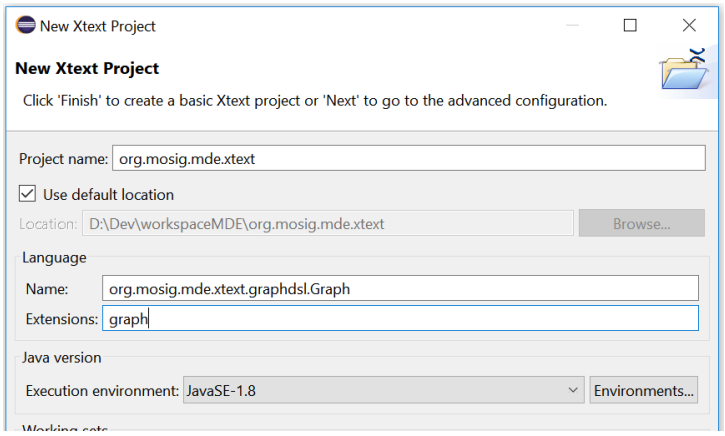
Xtext – OCL – Sirius TP

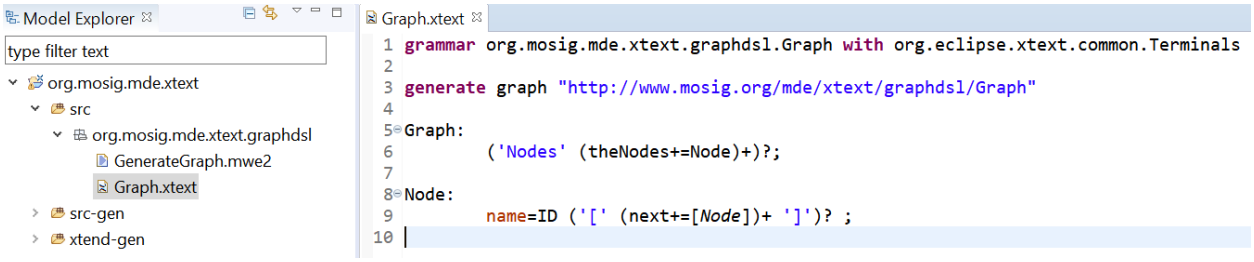
The goal of this TP is to define two concrete syntaxes (*graphical and textual*) relying on a language grammar.

Creation of an Xtext Grammar

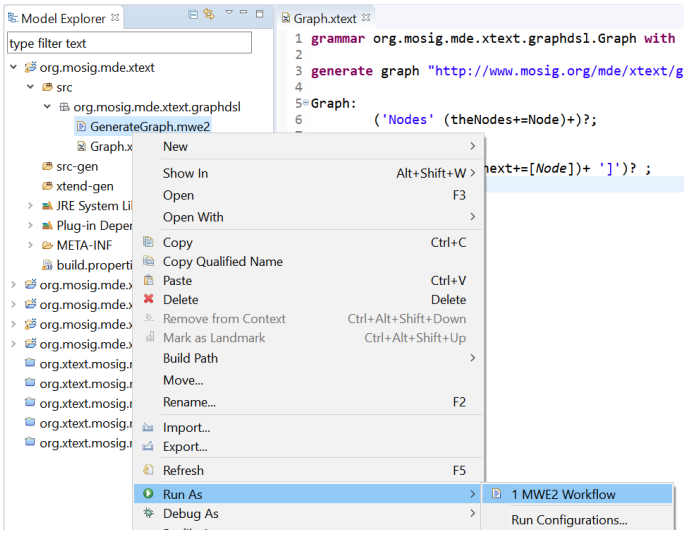
- Create a **Xtext Projet** (*org.mosig.mde.xtext*) with a language called *org.mosig.mde.graphdsl.Graph* with the extension *graph* and click Finish.

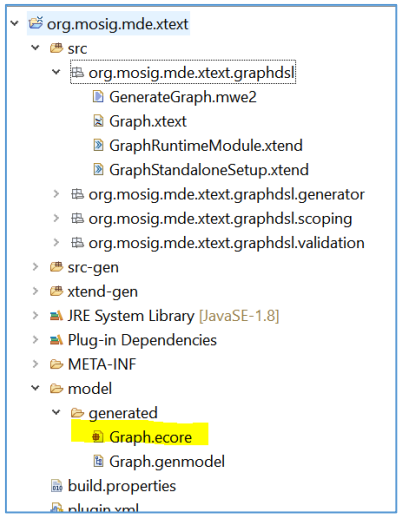


- Open the **Graph.xtext** file and write down the Graph grammar.



- Save and run the **GenerateGraph.mwe** file.

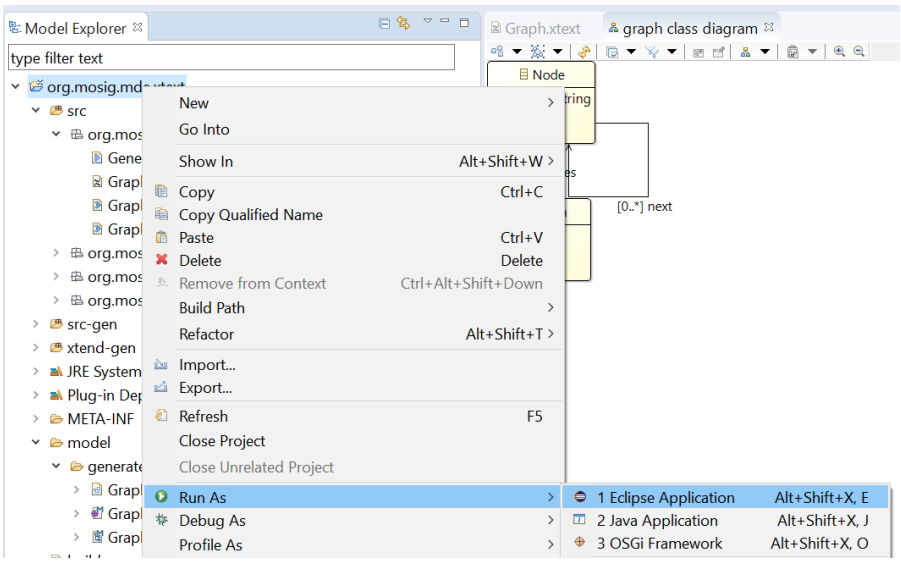




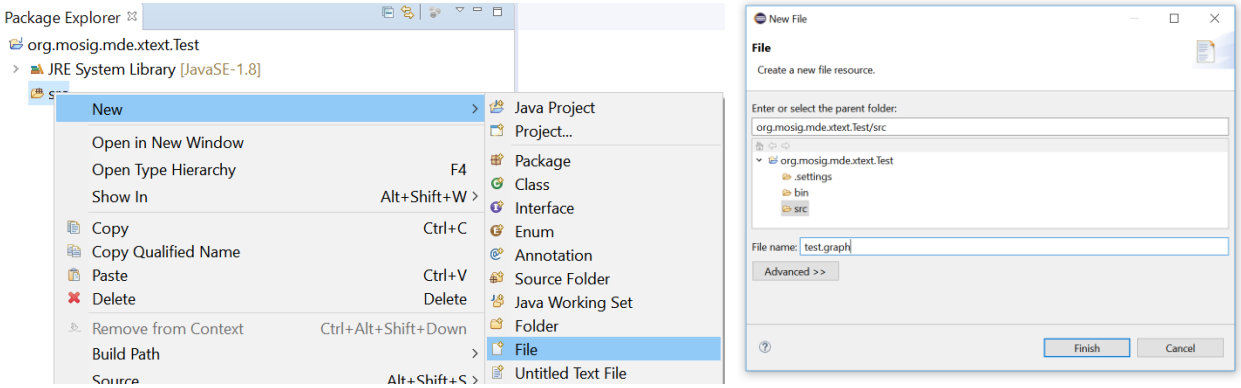
The execution generates among others, the **Graph.ecore MM** based on the previously defined grammar.

Xtext Language Test in a Runtime Environment

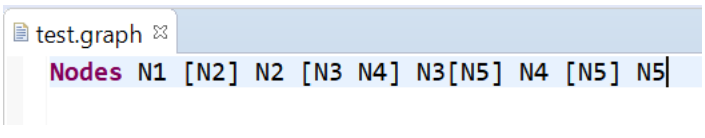
Right click on the project -> Run as -> Eclipse Application. This command launches another Eclipse instance containing our project. This means that in this Eclipse Application, we can already test our new graph DSL.

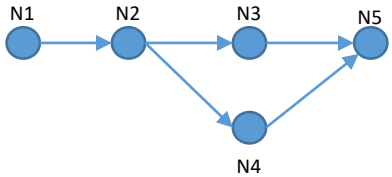


In this new environment, create a new Java Project (*org.mosig.mde.xtext.Test*). Inside src folder, create a new **test.graph** file. Say Yes if Eclipse ask you to show the Xtext perspective.



- In this new file we can now create graphs following the previously defined grammar. If you do *Crtl + Space* inside the fine, you will see that Eclipse propose you the elements from the grammar.
- Let's build the following graph using the textual language:

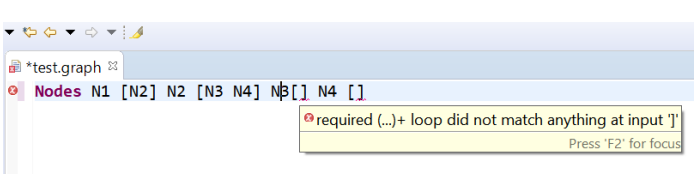


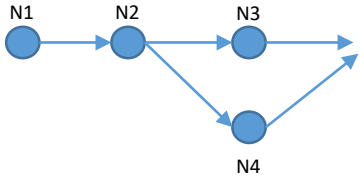


Note that the Xtext grammar forces to respect the correct syntax.

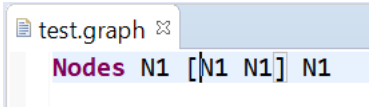
Adding OCL constraints to the ecore model

Note that the grammar constraint the good construction of the graph. For example, it avoids defining the following graph:

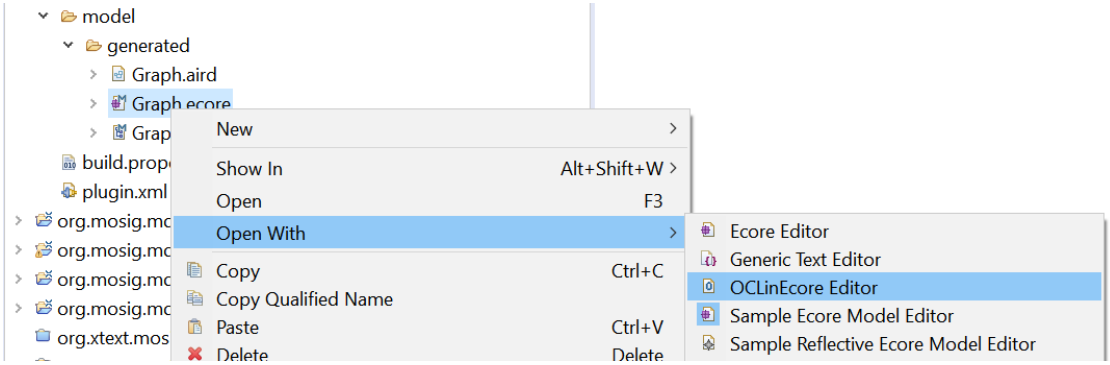




- However, we want to restrict even more the graph construction scenarios like the followings, but the grammar is not enough (no errors in the file).



- We will add **OCL constraints** to the ecore meta-model to avoid the aforementioned scenarios. Let's close the Test eclipse environment and go back to our main Eclipse instance in order to add these constraints.



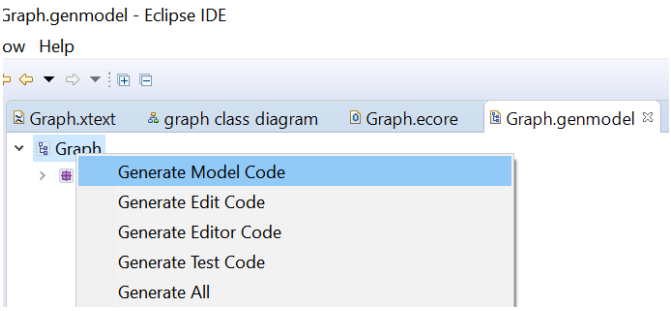
- Let's define the two following invariants:

Graph.xtext | graph class diagram | Graph.ecore

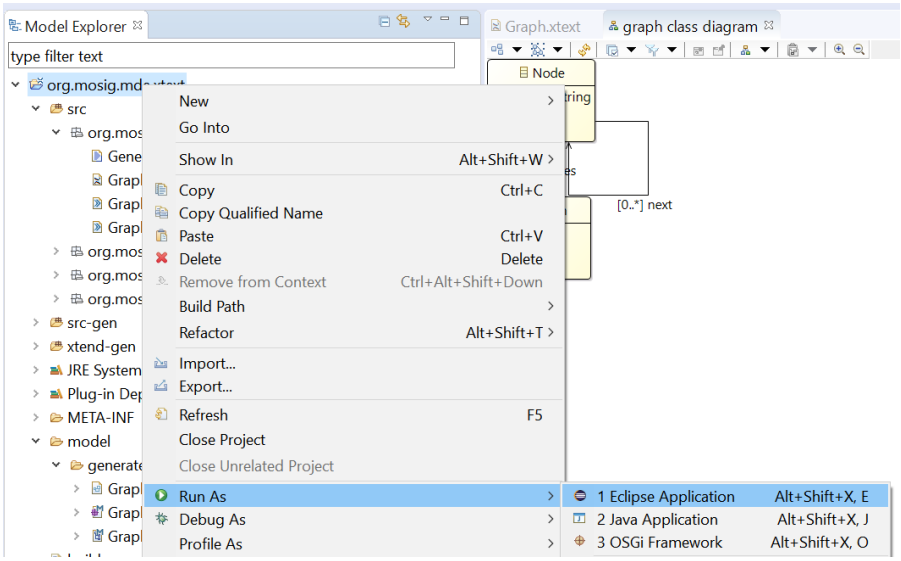
```
1  package graph : graph = 'http://www.mosig.org/mde/xtext/graphdsl/Graph'
2  {
3      class Graph
4      {
5          property theNodes : Node[*|1] { ordered composes };
6      }
7      class Node
8      {
9          attribute name : String[?];
10         property next : Node[*|1] { ordered !unique };
11         --Invariant 1 : Only one instance of a node within the graph
12         invariant UniqueNode: Node.allInstances()->forAll(n | n <> self implies n.name <> self.name) ;
13         --Invariant 2 : Only one arc between two nodes
14         invariant Next: self.next -> forAll(n | self.next->select(m | m = n)->size()=1) ;
15     }
16 }
```

Important: We just modified the ecore model. Therefore, if we re-execute the xtext generation, the changes will be lost.

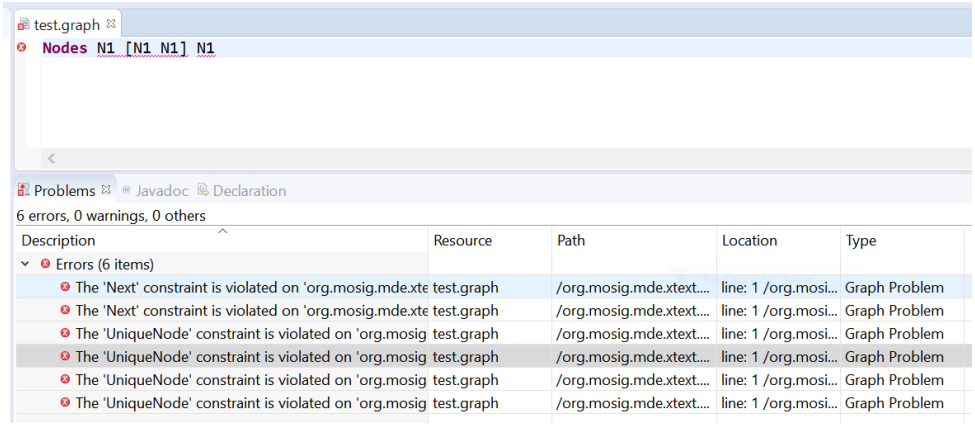
In order to support the defined invariants, we have to generate the code that supports them. For that, we will use the Graph.genmodel file (in the same folder that the ecore file). Right click on the root (Graph) -> Generate Model Code



- After that, we can re-launch our Test Eclipse application, and verify the OCL constraints are taken into account.

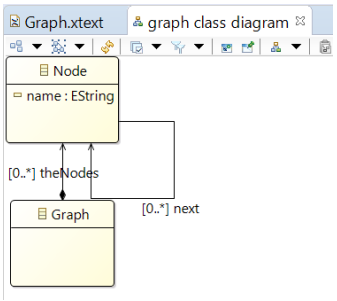


- Indeed, if we re-open and safe the incorrect graph implemented before, the errors corresponding to the OCL constraints are highlighted:

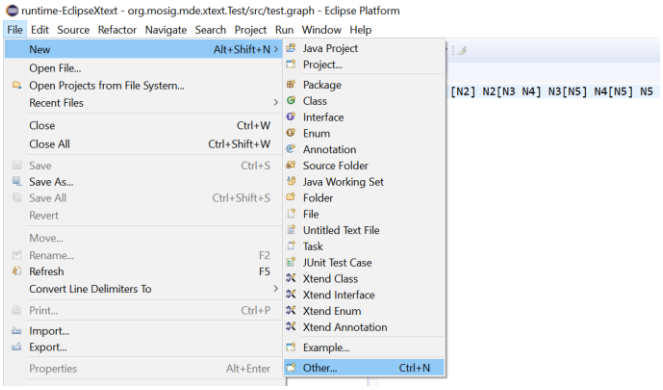


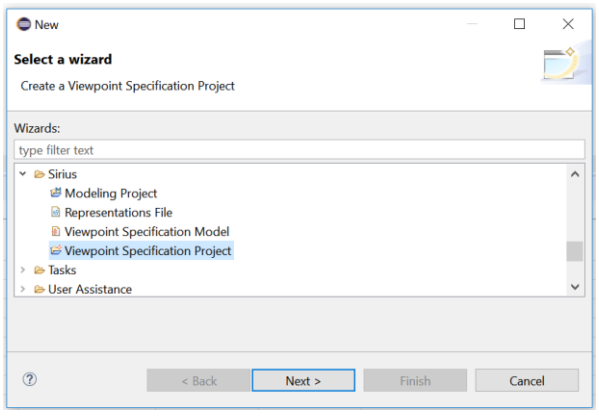
Adding a graphical concrete syntax to the graph models using Sirius

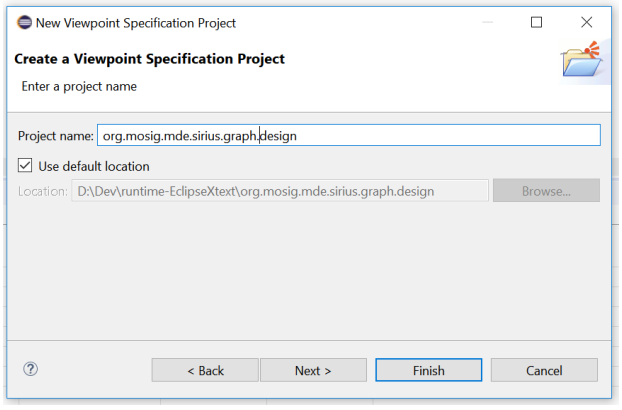
Now, we will focus on the ecore model. Note that if we want to visualize it graphically : Right click on the Graph.ecore -> Initialize ecore diagram -> Next -> Next. You will end up with the graphical representation of the ecore model. We want to do a similar thing but with our graph models. To accomplish that, we will use the Sirius framework : https://www.eclipse.org/sirius/

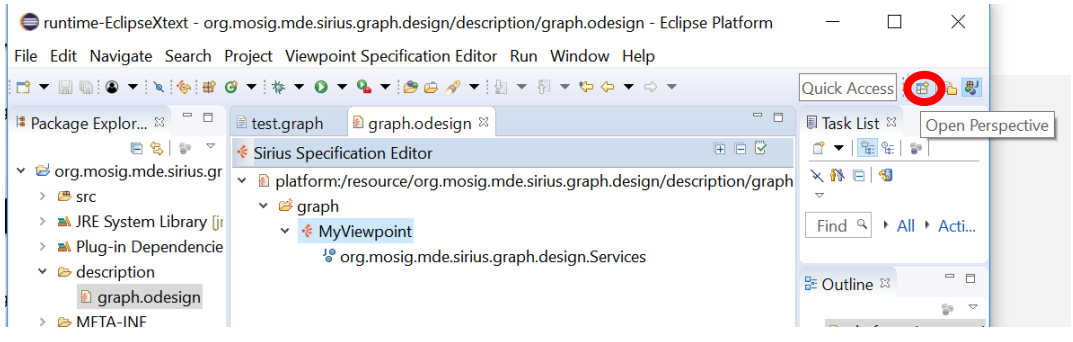


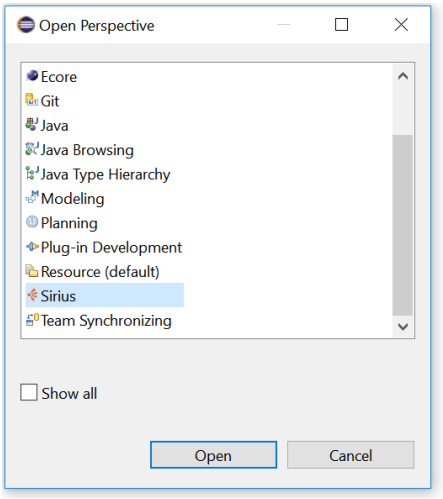
We will directly work on the **Test Eclipse Application** in orther to define a **graphical modeler** for our graphs. We create a new **Viewpoint Specification Project** (*org.mosig.mde.sirius.design*)





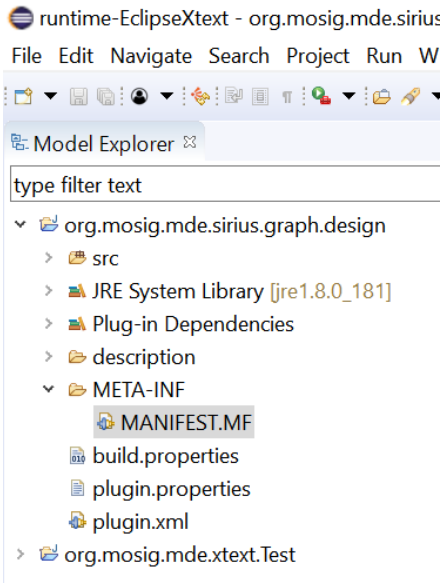


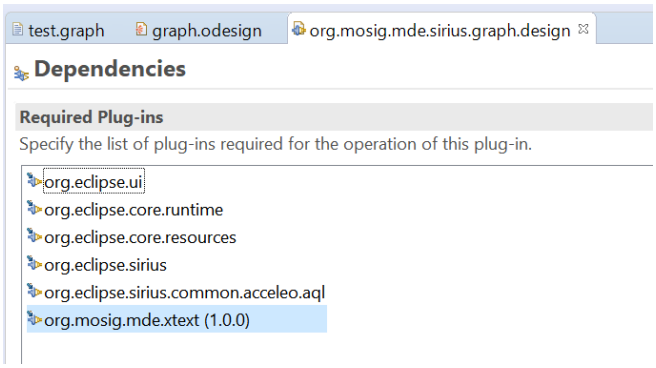




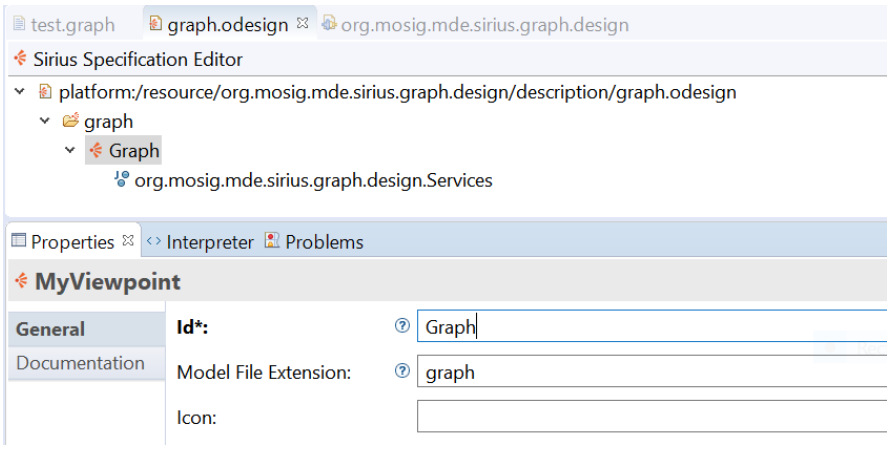
We will work with the **Sirius perspective** proposed by Eclipse.

Add our xtext project (containing the ecore meta-model) to the Sirius project. Open the MANIFEST file and **add the mde.xtext project to the project dependencies:**

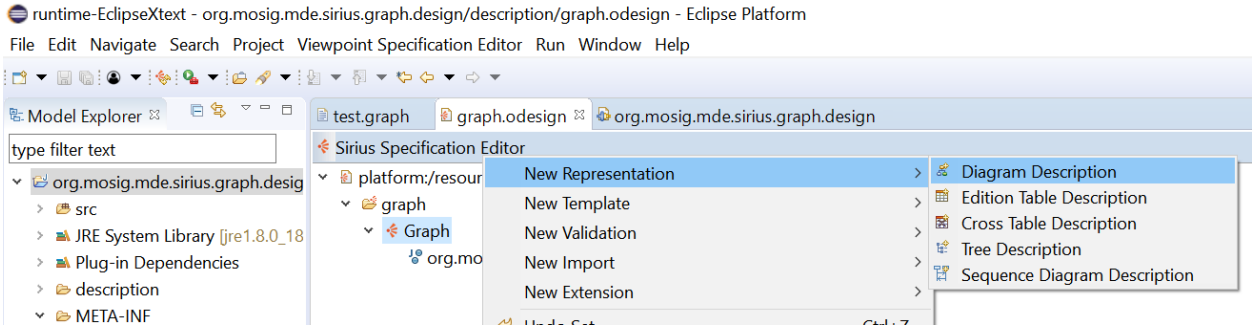




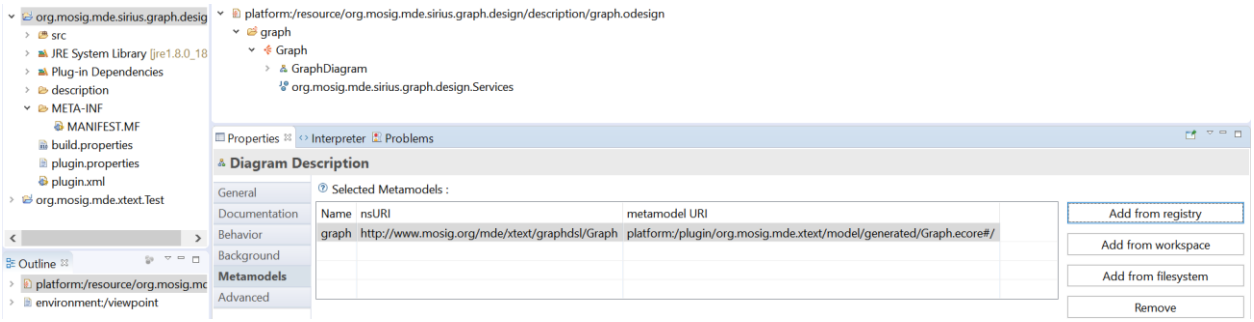
- Click on the **MyViewpoint** and change the name to Graph. Edit the **extension to graph** also.



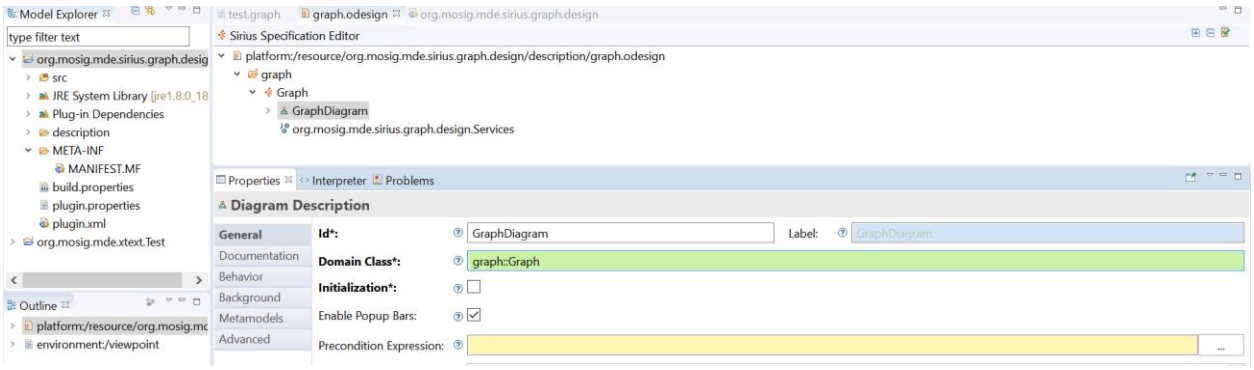
- In this viewpoint, then create a **New Representation** -> **Diagram Description**



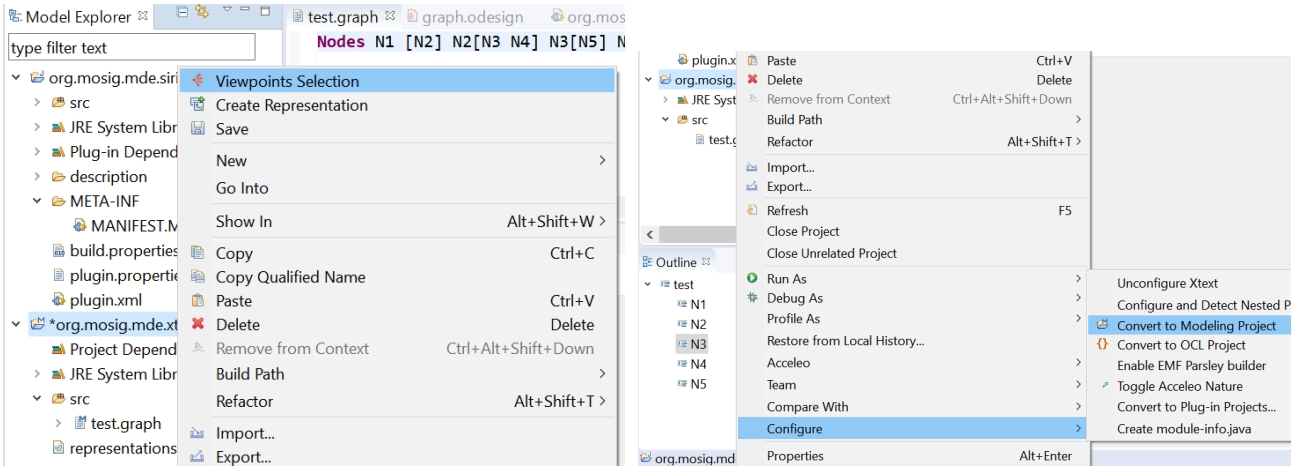
- This diagram has to be linked to the graph meta-model, in order to access easily to the meta-model concepts.

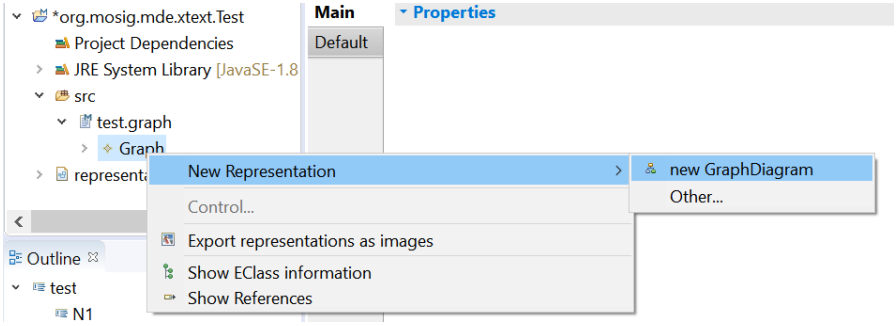


- Now, the **Doman Classes** are accessible with a Ctrl + Space. The diagram domain class is linked to the root of the meta-model : the meta-class Graph

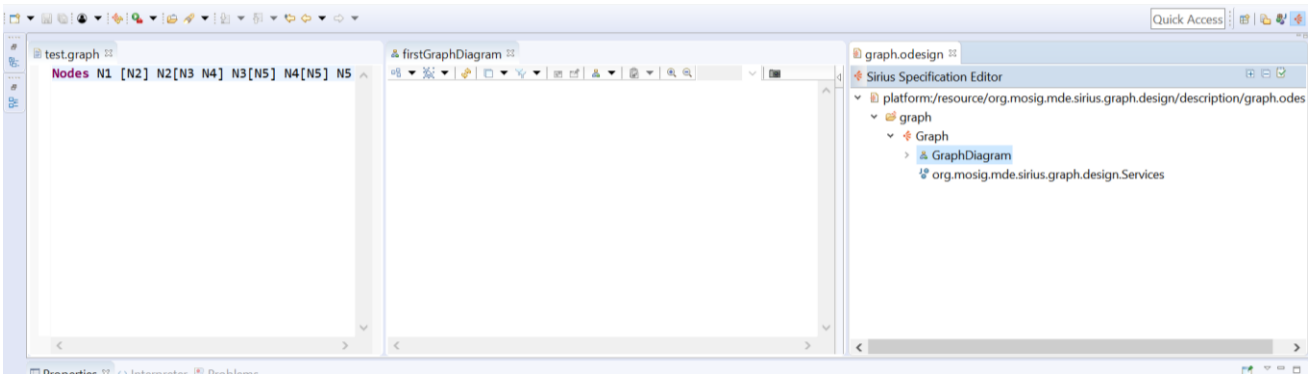


- We can now open a graph model (file with .graph extension) using the Sirius representation:

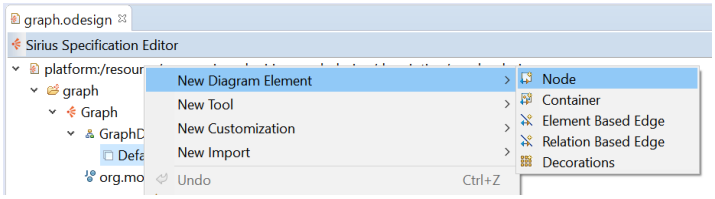


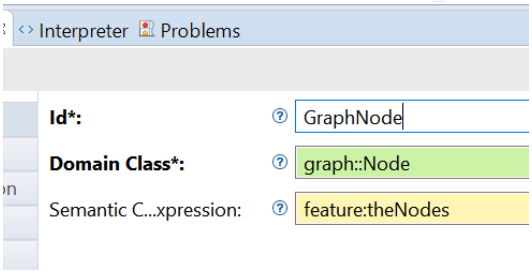


Note that **this first opened diagram is empty.** We have not specified yet the graphical elements for the nodes and arc. The idea is to define the graphical syntax for the meta-classes and associations in the .odesign file in order to have a graphical representation on the graph in the *firstGraphDiagram* file.

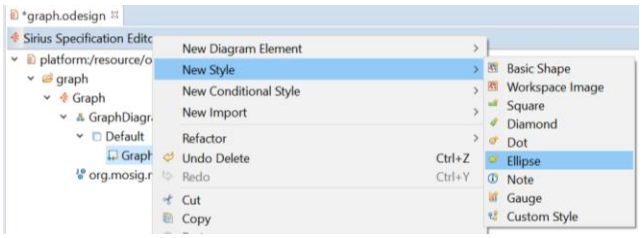


Create a graphical node and associate it to the graph::Node meta-class.

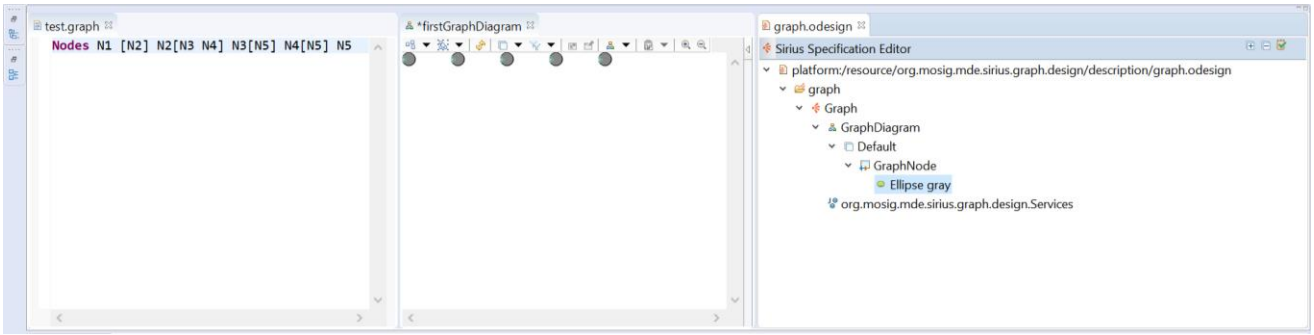




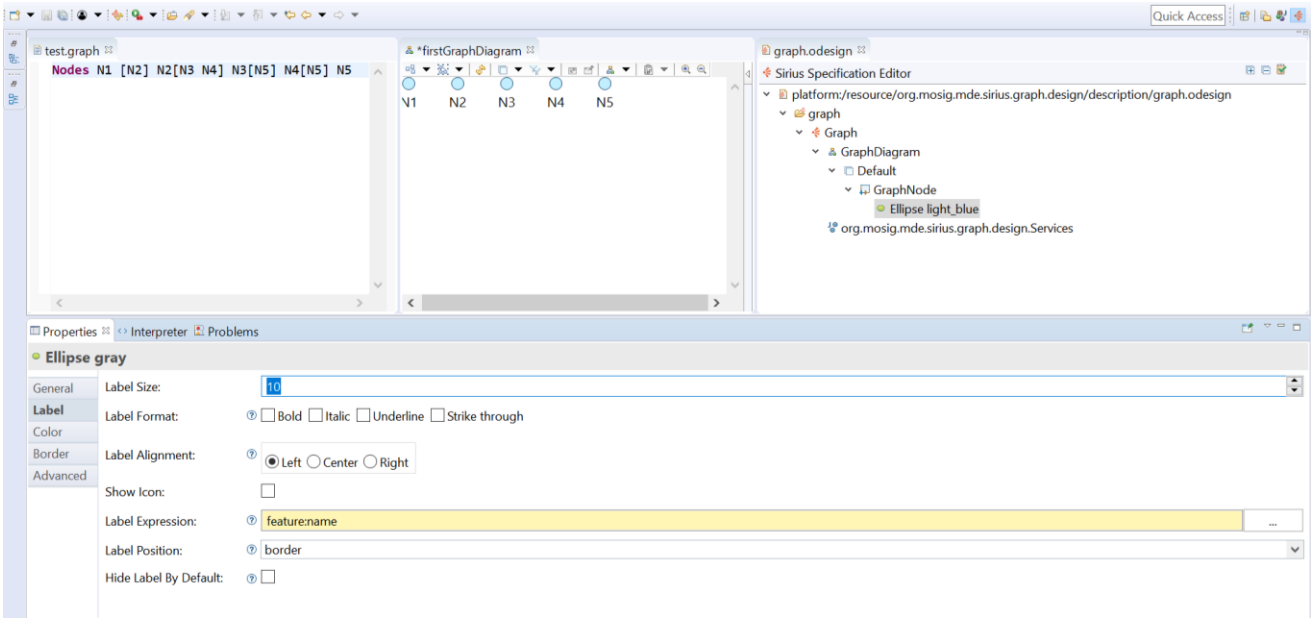
- Add a style to the node in order to define the graphical representation of the Node.



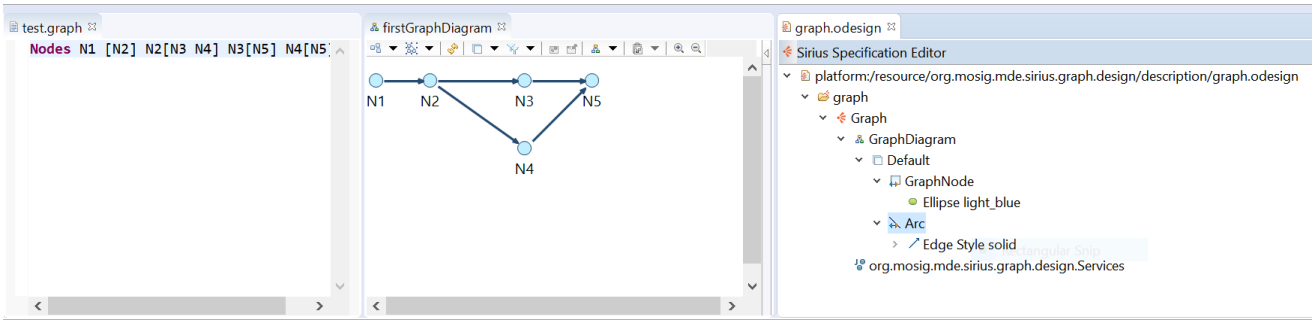
- When we save the .odesign file, the firstGraphDiagram shows the default style for ellipses nodes.



- If we modify the style of the elipses, we could have a result as following:



- Try to add now the arc that link the nodes (**new Relation Based Edge**) in order to have a result like the following:



Note that these views (textual and graphical) are synchronized. For example, if you remove a node in the textual file, the graphical representation will be automatically updated. Test that!

- The next step considering Sirius will be to **develop a Palette**. This mechanism will permit to create the nodes and arcs graphically:
  https://wiki.eclipse.org/Sirius/Tutorials/StarterTutorial#Add_a_Palette_for_edition_tools
  https://wiki.eclipse.org/Sirius/Tutorials/AdvancedTutorial#Start_extending_the_basic_modeling_tool