Eclipse Modeling Framework (EMF)

La version d'Eclipse à utiliser est la suivante :

```
/mnt/n7fs/ens/tp_dieumegard/eclipse_modeling_indigo/eclipse
```

Dans un TP précédent, nous avons saisi un métamodèle sous la forme d'un fichier Ecore. Nous allons ici nous intéresser à l'utilisation de ce métamodèle afin de générer de l'outillage et ainsi pouvoir manipuler des modèles instances de ce métamodèle.

1 Creation et manipulation d'éditeurs avec EMF

Exercice 1 : Engendrer le code Java et un éditeur arborescent
Nous commençons ici par générer les classes nécessaires à la création de notre éditeur.

1.1 *Configurer la génération du code Java.* Dans *EMF*, la génération de code Java est configurée à l'aide d'un modèle appelé *genmodel*. Pour créer ce modèle, cliquer droit sur notre fichier *SimplePDL.ecore*, puis faire *New / Other...* et rechercher *EMF Generator Model*. Cliquer sur *Next*, nommer le fichier genmodel (SimplePDL.genmodel fera l'affaire) et le placer dans le même dossier que le métamodèle. Nous importerons le modèle Ecore grace à l'importeur *Ecore model*, presser *Next*. Cliquer sur *Load* pour charger le fichier et terminer l'assistant. Le fichier *SimplePDL.genmodel* doit s'ouvrir automatiquement.

1.2 *Configurer la génération.* Ouvrir le fichier .genmodel. En séléctionnant sa racine (dans l'éditeur arborescent) et en ouvrant la vue *Properties* on peut modifier les options de génération de la structure de données Java. Par exemple, on peut décider si la valeur d'un attribut sera éditable ou non. Nous nous contenterons ici de garder les paramètres par default.

1.3 *Engendrer le code Java.* Cliquer à droite sur la racine (dans l'éditeur reflexif) et sélectionner l'action *Generate Model Code*. Cette action permet de générer la structure de données Java correspondant au métamodèle *Ecore*. Regarder dans le dossier src pour constater la génération des fichiers .java.

1.4 *Engendrer le code pour l'éditeur arborescent.* De la même manière que précédement effectuer les actions de génération : *Generate Edit Code* et *Generate Editor Code*. Ces actions nous permettent de générer un éditeur arborescent pour les modèles conformes au métamodèle SimplePDL. Cet éditeur est généré dans les nouveaux projets fr.enseeiht.simplepdl.edit et fr.enseeiht.simplepdl.editor.

1.5 Les plus attentifs d'entre vous auront remarqué une quatrième option nommé *Generate Test Code*. Elle permet de générer automatiquement une suite de Tests pour notre architecture. Une fois générée, cette suite de tests ne demanderais qu'à être complétée...

1.6 Afin de pouvoir se référer à des éléments du métamodèle SimplePDL dans la suite des TP et du BE, nous devons le rendre accessible dans Eclipse. Pour cela, faire un clique droit sur le métamodèle Ecore dans le Model Explorer et choisir l'action *Register EPackage*. Cette action référence le métamodèle dans un registre d'Eclipse et sera à effectuer à chaque redémarrage d'Eclipse.

Exercice 2 : Utiliser l'éditeur arborescent

Pour utiliser l'éditeur arborescent que l'on vient d'engendrer, il faut lancer un nouvel Eclipse.

2.1 *Lancer un nouvel Eclipse.* Sélectionner le projet *fr.enseeiht.simplePDL*, faire un clic droit, sélectionner *Run As / Run...*, puis *Eclipse Application*. Une nouvelle instance d'Eclipse se lance.

2.2 *Créer un projet.* Dans la nouvelle instance d'Eclipse, nous commençons, comme toujours, par créer un projet (*File / New / Project*, puis *General / Project*) que l'on peut appeler *ExemplePDL.*

2.3 *Lancer l'éditeur arborescent.* Dans le projet que l'on vient de créer, on peut lancer l'éditeur arborescent en faisant *New / Other...* puis dans *Example EMF Model Creation Wizards*, on sélectionne *SimplePDL Model*. C'est bien le notre ! On peut ensuite conserver le nom par défaut proposé pour le modèle (*My.SimplePDL*). Sur l'écran suivant, il faut choisir le *Model Object*, l'élément racine de notre modèle. On prend *Process*. On peut enfin faire *Finish*.

2.4 *Saisir un modèle de processus.* Le fichier `My.SimplePDL` est dans la fenêtre principale. Il contient l'élément *Process*. On peut cliquer à droite pour créer des activités (*WorkDefinition*) ou des dépendances (*WorkSequence*). Cet éditeur s'appuie sur la propriété *containment* pour savoir ce qui peut être créé (en utilisant *New Child* du menu contextuel). Par exemple, en se plaçant sur un élément *WorkDefinition*, on ne peut pas créer de *WorkSequence* puisque les référence *linkToPredecessors* ou *linkToSuccessors* sont des références avec *containment* positionné à faux.

Pour avoir accès aux propriétés, il est conseillé de repasser dans la perspective *Ecore* et d'utiliser la vue « Properties » (si elle ne s'est pas affichée lors du changement de perspective, faire *Windows > Show View > Properties*).

Créer le modèle de processus qui est présenté à la figure 1.

On peut utiliser l'action *Validate* du menu contextuel sur chacun des éléments du modèle. Ceci vérifie que cet élément et ses sous-éléments sont conformes au métamodèle SimplePDL.

2.5 *Quitter le deuxième Eclipse.* Maintenant que notre modèle de processus est saisi, nous pouvons le sauver et quitter Eclipse pour revenir au premier Eclipse que nous avions lancé.

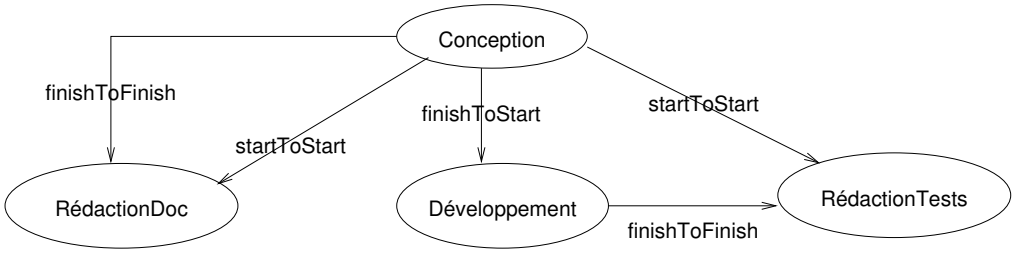

FIGURE 1 – Exemple de modèle de procédé

2 Utilisation du code Java/EMF pour manipuler des modèles

Exercice 3 : Manipulation de modèles en Java
Dans cet exercice, nous allons manipuler des modèles EMF à partir de code Java.

3.1 *Chargement de l'exemple de code.* Nous fournissons avec ce TP deux classes Java nommées *SimplePDLCreator.java* et *SimplePDLManipulator.java*. Dans le projet contenant les sources générées à l'aide du fichier *.genmodel*, ouvrir le dossier contenant les sources Java (dossier *src* à la racine du projet). Créer un *Package* Java nommé par exemple *SimplePDL.utils*. Importer les fichiers Java précédemment cités dans ce nouveau *Package*.

3.2 *Exemple de code pour la création de modèles.* Comprendre le contenu du fichier *SimplePDLCreator.java* puis l'exécuter. Constater qu'un nouveau dossier *models* a été créé dans le projet (faire *Refresh*, F5, sur le projet si le dossier n'apparaît pas). Ouvrir le fichier *SimplePDL-Creator_Created_Process.xmi* qu'il contient et vérifier son contenu.
Remarque : Pour exécuter une classe Java contenant une méthode *main*, cliquer droit sur le fichier source à exécuter puis *Run As / Java Application*. Le résultat de l'exécution doit s'afficher dans la console d'Eclipse.

3.3 *Exemple de code pour la manipulation de modèles.* Comprendre le contenu du fichier *SimplePDLManipulator.java* puis l'exécuter. Vérifier les affichages produits dans la console.

3.4 Écrire un code Java qui transforme un modèle de processus en un modèle de réseau de Pétri. Appliquer cette transformation à l'instance précédemment généré.

Il est conseillé d'avancer progressivement. En particulier d'exécuter régulièrement le programme pour constater les effets sur le modèle produit.

On pourra utiliser l'exemple contenu dans le dossier *models* mais il faudra envisager d'autres exemples car il ne couvre pas tous les cas.