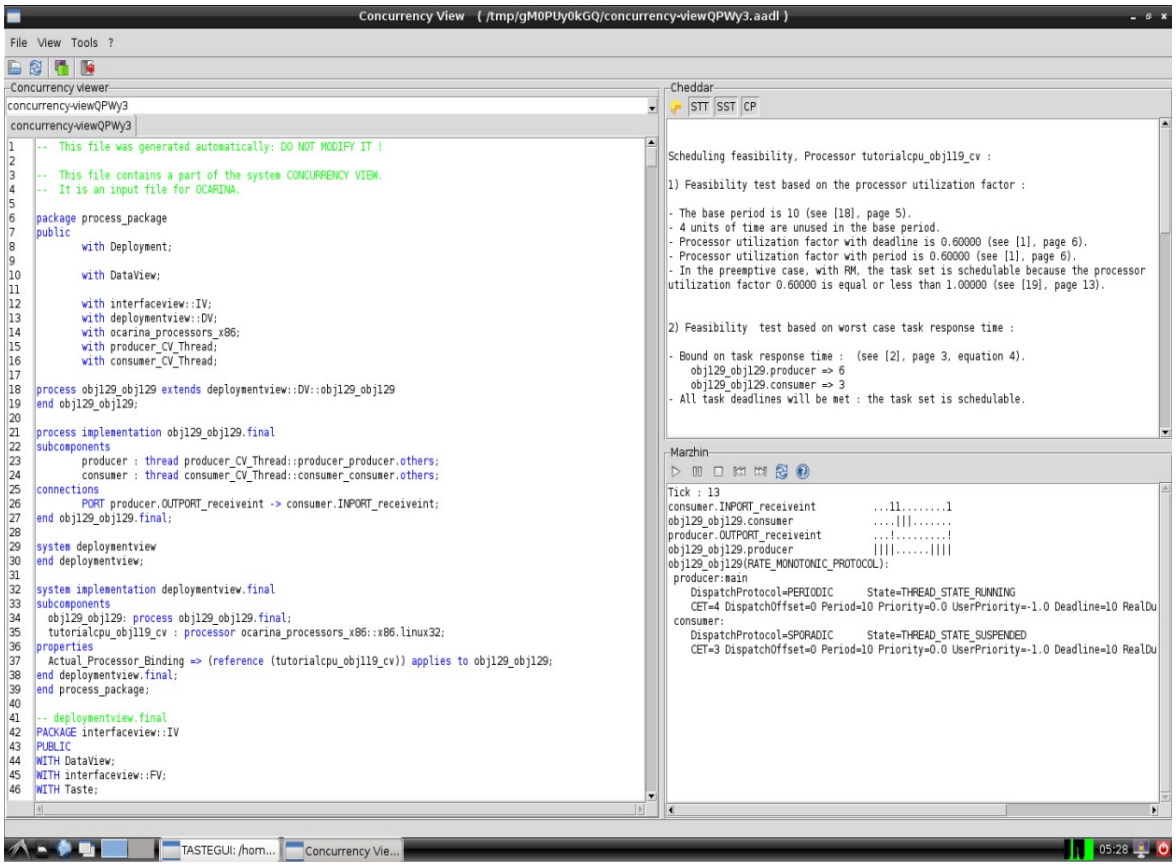

Illustration 13: Scheduling simulation with Marzhin

However, users must have in mind that this simulation functionality has some limitations:

1. Time units are not taken in account, the tool consider only the numeric value of tasks (period, deadline, etc.) so that all values of the AADL model must have the same unit.
2. The tool consider only the worst case execution time whereas execution can be faster
3. At this time, only local system (not distributed system) can be simulated.

Scheduling analysis with MAST

TASTE also provides the capability to export the AADL concurrency view (software and hardware specifications) into a MAST model to perform schedulability analysis. To do so:

1. Open the scheduling analysis menu blicking on the Scheduling Analysis menu of the TASTE GUI tool
2. In the MAST schedulability analysis menu, choose a scheduling algorithm for the analysis. Select for example *“Offset Based Optimized”*.
3. Click on Launch MAST. The tool shall appear with the analysis result. As a result, your workspace shall look like Illustration 14.

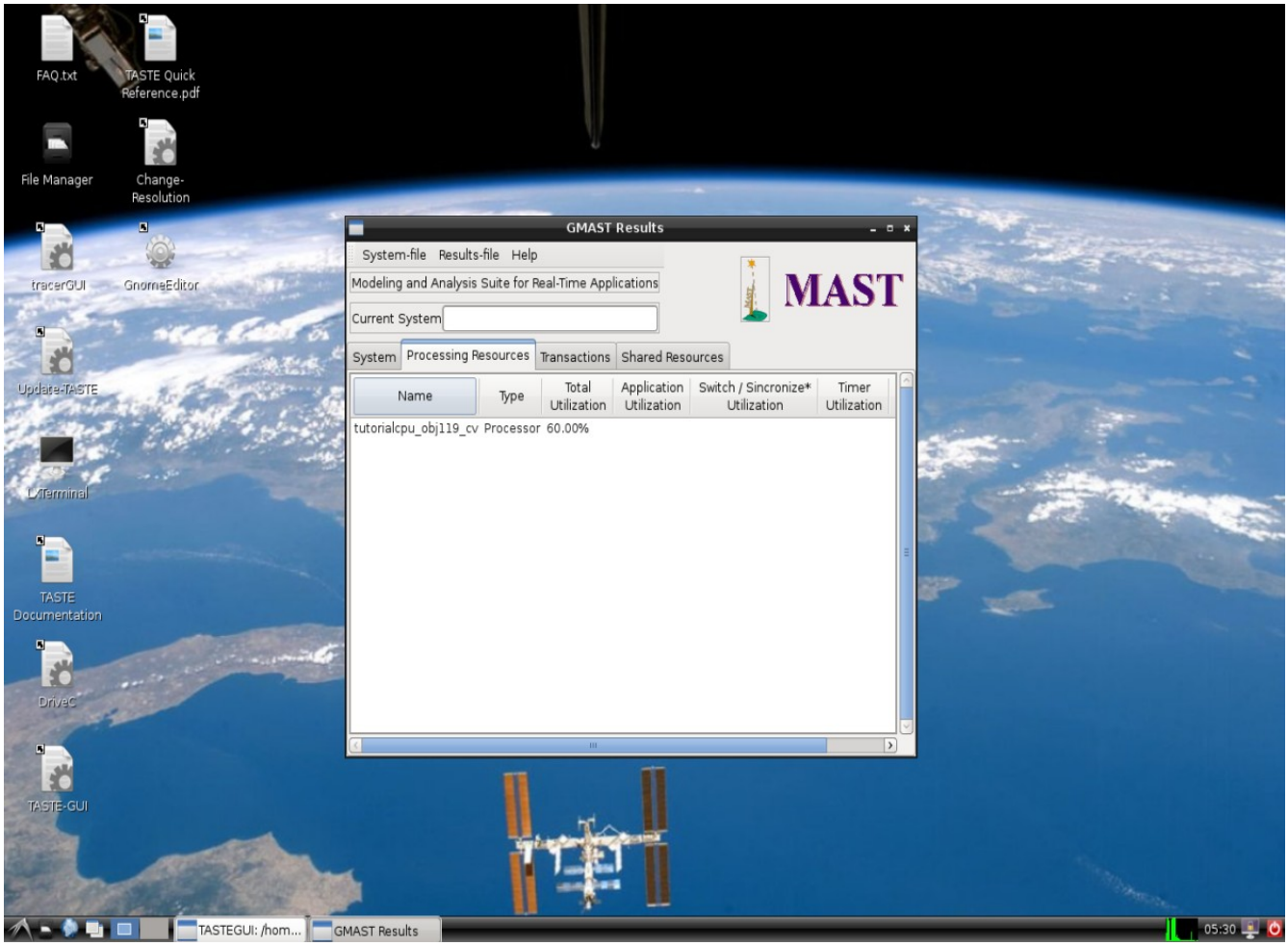

Illustration 14: Scheduling feasibility test using MAST

Using a safe project already completed

If you experience any issue, you can open an existing project within the VM. It contains the full project with the interface view, deployment view and also the functional code. Usnig it could help you to find out where are errors in your system. To open it, start the TASTE GUI (double-click on TASTE-GUI icon on the desktop) and then:

1. Click on `File` → `Open Project`
2. Click on the `assert` home logo on the left colon
3. Choose the `taste-demo.taste` file

Known issues

I need to update the virtual machine, how can I do ?

Click on the UPDATE-TASTE icon on the desktop. You may experience network issues while updating your VM. In that case, please read the appropriate section.

I don't have the network

Please check the virtual machine settings and that they are correct according to your machine configuration. For example, have a look at the connection method for your network interface (bridged network/NAT/etc...). Please refer to the official virtualbox documentation if necessary. Most of the time, the network interface shall be configured as NAT but this might not be suitable for all configuration, especially if your machine is under strong network constraints with a strong filtering policy.

You might also experience another error due to the import of the VM. In that case, pleas try the following:

1. Open a Terminal (double-click on LXTerminal on the desktop)
2. Type the following command: `sudo dhclient eth1`

I don't see any outputs when executing the system

Please double check that you use the PolyORB-HI-C runtime by choosing the following item menu in TASTE GUI: `Options` → `PolyORB-HI-C`.

When I modify the system, my changes are not taken in account

To optimize system build, the TASTE compilation process caches as much as possible all binary files and intermediate outputs. When it detects a change in the input, new output is automatically created. However, sometimes, in unexpected case, the build system fails to identify parts that were modified so that it keeps cached items and does not use modified parts of your system.

To handle this issue, we add a special option in TASTE GUI to flush the build cache and make sure the build system will build the whole system from scratch. It result in a longer build but by using it, you are sure that your changes are applied and that everything is generated again from scratch.

To flush the cache, click on `Advanced` → `Delete output directory` in TASTE GUI interface.

I have a technical question, how can I contact the developer team ?

You can send a mail to the TASTE developer mailing list: taste-dev@lists.tuxfamily.org

Links and references

- TASTE website: http://www.assert-project.net/taste
- Semantix TASTE area (ASN1Scc and other related tools): http://www.semantix.gr/assert/downloads.html
- TASTE download area: http://download.tuxfamily.org/taste/
- Ellidiss website (TASTE graphical tools): http://www.ellidiss.com