Transformation de modèle à texte

La version d'Eclipse à utiliser est la suivante :

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

Nous avons vu dans les TP précédents comment saisir ou modifier un modèle en utilisant un éditeur reflexif arborescent. Nous allons maintenant nous intéresser à la transformation d'un modèle en sa représentation textuelle. On parle ici de transformation modèle vers texte (M2T). Nous allons utiliser l'outil Acceleo [1] de la société Obeo.

1 Transformation de modèle à texte avec Acceleo

Nous commençons par engendrer une syntaxe concrète à partir d'un modèle SimplePDL. Ensuite, nous engendrerons un fichier *dot* pour pouvoir visualiser graphiquement un modèle de procédé.

Exercice 1 : Engendrer une syntaxe textuelle d'un modèle
Dans un premier temps, nous souhaitons pouvoir engendrer la représentation d'un modèle SimplePDL dans une syntaxe concrète textuelle.

Voici la syntaxe choisie illustrée sur un modèle de processus.

```
process ExempleProcessus {
      wd RedactionDoc
      wd Conception
      wd Developpement
      wd RedactionTests
      ws Conception f2f RedactionDoc
      ws Conception s2s RedactionDoc
      ws Conception f2s Developpement
      ws Conception s2s RedactionTests
      ws Developpement f2f RedactionTests
}
```

Le principe d'Acceleo est de s'appuyer sur des templates des fichiers à engendrer. Le template qui correspond à la syntaxe PDL1 est donné au listing 1.

1.1 Expliquer les différents éléments qui apparaissent sur le listing 1. On pourra s'appuyer sur la documentation fournie dans Eclipse (*Help > Help Contents > Acceleo Model To Text Transformation Language*).

1. www.acceleo.org

Listing 1 – Template Acceleo pour engendrer la syntaxe PDL1 à partir d'un modèle SimplePDL

```
1  [comment encoding = UTF-8 /]
2  [module toPDL('http://simplepdl')]
3
4  [comment Generation de la syntaxe PDL1 à partir d'un modèle de processus /]
5
6  [template public toPDL(proc : Process)]
7  [comment @main/]
8  [file (proc.name.concat('.pdl1'), false, 'UTF-8')]
9  process [proc.name/]{
10 [for (wd : WorkDefinition | proc.processElements->getWDs())]
11       wd [wd.name/]
12 [/for]
13 [for (ws : WorkSequence | proc.processElements->getWSs())]
14       ws [ws.predecessor.name/] [ws.getWSType()/] [ws.successor.name/]
15 [/for]
16 }
17 [/file]
18 [/template]
19
20 [query public getWDs(elements : OrderedSet(ProcessElement)) : OrderedSet(WorkDefinition) =
21       elements->select( e | e.oclIsTypeOf(WorkDefinition) )
22             ->collect( e | e.oclAsType(WorkDefinition) )
23             ->asOrderedSet()
24 /]
25
26 [query public getWSs(elements : OrderedSet(ProcessElement)) : OrderedSet(WorkSequence) =
27       elements->select( e | e.oclIsTypeOf(WorkSequence) )
28             ->collect( e | e.oclAsType(WorkSequence) )
29             ->asOrderedSet()
30 /]
31
32 [template public getWSType(ws : WorkSequence)]
33 [if (ws.linkType = WorkSequenceType::startToStart)]
34 s2s[elseif (ws.linkType = WorkSequenceType::startToFinish)]
35 s2f[elseif (ws.linkType = WorkSequenceType::finishToStart)]
36 f2s[elseif (ws.linkType = WorkSequenceType::finishToFinish)]
37 f2f[/if]
38 [/template]
```

Exercice 2 : Créer et appliquer un template Acceleo
Pour créer et appliquer un template Acceleo, nous allons nous servir du métamodèle de SimplePDL.

2.1 *Importer le métamodèle SimplePDL.* Dans un nouveau projet eclipse, importer le métamodèle de SimplePDL ainsi que son instance (developpement.simplepdl), fournis en annexe, dans un nouveau projet (fr.enseeiht.simplepdl.m2t).

2.2 *Créer un projet de génération Acceleo.* Pour créer un projet de génération Acceleo, faire *New* > *Other* > *Acceleo Model to Text* > *Acceleo Project*.
Donner un nom au projet (fr.enseeiht.simplepdl.m2t) puis faire *Next*. Dans la fenêtre qui s'affiche nous allons définir les paramêtres de notre génération Acceleo. Saisir le nom du module (toPDL1), cliquer sur le + pour sélectionner le metamodèle SimplePDL (en prenant soin de sélectionner *Runtime Version*). Définir le nom du template (toPDL1). Cocher les cases afin de générer un fichier et un template de type *main*. Faire *Finish* pour terminer la création du projet.

2.3 Un nouveau projet a été engendré. Ce projet contient un dossier de sources. Dans le paquetage fr.enseeiht.simplepdl.m2t.main, un template de génération a été engendré (toPDL1.mtl). Ouvrir ce fichier.

2.4 *Saisir le template.* Remplacer le contenu du fichier *toPDL1.mtl* par celui du listing 1.

2.5 Pour exécuter la transformation m2t, cliquer droit sur le fichier `.mtl`, faire *Run as* > *Launch Acceleo Application*. Dans la fenêtre de configuration de la transformation sélectionner le modèle d'entrée (dans le champ *Model*) ainsi qu'un dossier cible de la transformation (*Target*) où sera engendré le résultat de la transformation.

Exercice 3 : Application à la génération d'un fichier .dot
Écrire une transformation modèle à texte qui permet de traduire un modèle de procédé en une syntaxe dot. Voici un exemple simple de fichier dot pour le même modèle de processus :

```
digraph ExempleProcessus {
      Conception->RedactionDoc
      Conception->RedactionDoc
      Conception->Developpement
      Conception->RedactionTests
      Developpement->RedactionTests
}
```

Une fois le fichier `.dot` obtenu, on obtient le graphe correspondant en PDF en faisant :

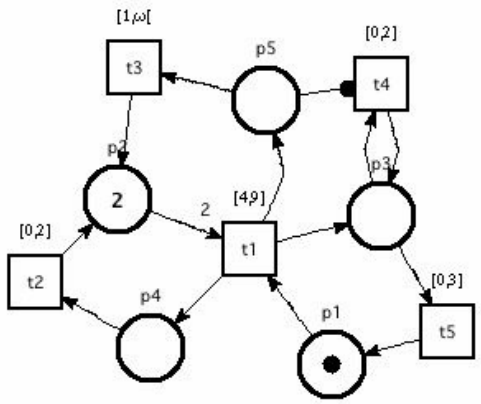
```
dot ficher.dot -Tpdf -o fichier.pdf
```

La documentation et des exemples concernant le langage *dot* peuvent être trouvés à l'URL : `http://www.graphviz.org/Documentation.php`

3.1 Écrire et tester le template `toDot.mtl` pour engendrer un fichier `.dot` correspondant à un modèle de processus. On pourra ajouter un template principal à notre projet de génération avec : *New* > *Other* > *Acceleo Model to Text* > *Acceleo Main Module File*.

2 Application aux réseaux de Petri

Exercice 4 : Transformations modèle à texte pour les réseaux de Petri
Nous allons maintenant définir une transformation modèle à texte pour les réseaux de Petri. L'objectif est d'engendrer la syntaxe textuelle utilisée par les outils Tina [2], en particulier `nd` (NetDraw). La figure 1(a) illustre les principaux concepts présents des réseaux de Petri temporels que nous considérons. Le fichier textuel de ce réseau est donné au listing 1(b) au format Tina.



(a) syntaxe graphique

```
1 tr t1 [4,9] p1 p2*2 -> p3 p4 p5
2 tr t2 [0,2] p4 -> p2
3 tr t3 [1,w[ p5 -> p2
4 tr t4 [0,2] p3 p5?1 -> p3
5 tr t5 [0,3] p3 -> p1
6 pl p1 (1)
7 pl p2 (2)
8 net ifip
```

(b) syntaxe textuelle

FIGURE 1 – Exemple de réseau de Petri

4.1 Écrire un template Acceleo pour transformer un modèle de réseau de Petri en un fichier `.net` de Tina. Pour visualiser le réseau, faire `nd fichier.net`, puis *edit* > *draw*.

4.2 Écrire un template Acceleo pour transformer un modèle de réseau de Petri en un fichier `.dot` qui permettra de visualiser graphiquement le réseau.

2. http ://www.laas.fr/tina/