



Ingénierie Dirigée par les Modèles

Mini-Projet IDM

Groupe L34-02

Élèves :

THEVENET Louis

SABLAYROLLES Guillaume

Octobre 2024

Contents

1. Méta-Modèles	3
1.1. SimplePDL	3
1.2. Exemples	3
1.3. PetriNet	3
2. Transformation Modèle à Modèle	5
2.1. SimplePDL vers PetriNet	5
3. Transformation Modèle à Texte	6
4. Transformation Texte à Modèle	6

1. Méta-Modèles

1.1. SimplePDL

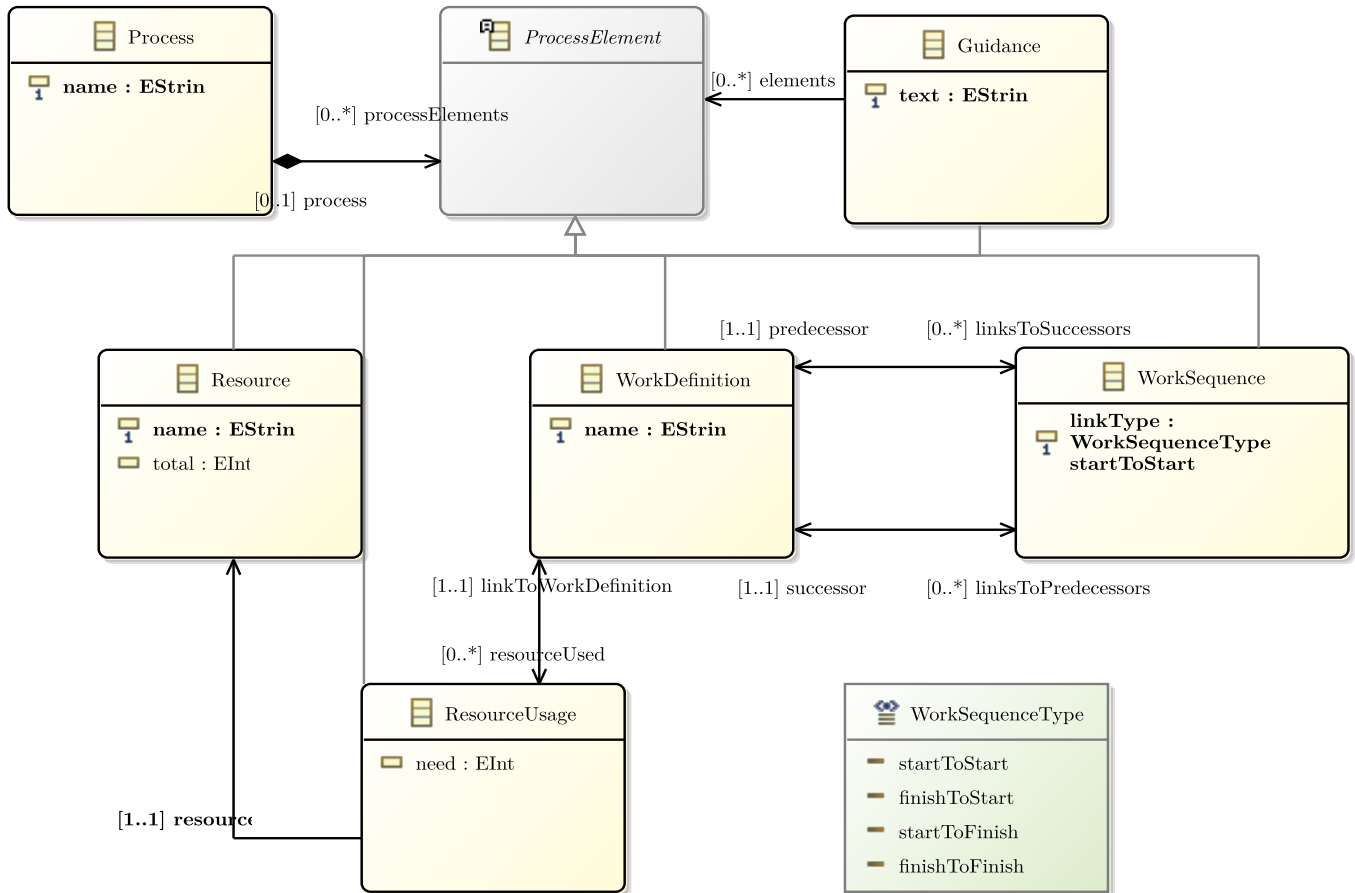


Figure 1: Méta-Modèle de SimplePDL

A partir du méta-modèle fourni, nous avons rajouté l'utilisation des ressources. Une ressource est défini comme une EClass **Resource** ayant :

- un nom
- et un nombre total d'éléments disponible (**total**)

Une **WorkDefinition** utilise une **Resource** en ajoutant une référence à une **ResourceUsage** qui contient :

- référence à la **Resource** en question
- la quantité demandée (**need**)

1.2. Exemples

Ici mettre des exemples

1.3. PetriNet

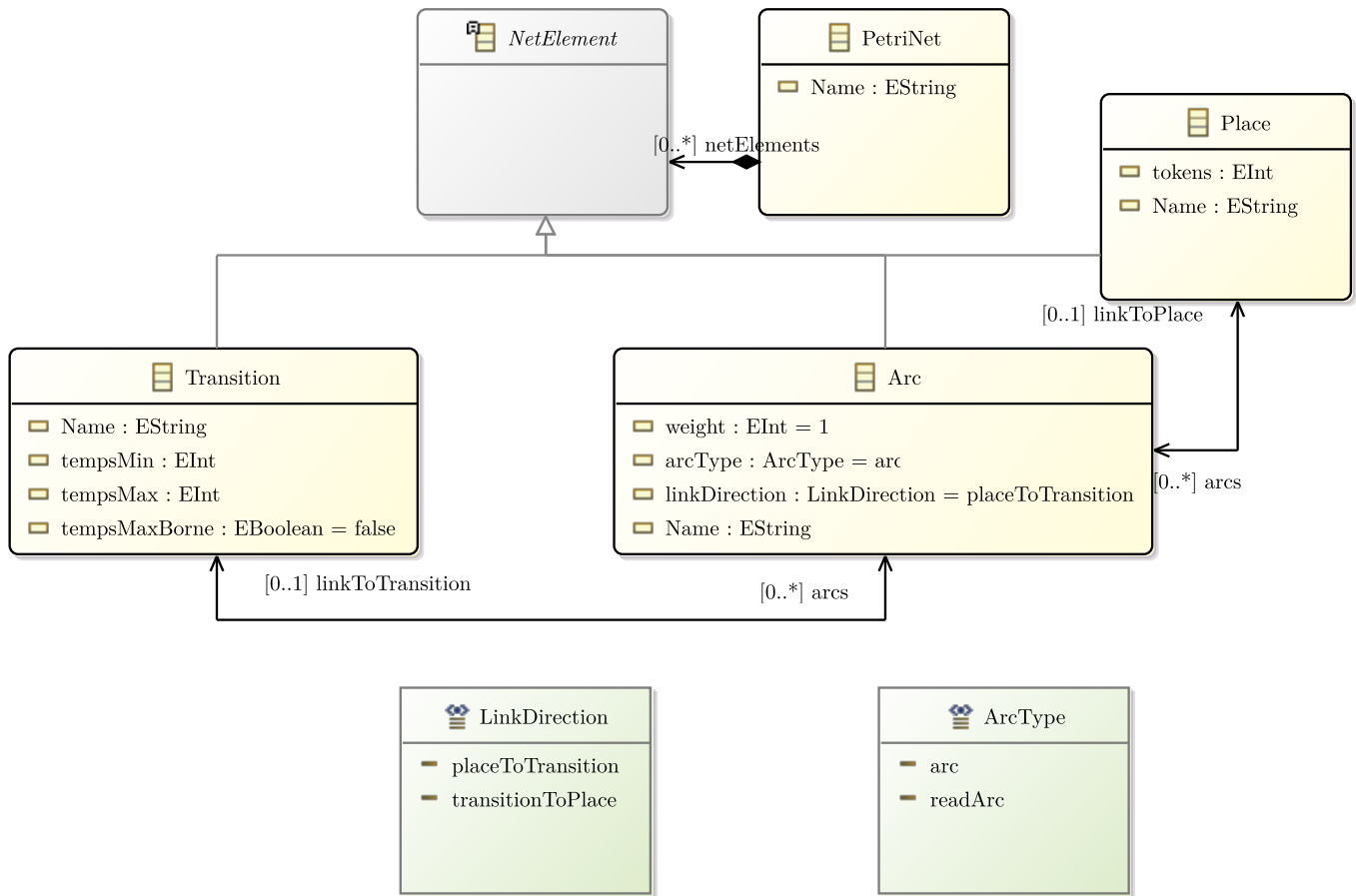


Figure 2: Méta-Modèle de PetriNet

Un **PetriNet** est constitué de **NetElement**. Ces éléments sont les **EClass** **Place**, **Transition** et **Arc**.

Une **Place** est définie par un nom et un nombre de jetons (**tokens**).

Une **Transition** est définie par un nom.

Un **Arc** contient :

- un nom
- un poids
- une référence vers une **Place**
- une référence vers une **Transition**
- Une **LinkDirection** (soit **placeToTransition**, soit **transitionToPlace**)
- Un **arcType** (soit **arc**, soit **readArc**).

Nous avons également ajouté des attributs **tempsMin**, **tempsMax** et **tempsMaxBorne** pour ajouter la notion de temps aux **Transition**.

Ce méta-modèle permet de s'assurer que les **Arc** ne relient jamais deux **Transition** ou deux **Place**.

2. Transformation Modèle à Modèle

2.1. SimplePDL vers PetriNet

2.1.1. Java/EMF

Principe de la transformation d'un modèle de processus en réseau de Petri:

- Un élément **Process** devient un élément **PetriNet**
- Une **WorkDefinition** devient 4 places **ready** (avec 1 jeton), **started**, **running** et **finished** et deux transitions **start** et **finish**
- Une **WorkSequence** devient un read-arc entre une place de l'activité précédente (**started** ou **finished**) et une transition de l'activité cible (**start** ou **finish**)

Transformation des ressources :

- Une **Resource** devient une place dont le nombre de jetons est égal au nombre de ressources initialement disponibles
- Une **ResourceUsage** devient deux arcs avec pour poids le nombre de ressources demandé :
 - De la place représentant la **Resource** utilisée à la transition **start** de la **WorkDefinition**
 - De la transition **finish** de la **WorkDefinition** à la place représentant la **Resource** utilisée

Lors de la transformations, on traite les **ProcessElement** dans cet ordre :

1. **Process**
2. **Resource**
3. **WorkDefinition**
 - **ResourceUsage** (on traite les **ResourceUsage** attachées à la **WorkDefinition** courante)
4. **WorkSequence**

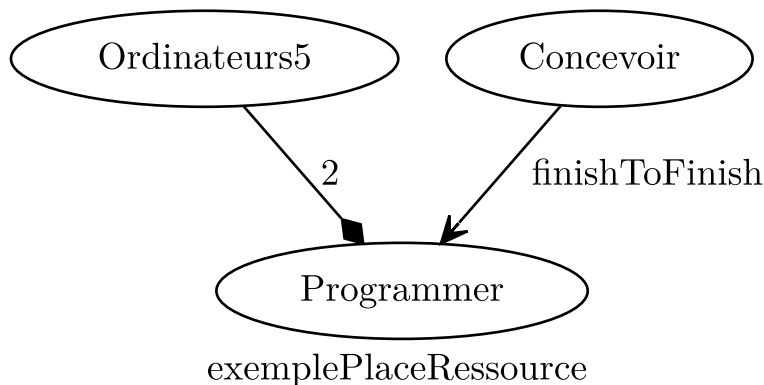


Figure 3: Entrée : modèle de processus simple avec une ressource

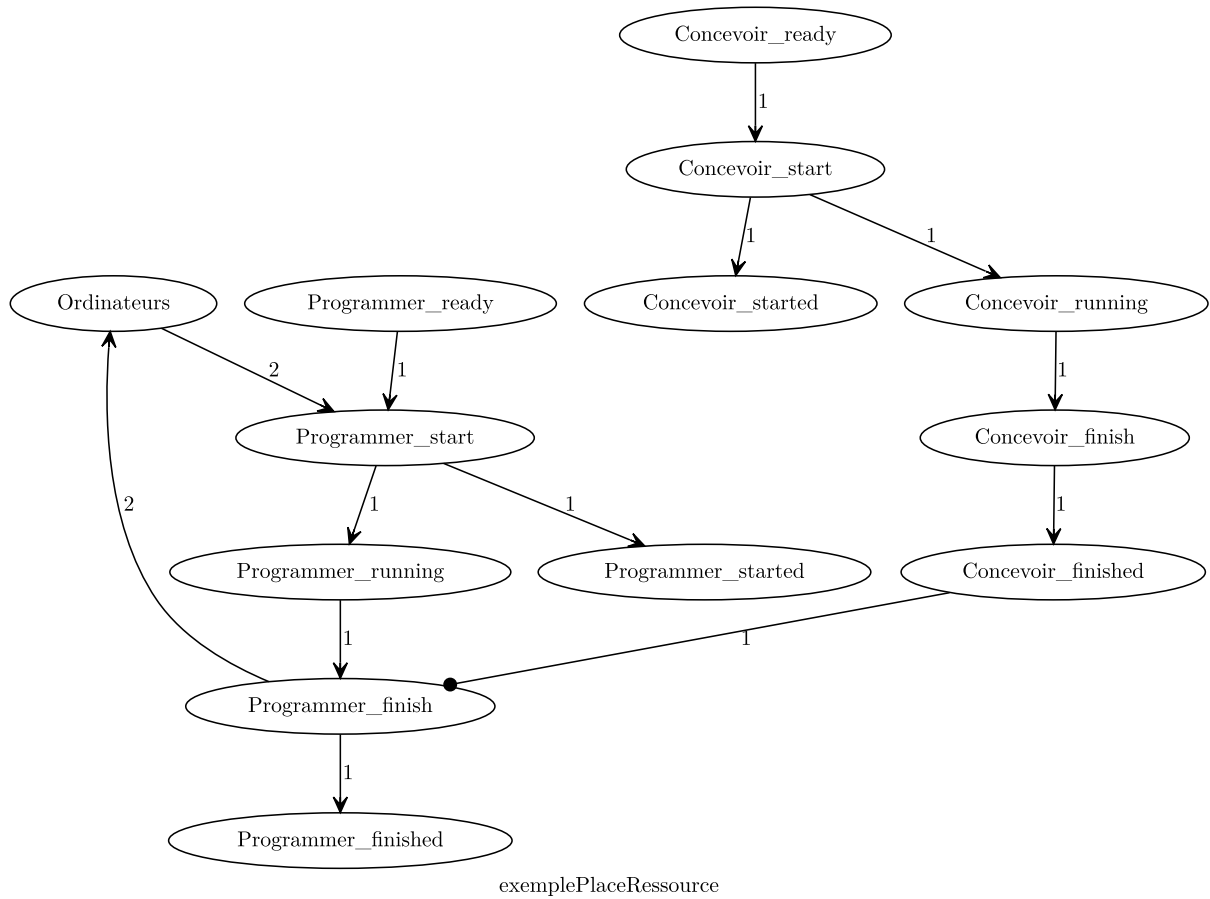


Figure 4: Sortie : réseau de Petri résultant (les read-arc sont affichés à l'envers)

On distingue aisément les différents sous-réseaux de Petri associés aux **WorkDefinition** ainsi que la Ressource et les arcs qui la relie au sous-réseau associé à *Programmer*

2.1.2. ATL

3. Transformation Modèle à Texte

4. Transformation Texte à Modèle