

AMW Use case Variability Management in a Model-Driven Software Product Line	Kelly Garcés kellygarce@gmail.com Software Construction Group University of Los Andes Bogotá – Colombia
User Guide	07/08/2007

Installation

This use case requires Eclipse 3.2.1+ EMF 2.2.1 + ATL + AMW + Acceleo 1.2

Download ATL: <http://www.eclipse.org/m2m/atl/download/>

Download AMW: <http://www.eclipse.org/gmt/amw/download/>

Download Acceleo: <http://www.acceleo.org>

Download and unzip the use case example. It contains ATL and Acceleo project.

The ATL project structure should look like this:

LineaCupi2/

- **Arquitectura**
 - metamodelo
 - negocio.ecore Business metamodel
 - interface.ecore Graphical user interface metamodel
 - modelo
 - expauto_interfaz.xmi Auto Show interface model
 - expauto_negocio.xmi Auto Show business model
- **ATL**
 - arquitectura2java.atl Architecture to java transformation
 - arquitectura2java.launch
 - mundo2interfaz.atl Business logic to interface transformation
 - mundo2interfaz.launch
 - mundo2Negocio.atl Business logic to business transformation
 - mundo2negocio.launch
- **Features**
 - metamodelo
 - mmFeature.ecore Feature metamodel
 - modelo
 - featuresinterfazcupi2.xmi Cupi2 interface features
 - featuresjavacupi2.xmi Cupi2 java features
 - featuresnegociocupi2.xmi Cupi2 business features
- **Java**
 - metamodelo
 - java.ecore Java metamodel
 - modelo
 - expauto_java.xmi Auto Show java model
- **Mundo**
 - metamodelo
 - mundo.ecore Business logic metamodel
 - modelo
 - expauto.xmi Auto Show business logic model

- **Weaving**

- metamodelo

- mw_base_ext.ecore Weaving metamodel

- modelo

- expauto_intWfeaturesjava.amw Weaving model between expauto_interfaz.xmi and featuresjavacupi2.xmi
 - expauto_negWfeaturesjava.amw Weaving model between expauto_negocio.xmi and featuresjavacupi2.xmi
 - expautoWfeaturesinterfaz.amw Weaving model between expauto.xmi and featuresinterfazcupi2.xmi
 - expautoWfeaturesnegocio.amw Weaving model between expauto.xmi and featuresnegociocupi2.xmi

The Acceleo project structure should look like this:

GeneradorLineaCupi2/

- src: contains the Acceleo templates
- Gen: contains the generated code
- mdsplGenerador.chain: Acceleo configuration file

These projects contain the artefacts needed to generate Cupi2 examples.

Execute the Use Case

Now we explain the process to generate a Cupi2 example. The following steps are necessary:

1. Create business logic model.
2. Weave models and execute model-to-model transformations.
3. Execute model-to-text transformation.

We will generate code for the Auto Show application¹ following these steps.

1. Create business logic model

The first step is to create the business logic model. Business logic model represents the problem description. Figure 1 shows a business logic model (expauto.xmi). In this model, the *ExposicionAutomovil* element (conforms to *Agrupador*) groups *Automovil* element (conforms to *Simple*). *Automovil* element contains a set of *Atributo* and *AtributoCupi2* elements.

¹ Auto Show application manages information relating to automobiles.

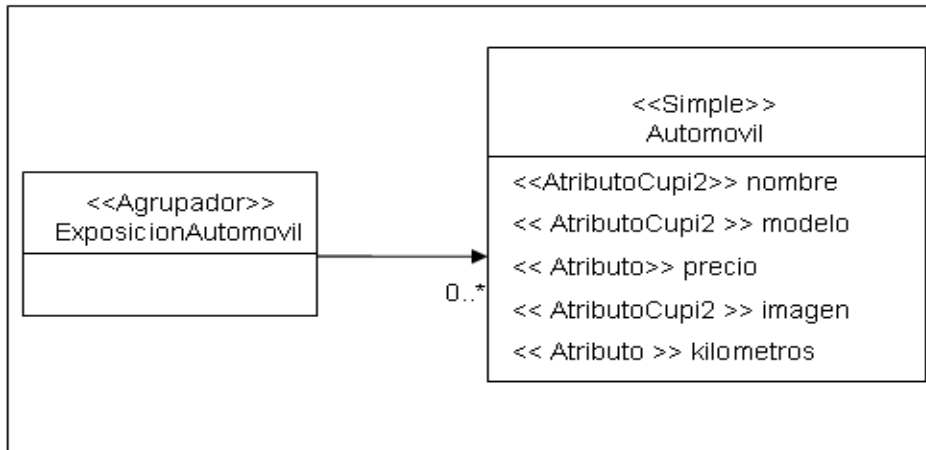


Figure. 1. Model conforms to business logic metamodel

2. Weave models and execute model-to-model transformations

We should create weaving models and execute transformations to generate a Cupi2 example. First, we create a weaving model between (expautoWfeaturesinterfaz.amw) the business logic model (expauto.xmi) and the interface features (featuresinterfazcupi2.xmi). Later, we create a weaving model (expautoWfeaturesnegocio.amw) between the business logic model (expauto.xmi) and the business features (featuresnegociocupi2.xmi). We execute mundo2interfaz and mundo2negocio transformation. Afterwards, we weave the generated interface model (expauto_interfaz.xmi) and the java features (featuresjavacupi2.xmi). We name the weaving model expauto_intWfeaturesjava.amw. We weave the generated business model (expauto_negocio.xmi) and the java features (featuresjavacupi2.xmi). We name the weaving model expauto_negWfeaturesjava.amw. Finally, we execute arquitectura2java transformation.

To create a weaving model, we use the Weaving Model Wizard. First, we select the weaving metamodel (mw_base_ext.ecore). Later, we enter a name for the weaving model, for example expautoWfeaturesinterfaz.amw. Finally, we select the left (e.g. expauto.xmi) and right (e.g. featuresinterfazcupi2.xmi) model.

Figure 2 shows expautoWfeaturesinterfaz.amw. It contains links between the business logic model elements and the interface features. For instance, one link associates *Automovil* element and *VistaConjunto* feature. It generates a Set View which groups automobiles by using a combo box.

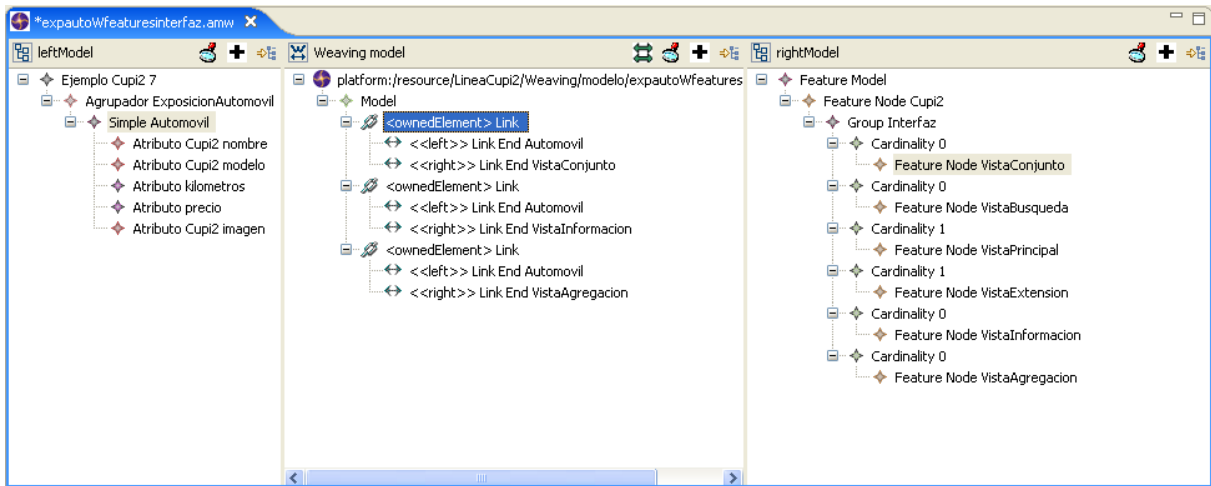


Figure 2. Weaving model example

To create a link, we make a right click on the Model element on middle panel. We choose New Child -> Link. We select the Automovile element on the left and drag and drop it to the link. We see a menu appearing, select <<left>> LinkEnd. Later, we select the VistaInformacion element on the right and drop it to the link. We see a menu appearing, select <<right>> LinkEnd.

To execute the transformations, we use the launch configurations. Figure 3 shows the mundo2interfaz launch configuration.

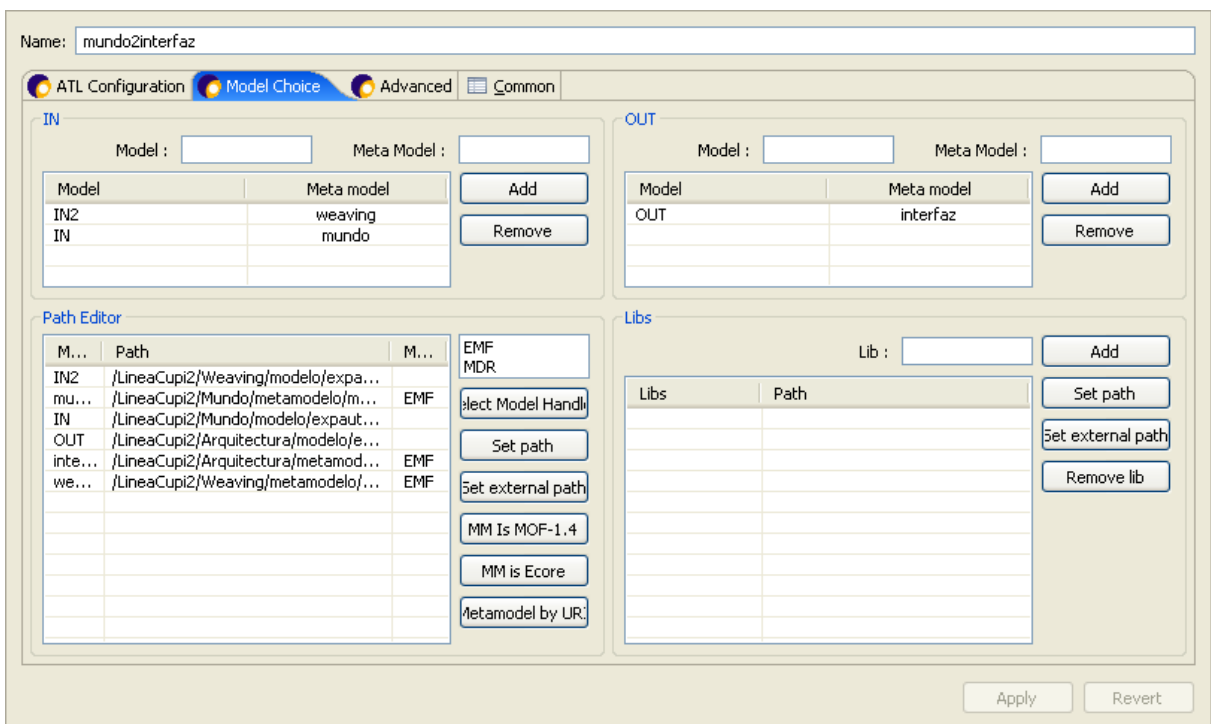


Figure 3. mundo2interfaz launch configuration

3. Execute model-to-text transformation

When the last transformation (arquitectura2java) is executed, we obtain a java model. Acceleo generates source code based on templates and this java model. To generate the source code, we make right click on mdsplGenerator.chain on GeneratorLineaCupi2 project. We choose Launch option. Later, Acceleo engine generates the source code into the Gen directory. At last, we execute the generated code. We make right click on InterfazExposicionAutomovil.java on GeneratorLineaCupi2/Gen/interfaz, we select Run As -> Java Application (see Figure 5). We can see the Auto Show GUI (Figure 6).

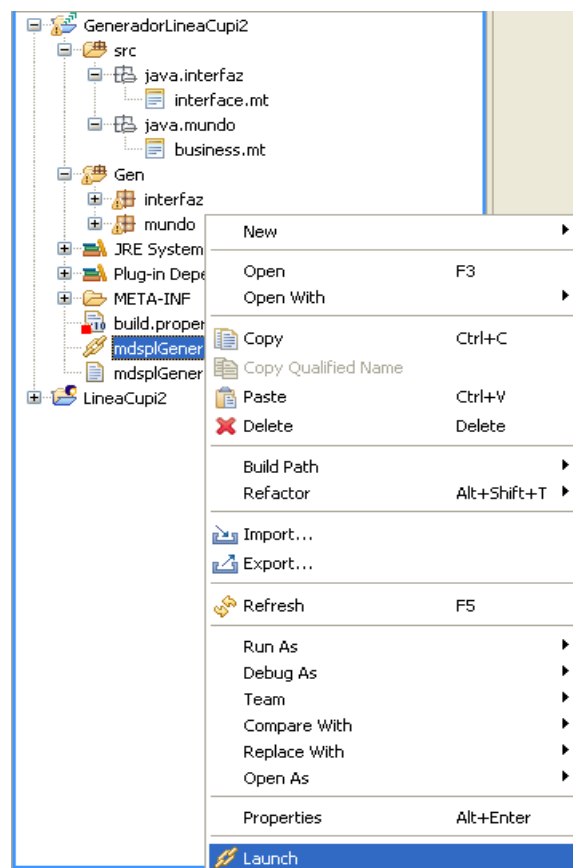


Figure 4. Source code generation

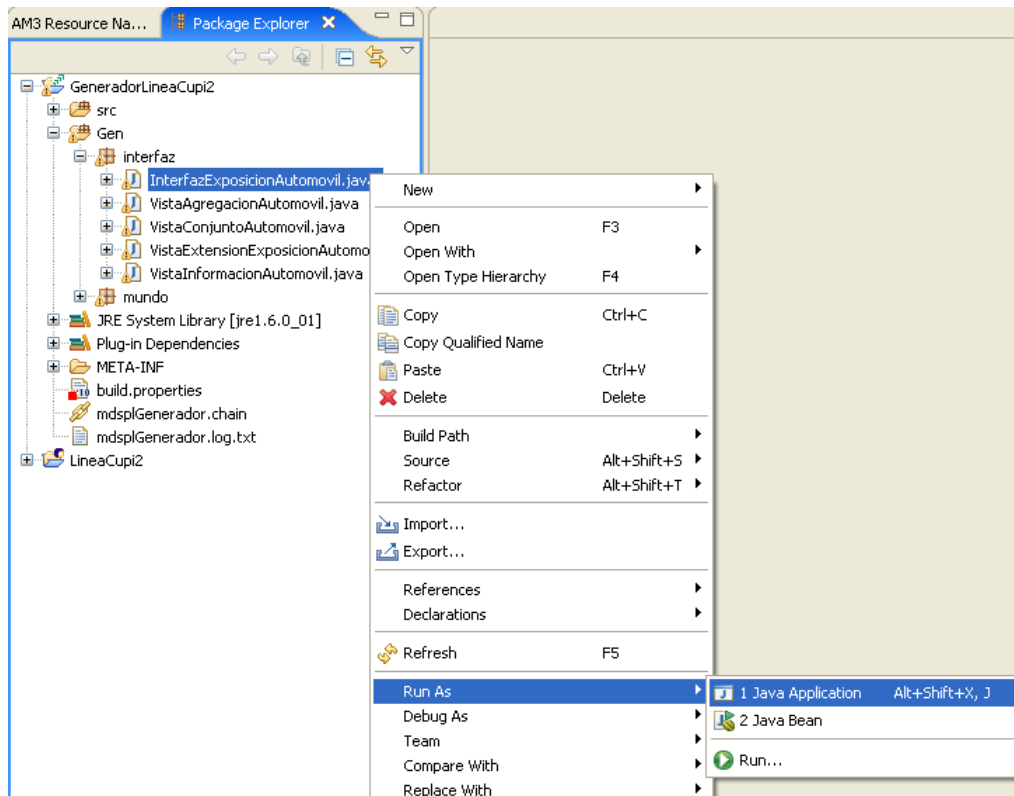


Figure 5. Auto show execution

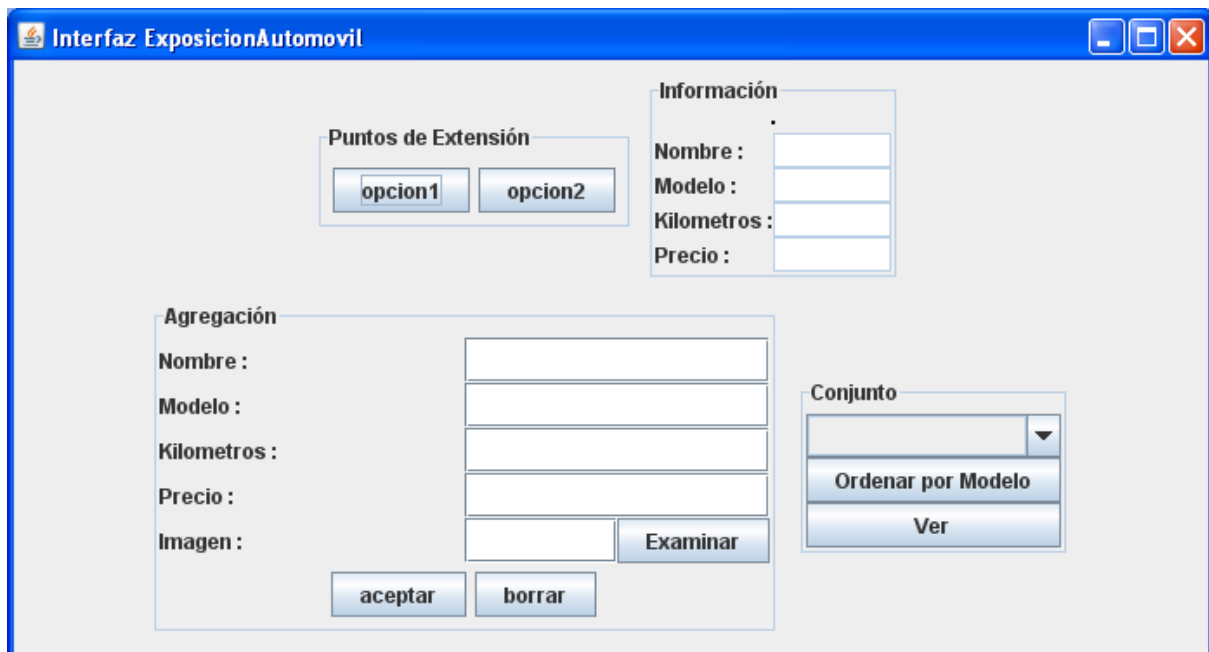


Figure 6. Auto Show GUI

Generate another application

We can generate another application by changing the weaving models. For example, we want to change the Auto Show GUI. Now we prefer to use JList to group the automobiles. In order to achieve it, we should edit `expauto_intWfeaturesjava.amw`. We select the single Link element on middle panel, we change the JComboBox feature (`<<right>> LinkEnd`) for the JList feature. We re-execute `arquitectura2java` transformation. We follow the indications on step 3 (Execute model-to-text transformation). When we finish the process, we obtain the Auto Show GUI displayed in the figure 7.

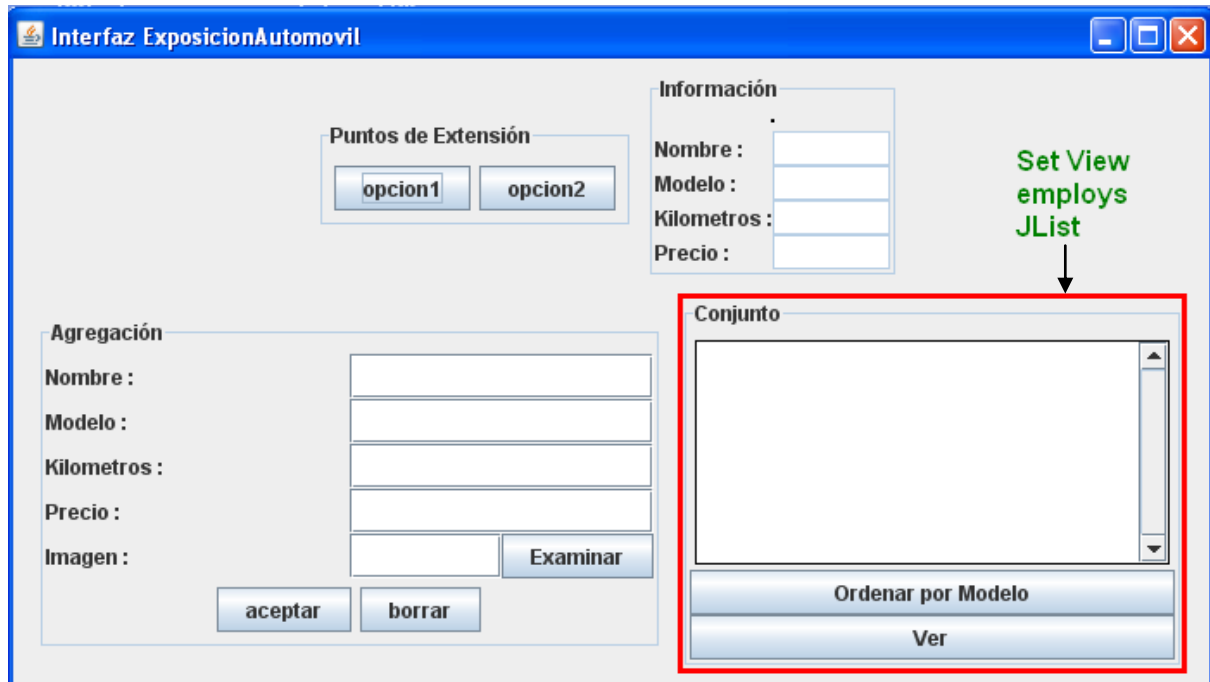


Figure 7. Using JList in Auto Show GUI