Metamodeling with Eclipse

Luis Pedro
Centre Universitaire D'Informatique
Universite de Geneve

Matteo Risoldi
Centre Universitaire D'Informatique
Universite de Geneve

SMV technical report series, N^o N/A, May 2008

Centre Universitaire d'Informatique
Battelle Bat. A
Route de Drize 7
1227 Carouge
Switzerland

Abstract:
This technical report describes how to use the Eclipse Framework, in particular the Eclipse Modeling Framework (EMF) and the Graphical Modeling Framework (GMF) in order to produce metamodels, instances of them and how to transform them.

Document Version 0-2-0 Updated to Eclipse 3.3.2

Contents

1 Introduction **2**
1.1 Requirements 2

2 Create the Petri Nets Metamodel **4**

3 Creating Models **9**

4 Providing the Model with a Graphical Interface **10**
4.1 Customize Graphical Elements 15

5 Optional Informations **21**
5.1 Create an Eclipse Plug-in 21
5.2 Optional Sources of Information 22

Software Modeling and Verification
Department of Computer Science
University of Geneva
Battelle Bat. A
Route de Drize 7
CH-1227 Carouge
Switzerland

tel: +41 22 37 90 161
fax: +41 22 37 90 250
`http:/smv.unige.ch/`

Corresponding author:
Luis Pedro
tel: +41 22 37 91 117
`Luis.Pedro@cui.unige.ch`
`http://smv.unige.ch/~pedro`

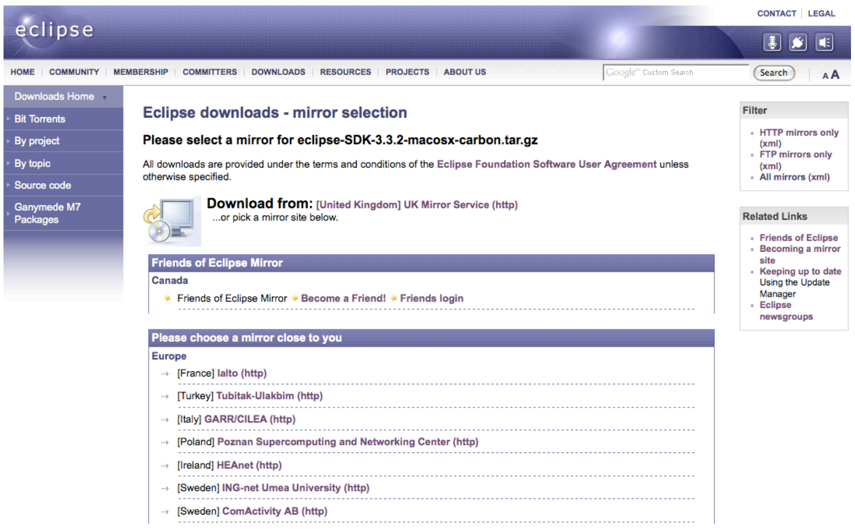
List of Figures

1 Download Eclipse form http://www.eclipse.org/downloads/ 2
2 Select Eclipse Components for installation 3
3 Creating a New EMF Empty Project 4
4 Add an Ecore model to the EMF project 5
5 Renaming the default ECore package 5
6 Bidirectional association to Direct associations 7
7 Petri Nets metamodel in UML-like syntax 7
8 Petri Nets metamodel EMF Tree view 8
9 Eclipse Workspace after generating code from the PetriNets.genmodel 8
10 Initializing a new Petri Nets model 9
11 Creation of a new GMF Project 10
12 GMF Overview 11
13 Specifying the basic graphical definition of the domain model 12
14 Specifying the map domain model elements 13
15 Output Arc Map Configuration 13
16 Input Arc Map Configuration 14
17 Graphically creating Petri Nets 15
18 Modifying .gmfmap Tool mapping 15
19 Creating Petri Nets models, graphically and tree-based editor 16
20 Adding a new Figure and Descriptors to the gmfgraph 17
21 Petri Nets model with correct concrete syntax for the Petri Net Place 18
22 Petri Nets model with Arcs and Places initial marking explicitly in the diagram 19
23 Petri Nets model the correct graphical syntax for Places and Arcs . 20
24 Select Available Plug-ins and Features 21

1 Introduction

In this report we are going to present how to metamodel with Eclipse Modeling Framework (EMV) [1] and Graphical Modeling Framework (GMF) [2]. During this discussion we will use the Petri Nets language as case study. At the end we will also show how to write transformations using the automatically code automatically generated from the metamodels. As an example of transformation we will show how to pass from standard Petri Nets to Algebraic Petri Nets.

1.1 Requirements



Eclipse downloads - mirror selection

Please select a mirror for eclipse-SDK-3.3.2-macosx-carbon.tar.gz

All downloads are provided under the terms and conditions of the Eclipse Foundation Software User Agreement unless otherwise specified.

Download from: [United Kingdom] UK Mirror Service (http)
...or pick a mirror site below.

Friends of Eclipse Mirror
Canada
Friends of Eclipse Mirror Become a Friend! Friends login

Please choose a mirror close to you
Europe
[France] Ialto (http)
[Turkey] Tubitak-Ulakbim (http)
[Italy] GARR/CILEA (http)
[Poland] Poznan Supercomputing and Networking Center (http)
[Ireland] HEAnet (http)
[Sweden] ING-net Umea University (http)
[Sweden] ComActivity AB (http)

Figure 1: Download Eclipse form http://www.eclipse.org/downloads/

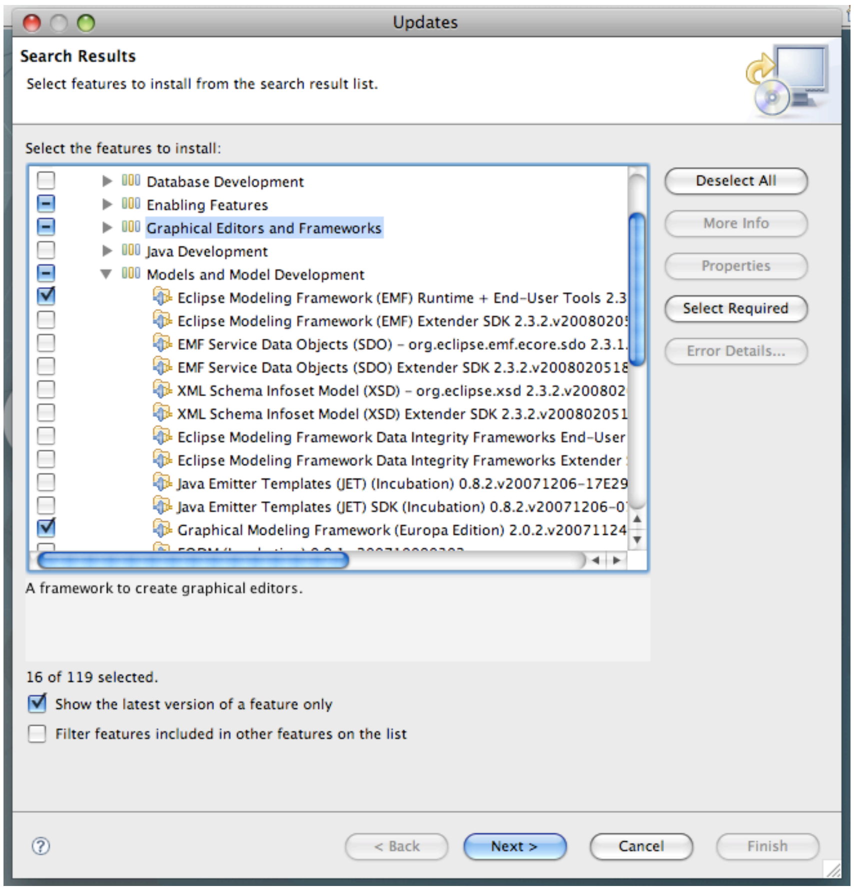
What do you need to run this example:

- Install Eclipse(Fig. 1)
- In Eclipe go to Help → Software Updates → Find and Install, then:
  - Search for new features to install
  - Europa Discovery Site

  If the Europa Discovery site is not in the list of sites, click ”New remote site” and add the Europa Discovery site with the url: `http://download.eclipse.org/europa/releases/` Finally choose to install the following components

  - Eclipse Modeling Framework (EMF)
  - Graphical Modeling Framework (GMF)

  and click the ”Select Required” button to install all dependencies. See Fig. 2 for details.



Search Results

Select features to install from the search result list.

Select the features to install:

- [ ] Database Development
- [–] Enabling Features
- [–] Graphical Editors and Frameworks
- [ ] Java Development
- [–] Models and Model Development
  - [✓] Eclipse Modeling Framework (EMF) Runtime + End-User Tools 2.3
  - [ ] Eclipse Modeling Framework (EMF) Extender SDK 2.3.2.v2008020
  - [ ] EMF Service Data Objects (SDO) - org.eclipse.emf.ecore.sdo 2.3.1.
  - [ ] EMF Service Data Objects (SDO) Extender SDK 2.3.2.v200802051
  - [ ] XML Schema Infoset Model (XSD) - org.eclipse.xsd 2.3.2.v200802
  - [ ] XML Schema Infoset Model (XSD) Extender SDK 2.3.2.v20080205
  - [ ] Eclipse Modeling Framework Data Integrity Frameworks End-User
  - [ ] Eclipse Modeling Framework Data Integrity Frameworks Extender
  - [ ] Java Emitter Templates (JET) (Incubation) 0.8.2.v20071206-17E29
  - [ ] Java Emitter Templates (JET) SDK (Incubation) 0.8.2.v20071206-0
  - [✓] Graphical Modeling Framework (Europa Edition) 2.0.2.v20071124

A framework to create graphical editors.

16 of 119 selected.

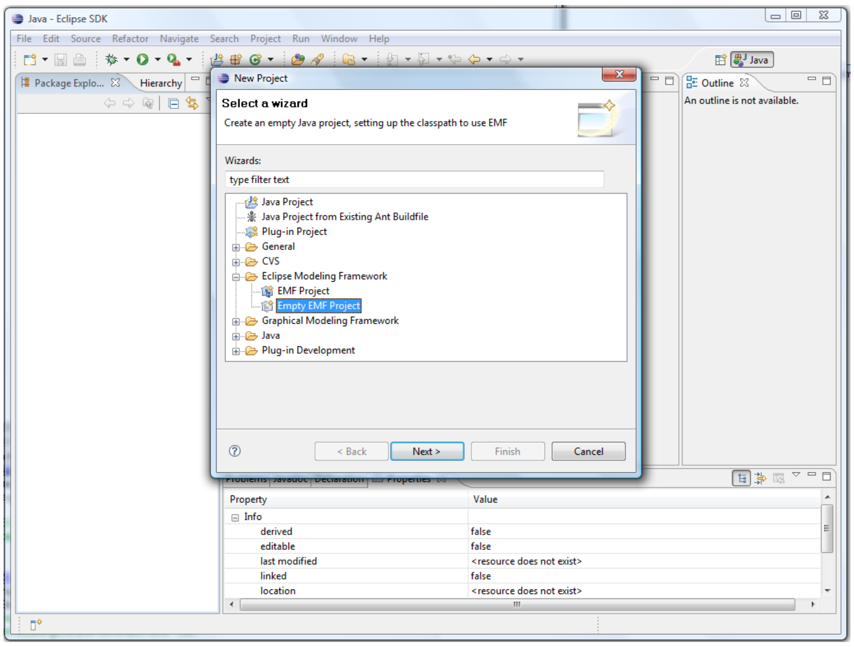
[✓] Show the latest version of a feature only

[ ] Filter features included in other features on the list

Figure 2: Select Eclipse Components for installation

2 Create the Petri Nets Metamodel

First of all let's create a new EMF project. Go to File → New → Project and choose Empty EMF Project. See Fig. 3.



Select a wizard

Create an empty Java project, setting up the classpath to use EMF

An outline is not available.

Property	Value
Info	
derived	false
editable	false
last modified	<resource does not exist>
linked	false
location	<resource does not exist>

Figure 3: Creating a New EMF Empty Project

Provide your project with a name, e.g. PetriNetsMM. You'll notice that in the structure of the newly created project there is a model folder. Right click on it and select New → Other. In the next panel choose Ecore Model click Next and provide a name to your model e.g. PetriNets.ecore. Please note that all Ecore models must finish with the .ecore extension (see Fig. 4).

Just press the Finish button and you'll notice that you have a new file listed in the model folder of your project.

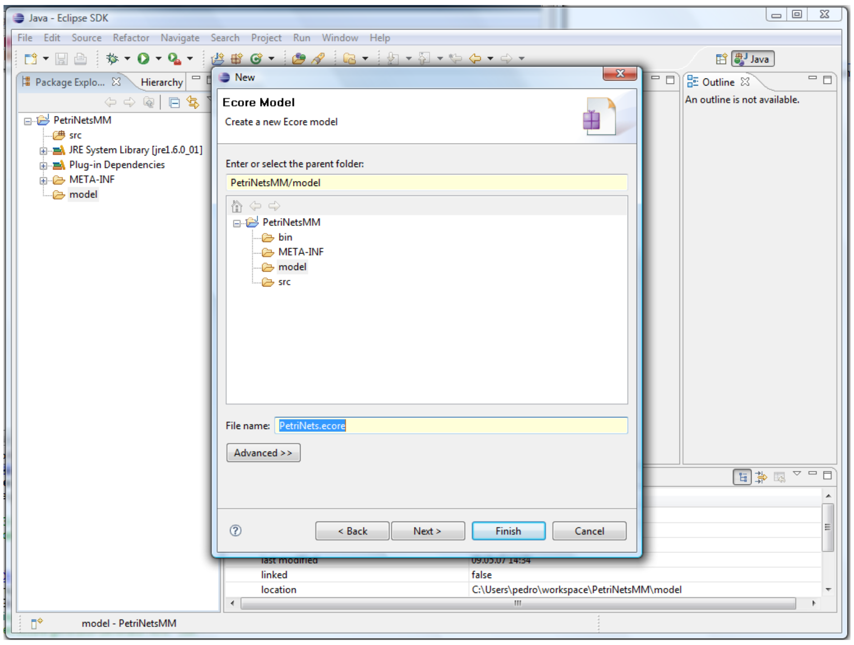
Exploring the PetriNets.ecore tree will allow you to see that an ECore package "null" was created by default. Click on it and change its name, NS prefix and NS URI by using the Properties tab (Fig. 5). Set these features to StandardPetriNets.

At this point you can choose wether to create the metamodel using the tree-based editor or the graphical editor provided by EMF. In order to edit the metamodel using the graphical editor you must first initialize an ECore diagram. This is done by right-clicking on the file PetriNets.ecore and chosing the option Initialize ecore_diagram diagram file. Provide the diagram with a name (PetriNets.ecore_diagram in our case) making sure that the name ends by .ecore_diagram.

The Petri Nets metamodel we are going to implement is composed by:

Classes representing the Petri Nets entities

- PetriNet with attributes name:String



Ecore Model

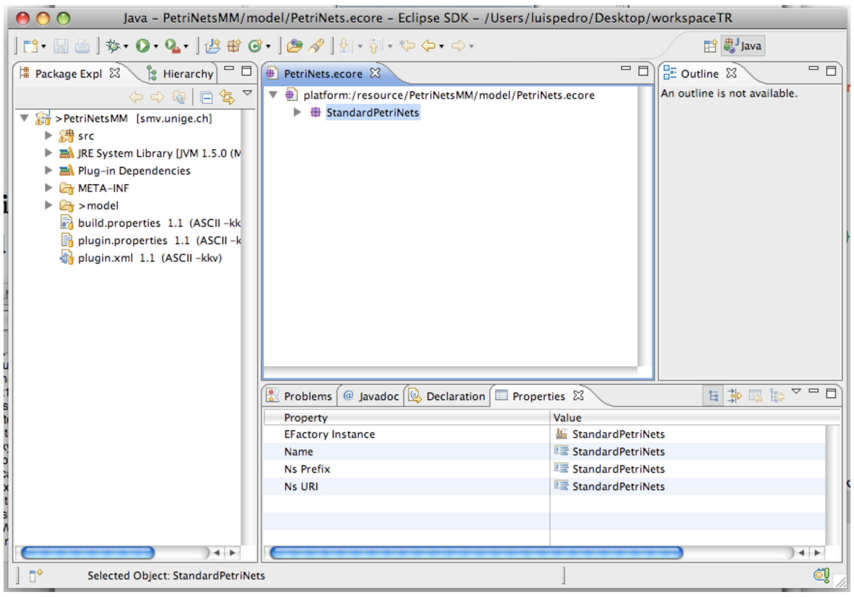
Create a new Ecore model

Enter or select the parent folder: PetriNetsMM/model

File name: PetriNets.ecore

linked	false
location	C:\Users\pedro\workspace\PetriNetsMM\model

Figure 4: Add an Ecore model to the EMF project


Java – PetriNetsMM/model/PetriNets.ecore – Eclipse SDK – /Users/luispedro/Desktop/workspaceTR
Java
Package Expl
Hierarchy
>PetriNetsMM [smv.unige.ch]
src
JRE System Library [JVM 1.5.0 (M
Plug-in Dependencies
META-INF
>model
build.properties 1.1 (ASCII -kk
plugin.properties 1.1 (ASCII -k
plugin.xml 1.1 (ASCII -kkv)
PetriNets.ecore
platform:/resource/PetriNetsMM/model/PetriNets.ecore
StandardPetriNets
Outline
An outline is not available.
Problems
Javadoc
Declaration
Properties
Selected Object: StandardPetriNets

Property	Value
EFactory Instance	StandardPetriNets
Name	StandardPetriNets
Ns Prefix	StandardPetriNets
Ns URI	StandardPetriNets

Figure 5: Renaming the default ECore package

- Place with attributes name:String, capacity:Int, numberOfTokens:Int
- InputArc with attributes weight:Int
- OutputArc with attributes weight:Int
- Transition with attributes name:String

Compositions/Aggregations representing containment relations between entities

- containsPlaces between PetriNet and Places with multiplicity in the Lower Bound equal to 0 and in the Upper Bound equal to *
- containsTransitions between PetriNet and Transition with multiplicity in the Lower Bound equal to 0 and in the Upper Bound equal to *
- containsInputArcs between PetriNet and InputArc with multiplicity in the Lower Bound equal to 0 and in the Upper Bound equal to *
- containsOutputArcs between PetriNet and OutputArc with multiplicity in the Lower Bound equal to 0 and in the Upper Bound equal to *

Associations representing the relations between the entities of a Petri Net

- inputArcFromPlace between OutputArc and Place with with multiplicity in the Lower Bound equal to 1 and in the Upper Bound equal to 1
- inputArcToTransition between InputArc and Transition with with multiplicity in the Lower Bound equal to 1 and in the Upper Bound equal to 1
- outputArcToPlace between OutputArc and Place with with multiplicity in the Lower Bound equal to 1 and in the Upper Bound equal to 1
- outputArcFromTransition between outputArc and Transition with with multiplicity in the Lower Bound equal to 1 and in the Upper Bound equal to 1

Please note that: Associations are **always** directed associations. It means that, if you have a UML-like association which is bidirectional like in the upper part of Fig. 6, you must rewrite it as depicted in the lower part of Fig. 6, separating the two directions of the association into two different associations. Also note that cardinality * in EMF is denoted by -1 (the editor will automatically display it as * in the diagram).

The Petri Nets metamodel should resemble the model in Fig. 7 - if you view it by opening your PetriNets.ecore_diagram file - and in Fig. 8 - providing the EMF tree view of the metamodel that you obtain by opening the PetriNets.ecore file.

The Petri Nets metamodel has now been defined and implemented in the EMF. The next thing is to automatically generate the Java code for accessing the metamodel, for creating Petri Nets models and for serializing and de-serializing them. In order to do this you need to go to File → New → Other and select EMF Model. Provide your EMF Model with a name that must be terminated by .genmodel, like PetriNets.genmodel, and place it under you model folder. Then select Ecore model

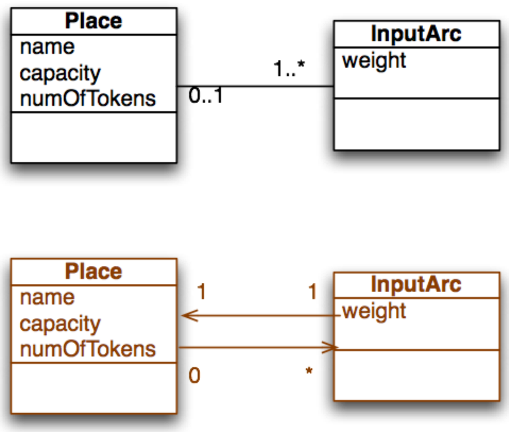


Figure 6: Bidirectional association to Direct associations

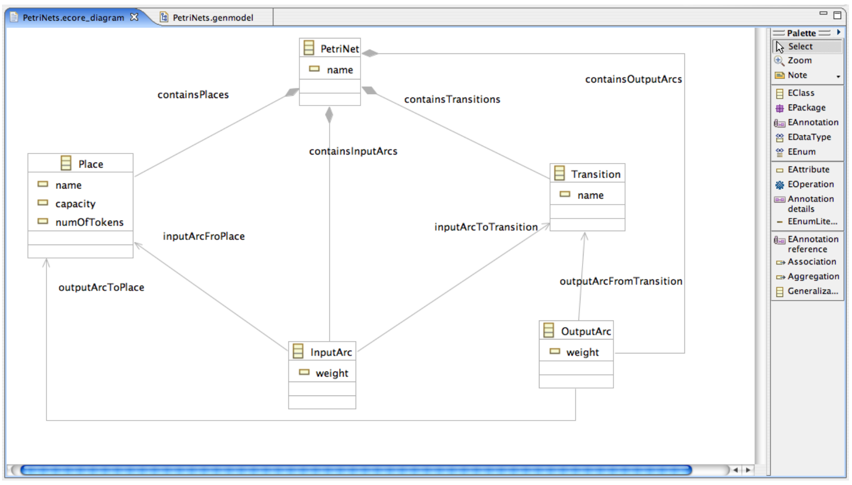
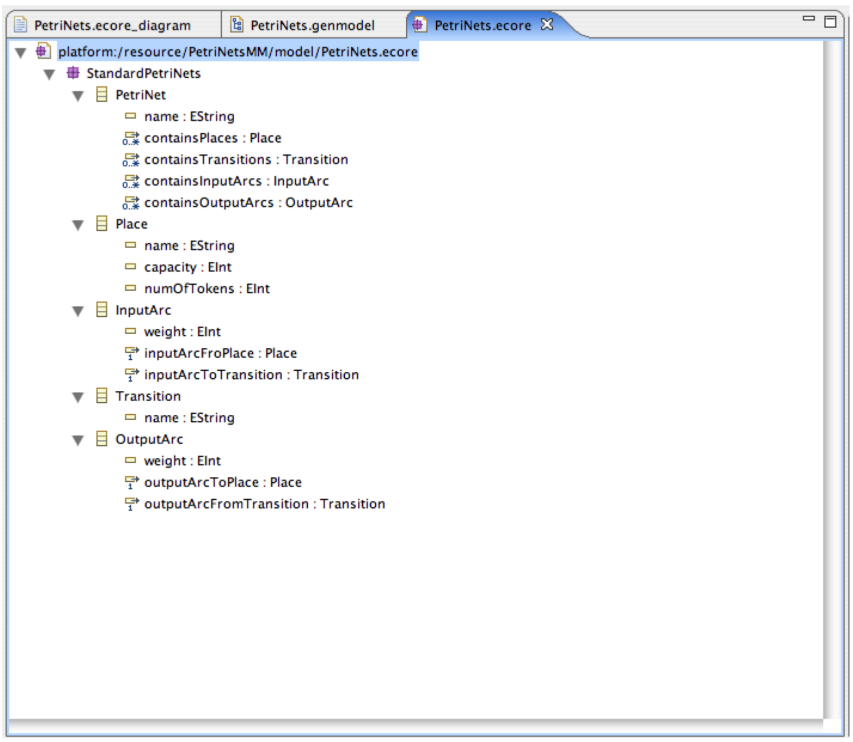


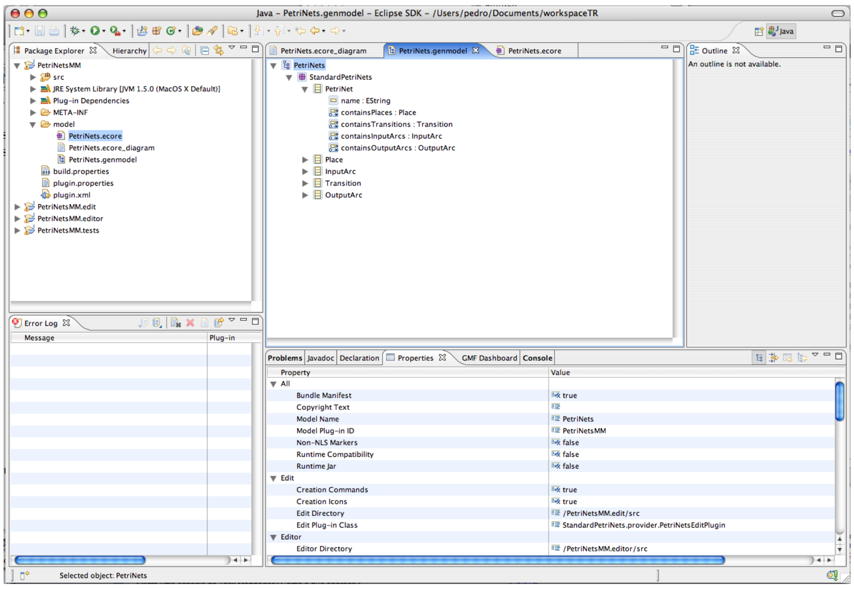
Figure 7: Petri Nets metamodel in UML-like syntax

in the Model Importer selection menu. Next choose Browse Workspace and provide the PetriNets.ecore file as selection. Press Next and then Finish. Now you should double click on the PetriNets.genmodel file and select the root of it. Right click on it and select Generate All. After this process you should find four new folders in your Eclipse workspace as presented in Fig. 9 (a src folder inside your project, and three new projects named PetriNetsMM.edit, PetriNetsMM.editor, PetriNetsMM.tests).



```
platform:/resource/PetriNetsMM/model/PetriNets.ecore
  StandardPetriNets
    PetriNet
      name : EString
      containsPlaces : Place
      containsTransitions : Transition
      containsInputArcs : InputArc
      containsOutputArcs : OutputArc
    Place
      name : EString
      capacity : EInt
      numOfTokens : EInt
    InputArc
      weight : EInt
      inputArcFroPlace : Place
      inputArcToTransition : Transition
    Transition
      name : EString
    OutputArc
      weight : EInt
      outputArcToPlace : Place
      outputArcFromTransition : Transition
```

Figure 8: Petri Nets metamodel EMF Tree view



```
PetriNetsMM
  src
  JRE System Library [JVM 1.5.0 (MacOS X Default)]
  Plug-in Dependencies
  META-INF
  model
    PetriNets.ecore
    PetriNets.ecore_diagram
    PetriNets.genmodel
  build.properties
  plugin.properties
  plugin.xml
PetriNetsMM.edit
PetriNetsMM.editor
PetriNetsMM.tests
```

```
PetriNets
  StandardPetriNets
    PetriNet
      name : EString
      containsPlaces : Place
      containsTransitions : Transition
      containsInputArcs : InputArc
      containsOutputArcs : OutputArc
    Place
    InputArc
    Transition
    OutputArc
```

An outline is not available.

Message	Plug-in

Property	Value
All	
Bundle Manifest	true
Copyright Text	
Model Name	PetriNets
Model Plug-in ID	PetriNetsMM
Non-NLS Markers	false
Runtime Compatibility	false
Runtime Jar	false
Edit	
Creation Commands	true
Creation Icons	true
Edit Directory	/PetriNetsMM.edit/src
Edit Plug-in Class	StandardPetriNets.provider.PetriNetsEditPlugin
Editor	
Editor Directory	/PetriNetsMM.editor/src

Selected object: PetriNets

Figure 9: Eclipse Workspace after generating code from the PetriNets.genmodel

3 Creating Models

Once at this stage, it is possible to generate instances of the Petri Nets metamodel by using the previously generated code. Press the button Run (or click Run → Run) and configure it for running as an Eclipse Application. Provide it with a name and a workspace at your wish. Click Run and a new Eclipse instance will be opened.

In the new instance of Eclipse click New → Project and select Project from the General folder. Press Next and provide the project with a name. For this example we are going to use PetriNetsModels.

Now, right-click on the PetriNetsModels project and click New → Other. Search for the Example EMF Model Creation Wizards. Inside it you'll find a project named StandardPetriNets Model. The name might be different depending on the name of the package that you've provided your metamodel in Sec. 2, Fig. 5.

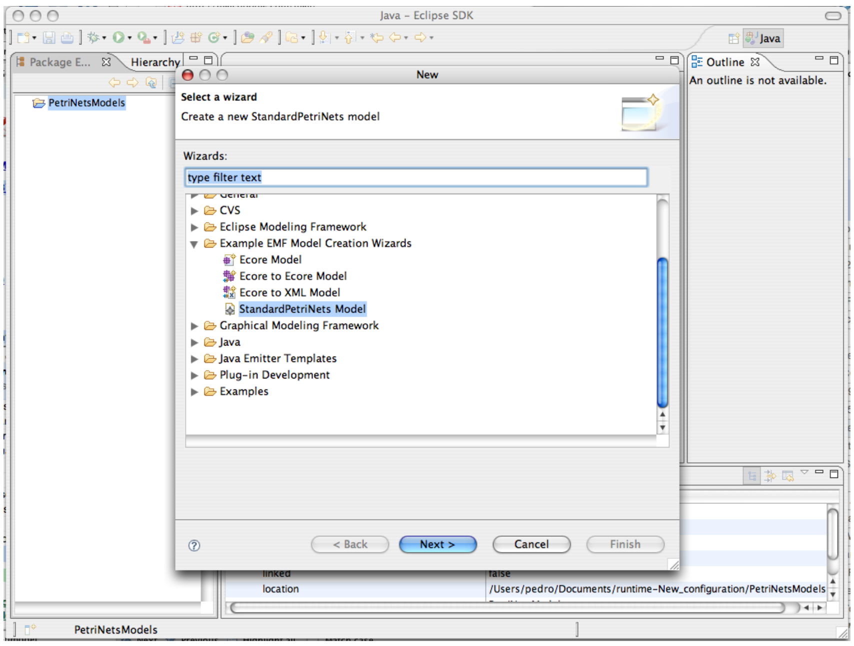


Figure 10: Initializing a new Petri Nets model

Fig. 10 shows how your Eclipse instance should look like while creating a StandardPetriNets Model. Press Next and give a name to your Petri Nets model. By convention, your model must have a name which extension ends with .nameOfPackage. If you followed this example until here, you should name you model ending in .standardpetrinets. We are going to use the name MyFirstNet.standardpetrinets. Click on Next and choose Petri Net as Model Object. Pressing Finish will open the editor that allow to create Petri Nets.

So far so good! Please close your second Eclipse instance now and let's define a graphical editor for Petri Nets models.

4 Providing the Model with a Graphical Interface

Before we start, you should configure Eclipse to use Java 5 compiler settings for your workspace (or at least for your GMF projects). In Window → Preferences → Java → Compiler options, make sure you have selected a Compliance Level of 5.0.

The first thing now is to create a new GMF project File → New → Project and choose New GMF Project under the Graphical Modeling Framework folder. Select Next provide the project with a name e.g. PetriNetsGMF and Next again. In the next screen, make sure you select the option Show dashboard view for the created project. After pressing Finish you should get a window equivalent to the one on Fig. 11. Should you close the GMF dashboard, you can reopen it via Window → Show View → Other... and select GMF Dashboard.

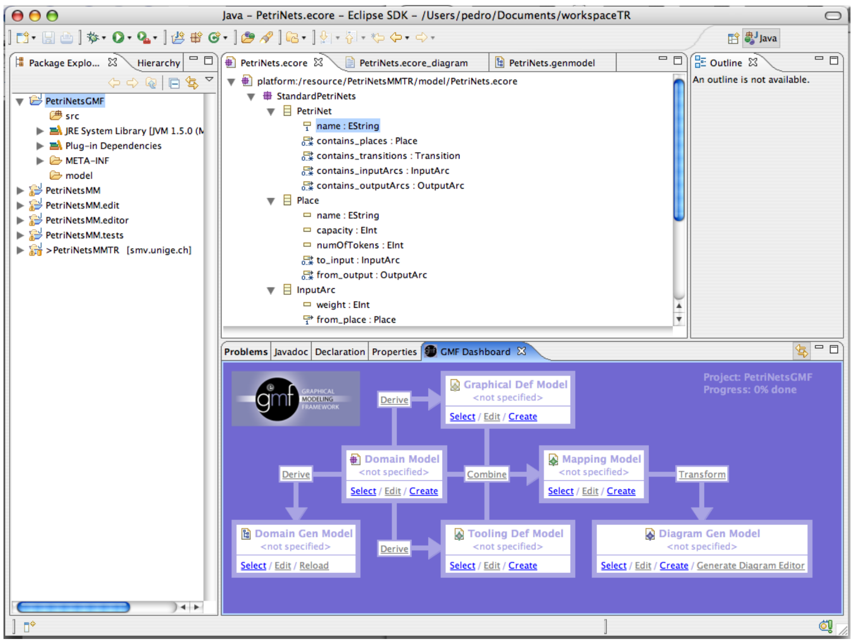


Figure 11: Creation of a new GMF Project

The dashboard will provide you with a guide line one how to create the entire GMF project. A GMF project follows the structure and workflow defined in Fig. 12. This means that, in order to have a valid GMF project we must:

- Develop a Domain Model. The previous sections of this report showed how to develop a domain model for Petri Nets.
- Develop a Graphical Definition. This model contains information related to the graphical elements that will appear in a GEF-based runtime, but have no direct connection to the domain models for which they will provide representation and editing.
- Developing a Tooling Definition. An optional tooling definition model is used to design the palette and other periphery (menus, toolbars, etc.).

- Developing a Mapping Model. A goal of GMF is to allow the graphical definition to be reused for several domains. This is achieved by using a separate mapping model to link the graphical and tooling definitions to the selected domain model(s).

- Create a Generator Model. Once the appropriate mappings are defined, GMF provides a generator model to allow implementation details to be defined for the generation phase.

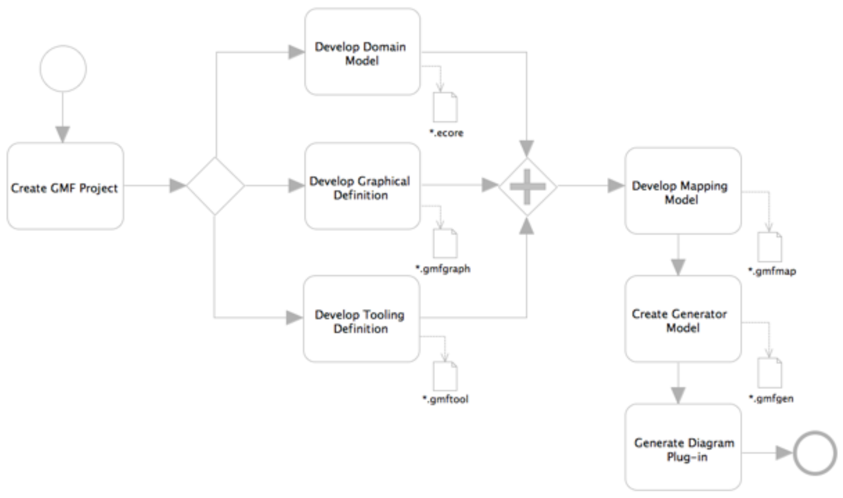


Figure 12: GMF Overview

Taking into account that we have already created a Domain Model, lets just select it by clicking on Select in the Domain Model box of the GMF Dashboard. Before you click on Select make sure you have selected, in the Eclipse Navigator, the project that contains the metamodel to be used (in our case PetriNetsMM project).

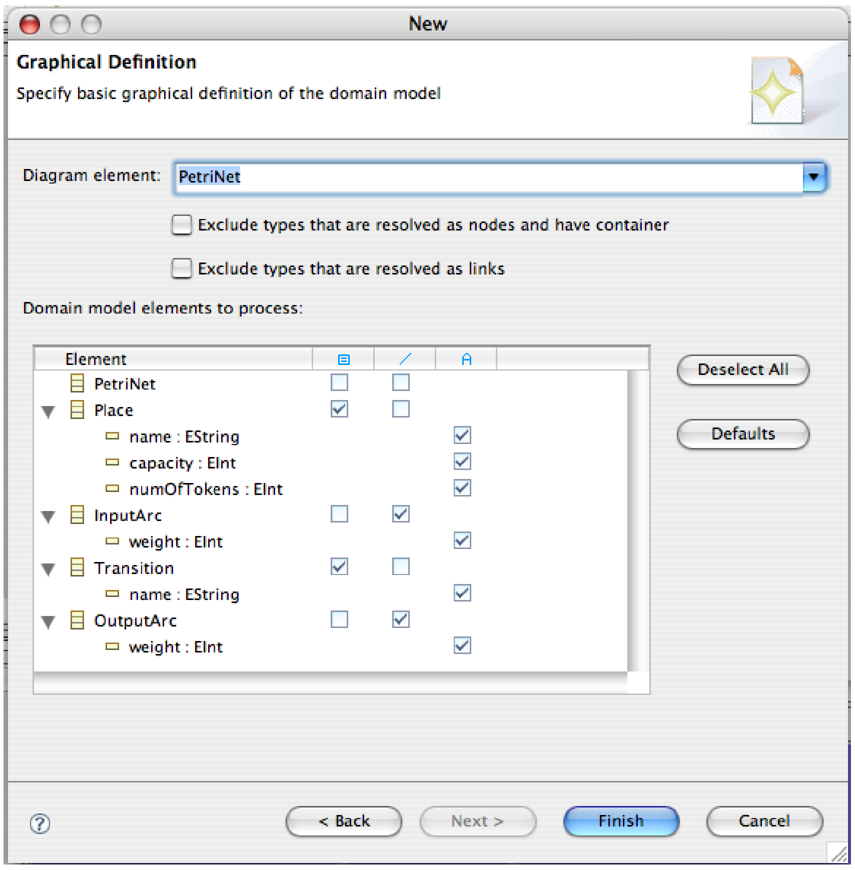
While thePetriNetsMM Project is still selected in the Navigator, click on Select in the Domain Gen Model box and choose the PetriNets.genmodel file.

Now click on the link Derive on the top of the Domain Model box (on the arrow that goes to Graphical Def Model). Select as Parent Folder the newly created project PetriNetsMM/model and as File name a name for your Graphical Definition Model that must have .gmfgraph as extension, e.g. PetriNets.gmfgraph. Clicking on Next provides you with a window for specifying the basic graphical definition of the domain model as shown in Fig. 13. Make sure you choose PetriNet as the Diagram element.

At this point you should have marked 50% on your progress mark from the dashboard.

Repeat the previous operation for the Tooling Definition Model by clicking on Derive below the Domain Model box.

Now you should be able to successfully generate the Grahical Definition Model and Tooling Definitions Model. For the moment we will leave them as generated but at some point we will come back in order to perform some graphical re-definitions.


New

Graphical Definition

Specify basic graphical definition of the domain model

Diagram element: PetriNet

☐ Exclude types that are resolved as nodes and have container

☐ Exclude types that are resolved as links

Domain model elements to process:

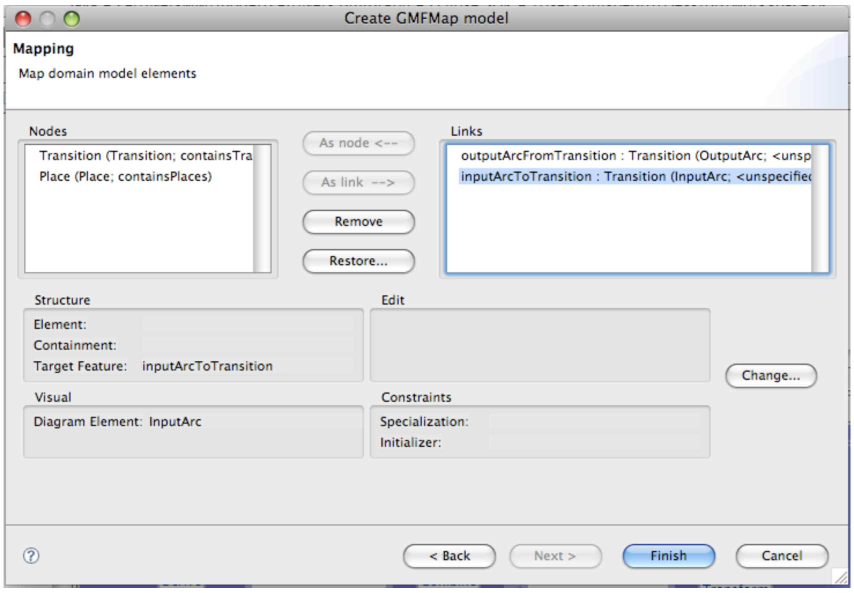
Element	Node	Link	Label
PetriNet	☐	☐	
Place	✓	☐	
name : EString			✓
capacity : EInt			✓
numOfTokens : EInt			✓
InputArc	☐	✓	
weight : EInt			✓
Transition	✓	☐	
name : EString			✓
OutputArc	☐	✓	
weight : EInt			✓

Figure 13: Specifying the basic graphical definition of the domain model

You should now generate the Mapping Model by clicking on Combine link that is located between the Graphical Def Model and Tooling Def Model boxes. Provide as parent folder PetriNetsMM/model and a filename that must end by .gmfmap. We are using the default name PetriNets.gmfmap. Click on Next and make sure that:

- the Domain Model URI is set to platform:/resource/PetriNetsMM/model/PetriNets.ecore and the selected class is PetriNet
- the Diagram Palette URI is platform:/resource/PetriNetsMM/model/PetriNets.gmftool
- the Diagram Canvas URI is set to platform:/resource/PetriNetsMM/model/PetriNets.gmfgraph
- In the Map domain model elements dialog you have a situation similar to Fig. 14.

Note that in Fig. 14, the Nodes list only has Place and Transition; that is becaus eyou only want these two to appear as graphical objects in the editor. Arcs should appear only as associations between places and transitions. On the other hand, in the Links list you only have one link for each type of arc. That is enough to create arcs.


Create GMFMap model

Mapping

Map domain model elements

Nodes
- Transition (Transition; containsTra
- Place (Place; containsPlaces)

Links
- outputArcFromTransition : Transition (OutputArc; <unsp
- inputArcToTransition : Transition (InputArc; <unspecifiec

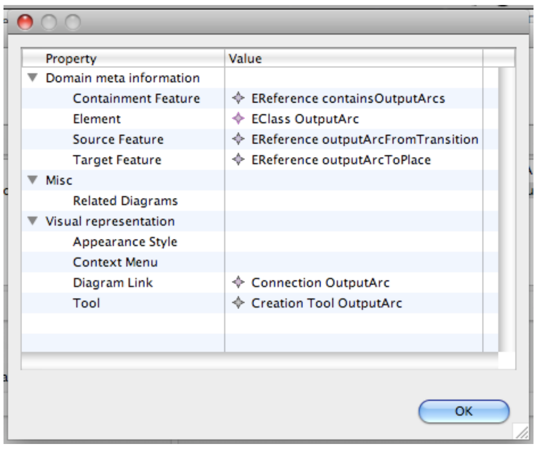
Structure
- Element:
- Containment:
- Target Feature: inputArcToTransition

Visual
- Diagram Element: InputArc

Constraints
- Specialization:
- Initializer:

Figure 14: Specifying the map domain model elements

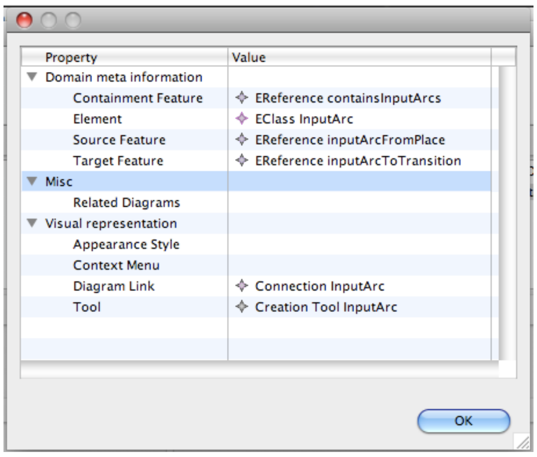
For each one of the elements presented in the Nodes and Links lists of the Map domain model elements dialog of Fig. 14, click on Change and check if the map is defined as expected for a Petri Nets Model. For example check that the outputArcFromTransition and inputArcToTransition are configured like in Fig. 15 and Fig. 16 respectively. Click Finish.



Property	Value
Domain meta information	
Containment Feature	EReference containsOutputArcs
Element	EClass OutputArc
Source Feature	EReference outputArcFromTransition
Target Feature	EReference outputArcToPlace
Misc	
Related Diagrams	
Visual representation	
Appearance Style	
Context Menu	
Diagram Link	Connection OutputArc
Tool	Creation Tool OutputArc

Figure 15: Output Arc Map Configuration

The Mapping created will still be missing a few crucial elements. Open the PetriNets.gmfmap file, and navigate the tree until the Node Mapping ¡Place/Place¿. Click New Child → Feature Label Mapping. Click on the node that is created, and in the Properties tab set Diagram Label to Diagram Label PlaceName and set Features to name : EString. Do the same for the Node Mapping ¡Transition/Transition¿



Property	Value
Domain meta information	
Containment Feature	EReference containsInputArcs
Element	EClass InputArc
Source Feature	EReference inputArcFromPlace
Target Feature	EReference inputArcToTransition
Misc	
Related Diagrams	
Visual representation	
Appearance Style	
Context Menu	
Diagram Link	Connection InputArc
Tool	Creation Tool InputArc

Figure 16: Input Arc Map Configuration

(but use instead the Diagram Label TransitionName). Check also that in Node Mapping ¡Place/Place¿ and in Node Mapping ¡Transition/Transition¿ the Tool property is set respectively to Creation Tool Place and Creation Tool Transition. Save your PetriNets.gmfmap file.

Go back to your GMF Dashboard (if the models don't show up in the dashboard, select the PetriNetsMM project in the Navigator) and click on Transform link, providing a name for the the Diagram Generator Model.

To finish the procedure, click on Generate diagram editor link in the Diagram Editor Gen Model box.

If you got until this step without any errors, you're now able to graphically create Petri Nets models. Let's try the editor. Click the Run button on the menu bar of Eclipse in order to launch another instance of Eclipse. You should be able to see the project you've created while generating Petri Nets models using the tree base editor (Sec. 3). Right-click on the PetriNetsModels project and New → Example. You'll be prompted to select a wizard. Choose PetriNetsGMF Diagram. Provide a name to your diagram, e.g. MyFirstPetriNetsModel.standardpetrinets_diagram, then click Next. Now provide a name for the model corresponding to the diagram, e.g. MyFirstPetriNetsModel.standardpetrinets. You should obtain an Eclipse workspace equivalent to the one in Fig. 17.

Now try to add two Places and two Transitions using the buttons available in the Palette. Save your diagram and double click on the file MyFirstPetriNetsModel.standardpetrinets. If tree view shows a model that does not correspond to what you have created in the graphical editor it means that the your mapping model generated automatically it is not correct. Close your second Eclipse instance again and go back to your PetriNets.gmfmap.

Browse the tree and find the Node Mapping for each one of the elements. For each one of them, in the properties view of Eclipse verify that the Tool property actually corresponds to the correct one. Refer to Fig. 18.

After fixing all mapping problems, you should redo the Transform and the Generate Diagram Editor processes. At the end you should be able to generate Petri Nets

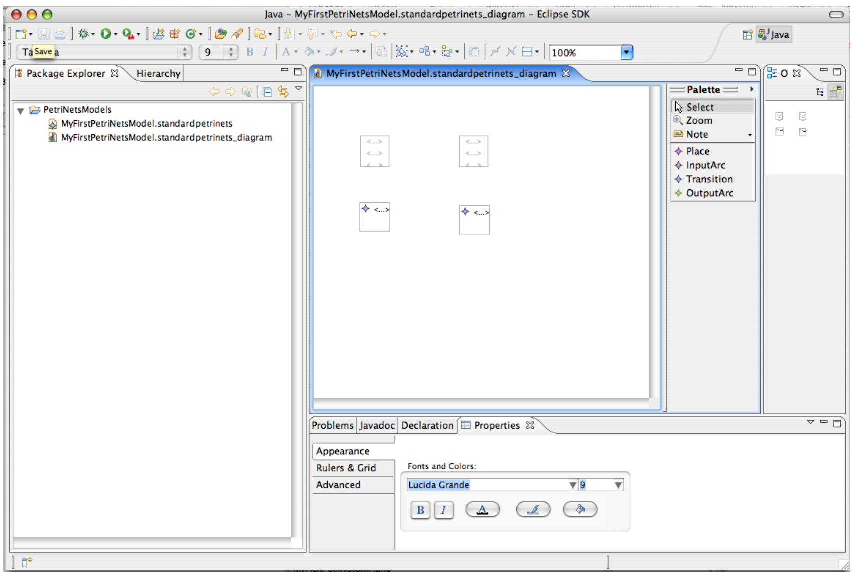
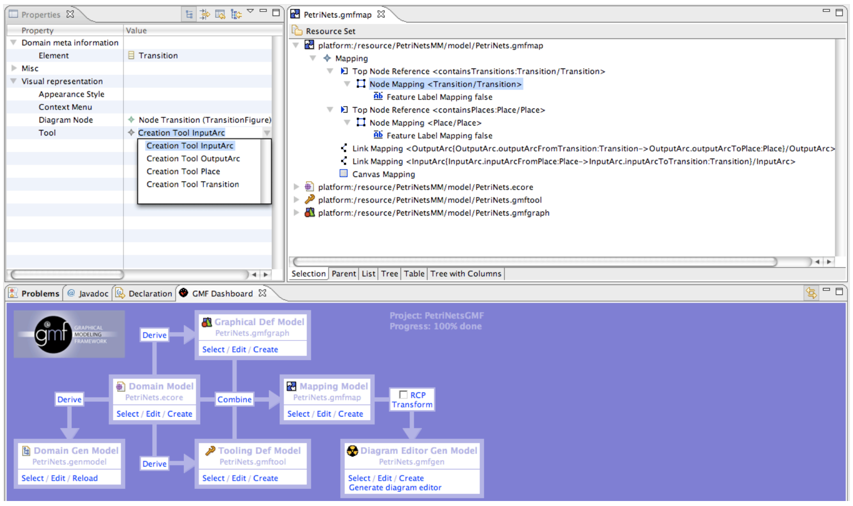

Java – MyFirstPetriNetsModel.standardpetrinets_diagram – Eclipse SDK
Package Explorer
Hierarchy
PetriNetsModels
MyFirstPetriNetsModel.standardpetrinets
MyFirstPetriNetsModel.standardpetrinets_diagram
MyFirstPetriNetsModel.standardpetrinets_diagram
Palette
Select
Zoom
Note
Place
InputArc
Transition
OutputArc
Problems
Javadoc
Declaration
Properties
Appearance
Rulers & Grid
Advanced
Fonts and Colors:
Lucida Grande
9
100%

Figure 17: Graphically creating Petri Nets



Properties

Property	Value
Domain meta information	
Element	Transition
Misc	
Visual representation	
Appearance Style	
Context Menu	
Diagram Node	Node Transition (TransitionFigure)
Tool	Creation Tool InputArc

```
Creation Tool InputArc
Creation Tool OutputArc
Creation Tool Place
Creation Tool Transition
```

PetriNets.gmfmap

```
Resource Set
platform:/resource/PetriNetsMM/model/PetriNets.gmfmap
  Mapping
    Top Node Reference <containsTransitions:Transition/Transition>
      Node Mapping <Transition/Transition>
        Feature Label Mapping false
    Top Node Reference <containsPlaces:Place/Place>
      Node Mapping <Place/Place>
        Feature Label Mapping false
    Link Mapping <OutputArc{OutputArc.outputArcFromTransition:Transition->OutputArc.outputArcToPlace:Place}/OutputArc>
    Link Mapping <InputArc{InputArc.inputArcFromPlace:Place->InputArc.inputArcToTransition:Transition}/InputArc>
    Canvas Mapping
platform:/resource/PetriNetsMM/model/PetriNets.ecore
platform:/resource/PetriNetsMM/model/PetriNets.gmftool
platform:/resource/PetriNetsMM/model/PetriNets.gmfgraph
```

Selection | Parent | List | Tree | Table | Tree with Columns

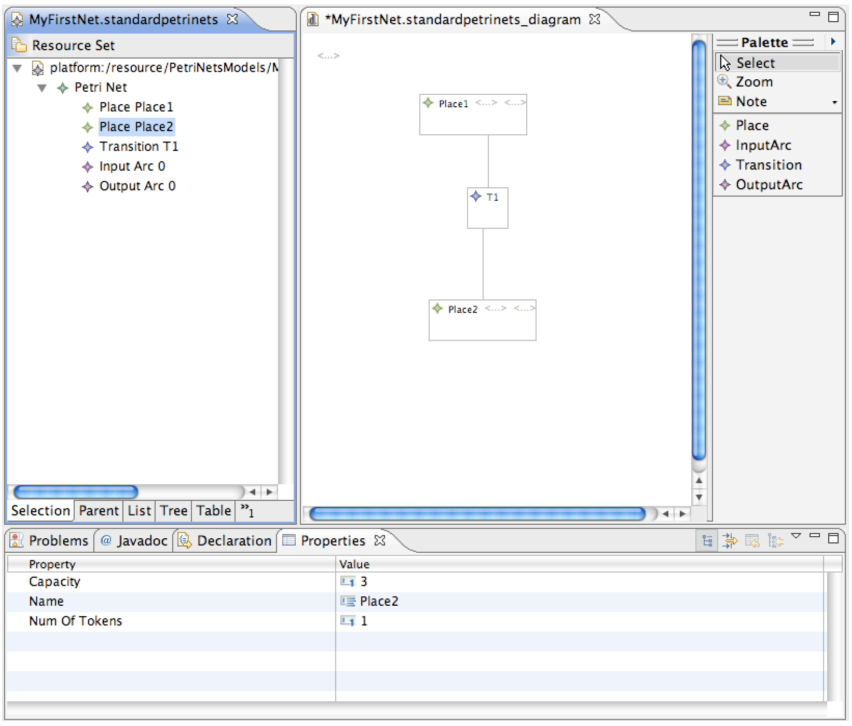
Figure 18: Modifying .gmfmap Tool mapping

models like presented in Fig. 19.

4.1 Customize Graphical Elements

Now that you've successfully created a graphical editor for Petri Nets Models lets start customization of the graphical editor.

The first thing we are going to do is to change the graphical layout of a place providing it with the appropriate concrete graphical syntax. In order to modify the graphical syntax of our Place entity from a UML Class-like rectangle to a circle proceed with the following steps:



Property	Value
Capacity	3
Name	Place2
Num Of Tokens	1

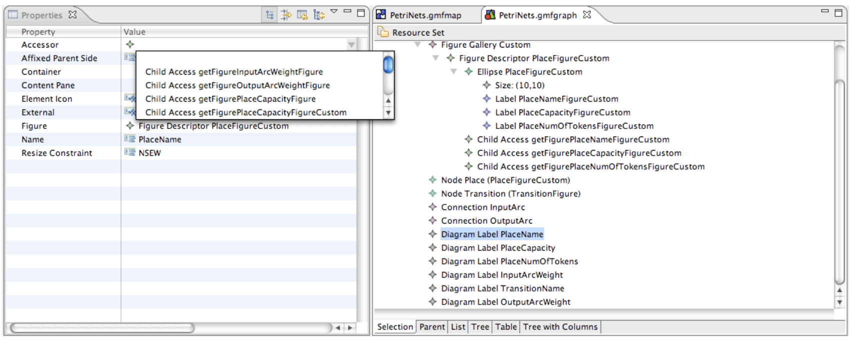
Figure 19: Creating Petri Nets models, graphically and tree-based editor

- Open your PetriNets.gmfgraph file:
- In the Canvas StandardPetriNets node right-click and New Child → Figure Gallery. Name your new Figure Gallery section as, for example, Custom Images. This step will allow us to easily distinguish between the figures created by default and the ones we are going to customize;
- right-click on the new Figure Gallery node and select New Child → Figure Descriptor; name the Figure Descriptor PlaceCustomFigure.
- right-click on the new Figure Descriptor node and select New Child → Ellipse;
- Name the Elipse as PlaceCustomFigure and modify the property Line Width to 3;
- Right-click on the Elipse and New Child → Size Point;
- Set both X and Y to the value of 10.
- In the root of your Canvas StandardPetriNets search for Node Place and replace the property Figure from Figure Descriptor PlaceFigure to the new one created Figure Descriptor PlaceCustomFigure;
- Inside the Ellipse, add as many Label nodes as there are inside the Rectangle node of the original default Figure Descriptor PlaceFigure. Name them in a

similar way as the original default ones, but not with the same names, because you want to be able to tell them apart.

- Inside the Figure Descriptor, add as many Child Access nodes as there are in the original Figure Descriptor PlaceFigure. Also here, use similar but different names from the original ones. Then, for each created Child Access node, set the Figure property to the corresponding Label node of the Ellipse. This is where having similar but different names is useful: if you used identical names as the originals, you would not be able to distinguish default labels from the ones you created.

- Further down in the tree, select the Diagram Label PlaceName node. Set its Accessor property to the corresponding Child Access node you created earlier, and set its Figure property to the Figure Descriptor PlaceCustomFigure you created. Then do the same also for the Diagram Label PlaceCapacity and Diagram Label PlaceNumberOfTokens nodes. See Fig. 20 for a screenshot.



Property	Value
Accessor	
Affixed Parent Side	
Container	
Content Pane	
Element Icon	
External	
Figure	Figure Descriptor PlaceFigureCustom
Name	PlaceName
Resize Constraint	NSEW

```
Child Access getFigureInputArcWeightFigure
Child Access getFigureOutputArcWeightFigure
Child Access getFigurePlaceCapacityFigure
Child Access getFigurePlaceCapacityFigureCustom
```

PetriNets.gmfmap | PetriNets.gmfgraph

```
Resource Set
  Figure Gallery Custom
    Figure Descriptor PlaceFigureCustom
      Ellipse PlaceFigureCustom
        Size: (10,10)
        Label PlaceNameFigureCustom
        Label PlaceCapacityFigureCustom
        Label PlaceNumOfTokensFigureCustom
      Child Access getFigurePlaceNameFigureCustom
      Child Access getFigurePlaceCapacityFigureCustom
      Child Access getFigurePlaceNumOfTokensFigureCustom
  Node Place (PlaceFigureCustom)
  Node Transition (TransitionFigure)
  Connection InputArc
  Connection OutputArc
  Diagram Label PlaceName
  Diagram Label PlaceCapacity
  Diagram Label PlaceNumOfTokens
  Diagram Label InputArcWeight
  Diagram Label TransitionName
  Diagram Label OutputArcWeight
```

Figure 20: Adding a new Figure and Descriptors to the gmfgraph

Again in the GMF Dashboard, redo the Transform operation and Generate Diagram Editor again. Make sure that you close your second Eclipse instance (if you didn't already) and re-open it via the Run menu.

You should now be able to create models in which a Place in the Petri Net is actually a circle with a labeled name inside like in Fig. 21.

In a Petri Net model we usually have the information concerning the number of tokens in each place (marking) and the weight on arcs. This information is normally explicit in the diagram. Until now we only have the Place Name and Transition Name as explicit information in the graphical representation. In order to add the arcs' weight and places' numberOfTokens we do the following:

- Open the PetriNets.gmfmap file

- right-click on the Link Mapping <inputArc{InputArcFromPlace:Place → inputArcToTransition:Transition/InputArc > and add a new Feature Label Mapping. Set the Diagram Label property to Diagram Label InputArcWeight, and the Features property to weight;

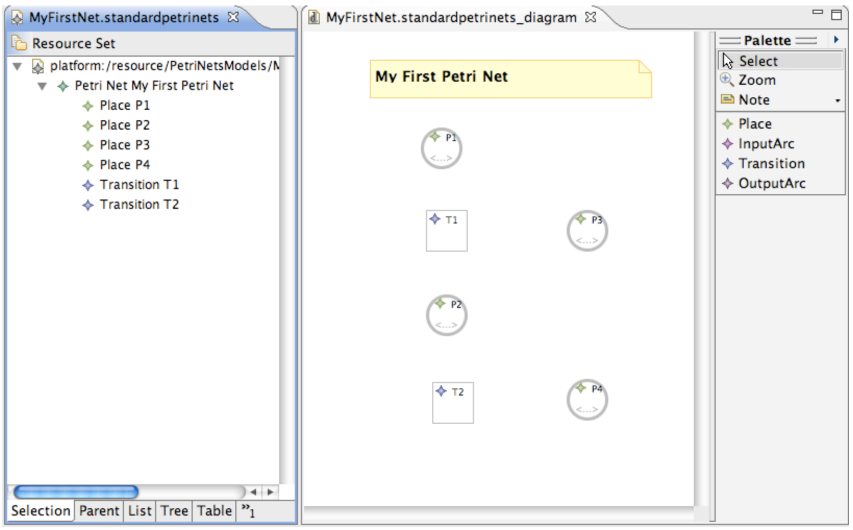


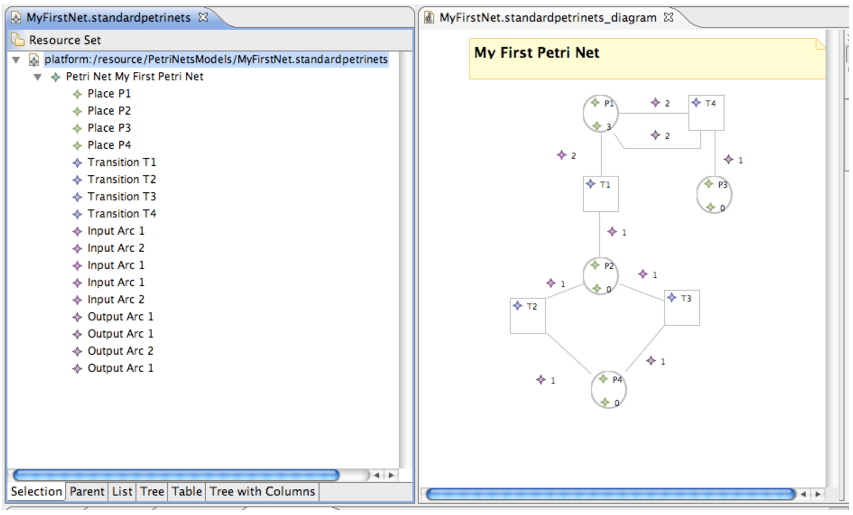
Figure 21: Petri Nets model with correct concrete syntax for the Petri Net Place

- Repeat the process similarly for the OutputArc;
- in the Top Node Reference<containsPlaces(Place/Place)> → Node Mapping <Place/Place> add a new Feature Label Mapping. Set Diagram Label to Diagram Label PlaceNumberOfTokens, and set Features to numberOfTokens.
- In the Feature Label Mapping that already existed in the Node Mapping <Place/Place> add to the Edit Pattern a label that you find suitable, "Name: " in this example.

Again perform both Transform and Generate Diagram Editor processes and launch the second Eclipse instance. By creating a Petri Nets with some Places, Transition, InputArc and OutputArc you should be able to get the diagram depicted in 22.

The arcs' visual representation is still not exactly what we expect it to be in a Petri Net. At this point we are able to create input and output arcs and link them to a place and a transition as needed. In a more exact Petri Nets graphical representation, an input arc should be represented by an arrow that connects Place to Transition directly, and an output arc should be an arrow from Transition to Place. In order to accomplish this let's proceed as follows:

- Open the PetriNets.gmfgraph file; Right-click on one of the nodes of the tree and select Load Resource; Enter platform:/plugin/org.eclipse.gmf.graphdef/models/classDiagram.gmfgraph in the dialog; This allows to load other pre-defined figures in your workspace. After loading this resource you will get another entry in the PetriNets.gmfgraph which you can browse to check how more elaborated figures are defined.
- Now open the default Figure Gallery, navigate to the Figure Descriptor InputArcFigure, and in the Target Decoration property of the Polyline Connection InputArcFigure choose Polygon Decoration ClosedArrow;



MyFirstNet.standardpetrinets

Resource Set

- platform:/resource/PetriNetsModels/MyFirstNet.standardpetrinets
  - Petri Net My First Petri Net
    - Place P1
    - Place P2
    - Place P3
    - Place P4
    - Transition T1
    - Transition T2
    - Transition T3
    - Transition T4
    - Input Arc 1
    - Input Arc 2
    - Input Arc 1
    - Input Arc 1
    - Input Arc 2
    - Output Arc 1
    - Output Arc 1
    - Output Arc 2
    - Output Arc 1

Figure 22: Petri Nets model with Arcs and Places initial marking explicitly in the diagram

- Repeat this process for the OutputArc;

If you now perform the Transform and Generate Diagram Editor operations on your GMF Dashboard again, and relaunch the second Eclipse instance, you can see how arcs created from a Place to a Transition have an arrow. The same thing happens if you create an arc from a Transition to a Place. To create arcs, leave the mouse pointer on a Place (or Transition); wait a moment until handles appear on the edge of the object; click the outgoing arrow and then drag the line until the target object. You should now be able to create a Petri Net model with a layout like the one presented in Fig. 23.

For more detailed informations and links to other related Eclipse projects please refer to [3].

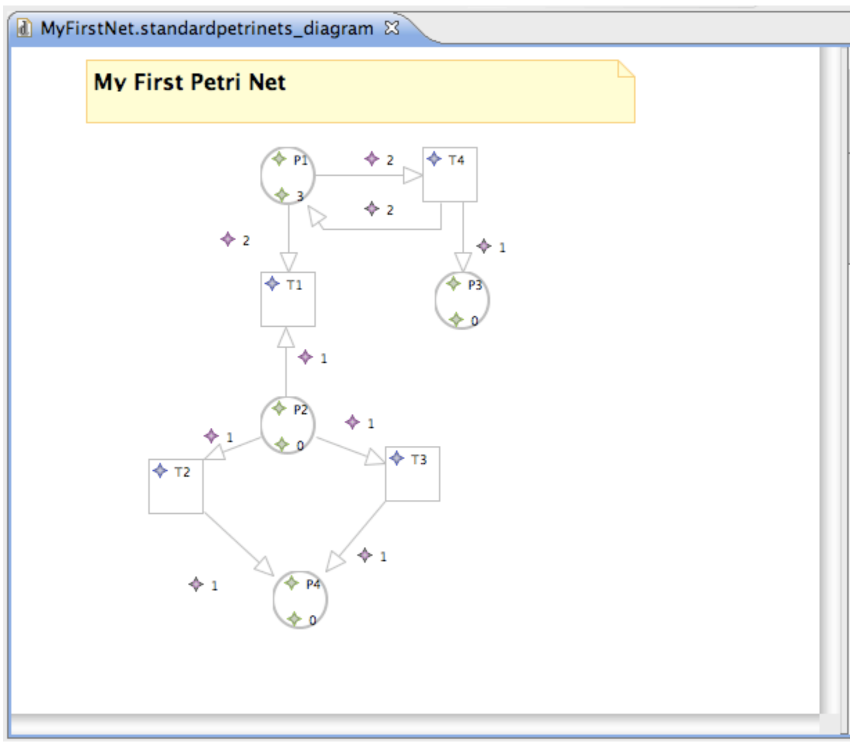


Figure 23: Petri Nets model the correct graphical syntax for Places and Arcs

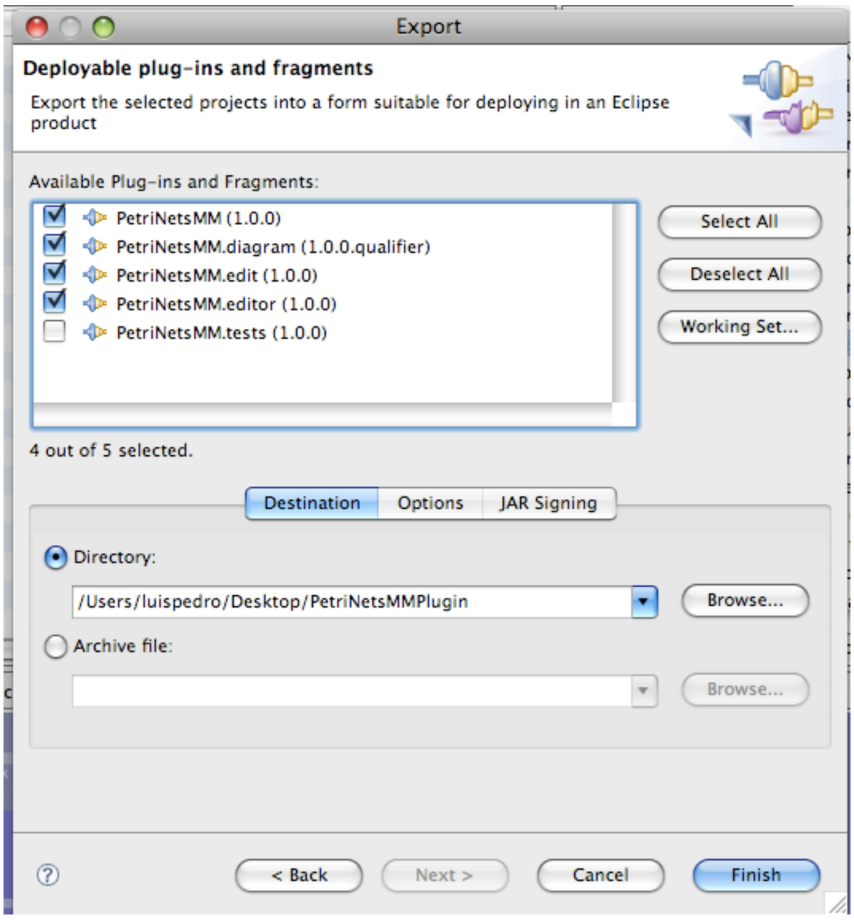
5 Optional Informations

5.1 Create an Eclipse Plug-in

If you want to create a default plugin for Eclipse allowing you to create Petri Nets models from the standard Eclipse instalation proceed with the following steps:

- On the PetriNetsMM.diagram project select Export;
- Choose Deployable plug-ins and fragments as in Fig. 24;
- Choose your own Destination Directory;
- Click Finish

Finally copy the generated .jar files to the Eclipse plugins directory and re-open Eclipse. You can now create petri nets models by selecting File → New → Other and searching for PetriNets diagram.


Export

Deployable plug-ins and fragments

Export the selected projects into a form suitable for deploying in an Eclipse product

Available Plug-ins and Fragments:

- [✓] PetriNetsMM (1.0.0)
- [✓] PetriNetsMM.diagram (1.0.0.qualifier)
- [✓] PetriNetsMM.edit (1.0.0)
- [✓] PetriNetsMM.editor (1.0.0)
- [ ] PetriNetsMM.tests (1.0.0)

4 out of 5 selected.

[✓ Directory:] /Users/luispedro/Desktop/PetriNetsMMPlugin

Archive file:

Figure 24: Select Available Plug-ins and Features

5.2 Optional Sources of Information

- GMF Documentation Site [4];
- Eclipse Modeling Framework book [5];
- Eclipse Development using the Graphical Editing Framework and the Eclipse Modeling Framework [6];
- GMF Documentation Site [7];
- Tutorial on how to create a BPMN Editor[8] ;

References

[1] Eclipse. Eclipse modeling framework, 2007. `http://www.eclipse.org/modeling/emf/?project=emf`.

[2] Eclipse Project. Eclipse graphical modeling framework.

[3] Eclipse Project. Gmf tutorial.

[4] Eclipse Project. Emf online documentation.

[5] Frank Budinsky, David Steinberg, Ed Merks, Raymond Ellersick, and Timothy J. Grose. *Eclipse Modeling Framework*. The Eclipse series. Addison Wesley, 2004.

[6] William Moore, David Dean, Anna Gerber, Gunnar Wagenknecht, and Philippe Vanderheyden. *Eclipse Development using the Graphical Editing Framework and the Eclipse Modeling Framework*. IBM RedBooks, February 2004. http://www.redbooks.ibm.com/abstracts/sg246302.html.

[7] Eclipse Project. Gmf online documentation.

[8] Eclipse Project. Gmf bpmn diagram tutorial.