# Transformation de modèle à texte

La version d'Eclipse à utiliser est la suivante :

```
/mnt/n7fs/ens/tp_babin/gls/eclipse-gls/bin/eclipse-gls
```

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 : Comprendre la définition d'un syntaxe concrète textuelle avec Acceleo**
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 developpement {
      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*).

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 toPDL1('http://simplepdl')]
3
4  [comment Generation de la syntaxe PDL1 Ã  partir d'un modÃ¨le de processus /]
5
6  [template public toPDL1(aProcess : Process)]
7  [comment @main/]
8  [file (aProcess.name.concat('.pdl1'), false, 'UTF-8')]
9  process [aProcess.name/] {
10 [for (wd : WorkDefinition | aProcess.processElements->getWDs())]
11       wd [wd.name/]
12 [/for]
13 [for (ws : WorkSequence | aProcess.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.

On utilisera le métamodèle et les modèles SimplePDL des TP précédents.

**2.1** *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.topdl1) puis faire *Next*. Dans la fenêtre qui s'affiche nous allons définir les paramètres de notre génération Acceleo (fig. 1). Il s'agit de :

1. saisir le nom du module : `toPDL1`
2. sélectionner le métamodèle : `http://simplepdl`. Il faut cliquer sur le `+`, cocher *Runtime Version* et utiliser le motif `*simp`.
3. sélectionner le type de l'élément sur lequel s'appliquera la transformation : `Process`
4. définir le nom du template : `toPDL1`
5. cocher *Generate file* et *Main template*.

Faire *Finish* pour terminer la création du projet. Un nouveau projet est alors engendré.

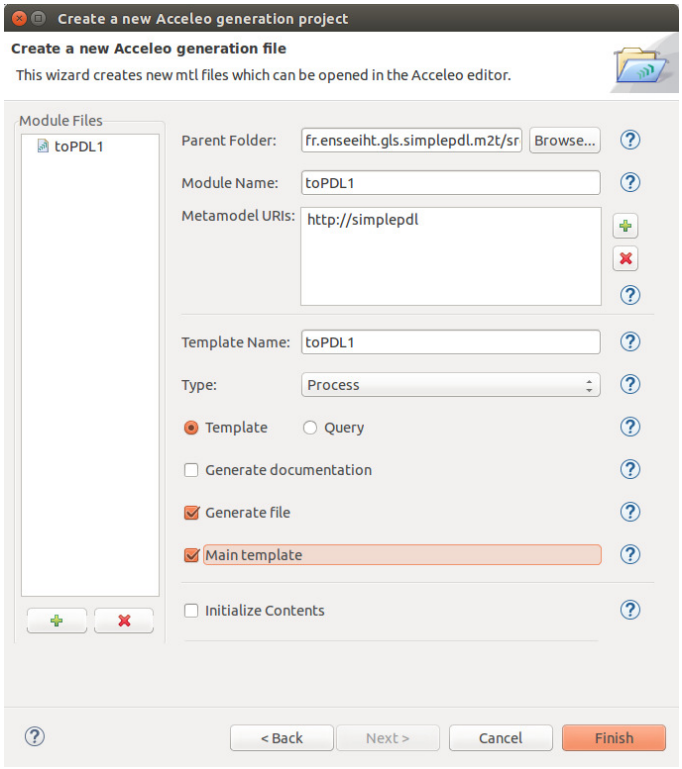



FIGURE 1 – Informations à fournir à l'assistant de création Acceleo

**2.2** Le projet contient un dossier de sources (*src*). Dans le paquetage fr.enseeiht.simplepdl.topdl1.main, un *template* de génération a été engendré (toPDL1.mtl). Ouvrir ce fichier.

**2.3** Remplacer le contenu du fichier *toPDL1.mtl* par celui du listing 1 (disponible sur le site du module).

**2.4** Créons un lanceur Acceleo pour lancer la transformation depuis un modèle SimplePDL.

**2.4.1** *Créer un lanceur pour Acceleo.*

1. Se placer dans la perspective Acceleo.
2. Sélectionner dans le "Package Explorer" le fichier "toPDL1.mtl" dans le greffon Acceleo "fr.enseeiht.simplepdl.topdl1" (répertoire src/fr.enseeiht.simplepdl.topdl1.main).
3. Créer un greffon "Acceleo UI Launcher Project" (menu File, entrée New). Celui-ci doit s'appeler "fr.enseeiht.simplepdl.m2t.topdl1.ui". Avancer avec "Next" deux fois et préciser la valeur "*.simplepdl" pour le champ "Model file name filter :".
4. Déployer les greffons "fr.enseeiht.simplepdl.topdl1" et "fr.enseeiht.simplepdl.topdl1.ui".

**2.4.2** *Tester le greffon M2T déployé.* Après le redémarrage d'Eclipse, sélectionner un modèle SimplePDL, faite un clic droit et sélectionner l'entrée "Acceleo Model to Text", puis l'entrée correspondant à la transformation. La transformation va créer un dossier `src-gen` lors de la première application qui contiendra les fichiers générés.

**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 developpement {
      Conception -> RedactionDoc [arrowhead=vee label=f2f]
      Conception -> RedactionDoc [arrowhead=vee label=s2s]
      Conception -> Developpement [arrowhead=vee label=f2s]
      Conception -> RedactionTests [arrowhead=vee label=s2s]
      Developpement -> RedactionTests [arrowhead=vee label=f2f]
}
```

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

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

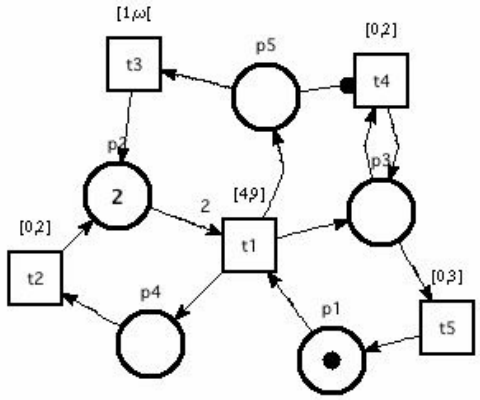
La documentation et des exemples concernant le langage *dot* peuvent être trouvés à l'URL : `http://www.graphviz.org/Documentation.php`

**3.1** Créer un projet `fr.enseeiht.petrinet.todot` et écrire puis tester un template Acceleo `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 2(a) illustre les principaux concepts présents des réseaux de Petri temporels que nous considérons. Le fichier textuel de ce réseau est donné au listing 2(b) au format Tina.




(a) syntaxe graphique

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

(b) syntaxe textuelle

FIGURE 2 – Exemple de réseau de Petri

**4.1** Créer un projet `fr.enseeiht.petrinet.totina` et écrire un template Acceleo `toTina.mtl` 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*. N'oubliez pas de créer le lanceur `fr.enseeiht.petrinet.totina.ui` et de déployer les projets.

**4.2** Créer un projet `fr.enseeiht.petrinet.todot` et écrire un template Acceleo `toDot.mtl` pour transformer un modèle de réseau de Petri en un fichier `.dot` qui permettra de visualiser graphiquement le réseau. N'oubliez pas de créer le lanceur `fr.enseeiht.petrinet.todot.ui` et de déployer les projets.

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