



Inria

TP: Un langage de graphes marqués exécutables avec Gemoc Studio



Joskel NGOUFO T .

30 Avril 2020

PLAN DE PRESENTATION

- ❑ Réseaux de Petri et graphes marqués
- ❑ Objectifs du TP
- ❑ Travail Pratique
- ❑ Conclusion et perspectives

RÉSEAUX DE PETRI ET GRAPHES MARQUÉS

-> Réseaux de Petri

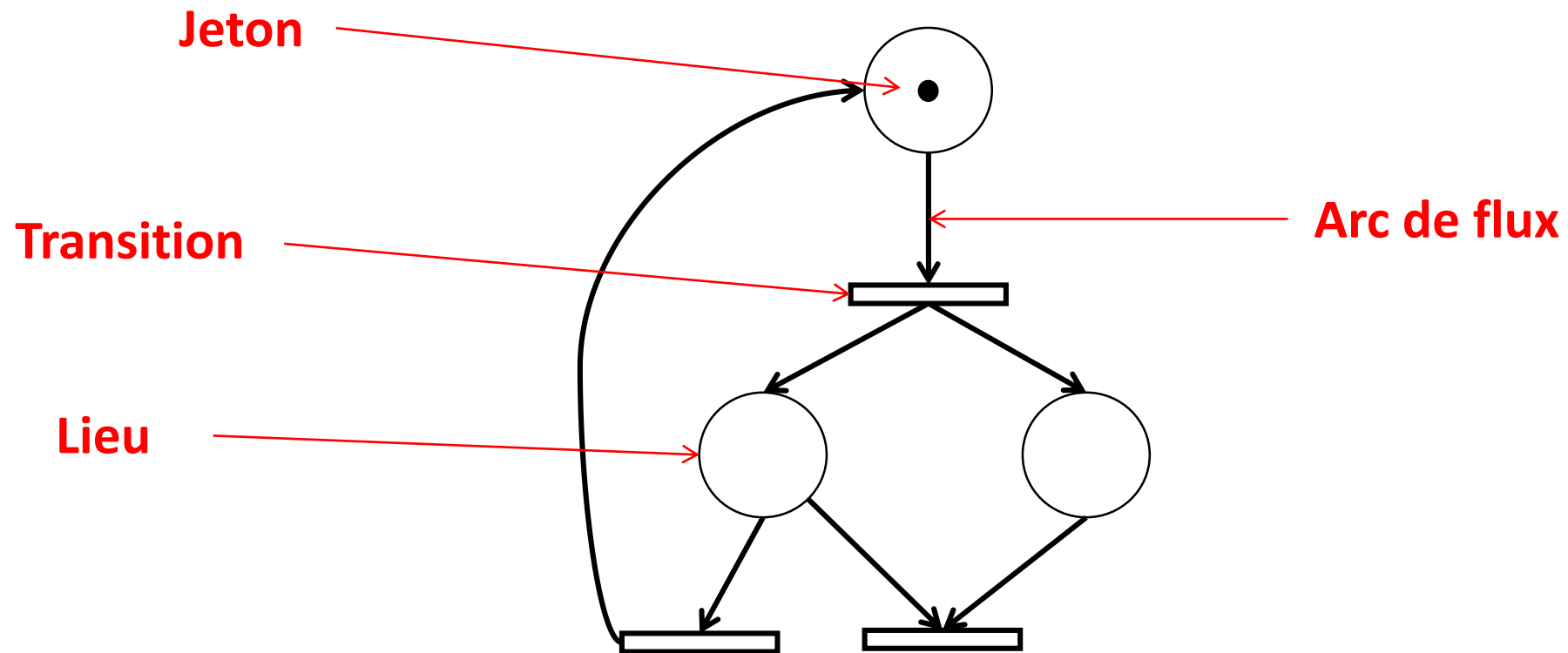
Un **réseau de Petri** est un modèle mathématique servant à représenter divers systèmes (informatique, industriel, ...). Il est défini formellement comme suit:

Un réseau de Petri est un tuple $N = (P, T, F, m_0)$ où

- P est un ensemble fini de lieux.
- T est un ensemble fini de transitions disjoint de P .
- F est une relation de flux $F \subseteq (P \times T) \cup (T \times P)$.
- m_0 est le marquage initial du réseau : un marquage $m : P \rightarrow \mathbb{N}$ représente un état du réseau de Petri et est visualisé comme une distribution de jetons sur les lieux. La transition t est activée dans le marquage m si et seulement si, pour tout $(p, t) \in F$, $m(p) > 0$.

RÉSEAUX DE PETRI ET GRAPHES MARQUÉS

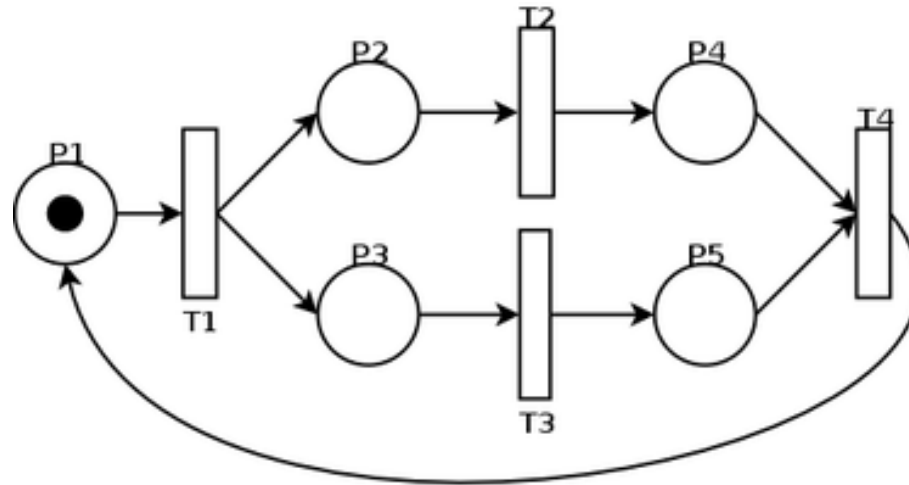
-> Exemple de réseaux de Petri



RÉSEAUX DE PETRI ET GRAPHES MARQUÉS

-> Les graphes marqués

- ❑ Un **graphe marqué** est un réseau de Petri dans lequel chaque lieu a exactement un arc entrant et exactement un arc sortant.
- ❑ Cela signifie **qu'il ne peut pas avoir de conflit**, mais **qu'il peut avoir de la concurrence**.



[Exemple de graphe marqué \[Wikipedia\]](#)

OBJECTIFS DU TP

En utilisant Gemoc studio, créer un langage de modélisation des graphes marqués qui dispose des éléments suivants:

- Un **méta-modèle ecore** permettant de représenter des graphes marqués.
- Un **éditeur graphique sirius** pour éditer graphiquement des graphes marqués.
- Un **modèle d'exécution** permettant d'exécuter et déboguer des graphes marqués.
- Un **animateur** fournissant une visualisation graphique de l'exécution d'un graphe marqué.

**NB : Nous ne réaliserons pas manuellement le meta-modèle ecore et l'éditeur sirius dans ce TP.
Nous partirons sur la base d'un prototype contenant déjà ces éléments**

TRAVAIL PRATIQUE


TRAVAIL PRATIQUE

-> Plan

- ❑ Environnements et outils
- ❑ Partie 1 : Meta-modèle ecore et éditeur sirius
- ❑ Partie 2 : Modèle d'exécution du langage des graphes marqués
- ❑ Partie 3 : Animation Graphique de l'exécution des graphes marqués

TRAVAIL PRATIQUE

Environnement et outils

1. Gemoc studio version 2.2.0 ( important) :
http://gemoc.org/studio_releases/eclipse_package/virtualbox/updatesite/2016/10/07/v2.2.0.html
2. Projet de démarrage contenant le meta-modèle ecore et l'éditeur sirius : http://gemoc.org/gemoc-studio-old/publish/tutorial_markedgraph/html_single/MarkedGraph/org.gemoc.sample.markedgraph.zip
3. Code source d'un exemple de graphe marqué :
http://gemoc.org/gemoc-studio-old/publish/tutorial_markedgraph/html_single/MarkedGraph/org.gemoc.sample.markedgraph.sample.zip

TRAVAIL PRATIQUE

Environnement et outils

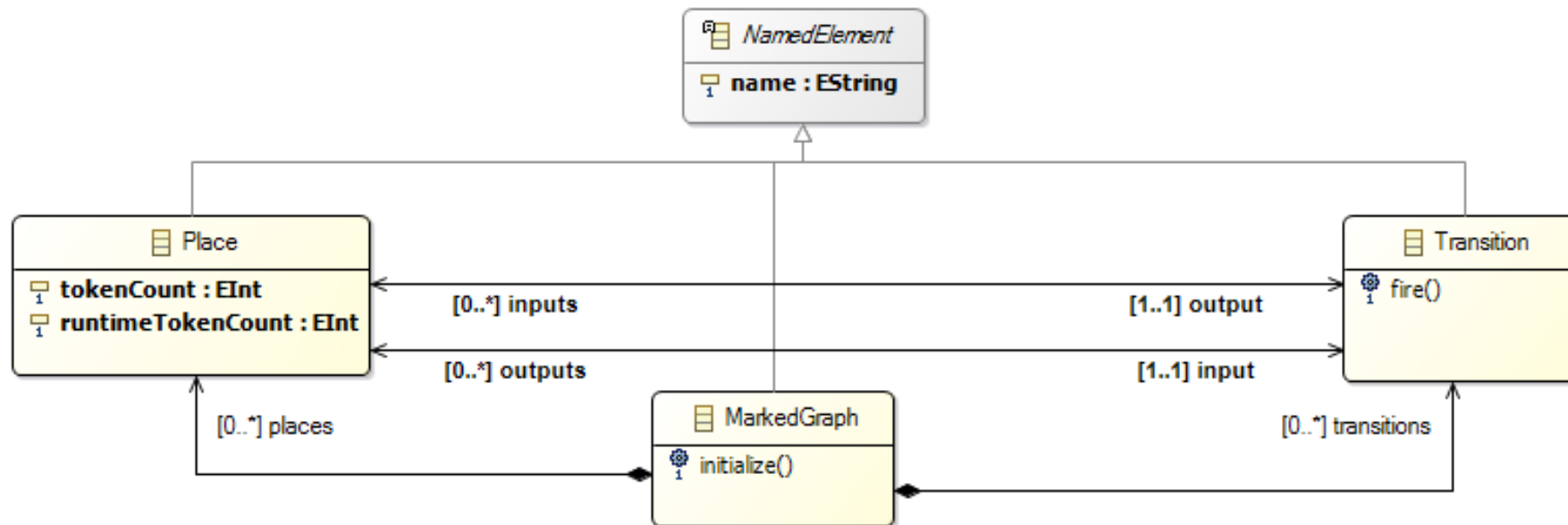
3. Tutoriel Gemoc sur la modélisation des graphes marqués :
http://gemoc.org/gemoc-studio-old/publish/tutorial_markedgraph/html_single/GuideTutorialMarkedGraph.html
4. Une discussion qui traite de problèmes survenant lors de l'application du tutoriel ci-dessus : <https://github.com/gemoc/gemoc-studio-old/issues/34>

TRAVAIL PRATIQUE

Partie 1 : Meta-modèle ecore et éditeur sirius

Le modèle de domaine, également appelé syntaxe abstraite ou méta-modèle, définit un graphe marqué comme un ensemble de lieux et de transitions.

- Chaque lieu a exactement une transition d'entrée et une transition de sortie et un nombre de jetons.
- Une transition a plusieurs lieux d'entrée et plusieurs lieux de sortie.

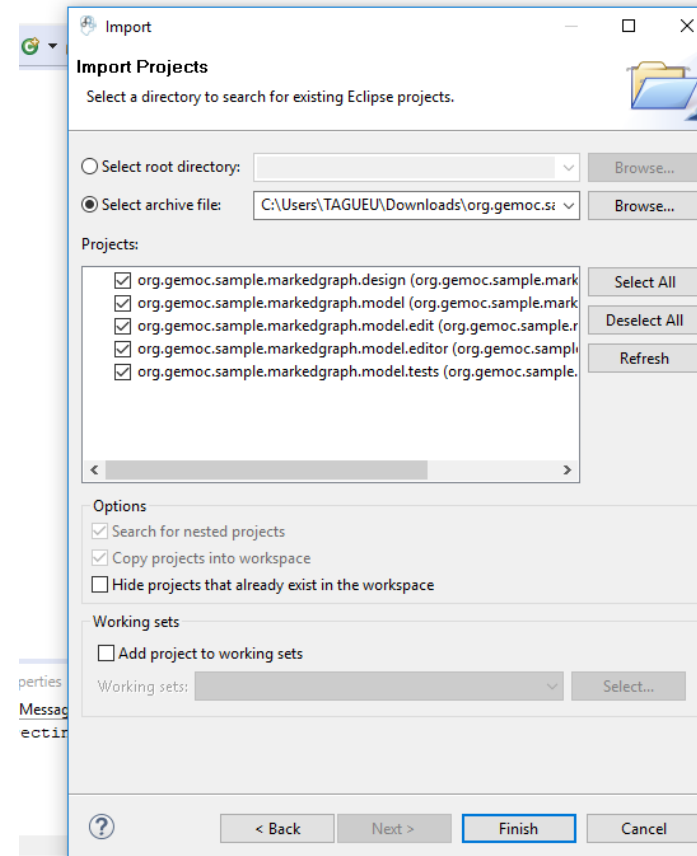
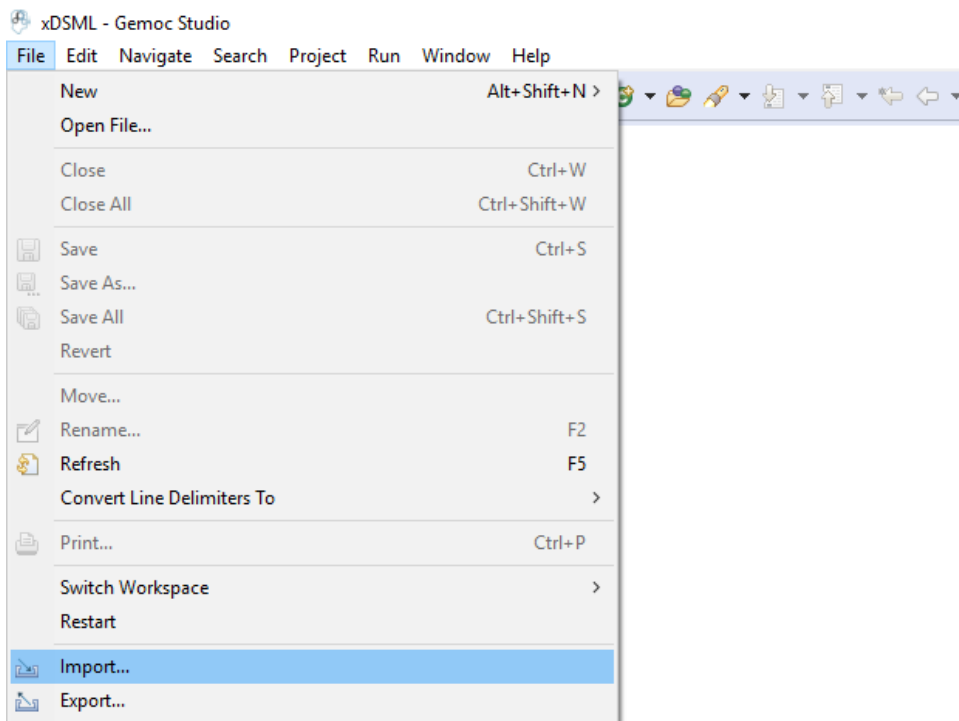


TRAVAIL PRATIQUE

Partie 1 : Meta-modèle ecore et éditeur sirius

-> Importation du méta-modèle ecore et de l'éditeur sirius

Téléchargez l'archive si cela n'est pas encore fait : http://gemoc.org/gemoc-studio-old/publish/tutorial_markedgraph/html_single/MarkedGraph/org.gemoc.sample.markedgraph.zip



TRAVAIL PRATIQUE

Partie 1 : Meta-modèle ecore et éditeur sirius

-> Importation du méta-modèle ecore et de l'éditeur sirius

Finalisation de l'importation et découverte du méta-modèle

The screenshot displays the Eclipse IDE interface. On the left, the Project Explorer shows the project structure for 'org.gemoc.sample.markedgraph.model', with the 'markedgraph' package highlighted by a red circle. The central workspace shows a UML class diagram for 'markedgraph class diagram'. The diagram features three classes: 'NamedElement' (base class), 'Place', and 'Transition'. 'Place' has attributes 'tokenCount : EInt' and 'runtimeTokenCount : EInt'. 'Transition' has an operation 'fire()'. 'MarkedGraph' has an operation 'initialize()'. Relationships include '[0..*] inputs' and '[0..*] outputs' between 'Place' and 'Transition', and '[1..1] output' and '[1..1] input' between 'Transition' and 'MarkedGraph'. 'MarkedGraph' also has '[0..*] places' and '[0..*] transitions'. The right side of the IDE shows the Palette and Quick Access toolbars.

TRAVAIL PRATIQUE

Partie 1 : Meta-modèle ecore et éditeur sirius

-> Importation du méta-modèle ecore et de l'éditeur sirius

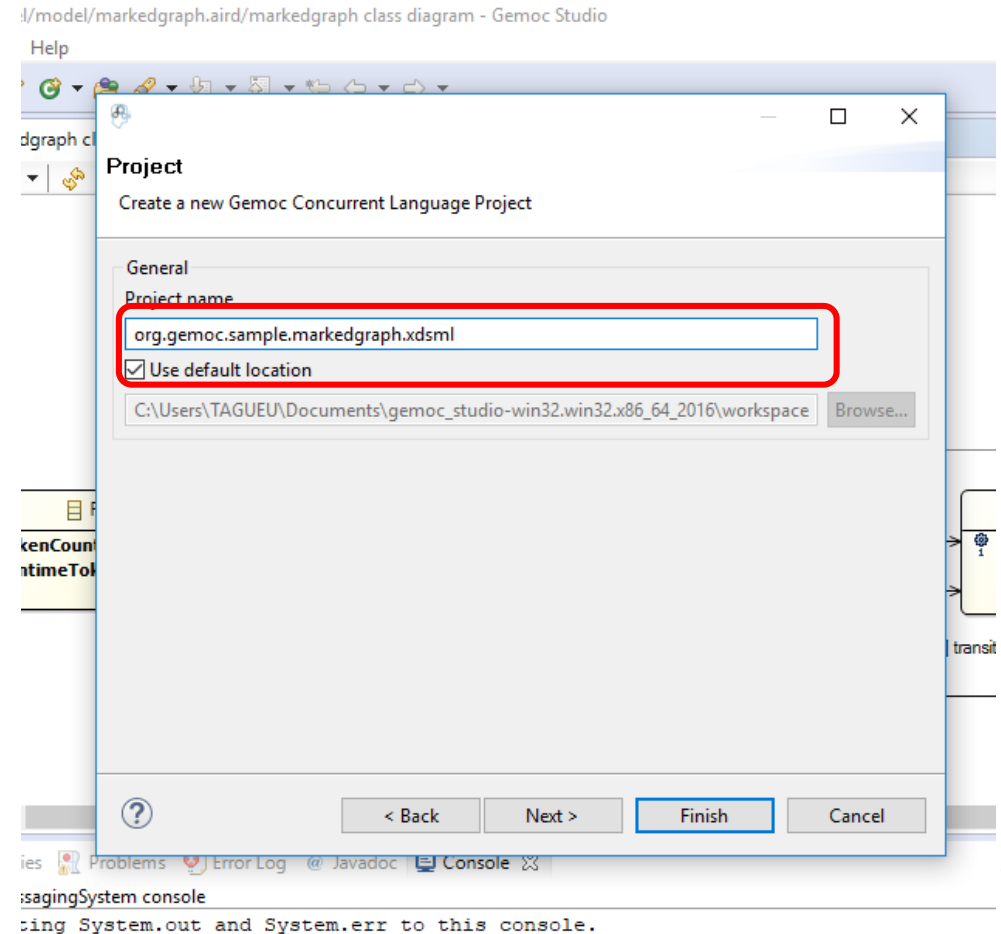
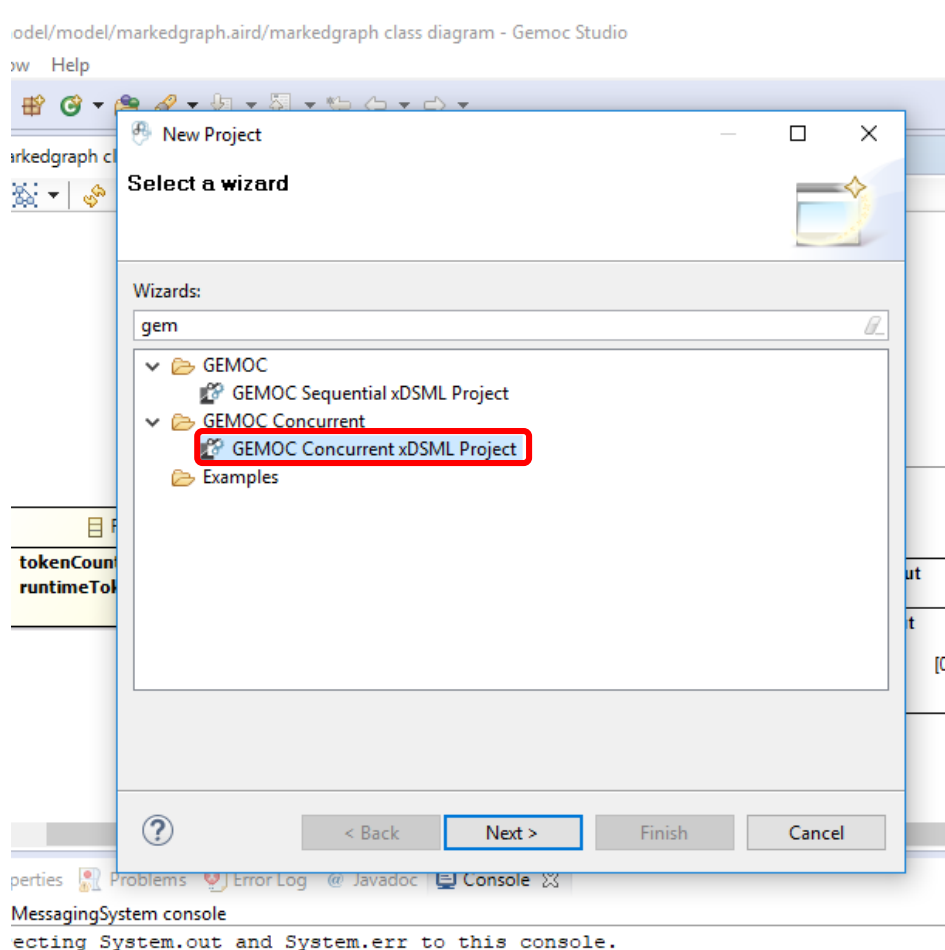
Aperçu du modèle sirius pour la syntaxe graphique

The screenshot displays the Eclipse IDE interface with the Sirius Specification Editor open. The Project Explorer on the left shows the project structure for 'org.gemoc.sample.markedgraph.design', with the 'markedgraph.odesign' file highlighted. The Sirius Specification Editor window shows the configuration for the 'markedgraph.odesign' file, including the 'markedgraph' package, the 'MarkedGraph diagram', and the 'Default' package. The 'Default' package contains several elements: 'Transition' (Square white), 'Place' (Bordered PlaceName, Ellipse white), 'placeInput' (Edge Style solid), and 'placeOutput' (Edge Style solid). The Outline view on the right shows the project structure, including 'platform:/resource/org.gemoc.sample', 'environment/viewpoint', and 'platform:/resource/org.gemoc.sample'.

TRAVAIL PRATIQUE

Partie 1 : Meta-modèle ecore et éditeur sirius

-> Création du projet principal xDSML intégrant le méta-modèle ecore

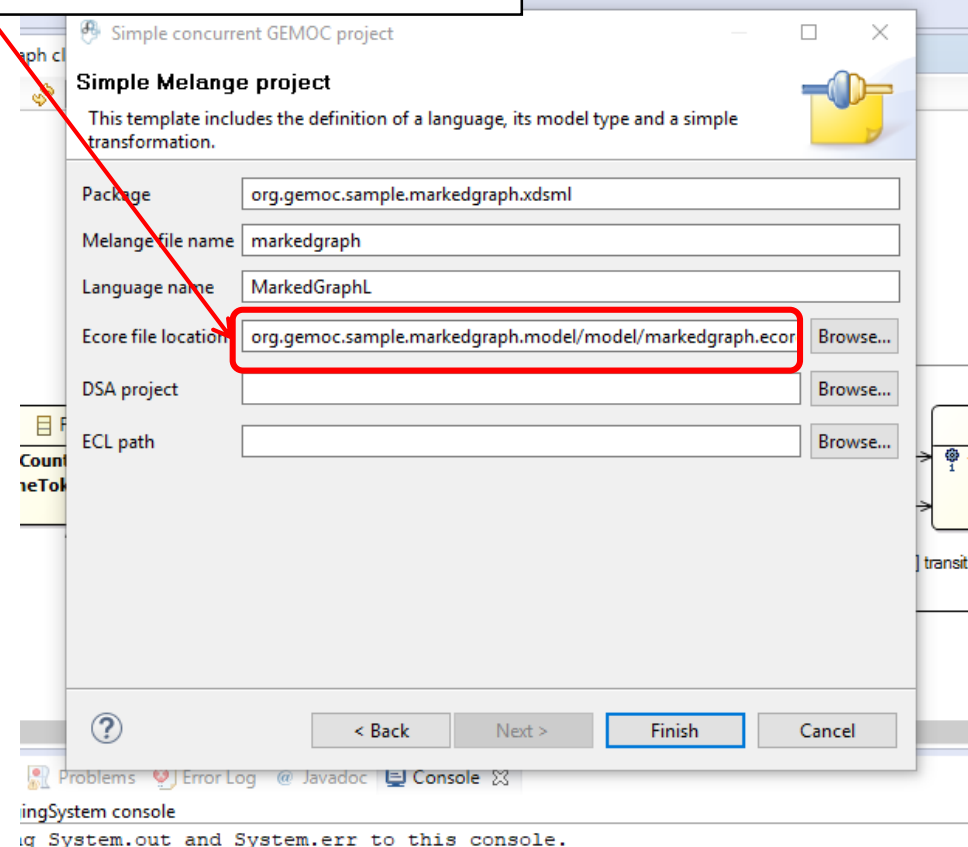
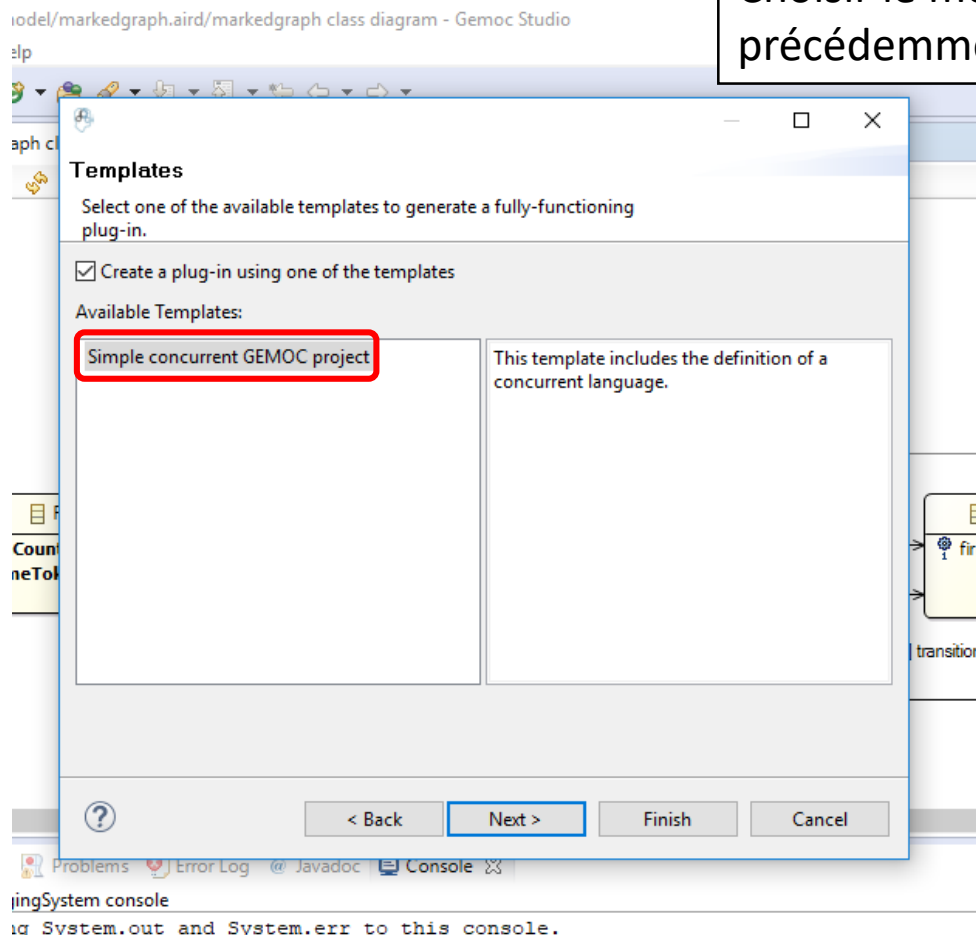


TRAVAIL PRATIQUE

Partie 1 : Meta-modèle ecore et éditeur sirius

-> Création du projet principal xDSML intégrant le méta-modèle ecore

Choisir le méta-modèle ecore importé précédemment : le fichier « **markedgraph.ecore** »



TRAVAIL PRATIQUE

Partie 1 : Meta - modèle ecore et éditeur sirius

-> Création du projet principal xDSML intégrant le méta-modèle ecore

Aperçu du fichier « melange »

The screenshot shows an IDE window with the following content:

```
package org.gemoc.sample.markedgraph.xdsmllanguage MarkedGraphL {syntax "platform:/resource/org.gemoc.sample.markedgraph.model/model/markedgraph.ecore"}with /*with qualified.class.name*/ecl "eclFilePath"exactType MarkedGraphLMT}
```

Annotations in the image:

- A red box highlights the `with /*` line, with a callout box stating: "Mettre le mot-clé « **with** » en commentaire pour corriger l'erreur générée".
- A red arrow points from the `platform:/resource/org.gemoc.sample.markedgraph.model/model/markedgraph.ecore" string to a callout box stating: "Référence vers le méta-modèle ecore".`

The IDE interface includes a Project Explorer on the left, a Model Explorer at the top, and a Properties/Problems/Error Log/Console area at the bottom.

Partie 2 : Modèle d'exécution du langage des graphes marqués

TRAVAIL PRATIQUE

Partie 2 : Modèle d'exécution du langage

-> Sémantique d'exécution

La sémantique d'exécution que nous voulons définir est la suivante :

- Une transition peut être déclenchée s'il y a au moins un jeton dans chaque lieu d'entrée.
- Lorsqu'une transition est déclenchée, un jeton est retiré de chacun de ses lieux d'entrée et un jeton est ajouté à chacun de ses lieux de sortie.
- Plusieurs transitions peuvent être lancées en même temps.

TRAVAIL PRATIQUE

Partie 2 : Modèle d'exécution du langage

-> Sémantique d'exécution

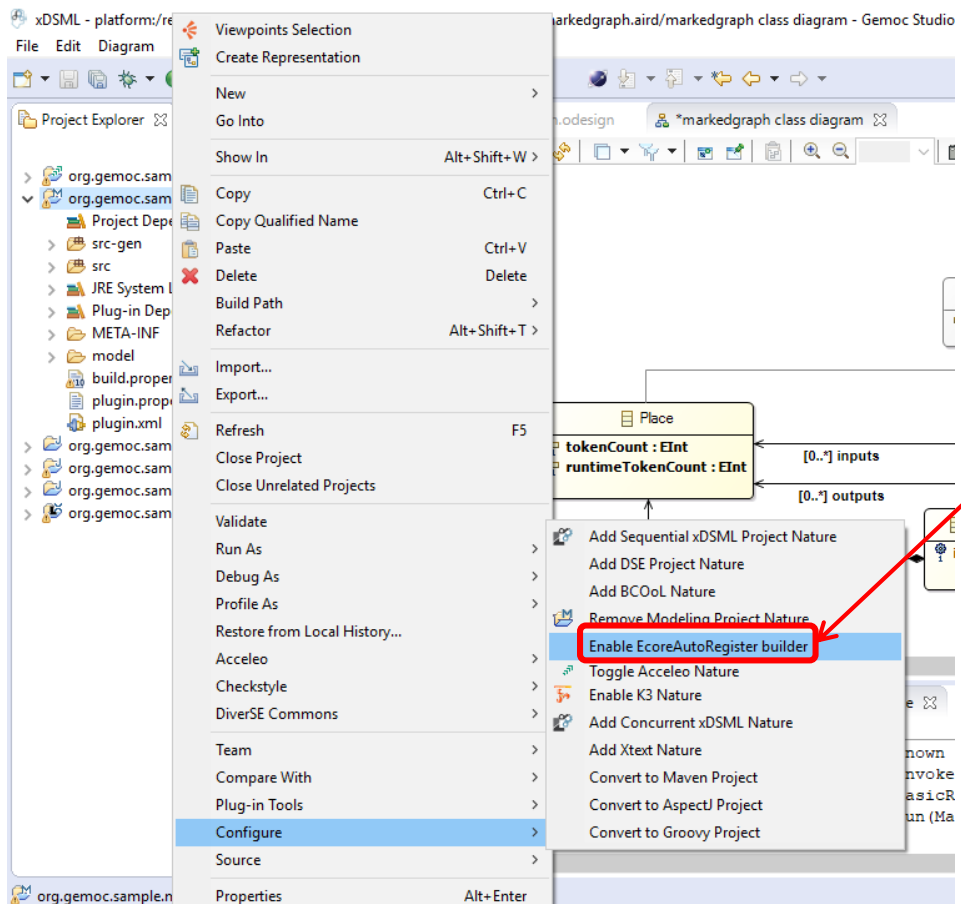
Pour définir la sémantique précédente sur GEMOC, Il faut :

- Définir les **données d'exécution (ED)**, comme le nombre de jetons dans un lieu, et des **fonctions d'exécution (EF)**, comme le déclenchement d'une transition. Les ED et EF constituent le DSA.
- Définir le **modèle de concurrence** comme un ensemble d'événements et de contraintes sur ces événements. C'est la préoccupation du MoCC qui est définie dans un projet DSE (**utilisant ECL, Event Constraint Language**) éventuellement complété par des projets MoCCML pour définir des bibliothèques de contraintes.
- Mettre en correspondance le DSA et le MoCC.

TRAVAIL PRATIQUE

Partie 2 : Modèle d'exécution du langage

-> Préliminaire : Activation de l'enregistrement automatique du meta-modèle ecore

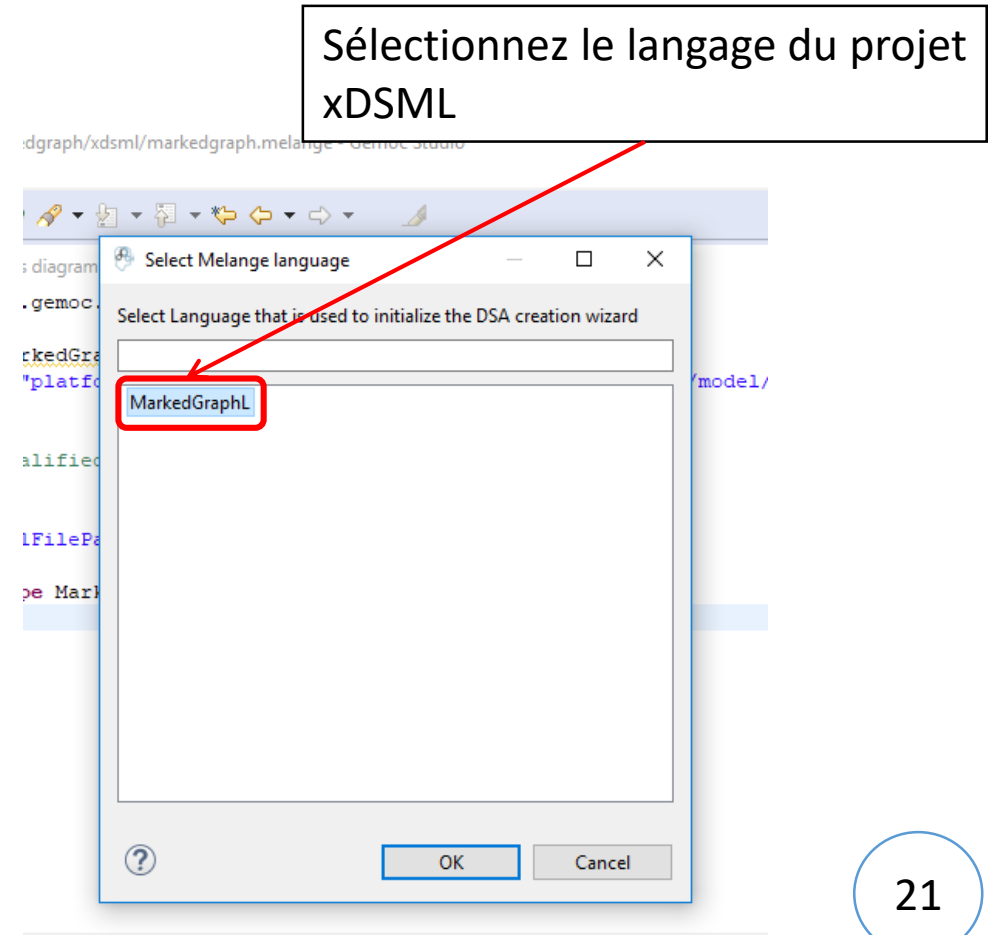
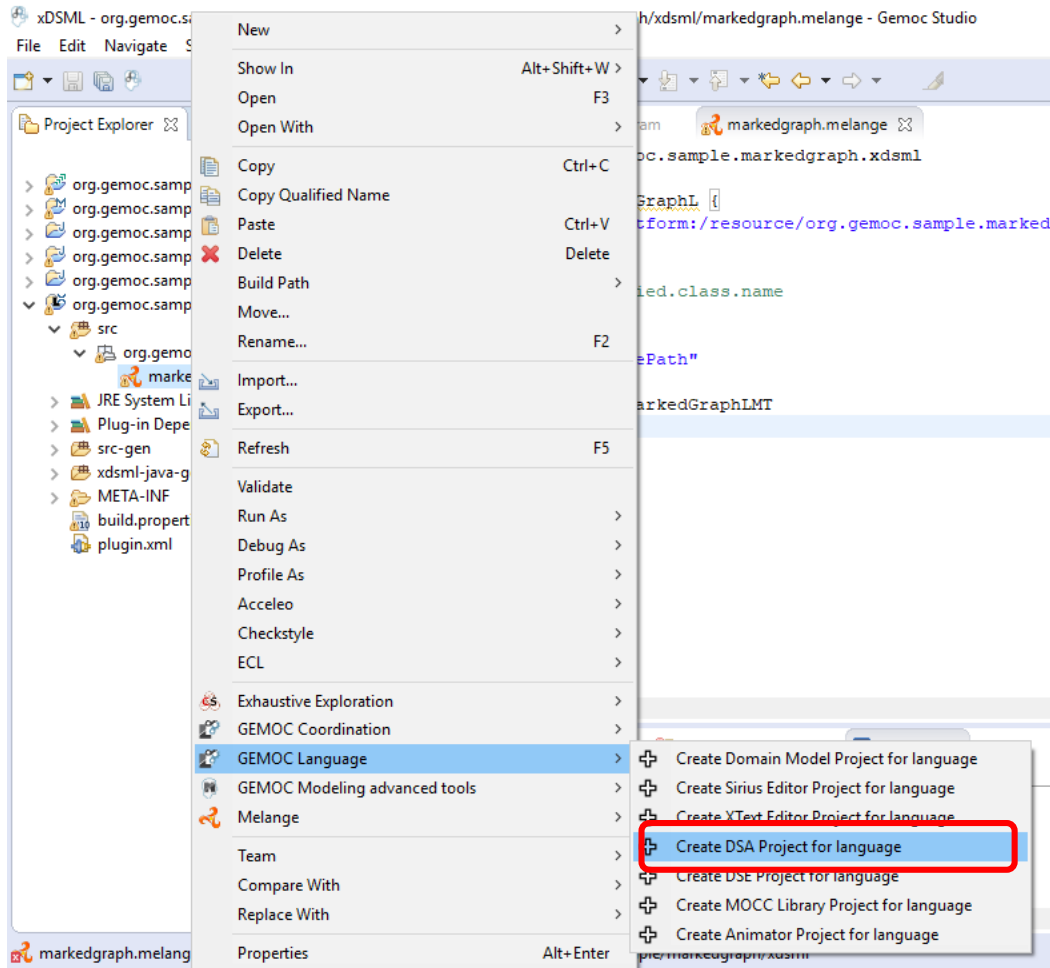


Ceci permet de bien enregistrer le plugin associé au méta-modèle ecore et d'éviter des erreurs de génération de codes. Voir discussion <https://github.com/gemoc/gemoc-studio-old/issues/34>

TRAVAIL PRATIQUE

Partie 2 : Modèle d'exécution du langage

-> Création du projet DSA

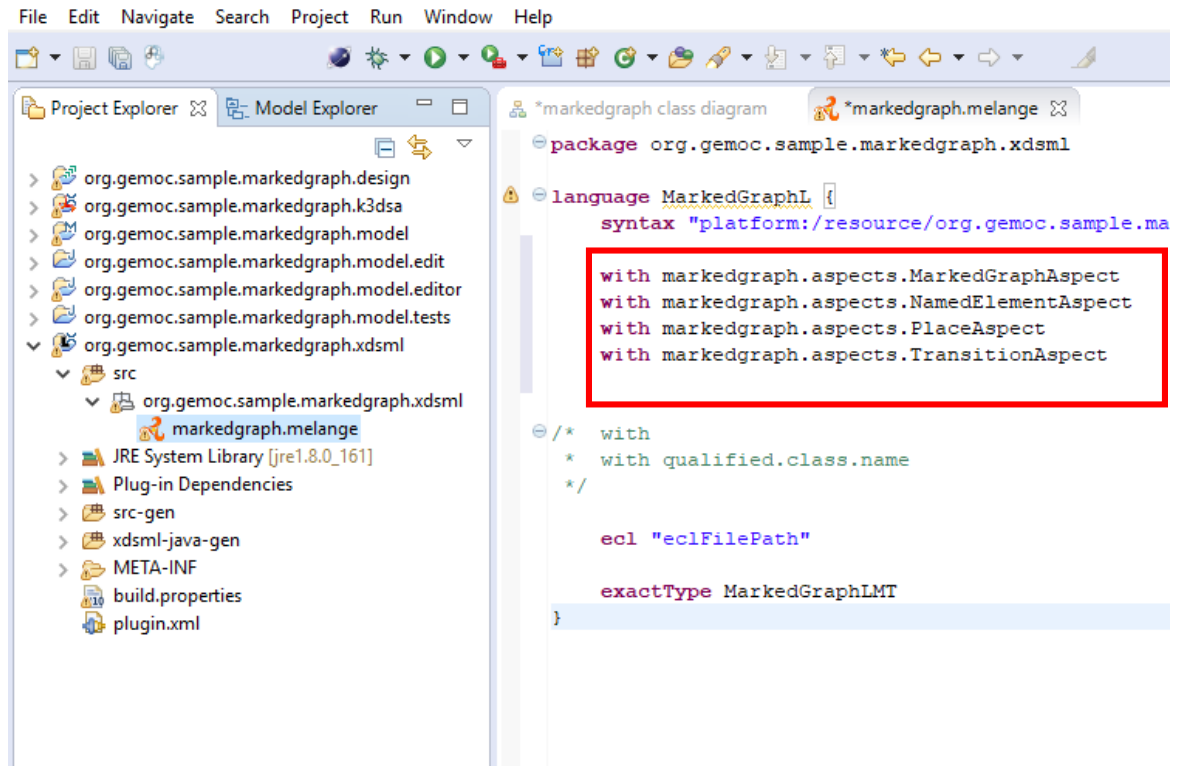
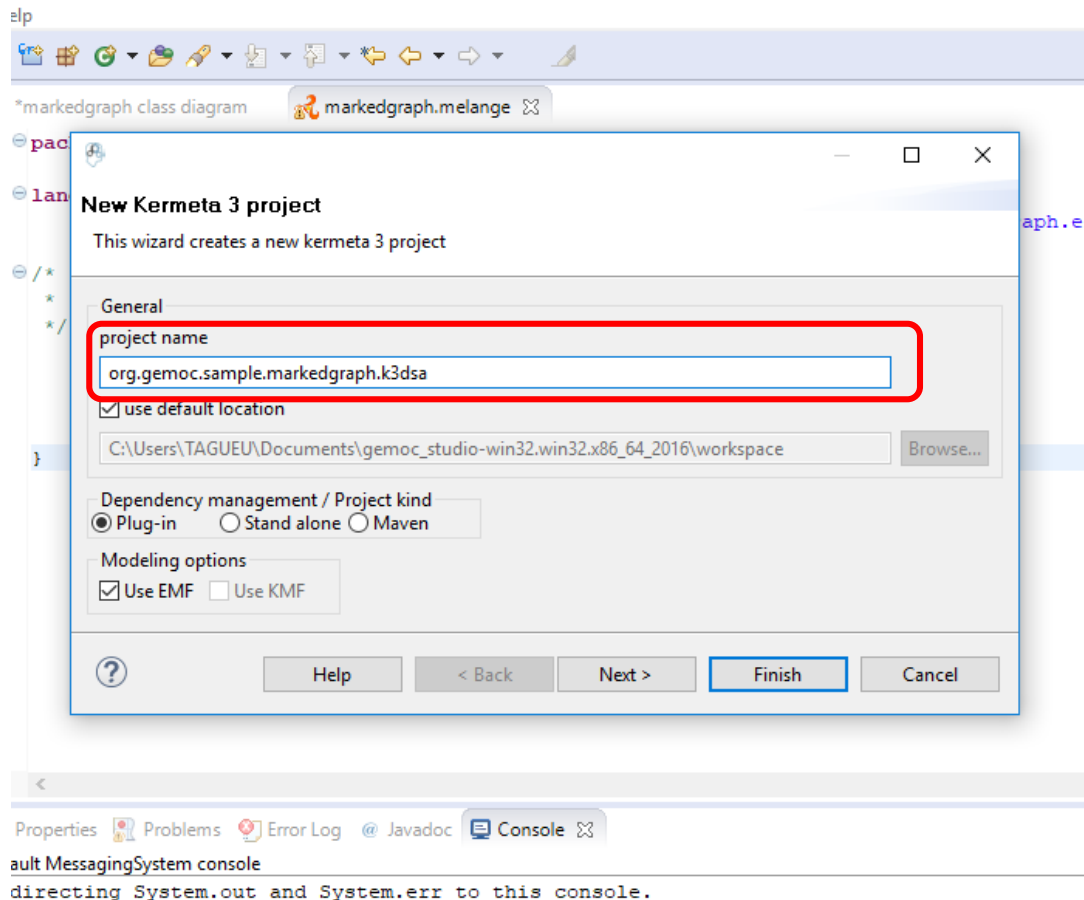


TRAVAIL PRATIQUE

Partie 2 : Modèle d'exécution du langage

-> Création du projet DSA

Fichier « markedgraph.melange » modifié automatiquement pour intégrer les aspects du DSA



TRAVAIL PRATIQUE

Partie 2 : Modèle d'exécution du langage

-> Création du projet DSA

Aperçu du projet DSA

The screenshot displays an IDE interface with the following components:

- Project Explorer:** Shows the project structure. A red box highlights the path: `org.gemoc.sample.markedgraph.design` > `org.gemoc.sample.markedgraph.k3dsa` > `src` > `markedgraph.aspects` > `markedgraphAspects.xtend`.
- Model Explorer:** Shows the loaded models: `*markedgraph class diagram`, `*markedgraph.melange`, and `markedgraphAspects.xtend`.
- Source Editor:** Displays the content of `markedgraphAspects.xtend`, which is also highlighted with a red box. The code is as follows:

```
package markedgraph.aspects

import fr.inria.diverse.k3.al.annotationprocessor.Aspect

@Aspect(className=MarkedGraph)
class MarkedGraphAspect extends NamedElementAspect {

}

@Aspect(className=NamedElement)
abstract class NamedElementAspect {

}

@Aspect(className=Place)
class PlaceAspect extends NamedElementAspect {

}

@Aspect(className=Transition)
class TransitionAspect extends NamedElementAspect {

}
```
- Right Panel:** Shows a tree view of the loaded models, including `markedgraph`, `import c`, `MarkedG`, `PlaceAs`, and `Transiti`.
- Bottom Panel:** Shows the `Properties`, `Problems`, `Error Log`, `Javadoc`, and `Console` tabs. The console shows `Default MessagingSystem console`.

TRAVAIL PRATIQUE

Partie 2 : Modèle d'exécution du langage

-> Projet DSA : Sémantique d'exécution

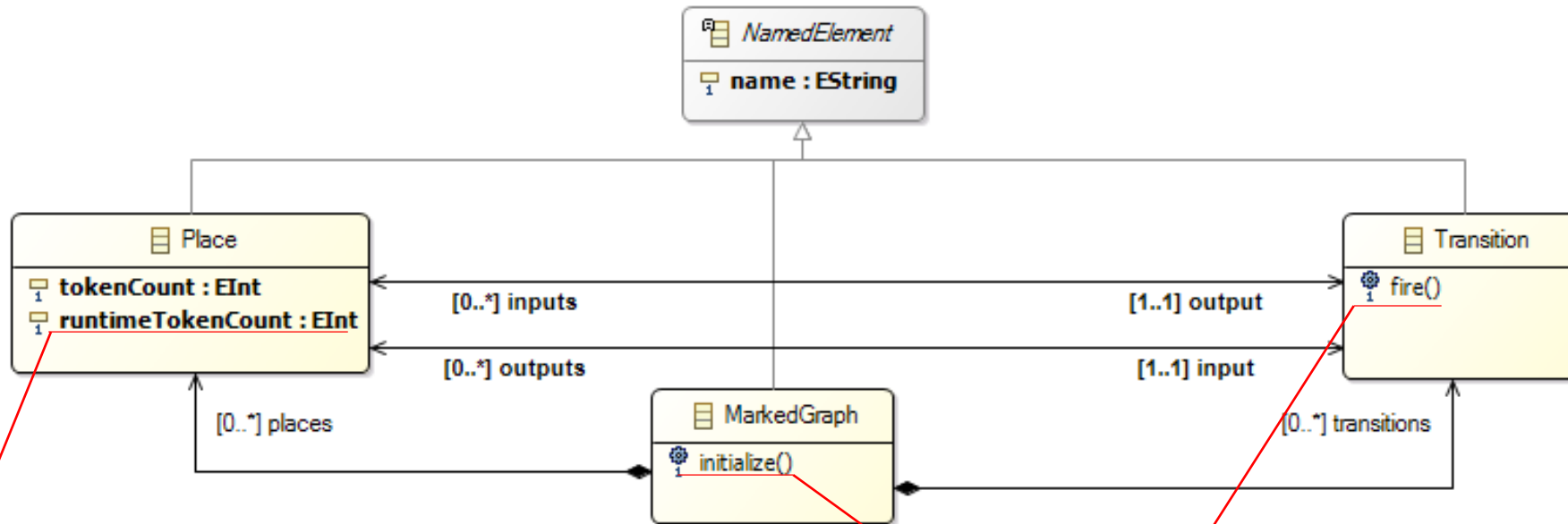
Le DSA d'un graphe marqué est composé de :

- ❑ Une ED appelé **runtimeTokenCount** défini sur « Place ». Il représente le nombre de jetons dans un lieu lorsque le modèle est exécuté.
- ❑ Une EF appelé **initialize()** défini sur « MarkedGraph ». Il initialise le nombre de jetons d'exécution de chaque lieu avec le nombre de jetons initial.
- ❑ Une EF appelé **fire()** défini sur « Transition ». Il retire un jeton de chacun de ses lieux d'entrée et ajoute un jeton à tous ses lieux de sortie.

TRAVAIL PRATIQUE

Partie 2 : Modèle d'exécution du langage

-> Projet DSA : Sémantique d'exécution



Donnée d'exécution

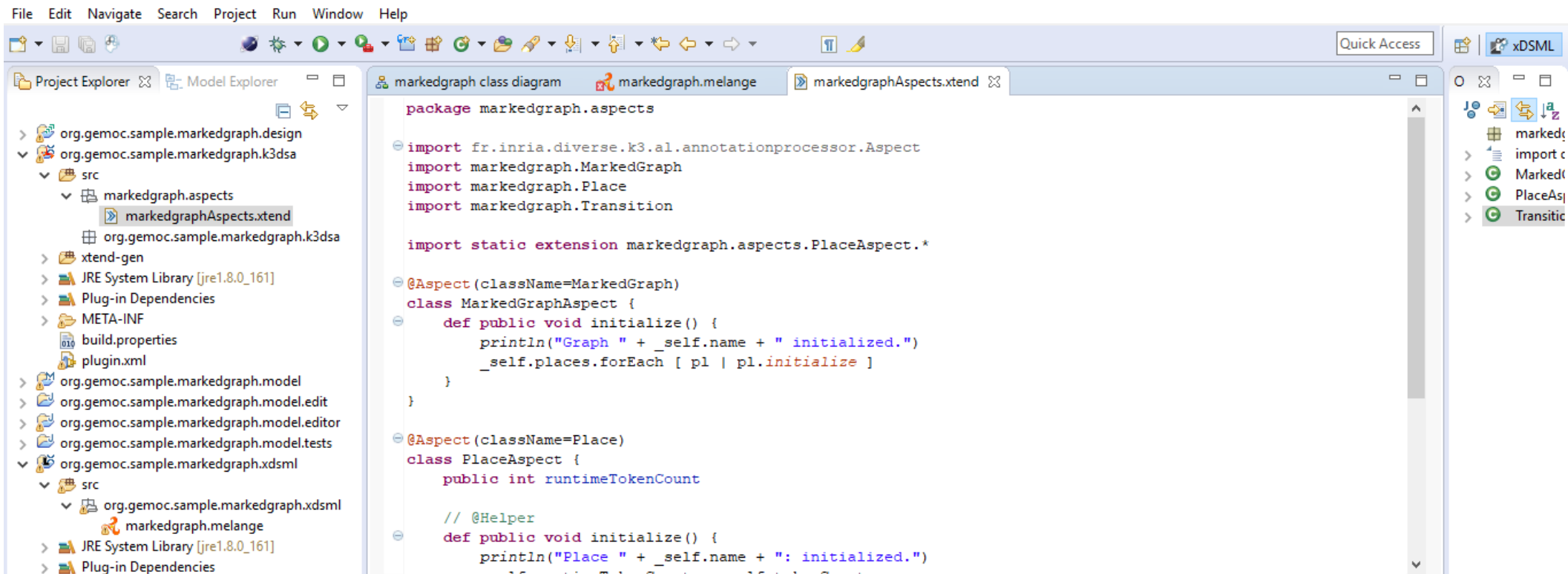
Fonctions d'exécution

TRAVAIL PRATIQUE

Partie 2 : Modèle d'exécution du langage

-> **Projet DSA: intégration de la sémantique d'exécution**

Modifiez manuellement le fichier « markedgraphAspects.xtend ». Remplacez son contenu par celui précisé dans le tutoriel (section 4.1.3) : [http://gemoc.org/gemoc-studio-old/publish/tutorial markedgraph/html single/GuideTutorialMarkedGraph.html](http://gemoc.org/gemoc-studio-old/publish/tutorial%20markedgraph/html%20single/GuideTutorialMarkedGraph.html)



The screenshot shows the Gemoc Studio IDE interface. The Project Explorer on the left shows the project structure, with the file 'markedgraphAspects.xtend' selected under 'org.gemoc.sample.markedgraph.k3dsa'. The main editor displays the following code:

```
package markedgraph.aspects

import fr.inria.diverse.k3.al.annotationprocessor.Aspect
import markedgraph.MarkedException
import markedgraph.Place
import markedgraph.Transition

import static extension markedgraph.aspects.PlaceAspect.*

@Aspect(className=MarkedException)
class MarkedExceptionAspect {
    def public void initialize() {
        println("Graph " + _self.name + " initialized.")
        _self.places.forEach [ pl | pl.initialize ]
    }
}

@Aspect(className=Place)
class PlaceAspect {
    public int runtimeTokenCount

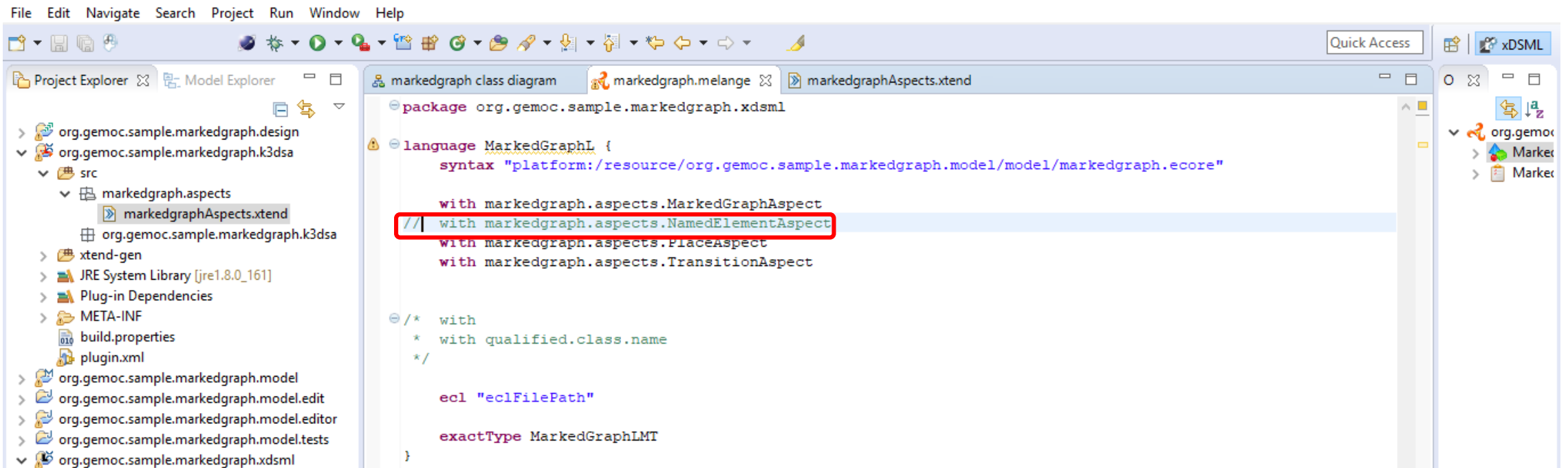
    // @Helper
    def public void initialize() {
        println("Place " + _self.name + ": initialized.")
    }
}
```

TRAVAIL PRATIQUE

Partie 2 : Modèle d'exécution du langage

-> **Projet DSA: intégration de la sémantique d'exécution**

Correction de l'erreur générée dans le fichier de « melange » après avoir modifié le fichier « markedgraphAspects.xtend ».



The screenshot shows an IDE window with the following components:

- Project Explorer:** Shows the project structure for 'org.gemoc.sample.markedgraph.k3dsa', including 'src', 'markedgraph.aspects', and 'markedgraphAspects.xtend'.
- Model Explorer:** Shows the 'markedgraph class diagram'.
- Editor:** Displays the content of 'markedgraphAspects.xtend'. The code is as follows:

```
package org.gemoc.sample.markedgraph.xdsl

language MarkedGraphL {
    syntax "platform:/resource/org.gemoc.sample.markedgraph.model/model/markedgraph.ecore"

    with markedgraph.aspects.MarkedGraphAspect
    // with markedgraph.aspects.NamedElementAspect
    with markedgraph.aspects.PlaceAspect
    with markedgraph.aspects.TransitionAspect

    /* with
     * with qualified.class.name
     */

    ecl "eclFilePath"

    exactType MarkedGraphLMT
}
```

The line `// with markedgraph.aspects.NamedElementAspect` is highlighted with a red box, indicating it was the subject of the correction.
- Right Panel:** Shows the 'xDSML' view with a tree structure of the model elements.

TRAVAIL PRATIQUE

Partie 2 : Modèle d'exécution du langage

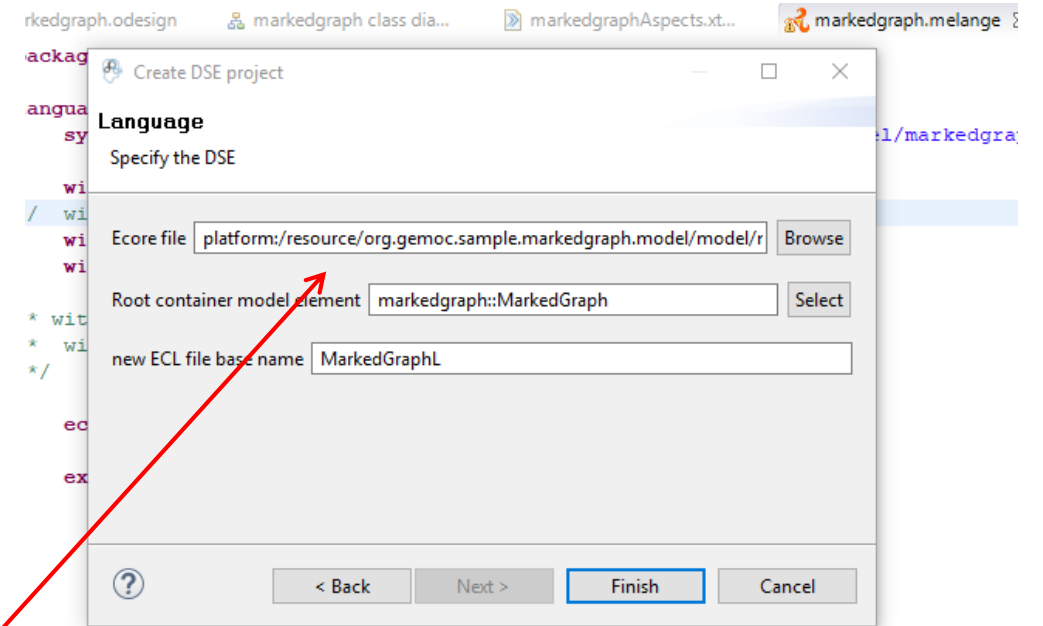
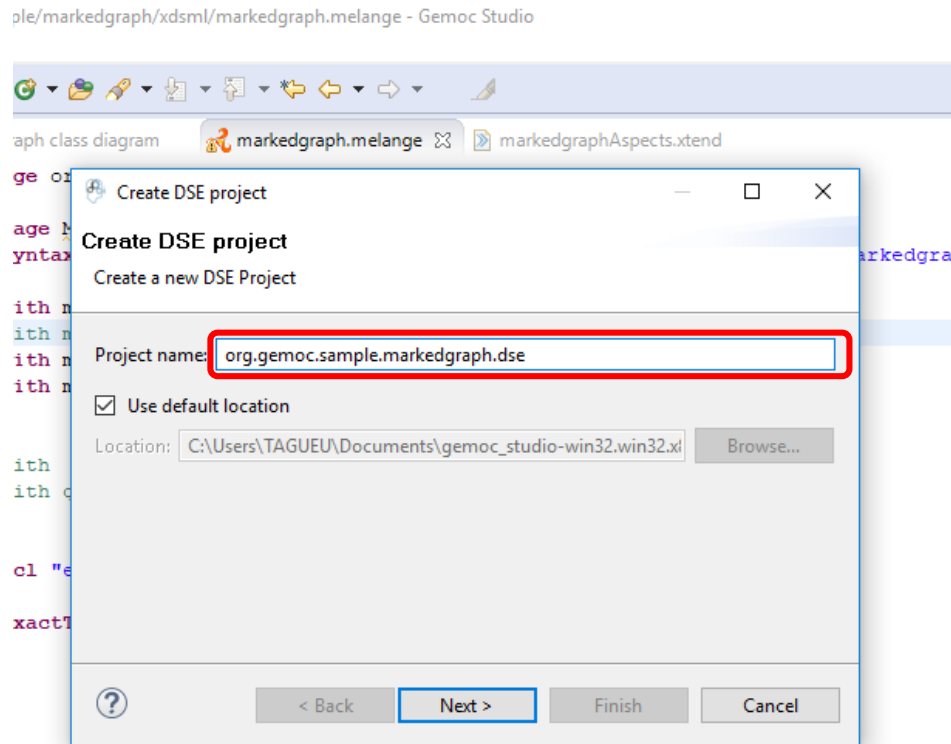
-> Création du projet DSE

The image shows two screenshots from the Gemoc Studio IDE. The left screenshot displays the 'File' menu with the 'GEMOC Language' option selected, which has opened a sub-menu. In this sub-menu, the option 'Create DSE Project for language' is highlighted with a red rectangle. The right screenshot shows a dialog box titled 'Select Melange language' with the text 'Select Language that is used to initialize the DSE creation wizard'. The dialog box contains a list with 'MarkedGraphL' selected. A blue arrow points from the 'Create DSE Project for language' option in the left screenshot to the dialog box in the right screenshot.

TRAVAIL PRATIQUE

Partie 2 : Modèle d'exécution du langage

-> Création du projet DSE



Corrigez le chemin vers le méta-modèle ecore au cas où il est mal généré

TRAVAIL PRATIQUE

Partie 2 : Modèle d'exécution du langage

-> Création du projet DSE

Aperçu du projet généré

The screenshot displays an IDE interface with the following components:

- Project Explorer:** Shows a project structure under the package `org.gemoc.sample.markedgraph.design`. A red box highlights the sub-project `org.gemoc.sample.markedgraph.dse`, which contains an ECL file named `MarkedGraphL.ecl`, along with folders for `META-INF`, `mtl-gen`, and `qvto-gen`, and files for `build.properties` and `moc2as.properties`.
- Model Explorer:** Shows the package `markedgraph`.
- Editor:** Displays the content of the `MarkedGraphL.ecl` file, which is enclosed in a red box. The code is as follows:

```
import "platform:/resource/org.gemoc.sample.markedgraph.model/model/markedgraph.ecore"  
package markedgraph  
endpackage
```
- Right Panel:** Shows the package `platform:/` with sub-packages `platform` and `marker`.

TRAVAIL PRATIQUE

Partie 2 : Modèle d'exécution du langage

-> projet DSE : sémantique d'exécution

Pour notre langage de graphes marqués, on définit :

❑ Deux DSE **fireIt** et **initIt**.

❑ **fireIt** est défini dans le contexte d'une « Transition ». Il est lié à EF « fire » de « Transition »

❑ **initIt** est défini dans le contexte d'un « MarkedGraph ». Il est lié à l'EF « initialize » de MarkedGraph

TRAVAIL PRATIQUE

Partie 2 : Modèle d'exécution du langage

-> projet DSE : sémantique d'exécution

Pour notre langage de graphes marqués, nous définissons les contraintes suivantes sur les évènements:

1. Contrainte sur les événements « firelt ».
 - a) S'il n'y a pas de jetons, alors le « firelt » de la transition de sortie ne peut se produire qu'après que l'évènement « firelt » de la transition d'entrée ait eu lieu.
 - a) S'il y a des jetons dans un lieu, alors l'évènement « firelt » de la transition de sortie peut se produire autant de fois qu'il y a de jetons dans ce lieu. Ensuite, il ne se produira que lorsque le « firelt » de la transition d'entrée du lieu s'est produit.

TRAVAIL PRATIQUE

Partie 2 : Modèle d'exécution du langage

-> projet DSE : sémantique d'exécution

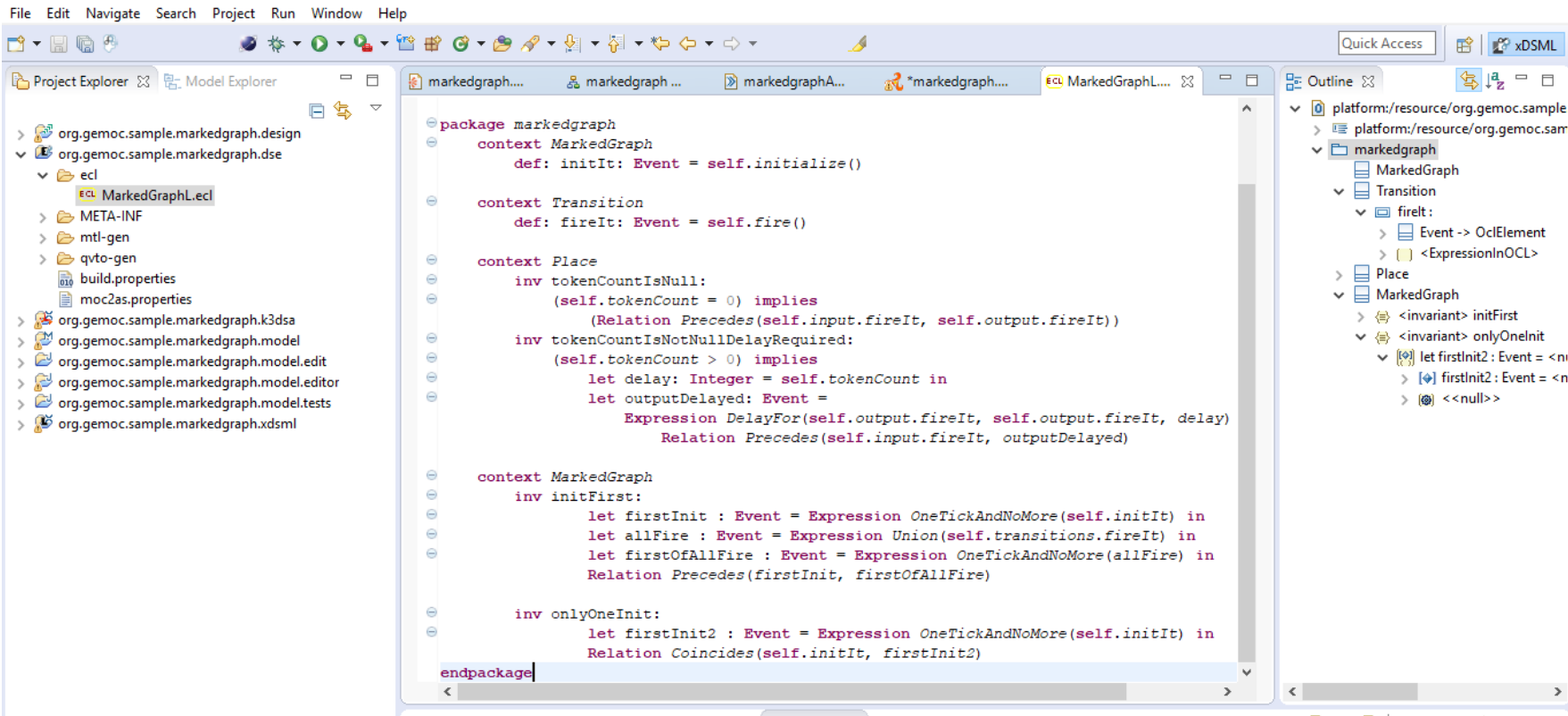
2. Une contrainte dans le contexte de l'élément « MarkedGraph » exprimant que l'événement « initIt » doit se produire avant tout événement « fireIt ».
3. Une autre contrainte dans le contexte de l'élément « MarkedGraph » exprimant que l'événement « initIt » ne peut se produire qu'une seule fois.

TRAVAIL PRATIQUE

Partie 2 : Modèle d'exécution du langage

-> projet DSE : intégration de la sémantique d'exécution

Modifiez manuellement le fichier « MarkedGraphL.ecl ». Remplacez son contenu par celui précisé dans le tutoriel (section 4.2.2) : [http://gemoc.org/gemoc-studio-old/publish/tutorial markedgraph/html single/GuideTutorialMarkedGraph.html](http://gemoc.org/gemoc-studio-old/publish/tutorial%20markedgraph/html%20single/GuideTutorialMarkedGraph.html)



The screenshot shows the Eclipse IDE with the file 'MarkedGraphL.ecl' open. The code defines a package 'markedgraph' with several contexts and invariants. The 'MarkedGraph' context includes an 'initIt' event and an 'initFirst' invariant. The 'Transition' context includes a 'fireIt' event. The 'Place' context includes a 'tokenCountIsNull' invariant, an 'implies' relation between 'Precedes' and 'fireIt', and a 'tokenCountIsNotNullDelayRequired' invariant that defines a 'delay' and an 'outputDelayed' event. The 'MarkedGraph' context also includes an 'onlyOneInit' invariant that defines 'firstInit2' and a 'Coincides' relation.

```
package markedgraph
context MarkedGraph
  def: initIt: Event = self.initialize()

context Transition
  def: fireIt: Event = self.fire()

context Place
  inv tokenCountIsNull:
    (self.tokenCount = 0) implies
      (Relation Precedes(self.input.fireIt, self.output.fireIt))
  inv tokenCountIsNotNullDelayRequired:
    (self.tokenCount > 0) implies
      let delay: Integer = self.tokenCount in
      let outputDelayed: Event =
        Expression DelayFor(self.output.fireIt, self.output.fireIt, delay)
      Relation Precedes(self.input.fireIt, outputDelayed)

context MarkedGraph
  inv initFirst:
    let firstInit : Event = Expression OneTickAndNoMore(self.initIt) in
    let allFire : Event = Expression Union(self.transitions.fireIt) in
    let firstOfAllFire : Event = Expression OneTickAndNoMore(allFire) in
    Relation Precedes(firstInit, firstOfAllFire)

  inv onlyOneInit:
    let firstInit2 : Event = Expression OneTickAndNoMore(self.initIt) in
    Relation Coincides(self.initIt, firstInit2)
endpackage
```

TRAVAIL PRATIQUE

Partie 2 : Modèle d'exécution du langage

-> Exécution : configuration de l'atelier de modélisation

The screenshot shows the Eclipse IDE interface with the 'Run Configurations' dialog box open. The dialog is titled 'Run Configurations' and has a subtitle 'Create, manage, and run configurations'. The main area is divided into several sections:

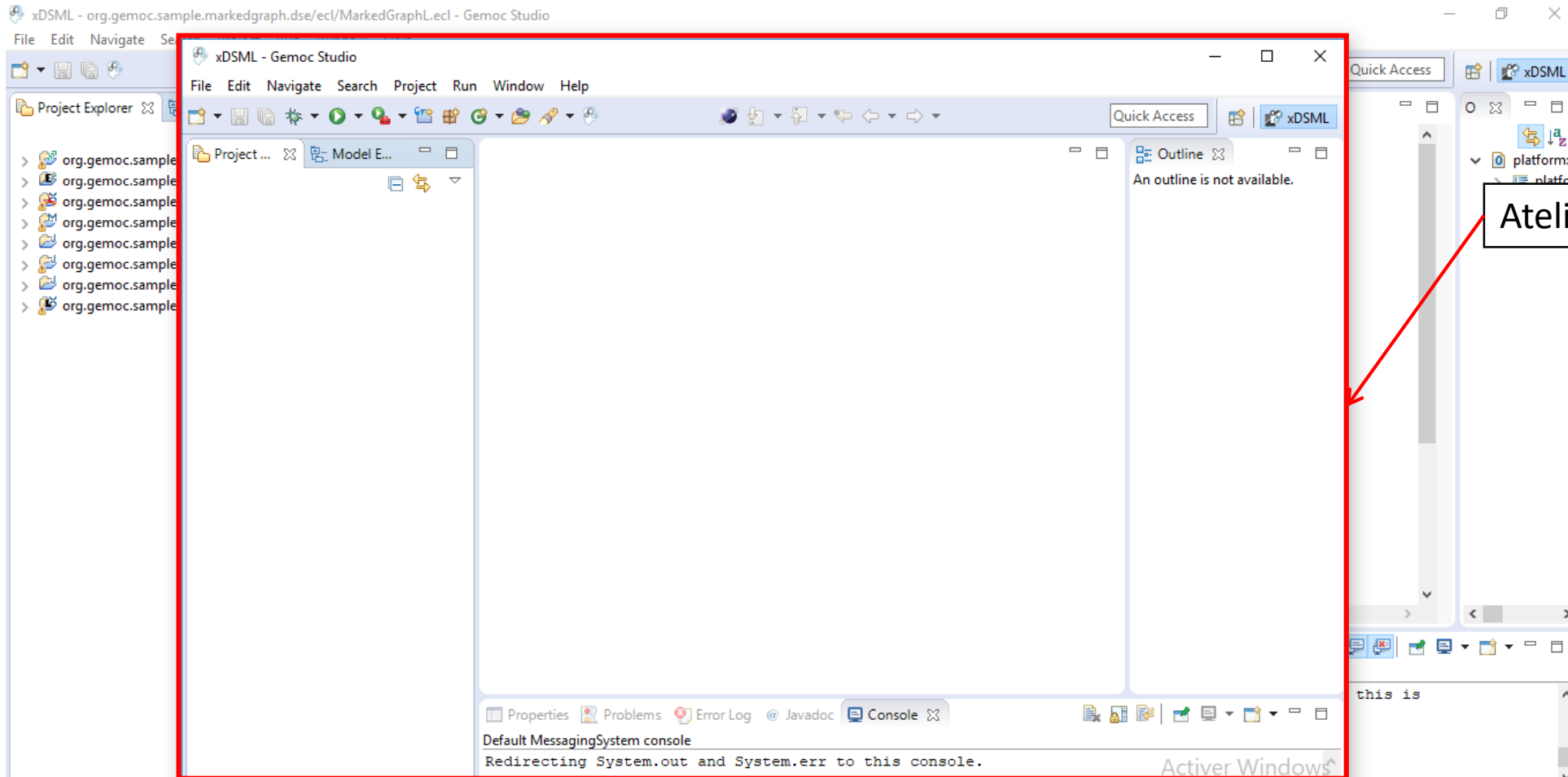
- Name:** MarkedGraph Modeling Workbench
- Workspace Data:** Location is set to `${workspace_loc}/../runtime-MarkedGraphModelingWorkbench`. There are buttons for 'Workspace...', 'File System...', and 'Variables...'. A checkbox 'Ask for confirmation before clearing' is checked.
- Program to Run:** 'Run a product' is selected with the value `org.gemoc.gemoc_studio.branding.gemoc_studio`. 'Run an application' is also visible with the value `org.eclipse.ui.ide.workbench`.
- Java Runtime Environment:** 'Java executable' is set to 'default'. 'Execution environment' is set to 'JavaSE-1.8 (jre1.8.0_161)'. There are buttons for 'Environments...' and 'Installed JREs...'. 'Runtime JRE' is also visible with the value 'jre1.8.0_161'.
- Bootstrap entries:** An empty text field.

At the bottom of the dialog, there are buttons for 'Revert', 'Apply', 'Run', and 'Close'. The 'Run' button is highlighted with a red box. In the background, the Eclipse IDE is visible, showing the 'Project Explorer' on the left with a tree view of project files, and the 'Platform' view on the right showing a tree view of platform files. The 'Run Configurations' dialog is also highlighted with a red border.

TRAVAIL PRATIQUE

Partie 2 : Modèle d'exécution du langage

-> Exécution : configuration de l'atelier de modélisation



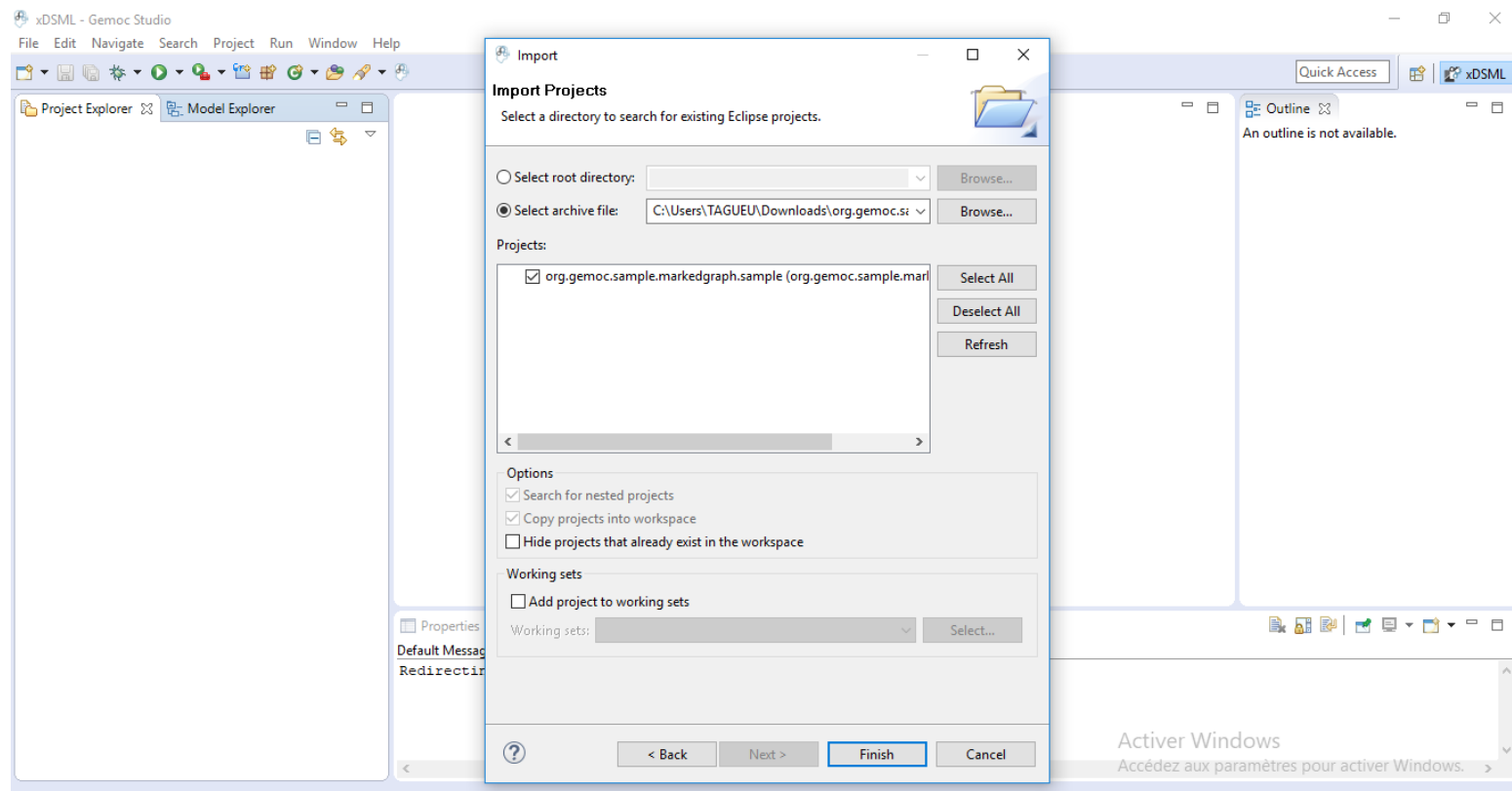
Atelier de modélisation

TRAVAIL PRATIQUE

Partie 2 : Modèle d'exécution du langage

-> Exécution : importation d'un exemple de graphe marqué

Téléchargez l'archive si cela n'est pas encore fait : [http://gemoc.org/gemoc-studio-old/publish/tutorial markedgraph/html single/MarkedGraph/org.gemoc.sample.markedgraph.sample.zip](http://gemoc.org/gemoc-studio-old/publish/tutorial%20markedgraph/html%20single/MarkedGraph/org.gemoc.sample.markedgraph.sample.zip)



TRAVAIL PRATIQUE

Partie 2 : Modèle d'exécution du langage

-> Exécution : présentation de l'exemple de graphe importé

The screenshot displays the Gemoc Studio interface. The main workspace shows a Petri net diagram with places p1, p2, p3 and transitions t1, t2, t3. A red box highlights the diagram and the Properties window. The Properties window shows the following data for Place p1:

Property	Value
Input	Transition t4
Name	p1
Output	Transition t1
Runtime Token Count	0
Token Count	1

Annotations with red arrows point to the diagram and the Properties window:

- Editeur graphique sirius**: Points to the Petri net diagram.
- Présentation des propriétés d'un élément du graphe**: Points to the Properties window.

TRAVAIL PRATIQUE

Partie 2 : Modèle d'exécution du langage

-> Exécution : ajout d'une configuration de débogage à l'exemple

Précisez le modèle « wikipedia.markedgraph »

Précisez le langage xDSML des graphes marqués « org.gemoc.sample.markedgraph.xdxml.MarkedGraphL »

TRAVAIL PRATIQUE

Partie 2 : Modèle d'exécution du langage

-> Exécution : débogage de l'exemple importé

The screenshot shows the Eclipse IDE interface with the following components:

- Project Explorer:** Shows the project structure for 'org.gemoc.sample.markedgraph', including 'gemoc-gen', 'wikipedia.aird', and 'Representations per category'.
- Diagram:** A Petri net diagram with places p1, p2, p3 and transition t1. A box labeled 'Vue d'exécution' points to the diagram.
- Execution View:** A window on the right showing a list of events. The event 'MSE_fromWikipedia_0_initItt' is highlighted in green. Other events include 'MSE_t4_2_fireItt', 'MSE_t2_1_fireItt', 'MSE_t3_4_fireItt', and 'MSE_t1_3_fireItt'.
- Console:** A window at the bottom showing the output of the GEMOC Execution Engine. The text includes: 'Modeling Workbench Console', 'About to initialize and run the GEMOC Execution Engine...', 'Input resources:', and '*** Model initialization done. (no modelInitialization method defined for the la...)'.

TRAVAIL PRATIQUE

Partie 2 : Modèle d'exécution du langage

-> Exécution : débogage de l'exemple importé

xDSML - platform:/resource/org.gemoc.sample.markedgraph.sample/wikipedia.aird/wikipedia MarkedGraph diagram - Gemoc Studio

File Edit Diagram Navigate Search Project Run Window Help

Stimuli Manager Gemoc Engines Status Concurrent Logical Steps Decider MultiBranch Timeline

Logical Steps/MSEs DSA

- LogicalStep [1395966319]
 - MSE_t1_3_fireIt Transition->t1.fire()



xDSML - platform:/resource/org.gemoc.sample.markedgraph.sample/wikipedia.aird/wikipedia MarkedGraph diagram - Gemoc Studio

File Edit Diagram Navigate Search Project Run Window Help

Stimuli Manager Gemoc Engines Status Concurrent Logical Steps Decider MultiBranch Timeline

Logical Steps/MSEs DSA

- LogicalStep [921497246]
 - MSE_t3_4_fireIt Transition->t3.fire()
- LogicalStep [1039522550]
 - MSE_t2_1_fireIt Transition->t2.fire()
- LogicalStep [1571691227]
 - MSE_t2_1_fireIt Transition->t2.fire()
 - MSE_t3_4_fireIt Transition->t3.fire()

Evènements en cours. La progression se fait au moyen du « double-clic » permettant de terminer un évènement et lancer les prochains (ceux engendrés par sa terminaison).

Partie 3 : Animation graphique de l'exécution des graphes marqués

TRAVAIL PRATIQUE

Partie 3 : Animation graphique

-> Vue d'animation

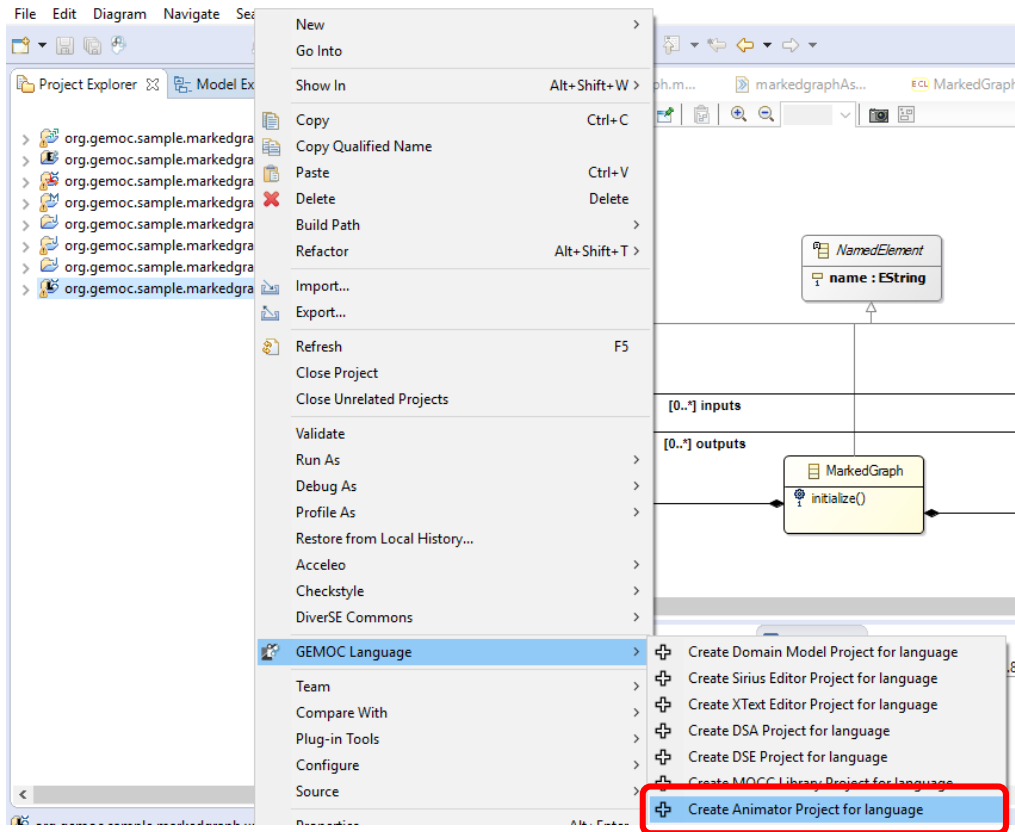
Nous créons la vue d'animation comme un projet « Animator » :

- Elle est associée au meta-modèle ecore.
- Elle intègre la représentation graphique des graphes marqués, définie dans le projet sirius.
- En plus des éléments graphiques du projets sirius, notre vue définira des customisations (changements de couleurs de certains éléments) à l'exécution, basées sur des valeurs de vérité de prédicats.

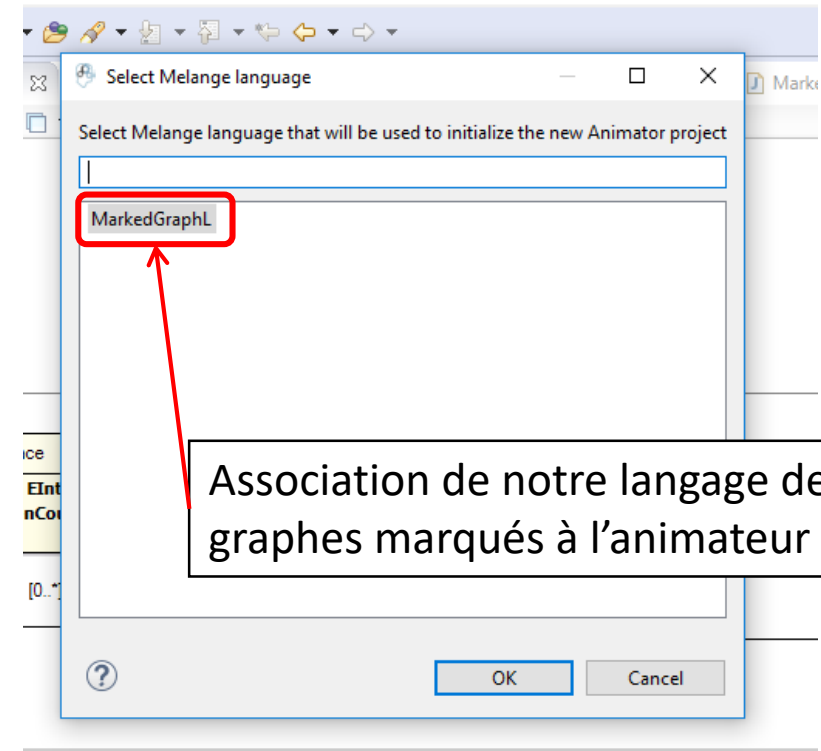
TRAVAIL PRATIQUE

Partie 3 : Animation graphique

-> Création du projet animateur



arkedgraph.aird/markedgraph class diagram - Gemoc Studio

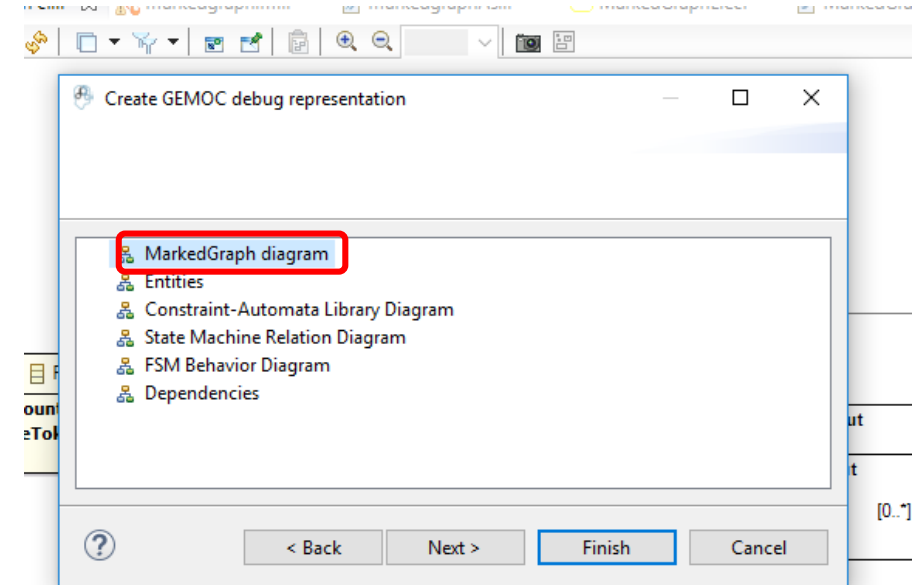
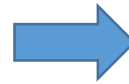
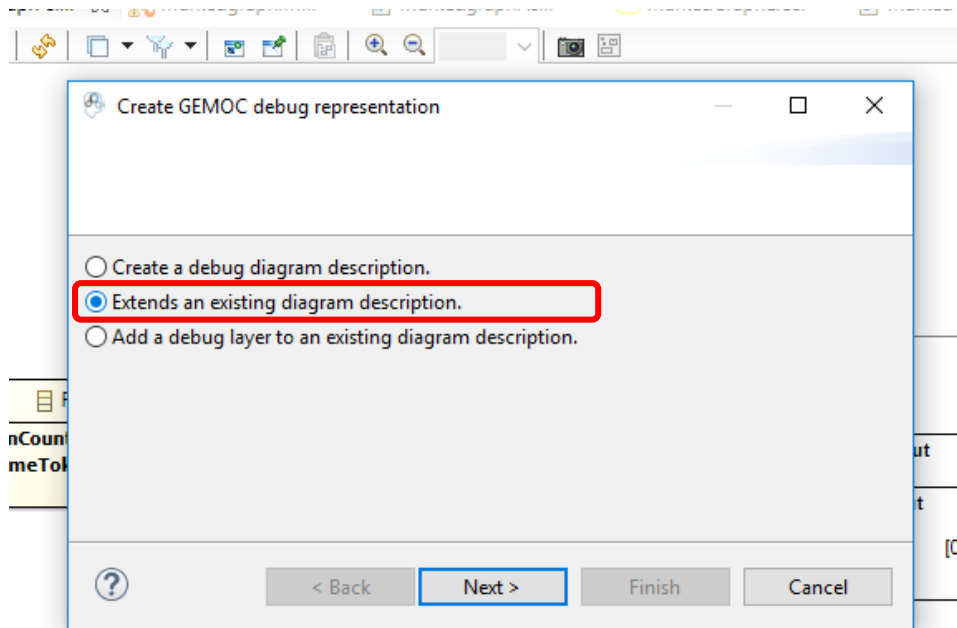


TRAVAIL PRATIQUE

Partie 3 : Animation graphique

-> Création du projet animateur

Association du débogage de l'animateur au diagramme de graphes marqués

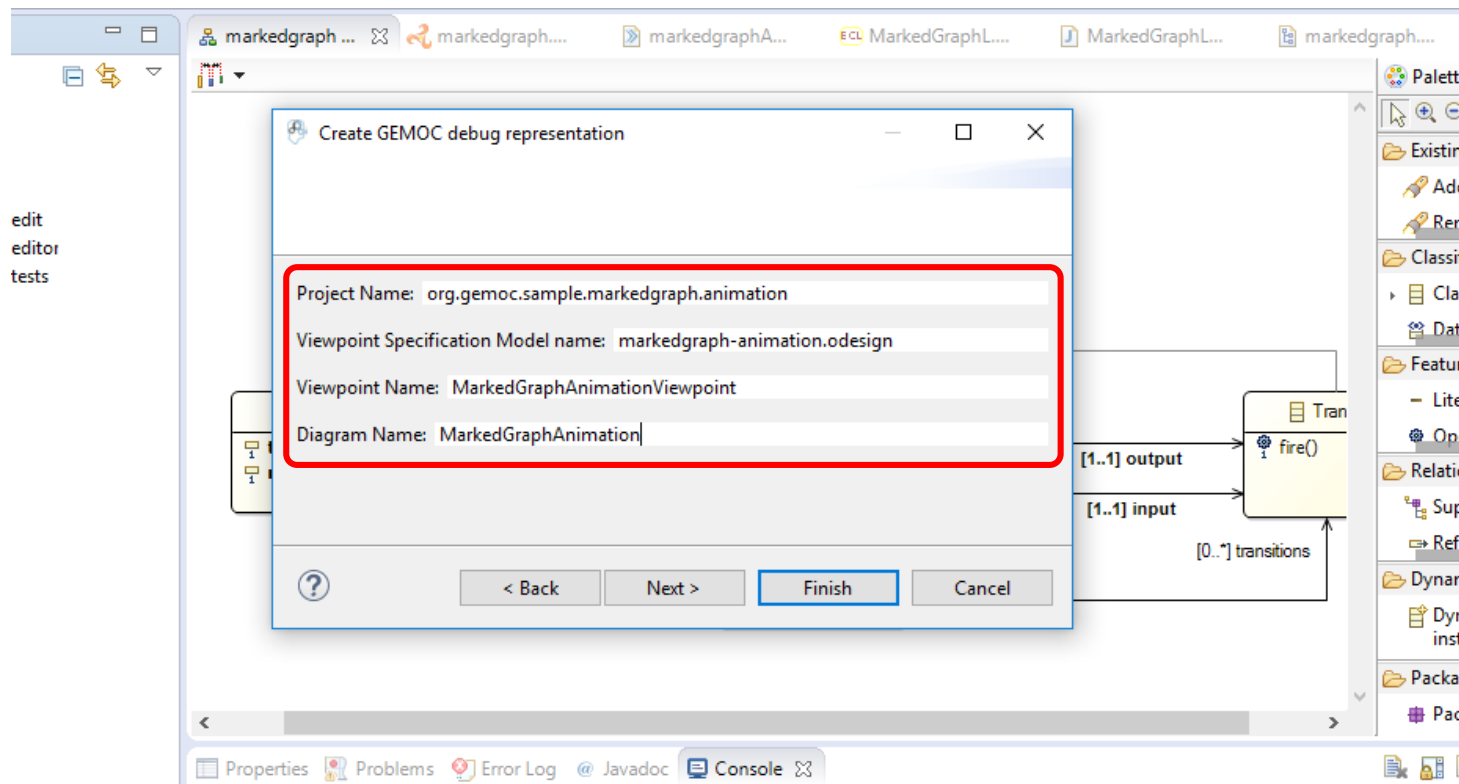


TRAVAIL PRATIQUE

Partie 3 : Animation graphique

-> Création du projet animateur

Définition du nom du projet et des autres noms associés à la vue animation



TRAVAIL PRATIQUE

Partie 3 : Animation graphique

-> Création du projet animateur

Aperçu de notre projet animateur

The screenshot displays the Sirius Specification Editor interface. The Project Explorer on the left shows the project structure for 'org.gemoc.sample.markedgraph.animation', with the 'description' folder containing the 'markedgraph-animation.odesign' file highlighted. The central Sirius Specification Editor shows the 'markedgraph-animation' package structure, including 'MarkedGraphAnimationViewpoint' and 'Diagram Extension MarkedGraphAnimation'. The Properties view at the bottom shows the configuration for the 'Diagram Extension MarkedGraphAnimation'.

Property	Value
Name*	MarkedGraphAnimation
Viewpoint URI*	viewpoint/org.gemoc.sample.markedgraph.design/markedgraph
Representation Name*	MarkedGraph diagram

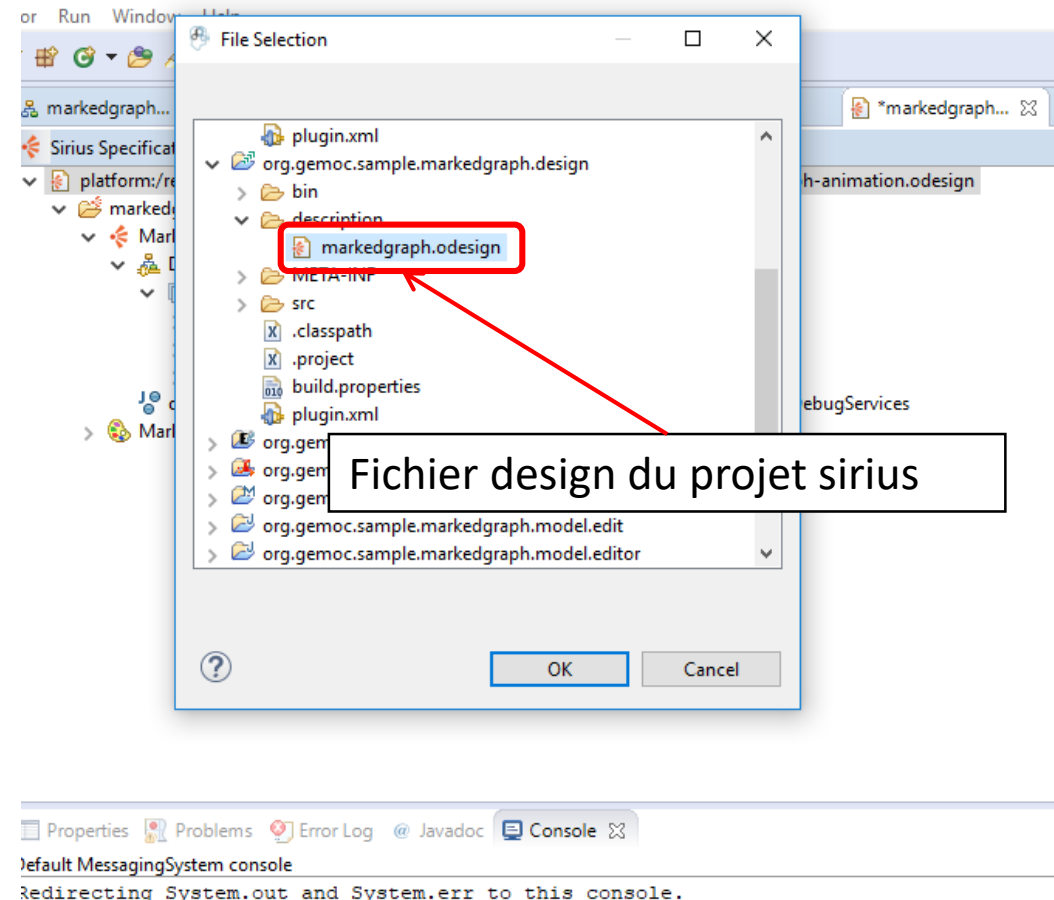
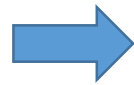
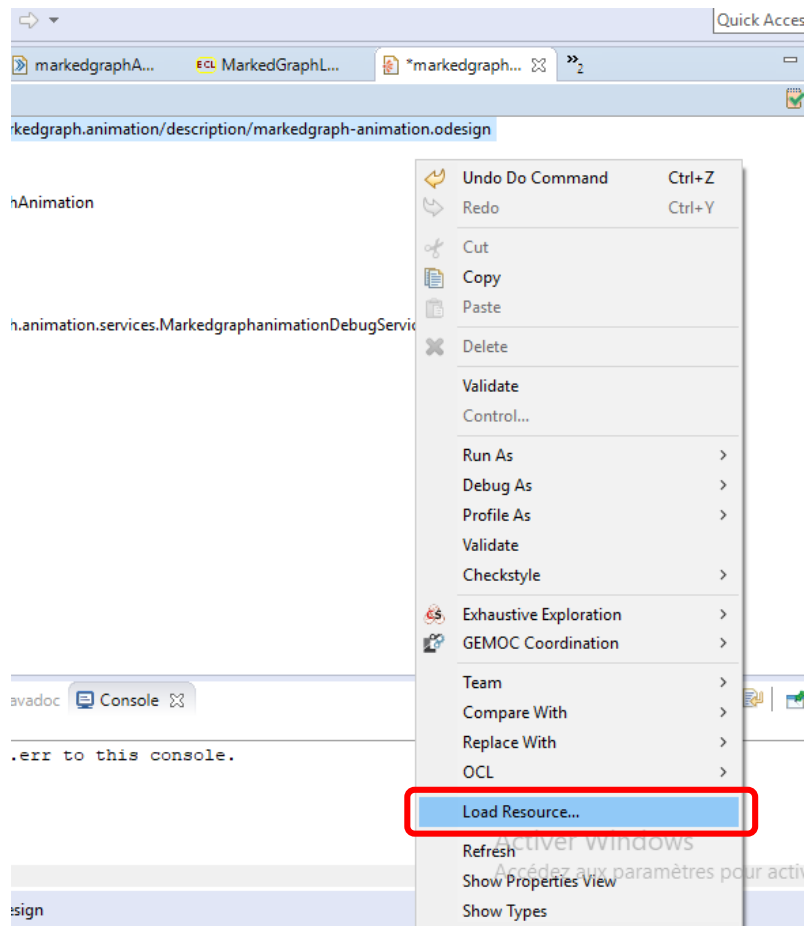
Vue/modification des propriétés des objets de l'animateur

Fichier « odesign » de l'animateur

TRAVAIL PRATIQUE

Partie 3 : Animation graphique

-> Ajout de l'éditeur graphique sirius comme ressource à l'animateur

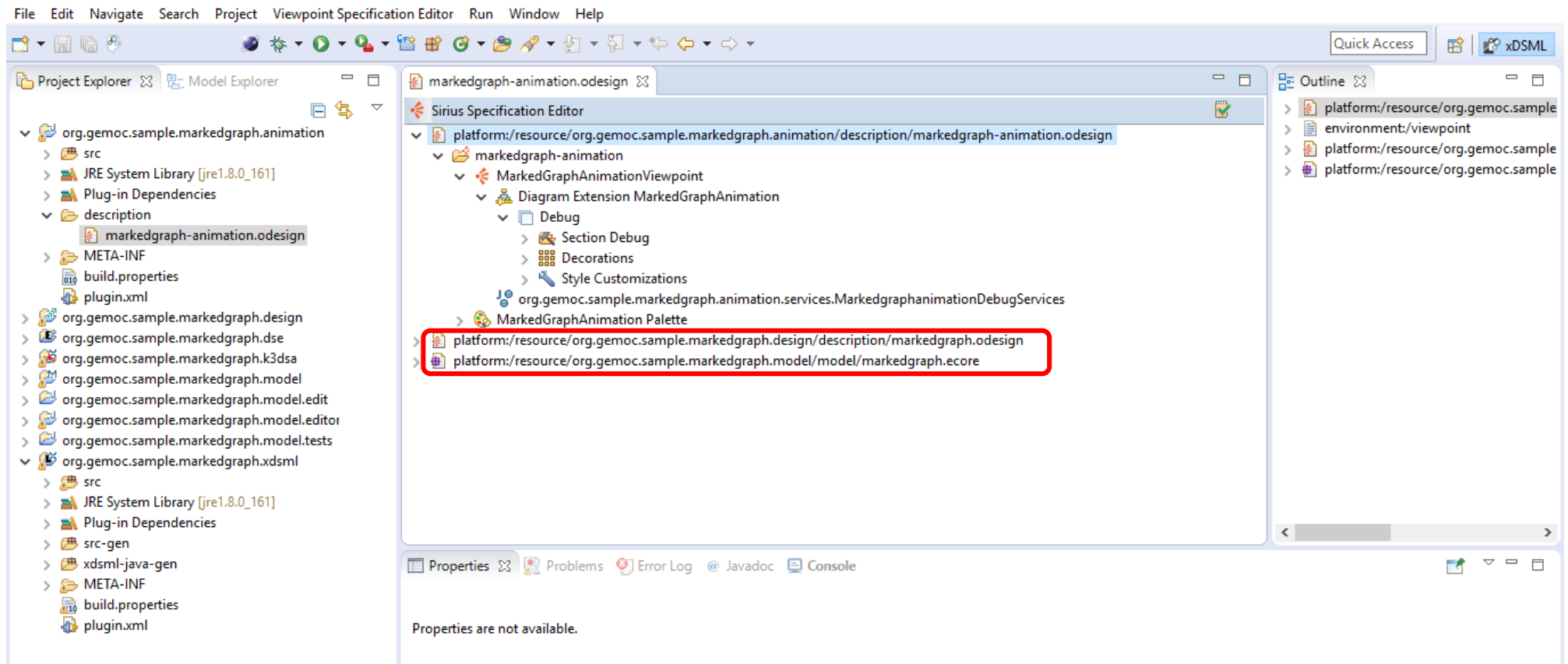


TRAVAIL PRATIQUE

Partie 3 : Animation graphique

-> Ajout de l'éditeur graphique sirius comme ressource à l'animateur

Aperçu de l'animateur après ajout



TRAVAIL PRATIQUE

Partie 3 : Animation graphique

-> Ajout du méta-modèle ecore au diagramme de l'animateur

The screenshot shows the Sirius Specification Editor interface. The left sidebar displays the project structure, with 'Diagram Extension MarkedGraphAnimation' selected and highlighted by a red box. The bottom panel shows the 'Properties' view for the selected element, with the 'Metamodels' section containing a table with columns for Name, nsURI, and metamodel URI. Below the table are two buttons: 'Add from registry' and 'Add from workspace', with the latter highlighted by a red box. A text box with a black border contains the text: 'Ajout du méta-modèle ecore en le sélectionnant parmi ceux présent dans l'espace de travail'. A red arrow points from this text box to the 'Add from workspace' button.

Name	nsURI	metamodel URI

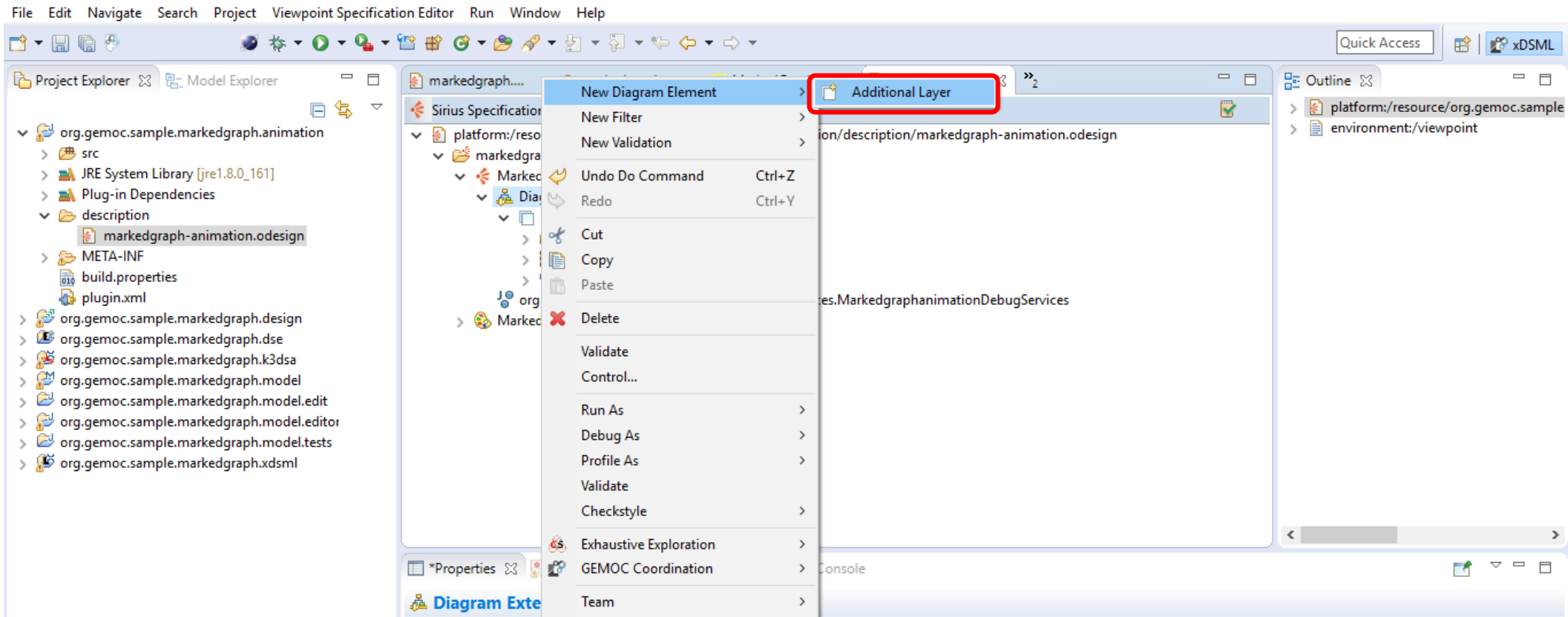


The screenshot shows the 'Ecore resource selection' dialog box. The dialog displays a tree view of the project structure, with 'markedgraph.ecore' selected and highlighted by a red box. The dialog has 'OK' and 'Cancel' buttons at the bottom.

TRAVAIL PRATIQUE

Partie 3 : Animation graphique

-> Ajout de la couche d'animation à l'animateur



TRAVAIL PRATIQUE

Partie 3 : Animation graphique

-> Ajout de la couche d'animation à l'animateur

The screenshot shows the Sirius Specification Editor interface. The Project Explorer on the left displays the project structure for 'org.gemoc.sample.markedgraph.animation'. The central Sirius Specification Editor shows the 'MarkedGraphAnimationViewpoint' configuration, with the 'Animation' layer highlighted in red. The Properties view at the bottom shows the 'Additional Layer' configuration, where the 'Id*' and 'Label' fields are both set to 'Animation' and are also highlighted in red. A red arrow points from a text box to the 'Id*' field.

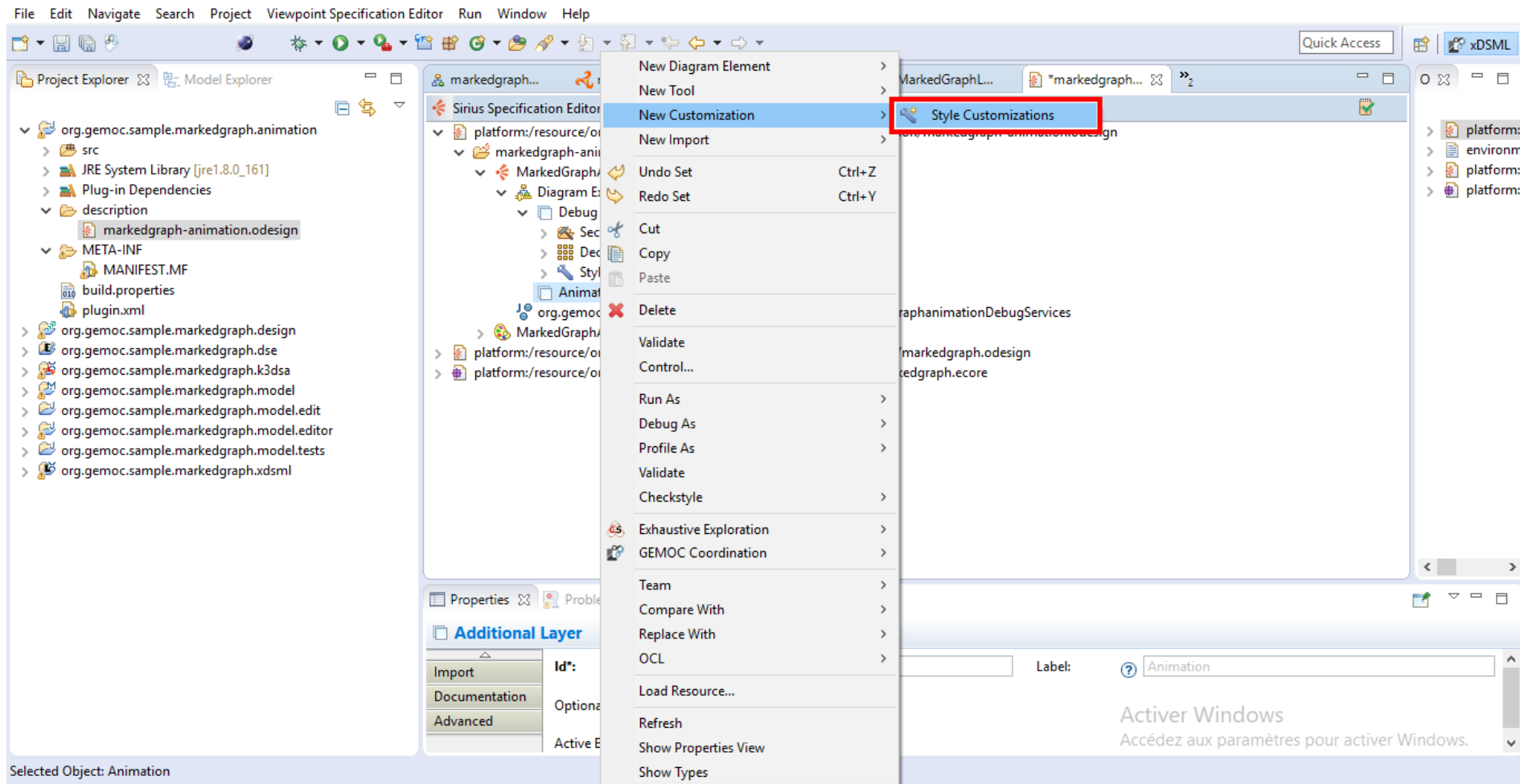
Propriété de la couche :
assignation du nom

Import	Id*	Label
	Animation	Animation

TRAVAIL PRATIQUE

Partie 3 : Animation graphique

-> Couche animation : intégration d'une customisation de style

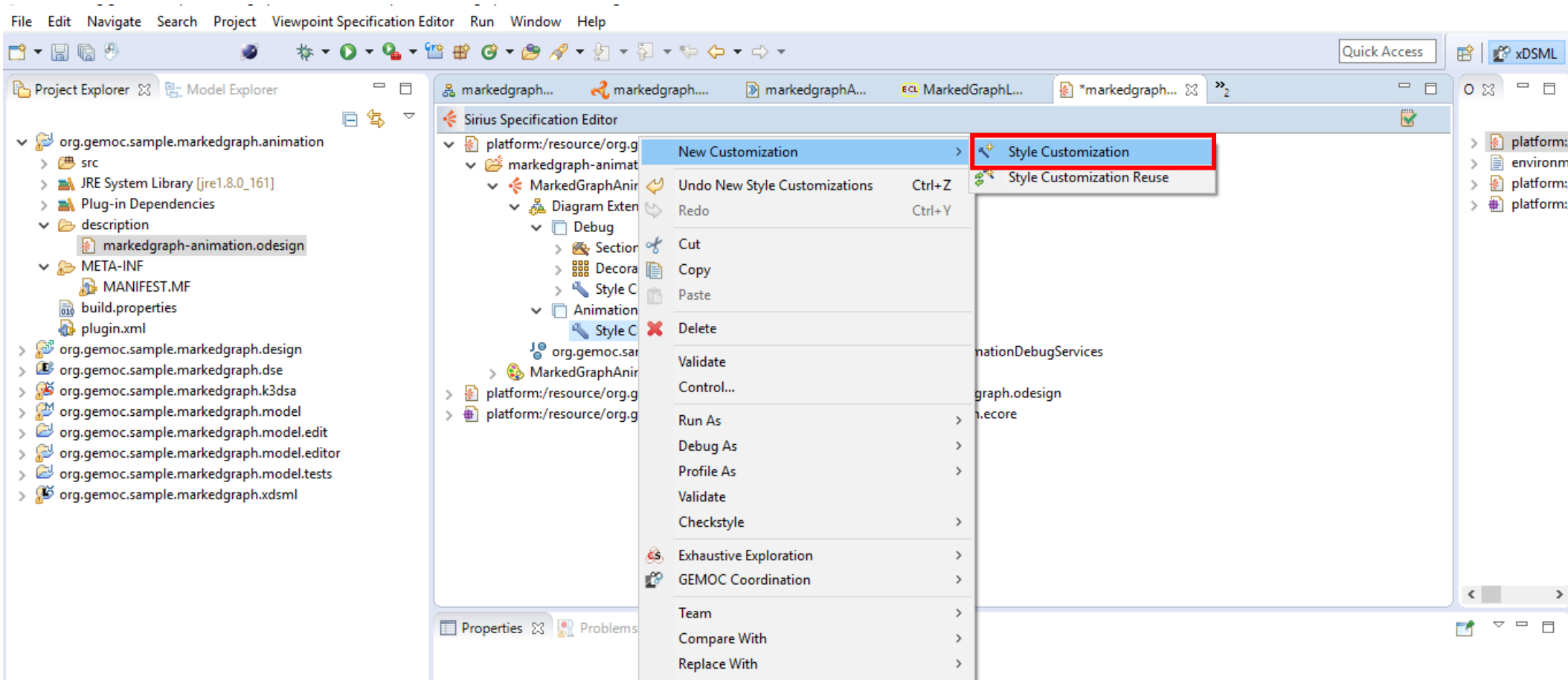


TRAVAIL PRATIQUE

Partie 3 : Animation graphique

-> Couche animation : intégration d'une customisation de style

On rajoute une « customisation de style » à la customisation de style pour définir un prédicat



TRAVAIL PRATIQUE

Partie 3 : Animation graphique

-> Couche animation : intégration d'une customisation de style

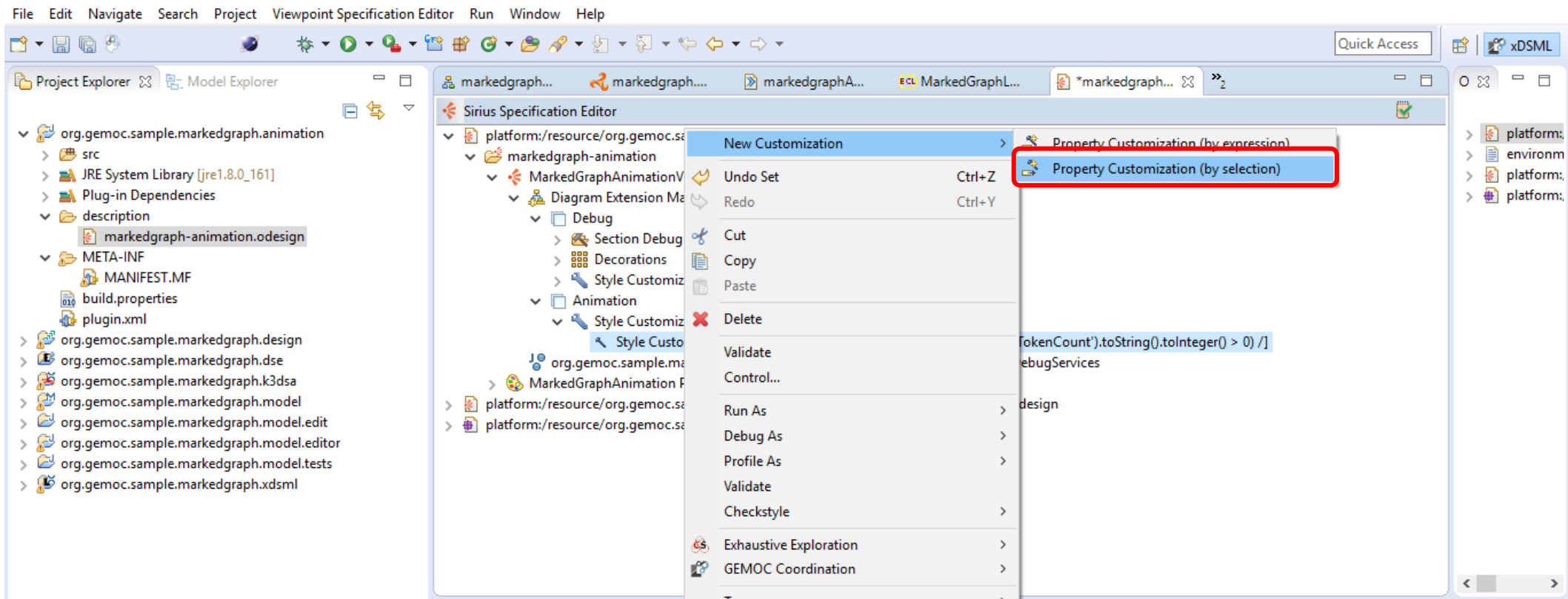
Intégration du prédicat de la customisation de style

The screenshot displays the Sirius Specification Editor interface. The left sidebar shows the Project Explorer with the project structure. The main editor area shows the Sirius Specification Editor with a tree view of the specification. A red box highlights a 'Style Customization' element with the predicate expression: `[self.eGet('inputs')->forAll(p | p.eGet('runtimeTokenCount').toString().toInteger() > 0) /]`. A callout box points to this element with the text: 'Prédicat sur lequel la customisation de style s'applique'. The bottom panel shows the 'Style Customization' properties, with the 'Predicate Expression' field containing the same predicate expression, also highlighted by a red box.

TRAVAIL PRATIQUE

Partie 3 : Animation graphique

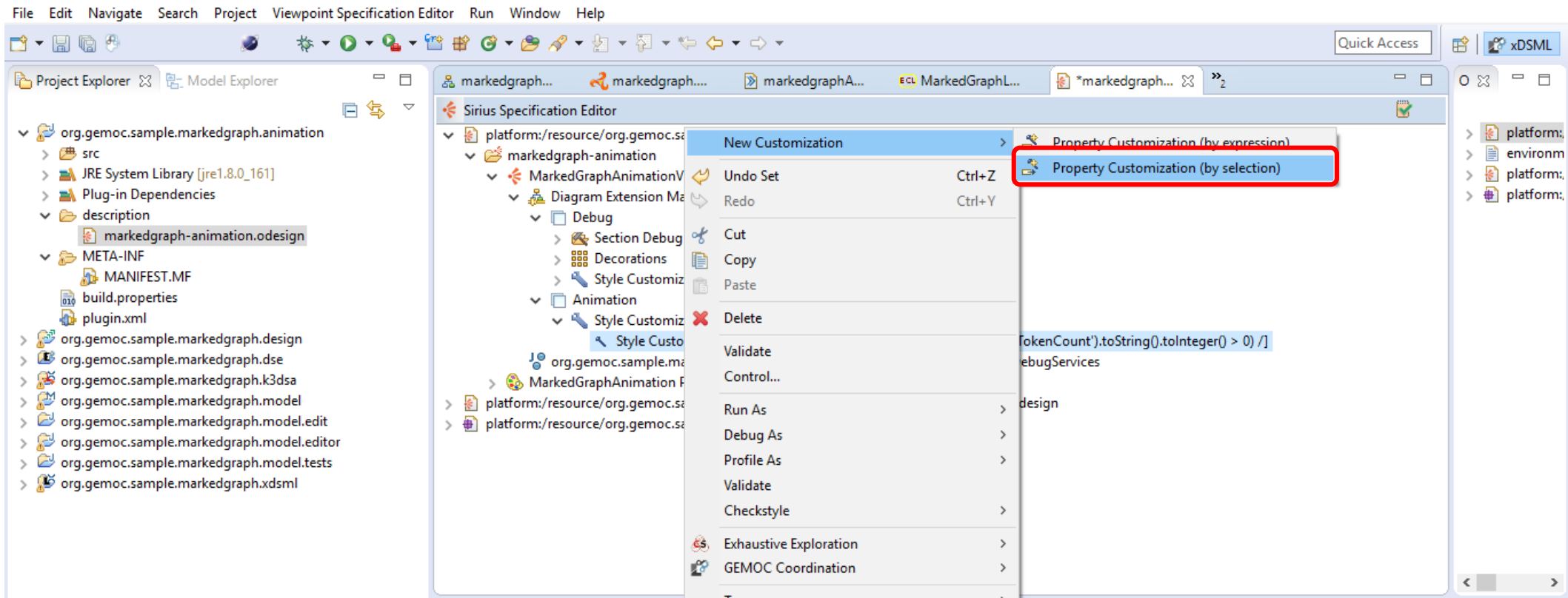
-> Couche animation : affichage des transitions activées en vert clair



TRAVAIL PRATIQUE

Partie 3 : Animation graphique

-> Couche animation : affichage des transitions activées en vert clair



TRAVAIL PRATIQUE

Partie 3 : Animation graphique

-> Couche animation : affichage des transitions activées en vert clair

The screenshot shows the Sirius Specification Editor interface. The Project Explorer on the left shows the project structure for 'org.gemoc.sample.markedgraph.animation'. The central editor displays the 'MarkedGraphAnimationViewpoint' with a 'Style Customization' rule highlighted in red. The rule is: `Style Customization [self.eGet('inputs')->forAll(i | p.eGet('runtimeTokenCount').toString().toInteger() > 0) /]`. Below the editor, the 'Property Customization (by selection)' configuration window is open, also highlighted in red. It shows the following settings:

Property Customization (by selection)	
Applied On*	Square white
Property Name*	color
Value Selection:	light_green

A red arrow points from the configuration window to the text box on the right.

Définition de la propriété de la customisation

TRAVAIL PRATIQUE

Partie 3 : Animation graphique

-> Couche animation : réglage de la couleur de fond d'un lieu sur jaune clair

The screenshot displays the Sirius Specification Editor interface. The central workspace shows a tree view of the specification editor, with the following structure:

- platform:/resource/org.gemoc.sample.markedgraph.animation/description/markedgraph-animation.odesign
 - markedgraph-animation
 - MarkedGraphAnimationViewpoint
 - Diagram Extension MarkedGraphAnimation
 - Debug
 - Animation
 - Style Customizations
 - Style Customization [self.eGet('inputs')->forAll(p | p.eGet('runtimeTokenCount').toString()).toInt()
 - Property Customization (by selection) color** (highlighted with a red box)
 - Property Customization (by selection) color

The Properties view at the bottom shows the configuration for the selected **Property Customization (by selection)** element:

Property	Value
Apply On All:	<input type="checkbox"/>
Applied On*:	Ellipse white
Property Name*:	color
Value Selection:	light_yellow

A red arrow points from the text box to the selected property in the tree view.

Définition des propriétés de la customisation

TRAVAIL PRATIQUE

Partie 3 : Animation graphique

-> Couche animation : affichage du nombre de jetons sur un lieu

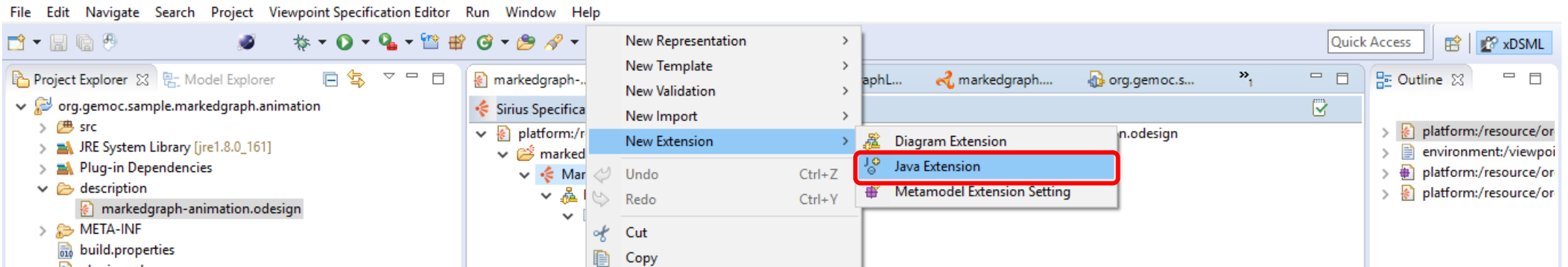
The screenshot shows the Sirius Specification Editor interface. The Project Explorer on the left lists the project structure. The Model Explorer in the center shows the hierarchy of the 'markedgraph-animation' viewpoint, with 'Property Customization (by expression) labelExpression' selected and highlighted with a red box. A callout box with the text 'Définition des propriétés de la customisation' points to this element. At the bottom, the Properties window for 'Property Customization (by expression)' is open, showing the following configuration:

Property Name*	Value Expression
Applied On*	Ellipse white
Property Name*	labelExpression
Value Expression	feature:runtimeTokenCount

TRAVAIL PRATIQUE

Partie 3 : Animation graphique

-> Ajout du service d'animation : définition d'une extension java pour le service



Nom de l'extension java associée au service

> org.gemoc.sample.markedgraph.model.tests
> org.gemoc.sample.markedgraph.xdsm1

org.gemoc.sample.markedgraph.animation.services.MarkedgraphanimationDebugServices
org.gemoc.sample.markedgraph.animation.services.MarkedgraphanimationAnimationServices
> MarkedGraphAnimation Palette
> platform:/resource/org.gemoc.sample.markedgraph.model/model/markedgraph.ecore
> platform:/resource/org.gemoc.sample.markedgraph.design/description/markedgraph.odesign

Properties Problems Error Log @ Javadoc Console

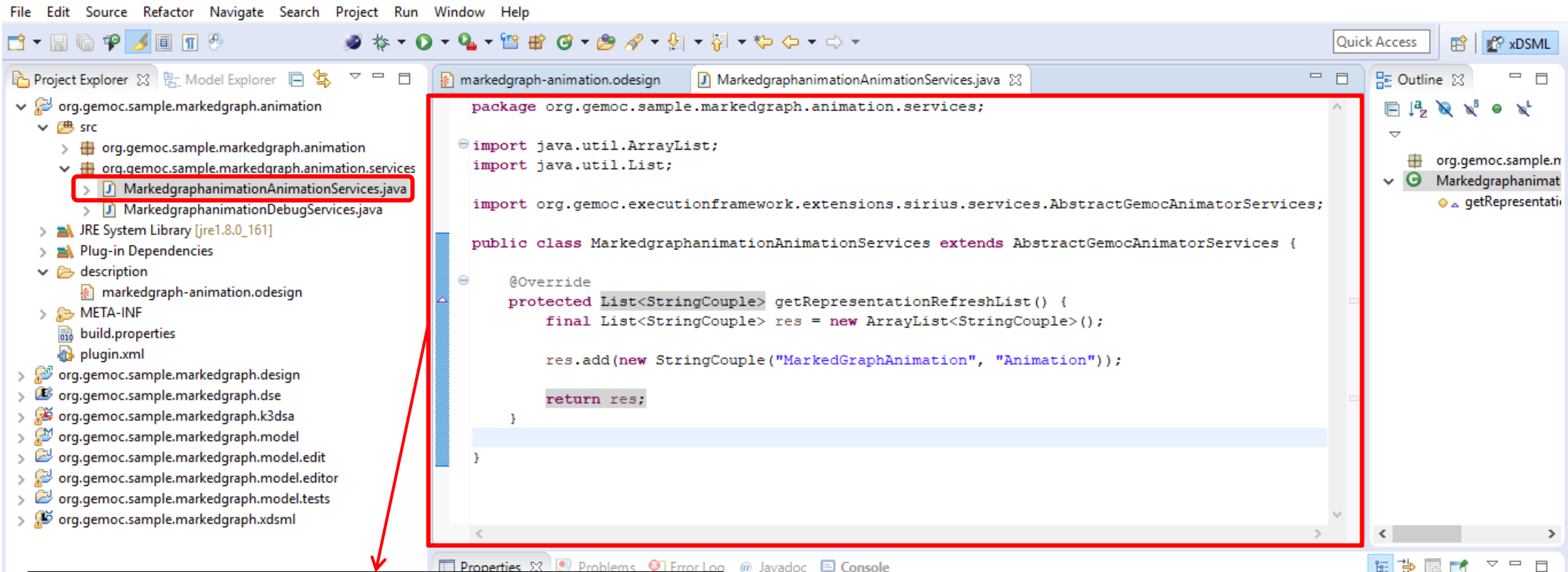
org.gemoc.sample.markedgraph.animation.services.MarkedgraphanimationAnimationServices

General Qualified Class Name*: org.gemoc.sample.markedgraph.animation.services.MarkedgraphanimationAnimationServices

TRAVAIL PRATIQUE

Partie 3 : Animation graphique

-> Ajout du service d'animation : création de la classe définissant le service



```
package org.gemoc.sample.markedgraph.animation.services;

import java.util.ArrayList;
import java.util.List;

import org.gemoc.executionframework.extensions.sirius.services.AbstractGemocAnimatorServices;

public class MarkedgraphanimationAnimationServices extends AbstractGemocAnimatorServices {

    @Override
    protected List<StringCouple> getRepresentationRefreshList() {
        final List<StringCouple> res = new ArrayList<StringCouple>();

        res.add(new StringCouple("MarkedGraphAnimation", "Animation"));

        return res;
    }
}
```

Contenu de la classe : copie modifiée de la classe « MarkedgraphDebugServices.java » où l'on remplace « Debugger » par « Animator » et « Debug » par « Animation »

TRAVAIL PRATIQUE

Partie 3 : Animation graphique

-> Exécution : préliminaire, correction du fichier plugin.xml du projet xDSML

Remplacez son contenu par le contenu ci-dessus. Ceci permet de bien référencer le plugin et d'éviter des erreurs d'exécution lors de l'animation

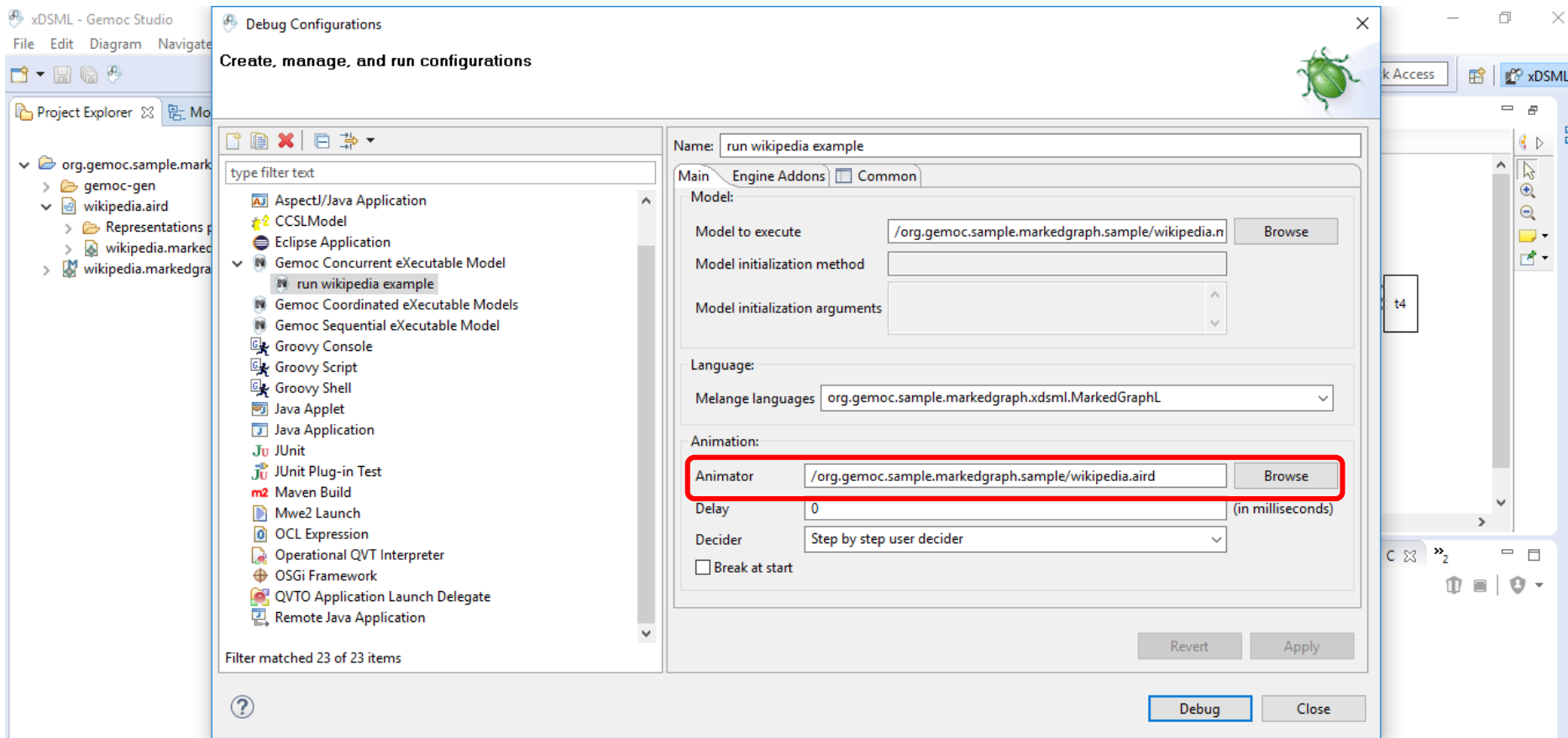
```
<plugin>
  <extension point="org.gemoc.gemoc_language_workbench.concurrent.xdxml">
    <XDSML_Definition codeExecutor_class="markedgraphl.xdxml.api.impl.MarkedGraphLCodeExecutor"
      modelLoader_class="org.gemoc.executionframework.extensions.sirius.modelloader.DefaultModelLoader"
      name="org.gemoc.sample.markedgraph.xdxml.MarkedGraphL"
      solver_class="org.gemoc.execution.concurrent.ccsjavaengine.extensions.timesquare.moc.impl.CcsISolver"
      toCCSLQVTOFilePath="/org.gemoc.sample.markedgraph.dse/qvto-gen/modeling/MarkedGraphL.qvto"
      xdxmlFilePath="/org.gemoc.sample.markedgraph.xdxml/src/org/gemoc/sample/markedgraph/xdxml/markedgraph.melange" />
  </extension>
  <extension point="fr.inria.diverse.melange.modeltype">
    <modeltype id="org.gemoc.sample.markedgraph.xdxml.MarkedGraphLMT" uri="http://markedgraphlmt/" />
  </extension>
  <extension point="fr.inria.diverse.melange.language">
    <language
      aspects="Transition:markedgraph.aspects.TransitionAspect;MarkedGraph:markedgraph.aspects.MarkedGraphAspect;Place:markedgraph.aspects.PlaceAspect"
      entryPoints="" exactType="org.gemoc.sample.markedgraph.xdxml.MarkedGraphLMT"
      id="org.gemoc.sample.markedgraph.xdxml.MarkedGraphL" uri="http://www.example.org/markedgraph">
      <adapter class="org.gemoc.sample.markedgraph.xdxml.markedgraphl.adapters.markedgraphlmt.MarkedGraphLAdapter"
        modeltypeId="org.gemoc.sample.markedgraph.xdxml.MarkedGraphLMT" />
    </language>
  </extension>
</plugin>
```


TRAVAIL PRATIQUE

Partie 3 : Animation graphique

-> Exécution : re-lancement de l'atelier de modélisation

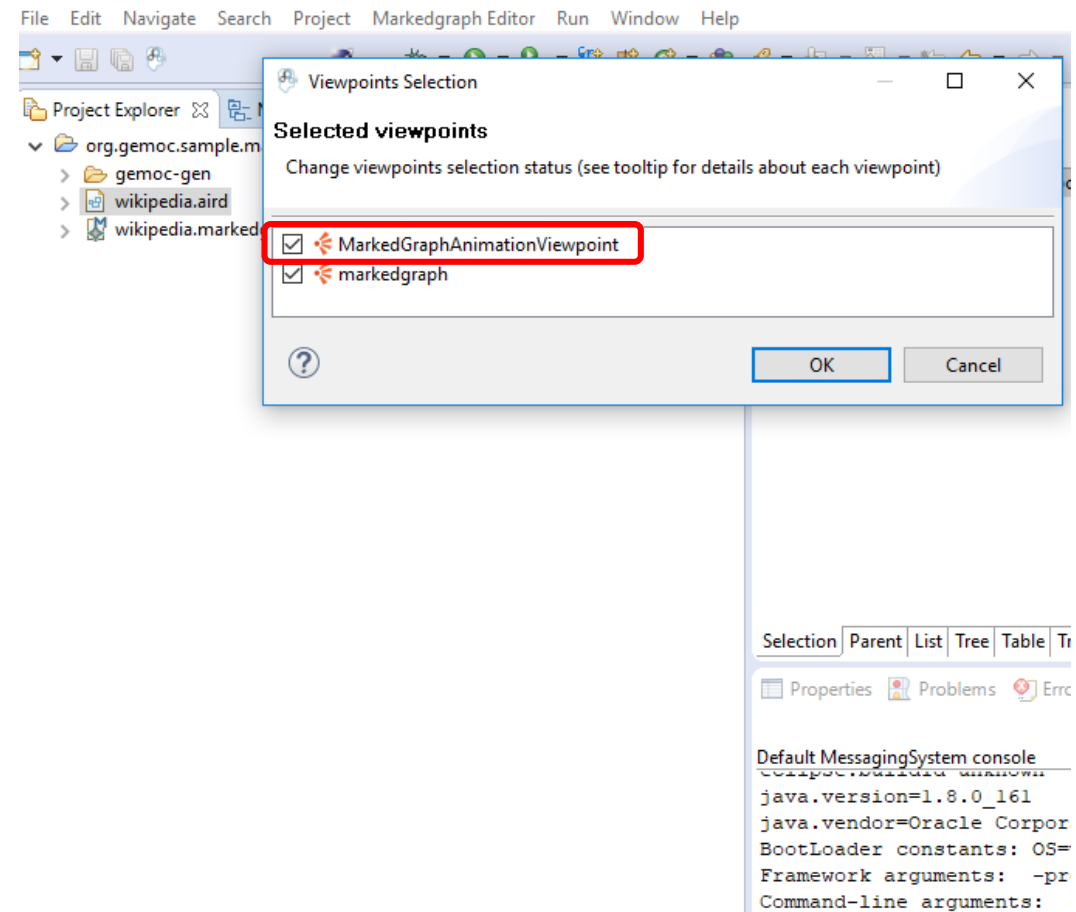
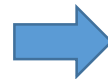
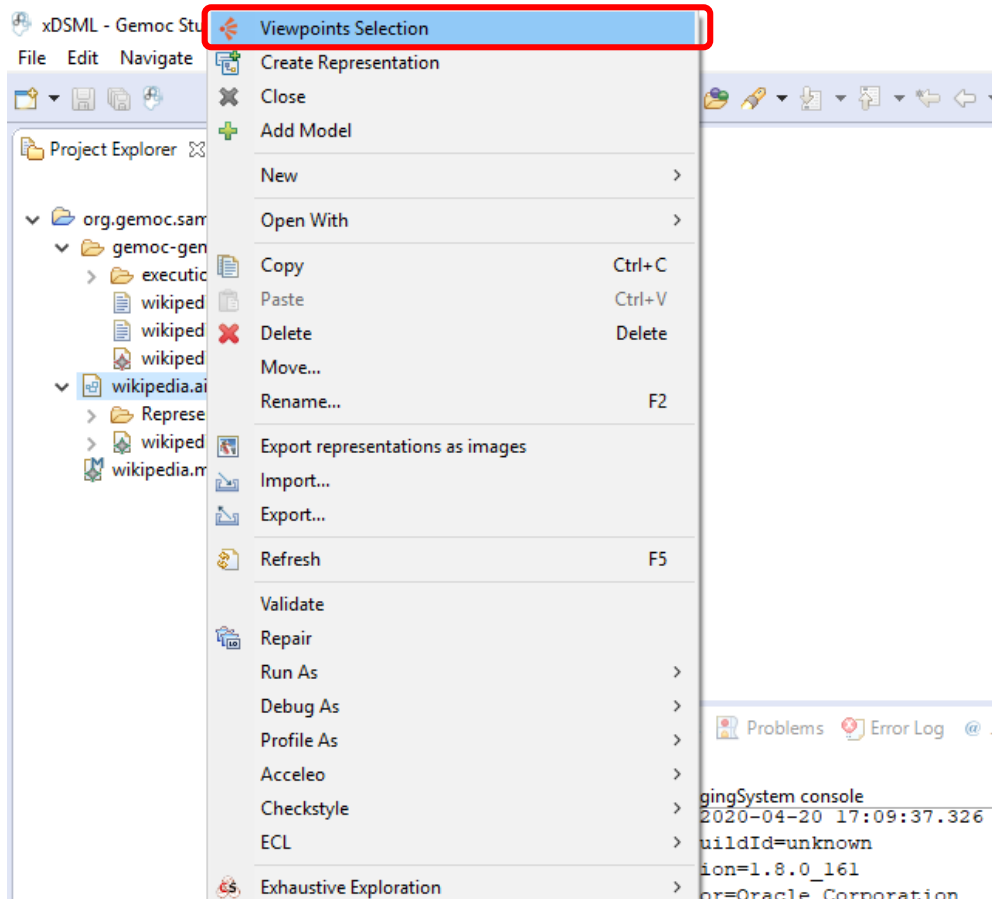
Après avoir redémarrer gemoc studio et lancer l'atelier de modélisation, mettre à jour la configuration de débogage en complétant la partie animation



TRAVAIL PRATIQUE

Partie 3 : Animation graphique

-> Exécution : ajout de la vue d'animation à l'exemple de graphes marqués



TRAVAIL PRATIQUE

Partie 3 : Animation graphique

-> Exécution : ajout de la vue d'animation à l'exemple de graphes marqués

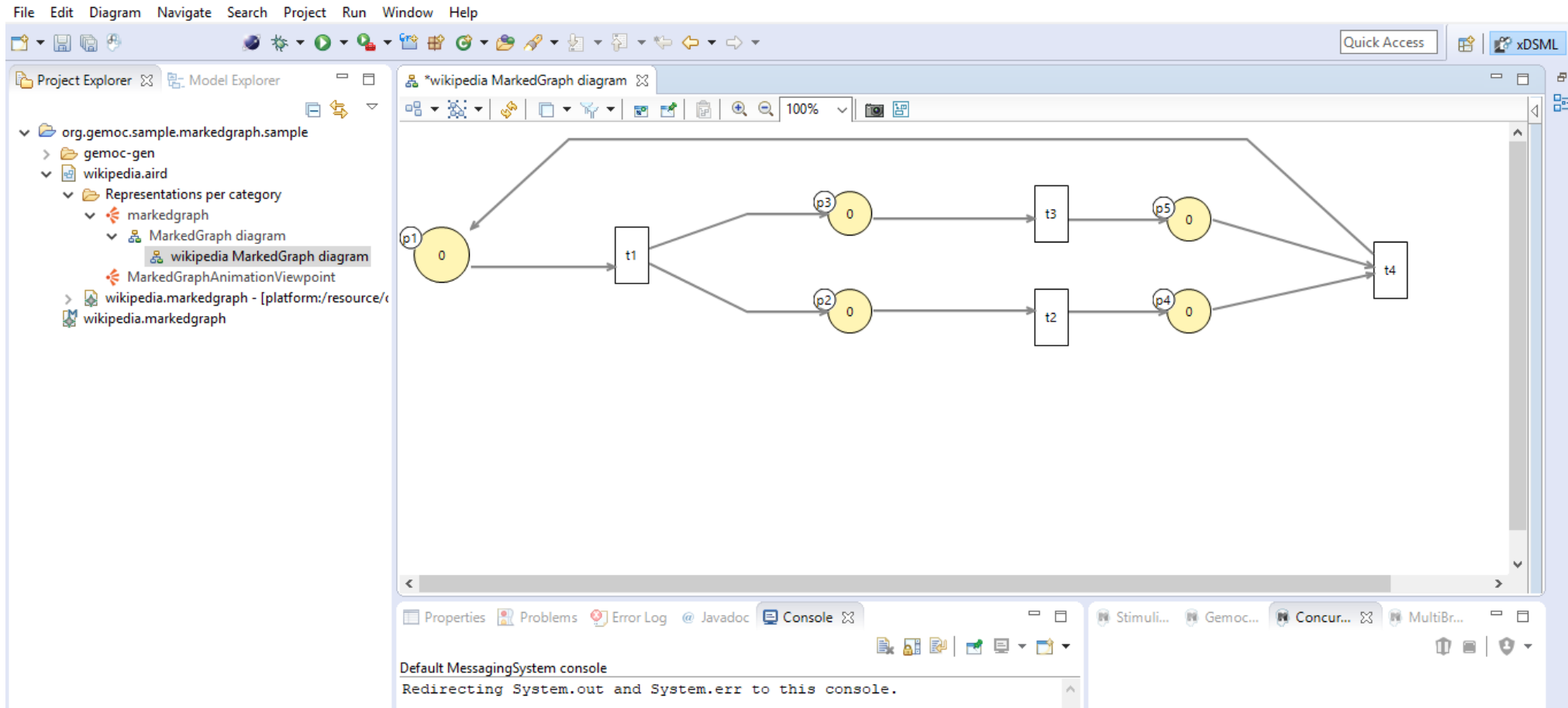
The screenshot displays an IDE interface with a project explorer on the left and a diagram editor in the center. The project explorer shows a hierarchy: `org.gemoc.sample.markedgraph.sample` > `gemoc-gen` > `wikipedia.aird` > `Representations per category` > `markedgraph` > `MarkedGraph diagram` > `wikipedia MarkedGraph diagram`. The diagram editor shows a MarkedGraph diagram with nodes `p1` (circle), `t1` (rectangle), `p3` (circle), `p2` (circle), `t3` (rectangle), `t2` (rectangle), `p5` (circle), `p4` (circle), and `t4` (rectangle). Edges connect `p1` to `t1`, `t1` to `p3` and `p2`, `p3` to `t3`, `p2` to `t2`, `t3` to `p5`, `t2` to `p4`, and `p5` and `p4` to `t4`. A toolbar above the diagram has a red box around the `Animation` button. The bottom status bar shows the console output: `Default MessagingSystem console` and `Redirecting System.out and System.err to this console.`

TRAVAIL PRATIQUE

Partie 3 : Animation graphique

-> Exécution : ajout de la vue d'animation à l'exemple de graphes marqués

Aperçu du graphe après ajout de la couche d'animation

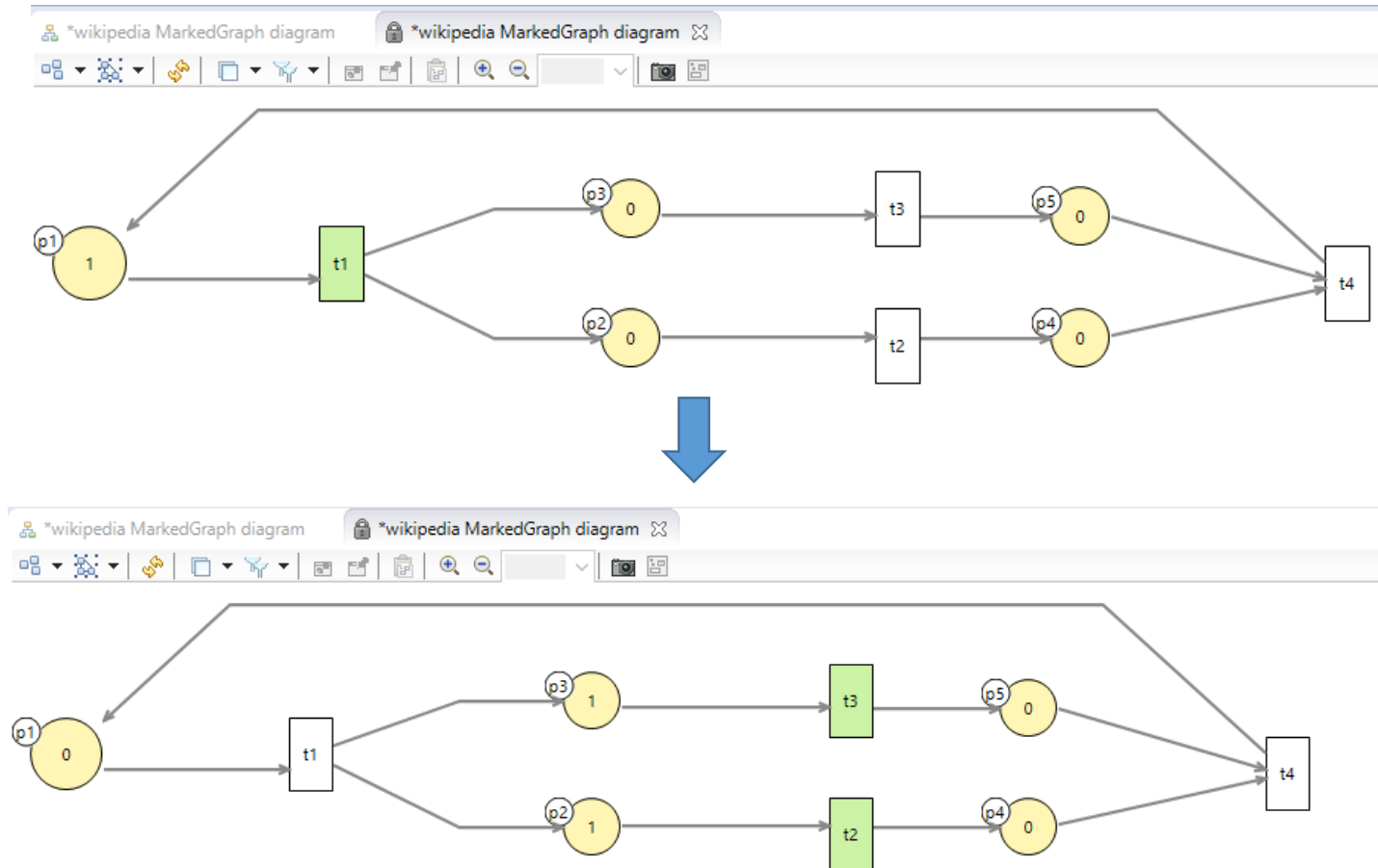


TRAVAIL PRATIQUE

Partie 3 : Animation graphique

-> Exécution : Animation

Lancer l'exécution du graphe grâce au bouton « run »



CONCLUSION ET PERSPECTIVES

CONCLUSION ET PERSPECTIVES

-> code source final

- ❑ Le code source final obtenu à la fin du travail pratique est disponible ici :
<https://drive.google.com/file/d/1uBkxLKhvhvcLMEPiGODQ0vDuNsTGCpKY/view?usp=sharing>
- ❑ Une vidéo de présentation de ce code est aussi disponible ici :
https://drive.google.com/file/d/11wMf26fG8Kauc36_uDIJ378aJqJHdCv4/view?usp=sharing

CONCLUSION ET PERSPECTIVES

-> perspectives

Les perspectives suivantes sont intéressantes si on veut aller plus loin dans la prise en main de l'outil Gemoc :

- Apprendre plus en profondeur la création des syntaxes graphiques avec Sirius.
- Apprendre plus en profondeur le langage **ECL** et **MoCCML** pour la définition d'évènements et contraintes d'évènements.
- Comprendre les notions de coordination entre langage et de définition de modèles hétérogènes proposés par Gemoc.

CONCLUSION ET PERSPECTIVES

-> discussions

Gemoc et les GAG ?