

Modif Documentation

-

Reuse User Guide

This document explains the steps for using ModifRoundtrip for Reuse

June, 2015

by Paola Vallejo, Jean Philippe Babau

Table of contents

1. Create project, folders and ecore metamodel.....	3
1.1. Create a project.....	3
1.2. Add Xtext nature.....	3
1.3. Add Folders.....	3
1.4. Create metamodel.....	3
1.5. Create model.....	6
2. Import the Tool project.....	9
3. Execute Modif Roundtrip.....	10
1. Refactoring.....	10
Specify Domain Metamodel and Generate Modif model.....	11
Edit Modif Model.....	12
Check and Refactor.....	13
2. Migration specification generation.....	14
3. Reuse code generation and Migration and Reuse.....	16
4. Reverse Migration and Recontextualization.....	19
4. Contact.....	21

1. Create project, folders and ecore metamodel

1.1. Create a project

Create a new Empty EMF project and, for example name it *Test_StateChart*

File New Other... Eclipse Modeling Framework/Empty EMF Project Next Specify the project name [Test_StateChart] Finish

1.2. Add Xtext nature

Add the Xtext Nature to the Test project

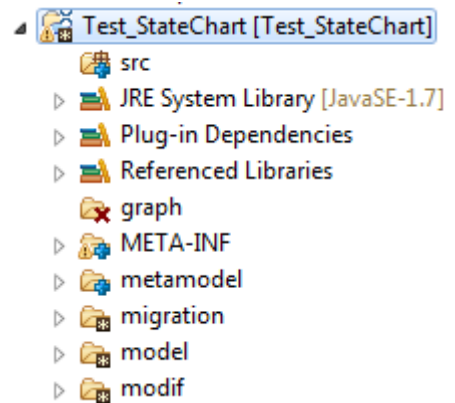
Right click Test_StateChart project Configure Add Xpand/Xtext Nature

1.3. Add Folders

Keep the **model** folder and add four additional folders:

Right click *Test_StateChart* New Folder [Specify the folder_name] Finish

1. graph
2. metamodel
3. migration
4. modif

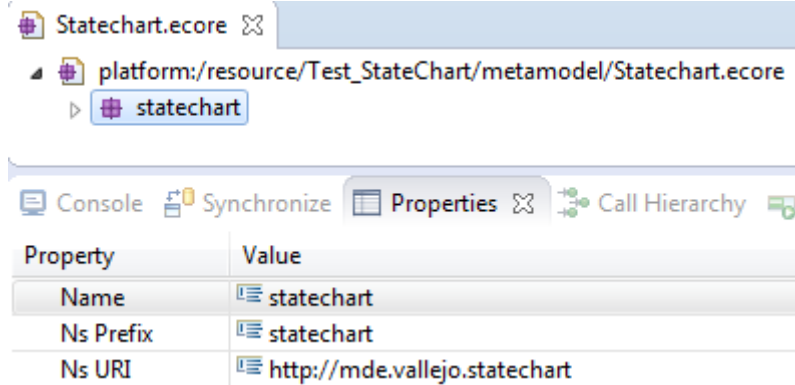


1.4. Create metamodel

In the **metamodel** folder, create a new ecore model. And for the example, name it *Statechart.ecore*

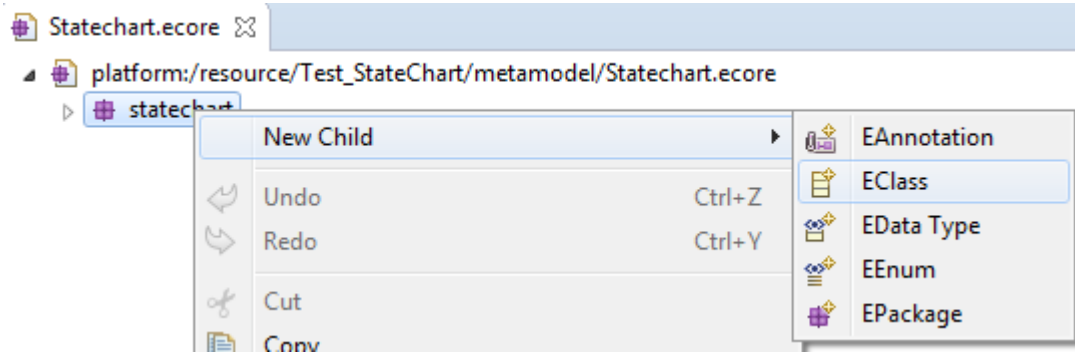
Right click on Test_StateChart/metamodel New Other... Eclipse Modeling Framework / Ecore Model Next [give a name for your ecore] Finish

Then set the ecore model properties as follows:



Add the EClasses to the metamodel.

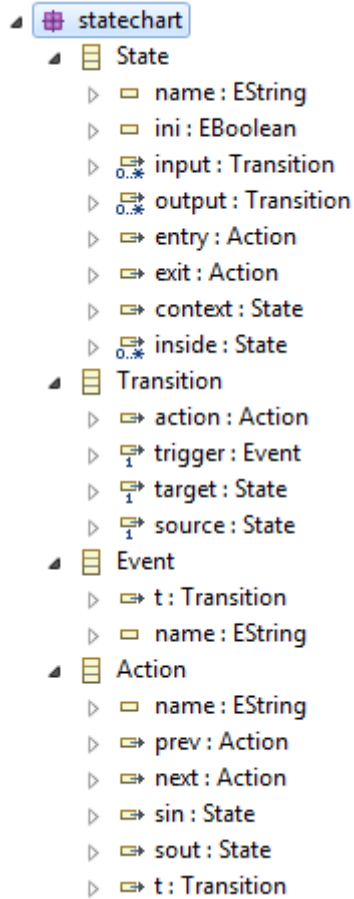
Right click on *statechart* package New Child EClass



Add the EReferences and EAttributes to the metamodel.

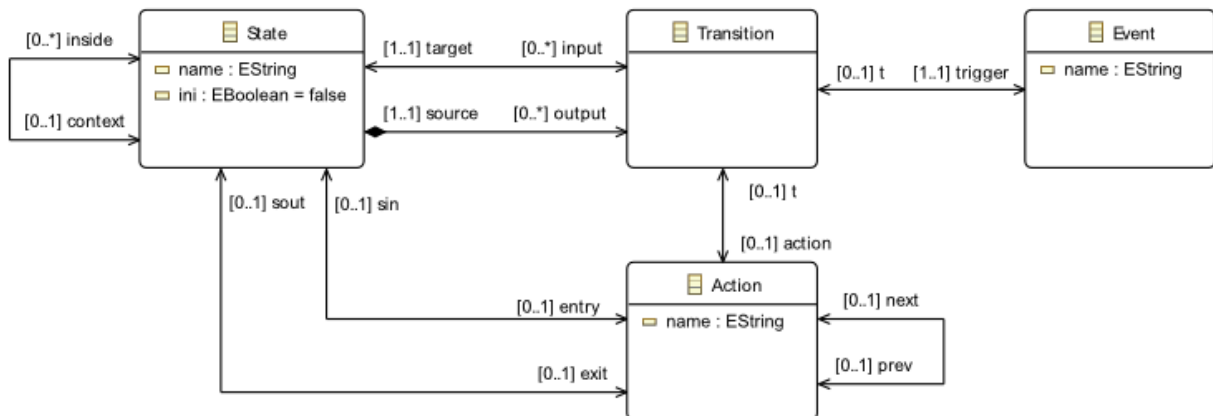
Right click on EClass New Child [Select EReference or EAttribute]

It must look as follows:



statechart.ecore models hierarchical statecharts. An State can contains inner States. Transitions relates States. Transitions can have associated Events. States and Transitions can have associated Actions.

Graphically, it looks as follows:



Please note: a *root* EClass is mandatory: the name *root* is not important and mandatory but, what is important is that, this EClass contains directly or indirectly all the other concrete EClasses.

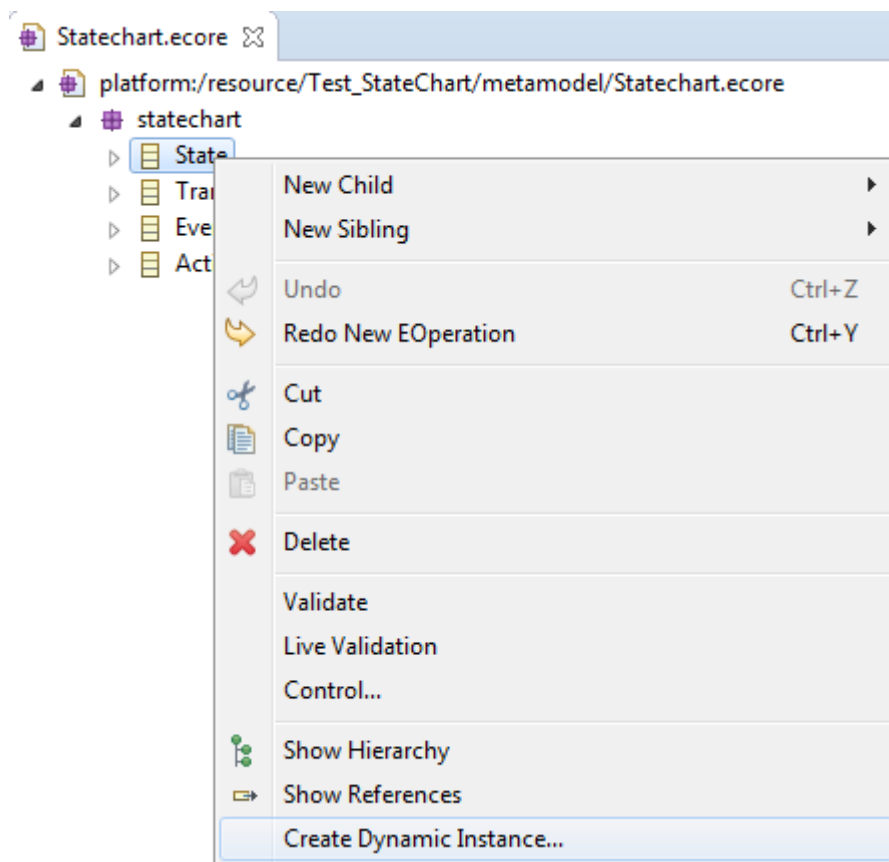
1.5. Create model

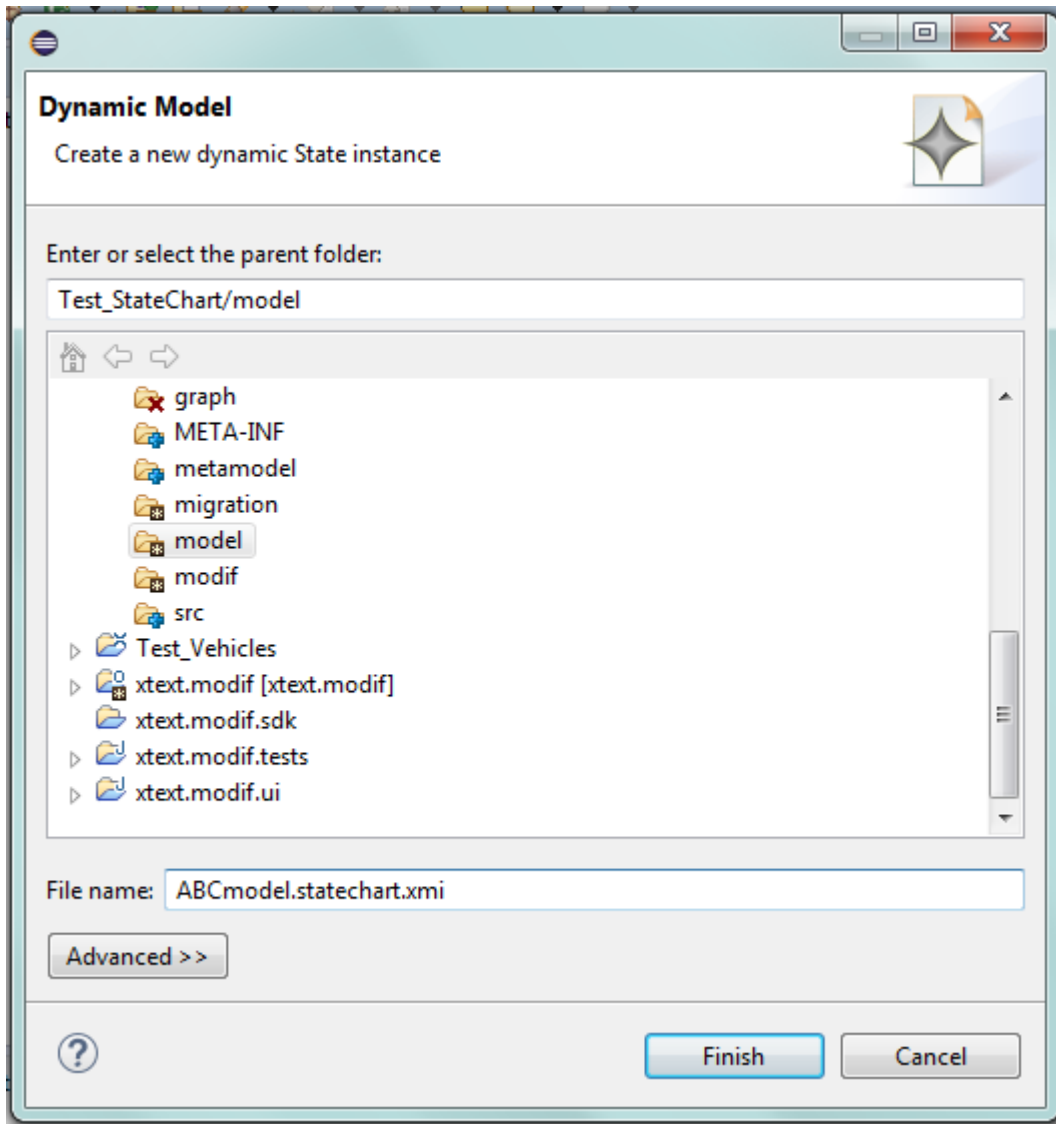
In the **model** folder, create a new ecore model. And for the example, name it **ABCmodel.statechart.xmi**.

Please note:

the model name must respect the following rule: modelName.**metamodelName.xmi**

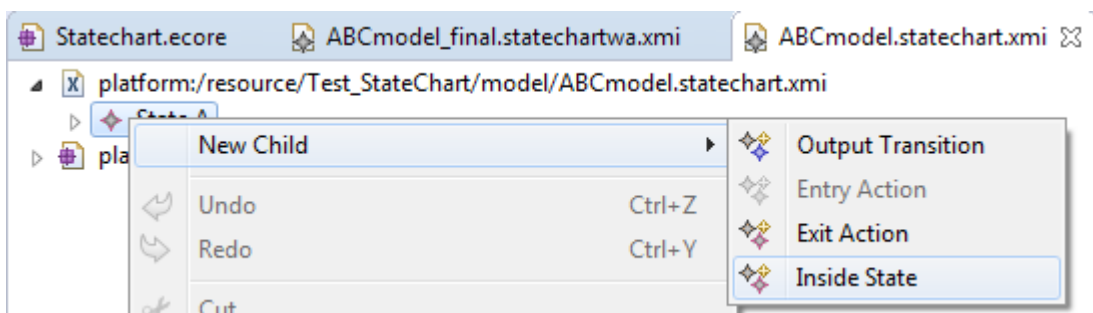
Right click on the root EClass [Root] Create Dynamic Instance



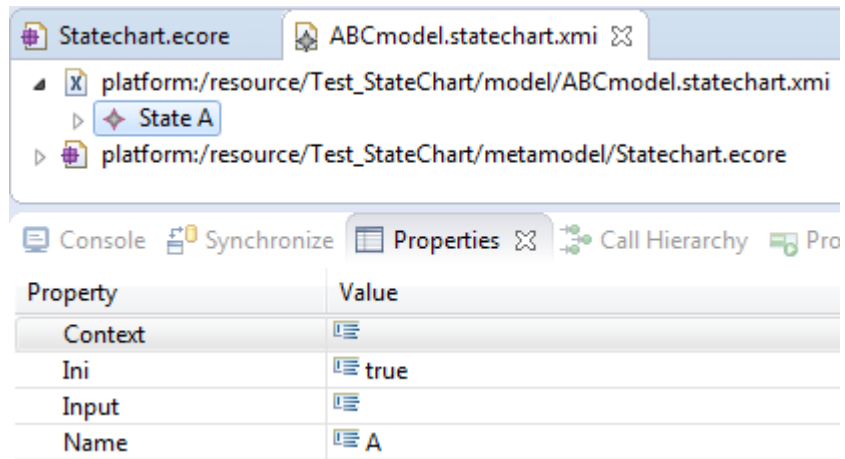


Add elements to the model

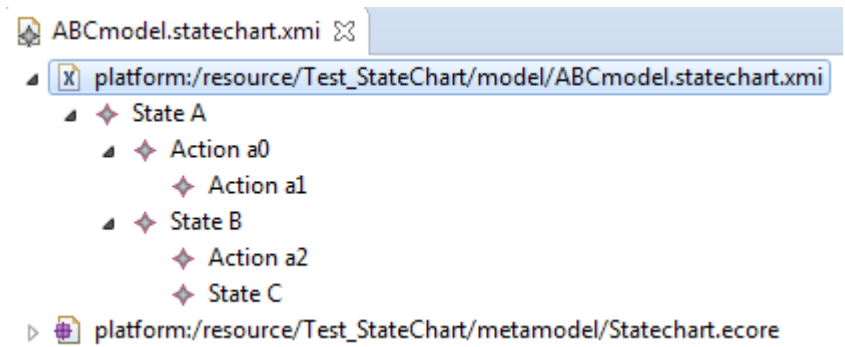
Right click on EClass New Child [Select element to create]



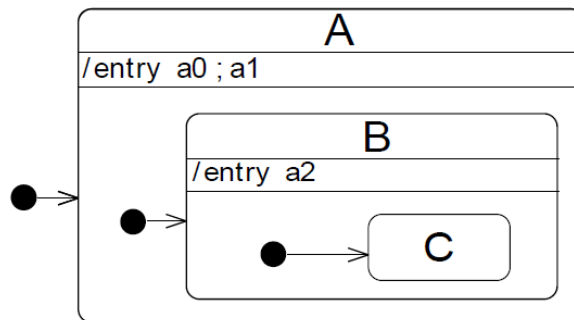
Fill the properties of the element



ABCmodel.statechart.xmi is a model compounded of three initial states: A, B and C. B is inside A and C is inside B. A has one entry Action, namely a0; a0 has one next Action, namely a1. B has one entry action, namely a2. It must look as follows:



Graphically, it looks as follows:



2. Import the Tool project

Follow the [Modif Project Import Documentation](#) in order to import the Tool_Statechart.

- **src folder:** it contains the source code of the tool to be reused. src/tool package contains three classes: Main, ToolService and ToolUI.

ToolService contains three functions: copy, identity and flatten.

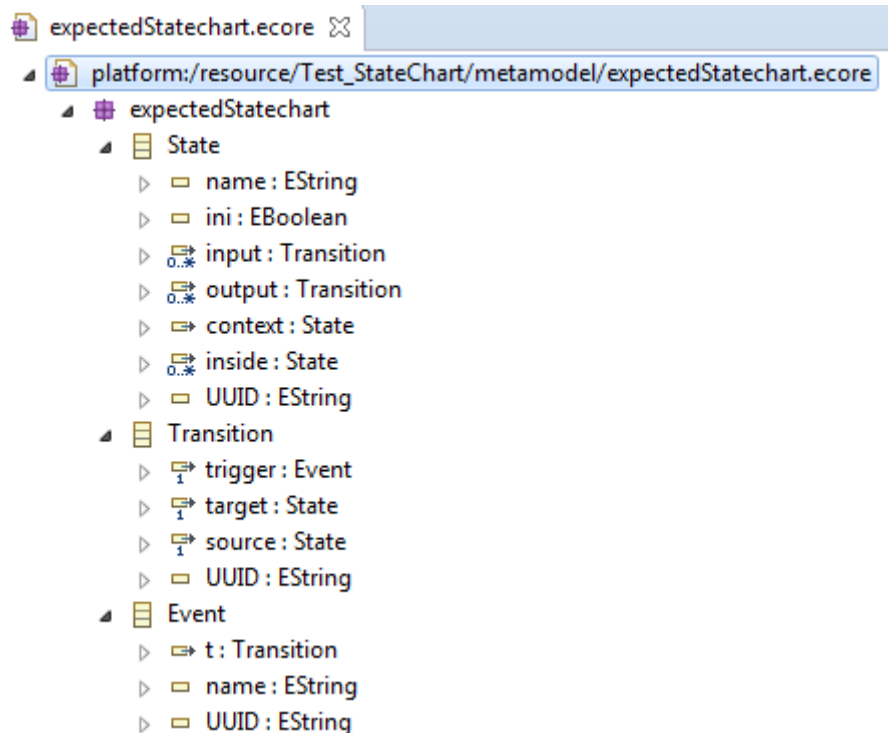
Copy produces a copy of the input model (identifiers of model elements are changed.)

Identity produces an exact copy of the input model (identifiers of model elements are preserved).

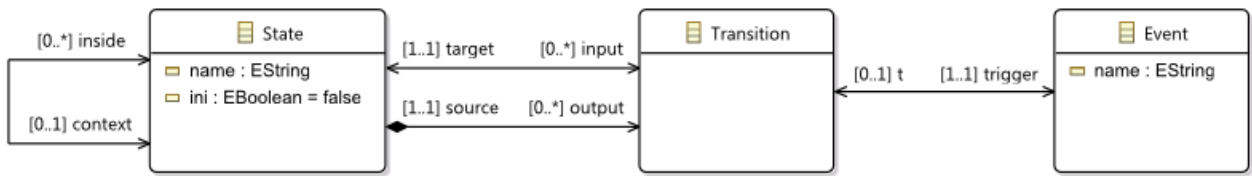
Flatten eliminates hierarchy and produces a model in which all states containing other states are removed.

- **metamodel folder:** it contains the metamodel of the tool. *expectedStatechart.ecore*
expectedStatechart.ecore models hierarchical statecharts. States can have other states inside them. States are related by means of Transitions. Transitions can have associated Events.

expectedStatechart.ecore looks as follows:



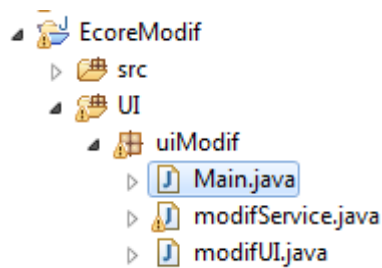
Graphically, it looks as follows:



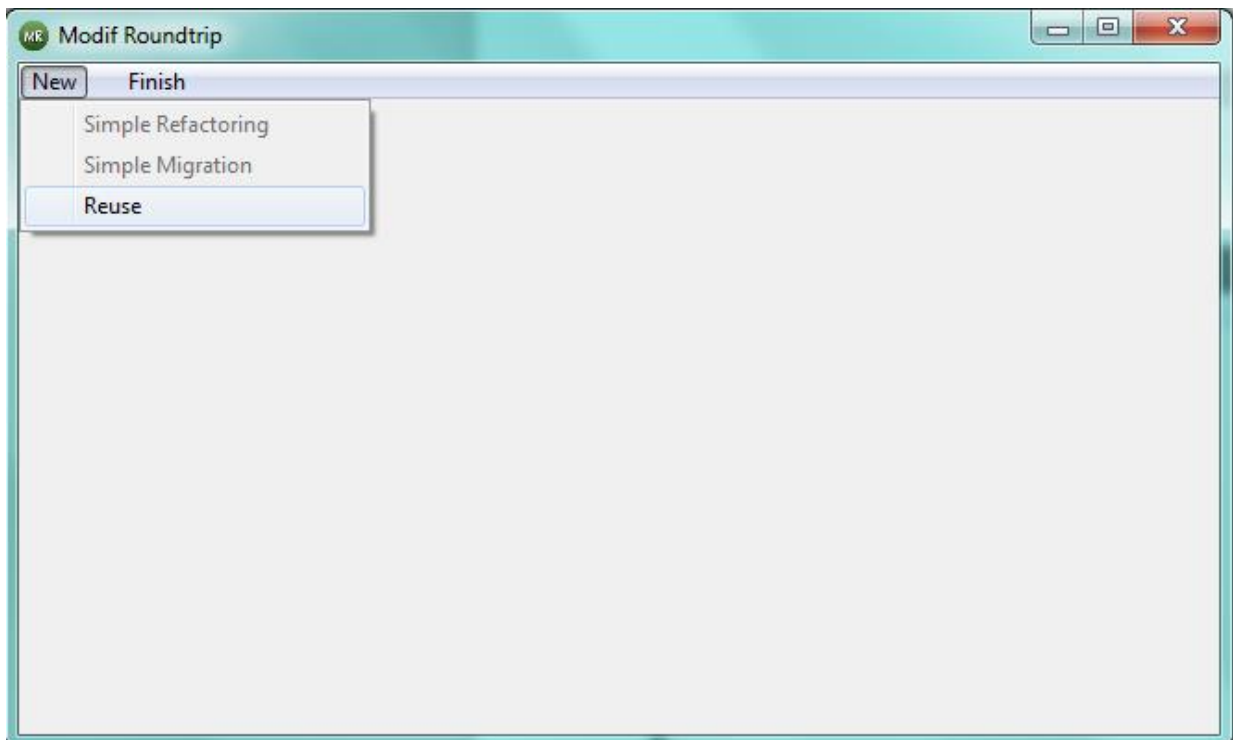
3. Execute Modif Roundtrip

1. Refactoring

Open the EcoreModif project, navigate to *UI/ uiModif* and execute *Main.java*



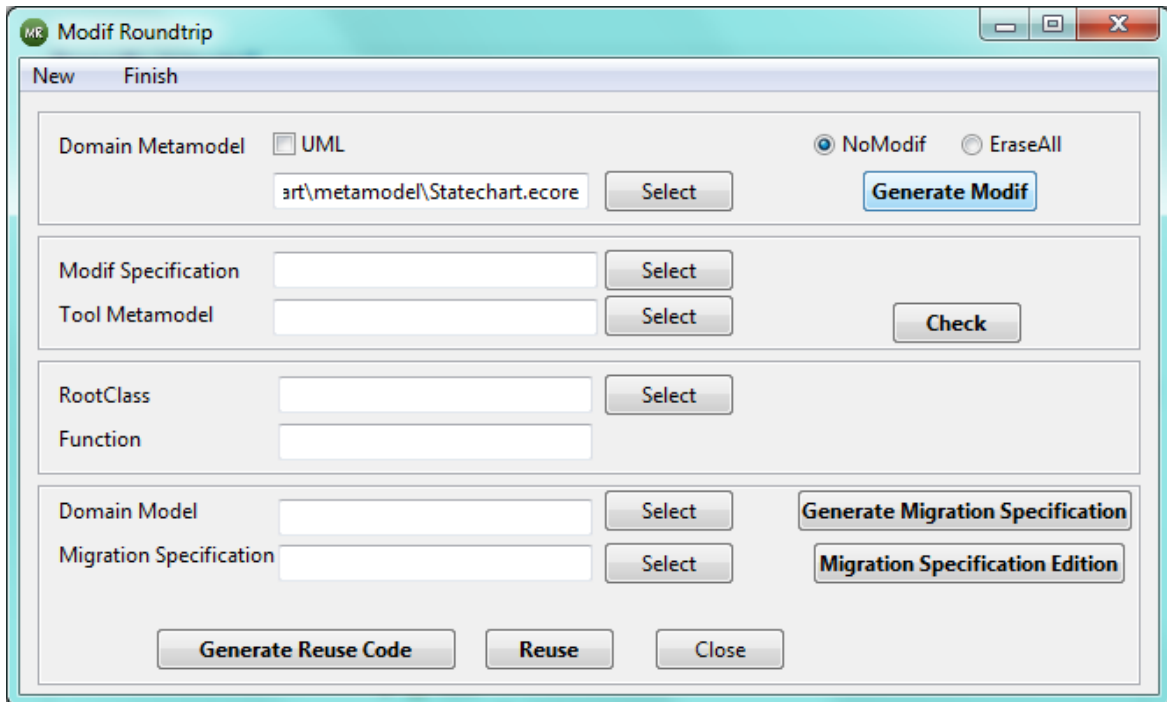
In the Modif Roundtrip form, click on *New* and then, click on *Reuse*.



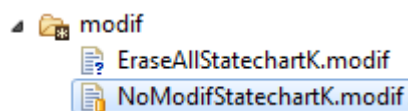
Now perform the following steps in order:

Specify Domain Metamodel and Generate Modif model

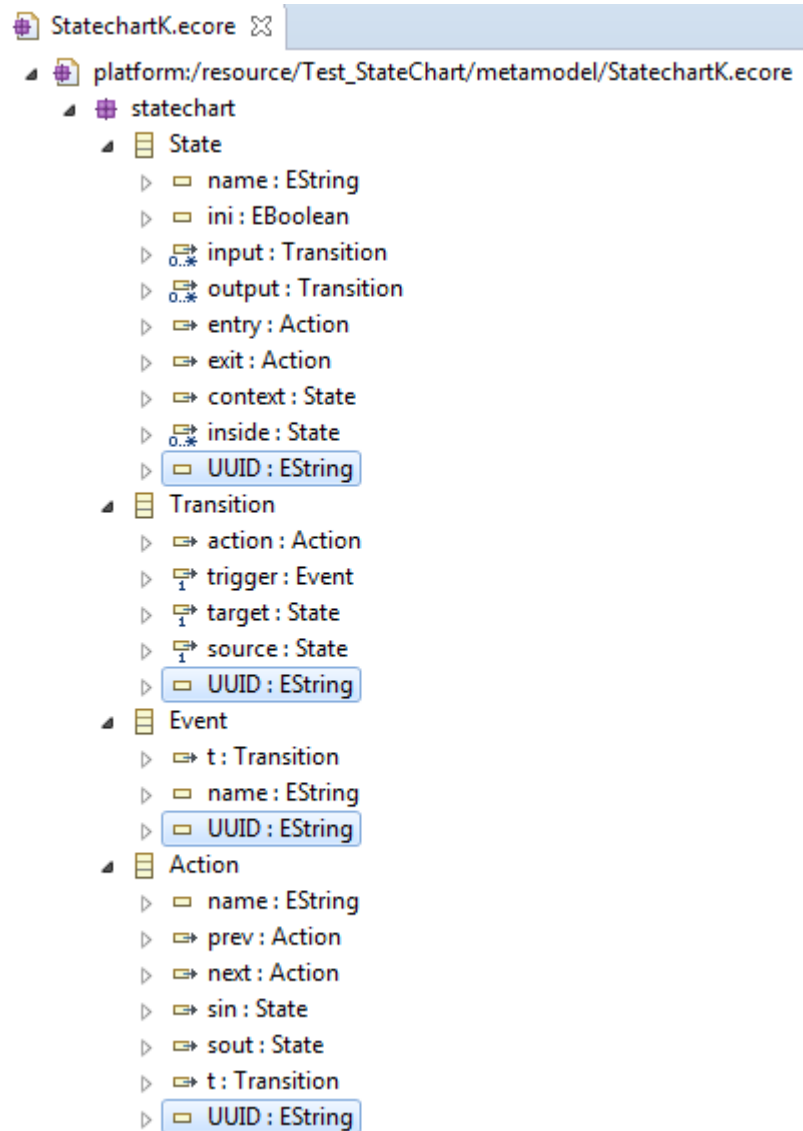
- Click *Select* for Domain Metamodel and specify the path to the ecore file Statechart.ecore (Test_StateChart/metamodel folder)
- *Select NoModif* (or *EraseAll*) and then click on the **Generate Modif** button. If the *Domain Metamodel* path is incorrect, an error message appears when executing the modif model generation



- Refresh the modif folder, so that the modif model appears.



- Refresh the metamodel folder, the file *StatechartK.ecore* appears. *StatechartK.ecore* is a copy of *Statechart.ecore* but with an additional *UUID* attribute on all its classes. The StatechartK ecore looks as follows:



Edit Modif Model

Edit the given Statechart2expectedStatechart.modif in order to put the location of your *Test_StateChart* project.

```
Statechart2expectedStatechart.modif
root statechart to expectedStatechart
Prefix statechart to expectedStatechart
URI "file:/C:/ModifRoundtrip/Test_StateChart/metamodel/StatechartK.ecore" to
"file:/C:/ModifRoundtrip/Tool_StateChart/metamodel/expectedStatechart.ecore"
```

We edited the generated modif file in this way:

- For **root**, change it to the root name of the root of the metamodel of the tool to be reused (*expectedStatechart* in this case)
- For **Prefix**, change it to the prefix of the *expectedStatechart*

- For **URI**, change it to the URI of the *expectedStatechart*
- For the references *entry* and *exit* of class *State*, put the key word **remove**
- For the reference *action* of the class *Transition*, put the key word **remove**
- For the class *Action*, put the key word **remove**

```

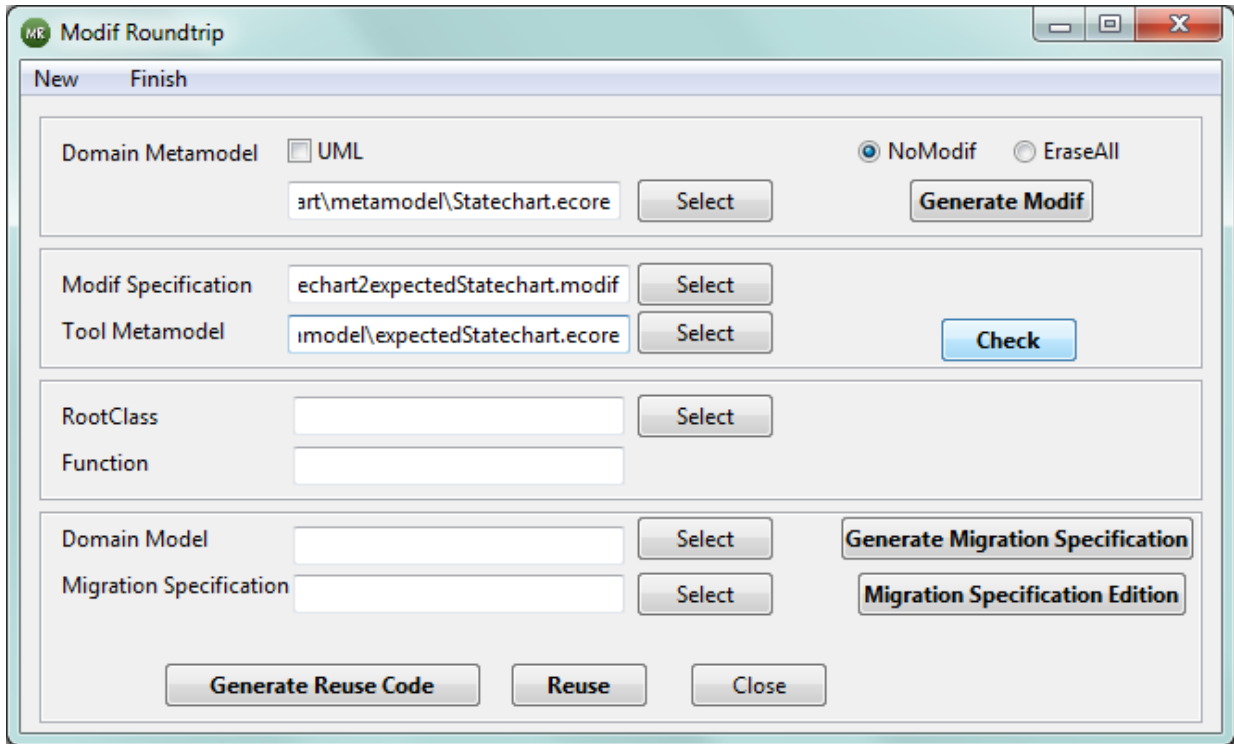
Statechart2expectedStatechart.modif
root statechart to expectedStatechart
Prefix statechart to expectedStatechart
URI "file:/C:/ModifRoundtrip/Test_StateChart/metamodel/StatechartK.ecore" to
"file:/C:/ModifRoundtrip/Tool_StateChart/metamodel/expectedStatechart.ecore"
class {
  State to State {
    att name to name bounds (0,1) to (0,1)
    att ini to ini bounds (0,1) to (0,1)
    ref input to input bounds (0,-1) to (0,-1)
    ref output to output bounds (0,-1) to (0,-1)
    remove ref entry to entry bounds (0,1) to (0,1)
    remove ref exit to exit bounds (0,1) to (0,1)
    ref context to context bounds (0,1) to (0,1)
    ref inside to inside bounds (0,-1) to (0,-1)
    att UUID to UUID bounds (0,1) to (0,1)
  };
  Transition to Transition {
    remove ref action to action bounds (0,1) to (0,1)
    ref trigger to trigger bounds (1,1) to (1,1)
    ref target to target bounds (1,1) to (1,1)
    ref source to source bounds (1,1) to (1,1)
    att UUID to UUID bounds (0,1) to (0,1)
  };
  Event to Event {
    ref t to t bounds (0,1) to (0,1)
    att name to name bounds (0,1) to (0,1)
    att UUID to UUID bounds (0,1) to (0,1)
  };
  remove Action to Action {
    att name to name bounds (0,1) to (0,1)
    ref prev to prev bounds (0,1) to (0,1)
    ref next to next bounds (0,1) to (0,1)
    ref sin to sin bounds (0,1) to (0,1)
    ref sout to sout bounds (0,1) to (0,1)
    att UUID to UUID bounds (0,1) to (0,1)
  }
}

```

Check and Refactor

- In the Modif Roundtrip form, set the Modif Specification *Statechart2expectedStatechart.modif*
- Set the Tool Metamodel (*Tool_StateChart/metamodel/expectedStatechart.ecore*)
- Click on the **Check** button

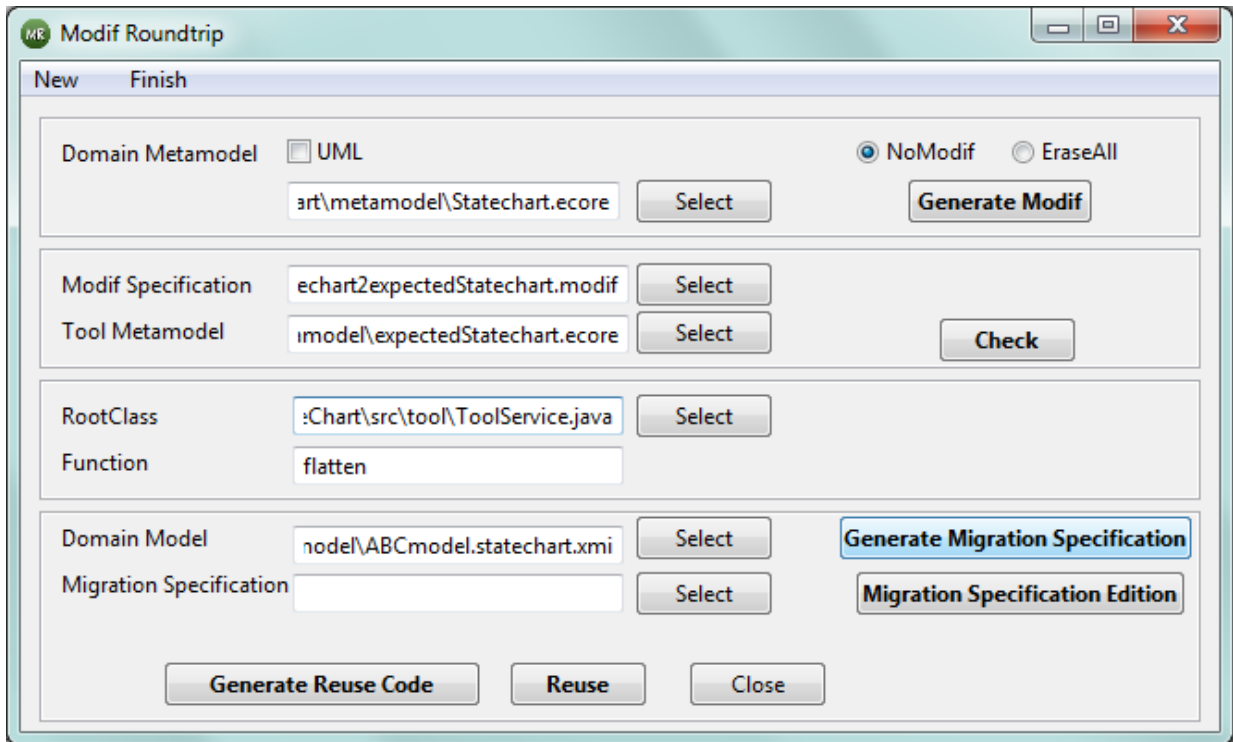
Modif Roundtrip, will execute the refactoring according to the operators specified in the modif file. Then, Modif Roundtrip checks if the refactored model fully matches with the Tool Metamodel, if not, an error message will appear.



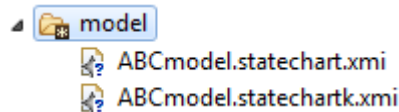
- Refresh the `Test_StateChart/metamodel` folder and you will notice that the file `expectedStatechart.ecore` is added. It will look as the tool metamodel introduced in section Import the Tool project (page 9).

2. Migration specification generation

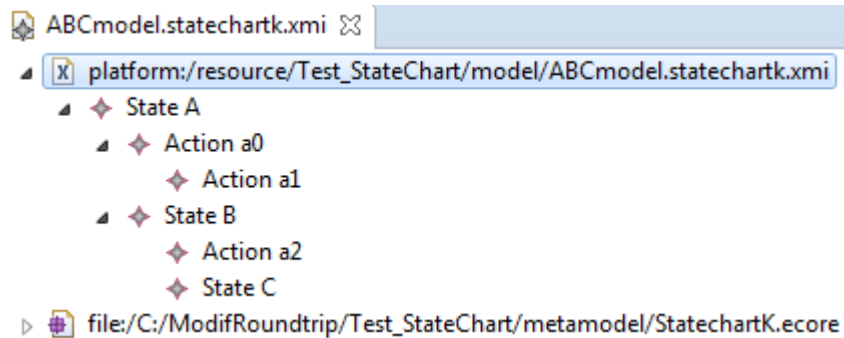
- In the Modif Roundtrip form, set the class containing the function to be reused, `ToolService.java` in this case (located at `Tool_StateChart/rcs/tool` folder)
- Write the name of the Function to be reused. In this example, `flatten` is the function to be reused, but you can also reuse `copy` and `identity`.
- Select the Domain Model (`ABCmodel.statechart.xmi`)
- Click on the **Generate Migration Specification** button



- Then go back to your Test project and refresh it, you will find a new model ABCmodel.statechartk.xmi



This model is a copy of the ABCmodel.statechart.xmi, but with a new UUID attribute.

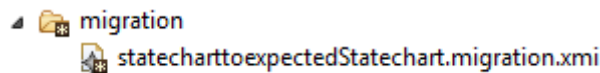


For this example, the UUID was filled as shown below,

A = 0	B = 3
a0 = 1	a2 = 4
a1 = 2	C = 5

Please note that the UUID values may change.

You will find that a migration specification is added in the migration folder



The migration specification indicates the modifications to be applied to `ABCmodel.statechartk.xml` in order to produce a model conforms to the `expectedStatechart.ecore` metamodel. In the example, instances identified with UUID 1, 2 and 4 are marked to be deleted. Instances identified with UUID 0, 3 and 5 are not marked to be deleted but their references *entry* and *exit* are.

The screenshot shows the migration specification file `statecharttoexpectedStatechart.migration.xml` with the following structure:

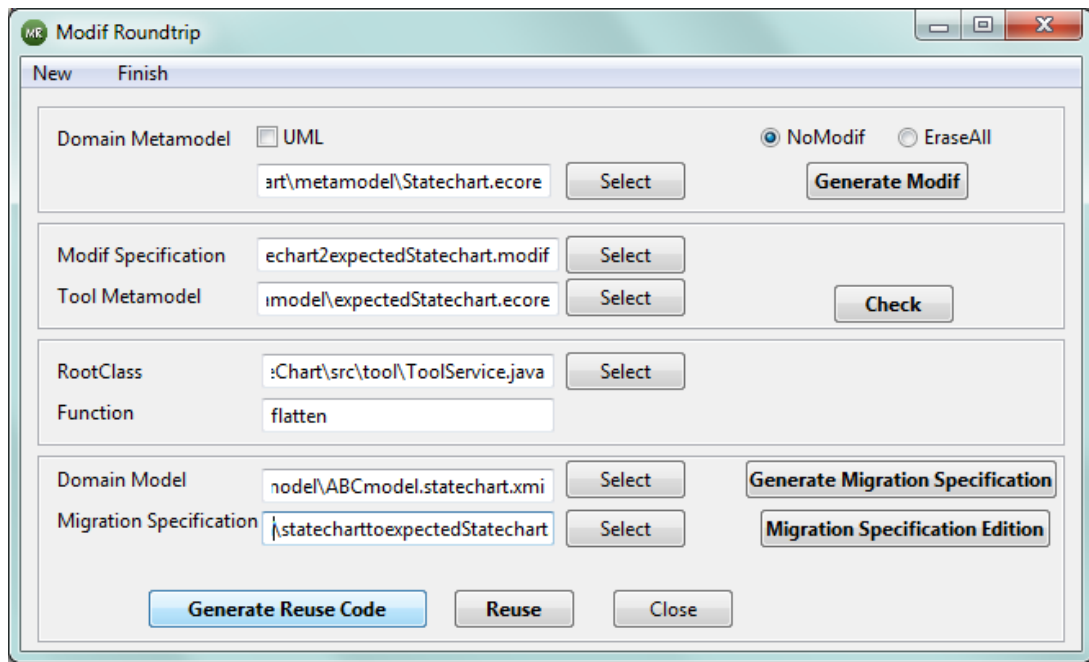
- Migration file: `C:/ModifRoundtrip/Test_StateChart/model/ABCmodel.statechartk.xml`
 - Instance 0
 - Deletion false
 - Deleted Reference entry
 - Deleted Reference exit
 - Instance 1
 - Deletion true
 - Instance 2
 - Deletion true
 - Instance 3
 - Deletion false
 - Deleted Reference entry
 - Deleted Reference exit
 - Instance 4
 - Deletion true
 - Instance 5
 - Deletion false
 - Deleted Reference entry
 - Deleted Reference exit

The properties table below the migration specification is:

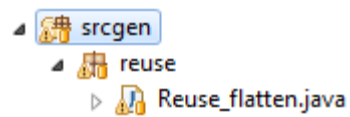
Property	Value
Input Metamodel URI	file:/C:/ModifRoundtrip/Test_StateChart/metamodel/StatechartK.ecore
Input Model URI	file:/C:/ModifRoundtrip/Test_StateChart/model/ABCmodel.statechartk.xml
Output Metamodel URI	file:/C:/ModifRoundtrip/Tool_StateChart/metamodel/expectedStatechart.ecore
Output Model URI	file:/C:/ModifRoundtrip/Test_StateChart/model/ABCmodel_migrated.expectedstatechart.xml

3. Reuse code generation and Migration and Reuse

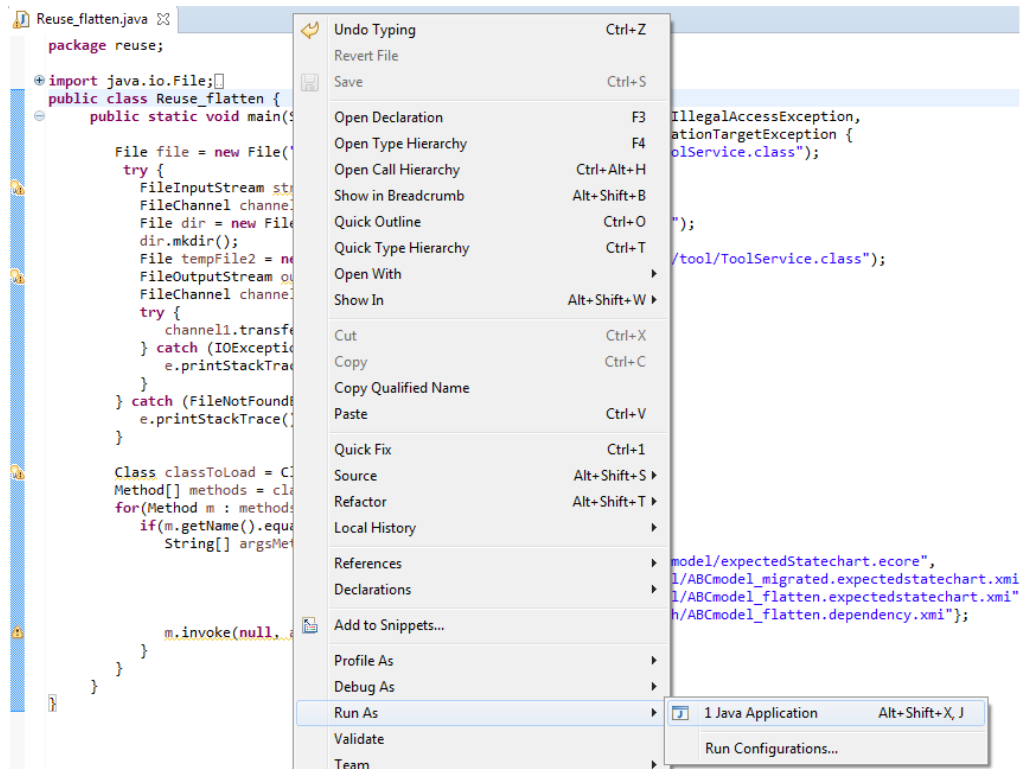
- In the Modif Roundtrip form, set the Migration Specification `statecharttoexpectedStatechart.migration.xml` and click on the **Generate Reuse Code** button



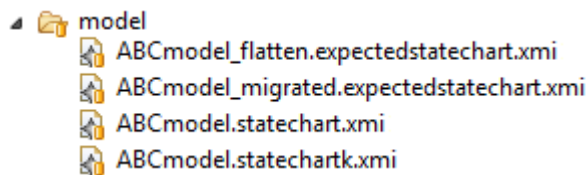
- Refresh the *srcgen* source folder; you will see a java class with the call code of the function you want to reuse.



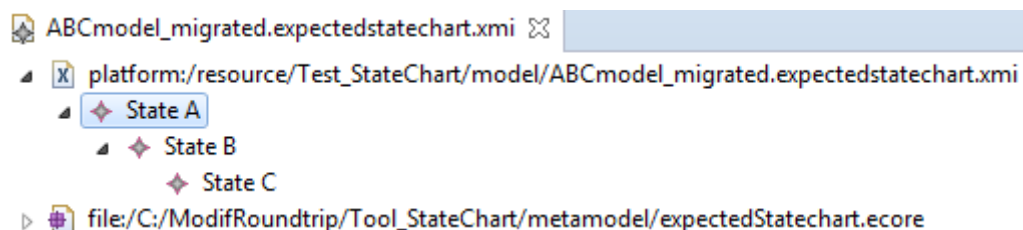
- Execute the code as a Java Application



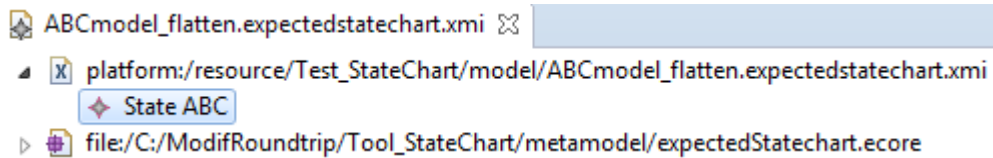
- Refresh the *model* folder. This will add two new models



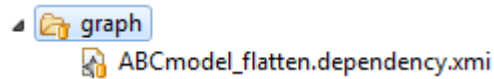
ABCmodel_migrated.expectedstatechart.xml, is the migrated model, it is a model conforms to *expectedStatechart* metamodel. You can see that all the changes specified by the Migration Specification are reflected in the instance model. Actions *a0*, *a1* and *a2* has been deleted. This model is used as input for the function *flatten* to be reused.



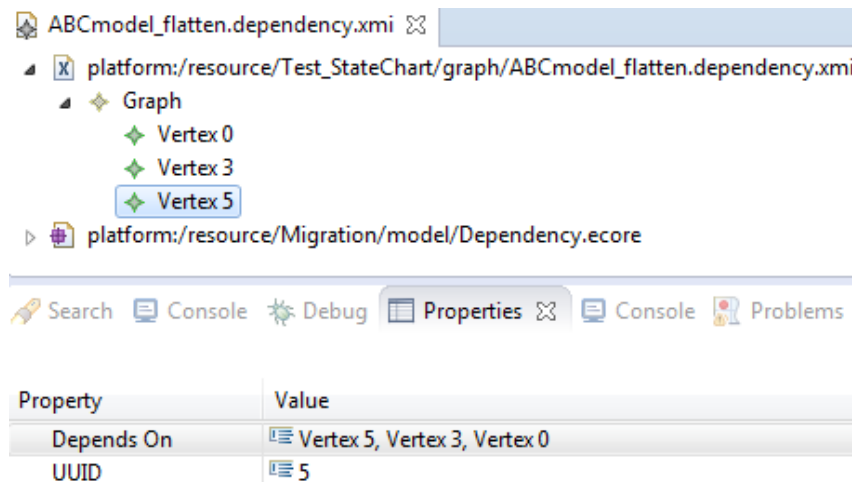
ABCmodel_flatten.expectedchart.xml is the output of the function *flatten*. In this model there is only one state, it has been renamed as *ABC*.



- Refresh the *graph* folder. The model graph *ABCmodel_flatten.dependency.xmi* will appear.



The model graph will look as follows.

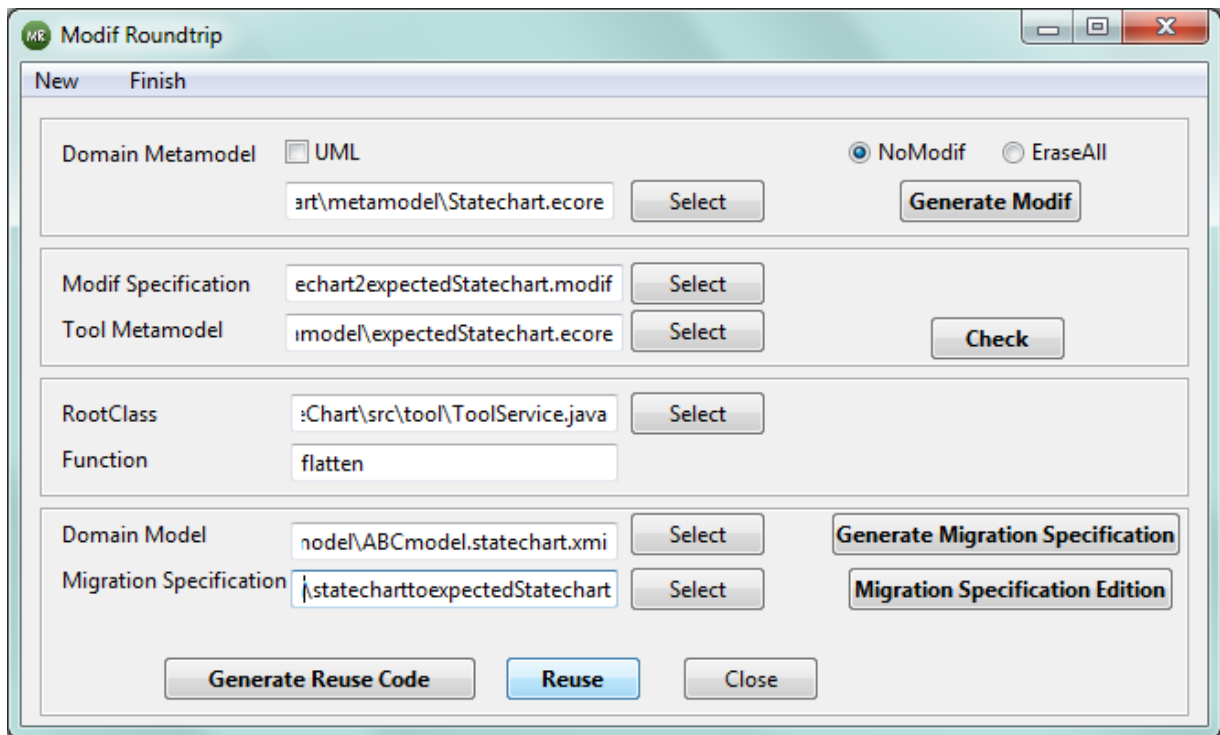


In the example, instance identified with UUID 3 (State B) depends on instances 0 and 3.

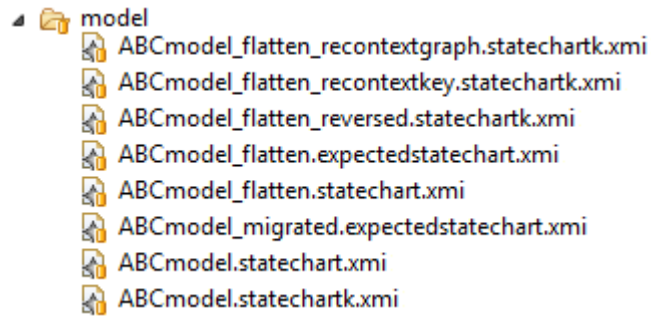
Instance identified with UUID 5 (State C) depends on instances 0, 3, and 5. It means, that the instance ABC (formerly called C) was renamed thanks to the names of instances 0 (A), 3 (B) and 5 (C).

4. Reverse Migration and Recontextualization

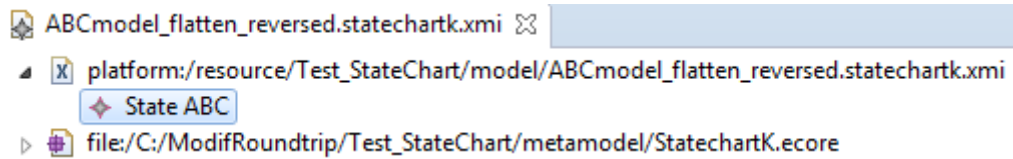
- In the Modif Roundtrip form, click on the **Reuse** button



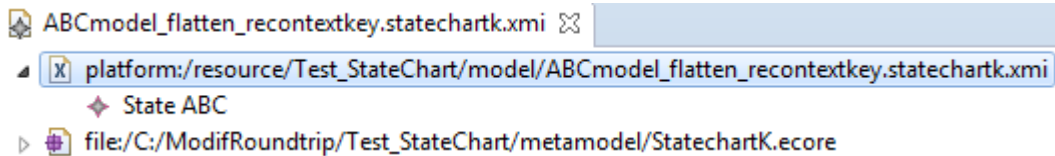
- Refresh the model folder. Four new models will be added to the model folder.



ABCmodel_flatten_reversed.statechartk.xmi, is the flattened model, but it is conform to the *StatechartK* metamodel.



ABCmodel_flatten_recontextkey.statechartk.xmi, is the model on which the recontextualization by keys has been applied. In this example, recontextualization by keys did not recover any deleted instance.



ABCmodel_flatten_recontextgraph.statechartk.xmi, is the model on which the recontextualization by graph has been applied. In this example, instance *a2* deleted during migration was recovered.

Property	Value
Name	a2
Prev	
Sin	State ABC
Sout	
T	
UUID	4

ABCmodel_flatten.statechart.xmi, is the recontextualized model, in which all UUID have been removed.

Property	Value
Name	a2
Prev	
Sin	State ABC
Sout	
T	

Then, exit the *Modif Roundtrip* form, by clicking on the *End* button.

Congratulations

You have used Modif Roundtrip tools to reuse an existing function.

Enjoy Modif Roundtrip

4. Contact

Jean-Philippe Babau: babau@univ-brest.fr

Paola Vallejo: vallejoco@univ-brest.fr