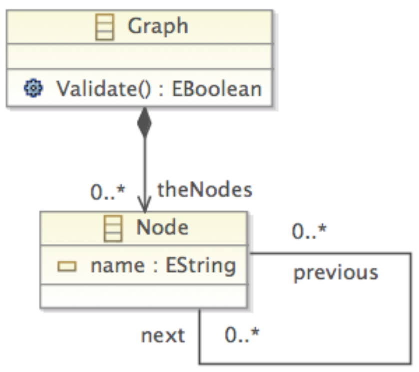
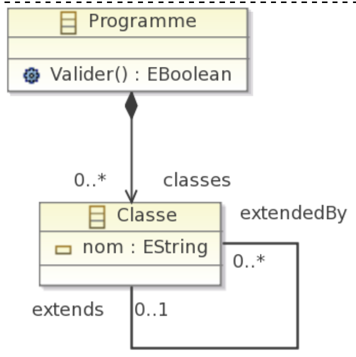
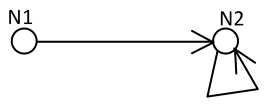
Models and Meta-models
– Eclipse Modelling Framework (EMF) –

A. Idani (akram.idani@univ-grenoble-alpes.fr)

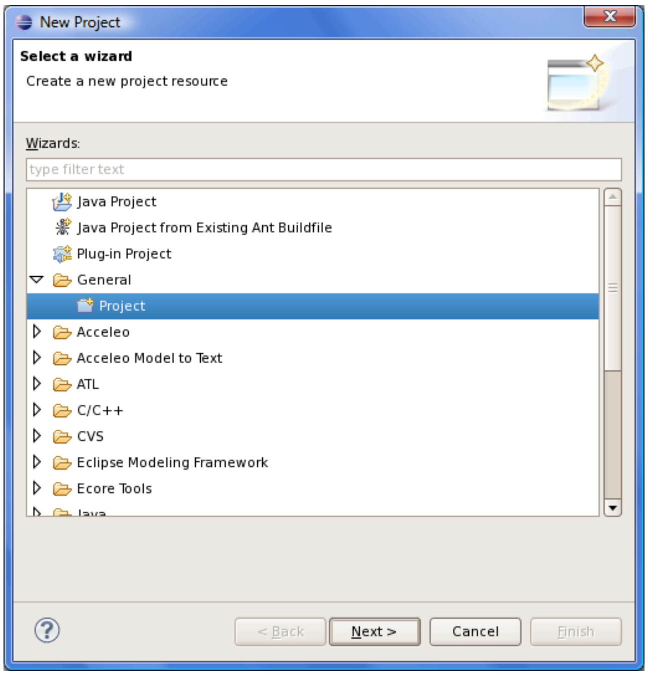
The objective of this lab is to present useful concepts for handling meta-models: creating, editing, editor generation and validation. We use for that the Eclipse Modeling Framework (EMF)

Case studies: modeling simple oriented graphs or class hierarchy in object-oriented languages.

	Domain: Oriented graphs	Domain: Object languages
Meta-model (M2 level) Abstract Syntax	 Graph Validate() : EBoolean 0..* theNodes Node name : EString 0..* previous next 0..*	 Programme Valider() : EBoolean 0..* classes Classe nom : EString extendedBy 0..* extends 0..1
Model (M1 level) Concrete syntax	 Concrete graphical syntax	Class Person ; Class Student extends Person ; Concrete textual syntax

Step 1: Empty project creation

1. Run eclipse
2. Click on: File / New / Project…
3. Choose General / Project
4. Name your project « Graphs » and click on Finish.

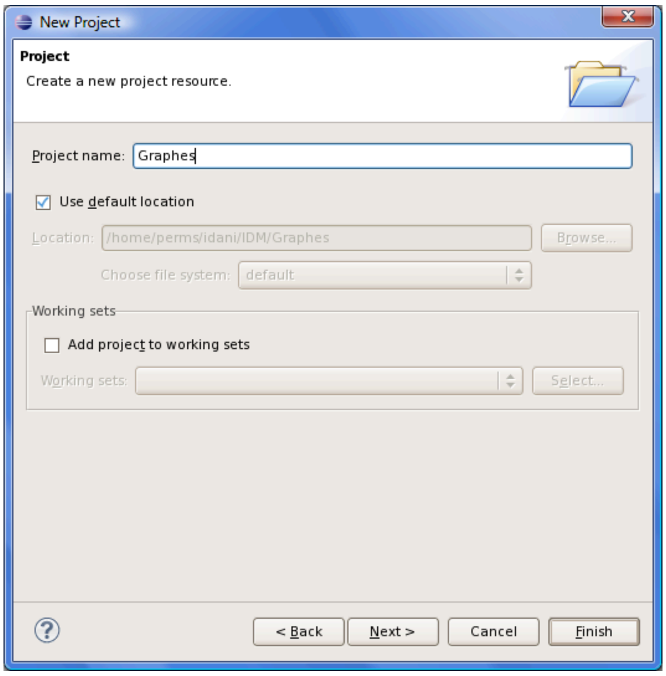

New Project

Select a wizard

Create a new project resource

Wizards:

type filter text


New Project

Project

Create a new project resource.

Project name: Graphes

☑ Use default location

Location: /home/perms/idani/IDM/Graphes

Choose file system: default

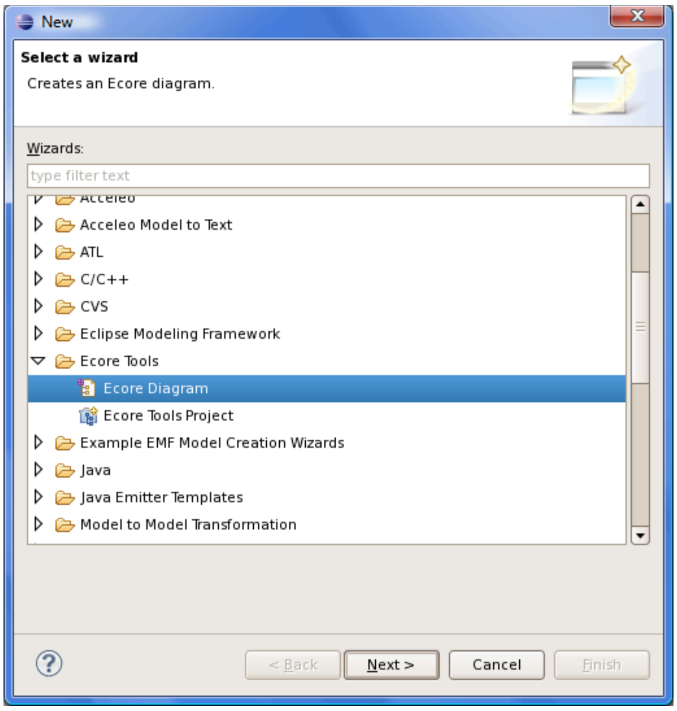
Working sets

☐ Add project to working sets

Working sets:

Step 2: Meta-mode definition

5. Right clic on project « Graphs » and choose: New / Other…
6. Choose Ecore Tools / Ecore Diagram
7. Name your meta-model « Graphs.ecore »

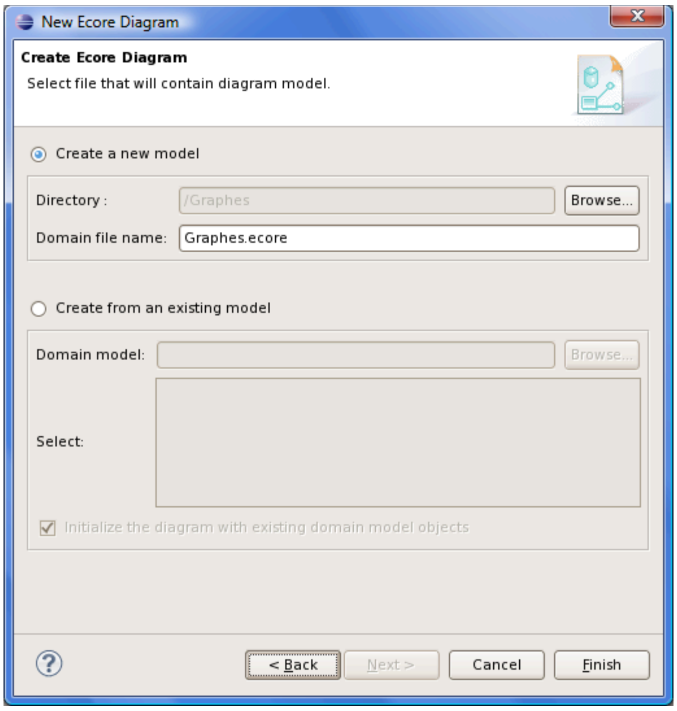

New

Select a wizard

Creates an Ecore diagram.

Wizards:

type filter text


New Ecore Diagram

Create Ecore Diagram

Select file that will contain diagram model.

◉ Create a new model

Directory : /Graphes

Domain file name: Graphes.ecore

○ Create from an existing model

Domain model:

Select:

☑ Initialize the diagram with existing domain model objects

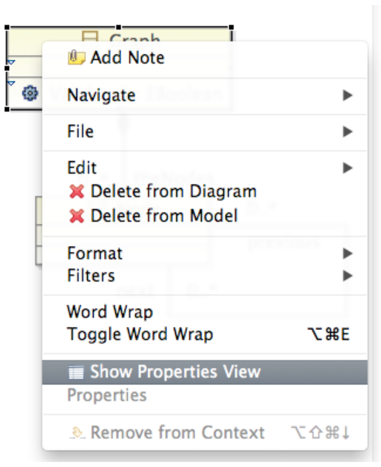
Note the existence of two files: Graphs.ecorediag and Graphs.ecore. The first one is manipulated by the EMF graphical editor and the second one is the xmi form of your meta-model which is handled by the EMF outline editor (also called TreeView editor).

Now you are able to:

1. create "graphically" a meta-model using the EMF graphical editor.
2. handle properties of ecore entities
3. read/modify the serialized (xmi) file of an Ecore meta-model.

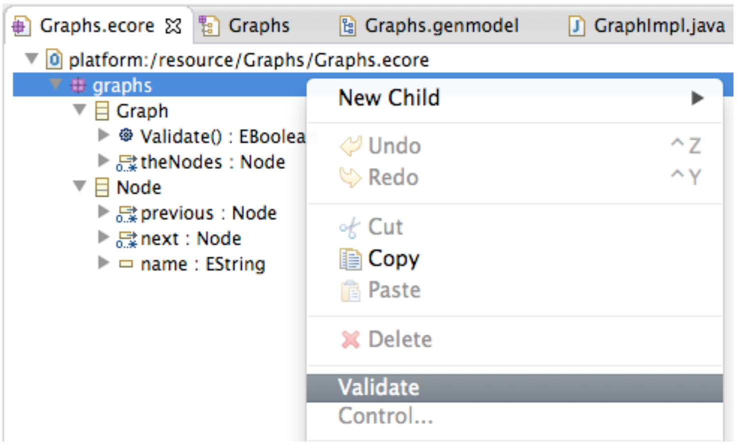
Create your meta-model and remark the simultaneous evolution of files Graphs.ecorediag and Graphs.ecore.

N.B. Your meta-model must be complete: typed attributes, correct cardinalities, useful operations, etc. To modify these information, use the eclipse "Properties View" accessible by doing right-click on an element of the meta-model.

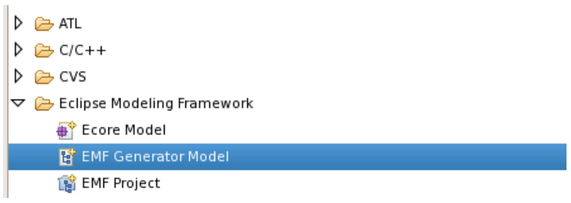


Step 3: code generation

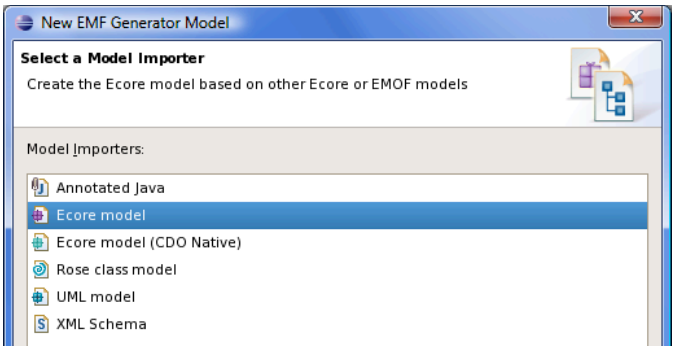
Before generating the Java code from your meta-model, you should validate it syntactically. Right-click on the root element of your meta-model in the EMF TreeView (like in Figure 4) and choose "Validate".



1. Right-click on file Graphs.ecore in your workspace,
2. Choose New → Other...
3. Select « Eclipse Modeling Framework / EMF Generator Model ».



4. select « Ecore Model » and click on « Next ».


New EMF Generator Model

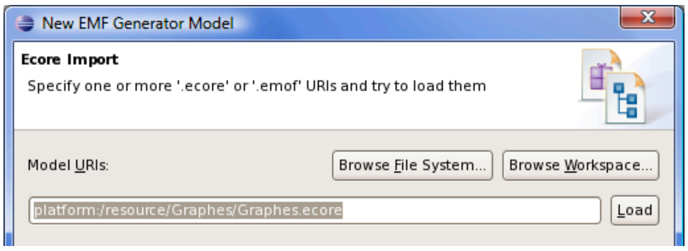
Select a Model Importer

Create the Ecore model based on other Ecore or EMOF models

Model Importers:

- Annotated Java
- Ecore model
- Ecore model (CDO Native)
- Rose class model
- UML model
- XML Schema

5. Click successively on « Load », « Next » and « Finish ».


New EMF Generator Model

Ecore Import

Specify one or more '.ecore' or '.emof' URIs and try to load them

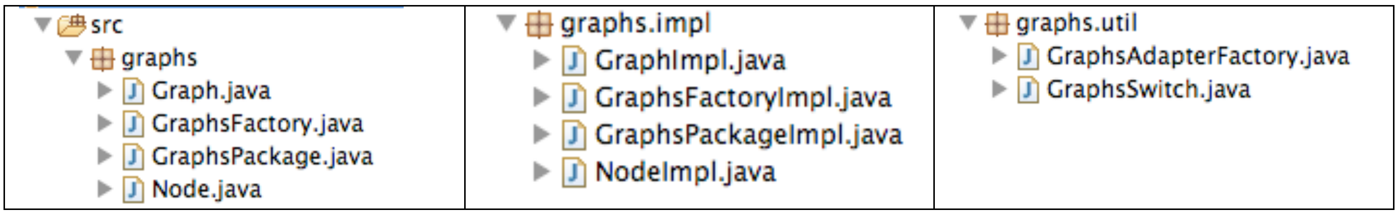
Model URIs: Browse File System... Browse Workspace...

platform:/resource/Graphes/Graphes.ecore Load

The steps above create in your workspace file "Graphs.genmodel".

6. Open this file and browse its properties
7. right click on the root element of this file
8. click on "Generate Model Code"

Finally, you get a new folder in your workspace called "src". It contains three packages: **`graphs`**, **`graphs.impl`** and **`graphes.util`**. These packages contain classes and interfaces to instantiate, in Java, your meta-model. Take a moment to see the content of the resulting Java code.



Method "**`Validate(): EBoolean`**" of meta-class graph should be implemented in order to verify whether a model is consistent with the meta-model constraints. For example, all instances of meta-class "**`Node`**" related to an instance of meta-class "**`Graph`**" must have distinct names (attribute "**`Name`**"). To encode this constraint, change its implementation in class "**`GraphImpl.java`**" as follows:

```
public boolean Validate() {
        EList<graphs.Node> nodes = this.getTheNodes();
        String Name1, Name2;
        for(int i = 0 ; i < nodes.size() ; i++){
                Name1 = (nodes.get(i)).getName();
                for(int j = 0 ; j < nodes.size() ; j++){
                        Name2 = (nodes.get(j)).getName();
                        if((i != j) && Name1.equals(Name2)) {
                                System.out.println(Name1 + " is defined several times.");
                                return false ;
                        }
                }
        }
        return true ;
}
```

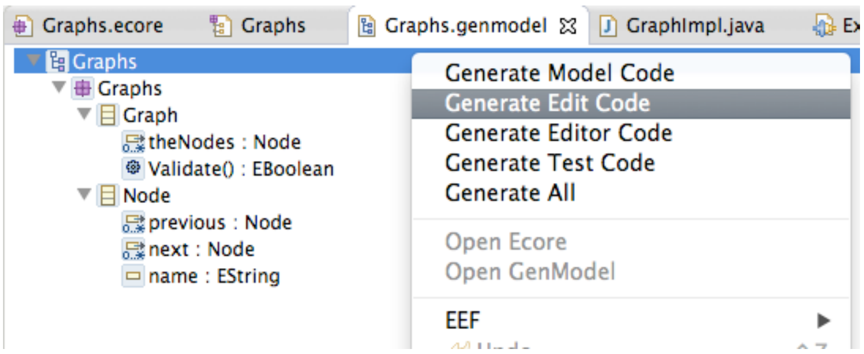
You can suggest another implementation of operation Validate.

Step 4: Create a model conformant to your meta-model

(a) Model editor generation

The editor generation is similar the Java code generation (review step 3).

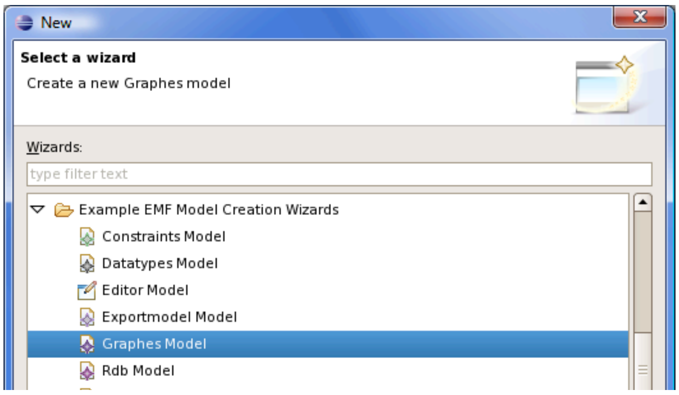
9. right click on the root element of "Graphs.genModel"
10. click successively on "Generate Edit Code" and "Generate Code Editor"



This produces Java code plugins for handling instances of your meta-model in your eclipse workspace. See "`Graphs.editor`" and "`Graphs.edit`" folders. The generated plugins can now be used in order to create models. To this purpose, right click on any of these plugins and select Run As/Eclipse Application. This will open another instance of Eclipse that we call Eclipse Runtime.

(b) Using the EMF model editor

In your Eclipse Runtime, create a general project named "`Test`".
Right click on this project and select "New → Others ...".
Under "Example EMF Model Creation Wizards" you will find the plugin "Graphs Model".



1. Select "Graphs Model"
2. Click on « Next » and name your model « `MyModel1.graphs` ».
3. Select « Graph » in section « Model Object ». This step specifies the root element that your model editor will handle (this is the meta-class Graph).
4. Click on "Finish".

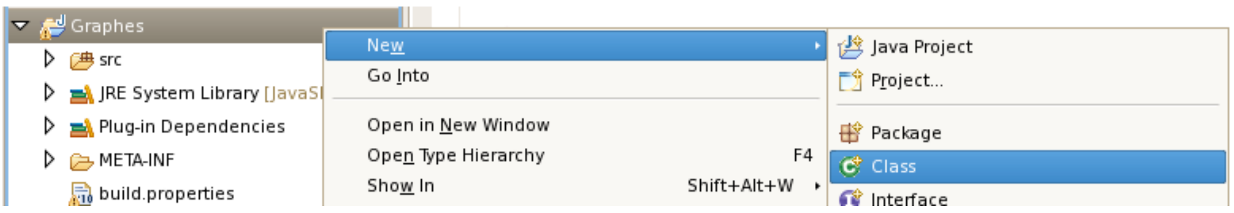
The EMF TreeView Editor is launched automatically. It allows you to instantiate your meta-model using a tree form which is serialized into xmi. Right click on "Graph", then New Child → Node. This creates an instance of meta-class Node. In the "View Properties" of this instance, associate to attribute "Name" value N1.

➔ Finish your model.

Step 5: Load a model editor and call user-defined operations

In this step you will create a Java program to load and read your model, and then call method "**Validate(): EBoolean**"

1. In your Eclipse Development, right-click on project « Graph »
2. Select New / Class



3. check « **public static void main(String[] args)** »
4. Name your class « **HandleGraph** »
5. Add the following java code to your main method of class « **HandleGraph** »:

```
// Initialize the model
GraphsPackage.eINSTANCE.eClass();

// Register the XMI resource factory for the .graphe extension
Resource.Factory.Registry reg = Resource.Factory.Registry.INSTANCE ;
Map<String, Object> m = reg.getExtensionToFactoryMap();
m.put("graphs", new XMIResourceFactoryImpl());

// Obtain a new resource set
ResourceSet resSet = new ResourceSetImpl();


// Get the resource
Resource res = resSet.getResource(URI.createURI("MyModel1.graphs"), true);

// Get the first model element and cast it to the right type
Graph myGraph = (Graph)res.getContents().get(0);


if(myGraph.Validate()){
        System.out.println("Your Graph is Valid.");
}else{
        System.out.println("Your Graph is NOT Valid.");
}
```

Your must add the following packages:

```
import java.util.Map;
import org.eclipse.emf.common.util.URI;
import org.eclipse.emf.ecore.resource.Resource;
import org.eclipse.emf.ecore.resource.ResourceSet;
import org.eclipse.emf.ecore.resource.impl.ResourceSetImpl;
import org.eclipse.emf.ecore.xmi.impl.XMIResourceFactoryImpl;
import graphs.Graph;
import graphs.GraphsPackage;
```

6. Right-click on class « **HandleGraph** »
7. Run As / Java Application.

- If method **`Validate():EBoolean`** returns a **`true`** value, then you obtain message « Your Graph is Valid ». Your model **`MyModel1`** does not violate the constraint.

- Modify « **`MyModel1.graphs`** » by adding a new instance of meta-class **`Node`** in which attribute **`Name`** is equal to N1.

- Run class « **`HandleGraph`** » again. You obtain the following messages:

```
N1 is defined several times.
 Your Graph is NOT Valid.
```

- The first line is produced by operation "**`Validate():EBoolean`**", and the second line is issued from « **`HandleGraph`** ».

Step 6: Load/Modify your model programmatically

In the main method of class « **`HandleGraph`** » add the following java instructions:

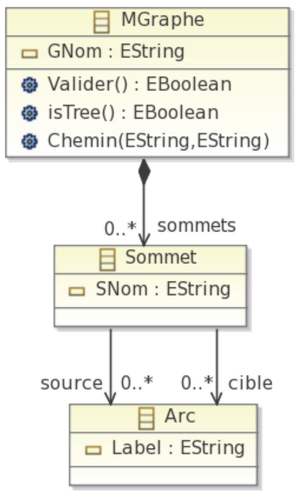
```
GraphsFactory factory = GraphsFactory.eINSTANCE ;
Node n3 = factory.createNode();
n3.setName("N3");
myGraph.getTheNodes().add(n3);
res.save(Collections.EMPTY_MAP);
```

This creates in **`MyModel1.graphs`** a node n3 in which attribute Name is equal to "N3". The save method allows to serialize the model.

➔ Verify in your EMF TreeView editor **`MyModel1.graphs`** contains node N3.

Exercise (to be evaluated)

1. Model multi-graphs and implement the following methods:
   – Validate: verifies that names (for nodes) and labels (for arcs) are unique
   – Path: prints a path between two nodes if it exists


MGraphe
GNom : EString
Valider() : EBoolean
isTree() : EBoolean
Chemin(EString,EString)
0..*
sommets
Sommet
SNom : EString
source
0..*
0..*
cible
Arc
Label : EString