Rapport de Mini-Projet

Génie du Logiciel et des Systèmes

Chaîne de vérification de modèles de processus

Groupe M-02
Fainsin Laurent
Guillotin Damien

Département Sciences du Numérique
Deuxième année
2021 — 2022

Table des matières

1 Métamodèles (avec Ecore)

1.1 simplePDL.ecore

Ce projet se base sur un langage simplifié de modélisation de processus de développement, appelé SimplePDL. Nous sommes donc parti du modèle Ecore de base du modèle SimplePDL auquel nous avons ajouté progressivement des nouveaux éléments. Dans un premier temps, il a fallu ajouter la modélisation des guidances comme indiqué dans le sujet.

Nous avons ensuite dû choisir une manière de modéliser les ressources nécessaires au processus de développement. Notre choix de modélisation s'est porté sur : une Ressource (qui implémente ProcessElement) est demandée par le biai d'une Request elle-même générée par une WorkDefinition. On se retrouve donc avec un nouveau modèle Ecore de la forme :

Les relations Ressource-Request et Request-WorkDefinition sont déclarées en EOpposite pour pouvoir facilement passer d'un fils à un parent et vice versa. Le modèle SimplePDL est maintenant complet pour représenter des processus de développement. Un exemple complet d'utilisation de ce modèle serait :

1.2 petriNet.ecore

En se basant sur ce que l'on a vu pour le modèle SimplePDL, nous avons créé un modèle Ecore permettant de modéliser les réseaux de pétri. Nous avons modélisé un réseau comme étant composé de nœuds. Ces nœuds peuvent être les places ou des transitions. Ils sont donc nommés et reliés entre eux par des arcs. Ses arcs ont un attribut entier nommé weight pour indiquer le poids de l'arc ainsi qu'un boolean outgoing pour indiquer si ce dernier est dirigé d'une Place vers une Transition ou d'une Transition vers une Place. (Si outgoing est vrai, alors l'arc va de la transition vers la place.)

2 Sémantique statique (avec OCL)

Les contraintes OCL sont là pour vérifier des informations du modèle vis-à-vis du métamodèle. Elles assurent certains points de cohérence et permettent d'éviter les ambiguïtés.

2.1 simplePDL.ocl

Pour les modèles SimplePDL, nous obligeons l'utilisateur à entrer des noms valides pour le Process, les WorkDefinition et les Resource. Les noms doivent aussi être unique pour les WorkDefinition et les Ressource pour améliorer la clarté du modèle. Nous avons aussi contraint l'utilisateur à utiliser les WorkSequence sur des WorkDefinition appartenant au même Process. Pour éviter des non-sens, les WorkSequence ne peuvent pas non plus avoir le même successeur et prédécesseur. Nous avons aussi ajouté des contraintes sur les quantités des Resource et Request. En effet, cela n'a pas de sens d'avoir des Resource ou des Request avec des quantités négatives. De plus, une Request ne peut pas être plus grande que le nombre initial de ressources. (Le nombre initial de ressources est le maximum puisqu'il n'y a pas de création.)

2.2 petriNet.ocl

Les modèles PetriNet étant relativement similaireq aux modèles SimplePDL, nous avons établi des contraintes OCL similaires. Nous obligeons le Network et les Node à avoir des noms uniques mais également sensés. Le nombre de jetons des Place et le poids des Arc doivent évidemment être positifs.

3 Eclipse Modeling Framework (EMF)

Pour permettre une meilleur intégration de nos métamodèles dans notre environnement de developpement (sous Eclipse), nous pouvons créer des greffons nous permetttant de les intégrer dans d'autres projets ainsi que des éditeurs arborescents nous permettant de mieux visualiser/éditer des modèles conformes à nos métamodèles Ecore. Le code java de ces éditeurs arborescent est engendré par nos métamodèles Ecore, mais nous pouvons le modifier manuellement pour que celui-ci convienne parfaitement à nos critères. Ces plugins seront déployés dans une Eclipse Application séparée de notre Application principale pour ne pas mélanger métamodèles et modèles.

3.1 plugin simplePDL

3.2 plugin petriNet

3.3 simplePDL → petriNet (avec Java)

4 Transformation de modèle à texte (avec Acceleo)

4.1 simplePDL → html

4.2 simplePDL → dot

4.3 petriNet → tina

5 Définition de syntaxes concrètes graphiques (avec Sirius)

5.1 Éditeur graphique simplePDL

5.2 Éditeur graphique petriNet

6 Définition de syntaxes concrètes textuelles (avec Xtext)

6.1 Éditeur textuel simplePDL

6.2 Éditeur textuel petriNet

7 Transformation de modèle à modèle (avec ATL)

7.1 simplePDL → petriNet