Modeling GAMs with SysML

&

Automatic GAMs code generation with Model-2-Text tools

G. De Tommasi, R. Vitelli

October 2011

0. DOCUMENT INFORMATION

0.1 DOCUMENT HISTORY AND VERSION CONTROL

REVISION	DATE	STATUS	COMMENTS/MODIFICATIONS
1.0	31/07/2011	First version	
1.1	4/10/2011	Minor changes	The document has been updated taking into account the recent software development of the Acceleo project

0.2 DEVELOPED AND APPROVED BY

		Date
Developed by	Gianmaria De Tommasi	4/10/2011
Reviewed by	???	

0.3 SOFTWARE

Document edition	Microsoft Word 2010

Table of Contents

1. INTRODUCTION

1.1 PURPOSE

This document describes how to model Generic Application Modules (GAM) by using the System Modeling Language (SysML) and how to automatically generate part of the code needed to deploy the modeled components in the MARTe framework.

1.2 SCOPE

GAMs are the atomic elements used to build applications with the MARTe framework ([1]). A GAM is a block of code implementing an interface specified in the BaseLib2 library. GAMs are setup using a Configuration Database (CDB). The core of a typical GAM processes the input accordingly to how it was configured and outputs the modified information.

A GAM is typically coded as a C++ class, hence a *.h* and a *.cpp* file must be produced to implement a GAM. Furthermore the GAM configuration must be specified into a text file named *configuration file*.

SysML is an extension of the Unified Modeling Language (UML) that can be used to model both hardware and software systems ([2],[3]).

This document describes:

- how to model GAMs by using SysML in the Topcased environment ([4]);
- how to exploit the *Acceleo* Model-2-Text tool ([5]) to automatically generate parts of the *.h* and *.cpp* files, together with parts the GAM *configuration file*.

1.3 DEFINITIONS, ACRONYMS AND ABBREVIATIONS

Bdd	SysML block definition diagram
CDB	Configuration Database
GAM	Generic Application Model
Ibd	SysML internal block diagram
SysML	System Modeling Language
UML	Unified Modeling Language

1.4 REFERENCES

[1] A. C. Neto et al., "MARTe: a Multi-Platform Real-Time Framework," *IEEE Transactions on Nuclear Science*, vol. 57, no. 2, pp. 479-486, April 2010.

[2] *SysML Open Source Specification Project website*, http://www.sysml.org/, 2011.

[3] T. Weilkiens, *"Systems Engineering with SysML/UML Modeling, Analysis, Design"*, Morgan Kaufmann OMG Press, 2007, also available at http://www.sysmod.de/

[4] *Topcased website*, www.topcased.org, 2011.

[5] *Acceleo website*, http://www.eclipse.org/acceleo/, 2011.

2. INSTALL TOPCASED

This chapter describes how to install Topcased on your PC, and how to install the Acceleo plug-in develop to automatically generate GAM code from SysML models.

Topcased is an integrated System/Software engineering toolkit compliant with the requirements of critical and embedded applications.

Topcased is based on the Eclipse platform, hence it is available on practically all the existing platforms.

To install Topcased:

1. go to http://www.topcased.org/index.php?idd_projet_pere=52&Itemid=60;
2. download version 5.0.0 for your platform.

Topcased Version 5.0.0 is based on the Eclipse 3.7 platform (Indingo).

Once TOPCASED is installed on your PC in *[TOPCASED_dir]*:

3. download the Acceleo plugin from:
   http://wpage.unina.it/detommas/MARTe-Downloads/SysML/TOPCASED_plugin.zip
4. unzip the contents of the file in *[TOPCASED_dir]/plugins*

3. MODELING GAMs WITH SYSML

This chapter describes how to model a simple GAM using Topcased and SysML.

3.1 Download the BaseLib2 SysML model

The Baselib2 SysML model contains all the common definitions (classes, types, etc.) needed to model GAMs with SysML by using Topcased.

The BaseLib2 SysML model is provided as a Topcased model. In order to use it to model GAMs:

1. download the BaseLib2.zip SysML model from
   http://wpage.unina.it/detommas/MARTe-Downloads/SysML/BaseLib2.zip
2. launch Topcased;
3. select the **File→Import…** menu;
4. select **General→Existing Projects into Workspace** as *import source* and press the *Next* button (see Figure 1);
5. select **Select Archive File** and browse to select the BaseLib2.zip; then press the *Finish* button (see Figure 2);
6. now the BaseLib2 Topcased project appears in the *Topcased Navigator* tree (see Figure 3);

The BaseLib2 model contains both SysML *blocks* (used to model BaseLib2 classes) and d*ata types*, which are used to model GAMs. The current version of the model contains two block definition diagrams (bdd) that specify the modeled classes and data types, respectively (Figure 4 and Figure 5[1]). To open the diagrams select them in the *Outline* tree (see Figure 6).

IMPORTANT: the current version of the BaseLib2 model is a *work-in-progress* version. Additional blocks and data types may be added in future if needed.

[1] Sometimes it is needed to resize the blocks in the diagrams in order to make them appear as shown in Figures 4 and 5. Such *rendering* problem is due to the use of different Topcased versions.

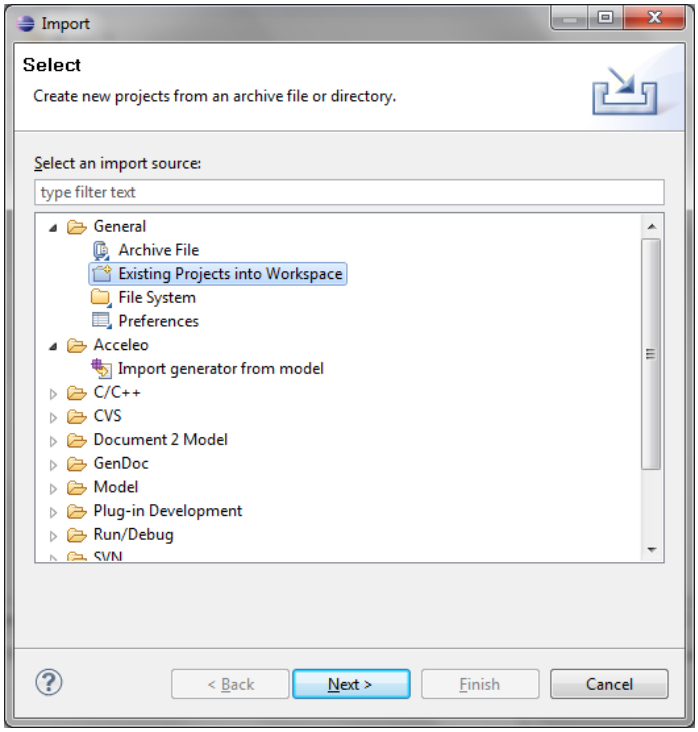

Figure 1 - TOPCASED. File→Import... menu.

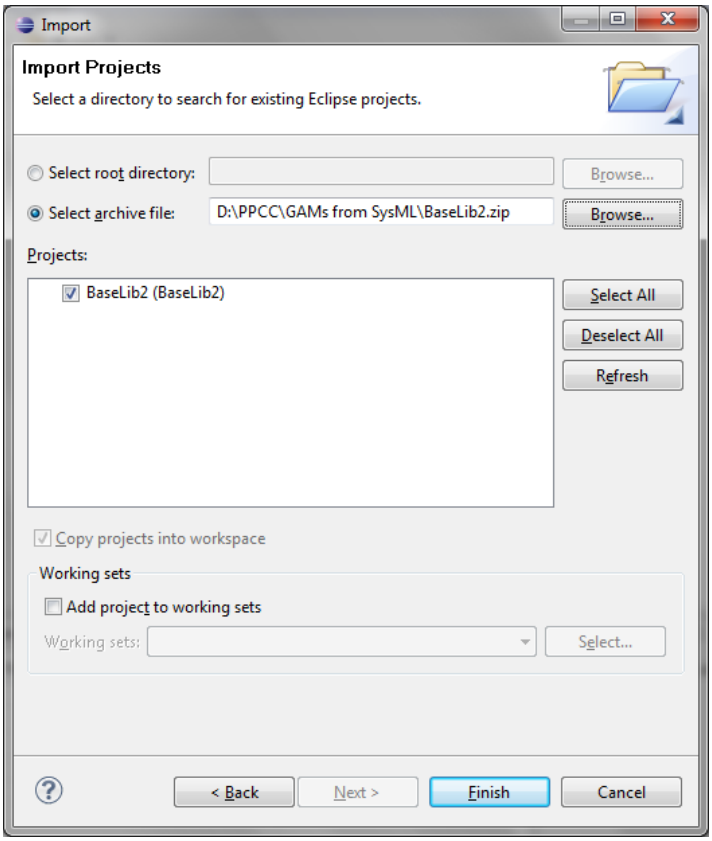

Figure 2 – Import Existing Projects into Workspace.

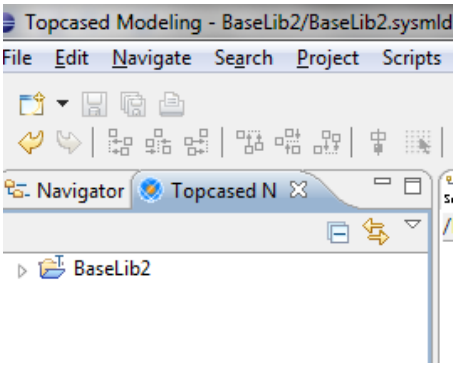

Figure 3 - BaseLib2 folder in the *Topcased Navigator* tree.

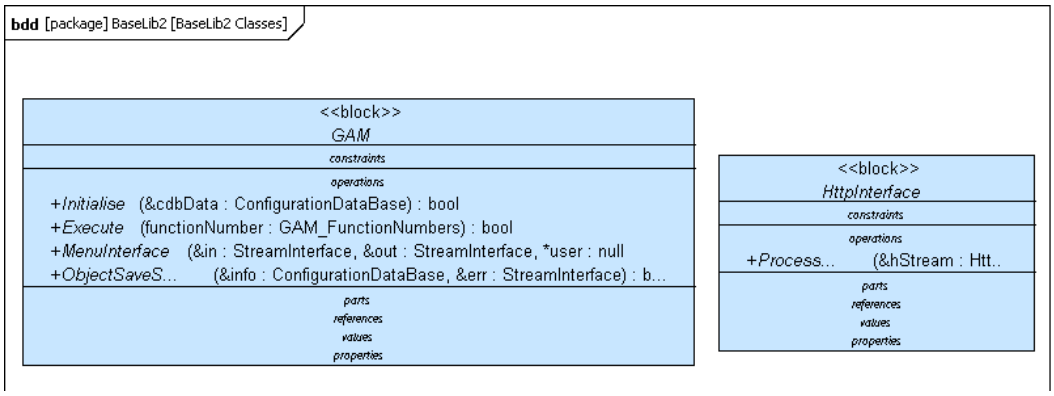

Figure 4 – BaseLib2. Modeled classes.

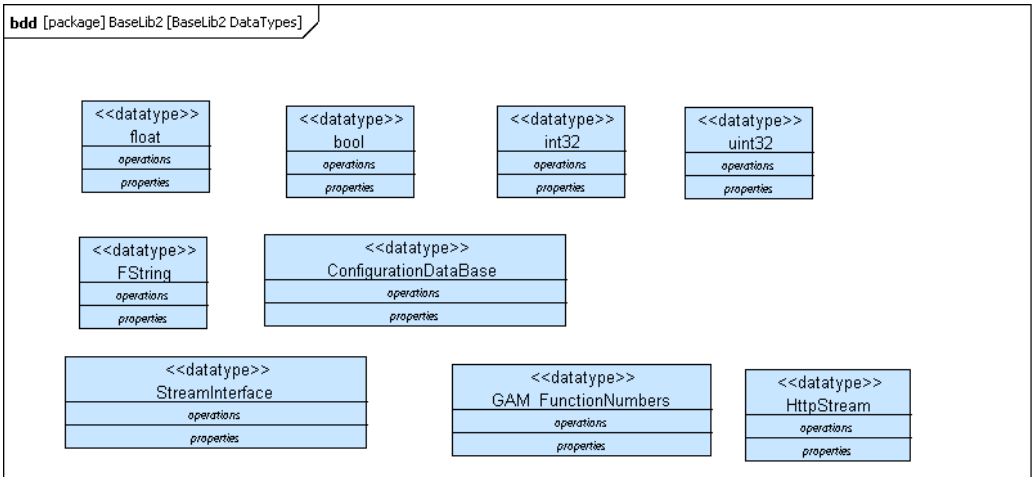

Figure 5 - BaseLib2. Modeled data types.

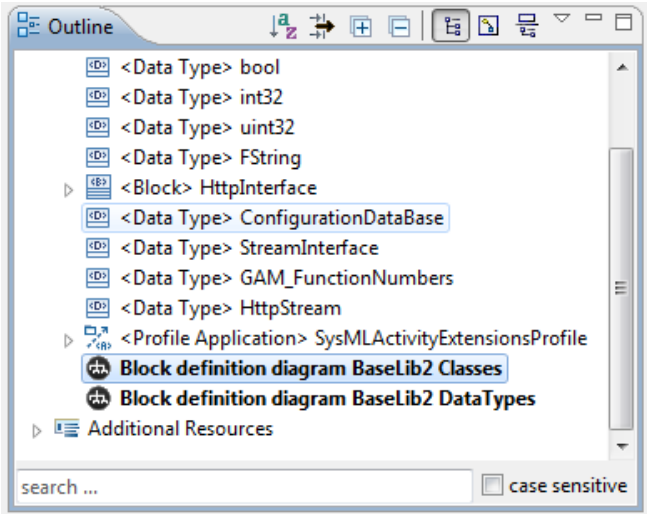

Figure 6 - Block definition diagrams in the *Outline* tree.

3.2 Create a GAM model

This section describes how to model a GAM by using SysML. In order to do that the blocks and the data types defined in the BaseLib2 model previously introduced will be needed. In particular, it will be shown how to build a simple PIDGAM.

1. Launch Topcased;
2. create a new Topcased project by choosing the **File→New→Project…** menu;
3. select the *Topcased Project* wizard (see Figure 7) and press the *Next* button;
4. give the project a name (see Figure 8) and press the *Finish* button;
5. insert a new SysML model in the project by right-clicking on the project folder and selecting **New→SysML Model with TOPCASED** menu (see Figure 9)
6. give the model a name and select *Block definition diagram* as default diagram (see Figure 10);
7. rename the package as *Online* (see Figure 11);
8. import the BaseLib2 model by right-clicking on the model root in the Outline tree and selecting the **Import From Model…** menu (see Figure 12);
9. select the BaseLib2 model and check the *Import by reference (diagrams are not imported)* option (see Figure 13) then press the *Ok* button;
10. drag'n'drop the imported *GAM* block from the *Outline* tree to the bdd of the example (see Figure 14);
11. drag'n'drop the imported *HttpInterface* block from the *Outline* tree to the bdd of the example;
12. add a new block to the model and name it *PIDGAM* (to add a new block to the model add it in the bdd as shown in Figure 15);
13. connect the PIDGAM block with both the GAM and the HttpInterface blocks using a *Generalization* connector (see Figure 15);
14. to add parameters to the PIDGAM, drag'n' drop a *Property* object into the PIDGAM block and assign it a name and a type by using the *Properties* tab (see Figure 16);

15. it is possible (but it is not mandatory) to add a comment to a parameter by right-clicking on the desired property in the *Outline* tree and selecting the **Create Child→Owned Comment→Comment** menu (see Figure 17);
16. the comment text must be specified in the *body* field of the Comment object (see Figure 18);
17. it also possible (but it is not mandatory) to specify an initial value for a *Property* object by defining the *Default Value* in the **Properties→Specification** tab as an Expression (see Figure 19 and Figure 20);
18. to add a method to the PIDGAM, drag'n'drop an *Operation* object into the PIDGAM block and assign it a name, a return type and a Visibility by using the *Properties* tab (see Figure 21);
19. to add input and output signals to the PIDGAM double click on the PIDGAM block and add an *Internal Block Diagram* (ibd, see Figure 22);
20. to add an input signal to the PIDGAM, drag'n'drop an *In Flow Port* object into the ibd and assign it a name, a data type by using the *Properties* tab (see Figure 23);
21. to add an input signal to the PIDGAM, drag'n'drop an *Out Flow Port* object into the ibd and assign it a name, a data type by using the *Properties* tab.

Note that more than one GAM can be modeled in a single SysML model by adding other block objects that inherit from the GAM block.

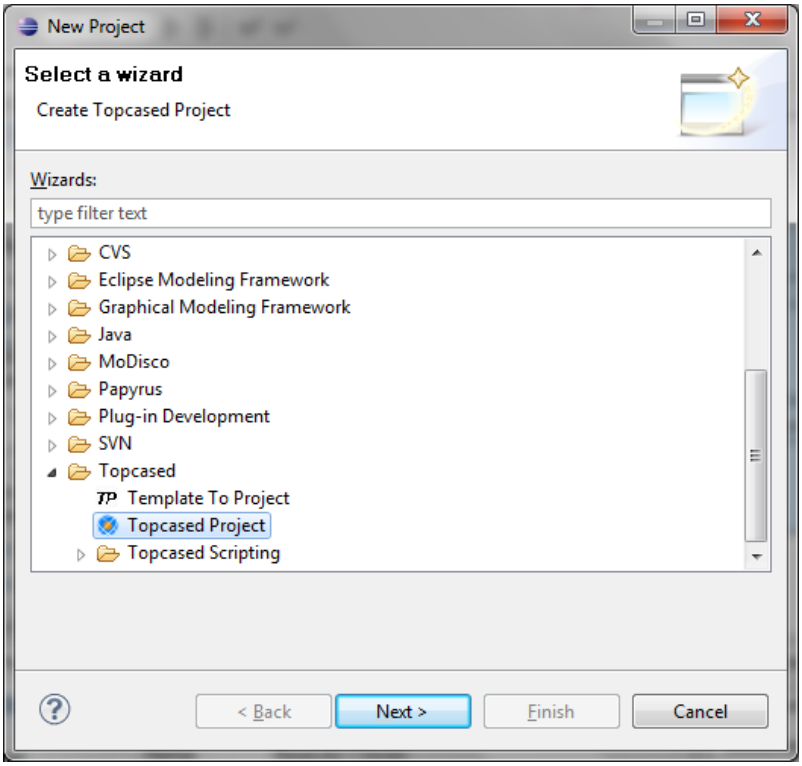

Figure 7 - New Project dialog window.

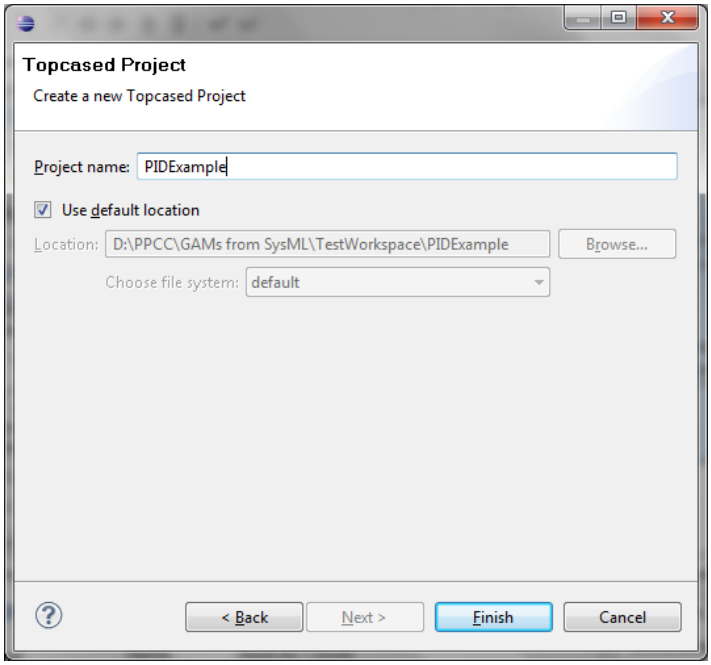

Figure 8 - Topcased Project wizard.

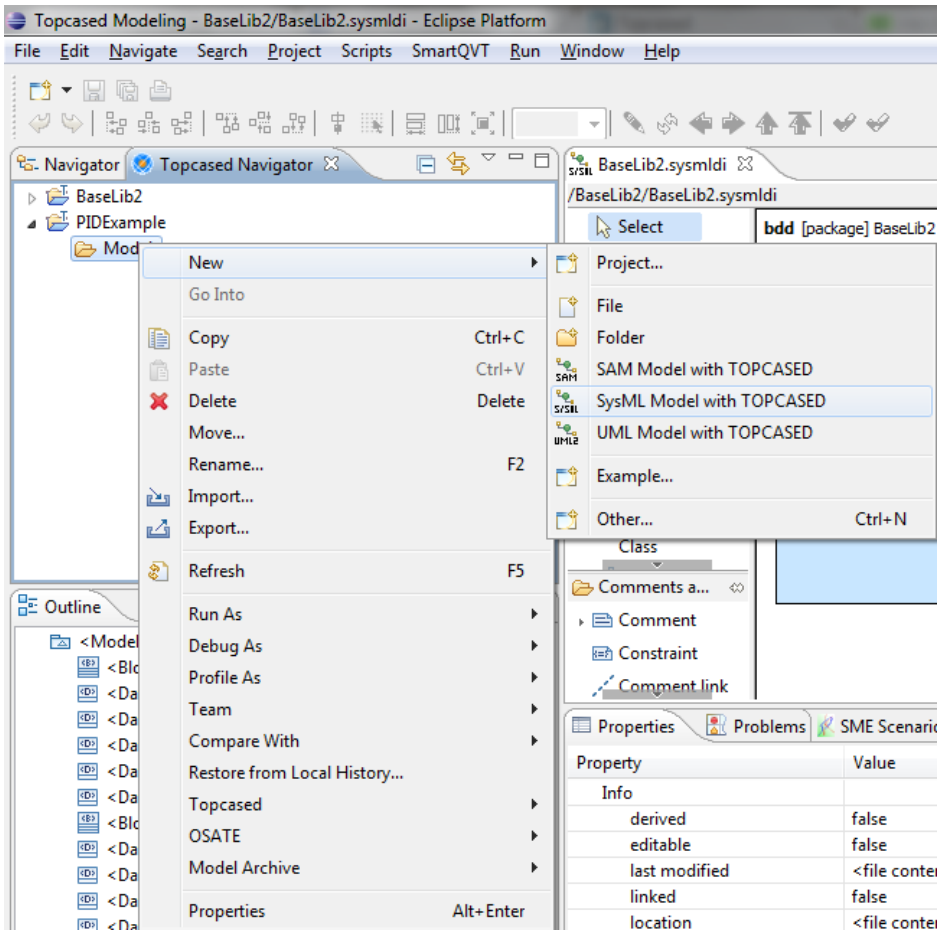

Figure 9 - Insert a new SysML model in the Topcased project.

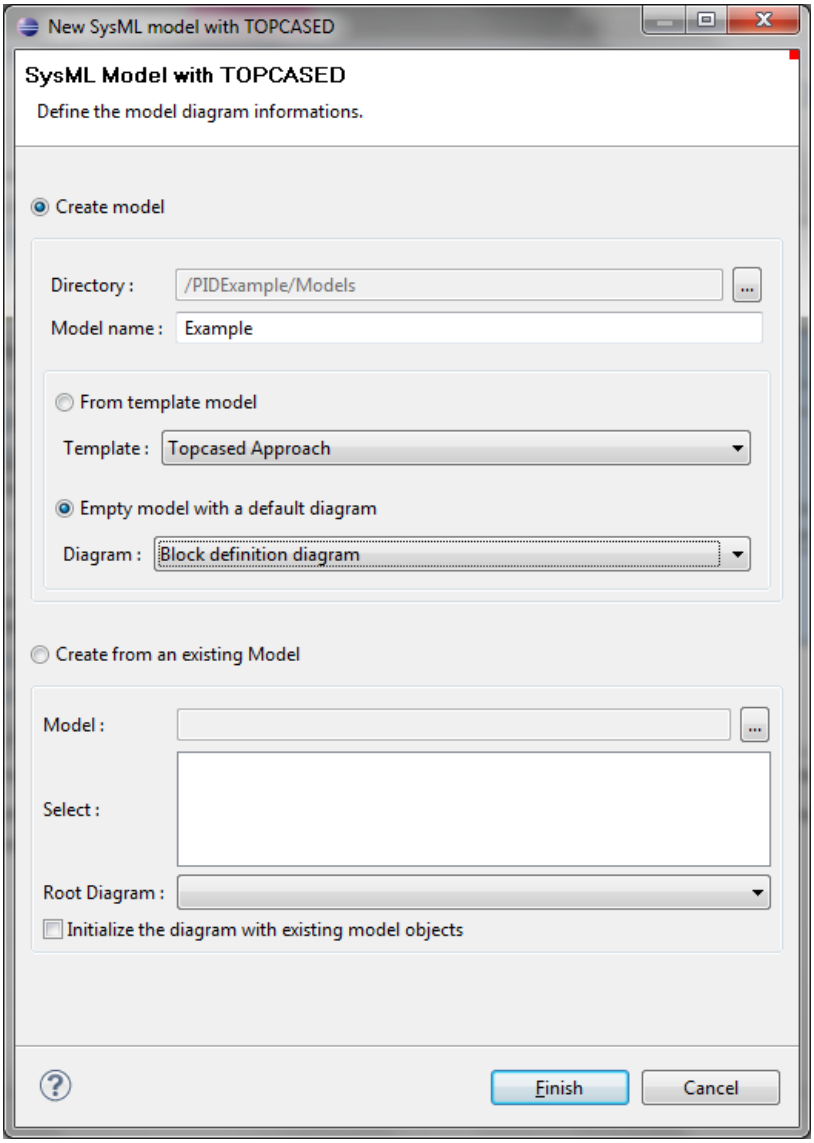

Figure 10 – New SysML Model with TOPCASED.

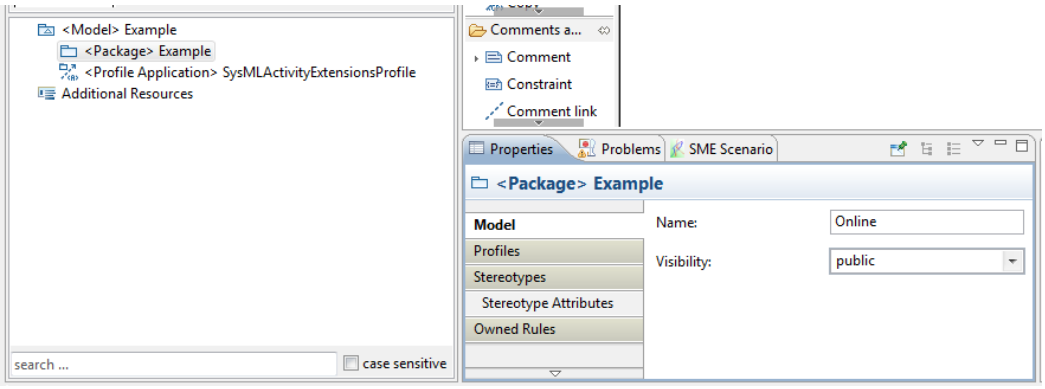

Figure 11 - Rename the package as *Online*.

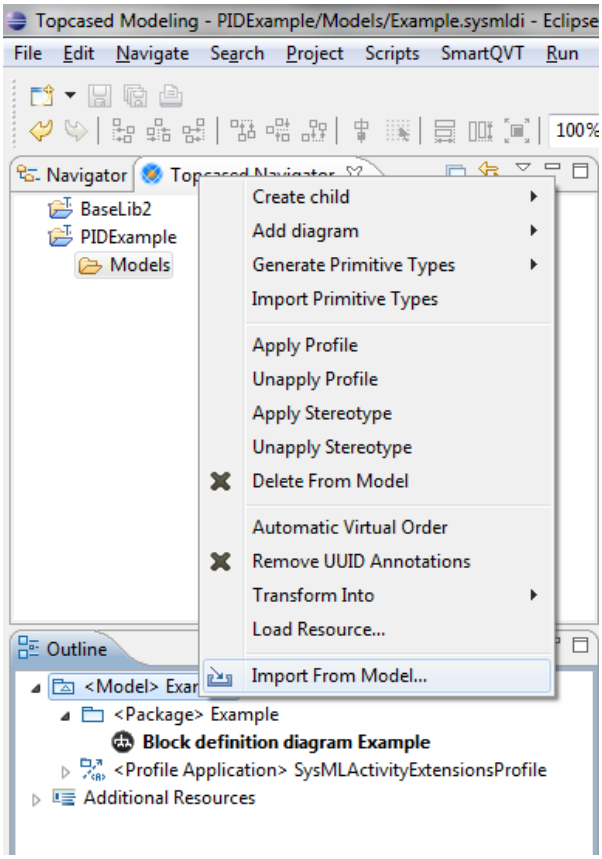

Figure 12 - Import BaseLib2 model in the example.

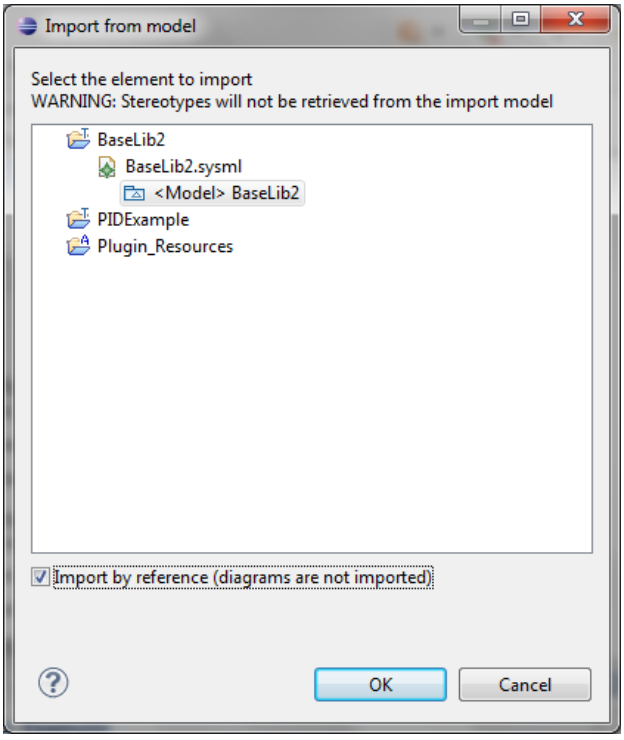

Figure 13 - Import BaseLib2 by reference without diagrams.

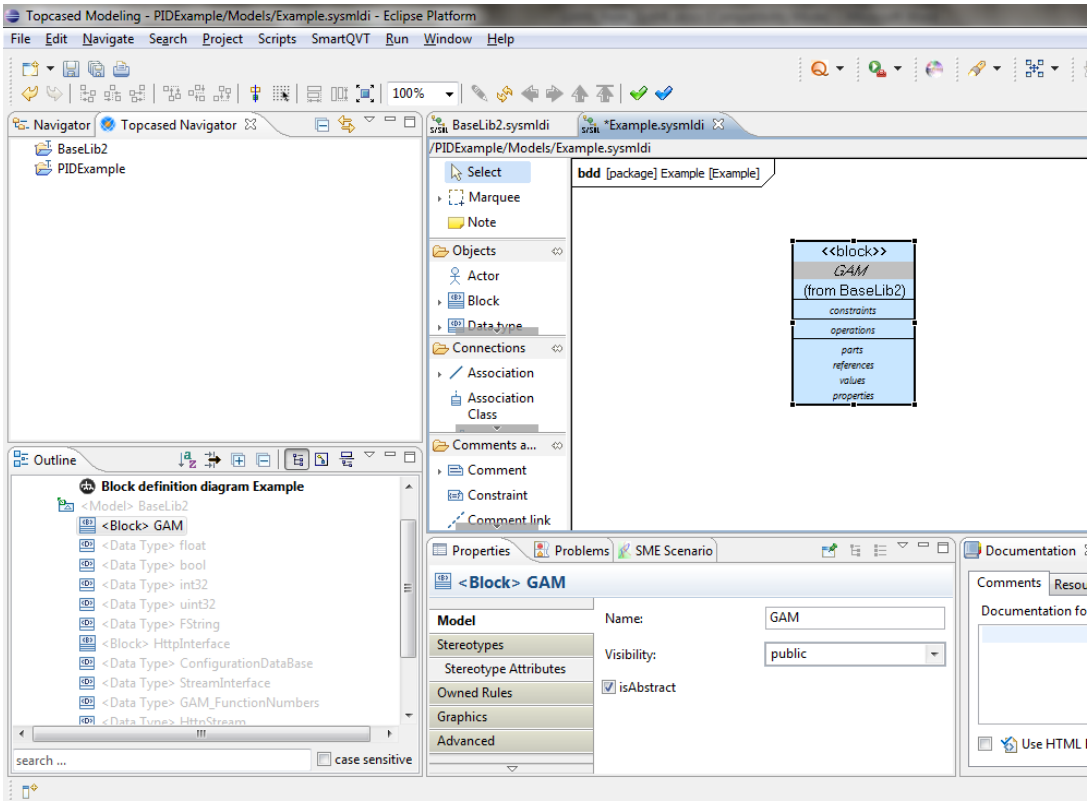

Figure 14 - Add the GAM block to the bdd.

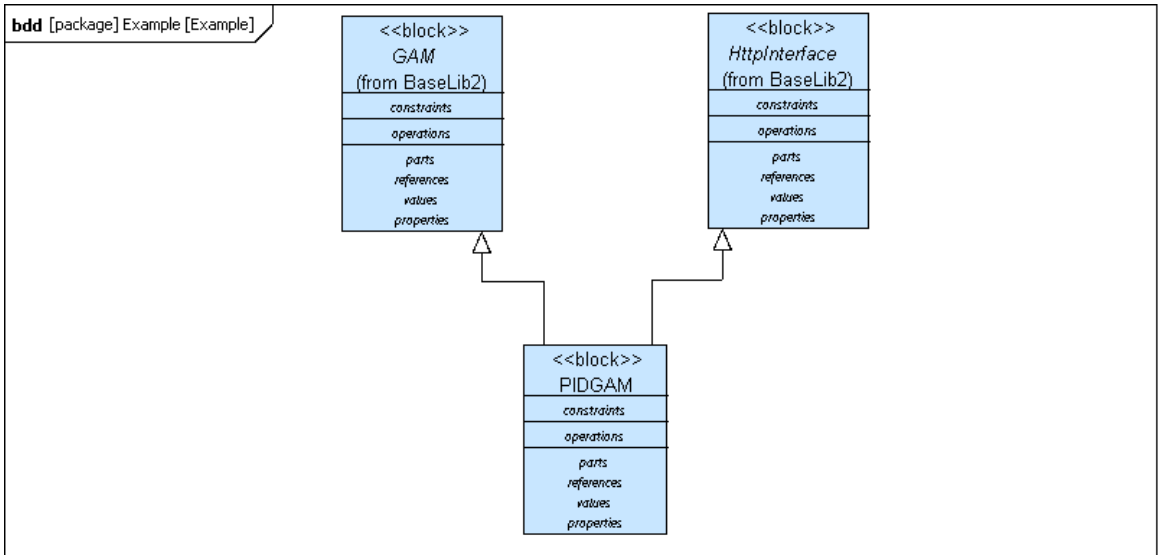

Figure 15 - PIDGAM block. This block inherits from both GAM and HttpInterface.

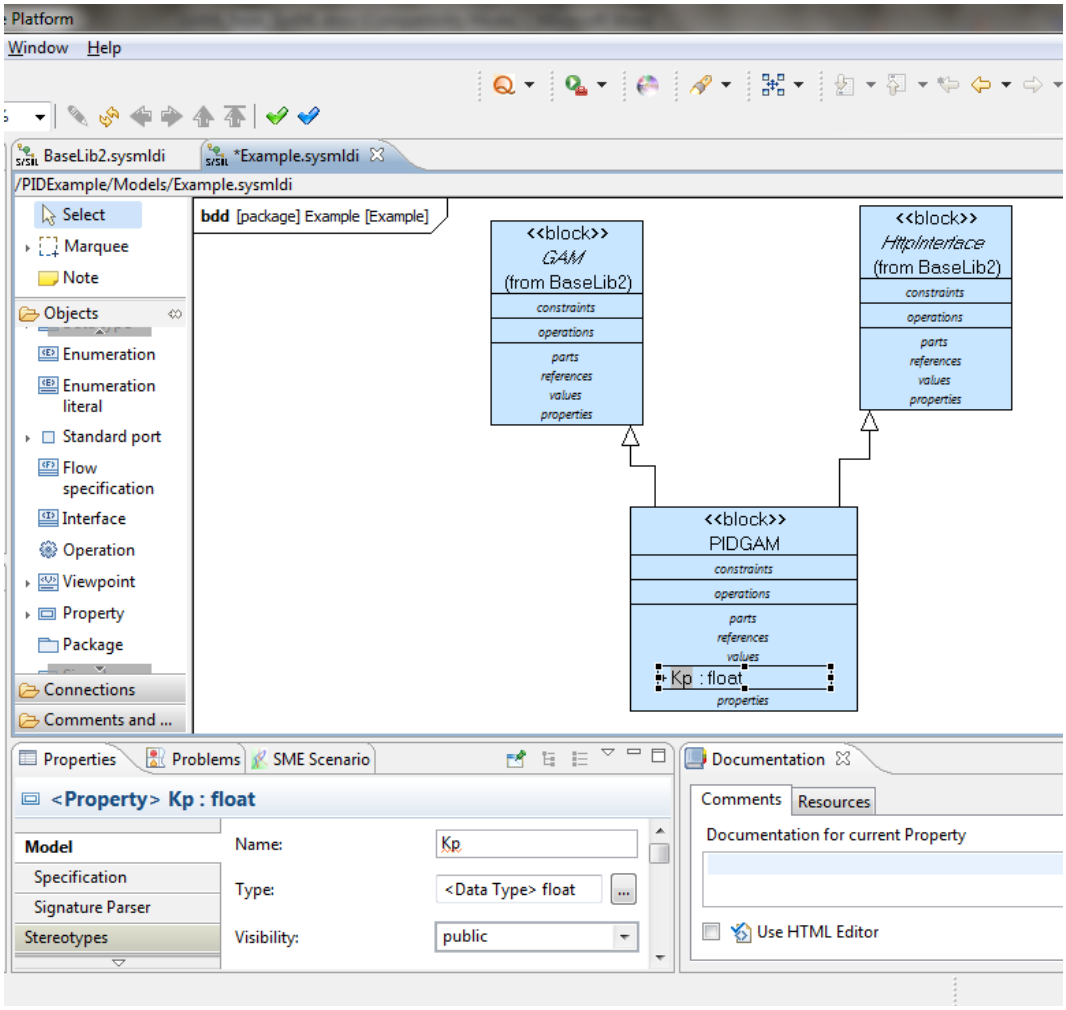

Figure 16 - Add parameters to PIDGAM.

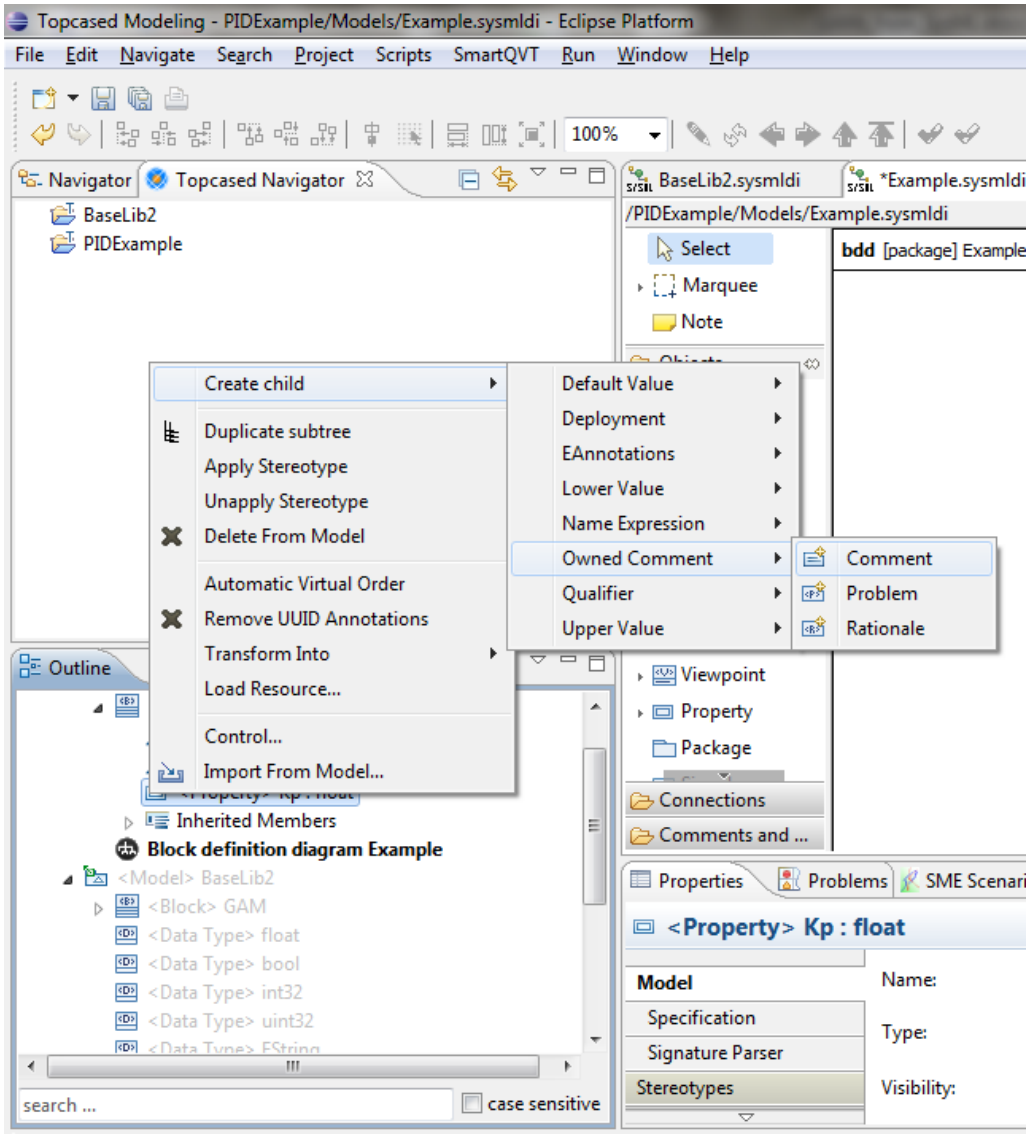

Figure 17 - Add a comment to a parameter.

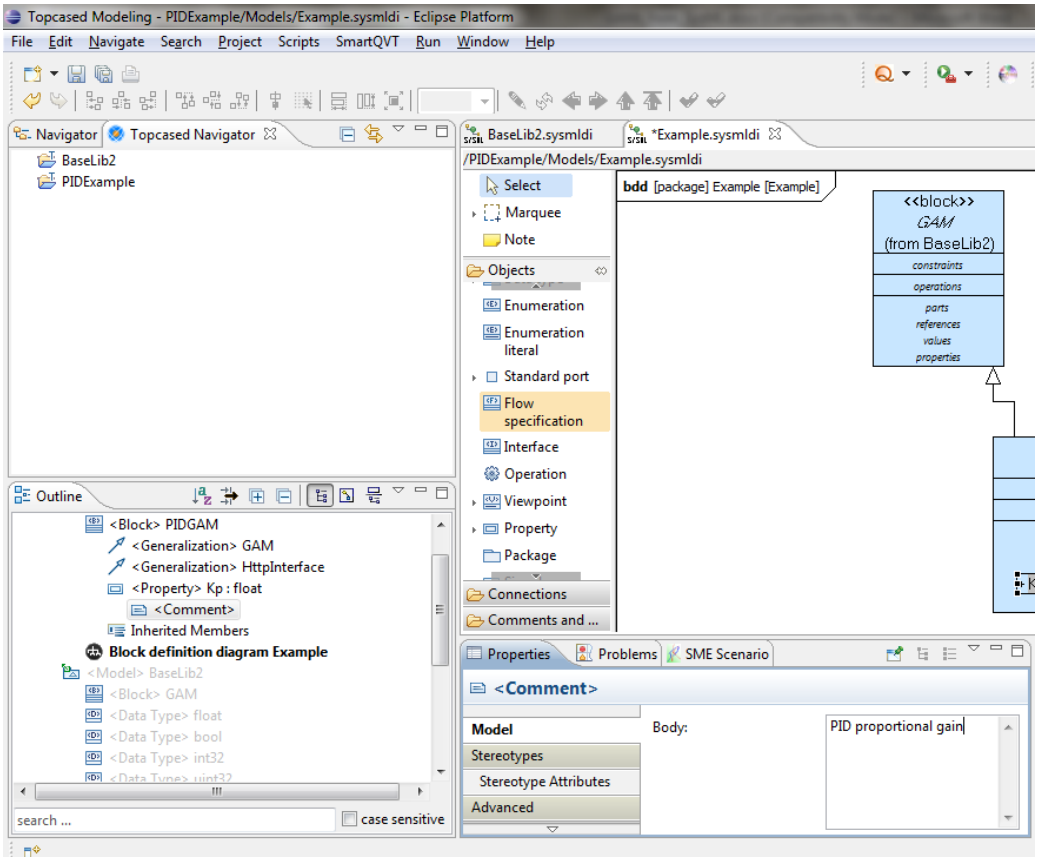

Figure 18 - Comment body.

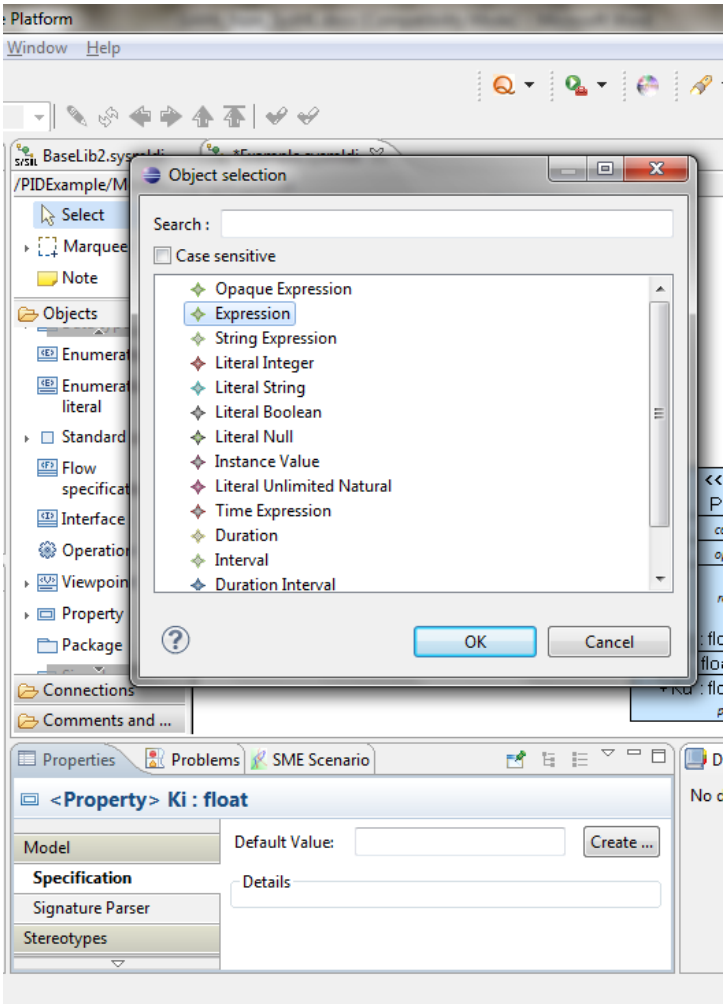

Figure 19 - Add default value as an Expression.

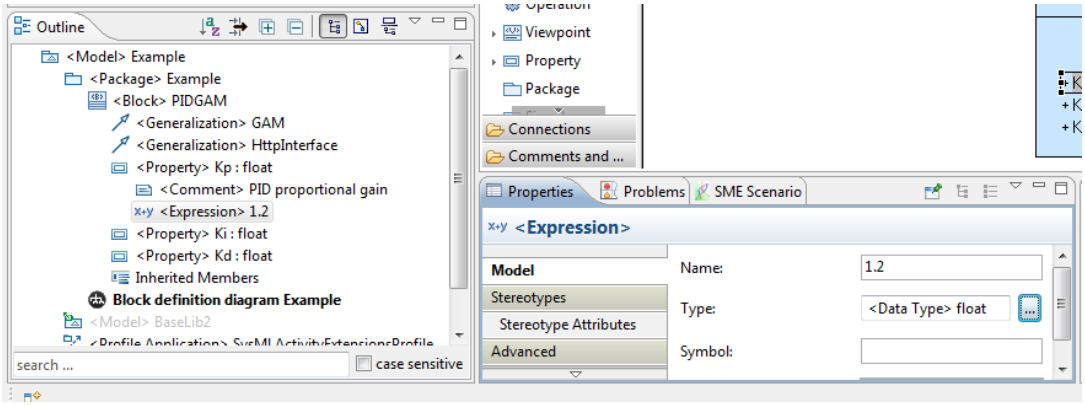

Figure 20 - Example of Expression object used as Default Value.

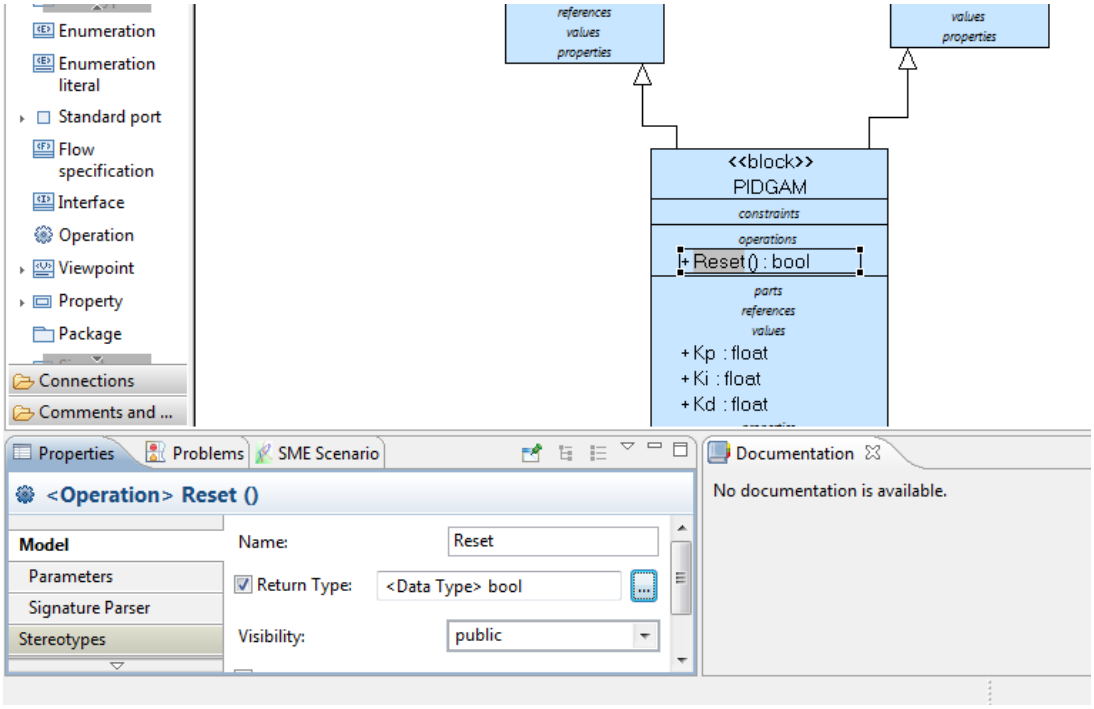

Figure 21 - Add the *Reset()* method to the PIDGAM.

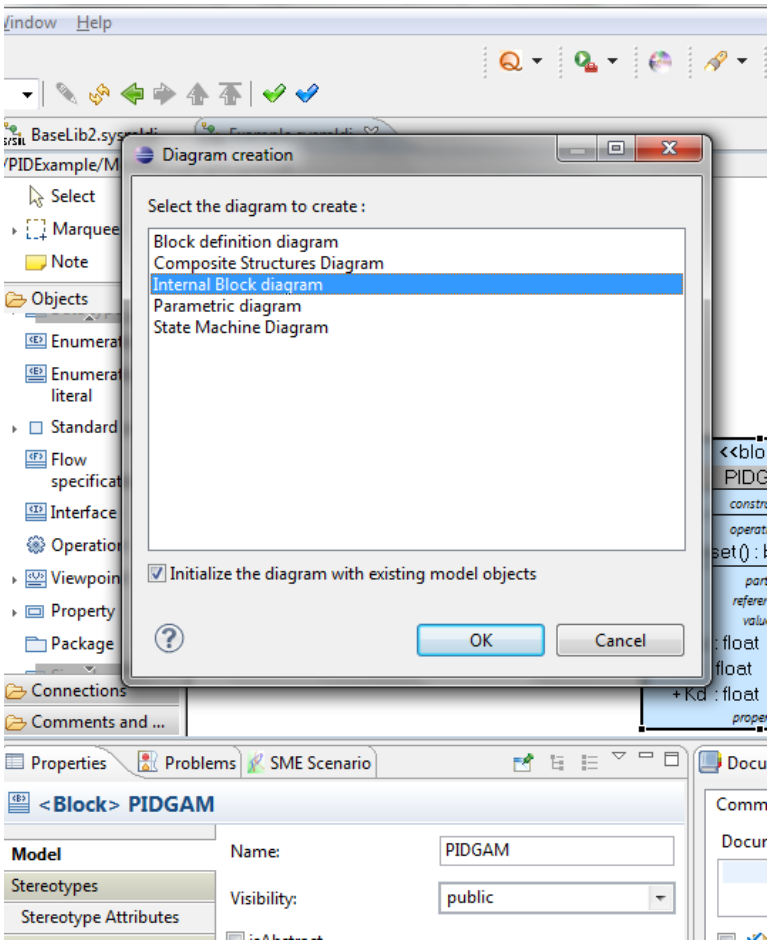

Figure 22 - Addan ibd to define input and output signals.

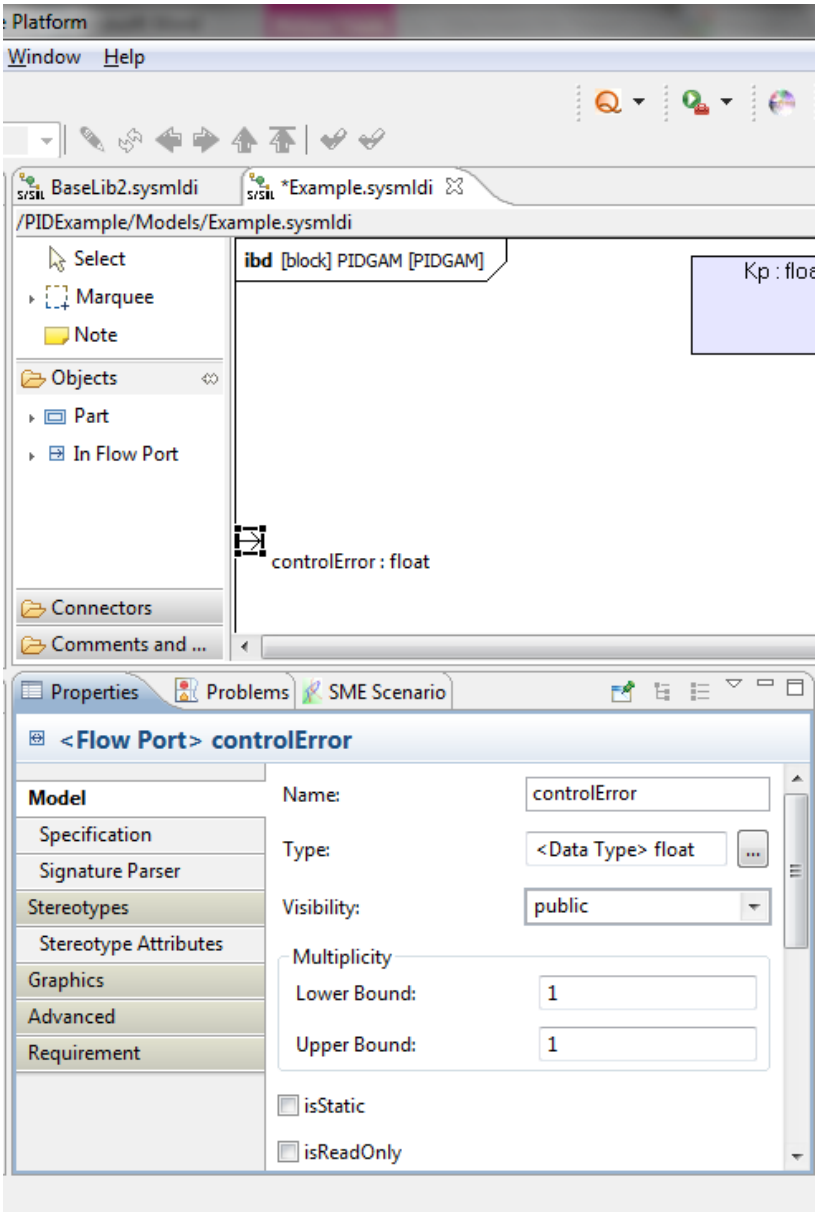

Figure 23 - Add an input signal in the PIDGAM ibd.

3.3 Connecting GAM blocks

TBD.

3.4 Example

An example of GAM modeling with Topcased and SysML can be downloaded from

http://wpage.unina.it/detommas/MARTe-Downloads/SysML/ModelingExample.zip

To import it in Topcased, follow the same procedure described in Section 3.1 to import the BaseLib2 model.

4. AUTOMATIC GAMs CODE GENERATION

This chapter describes how to automatically generate parts of the *.h* and *.cpp* files, together with parts of the configuration file, for a GAM modeled as described in Chapter 3.

4.1 Run the Acceleo plugin

To automatically generate the GAM code the Acceleo plugin installed in Chapter 2 has to be run. In order to do that:

1. select the Topcased project *.sysml* file (see Figure 24)
2. select the **Acceleo Model To Text→Generate MARTe System** menu as shown in Figure 24;
3. the generated file are placed in the *src-gen* folder (see Figure 25). In particular, the current version of the Acceleo plugin produces:
    a. in the *Modules* folder, for each GAM:
        i. the GAM files (.h, .cpp, and .def for Windows platforms);
        ii. the Makefiles (for the Linux and Windows platforms);
        iii. the input and output data structures (.h files);
        iv. the class info file (.h);
        v. the GAM configuration (.cfg file);
    b. in the *config* folder:
        i. the MARTe configuration file (config.cfg file).

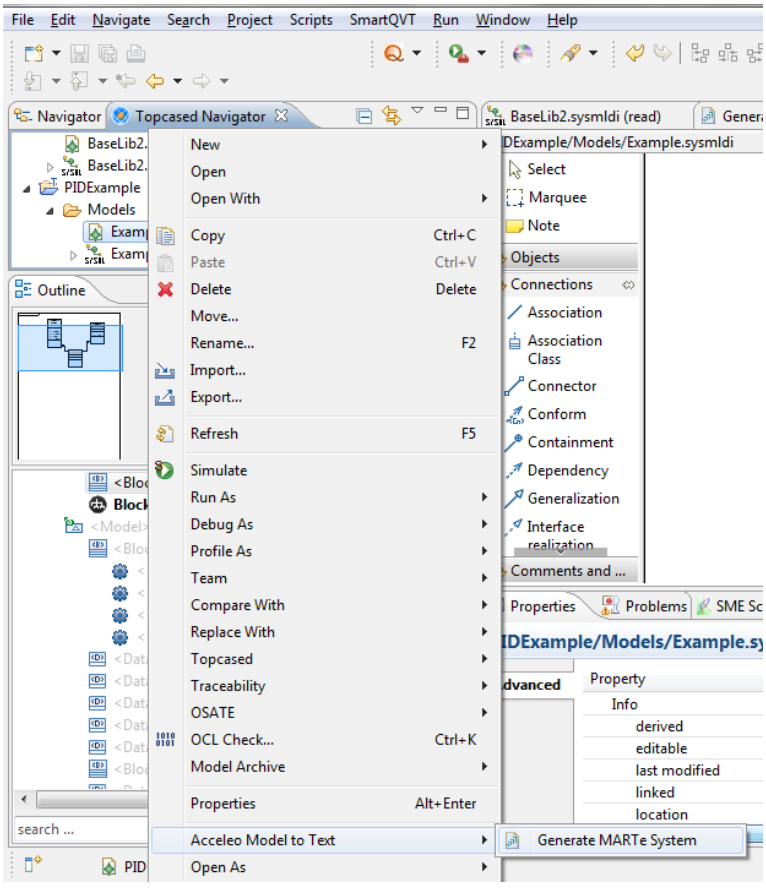

Figure 24 - Run the Acceleo plugin.

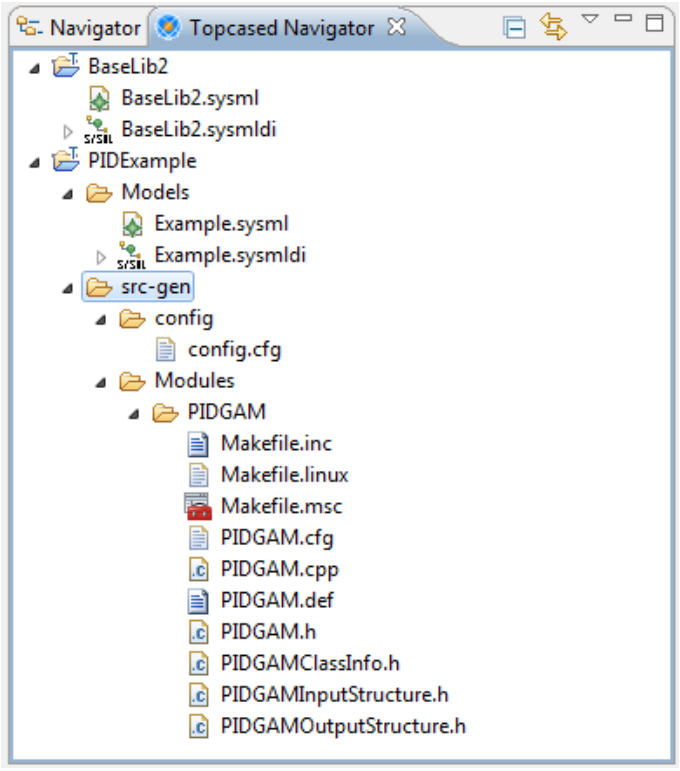

Figure 25 - Automatic generated files.

5. THE ACCELEO PROJECT

This chapter describes how to import the Acceleo project for the *Generate MARTe System* plugin in Topcased.

5.1 Open the Acceleo project

To open the Acceleo project:

1. download the project from

   http://wpage.unina.it/detommas/MARTe-Downloads/SysML/AcceleoProject.zip

2. import the Acceleo project in Topcased following the same procedure described in Section 3.1 to import the BaseLib2 model;
3. change the eclipse prespective by choosing the **Window→Open Prespective→Other…** menu and by selecting the Acceleo prespective (see Figure 26);
4. the Acceleo source code is in the G*enerateGAMs.mtl* file.

More details about how to use Acceleo and run it can be found at

http://www.eclipse.org/acceleo/documentation/

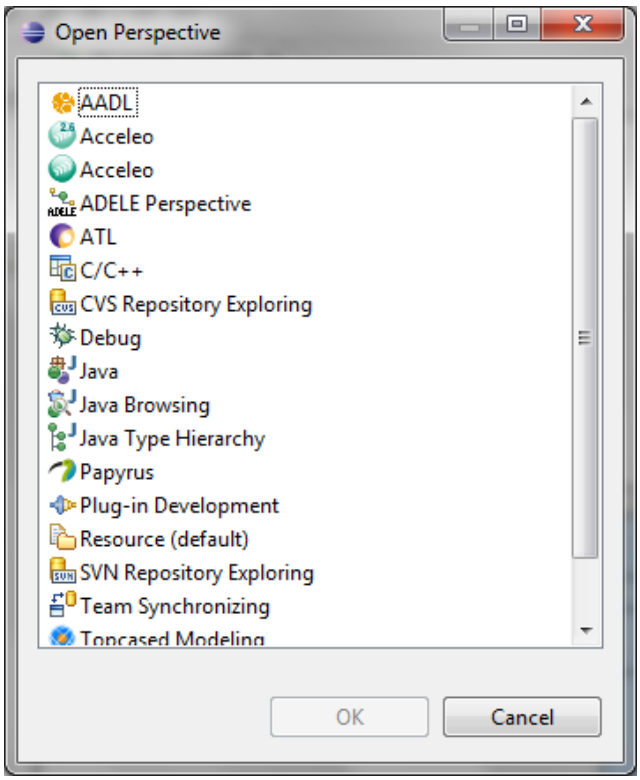

Figure 26 - Open Perspective dialog window.