Validation de modèles de procédé

L'objectif de ce projet est de compléter le travail fait pendant les séances IDM et de produire une chaîne de vérification de modèles de processus SimplePDL en utilisant les outils de *model-checking* disponibles dans la boîte à outils *Tina*.

1 Partie I

Cette première partie consiste pour l'essentiel à compléter la chaîne de vérification de modèle de processus en intégrant deux petites extensions aux modèles de processus : le temps et les ressources.

1.1 Validation de la transformation SimplePDL2PetriNet

Comme pour tout programme écrit, il est important de valider la transformation de modèle. Afin de valider la transformation SimplePDL vers PetriNet, une possibilité est de vérifier que les invariants sur le modèle de processus sont préservés sur le modèle de réseau de Petri correspondant. Ces invariants sont appelés *propriétés de sûreté*. En voici quelques exemples :

- chaque activité est soit non commencée, soit en cours, soit terminée ;
- une activité terminée n'évolue plus.

On peut alors écrire une transformation modèle à texte qui traduit ces propriétés de sûreté sur le modèle de Petri. L'outil *selt* permettra alors de vérifier si elles sont effectivement satisfaites sur le modèle de réseau de Petri. Si ce n'est pas le cas, c'est que la traduction contient une erreur ou que l'invariant n'en est pas un !

1.2 Ajout des ressources et du temps

Pour réaliser une activité, des ressources peuvent être nécessaires. Une ressource peut correspondre à un acteur humain, un outil ou tout autre élément jouant un rôle dans le déroulement de l'activité. Ici, nous nous intéressons simplement aux types de ressources nécessaires et au nombre d'occurrences d'un type de ressource. Par exemple, il peut y avoir deux développeurs, trois machines, un bloc-note, etc. Un type de ressource sera seulement caractérisé par son nom et la quantité d'occurrences de celle-ci.

Pour pouvoir être réalisée, une activité peut nécessiter plusieurs ressources (éventuellement aucune). Par exemple, l'activité *RedactionTest* nécessite un testeur (une occurrence de la ressource de type Testeur) et deux machines (deux occurrences de la ressource Machine). Les occurrences de ressources nécessaires à la réalisation d'une activité sont prises au moment de son démarrage et rendues à la fin de son exécution. Bien entendu, une même occurrence de ressource ne peut pas être utilisée simultanément par plusieurs activités. Les types de ressource et leur nombre d'occurrences sont définis en même temps que le procédé lui-même.

D'autre part, la réalisation d'une activité nécessite du temps qui sera, dans notre cas, modélisé comme un intervalle [temps_min, temps_max] où temps_min (respectivement temps_max) est le temps minimal (respectivement maximal) pour le déroulement de l'activité.

De manière analogue, un processus est caractérisé par un temps minimal et un temps maximal. Toutes les activités d'un processus doivent pouvoir s'exécuter dans cet intervalle de temps.

La figure 1 présente un exemple de procédé. Les activités sont représentées par des ovales. Pour chaque activité apparaît son temps minimal et son temps maximal entre crochets. Les relations de précédence sont dessinées sous forme de flèches. C'est l'activité du côté de la flèche qui dépend de l'activité à l'autre extrémité. Le type de dépendance est noté sur le trait. Par exemple l'activité *RedactionTest* ne peut être commencée que quand l'activité *Conception* est commencée et ne peut se terminer que l'activité *Developpement* est terminée. Les ressources sont modélisées, en bas de la figure, par des rectangles en faisant apparaître le type de ressource et nombre total de ressources. Par exemple, il y a 3 concepteurs. Un trait entre une activité et une ressource indique le nombre d'occurrences de ce type de ressource requis par l'activité pour s'exécuter.

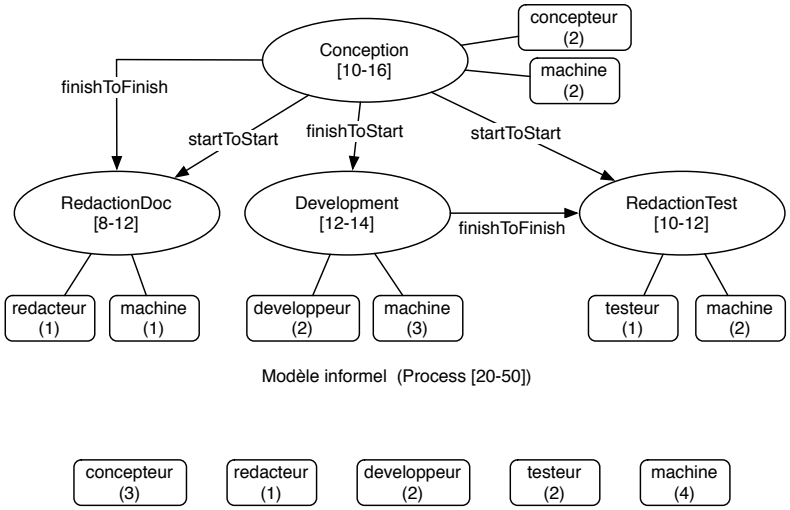

FIGURE 1 – Exemple de processus avec ressources et contraintes temporelles

2 Partie II

Dans cette deuxième partie, on propose trois extensions du modèle de processus qui peuvent être traitées de manière indépendante. Seulement deux des trois extensions proposées doivent être traitées. À vous de choisir.

2.1 Décomposition d'une activité

Nous envisageons maintenant la possibilité de décomposer une activité. À l'image d'un processus, une activité peut contenir d'autres activités que nous appellerons sous-activités. Bien

entendu, une sous-activité peut également être décomposée.

Une sous-activité ne peut utiliser que les ressources réservées par l'activité englobante.

Une activité composée est considérée :

– commencée quand ses ressources ont été allouées ;
– terminée quand toutes ses sous-activités sont terminées (et les ressources libérées).

2.2 Gestion plus fine des ressources

Pendant le déroulement d'un procédé, il est possible d'interrompre une activité. Ceci permet de libérer les ressources qu'elle utilisait. Elles peuvent alors être affectées à une autre activité. Pour pouvoir reprendre, l'activité devra retrouver les ressources nécessaires à son déroulement.

On considère que le temps de réalisation de l'activité suspendue continue à être décompté.

2.3 Ressources alternatives

Pour réaliser une activité, elle doit disposer d'un certain nombre de ressources. En fait, plusieurs configurations de ressources peuvent permettre de réaliser une activité. Par exemple, rédiger les tests peut nécessiter soit un testeur et deux machines, soit un analyste et un générateur de test et une machine. Si l'un de ces deux ensembles de ressources est disponible, il est utilisé par l'activité qui peut alors commencer.

3 Déroulement du projet

3.1 Réalisation

Ce projet sera réalisé en binôme. Les techniques mises en œuvre doivent être celles présentées dans le module de GLS.

3.2 Tâches à réaliser

Pour chaque partie, voici les tâches à réaliser et les documents à rendre.

T_1 Compléter le méta-modèle SimplePDL pour prendre en compte les ressources et le temps.

T_2 Adapter l'éditeur graphique SimplePDL pour permettre de saisir graphiquement les ressources disponibles et l'allocation des ressources nécessaires à chaque activité.

T_3 Donner une syntaxe concrète textuelle de SimplePDL pour prendre en compte les ressources et le temps et l'outiller avec xText.

T_4 Compléter les contraintes OCL pour capturer les contraintes qui n'ont pu l'être par le méta-modèle.

T_5 Compléter la transformation SimplePDL vers PetriNet.

T_6 Valider la transformation SimplePDL vers PetriNet.

T_7 Compléter la transformation PetriNet vers Tina.

T_8 Définir les contraintes OCL à associer au méta-modèle PetriNet.

T_9 Engendrer les propriétés LTL permettant de vérifier la terminaison d'un processus et les appliquer sur différents modèles de processus.

T_{10} Engendrer les propriétés LTL correspondant aux invariants de SimplePDL pour valider la transformation écrite (voir section 1.1).

3.3 Documents à rendre

D_1 Les méta-modèles SimplePDL.ecore et PetriNet.ecore.

D_2 Les fichiers de contraintes OCL associés à ces méta-modèles.

D_3 Le code ATL des transformations modèle à modèle.

D_4 Le code ATL ou Acceleo des transformations modèle à texte.

D_5 Les modèles GMF décrivant l'éditeur graphique pour SimplePDL.

D_6 Le modèle xText décrivant la syntaxe concrète textuelle de SimplePDL.

D_7 Des exemples de modèles de processus (en expliquant leur utilité).

D_8 Un document concis (rapport) qui explique le travail réalisé.

3.4 Évaluation

Le travail à faire est découpé en deux parties décrites ci-dessus.

Dans la deuxième partie, il n'est pas demandé d'adapter l'éditeur graphique. Les extensions demandées peuvent être traitées de manière indépendantes.

L'ajout des ressources sera évaluée lors du dernier TP de 2012. L'approche choisie pour prendre en compte le temps devra être expliquée (mais il n'est pas demandé à ce qu'elle soit implantée à cette date).

Il faudra alors faire une démonstration sur machine du travail réalisé et rendre éventuellement un rapport concis et précis sur ce qui a été réalisé. Le rapport doit expliquer l'approche suivie, les difficultés rencontrées, les choix faits et les solutions retenues.

La deuxième partie sera évaluée avant la fin du semestre. Une présentation et une démonstration seront également réalisées (voir l'EDT). Un rapport décrivant l'ensemble du travail réalisé devra être rendu (format électronique).

Lors de chaque évaluation, le rapport au format PDF et une archive contenant les documents demandés devront être envoyés par courrier électronique à l'enseignant de TP au plus tard le vendredi 25 janvier.

Le sujet du message doit être :

```
[N7 GLS] Livrables projet pour Evaluation 1
```

(ou `Evaluation 2` pour la deuxième évaluation, bien sûr).

La note finale tiendra compte des deux partie, des démonstration réalisée, des explications données lors de la démonstration, du rapport écrit et des fichiers rendus.