Sémantique statique d'un DSML avec OCL

Attention : Version d'Eclipse à utiliser : `/mnt/n7fs/ens/tp_cregut/eclipse-gls/eclipse`

1 Compléter un méta-modèle par des contraintes statiques

Exercice 1 : Vérification statique avec OCL
Nous avons utilisé Ecore (ou EMOF) pour définir un méta-modèle pour les processus. Il est rappelé à la figure 1. Certaines contraintes sur les modèles de processus ont pu être exprimées. C'est par exemple le cas de celles qui concernent les multiplicités. La propriété `opposite` permet aussi d'exprimer une contrainte entre deux références pour retrouver une notion proche de la relation d'association bidirectionnelle d'UML.

Cependant, le langage de méta-modélisation (Ecore ou EMOF) ne permet pas d'exprimer toutes les contraintes que doivent respecter les modèles de processus. Aussi, on complète la description structurelle du méta-modèle réalisée en Ecore (ou EMOF) par des contraintes exprimées en OCL.

Le méta-modèle Ecore et les contraintes OCL définissent la **syntaxe abstraite** du langage de modélisation considéré.

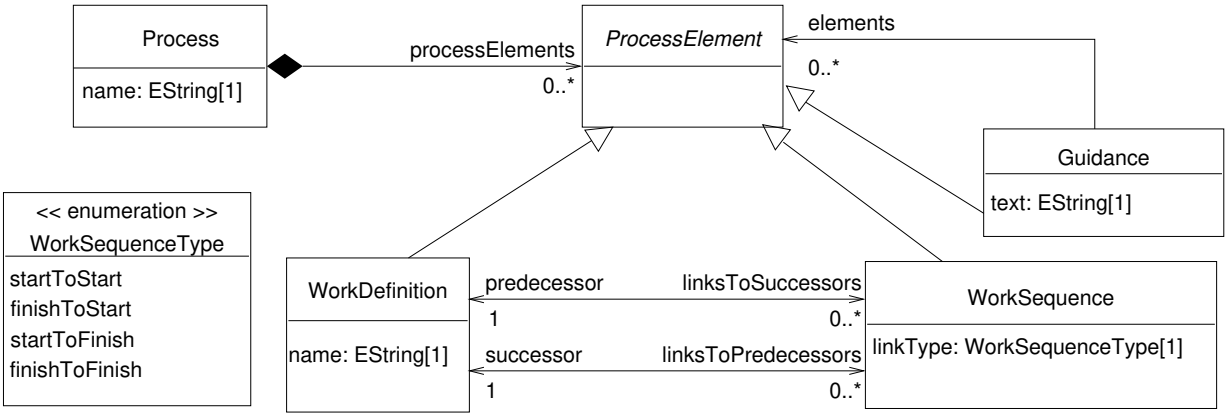

FIGURE 1 – Nouveau méta-modèle de SimplePDL

1.1. *OCL.* Expliquer les éléments apparaissant sur le fichier OCL du listing 1 correspondant au métamodèle de SimplePDL.
1.2. *Évaluation des contraintes OCL sur un modèle.* Vérifions maintenant notre modèle (process1-ko.xmi) par rapport à ce contraintes OCL.
1.2.1. Ajouter le fichier `process1-ko.xmi` au projet et le valider (clic droit sur l'élément racine Process, puis *Validate*). Seules les contraintes EMF sont vérifiées.

Listing 1 – Quelques contraintes OCL sur Le métamodèle SimplePDL

```
import 'SimplePDL.ecore'

package simplepdl

context Process
inv warningSeverity: false
inv withMessage('Explicit␣message␣in␣process␣' + self.name + '␣(withMessage)'): false
inv errorSeverity: null

context Process
inv validName('Invalid␣name:␣' + self.name):
   self.name.matches('[A-Za-z_][A-Za-z0-9_]*')

context ProcessElement
def: process(): Process =
   Process.allInstances()
      ->select(p | p.processElements->includes(self))
      ->asSequence()->first()

context WorkSequence
inv successorAndPredecessorInSameProcess('Activities␣not␣in␣the␣same␣process␣:␣'
      + self.predecessor.name + '␣in␣' + self.predecessor.process().name+ '␣and␣'
      + self.successor.name + '␣in␣' + self.successor.process().name):
   self.process() = self.successor.process()
   and self.process() = self.predecessor.process()

endpackage
```

1.2.2. Pour prendre en compte un fichier de contraintes OCL, il faut commencer par le charger. On fait un clic droit (par exemple sur la première ligne du modèle dans l'éditeur arborescent) et on fait *OCL > Load Document*. On peut alors choisir le fichier OCL qui est alors ajouté dans les ressources à la fin du modèle.

Valider le modèle (clic droit sur l'élément racine Process, puis *Validate*), constater que le modèle est invalide et comprendre l'origine des messages d'avertissements et d'erreurs affichés.

On constate que tous les avertissements ne sont pas affichés (une erreur stoppe l'évaluation des autres expressions du même contexte). Mettre en commentaire l'invariant `errorSeverity` dans le fichier OCL et valider à nouveau le modèle.

Remarque : Il est possible de charger un fichier OCL sur un métamodèle (par exemple SimplePDL.ecore). Les contraintes seront vérifiées sur tout modèle de ce métamodèle quand on en demandera la validation. Ceci peut être utile pour des propriétés intrinsèques d'un métamodèle alors que des fichiers OCL séparés pourront être appliqués suivant les modèles, la phase de développement, le contexte d'utilisation du modèle (par exemple, vérifier qu'un modèle de processus ne contient pas de dépendances de type *startToFinish*), etc.

1.3. *La console OCL.* La console OCL permet d'exécuter une expression OCL sur un élément d'un modèle. Ceci est pratique pour évaluer par morceaux une expression compliquée et ainsi la mettre au point.

Deux consoles existent, l'historique *OCL Console* et la nouvelle *Xtext OCL Console*. On préfèrera la seconde. Pour les obtenir, on fait un clic droit sur un élément du modèle (par exemple l'élément racine Process) et on choisit *OCL > Show Xtext OCL Console*.

Dans la partie inférieure de la console, on peut écrire une expression OCL (*self* correspond à l'élément sélectionné du modèle). Sa valeur s'affichera dans la partie supérieure. Écrire successivement les deux expressions suivantes :

```
self.name
self.processElements
```

1.4. *Nouveau fichier OCL.* Pour créer un nouveau fichier OCL, on peut sélectionner le métamodèle concerné dans l'explorateur, faire un clic droit puis *New > Others...*, choisir *Complete OCL File*, adapter le nom proposé et faire *Finish*. Le fichier est créé avec un exemple de contrainte.

Créer le fichier nouveau.ocl pour définir des contraintes sur SimplePDL.

1.5. *Compléter les contraintes de SimplePDL.* Exprimer les contraintes suivantes sur SimplePDL et les évaluer sur des exemples de modèles de processus :

1. deux activités différentes d'un même processus ne peuvent pas avoir le même nom.
2. une dépendance ne peut pas être réflexive.
3. le nom d'une activité doit être composé d'au moins deux caractères.
4. le nom d'une activité ne doit être composé que de lettres, chiffres ou soulignés, un chiffre ne peut pas être première position.

Exercice 2 : Contraintes sur les ressources (mini-projet)

Définir les contraintes appropriées concernant les ressources.

2 Application aux réseaux de Petri

Exercice 3 : Sémantique statique des réseaux de Petri
Définir les contraintes OCL définissant la sémantique statique du métamodèle des réseaux de Petri. Les valider sur différents modèles de réseaux de Petri.