Rapport Mini-Projet

Chaîne de vérification de modèles de processus

Mohamed IKICH
Kamal HAMMI
Groupe B-01

22 October 2021

1 Introduction

Ce document représente un petit rapport concis expliquant le travail réalisé lors du mini projet de la matière Génie du Logiciel et des Systèmes.
L'objectif du mini projet est de produire une chaîne de vérification de modèles de processus SimplePDL afin de vérifier leur cohérence, et en particulier savoir si le processus décrit peut se terminer ou non.
les principales étapes pour répondre à cette questions sont cités dans la section qui suive.

2 Plan

1. Définition des métamodèles (SimplePDL et PetriNet) avec Ecore.
2. Définition de la sémantique statique avec OCL des deux métamodèles.
3. Définition de transformations modèle à texte (M2T) avec Acceleo, celles pour engendrer la syntaxe attendue par Tina à partir d'un modèle de réseau de Petri (PetriNet), ou engendrer les propriétés LTL à partir d'un modèle de processus (SimplePDL).
4. Définition d'une transformation de modèle à modèle (M2M) avec EMF/Java et avec ATL : SimplePDL2PetriNet.
5. Définition de syntaxes concrètes textuelles avec Xtext pour un modèle de processus (*.pdl).
6. Définition de syntaxes concrètes graphiques avec Sirius pour un modèle de processus (SimplePdl).

3 Les métamodèles SimplePDL et PetriNet

3.1 SimplePDL

Le métamodèle SimplePDL est donné par la figure 1. Il est obtenu en rendant les activités (WorkDefinition) et les liens entre eux (WorkSequences) comme des éléments de processus (ProcessElement) créés à partir de ce processus, en introduisant une Guidance comme une étiquette pour décrire ces éléments. L'ajout des Ressources est faite en créant une classe (Ressource) considérée comme (ProcessElement) et associée à chaque activité par un poids (Poids) qui définit le nombre d'occurence nécessaire d'une ressource pour que l'activité peut se lancer. Le fichier nommé SimplePDL.ecore contient ce métamodele. Le fichier SimplePDL.aird associé est dans le dossier livrables.

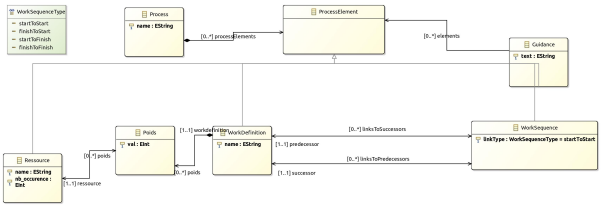

FIGURE 1 – Diagramm Class of SimplePDL

3.2 PetriNet

Le métamodèle PetriNet est donné par la figure 2. Il est obtenu en rendant les Places (Place) et les Transitions entre eux (Transition) comme des éléments du réseau Petri (PetriElement) créés à partir de ce Petri, en introduisant La classe EArc créée par le réseau petri (Petri) et qui a deux sous types Arc et ReadArc et un attribut Poids. Les EArc et PetriElement ont une référence au Petri associé qui s'appelle net. Le fichier nommé PetriNet.ecore contient ce métamodèle. Le fichier PetriNet.aird associé est dans le dossier livrables.

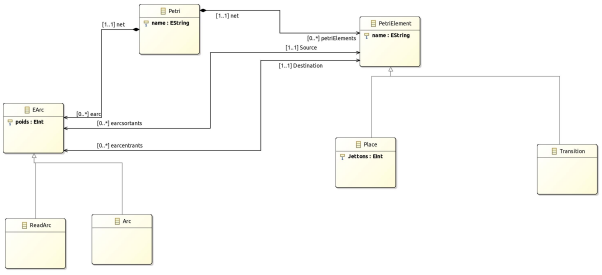

FIGURE 2 – Diagramm Class of PetriNet

4 Définition de la sémantique statique avec OCL des deux métamodèles.

4.1 Contraintes de SimplePDL

Les fichiers SimplePDL.ocl et SimplePDLOCL.ocl contienent les contraintes associées au métamodèle SimplePDL suivantes :

1. Le nom d'un processus ou d'une activité ne doit contenir que les alphabets miniscules ou majiscules avec des chiffres, et commence forcément par une lettre.
2. Le predecessor et le successor d'une WorkSequence doivent étre dans le mème processus et doivent étre différents.
3. Deux activités ou deux Ressources ne doivent pas avoir le mème nom Pour une activité, le lien vers les predecesseurs ne doit pas étre null
4. Pour une Ressource, la somme des valeurs de poids entre cette ressource et tous les activités associées doit étre inférieure au son nombre d'occurence.
5. Pour un Poids, la valeur doit étre inférieure au nombre d'occurenece de la Ressource associée.

4.2 Contraintes de PetriNet

Le fichier PetriNet.ocl contient les contraintes associées au métamodèle PetriNet suivantes :

1. Deux PetriElement ne doivent pas avoir le mème nom
2. Pour un Petri, la référence vers les petriElements ne doit pas étre null.
3. Un Arc peut étre entre une Place et une Transition, ou entre Transition et Place .
4. Un ReadArc doit etre entre une Place et une Transition.
5. Pour un EArc, le poids doit étre positif strictement

Le fichier net1.xmi contient un modèle qui respecte tous les contraintes.
Le fichier ko-net1.xmi contient un modèle qui ne respecte pas la contrainte : 3.
Le fichier ko-net2.xmi contient un mdèle qui ne respecte pas la contrainte : 1

5 Définition de transformations modèle à texte (M2T) avec Acceleo.

5.1 Engendrer la syntaxe attendue par Tina à partir d'un modèle de réseau de Petri (PetriNet)

Le fichier PetriNet2Tina.mtl contient la transormation du modèle de réseau de Petri vers la syntaxe Tina.

Le fichier pdl-sujet.net contient le résultat de la transformation à partir du modèle fourni en TP : Developpement.petrinet en ajoutant deux Ressources.
La figure 3 montre le graphe Tina associé .

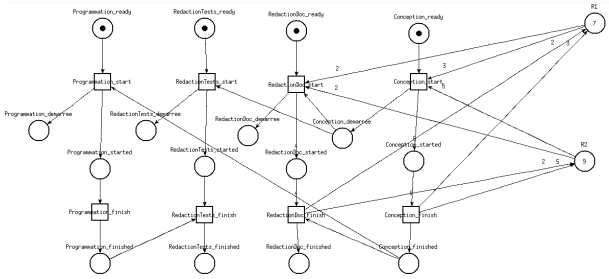

FIGURE 3 – Graph of Tina of Develeppement Process

5.2 Engendrer les propriétés LTL à partir d'un modèle de processus (SimplePDL)

Le fichier SimplePDL-finish.mtl contient la transformation vers la Syntaxe LTL à partir d'un modèle de processus en vérifiant les propriétées de terminaison de tous les activités du processus.
Le fichier SimplePDL-invariants.mtl transforme le modèle d'un processus aux propriétées LTL des invariants dans le processus :

1. Chaque activité est soit non commencée, soit en cours, soit terminée.
2. Une activité terminée n'évolue plus.

6 Définition d'une transformation de modèle à modèle (M2M) avec EMF/Java et avec ATL.

6.1 SimplePDL2PetriNet avec EMF/Java

Le fichier SimplePDL2PetriNet.java contient la transformation du modèle de Processus vers un réseau de Petri avec EMF/Java, qui prend deux arguments, le premier est le chemin du fichier xmi qui contient le modèle de Processus, et le deuxième est le chemin de fichier xmi destination pour enregistrer le modèle de réseau de Petri engendré.
La transformation d'une WorkDefinition est donnée par la figure 4.

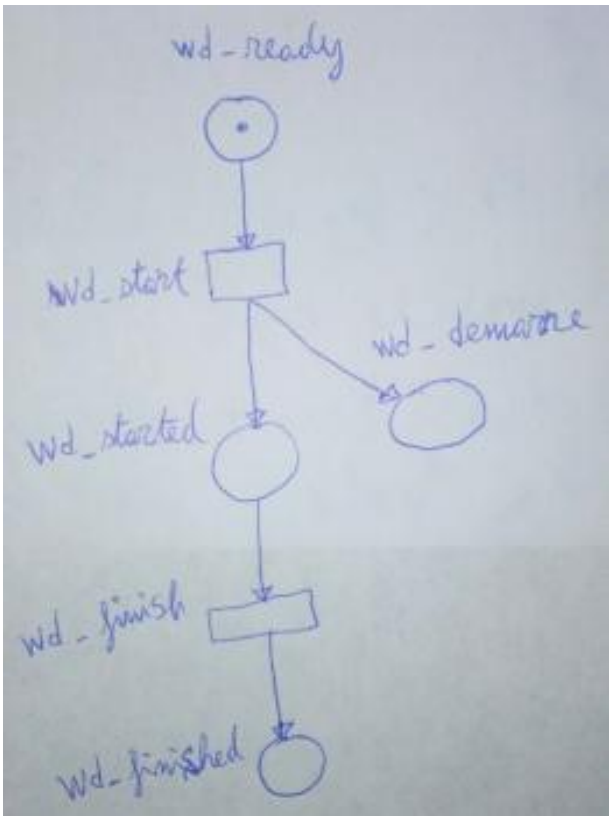

FIGURE 4 – Graph of petrinet of WorkDefinition

La transformation d'une WorkSequence entre deux activités est donnée par la figure 5.

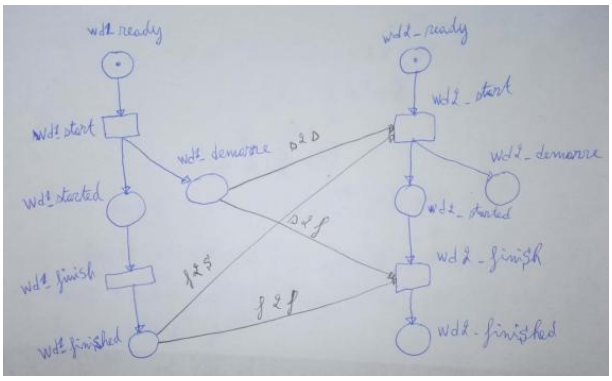

FIGURE 5 – Graph of petrinet of WorkSequence

Enfin la transformation d'une Ressource est donnée par la figure 6.

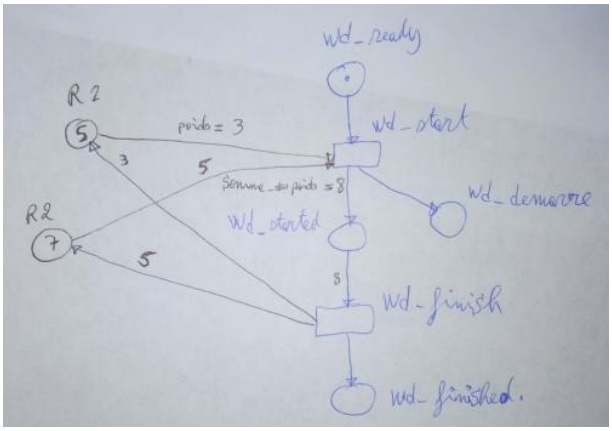

FIGURE 6 – Graph of petrinet of Ressource

6.2 SimplePDL2PetriNet avec ATL

Le fichier SimplePDL2PetriNet.atl contient la transformation du modèle de Processus vers un réseau de Petri avec ATL. Elle à le mème principe de celle de java et produit le mème résultat.

7 Définition de syntaxes concrètes textuelles avec Xtext pour un modèle de processus

Le fichier SimplePDL.xtext contient la définition de syntaxes concrètes textuelles pour un modèle de processus, en introduisant tous les éléments d'un Process, la Ressource et le Poids inclus.

8 Définition de syntaxes concrètes graphiques avec Sirius pour un modèle de processus.

Le fichier SimplePDL.odesign contient la définition de syntaxes concrètes graphiques pour un modèle de processus, avec la possibilité de créer les Noeuds ou Relations de chaque élément de Process, Ressource inclus.
La figure 7 montre le diagram engendré par Sirius du modèle de processus fournit en TP, avec deux Ressources.

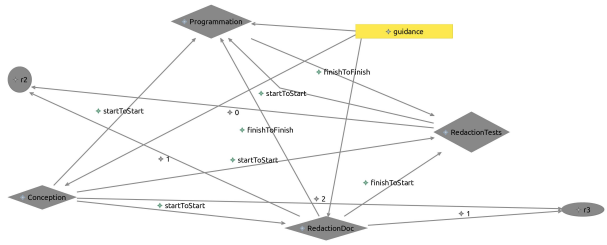

FIGURE 7 – Sirius Graph of Developpement Process