**République Algérienne Démocratique et Populaire**
**Ministère de l'enseignement Supérieur et de la Recherche Scientifique**
**Université de Mohamed El Bachir El Ibrahim de Bordj Bou Arreridj**
**Faculté des Mathématiques et d'Informatique**
**Département d'informatique**


**MEMOIRE**

Présenté en vue de l'obtention du diplôme

**Master en informatique**

Spécialité : Ingénierie d'informatique décisionnelle

**THEME**

# Génération automatique des modèles de réseaux de pétri associes au comportement des system de production en utilisant la transformation de graphe

**Présenté par :**

Aymen ben toumi.
Zakaria silem.
Soutenu publiquement le :

**Devant le jury composé :**

| | | | |
|---|---|---|---|
| ..... | ..... | Université de BBA | Président |
| ..... | ..... | Université de BBA | Examinateur |
| ..... | ..... | Université de BBA | Examinateur |
| ..... | ..... | Université de BBA | Encadreur |

**Promotion : 2021/2022**

# *REMERCIEMENT*

*Nous remercions d'abord et avant tout* ***Allah*** *qui nous adonné le courage et la patience pour réaliser ce travail.*

*Un remerciement particulier à notre encadreur BENDIAF MESSAOUD pour saprésence, son aide et surtout pour ses précieux conseils qui nous ontassistés pour l'accomplissement de notre projet,*

*Nous adressons également nos vifs remerciements aux membres du jury qui ont bien voulu et accepté d'examiner ce modeste travail,*

*Á mes très chers frères, soeurs, ami (es) et à toutes les personnes qui ont participé de près ou de loin à la réalisation de ce projet.*

*Enfin, je tiens à remercier toute la promotion 2020-2021 Master informatique. Ainsi que tous mes enseignants et les membres du département informatique d'Université Bordj Bou-Arreridj*

***AYMEN et ZAKARIA***

# *Dédicace*

*À ma mère*

À mes frères et mes soeurs

*À toute ma famille*

*À tous mes amis et collègues*

À tous ceux qui m'aiment et que j’aime

## Résume

L'ingénierie basée sur les modèles joue un rôle très important dans le développement logiciel où la transformation de modèle consiste à transformer le modèle source en modèle cible selon les méta-modèles source et cible, en corrigeant plusieurs problèmes (réutilisation, interopérabilité, migration de modèle). La représentation graphique joue un rôle très important dans le développement logiciel, ce qui nous ramène à l'intégration des graphes dans les transformations de modèles et les transformations de graphes. Pour cela, il est important de proposer des solutions qui intègrent les graphes au processus de transformation. L'idée de base de notre travail est de créer un pont technologique entre l'environnement IDM et l'environnement de grammaire Graph et d'effectuer une transformation bidirectionnelle à l'aide de l'outil d'interprétation TGG.

Dans ce mémoire, nous avons présente une transformation de system de production vers les réseaux de pétri (RdP) nous nous basant sur la transformation de graphes. Notre approche consiste a Proposer une grammaire de graphes qui contient un ensemble de règles pour la transformation entre deux formalismes diffèrent. Nous avons réalisé ce travail a l'aide de l'outil éclipse, par la définition de les deux métas modèles (L'un pour le system de production et l'autre pour les réseaux de pétri) et la grammaire de graphes.

Finalement nous terminons notre contribution par une étude de cas bien illustrée, et nous représentons les résultats qui agrémentent notre approche.

**Mots Clés :** Ingénierie dirigée par les modèles, modèles, méta modèles, méta modèles de correspondance, transformations de modèles, moteurs de transformation, grammaires de graphes, transformations de graphes, formalisme TGG, interpréteur TGG.

## Abstract

Model-based engineering plays a very important role in software development where model transformation consists of transforming the source model into the target model according to the source and target meta-models, fixing several problems (reuse, interoperability, migration of model). Graph representation plays a very important role in software development, which brings us back to the integration of graphs in model transformations and graph transformations. For this, it is important to offer solutions that integrate graphs into the transformation process. The basic idea of our work is to create a technological bridge between the IDM environment and the Graph grammar environment and to perform a two-way transformation using the TGG interpretation tool.

In this thesis, we have presented a transformation of a production system towards Petri nets (PnR) based on the transformation of graphs. Our approach consists in proposing a graph grammar which contains a set of rules for the transformation between two different formalisms. We carried out this work using the eclipse tool, by defining the two meta models (one for the production system and the other for the Petri nets) and the graph grammar.

Finally, we end our contribution with a well-illustrated case study, and we represent the results that complement our approach.

**Keywords:** Model-driven engineering, models, metamodels, correspondence metamodels, model transformations, transformation engines, graph grammars, graph transformations, TGG formalism, TGG interpreter.

## الملخص

تلعب الهندسة القائمة على النموذج دورًا مهمًا للغاية في تطوير البرامج حيث يتكون تحويل النموذج من تحويل النموذج المصدر إلى النموذج المستهدف وفقًا للنماذج الوصفية المصدر والهدف ، وإصلاح العديد من المشكلات (إعادة الاستخدام ، وإمكانية التشغيل البيني ، وترحيل النموذج). يلعب تمثيل الرسم البياني دورًا مهمًا للغاية في تطوير البرامج ، مما يعيدنا إلى تكامل الرسوم البيانية في تحويلات النموذج وتحولات الرسم البياني. لهذا ، من المهم تقديم حلول تدمج الرسوم البيانية في عملية التحول. الفكرة الأساسية لعملنا هي إنشاء جسر تكنولوجي بين بيئة IDM وبيئة قواعد الرسم البياني وإجراء تحويل ثنائي الاتجاه باستخدام أداة تفسير TGG.

في هذه الأطروحة ، قدمنا تحولًا في نظام الإنتاج نحو شبكات بتري (PnR) بناءً على تحويل الرسوم البيانية. يتمثل نهجنا في اقتراح قواعد نحوية للرسم البياني تحتوي على مجموعة من القواعد للتحول بين شكليتين مختلفتين. نفذنا هذا العمل باستخدام أداة الكسوف ، من خلال تحديد نموذجين ميتا (أحدهما لنظام الإنتاج والآخر لشبكات بتري) وقواعد الرسم البياني.

أخيرًا ، ننهي مساهمتنا بدراسة حالة مصورة جيدًا ، ونمثل النتائج التي تكمل نهجنا.

**الكلمات المفتاحية:** الهندسة المعتمدة على النموذج ، النماذج ، النماذج المعدنية ، نماذج المطابقة ، تحويلات النماذج ، محركات التحويل ، قواعد الرسم البياني ، تحويلات الرسم البياني ، شكليات TGG ، مترجم TGG.

## Liste des abréviations

**AGG :** Attribue Graph Grammaire.

**DPO :** Double PushOut.

**AToM3 :** A Tool for Multi-formalism and Meta-Modelling

**EMF :** Eclipse Modeling Framework .

**EMF Tiger :** Transformation generation.

**IDM :** Ingénierie Dirigée par les Modèles.

**MDA :** Model Driven Architecture.

**MDE :** Model-Driven Engineering.

**MOF :** Meta Object Facility.

**GMF :** Graphical Modeling Framework.

**OMG :** Object Management Group.

**OCL :** Object Constraint Language.

**TG :** Triple Graph.

**TGG :** Triple Graph Gramma.

**VIATRA :** VIsual Automated model TRAnsformations.

**XMI :** XML Metadata Interchange.

**XML :** eXtended Markup Language.

**TCL :** Langage de Contrôle de Transaction.

**PSM** : Platform Specific Model.

**QVT :** Query/View/Transformation.

**OMG :** Object Management Group.

**PDM :** Platform Description Model**.**

**PIM :** Platform Independent Model.

**M2M :** Model to Model.

**M2T :** Model to Text.

**Rdp :** resoux de pétri .

## Table des matières

## Table des figures

## Liste des tableaux

# CHAPITRE 1

# Introduction générale.

## 1.1 Contexte

Aujourd'hui, les systèmes informatique que ce soit simples ou complexes sont de plus en plus utilisés au niveau de différents domaines tel que (la médecine , léavionique, le domaine militaire l'architecteur, etc . . . .). La modélisation des systèmes estune phase laborieuse pour la conception et la validation des systèmes qui est l'objet de notre mémoire de fin d'étude .

Depuis plus d'une décennie, en termes d'interopérabilité, d'extensibilité et de réutilisabilité, l'ingénierie pilotée par les modèles (MDE) a fait du code non plus le résultat final d'un projet, mais son aspect simple. Le projet se concentre maintenant sur un élément essentiel : "le modèle". Ceci est spécifié sous une forme modélisée, mais indépendante de toute plate-forme, et dans une abstraction de haut niveau où nous nous concentrons sur les concepts plutôt que sur les implémentations. Cela nous permet de conserver le modèle, de transférer le modèle, puis de le transformer pour obtenir le code. Par exemple, lors d'un changement de plate-forme, nous essayons de générer ce code par l'amélioration continue du modèle de la manière la plus automatisée possible. On parle de passer de l'étape de "contemplation" à "productif «stade auquel le modèle est, très souvent disponible sous forme de code.

Dans le domaine du génie logiciel, il existe deux approches pour représenter les modèles des systèmes. La première approche et traditionnelle est l'approche informelle. Elle utilise une notation graphique et commune pour la représentation. Elle est plus facile à comprendre et à communiquer. La deuxième et importante approche pour spécifier les modèles des systèmes est l'approche formelle. Elle repose sur de solides notations et de preuves mathématiques.

## 1.2 Problématique

Dans ce travail, nous proposons une approche de transformation des system de production vers les réseaux de Pétri imbriqués. L'approche que nous définissons vise à proposer :

1. Un modélisation des system de production et des réseaux de Pétri.
2. Une grammaire de graphe pour appliquer la transformation des system de production vers des réseaux de Pétri. La construction d'un outil de modélisation à partir de zéro est une tâche prohibitive. Les approches de méta-modélisations sont utiles pour faire face à ce problème, car elles permettent la modélisation des formalismes eux-mêmes. Un modèle de formalisme qui contient suffisamment d'informations permet la génération automatique d'un system de production pour construire des modèles conforme à la syntaxe du formalisme décrit. Pour cela, nous utilisons éclipse.

## 1.3 Contribution

Pour résoudre ces problèmes, l'idée générale est de transformer le System de production vers des méthodes formelles. Les méthodes formelles offrent un cadre mathématique au processus de développement .Parmi le grand nombre de techniques formelles qui ont déjà été proposées on a les Réseaux de Pétri(RdP) .Ce dernier offre un outil formel et une bonne représentation graphique qui permet de modéliser les systèmes discrets, la facette graphique

des réseaux de pétri, nous aide à comprendre aisément le système modélisé. Par ailleurs ,leur puissance d'expression mathématique permet de simuler des activités dynamiques.

La transformation de System de production vers des RdP nécessite une connaissance sur Modèle Drivent Architecture (MDA), MDA est le domaine qui met à disposition des outils , concepts et langage pour créer et transformer des modèles afin de mécaniser le processus que les ingénieurs suivent habituellement à la main. Le MDA se concentre sur une préoccupation plus abstrait que la programmation classique ce qui permet d'obtenir plusieurs améliorations dans le développement de système complexe grâce à l'utilisation importante des modèles et des transformations automatique entre les modèles.

Notre travail se situe donc dans le contexte de la transformation des modèles .Plus précisément, il s'agit de transformation de System de production vers les réseaux de pétri, notre objectif est alors de proposer, une approche basée sur la transformation
de graphes pour générer un réseau de pétri à de System de production d'une manière automatique. Pour atteindre cet objectif nous définissons des méta-modèles pour les modèles d'entrée, sortie et une grammaire de graphes .Ces derniers sont implémenté avec l'outil éclipse.

## 1.4 Organisation de mémoire

Nous avons organisé notre mémoire comme la suite :

- Dans Le **premier chapitre** présente les notions de base de l'ingénierie dirigée par les modèles en présentant les deux axes principaux de l'IDM qui sont la méta modélisation et la transformation de modèles et présente plusieurs outils de transformation de graphes en mettant l'accent sur une étude comparative de ces outils selondifférents critères.
- Dans Le **deuxième chapitre** regroupe une description générales des systèmes de production ; et l'importance de la modélisation d'un système de production.
- Dans le **troisième chapitre** la modélisation par Réseaux de Petri (RdP) est rappelée, les propriétés des RdP sont ensuite discutées. Les extensions des RdP sont mises en perspective. Le chapitre se termine par une discussion succincte sur la modélisation des systèmes adaptatifs et mobiles par les RdP.
- Dans le **quatrième chapitre** on a présenté notre proposition d'une grammaire de graphe pour transformer les system de production vers les réseaux de Petri Nets en utilisant l'outil éclipse. Dans ce chapitre on a commençais par une proposition d'un méta modèle de system de production, suivie par une autre proposition d'un méta modèle pour le formalisme de RdP Nets, et enfin nous avons proposés une grammaire de graphe pour transformer le formalisme source vers le formalisme cible.

Et nous concluons ce mémoire par des perspectives qui permettront de donner unesuite à cette recherche.

# CHAPITRE 2
# Ingénierie Dirigée par les Modèles (IDM)

## 2.1 Introduction

Au sens large, la modélisation est en effet l'utilisation efficace de représentations simplifiées d'aspects réalistes d'un but donné. Loin de se réduire à une expression de solution à un niveau d'abstraction supérieur à celui du code, la modélisation informatique permet de prendre en compte des exigences fonctionnelles différentes et des préoccupations fonctionnelles supplémentaires (e.g. sécurité, fiabilité, efficacité, performance, ponctualité, flexibilité, etc.) à partir de la demande. Les méthodes d'ingénierie dirigée par les modèles (IDM) mécanisent simplement un processus effectué à la main par des ingénieurs expérimentés. À la fin du XXe siècle, l'organisme de normalisation OMG (Object Modeling Group) a dévoilé son initiative MDA (Model Driven Architecture), et l'intérêt d'IDM pour l'IDM s'est considérablement accru, ce qui peut être considéré comme une limitation de la gestion de l'IDM. Un aspect spécifique de la dépendance d'un logiciel vis-à-vis de la plate-forme d'exécution. Dans le cadre du MDE, le concept de transformations de modèles joue un rôle fondamental ; le processus de conception peut alors être vu comme un ensemble de transformations de modèles partiellement ordonnées, chacune prenant un modèle en entrée et produisant un modèle en sortie, jusqu'à une pièce exécutable. Dans ce chapitre, Nous allons introduire le concept de transformation de modèle en commençant par la représentation du concept de transformation de modèle dans le cadre général, puis énumérer les différents types de transformations, puis classer les différentes méthodes, et enfin nous allons introduire un cadre spécifique de transformation de modèle basé sur les transformations de graphes .

## 2.2 Définition Ingénierie Dirigée par les Modèles (IDM)

L'ingénierie dirigée par les modèles est une recherche qui utilise Modèles de conception d'artefacts (créer, produire, modifier ou Une personne utilise lors du développement d'un "système" informatique). elle Plaider pour changer l'utilisation des modèles de simples définitions à Articuler un aspect du design comme une véritable ressource opérationnelle Fabrication d'artefacts. Utilisez ensuite le modèle comme données d'entrée Transformez l'algorithme pour produire tout ou partie de l'artefact souhaité. Une approche de la conception technique basée sur des modèles repose sur Utilisation de métamodèles : définitions formelles interprétables par logiciel Langage de modélisation de modélisation.

La validité des modèles ainsi conçus peut être contrôlée auprès du méta-modèle. Un algorithme de transformation peut ensuite être conçu à partir du méta-modèle afin de transformer n'importe quel modèle en artefact final.Par son principe de modélisation et de transformation, l'ingénierie dirigée par les modèles permet de concevoir des artefacts en abstrayant les éléments simplifiés par le modèle [1].

## 2.3 Les Concepts de base de l’ingénierie dirigée par les modèles

L’ingénierie dirigée par les modèles a permis de prendre en charge la croissance de la complexité des systèmes logiciels développés ou la modélisation de ces systèmes se base sur l’usage intensif de modèles. De cette manière le développement est réalisé avec un niveau d’abstraction plus élevé que celui de la programmation classique. Cette approche permet alors

d'automatiser, ou au moins de dissocier et de reporter, la part du développement qui est proprement technique et dédiée à une plateforme d'implémentation [2].

L'ingénierie dirigée par les modèles (IDM) est donc une approche du génie logiciel sur laquelle le modèle est considéré comme une première présentation du système à modéliser, et qui vise à développer, maintenir et faire évoluer le logiciel en réalisant des transformations de ce modèle. Au sens large, le paradigme de l'IDM propose d'unifier tous les aspects du processus de cycle de vie en utilisant les notions de modèle et de transformation.

Dans ce qui suit, nous présentons des définitions pour bien comprendre les concepts de base de l'ingénierie des modèles et les relations existantes entre ces concepts.

- **Système :** Dans le cadre de l'ingénierie dirigée par les modèles, on parle souvent de systèmes, quelle que soit leur nature (existante ou à implémenter). Un système est défini comme un ensemble d'entités organisées qui coopèrent de manière unique et en interaction permanente pour assurer une ou plusieurs fonctions dans le système, nous nous intéressons principalement aux logiciels. [3].
- **Modèle :** Un modèle est une description abstraite d'un système construit dans un but donné. Il devrait donc fonctionner pour répondre aux questions sur le système. Un modèle doit généralement respecter les contraintes décrites par le méta modèle [3].
- **Méta modèle :** Un méta-modèle est un modèle d'un langage de modélisation. Le terme "méta" signifie transcendant ou au-dessus. Un méta-modèle décrit un langage de modélisation a un niveau d'abstraction supérieur que le langage de modélisation lui-même [4].
- **Méta-Méta-Modèle :** est un modèle de méta-modèle.

La figure suivante permet l'avoir du modèle qui doit être conforme à un méta-modèle. La notion de conformité est très importante [3].

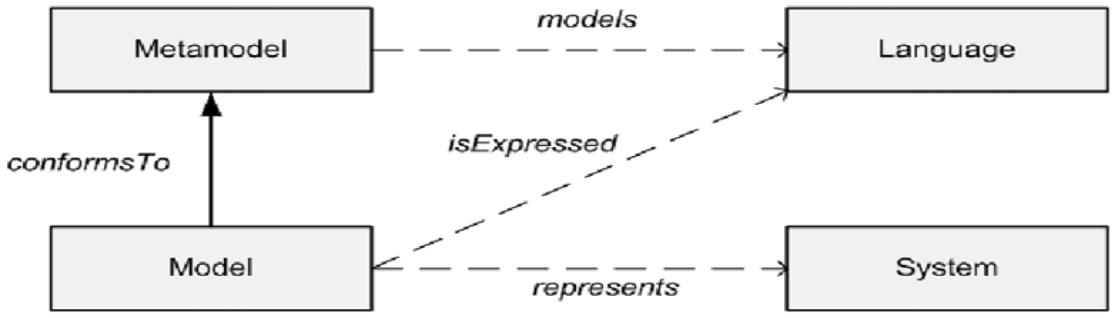


**Figure 1.1 :** Relations entre système, modèle, méta-modèle et langage.

L'architecture de méta modèle traditionnelle proposée par OMG est basée sur 4 méta niveaux différents. Dans ce schéma, les modèles au niveau méta sont utilisés pour spécifier les modèles dans les niveaux méta suivants. À leur tour, les modèles au niveau méta peuvent être considérés comme des instances de certains des modèles au niveau méta décrits ci-dessus [5].

**Les quatre méta-niveaux sont :**

1. q ***(M0) :*** à ce niveau se trouve le système réel, système modélisé.
2. q ***(M1) :*** à ce niveau se trouve le modèle du système réel défini dans un certain langage. Il peut être les classes d'un système orienté objet, ou les définitions des tables d'une base de données relationnelle.
3. q ***(M2) :*** à ce niveau se trouve le méta-modèle définissant ce langage, par exemple, les éléments UML comme la classe, l'attribut et l'opération.
4. q ***(M3) :*** à ce niveau se trouve le méta-méta-modèle définissant les propriétés de tous les méta-modèles.

Le tableau définit les concepts qui différencient ces méta-niveaux.

**Tableau 2.1 :** Niveaux d'abstraction de la méta-modélisation.

| | |
|---|---|
| Meta-méta-modèle | Langage de spécification des méta-modèles |
| Méta-modèle | Définition du langage utilisé pour exprimer Modèle |
| Modèle | Abstraction du système |
| Système | Information et flux de contrôle d'un domaine |

- **Conforme :** Elle est définie par un ensemble de contraintes entre un modèle et son méta-modèle et qui expriment les relations entre eux [6].

## 2.4 Les approche MDA et IDM

Il s'agit de modéliser l'application que l'on veut créer indépendamment de l'implémentation cible (niveau matériel ou logiciel). Cela permet de nombreuses réutilisations de modèles. Le modèle ainsi créé (PIM-Platform Independent Model) est associé au Platform Model (PM-Platform Model) et transformé pour obtenir un Platform-spécifique Application Model (PSM-Platform-Spécifique Model). Les outils de génération automatique de code peuvent alors créer des programmes directement à partir du modèle. Ces méthodes permettent également aux applications d'évoluer facilement à partir de modèles. Lorsque tous les modèles nécessaires sont disponibles, le développement de nouveaux modules peut ne pas prendre plus de quelques minutes

### 2.4.1 L'ingénierie des logiciels (IDM)

Le Model Drivent Engineering est une approche plus globale et générale que le MDA. Il tente d'appliquer ces principes à n'importe quel domaine technique, comme l'orientation objet, les ontologies, les documents XML ou les grammaires de langage. Par conséquent, MDA peut être considéré comme un processus spécifique de type IDM.IDM est basé sur les principes suivants :

- **Capitalisation :** les modèles doivent être réutilisables.

- **Abstraction :** les modèles doivent être indépendant des technologies de mise en œuvre afin d'adapter la logique métier à différents contextes et de permettre de faire plus facilement évoluer les applications vers de nouvelles technologies.
- **Modélisation :** abordée selon une vision productive (pour générer le code final du logiciel pour une technologie de mise en œuvre donnée) par opposition à la traditionnelle vision contemplative (but de documentation, spécification, communication).
- **Séparation des préoccupations :** l'IDM s'illustre généralement selon les deux principales préoccupations du métier et de la plateforme de mise en œuvre mais d'autres préoccupations sont possibles.

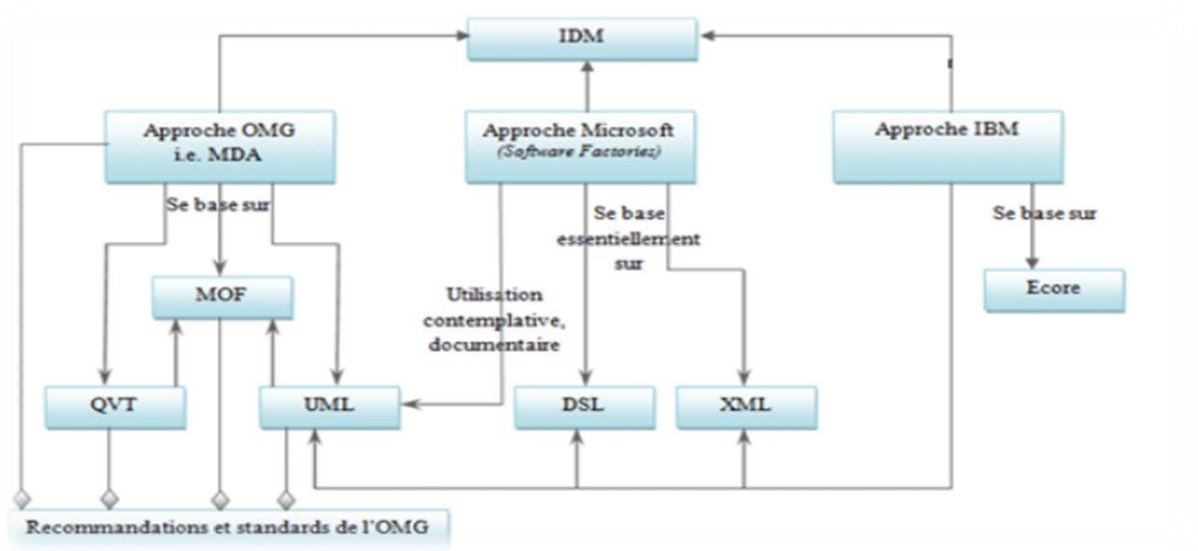


**Figure 2.2** : Les variantes de l'IDM.

## 2.4.2 L'approche MDA Architecture de méta modélisation(MDA)

MDA est une implémentation de la méthode IDM. Elle repose sur la séparation de la spécification fonctionnelle d'un système de son implémentation sur une plate-forme spécifique, approche basée sur le principe de séparation des préoccupations. De la phase IE (ingénierie des exigences) à la phase d'implémentation, selon le processus de développement orienté MDA, tout est traité comme un modèle [7]. MDA identifie quatre types de modèles : CIM, PIM, PDM et PSM.

- **CIM (Computer Independent Model) :** Correspondant aux modèles du domaine (domaine/business model) indépendant de tout implémentation informatique et recensant les besoins des utilisateurs en utilisant le vocabulaire du praticien [7].
- **PIM (Plateforme Independent Model) :** Correspondant à la spécification de la partie ”métier” d'une application, conformément à une analyse informatique cherchant à répondre aux besoins métiers indépendamment de la technologie de mise en œuvre [7].
- **PSM (Plateforme Specific Model) :** est le résultat de la transformation du Modèle PIM prenant en compte les spécificités de la plateforme d'exécution [8].

- **PDM (Plateforme Description Modèle) :** Ces modèles fournissent l'ensemble de concepts techniques liés à la plateforme d'exécution et à ses services. Ils contiennent toutes les informations nécessaires à sa manipulation.

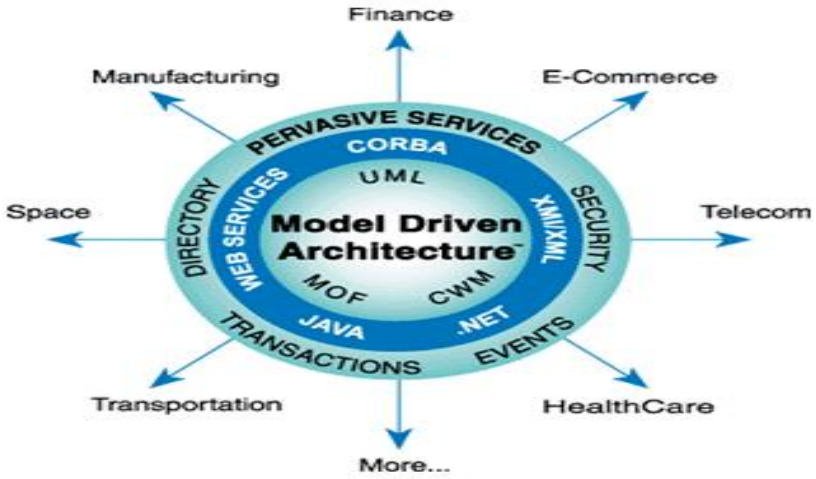


**Figure 2.3:** illustration OMG du MDA.

Il existe des transformations de modèle du système vers d'autre modèle du même système, on peut citer [9].

1. Transformation de Modèle CIM vers PSM.
2. Transformation de Modèle CIM vers PIM.
3. Transformation de Modèle PIM vers PSM.

### 2.4.2.1 Architecture de méta modélisation(MDA)

L'architecture MDA proposée par L'OMG est basée sur quatre niveaux distincts. Le MDA résumé sur le pyramide suivant :

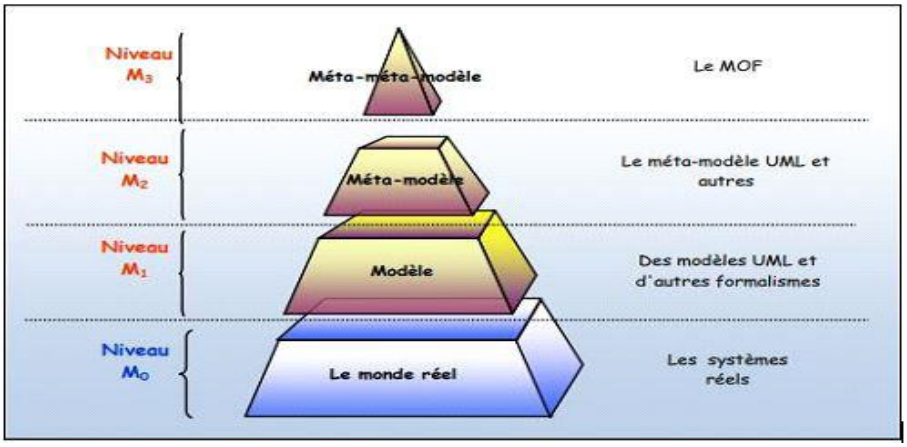


**Figure 2.4**: Pyramide de modélisation de l'OMG.

1. **n**iveau M0 : Le monde réel.
2. **n**iveau M1 : Les modèles représentant cette réalité.
3. **n**iveau M2 : Les méta-modèles permettant la définition de ces modèles.
4. **n**iveau M3 : Le méta-méta-modèle, il est unique et méta circulaire, il est noté MOF.

### 2.4.2.2 Les outils d'MDA

Pour atteindre cette efficacité, plusieurs outils conceptuels sont disponibles. La technologie MDA (Model-Driven Architecture) est développée par OMG (Object- Management Group), qui fournit également UML (Unified Modeling Language) et Corbas (courtier de demande d'objet). Ces outils sont :

- UML largement utilisé par ailleurs qui permet une mise en œuvre aisée de MDA en offrant un support connu.
- XMI (XML Meta data Interchange) qui propose un formalisme de structuration des documents XML de telle sorte qu'ils permettent de représenter des métadonnées d'application de manière compatible.
- MOF (Meta Object Facilité) spécification qui permet le stockage, l'accès la manipulation, la modification de métadonnées. Le MOF permet une unification de l'expression des méta-modèles, qu'ils soient ensuite utilisés comme profils UML ou non .
- CWM base de données pour métadonnées.

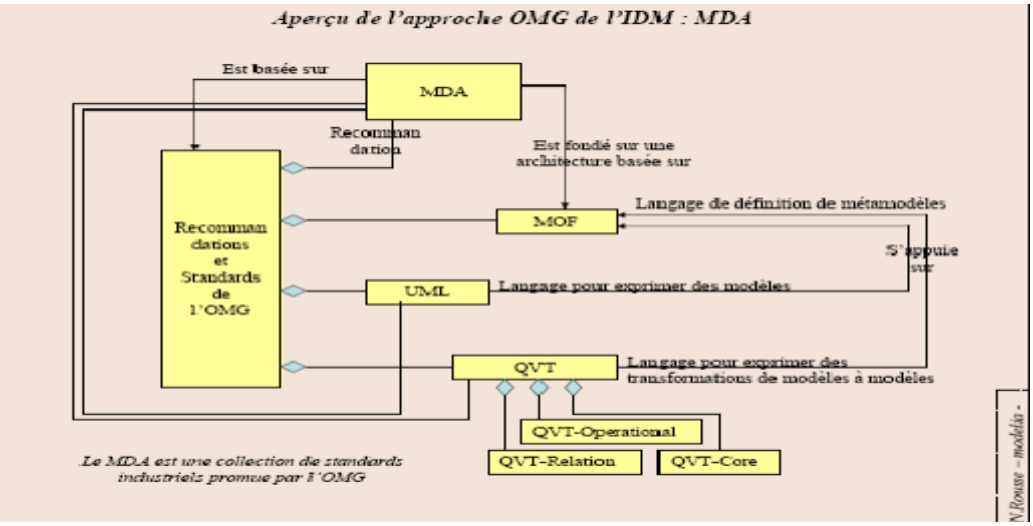


**Figure 2.5** : Les outils méta modélisation.

## 2.5 Les avantage de l'IDM

- **Traçabilité et synchronisation entre modèle et code source** : une approche Développement de logiciels traditionnels, les modèles sont fabriqués en marge L'implémentation et le code source sont générés manuellement, ce qui prend du temps. Ce Cela peut causer des problèmes majeurs lors de la modification du code. En fait, ils sont très peu nombreux Mises à jour du modèle d'accompagnement [10]. En ce sens, MDI améliore Traçabilité et synchronisation entre modèles.
- **Meilleure réutilisation *des* modèles :** une fois définie, l'architecture du logiciel et les modèles peuvent être réutilisés dans d'autres systèmes [11].
- **Logiciel de qualité :** Le code source généré à partir du modèle est en principe sans erreur Parce que le modèle est testé et validé (ex : architecture JEE, .NET, etc.). Par conséquent,

la qualité logicielle devient dépendante de la réalisation de concepts de domaine métier plutôt que d'éléments architecturaux logiciels [11].

- **Séparation des préoccupations :** la réutilisation des modèles offre l'avantage de focaliser davantage sur les concepts liés aux domaines d'affaires plutôt qu'aux concepts liés à leur implémentation [11 [12].

## 2.6 Transformation de Modèle

La transformation de modèle est au cœur du MDE et y joue un rôle important. Il peut être défini comme la génération d'un modèle cible à partir d'un modèle source, suivant une définition de transformation, qui est un ensemble de règles de transformation décrivant comment transformer un modèle dans le langage source en un modèle dans le langage cible. Une règle de transformation est une description de la manière de transformer une ou plusieurs constructions du langage source en une ou plusieurs constructions du langage cible. Pour qu'une transformation soit appliquée de manière répétée, elle doit être appliquée indépendamment du modèle source. Le concept de transformation de base est illustré à la figure 7.

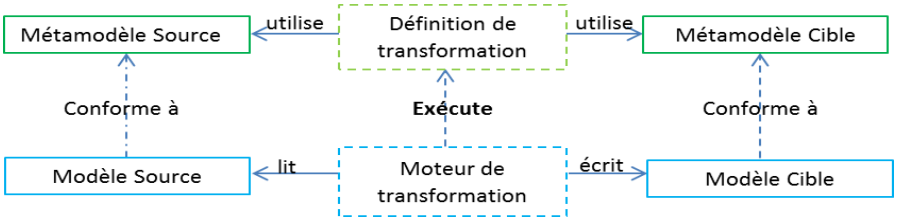


**Figure 2.6**: Concepts de base de transformation de modèles.

### 2.6.1 Définition

Les transformations sont générées automatiquement à partir d'un modèle cible Le modèle source, selon la définition de la transformation. Une définition de transformation est un ensemble de règles de transformation Décrit comment convertir le modèle source en modèle cible [13].

### 2.6.2 Pourquoi la transformation de modèles

La transformation des modèles joue un rôle fondamental dans le développement Logiciel et sert à diverses fins, telles que la traduction de modèles, l'ingénierie inverse, la génération de code et la migration de données. L'image ci-dessous représente ce concept.

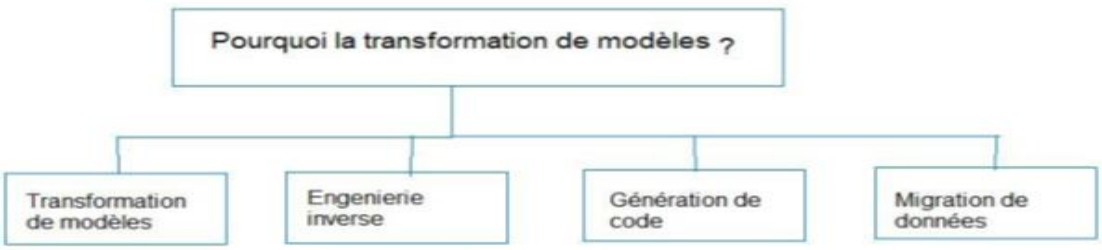


**Figure 2.7:** Les activités de la transformation de modèles.

## 2.7 Propriétés de transformation de modèles

Les principales propriétés qui caractérisent les transformations de modèles sont :

- **Traçabilité :** La traçabilité est la propriété d'avoir des fichiers liés entre les éléments du modèle source et cible et les différentes étapes du processus de transformation. Les liens de traçabilité peuvent être stockés dans le modèle source, dans le modèle cible ou dans un modèle séparé.
- **Direction alité ou Réversibilité :** Une transformation est dite réversible si elle peut être effectuée dans les deux sens. Exemple : modèle vers texte et texte vers modèle. On dit aussi qu'une transformation est réversible s'il existe une transformation qui permet de retrouver le modèle source à partir du modèle cible.
- **Réutilisabilité :** La réutilisabilité peut être mesurée par la possibilité d'adapter et de réutiliser les règles de transformation du modèle. d'autres métamorphoses. L'identification des patrons de transformation est un moyen d'atteindre cette réutilisabilité.
- **Planifications :** les planifications incluent des séquences de règles qui représentent l'exécution des transitions. En fait, les règles de transformation peuvent déclencher d'autres règles.
- **Modularité :** les transformations modulaires peuvent regrouper des règles de transformation en partitionnant le problème. Un langage de transformation de modèle qui prend en charge la modularité facilite la réutilisation des règles de transformation.

## 2.8 Les étapes de la transformation de modèles en MDA

Dans MDA, la transformation de modèle est basée sur le méta modèle, qui se compose de deux Les étapes successives sont [9] :

1. La spécification des règles de transformation permet de définir la correspondance entre les concepts du méta modèle du modèle source et les concepts du méta modelé du modèle cible .
2. L'application des règles de transformation peut basculer automatiquement du modèle source vers le modèle cible. Un outil de conversion est nécessaire pour exécuter ces modèles.

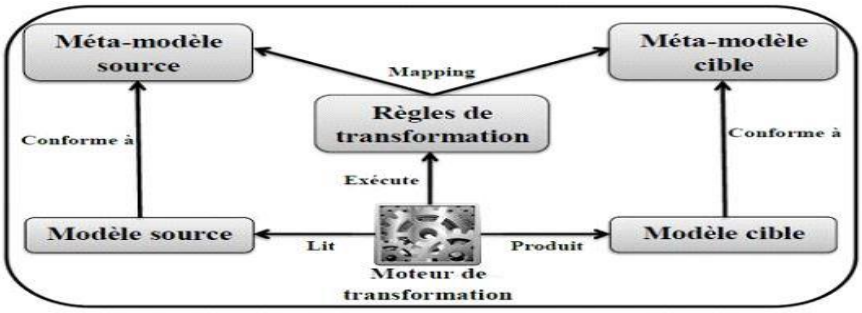


**Figure 2.8:** Concepts de base de la transformation de modèles.

## 2.9 Les types de transformation de modèles

Dans la littérature, on peut distinguer trois type de transformation :

- **Transformation horizontale :** Ces transformations maintiennent le même niveau d'abstraction en modifiant la représentation du modèle source (ajouter, modifier, supprimer

- **Transformation verticale :** Les sources et les objectifs de la transformation verticale sont définis à différents niveaux abstraction. Le raffinement fait référence à une transformation d'ingénierie pilotée par les modèles (MDE) qui réduit le niveau d'abstraction. L'abstraction représente des transformations qui élèvent le niveau d'abstraction [8].
- Transformation oblique : Ces transformations sont généralement utilisées par l'optimisation des compilateurs Code source avant de générer le code exécutable. Elles résultent d'une combinaison des deux premières conversions de type [8]. On peut également distinguer deux types de transformations de modèles sur la base de méta-modèles, qui représentent les modèles source et cible de la transformation :

1. **Endogène :** C'est une transformation entre modèles exprimés dans le même Méta-modèle [8].
2. **Exogène :** C'est une transformation entre modèles exprimés dans différents Méta-modèles .

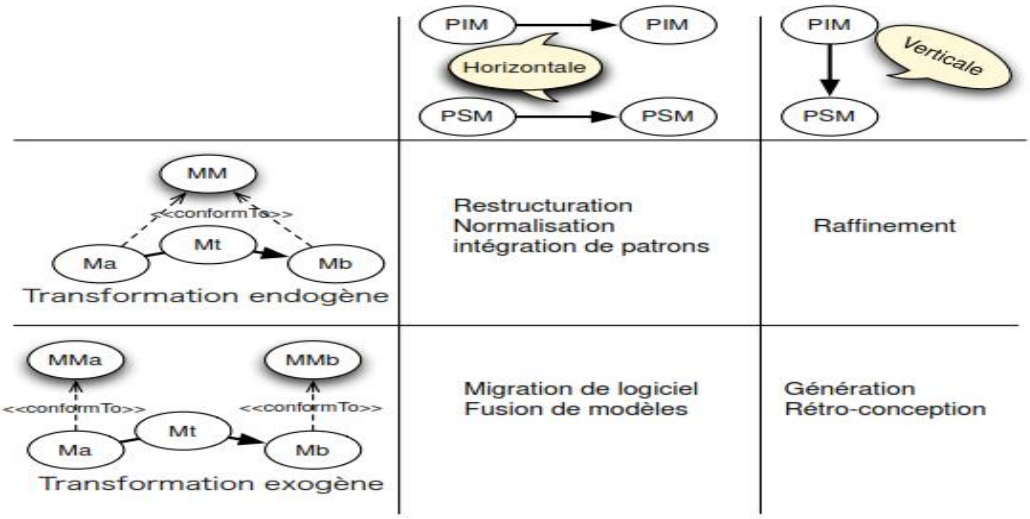


**Figure 2.9**: Les types de transformation de modèles.

## 2.10 Approches de transformation de modèles

On va présenter une classification des approches de transformation en se basant sur le travail de CZarnneki [14] qui décompose la transformation de modèle en deux catégories : les transformations de type modèle vers code et qui les transformations de type modèle vers modèle.

**Tableau 1.2 :**Les approches de transformation de modèles .

| Approches de transformation de modèles | |
|---|---|
| M2M | M2T |
| Dirigée par la structure | Parcours de modèles (programmation) |
| Manipulation directe | |
| Approche relationnelle | Template |
| Transformation de graphes | |
| Hybride | |

### 2.10.1 Une transformation de type modèle vers code

Il existe deux approches de transformations de type modèle vers code :
es approches basées sur le principe du visiteur (Visitor-based approach) et les approches basées sur le principe des patrons (Template-based approach).

- L'approche basée sur le principe du visiteur consiste à réduire les différences sémantiques entre le modèle et le langage de programmation cible en fonction du mécanisme du visiteur, notamment en ajoutant des éléments au modèle et en parcourant le modèle riche pour obtenir du code afin de créer du texte de flux. Le projet Jamda de génération de code Java est un exemple de cette approche [16].
- L'approche basée sur le principe de modèle est actuellement la plus largement utilisée, et la plupart des outils MDA actuellement disponibles prennent en charge ce principe de génération de code à partir de modèles. Parmi les outils basés sur ce principe, on peut citer : OptimalJ, XDE (providing model transformation).
- vers modèle aussi), JET, ArcStyler et AndroMDA (un générateur de code qui se repose notamment sur la technologie ouverte Velocity pour l'écriture des patrons). Le principe de ces approche repose sur l'utilisation des morceaux de méta-code obtenu a partir du code cible et les utilisés pour accéder aux informations du modèle source.

### 2.10.2 Une transformation de type modèle vers modèle

Depuis l'avènement de MDA, la conversion de type de modèle à modèle s'est poursuivie à ce jour. [17]. Il existe un grand espace d'abstraction entre PIM et PSM, il est donc recommandé d'utiliser des modèles intermédiaires au lieu de passer directement de PIM à PSM, qui sont utilisés pour l'optimisation ou le débogage. Ces transformations sont utiles pour différentes vues des systèmes informatiques, leur synchronisation, leur vérification et leur validation.

## 2.11 Mécanismes de Transformation

On distingue deux ensembles de mécanismes de transformation :
1. Les mécanismes de transformation qui reposent sur une approche déclarative. Elles se concentrent sur ce qui doit être transformé et en quoi doit être transformé.
2. Les mécanismes de transformation qui reposent sur une approche opérationnelle (impérative).
3. Elle se concentrent sur la façon dont la transformation doit être effectuée elle-même.
Les approches déclaratives comprennent les mécanismes suivants :
1. Programmation fonctionnelle.
2. Programmation logique.
3. Transformation de graphe.
Pour notre méthode de transformation on s'intéresse par la transformation de graphe.

## 2.12 La transformation des graphes

La théorie des graphes a ouvert un vaste champ de modélisation, conduisant à Solution efficace à de nombreux problèmes, et dans sa définition intuitive Un graphe est un graphe composé d'un ensemble de points et d'un ensemble de flèches, chaque flèche reliant deux d'entre eux.Les graphes sont utilisés pour modéliser des systèmes complexes d'une certaine manière Simple et intuitif, vous pouvez utiliser la transformation graphique pour spécifier Comment développer ces modèles.

Une transformation de graphe consiste à appliquer une règle au graphe, et la partie du graphe qui correspond à cette règle sera remplacé par une autre image. Répétez ce processus jusqu'à ce qu'il n'y ait plus de règles. peut être appliqué.Cet ensemble de règles de transformation de graphe constitue ce que l'on appelle Modèle de grammaire graphique. Une grammaire des graphes est une généralisation de la grammaire de Chomsky pour les graphes. chaque règle de grammaire Le graphique se compose d'une partie gauche et d'une partie droite. Chaque L'une de ces deux parties est le tableau.

Dans le processus de transformation de graphe, nous distinguons les graphes non terminaux en tant que résultats intermédiaires de l'application de règles et Plan de raccordement (les règles ne peuvent plus s'appliquer). On dit que ceux-ci sont dans les langues générées par la grammaire. Vérifiez si le graphique G est dans Le langage généré par la grammaire des graphes doit être parsé. Traiter L'analyse déterminera un ensemble de règles pour dériver G Pour bien comprendre le principe des transformations de graphes il faut comprendre quelque concept sur la théorie des graphes. Dans les paragraphes suivants on va présenter brièvement la notion de théorie des graphes.

## 2.12.2 Principe de transformation de graphes

Les méthodes de réécriture classiques, telles que la grammaire de Chomsky ou la réécriture de termes, manquent d'expressivité. Des transformations graphiques ont été développées pour résoudre ce problème. Le processus de transformation de graphe comprend Applique itérativement la règle au graphe. Chaque application de règle remplace une partie du graphique par une autre selon la définition de la règle. Le mécanisme de transformation de graphe est le suivant : sélectionner une règle applicable dans l'ensemble de règles ; appliquer cette règle au graphe d'entrée ; rechercher une autre règle applicable (itération) jusqu'à ce que plus aucune règle ne puisse être appliquée. Cette opération est basée sur un ensemble de règles qui respectent une grammaire spécifique, connue sous le nom de modèle de grammaire des graphes.

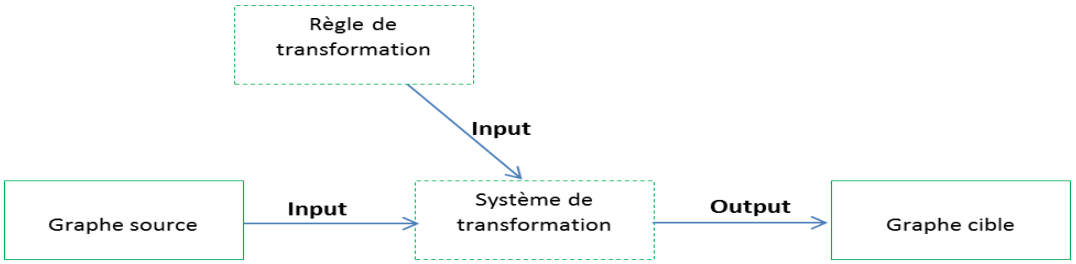


**Figure 2.10** : Principe de mise en œuvre de transformation de modèles.

## 2.13 TGG

La méthode de Triple Graph Grammaire (Triple Graph Grammaire : TGG), proposée par Andy Shūr [20] est une tentative de créer une méthode pour connecter différents systèmes/modèles selon des règles/critères prédéfinis, de sorte que dans un système, les changements dans/modèle conduisent inévitablement à d'autres changements. TGG peut être utilisé pour différents scénarios de transformation et de synchronisation de modèles. TGG est spécifié pour les transformations bidirectionnelles (transformations avant et arrière).
TGG est un triplé de grammaires de graphes, dans lequel nous avons la grammaire de graphes source, la grammaire de graphes cible et la grammaire de graphes correspondante et deux morphismes, le premier reliant la grammaire de graphes source et la grammaire de graphes correspondante, le second Concaténer les grammaires de graphes correspondants et les grammaires de graphes cibles.

### 2.13.1 Processus de transformation de modèles avecTGG

Le formalisme TGG vise à générer et transformer des graphes de corrélation par paires (souvent appelés graphes source et cible) sous un mécanisme de synchronisation bien formé (par exemple, un graphe de correspondance) qui préservera les graphes correspondants après l'application des transformations. La motivation initiale du formalisme était de fournir des outils graphiques de haut niveau pour la modélisation et la spécification de problèmes impliquant des diagrammes connexes (arbres de syntaxe, diagrammes de contrôle) et des structures d'information (exigences, documentation de conception et traçabilité). Plus récemment, des concepts clés du formalisme TGG ont été utilisés dans le langage de transformation OMG standard pour spécifier des "correspondances" pour les transformations. De plus, l'application de TGG dans le domaine des spécifications de conversion bidirectionnelles de modèles sont de plus en plus fréquentes.

### 2.13.2 Fondements théoriques de TGG

A fin de mieux comprendre TGG, nous explicitons à partir de [24] les notions de base pour TGG.

- **Un Graphe :** est défini comme G=(V,E,s,t) ou V ensemble de sommets (nœuds) et E ensemble des Arêtes et s,t: E→V sont des fonctions qui attribuent les sommets source et cible aux arêtes.
- **Un morphisme de graphe :** de G à G' est une paire h =($h_V$ , $h_E$), ou $h_V$ = V→V', $h_E$= E→E' sont définis de telle sorte qu'ils "préservent" les nœuds sources et cibles.
- **Une Grammaire de graphe :** est une paire G=($G_0$, **P**), ou $G_0 \in G$ est l'axiome et **P** ={p: (L $^{l}$ K $^{r}$ R)} est un ensemble de règles de transformation.
- **Une production de graphe monotone :** est une paire de graphes p= (L,R),ou $L \subset R$.
- **Une production de graphe :** p est applicable à un graphe G s'il existe un morphisme h: L→G.
- **Un triple de graphes :** G= (SG $\xrightarrow{hSG}$ CG $\xrightarrow{hTG}$ TG), ou SG graphe source, CG

graphe de correspondance et TG graphe cible, $h_{SG}$ est un morphisme de graphe entre les graphes SG ET CG et $h_{T\,G}$ est morphisme de graphe entre les graphes CG et TG.

- **Une triple de production :** Si la composante de production s'applique à la composante de graphe, alors p s'applique au triplet de graphe G. L'application de p donne un triplet de graphe G' tel que les composantes de production sont mappées sur les composantes de graphe.

TGG est utilisé pour la spécification des outils basés sur des graphes qui peu- vent être classés comme suit:

- La synchronisation entre les graphes source, cible et correspondants permet d'effectuer des modifications de manière synchrone en appliquant des triplets de production.
- La transformation source-cible donne un graphe source SG et une série de triplets de production source, et en appliquant les productions source-cible on obtient un graphe correspondant CG et un graphe cible TG.
- Propagation incrémentale des modifications, en commençant par des triplets de G=(SG,CG,TG), on applique d'abord la source sur SG pour générer une séquence de triplets, puis on propage les modifications sur CG et TG en appliquant la sortie source-cible .

## 2.14 Conclusion

Dans ce chapitre, on a essayé de présenter le concept de transformation de modèles qui est l'une des pièces centrales de l'approche MDA, car elle permet l'automati sation et/ou l'assistance des tâches qui concourent à l'élaboration des modèles, depuis les phases amonts du développement, jusqu'au test en passant par la production de code. Dans ce contexte, on a tenté de présenter les différentes approches existantes dans la littérature, en commençant par une brève introduction aux approches IDM et MDA, ensuite, on présente une énumération des différents types de transformations, suivie par une classification des différents approches, et enfin on présente un cadre spécifique des transformations de modèles basé sur les transformations de graphe.

# Chapitre 3
# Système de production.

## 3.1 Introduction

Tout système de production est associé à plusieurs problèmes illustrés à la figure 14. Pour évaluer la performance d'un système de production, il est important de comprendre sa structure, sa composition, ses différentes fonctions, ses propriétés, ses conditions de fonctionnement et son environnement. Dans ce chapitre, nous introduisons des notions sur les systèmes de production, la modélisation des systèmes de production, et divers outils de modélisation dont le but est d'acquérir des connaissances de base spécifiques aux réglementations et méthodes techniques.

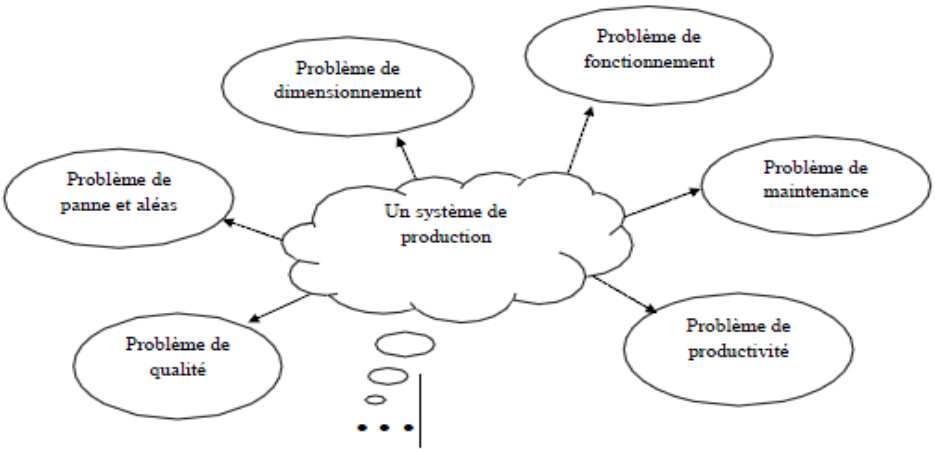


**Figure 3.1**: Les problèmes liés au système de production.

## 3.2 Notion du système de production

De la même manière, nous donnons quelques définitions du «système de production » citées dans la littérature :

- **Un système de production :** car **[**22] est un ensemble de ressources qui supportent cette transformation. Dans cette collection, on distingue essentiellement quatre types de ressources : les équipements (machines, outils, moyens de transport, moyens informatiques, etc.), les ressources humaines pour fluidifier le processus de transformation, la fabrication des produits à différentes étapes (matières premières, produits semi-finis produits, produits finis, etc.), entrepôt de matériel ou zone de stockage. En ce qui concerne les équipements de production, on distingue trois sous-types : les machines de production, qui permettent les opérations d'usinage, les machines de manutention, qui permettent le transport de pièces en atelier (robots, chariots mobiles, tapis roulants, gerbeurs, . . . ) et machines de contrôle qualité. Les deux derniers peuvent être considérés comme des machines de production spéciales ou fictives.
- **Systèmes de production :** Pour [23] la variété des éléments qui composent ces trois groupes et la variété des relations et interactions qui existent entre ces éléments crée un univers très complexe. Par conséquent, le bon fonctionnement d'une entreprise dépend de

la capacité des décideurs à parvenir à faire en sorte que tous ces éléments agissent de manière efficace et cohérente conformément à leurs objectifs.

- **Le système de production de l'entreprise** : L'objectif du système de production est de fournir aux clients des produits de qualité, à des prix compétitifs et dans les délais. Les entreprises doivent continuellement chercher à améliorer leurs produits et leurs délais, et doivent également continuellement réduire leurs coûts de production [22].

## 3.3 Caractéristiques des systèmes de production

Le système de production peuvent être caractérisé par :

- un grand nombre de processus (fabrication, transport, stockage…) .
- une forte interaction entre éléments (employés, ressources, stocks…).
- un manque d'information pour la gestion (marché, fournisseurs, concurrence…).
- des phénomènes aléatoires (pannes, arrivée des approvisionnements…).
- une nature incertaine des décisions à prendre (pas de règle unique et claire).
- une grande variété et hétérogénéité des critères d'appréciation [24].

## 3.4 Caractéristiques de la performance d'un système de production

Dans l'environnement concurrentiel et fluctuant de l'entreprise, quelques critères permettent de caractériser la performance d'un système de production. Ils sont :

- **La productivité :** à savoir la quantité de produits par unité de temps, par ressource utilisée
- **La flexibilité :** qui caractérise la capacité du système de production à changer de configuration en fonction de la demande ;
- **La qualité :** qui mesure l'aptitude du produit à satisfaire les besoins du client [23].

## 3.5 La maîtrise des systèmes de production

Pour des raisons de nature différente (techniques, organisationnelles, décisionnelles, dynamiques, etc.), maîtriser un SoP est une tâche difficile et délicate. Par conséquent, le contrôle du système dépend de plusieurs paramètres, et généralement de différents aspects :

- Il existe de nombreux paramètres qui composent un SoP. Il y a des paramètres physiques, des paramètres organisationnels, des paramètres décisionnels, des paramètres de gestion liés aux ressources et aux produits… ;
- La dynamique du SoP n'est ni linéaire ni permanente. Le système est constamment soumis à des risques et incertitudes de part et d'autre (clients, prévisions, pannes, absences, délais, qualité, etc.) ;
- SdP est un système distribué dans lequel plusieurs tâches sont exécutées simultanément, fonctionnant de manière synchrone ;

- Les objectifs des différentes fonctions du système sont souvent opposés. La satisfaction de l'objectif de base ne correspond pas nécessairement à la satisfaction de l'objectif global. Tout cela rend SdP difficile à comprendre et ses tâches de contrôle sont complexes, nécessitant des méthodes et des outils très sophistiqués. En ce sens, les processus, méthodes et techniques de production sont maîtrisés, la modélisation, le guidage, l'aide à la décision, la planification, l'évaluation des performances, etc. sont développés et mis en œuvre [25].

## 3.6 Modélisation d'un système de production

De manière générale, la modélisation des systèmes de production est un problème complexe. Avoir un "bon" modèle pour un système donné fournit l'assistance nécessaire à la fois au concepteur (le système en conception) et à l'utilisateur final (le système en fonctionnement). Le modèle leur permettra de simuler, d'évaluer les performances, d'optimiser les opérations et de tester différentes stratégies et architectures de gestion en fonction d'objectifs.

Si en théorie il est toujours possible de trouver un modèle pouvant représenter la dynamique d'un système de production, en pratique cette représentation est souvent une tâche très difficile. En fait, la plupart des méthodes existantes sont limitées à une classe spécifique de systèmes. Par ailleurs, il existe de nombreuses méthodes décrites dans la littérature qui, bien que bien formalisées mathématiquement, sont encore difficiles à mettre en œuvre en pratique.

Dans ce cadre, afin de faciliter l'étude des systèmes de production, de nombreuses hypothèses concernant la structure du système, sa dynamique et son environnement (les aléas pouvant survenir) sont souvent posées.

D'un point de vue pratique, il est important d'insister sur le fait qu'un modèle ne peut pas représenter de façon exacte le comportement d'un système réel. En effet, bien qu'il s'adapte à un aspect particulier du système, le modèle ne peut pas le représenter dans sa globalité. De ce fait, le choix de l'outil de modélisation ou de représentation doit dépendre des propriétés du système qu'on désire exhiber dans le modèle [26].

### 3.6.1 Définitions

#### 3.6.1.1 La modélisation

La modélisation est l'acte de conception soignée et de construction intentionnelle de modèles par une combinaison de symboles susceptibles de rendre intelligibles des phénomènes perceptifs complexes et d'amplifier le raisonnement d'acteurs qui envisagent d'intervenir intentionnellement sur le phénomène ; le raisonnement vise notamment à prédire ces éventuelles conséquences de plans d'action [27].

### 3.6.1.2 Un modèle

Il existe une multitude de définitions du terme ‘modèle’. Par modèles, on peut entendre :

- **Modèle :** Représentation simplifiée constituant un système, développée dans un but précis prédiction, compréhension, manipulation, etc.
- **Modèle :** est une représentation utile d'un sujet spécifique. C'est une abstraction (plus ou moins formelle) de la réalité (ou du monde du discours), exprimée à l'aide d'un formalisme (ou d'un langage) défini par la modélisation de concepts en fonction des besoins de l'utilisateur. Il s'agit d'un problème décrivant les processus organisationnels et opérationnels d'une entreprise, dans le but de simuler ces processus pour comparer différents scénarios, ou dans le but de les analyser et de les réorganiser pour améliorer la performance de l'entreprise [27].
- **Modèle :** Un modèle est une image simplifiée de la réalité qui nous sert à comprendre le fonctionnement d’un système en fonction d’une question. Tout modèle est constitué d’une part de la description de la structure du système, qui incorpore les spécifications sémantiques intégrées (nomenclature et règles d’affectation du sens pour une source, étendue et profondeur du consensus social donnant du sens dans le cas d’une représentation à dire d’acteurs) et d’autre part de la description des fonctionnements réguliers (ou non) et des dynamiques qui modifient cette structure au cours du temps. Ainsi, un modèle :
    1. Doit avoir un caractère de ressemblance avec le système réel,
    2. Doit aider à la compréhension du système réel [28].

Généralement on peut dire que ces définitions mettent l’accent sur le fait que le modèle doit permettre l’apprentissage du système modélisé. En effet, la construction d’un modèle est un processus d’apprentissage qui va nous permettre d’identifier les limites de notre connaissance sur le système [28].

## 3.6.2 Choix de modélisation

En vue d’une modélisation, tout système doit être appréhendé selon une vue externe et une vue interne. La vue externe décrit l’environnement du système ainsi que la finalité (les fonctions) du système dans cet environnement (ce pour quoi il a été conçu). La vue interne décrit tous les constituants du système et leurs interactions.

Nous retiendrons que pour modéliser un système, il est nécessaire de définir:

- Les limites du système avec son environnement, c'est-à-dire l’identification des éléments n’appartenant pas au système. Le choix de la frontière du système, qui permet de le distinguer de son environnement, dépend du point de vue de l’observateur. Celui-ci la choisit en fonction de son besoin de modéliser la réalité;
- Les finalités du système dans son environnement, qui « expriment sa raison d’être en termes économiques, éthiques et sociologiques. Elles reflètent l’idée qu’un groupe humain se fait des missions d’un système, en ce sens qu’elles ne sont pas directement opératoires »; [30].
- Les interactions du système avec son environnement, à savoir le type de relations et les influences de ces relations sur le système.

- La vue interne fonctionnelle du système qui présente les processus mis en oeuvre au sein du système.
- La vue interne organique du système qui décrit les éléments composant le système, et les relations entre ces composants.
- La dynamique d'évolution du système, en rapport avec l'évolution de l'environnement. [30].

### 3.6.2.1 Modélisation externe des systèmes de production

La vue externe met en évidence, pour un système donné, son rôle et ses interactions vis-à-vis de son environnement. Il s'agit des buts « de mission », qui caractérisent les sorties du système, attendues par son environnement.

### 3.6.2.2 Modélisation interne des systèmes de production

- **Modélisation fonctionnelle :** La vue fonctionnelle d'un système décrit les fonctions internes permettant à celui-ci d'exercer les fonctions externes, sans présomption de l'organisation elle-même [30].
- **Modélisation organique :** La modélisation organique d'un système décrit l'organisation des ressources rendant le système opérant. Il s'agit donc de rendre compte de la manière dont les fonctionnalités du système sont réalisées. En conséquence, la modélisation organique d'un système décrit une réalisation. Elle procure une description des ressources utilisées , et de la dynamique des flux matériels et informationnels entre ces ressources. [30].

### 3.6.3 Buts de la modélisation

L'objectif de la modélisation, dans son sens le plus général, permet :

- Une meilleure compréhension du système.
- Une mesure de performance du système.
- Un dimensionnement/optimisation de systèmes de production (avant/après réalisation).
- Une identification des facteurs critiques.
- Une réponse aux questions « Que se passe-t-il…si …? ».
- Un outil d'aide à la décision en « temps réel » (durant exploitation) .
- Un outil de formation du personnel (simulateurs de vol, simulateurs de choix…) [24].

## 3.7 Outils pour la modélisation des systèmes de production

Dans ce qui suit, nous présentons quelques outils de modélisation mathématique et informatique, souvent citée dans les travaux de modélisation:

## 3.7.1 Outils de modélisation mathématique

Les fondements des méthodes analytiques sont à la base d'outils assez pratiques, fréquemment utilisés dans l'évaluation des performances des SdPs. Ces méthodes exigent qu'un modèle mathématique soit d'abord trouvé pour représenter le système étudié et que l'on dispose des outils mathématiques qui permettent d'étudier ce modèle. Une démarche analytique se décompose en trois étapes :

- recherche d'une approche analytique adéquate (modèle mathématique qui s'adapte au cas étudié).
- développement du modèle (émission des hypothèses adaptatives du système réel à la théorie adoptée et déduction du modèle).
- implémentation, utilisation et exploitation du modèle [24].

Les techniques permettant d'étudier de manière analytique un SdP sont nombreuses on peut citer :

- **les chaînes de Markov :** sont utilisées dans la modélisation des systèmes stochastiques et dans lesquelles on définit l'état d'un système par un comportement entièrement probabiliste, [24].
- **Les réseaux de Pétri :** Les réseaux de Petri, sont une des nombreuses représentations mathématiques d'un système. Un réseau de Pétri décrit graphiquement la structure d'un système distribué comme un graphe direct biparti avec annotations [28].
- **Les réseaux files d'attente :** la théorie des files d'attente consiste à représenter le système sous forme d'un réseau de serveurs, de zones d'attente et de clients [24].

La théorie des files d'attente fut développée pour fournir des modèles permettant de prévoir le comportement de systèmes répondant à des demandes aléatoires. La théorie des files d'attente a de nombreuses applications dans:

1. La gestion de trafic (réseaux de communication, compagnies aériennes, embouteillages, …).
2. La planification (opérations sur des machines de production, programmes sur des ordinateurs, …).
3. Le dimensionnement d'infrastructures (usines, …) [31].

- **Les réseaux Bayésien :** Les réseaux bayésiens [32]**,** sont des modèles graphiques interprétés à partir de systèmes experts probabilistes pour représenter des relations qualitatives et quantitatives entre plusieurs variables au travers de dépendances et de probabilités conditionnelles. Ils sont encore peu connus et utilises en fiabilité mais tendent à émerger pour répondre a des problématiques d'optimisation des politiques de maintenance.

## 3.7.2 Outils de modélisation informatique

- **Le modèle entité / association :** Le modèle Entité/Association a été proposé par *Chen*, en 1976 pour la modélisation des données et des liens existant entre elles, avec des concepts simples et efficaces. C'est une représentation naturelle du monde réel du système à étudier.

Il est bâti autour de trois concepts : Entité, Association et Propriétés et permet une description graphique.

- **Le modèle relationnel :** Ce modèle permet une description tabulaire de données, c'est-à-dire, les données sont structurées en tables (relations). De plus, l'algèbre relationnelle permet la manipulation de ces données en s'appuyant sur la théorie des ensembles. Le succès du modèle relationnel auprès des chercheurs, concepteurs et utilisateurs est dû à la puissance et à la simplicité de ses concepts donc facile à comprendre par les utilisateurs. La méthode d'analyse MERISE s'appuie sur la combinaison des concepts entité/association et modèle relationnel.
- **Le modèle orienté objet :** Le modèle orientée objet considère le système comme une collection d'objets dissociés, identifiés et possédant des caractéristiques. Une caractéristique est soit un attribut (une donnée caractérisant l'état de l'objet ou attribut), soit une entité comportementale de l'objet (une fonction ou méthode). La fonctionnalité du système émerge alors de l'interaction entre les différents objets qui le constituent. L'une des particularités de cette approche est qu'elle rapproche les données et leurs traitements associés au sein d'un unique objet. La méthode UML qui se veut actuellement un standard dans les méthodologies orientées objet tire toute sa puissance de ce concept.
- **Le modèle orienté agents :** Depuis quelques années, les besoins en terme de méthodes de représentation, de conception et de logiciels ont évolué. La tendance est à la puissance dans la représentation et dans les traitements pour faire face aux grandes masses d'informations, la plupart du temps hétérogènes et distribuées. Ceci nécessite des outils puissants qui s'adaptent aux différentes situations sans toujours nécessiter une intervention externe (du programmeur). Il faut des moyens avec une certaine '*intelligence*', capable de ' *raisonner* ', de ' *prendre de l'initiative*' et qui soient '*apte d'interagir*' dans des '*environnements'*. Un nouveau paradigme de programmation émerge pour pallier les manques des approches traditionnelles : la programmation orientée agent. L'approche orientée agents offre une façon beaucoup plus naturelle de concevoir les systèmes. Elle s'intéresse à la manière de diviser un problème en un ensemble d'entités distribuées et coopérants et à la manière de partager la connaissance du problème afin d'en obtenir la solution. Ce domaine est apparu initialement pour résoudre les problèmes d'intelligence distribuée [28].

## 3.8 Conclusion

A partir de cette présentation on peut conclure que la modélisation des systèmes de production est difficile en raison du nombre et de la diversité des paramètres à prendre en compte et de la complexité des relations entre ces paramètres, et pour la phase d'analyse et d'évaluation d'un système de production utilisent habituellement des indicateurs de performance capables d'évaluer la pertinence du modèle utilisé par rapport au système de production.

Dans ce chapitre, nous avons donné une brève présentation sur la modélisation des systèmes de production : définition, l'objectif et les différents outils de modélisation dont l'objectif consiste à acquérir les connaissances de bases propres à la dispositions technique et méthodologie permettant d'améliorer la maitrise des risques.

# Chapiter4
# Le réseau de pétri.

## 4.1 Introduction

En raison de la complexité croissante des systèmes techniques, il apparaît de plus en plus nécessaire de disposer de méthodes et d'outils de conception et de réalisation particulièrement performants au centre de ces méthodes et outils, qui apparaissent souvent dans la modélisation des procédés. Parmi le grand nombre de techniques formelles qui ont été proposées, les réseaux de Pétri. Dans ce chapitre, nous nous intéressons à l'introduction de Rdp, donc d'abord nous sommes introduits aux concepts de base de Rdp, dans les rubriques suivantes nous abordons brièvement l'évolution de Rdp, les sous-classes de Rdp, puis l'extension de Rdp, les méthodes d'analyse, et enfin modéliser avec Rdp, les principales propriétés de Rdp, le but de Rdp.

## 4.2 Historique

Historiquement, ce réseau a été proposé par Carl Adam Pétri dans son article de 1962 "Communication with Automata" à Bonn, en Allemagne. Ce travail a continué à être développé par Anatol W. Holt, F. Commoner, M. Hack et leurs collègues du groupe de recherche MIT Of Technology (MIT) dans les années 1970. En 1975, la première conférence sur les réseaux de Petri et les méthodes relationnelles a eu lieu au MIT. En 1981, le premier netbook Petri a été publié en anglais par J. Peterson. Aujourd'hui, suivez la newsletter Petri-Net, où 600 à 800 ouvrages Petri Net sont publiés chaque année [7].
Ce formalisme a été proposé comme un outil mathématique permettant la modélisation des systèmes dynamiques à événements discrets. Les réseaux de pétri offrent un outil formel avec une bonne représentation graphique permettant de modéliser et d'analyser les systèmes discrets, notamment les systèmes concurrents et parallèles. L'intérêt majeur de ces réseaux réside dans leur possibilité d'analyser les systèmes modélisés.
En effet, Ce formalisme bénéficie d'une multitude de techniques d'analyse et d'outils [33].

## 4.3 Définitions Réseau de Pétri

Il est clair que le système dynamique ne peut pas être décrit par référence limités à leurs états initial et final. Une description suffisante doit tenir compte Leur comportement permanent est une séquence (éventuellement infinie) d'états. Quelque Des techniques formelles ont été proposées pour spécifier, analyser et vérifier ces système. Les réseaux de Petri représentent une technique formelle largement utilisée. Exister En tant qu'outil mathématique, ils permettent l'analyse des propriétés du système sa structure et son comportement. Ce formalisme bénéficie de riches Techniques et outils analytiques. Les résultats de cette analyse sont utilisés pour évaluer système et permettre sa modification ou son amélioration. La figure ci-dessous montre Modélisation et méthodes générales basées sur la forme de réseau de Petri Analyse du système [33]. En raison de leur polyvalence et de leur flexibilité,les réseaux de Petri sont utilisés dans une large variété de domaines tels que les protocoles de communication, les systèmes distribués, l'architecture des ordinateurs, etc [33].

## 4.4 Concepts de bases de réseau de pétri

### 4.4.1 Définition graphique

Un réseau de Petri (RDP) est un graphe biparti orienté valué, il a deux types de nœuds [35].

- **Les Places :** notées graphiquement par des cercles .Chaque place contient un nombre entier (positif ou nul) de marques (ou jetons), ces derniers sont représenté par des points noir.
- **Les Transitions :** notées graphiquement par un rectangle ou une barre. Une transition qui n'a pas de place en entrée est appelée transition puits. Les places et les transitions sont reliées par des arcs orientées :

1. Un arc relie, soit une place à une transition, soit une transition à une place mais jamais une place à une place ou une transition à une transition.
2. Chaque arc est étiqueté par une valeur (ou un poids), qui est un nombre entier positif, l'arc ayant 'k' poids peut être interprété comme un ensemble de 'k' arcs parallèles. Un arc qui n'a pas d'étiquette est un arc dont le poids est égal à 1.

La figure suivante illustre la transition graphique d'un réseau de pétri.

Les réseaux de Petri permettent de décrire le système en distinguant Clarté entre les aspects statiques et dynamiques, comme sa représentation graphique Facile à comprendre pour les utilisateurs. L'analyse du réseau de Petri peut illustrer Caractéristiques importantes de la structure et du comportement du système dynamique. Les résultats de cette analyse sont utilisés pour évaluer le système et permettre une modification ou une amélioration [9].

### 4.4.2 Définitions informelles de réseaux de pétri

Intuitivement, un réseau de Petri est un graphe bipartite orienté (avec deux types de nœuds) : les emplacements représentés par des cercles et rectangle. Les arcs graphiques peuvent uniquement connecter des positions à des transformations, ou Transitions vers les lieux (pas d'arcs entre les lieux ou les transitions).

D'autre part, les réseaux de Petri décrivent des systèmes dynamiques avec des événements discrets. Les positions permettent de décrire des états possibles du système (elles sont discrètes), et les transitions permettent de décrire des événements ou des actions qui conduisent à des changements d'état. Le réseau est un graphe à sémantique opérationnelle, c'est-à-dire qu'un comportement est associé au graphe, ce qui permet de décrire la dynamique du système représenté. Pour ce faire, ajoutez un troisième élément à la position et transformez, le marqueur. La distribution de jetons à divers endroits à un moment donné s'appelle un réseau de Petri à jetons. Les drapeaux donnent l'état du système. Le nombre de jetons contenus dans un emplacement est un entier positif ou nul et ne peut pas être un nombre négatif. Le drapeau donné permet une transition sensible ou insensible. Une transition est sensible si chaque position d'entrée de la transition contient au moins un jeton. Un ensemble de transitions sensibles pour un jeton donné définit un ensemble de changements d'état possibles pour le système à partir de l'état correspondant à ce jeton. C'est une façon de définir l'ensemble d'événements que le système peut accepter dans cet état. [33].

Un exemple de réseau de pétri est illustré par la figure 4.1.

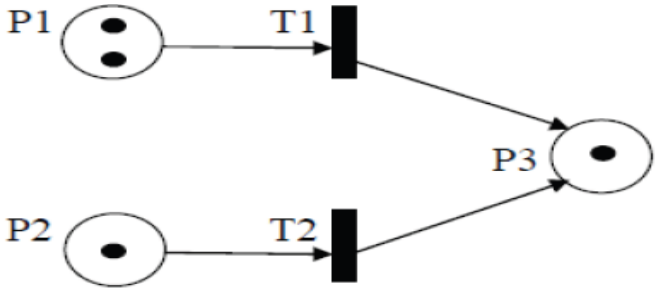


**Figure 4.1** : Exemple d'un Résau de Pétri.

- **Concepts de base pour les réseaux de pétri :** Il existe trios concepts de base : [33].

- **Condition :** Une condition est un prédicat ou une description logique d'un état du Système. Une condition peut être vraie ou fausse. Un état du système peut être décrit comme un ensemble des conditions.
- **Evénement :** Les événements sont des actions se déroulant dans le système. Le Déclenchement d'un événement dépend de l'état du système.
- **Déclenchement, pré-condition, post-condition :** Les conditions nécessaires au Déclenchement d'un événement sont les pré-conditions de l'événement. Lorsqu'un Événement se produit, certaines de ses pré-conditions peuvent cesser d'être vraies Alors que d'autres conditions, appelées post-conditions de l'événement, deviennent Vraies.

## 4.5 Le marquage d'un réseau de pétri

La notation d'un réseau de Petri permet de définir l'état du système modélisé par le réseau. Les jetons consistent à placer initialement un jeton entier (positif ou nul) à chaque position Pi du réseau. Les étiquettes du réseau seront définies par le vecteur M=mi [36].

Un marqueur donné peut être sensible ou insensible à une transition. Il est sensible si chaque position d'entrée de la transformation contient au moins un jeton. Pour un jeton donné, un ensemble de transitions sensibles définit un ensemble de changements d'état possibles pour le système à partir de l'état correspondant à ce jeton. C'est une façon de définir l'ensemble d'événements que le système peut accepter dans cet état [37].

La figure représente un réseau de pétri marqué avec un vecteur de marquage M tel que : M = (1, 0, 1, 0, 0, 2,0).

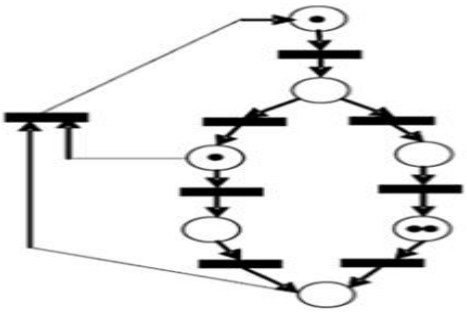

**Figure 4.2** : Le marquage d'un réseau de pétri.

## 4.6 Évolution d'un réseau de pétri

L'évolution de Rdp correspond à l'évolution de son jeton dans le temps (l'évolution de l'état du système), qui se traduit par le déplacement du jeton, pour le passage de toutes les positions d'entrée à l'ensemble des positions de sortie pour cette transition. Ce mouvement s'effectue selon la règle de croisement par la transition t [38].

### 4.6.1 Transition validée

L'évolution de l'état Rdp correspond à l'évolution du tag. Incarner l'état du réseau à un instant donné, capable de passer d'un endroit à un autre en traversant ou en déclenchant des transitions. Une transformation est validée (ou aussi appelée sensible ou transitive ou encore drawable) si tous ses emplacements d'entrée contiennent au moins un jeton [39].


**Figure 4.3** : Transition validée.

### 4.6.2 Franchissement

Le franchissement consiste à enlever un jeton dans chacun des places d'entrée de la transition et à ajouter un jeton dans chacune des places de sortie de la même transition [35].

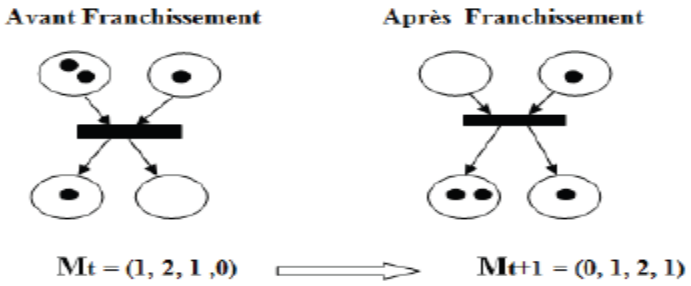


**Figure 4.4** : franchissement d'un Rdp.

« Le franchissement M1=[2 1 1 0], transition t1 à partir du marquage M1 conduit at : M2=[0 1 2 1 ] »,se note : M1(t1 →)M2.

#### 4.6.2.1 Règle de franchissement

Le franchissement consiste à :
1. **R**etirer $\varpi$(p, t) jeton dans chacune des places en entrée P de la transition T.
2. **A**jouter $\varpi$(t, p) jetons à chacune des places en sortie P de la transition T.

La règle de franchissement est illustrée par la figure 21, en utilisant la réaction Chimique connue [35].

2H2+O2 → 2H2O La présence des deux jetons dans chaque place d'entrée indique que 2 unité de H2 et deux unité d' O2 sont disponibles , donc la transition T est franchissable, Après avoir franchi T, le marquage va changer et on obtient le réseau ayant le marquage comme celui de la figure suivante, maintenant T n'est plus franchissable.

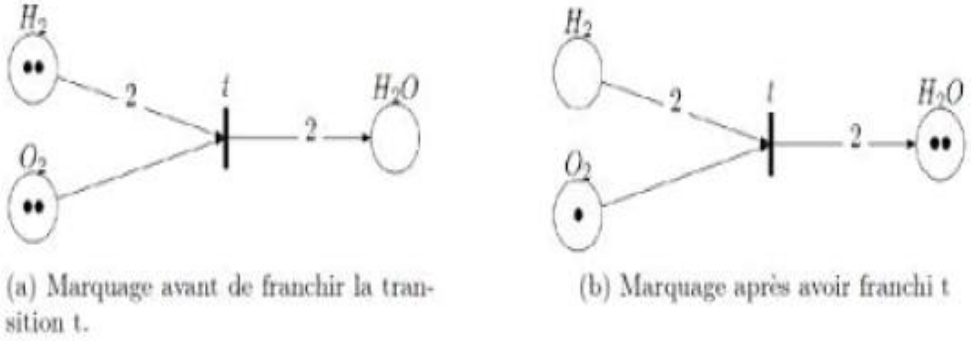


**Figure 4.5 :** exemple de règle de franchissable de transition.

## 4.7 Sous-Classes des réseaux de pétri

- **Graphe d'état :** Un réseau de Petri non étiqueté est un diagramme d'état si et seulement si une transition Il y a exactement une entrée et une sortie. Par exemple : transitions T1, T2, T3, T4 et T4 ont une entrée et une sortie. Comme lui Voir Figure 3.8 pour plus de détails : [40].

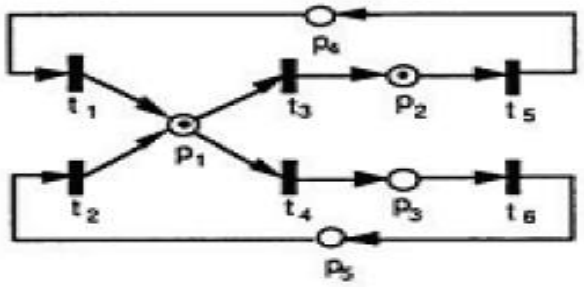


**Figure 4.6 :** Graphe d'état.

- **Graphe d'événement :**Un réseau de Petri est un graphe d'événements si et seulement si partout Exactement une transformation d'entrée et une transformation de sortie est Comme le montre la figure ci-dessous (Figure 23). parfois appelé graphe d'événements Diagramme de transition [41].

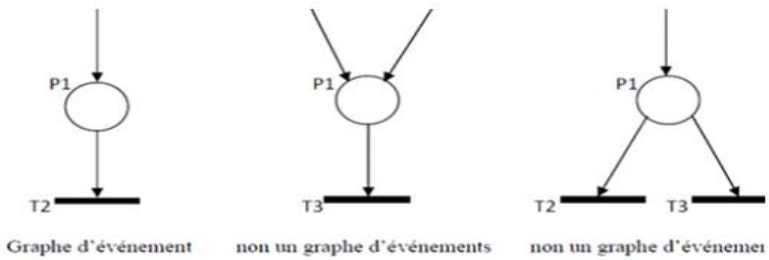


**Figure 4.7** : Graph d'événement.

- **Réseau de Pétri sans conflit :** Un emplacement P avec au moins deux transitions de sortie constitue un conflit dit structurel (également appelé décision ou choix). Nous allons représenter cela par un doublet formé par un ensemble de localisations et leurs transformations de sortie <p1 , t1, t2,... . .> Le conflit indique la possibilité d'exclusion mutuelle entre différents événements du système. Un conflit structurel devient un conflit valide lorsque le marqueur rend le nombre de marqueurs au site P inférieur à la somme des marqueurs des arcs menant à la transition de sortie à ce site, cela signifie donc que la transition vers l'intersection doit être sélectionnée. Dans la figure 10, le double <p1,t3,t4> est un conflit valide. Un Rdp sans conflit est, comme son nom l'indique : un Rdp sans conflits structurels [42].

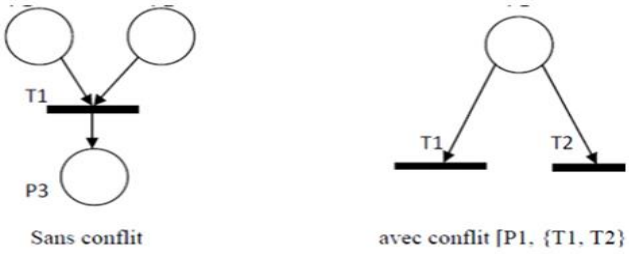


**Figure 4.8**: Graphe RdP sans conflit .

- **Réseau de Pétri pur :** En RdP, une transformation est dite pure si elle n'a pas de position Entrez et sortez. Si toutes les transformations de RdP sont pures. Dans RdP, si une transformation a une entrée et sortie. Si la transformation de RdP est impure, alors RdP est Impur [41].

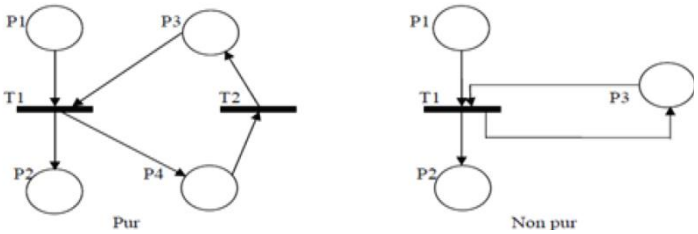


**Figure 4.9**: RdP pur.

- **Réseau de Pétri choix libre :** Si pour tout conflit <p1, t1, t2, ...>, P1 est la seule position d'entrée pour l'ensemble de transformations t1, t2, ...., on dira que Rdp est librement choisi, donc il y a un jeton à p1 Autoriser le croisement de toutes ses transformations exportatrices (sélectionner les transformations rentables) [Savi, 1994]. Cette définition peut être généralisée pour étendre la classe de réseau libre par : si pour tout conflit <p1,t1,t2,...> toutes les transitions t1,t2,... ont le même ensemble à l'entrée..les différentes structures qui caractérisent un réseau à choix libre, à choix libre étendu et a choix non libre sont illustrées dans la figure suivante [42].
- **Réseau de Pétri généralisés :** Un RdP généralisé est un RdP dans lequel des poids (nombres entiers strictement positifs) sont associés aux arcs. Si un arc (Pi,Tj) a un poids K la transition Tj n'est franchie que si la place Pi possède au moins K jetons. Le franchissement consiste à retirer K jetons de la place Pi. Si un arc (Tj, Pi) a un poids K : le

franchissement de la transition rajoute K jetons à la place Pi. Lorsque le poids n'est pas signalé, il est égal à un par défaut. Comme il est exprimé dans la figure suivante 3.14 [40].

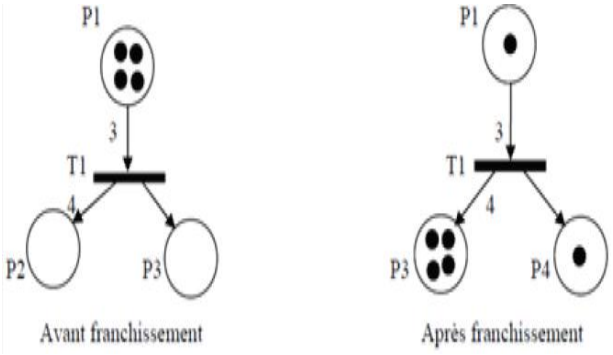


**Figure 4.10** : Rdp généralisé.

- **Réseau de Petri á priorités :** Dans un tel réseau, si on atteint un marquage tel que plusieurs transitions sont franchissables, on doit franchir la transition qui a la plus grande priorité. Dans l'exemple suivant on a présenté un Rdp à priorité comme le schématise dans la figure suivante [40].

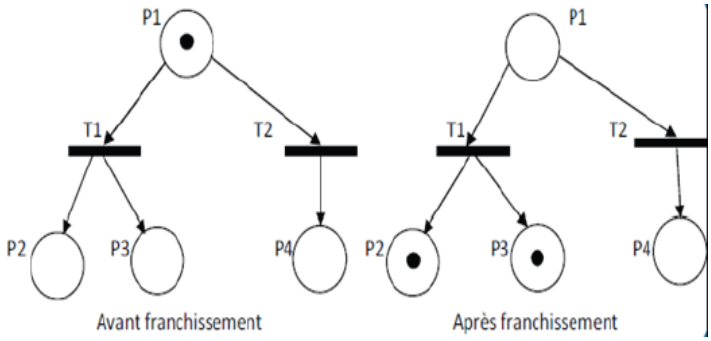


**Figure 4.11** : Rdp á priorité.

- **Réseau de Petri a capacité :** Un RdP à capacités est un RdP dans lequel des capacités (nombre entiers strictement positifs) sont associées aux places. Le franchissement d'une transition d'entrée d'une place Pi dont la capacité est cap(Pi) n'est possible que si le franchissement ne conduit pas à un nombre de jetons dans Pi qui est plus grand que cap(Pi). La Figure 3.15 montre le franchissement de Ti conduit à 3 jetons dans P2 d'où T1 ne peut plus être franchie [40].

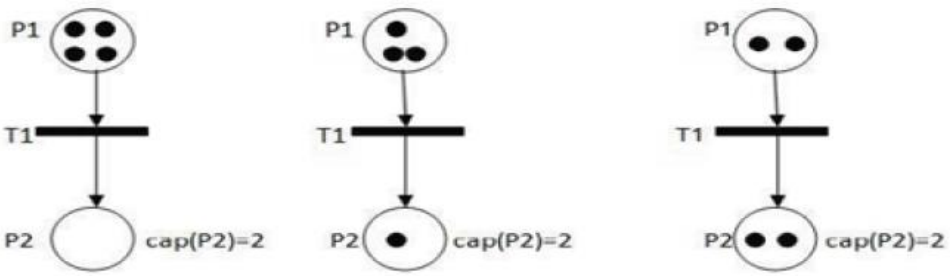


**Figure 4.12** : Rdp á capacité

## 4 .8 Extension des réseaux de pétri

Les systèmes réels imposent de nombreuses contraintes, de sorte que leur modélisation peut générer de grands réseaux de Pétri difficiles à manipuler et à analyser. Très difficile. De plus, Rdp normal ne permet pas d'exprimer certaines propriétés, tels que la liquidité, ce qui rend leur analyse assez difficile. Dans ce cas, plusieurs Des extensions de filets de Pétri ont vu le jour, notamment : les filets de Pétri en couleur, les filets de Pétri chronométrés, etc. Dans la section suivante, nous présenterons une Un aperçu de quelques extensions Rdps[38]**.**

- **Les réseaux de pétri colorés :** La taille du réseau lorsque le nombre d'entités système à modéliser est important Les boîtes de Pétri grossissent rapidement ; l'utilisation d'un réseau coloré peut compresser le modèle si les entités présentent un comportement similaire. Un filet de Petri coloré est un filet de Petri marqué d'une couleur. A,color est l'information attachée au jeton. Ainsi, les arcs sont marqués non seulement par le nombre de marqueurs, mais aussi par leur couleur. L'intersection des transitions dépend alors de la présence du nombre nécessaire de marqueurs dans les positions d'entrée qui satisfont également la couleur de l'arc marqueur. Après avoir étendu une transformation, les marqueurs marquant les arcs d'entrée sont supprimés des positions d'entrée, et les marqueurs marquant les arcs de sortie sont ajoutés aux positions de sortie de cette transformation. Par rapport aux réseaux de Petri, les réseaux colorés ne fournissent pas de capacités descriptives supplémentaires, ils condensent simplement les informations. Tout réseau de Petri coloré étiqueté correspond à un réseau de Petri qui lui est isomorphe. La relation entre RdP coloré et RdP simple est considérée comme la relation entre les langages de programmation de haut niveau et le code assembleur. Théoriquement les deux niveaux d'abstraction ont la même sémantique. De plus le langage de haut niveau offre une grande puissance de modélisation par apport au langage assembleur ; car il est bien structuré, bien typé et modulé [7].

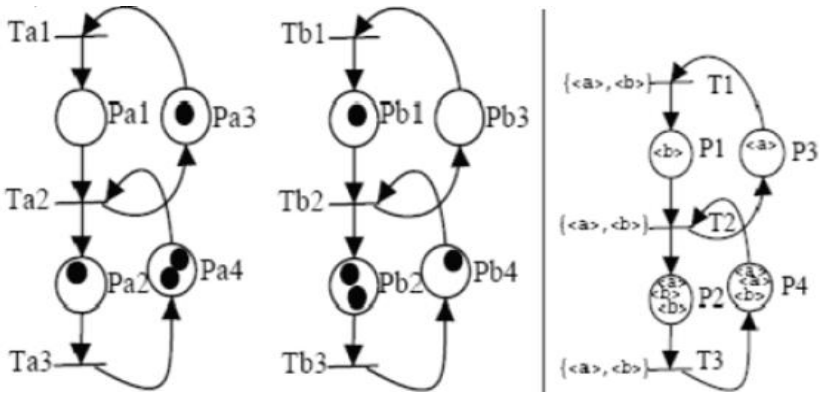


**Figure 4.13** : RdP (gauche) et RdP coloré (droite).

- **Les réseaux de pétri temporisés :** RdPs de synchronisation introduite RdPT étend le RdP avec un intervalle de temps associé à la transition, spécifiant la plage de retard de déclenchement pour la transition. Ils permettent la modélisation de systèmes à contraintes temporelles, tels que : les protocoles de communication, certains cas de systèmes temps réel, etc. L'allongement du temps peut être envisagé sur des lieux, des arcs ou des transitions Merlin définit RdPT comme un RdP avec deux valeurs de temps a et b

associées à des transitions ; et (0 _ a _ b) et b peuvent être infinis, précisez La transition traverse a limite de retard. Considérant la dernière transition temporelle t devient sensible à l'instant : q, alors t ne peut pas être franchi plus tôt qu'à l'instant : q+a ; et ne doit pas être franchi avant (ou juste après) : q+b. L'intervalle [a, b] est le moment où les marqueurs de l'emplacement d'entrée n'existent plus (ils sont conservés).mais pendant lequel les jetons produits ne sont pas encore visibles dans les places de sortie [43].

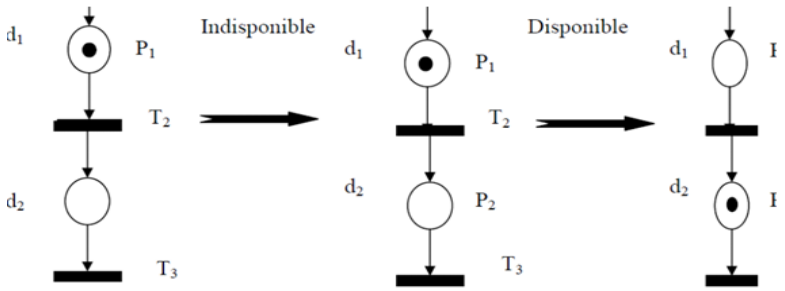


**Figure 4.14** : RdP temporisés

- **Les réseaux de pétri synchronisés :** Dans les modélisations Rdps que nous avons vues précédemment, le fait qu'une transition soit franchissable indique que toutes les conditions sont réunies pour qu'elle soit effectivement franchie. Le moment où se produira le franchissement n'est pas connu. Un Rdp synchronisé est un Rdp ou à chaque transition est associée un événement. La transition sera alors franchie si elle est validée et en plus, quand l'événement associé se produit.La transition est validée quand la condition sur les marquages est satisfaite. Elle devient franchissable quand l'événement externe associé à la transition se produit : elle est alors immédiatement franchie. Si en fonction du marquage de ses places d'entrée, plusieurs franchissements sont possibles, un seul se produira effectivement, celui dont l'événement associé se produit en premier [37].
- **Les réseaux de pétri stochastique :** Pour pouvoir utiliser la puissance de l'analyse de Markov, le système ne doit avoir aucune mémoire du passé, c'est-à-dire si un événement produit un croisement de transition t et convertit le marqueur M1 en M2, le futur évolutif Les transitions sensibilisées par M1 avant de franchir t doivent être les mêmes que les transitions qu'ils auraient subies s'ils venaient d'être sensibilisés par M2. Seules les distributions géométriques et exponentielles confirment ce fait. Les réseaux de Pétri stochastiques sont définis par de telles distributions afin qu'un processus de Markov équivalent puisse être construit pour analyser le comportement du réseau. Les réseaux de Petri stochastiques augmentent l'incertitude et la probabilité de transitions croisées [44].

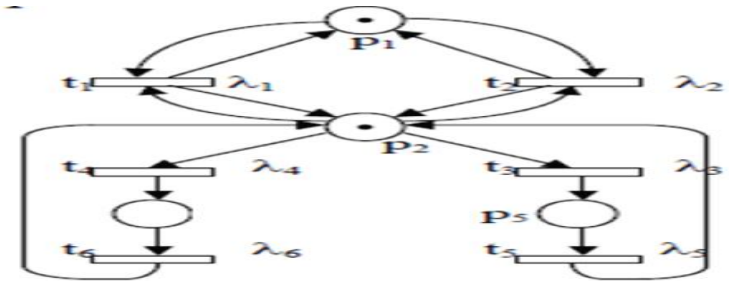


**Figure 4.15** : Réseaux de pétri stochastique.

- **Les réseaux de pétri avec un arc inhibiteur :** Une autre extension des réseaux ordinaires consiste à permettre de tester l'absence de marques dans une place, alors que lors d'un franchissement classique, on vérifie au contraire la présence d'une marque qui est consommée. Lorsqu'une place en entrée est reliée à une transition par un arc inhibiteur, cette transition n'est franchissable que si la place est vide (à ceci peut s'ajouter les conditions sur les autres places naturellement).Lors du franchissement la place en question reste vide [39].

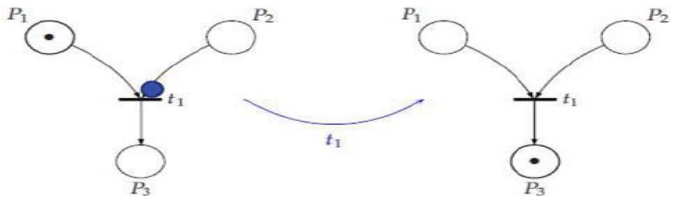


**Figure 4.16** : Un réseau d pétri à arc inhibiteur.

## 4.9 Modélisation Avec les réseaux de pétri

Les Rdps sont principalement utilisés pour la modélisation. beaucoup Les systèmes peuvent être modélisés via Rdp, ce qui peut être très Divers : matériel informatique, logiciels informatiques, systèmes physiques, systèmes sociaux, etc. [45]. Dans les sections suivantes, nous donnerons quelques exemples de modélisation de problèmes informatiques et mathématiques à l'aide de Rdp.

### 4.9.1 Parallélisme

Dans le Rdp représenté par la figure suivante le franchissement de la transition T1 met un jeton dans la place P2 (ce qui marque le déclenchement du processus 1) et un jeton dans la place P2 (ce qui marque le déclenchement du processus 2).

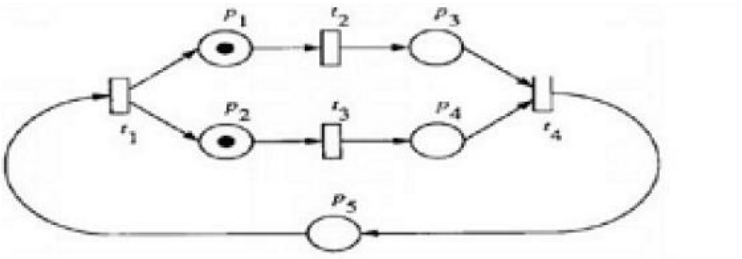


**Figure 4.17** : parallélisme dans les Rdp.

### 4.9.2 Synchronisation

Les Rdp ont été utilisés pour modéliser une variété de mécanismes de synchronisation, y compris les problèmes de l'exclusion mutuelle, producteur/consommateur, lecteurs/
Écrivains …etc [45].

**Exemple:**

Ce problème peut être résolu par un Rdp comme celui de la figure suivante, la place m représente la permission d'entrer dans la section critique. Pour qu'un processus entre dans la

section critique, il doit avoir un jeton dans p1 ou p2 pour signaler qu'il souhaite entrer dans la section critique et il doit y avoir un jeton dans la place m pour avoir la permission d'entrer dans la section critique. Si les deux processus souhaitent entrer simultanément, les transitions t1 et t2 seront en conflit. Seulement l'une d'elles peut être franchie. Le franchissement de t1 désactivé t2, ce qui oblige le processus 2 à attendre la sortie de processus 1 de sa section critique et à mettre un jeton a la place m [45].

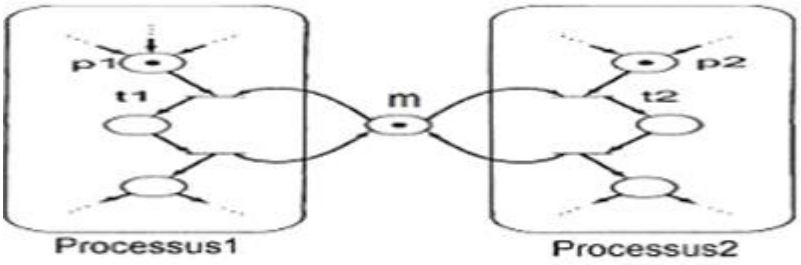


**Figure 4.18** : Exclusion mutuelle.

### 4.9.3 Calcule de flux de données

Un calcule de flux de données est celui dans lequel les instructions sont activées pour l'exécution par l'arrivée de leurs opérandes, ils peuvent être exécutées simultanément. Les Rdp peuvent être utilisés pour représenter non seulement le flux de contrôle, mais également le flux de données. Les jetons dénotent les valeurs de données en cours ainsi que la disponibilité des données. Les Rdp de la figure suivante représente un calcule de flux de données de la formule suivant : [35].

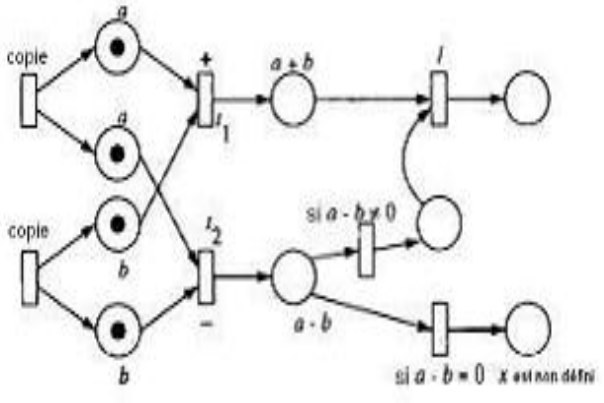


**Figure 4.19** : Exemple d'un calculé.

## 4.10 Principales propriétés des réseaux de pétri

- **Vivacité :** Un Rdp G est dit vivant pour un marquage initial M0 si pour tout marquage accessible M R(M0), il est possible de trouver une séquence de franchissements S qui permet de franchir n'importe quelle transition de G en partant de M. Un blocage correspond à un marquage ou aucune transition n'est validée. La propriété de vivacité assure donc le non-blocage [46].
- **Réseau borné :** Un Rdp est k-borné pour un marquage initial M0 si quel que soit marquage M R(M0) et quelle que soit la place p P, M(p)<=k< ,ou k est un entier naturel.

Un Rdp est dit sauf s'il est 1-borne. Du point de vue des systèmes de production, cette propriété garantit qu'il n'y aura pas d'accumulation d'en cours dans le système [42].

- **Consistance et réversibilité :** Un Rdp est dit consistant s'il existe un marquage initial M0 et une séquence de franchissements S contenant au moins une fois chaque transition, tel que M0[S > M0]. Un Rdp est réversible pour un marquage initial M0 si quel que soit le marquage accessible M $\in$ R(M0), il existe une séquence de franchissements S tel que M[S > M0], en d'autres termes dans un Rdp réversible il est toujours possible de revenir au marquage initial [42].
- **Persistance :** Un Rdp est persistant pour un marquage initial M0 si quel que soit le marquage accessible MR(M0) et quel que soit le couple de transitions validées pour ce marquage, le franchissement d'une des deux transitions n'empêchent pas le franchissement de l'autre. Un Rdp persistant ne nécessite pas que soient prises des décisions pour la résolution des conflits car l'ordre de franchissement ne conduira pas à annuler une possibilité de franchissement. Pour cette raison un Rdp persistant est aussi appelé un Rdp a décision. Un Rdp sans conflit est toujours persistant [42].
- **Invariants :** Les invariants permettent de caractériser le réseau vis-à-vis de certaines propriétés des marquages accessibles et des transitions franchissables. Les notions d'invariants sont importantes car elles permettent d'identifier les parties du réseau pour lesquelles il y a une conservation du nombre de jetons ou pour lesquelles les séquences de franchissement des transitions conduisent à nouveau au marquage de départ [42].

## 4.11 Utilisation des réseaux de pétri

Dans certains cas, les réseaux de pétri ordinaires ne peuvent exprimer toutes les propriétés que nous voudrons modéliser, et de ce fait, des extensions s'avèrent utiles afin de pallier à ces insuffisances. Parmi les extensions les plus utilisées, nous citons [47].

- **Les réseaux de pétri généralisés :** Un réseau de pétri généralisé est un réseau de pétri dans lequel des poids (nombres entiers strictement positifs) sont associés aux arcs.
- **Les réseaux de pétri temporisés :** C'est une extension temporelle des réseaux de pétri.
- **Les réseaux de pétri colorés : Dans** un réseau de pétri coloré, on associe une valeur à chaque jeton.
- **Les réseaux de pétri continus :** Le nombre de jetons dans un réseau de pétri continu est un réel positif. Le franchissement s'effectue comme un flot continu en introduisant la notion de vitesse traduite par le nombre de marques franchies pendant une unité de temps.

Quelques utilisations des réseaux de pétri dans les différents domaines, non seulement informatiques, ont été résumées dans la table .

**Tableau 2**: Exemples d'utilisation des réseaux de Petri

| | |
|---|---|
| Réseau de Petri ordinaire | Modélisation dessystèmes logiciels.<br>Modélisation desprocessus d'affaires.<br>Gestion des flux.<br>Programmation concurrente.<br>Génie de la qualité.<br>Diagnostic. |

| | |
|---|---|
| Réseau de Petri généralisé | Gestion des flux complexes.<br>Modélisationde chaînes logistiques.<br>Utilisation pour les techniques quantitatives. |
| Réseau de Petri temporisé | Gestion du temps.<br>Modélisation d'attentes. |
| Réseau de Petri coloré | Modélisation des systèmes de collaboration. |
| Réseau de Petri continu | Modélisation des réactions chimiques. |

## 4.12 Avantages et inconvénients des réseaux de pétri

Le formalisme des réseaux de Petri présente les avantages suivants :

- **Définition formelle :** Cette caractéristique efface toute ambigüité dans la spécification, car chaque modèle possède une sémantique bien définie.
- **Les réseaux de Petri sont exécutables :** Il existe des programmes construits sur la définition formelle de la notation qui permettent d'interpréter les modèles en réseaux de Petri et de simuler le fonctionnement du système en cours de spécification. Ceci permet une vision dynamique du système.
- **Expression puissante :** Les réseaux de Petri sont adaptés pour décrire des comportements complexes, réactifs ou concurrents.
- **Support de vérification :** Les réseaux de Petri disposent de nombreuses techniques de vérification automatique des propriétés génériques du système, comme la vivacité, ou des propriétés spécifique, comme l'existence d'invariants.
- **Représentation graphique :** cette qualité facilite l'interprétation et la compréhension des modèles.

Malgré ces multiples qualités, les réseaux de Petri souffrent de certains points négatifs :

- **Manque de structuration :** Plus le système est complexe et plus la taille du modèle produit est importante, et plus leur maîtrise est compliquée.
- **Structure de données :** Les réseaux places/transitions ne permettent pas de décrire la structure des données manipulées par le système.

Ces inconvénients ont été étudiés et commencent à se résoudre graduellement depuis le développement de la théorie des réseaux de Petri de haut-niveau.

## 4.13 Conclusion

Dans ce chapitre, nous avons présenté les réseaux de pétri. Nous avons séparé ce chapitre en huit partie principales : Dans la première partie nous avons présenté les concepts de base d'un Rdp , la deuxième partie également présenter l'évolution d'unRdp ,Dans la troisième partie du chapitre nous avons définie et étudié les sous-classes, ensuite la partie quatre nous

avons parlé sur les extensions des Rdp ,la cinquième partie : les méthodes d'analyse, ensuite la sixième : modélisation avec les Rdp, les principale propriété des Rdp dans la partie sept ,et enfin l'utilisation des Rdp. Le chapitre suivantva se concentrer sur la transformation de graphes.

# Chapitre 5
# Implémentation et mise en œuvre.

## 5.1 Introduction

La méthode de Triple Graph Grammaire (Triple Graph Grammar : TGG, Introduit par Andy Schürr) est une tentative de créer une méthode de connexion Différents systèmes/modèles pour certaines règles/normes prédéfinies, donc Les changements de système/modèle conduiront inévitablement à Changements de l'autre côté. TGG contient une grammaire de graphe source, une grammaire de graphe cible et un Deux morphismes correspondant à la grammaire des graphes et la première connexion Syntaxe des graphes de source et de correspondance et deuxième lien Correspondance et syntaxe du graphique cible.

Le formalisme TGG permet de spécifier des transformations de modèle bidirectionnelles. Il existe plusieurs outils TGG dont l'expressivité, l'applicabilité, l'efficacité et les algorithmes de traduction sous-jacents varient considérablement. Dans ce cas, TGG Interpréter [48] est considéré comme le meilleur choix pour la transformation de graphe.

Dans ce chapitre, nous présentons l'environnement d'implémentation des transformations de modèles ainsi que deux études de cas.

## 5.2 Environnement D'implémentation

Un aspect de notre approche implique l'utilisation de représentations et Manipuler le modèle. En effet, dans le monde des modèles, la structure de la représentation est Principalement des graphiques. Ensuite, la transformation du modèle est équivalente à la transformation graphique. Nous avons choisi le cadre de modélisation Eclipse, qui utilise Ecore pour créer et Manipuler le modèle, s'adapter à l'environnement technique de la modélisation et Conversion de modèle.

L'implémentation des transformations de modèles met en jeu trois étapes :

- la première étape permet la transformation des métamodèles de l'espace technique EMF en grammaires de graphes dans l'espace technique TGG.
- la seconde étape consiste à spécifier les règles de transformation et la grammaire de graphe de correspondance.
- la troisième étape sert à générer le moteur de transformation et exécuter ce dernier sur un modèle source pour avoir un modèle cible.

La figure 37 illustre ces trois étapes. Dans ce qui suit, nous présenterons l'environnement d'implémentation que nous avons utilisé pour notre recherche.

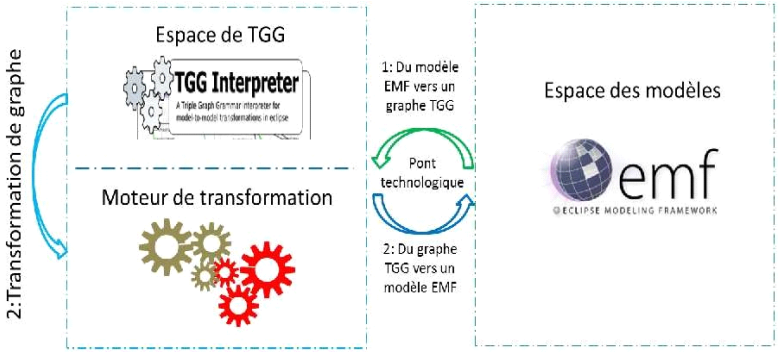


**Figure 5.1**: Principe de mise en œuvre de transformation de modèles

## 5.3 La plateforme Eclipse

Un aspect de notre approche implique l'utilisation de représentations et de modèles de manipulation. En effet, dans le monde des modèles, la structure de la représentation est avant tout un graphe. Ensuite, la transformation du modèle est équivalente à la transformation graphique. Nous avons choisi le cadre de modélisation Eclipse, qui utilise Ecore pour créer et manipuler des modèles, en s'adaptant à l'environnement technique de modélisation et de transformation de modèles.

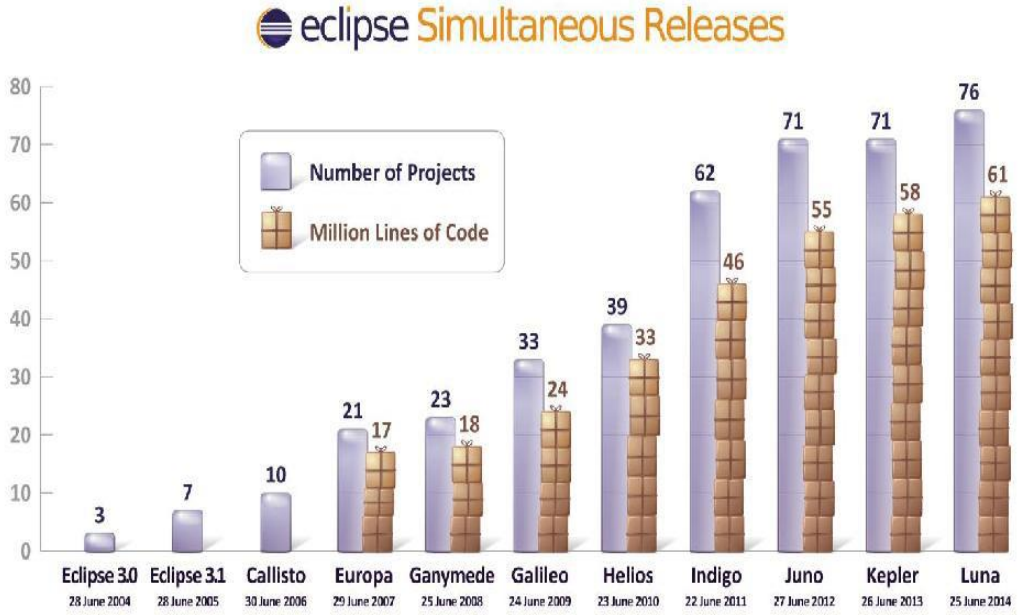


**Figure 5.2** : Croissance de la plateforme Eclipse dans le temps [49].

Figure 5.2 Compilé par Holger Voorman, montrant le nombre d'éléments et de lignes de code Sous diverses distributions Eclipse au fil des ans. Avant 2012, la plateforme Eclipse avait Sorti en version 3.x, publié sous plusieurs distributions dont Helios (3.6), Indigo (36) sera utilisé plus tard pour mettre en œuvre nos propositions, etc. Depuis 2012, La version de la plate-forme a été mise à niveau vers 4.x et les versions Juno (4.2), Kepler (4.3) et Luna (4.4) ont été publiées en 2014.

### 5.3.1 Eclipse Modeling Framework

Cadre de modélisation Eclipse Éclipse Mödling Framework (EMF) est un Framework de modélisation Java et un outil de génération de code pour la création d'applications basées sur des modèles. À partir du modèle de spécification décrit dans XMI, EMF fournit des outils et un support d'exécution pour la création de classes Java. De plus, EMF permet de stocker des modèles sous forme de plusieurs fichiers liés. Les modèles peuvent être définis avec des documents Java, UML, XSD, XML puis importés dans EMF. EMF, en revanche, ne propose pas d'éditeur graphique pour la modélisation. Plus important encore, l'EMF fournit la base de l'interopérabilité avec d'autres outils ou applications basés sur l'EMF. EMF utilise Ecore, qui est un langage de méta-modélisation graphique et textuel défini par IBM.

EMF permet également le développement et l'intégration rapides de nouveaux plugins Eclipse se compose d'un ensemble de blocs de construction appelés plugs. Parmi eux [50]

- Metamodel Ecore qui est un canevas pour les classes décrivant des modèles EMF et manipuler des référentiels de modèles.
- EMF et manipuler des référentiels de modèles.
- Le Modèle de génération Gen Model qui permet de personnaliser la génératio Java.
- Java Emitter Template qui est un moteur de template générique.
- Java Merge qui est un outil de fusion de code Java.

### 5.3.2 Ecore

Ecore [55] est un langage de méta modélisation graphique et textuel défini par IBM utilisé dans la modélisation EMF Eclipse. Il est utilisé pour définir des méta modèles, et le langage OCL est intégré pour écrire des contraintes afin de vérifier la compatibilité des modèles avec leurs méta modèles.

Pour créer les métas modèles à partir d'Ecore, EMF propose une forme arborescente (l'extension du méta modèle créé est alors « .ecore ») et une forme graphique (l'extension du méta modèle créé est alors « .ecorediag ») [50]. Nous choisissons d'utiliser ce langage à base des critères suivants :

Ecore [51] est un langage graphique et textuel. La représentation graphique du langage permet une manipulation directe des concepts, sans passer par une formalisation textuelle abstraite et difficilement compréhensible.

Ecore intègre le langage formel OCL, ce qui facilite la mise en œuvre et permet d'écrire des contraintes. Une contrainte est une expression qui peut être attachée à n'importe quel élément du méta modèle.

### 5.3.3 TGG interpreter

Approche Triple Graph Grammars (Triple Graph Grammars : TGG), introduite par Andy Schürr [52] est une technique utilisée pour définir la relation entre deux types des modèles. Et ensuite transformer le modèle d'un type à un autre pour calculer correspondance entre deux modèles existants. Ainsi, pour être cohérent entre deux types de modèles comme spécifié dans le TGG. Lorsque l'un des modèles est changé, le modèle cible peut être modifié, c'est-à-dire transformation ou synchronisation : appliquer progressivement [53] [54].

Le principal avantage de TGG est la possibilité de définir facilement des transformations. déclaratif, avec exécution dans les deux sens de transformation (i.e. définition est bidirectionnel). Dans ce contexte, l'interpréteur TGG [55] est considéré comme le meilleur choix pour transformer des graphiques.

L'interpréteur TGG a été développé pour transformer les modèles TGG et est un outil pour : mise à jour incrémentale résultant de la comparaison des normes TGG et OMG QVT bidirectionnel (query/view/transform) pour la transformation de modèles. ce L'interpréteur TGG est basé sur Eclipse et peut être installé à l'aide du gestionnaire de mise à jour Eclipse. ce La figure 37 montre une capture d'écran de l'éditeur Eclipse TGG.

**Ergonomie :** L'interpréteur TGG est basé sur Eclipse et peut être installé à l'aide du gestionnaire de mise à jour Eclipse. La figure37est une capture d'écran de l'éditeur de règles TGG montrant une syntaxe visuelle spécifique similaire à tous les autres outils

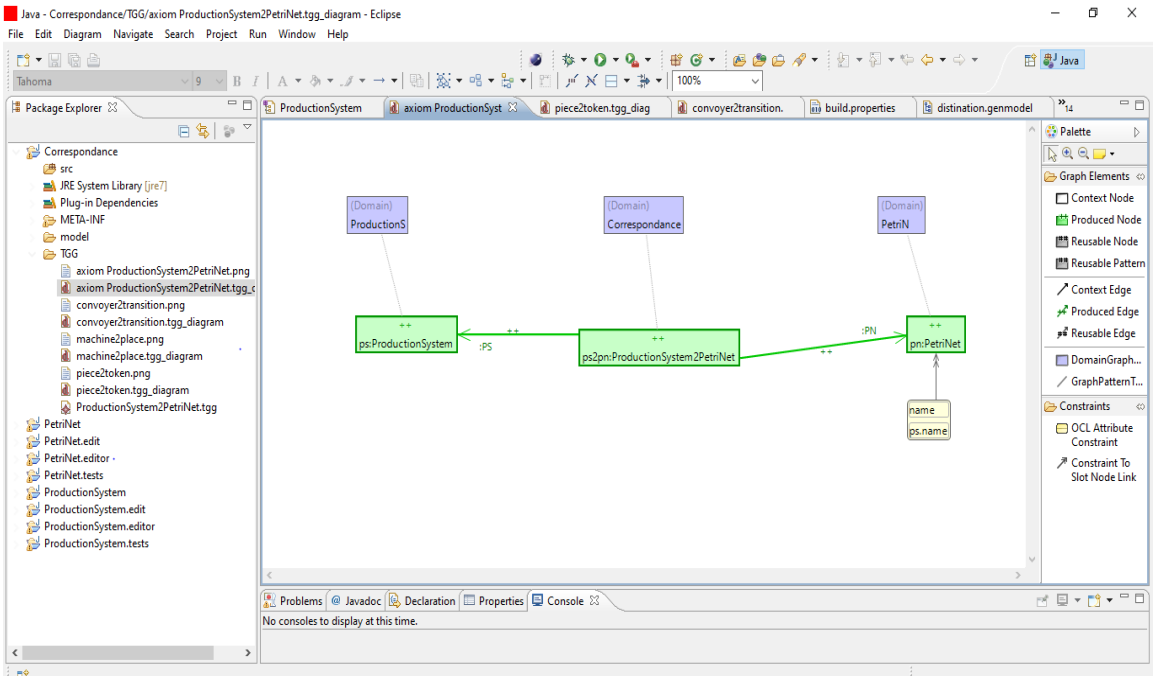


**Figure 5.3** : Éditeur de Règle TGG de TGG Interpréter.

Plusieurs vérifications (par exemple, nœud/arc conformité de type, validité de règle d'héritage, contrôles de syntaxe OCL, etc.) sont réalisées statiquement pour éviter les erreurs de modélisation. Cependant, il n'y a pas de soutien pour l'analyse de paire critique afin d'identifier les règles qui peuvent être contradic- toires. Un éditeur GMF est prévu pour l'édition de règles TGG ainsi que la fonctionnalité de commodité, par exemple, la création de nouvelles règles fondées sur des modèles dans d'autres règles, et la création des nœuds de correspondance sur le champs pour les nœuds de correspondance. TGG Interpreter exécute directement les spécifications TGG sans autre traitement. Il s'intègre dans Eclipse GUI permettant à des transformations sur scène par un clic droit sur un fichier de modèle ou via Eclipse Run configuration. Les transformations peuvent également être exécutées via un appel d'API.

- **Expressivité :** les conditions d'application et les contraintes d'attribut peuvent être formulées dans OCL, mais les relations bidirectionnelles sur les valeurs d'attribut doivent être exprimées sous forme d'affectations dans le sens avant et arrière. Il n'y a pas de fortes restrictions imposées à la structure des règles Les TGG, c'est-à-dire que les modèles n'ont pas besoin d'être connectés de manière lâche, peuvent créer des arcs entre les nœuds de contexte et les nœuds d'ajustement peuvent être joints aux éléments source et cible. L'interpréteur TGG prend en charge des concepts avancés tels que l'héritage de règles, les stéréotypes dans les domaines UML et les nœuds et modèles réutilisables pour conserver les informations dans un cas incrémentiel [55].
- **Propriétés du formulaire :** l'interpréteur TGG prend en charge des fonctionnalités avancées telles que la liaison de bord explicite. De plus, l'interpréteur TGG est interprété

pour effectuer des transformations, qui peuvent être mises à jour dynamiquement pendant les transformations sans recompiler ni calculer quoi que ce soit. Il prend en charge les mises à jour incrémentielles. À propos de l'algorithme de contrôle. L'interpréteur TGG nécessite un nœud de démarrage spécifié, il n'autorise qu'un seul axiome dans TGG et nécessite un comportement fonctionnel pour éviter les retours en arrière. Un seul "bord". L'étape suivante consiste à ne considérer que ces éléments, en limitant la recherche des règles applicables à ceux qui en ont besoin. Le processus se termine lorsqu'il n'y a plus de règles à exécuter.

Les noeuds de ce domaine sont les classes de ce métamodèle, il existe différents noeuds de règle de transformation : noeud de contexte, noeud de production, les contraintes et les noeuds réutilisables [55].

- **nœud de contexte :** Parfois, seule une partie du modèle est pertinente et doit être être transformé. Ensuite, les règles du TGG devraient être conçues pour les parties indispensables à la transformation. Si des parties d'un modèle moins pertinent affectent la transformation, ils peuvent apparaître comme des nœuds contextuels dans le règlement TGG. La représentation graphique du nœud de contexte est présenté dans une boîte avec un contour noir. Les nœuds de contexte doivent être mis en correspondance avec des objets de modèle traités précédemment. Cela signifie que le TGG, jusqu'à présent, spécifiez la grammaire complète pour toutes les parties du modèle. D'autre part, le TGG permet également la formulation d'un grammaire partielle sur le modèle. Là, les nœuds de contexte ne devraient pas doit d'abord être traité par une autre règle.
- **Nœud de production :** les nœuds de production sont affichés comme cases vertes avec une bordure verte et l'étiquette "++". Il y a des arrêts de production affichés sous forme de flèches vert foncé marquées "++". ce Les nœuds de production ne doivent pas correspondre à un nœud qui existe déjà. Si nous trouvons ces éléments dans le domaine source, les éléments de modèle cible et les éléments de modèle la correspondance peut être formée selon la règle. Tous les objets créés sont liés à nœuds de contexte de règle. Par conséquent, l'objet modèle ne peut pas être affiché comme nœud de production une seule fois. Nous l'appelons sémantique de liaison unique pour les nœuds de production.
- **Nœuds réutilisables :** notez que les types de composants ne doivent pas Généré, mais peut être réutilisé encore et encore à partir de nœuds déjà existants a les propriétés requises. Par conséquent, nous appelons ces nœuds "nœuds Réutilisable". Les graphiques sont grisés et Sémantique Les nœuds réutilisables sont qu'ils peuvent être régénérés ou réutilisés quelconque. Le gris reflète le fait que chaque arc réutilisable peut être Soit noir (nœud à gauche de la règle TGG, c'est-à-dire qu'il est réutilisation) ou vert (le nœud à droite de la règle TGG, c'est-à-dire qu'il est nouvellement généré), qui peut être sélectionné à chaque application de la règle. Nous pouvons utiliser la règle exponentielle TGG à la place. Mais, le concept de noeuds réutilisables permet de réduire le nombre et la complexité des règles de TGG. En plus si l'exemple est complexe, les règles sans noeuds réutilisables peuvent générer un désordre. Les noeuds réutilisables nous permettre d'avoir plus de règles simples et de concentrer sur l'essentiel des modèles pertinents.
- **Contraintes :** les contraintes réelles dans le nœud Contraintes peuvent être Une expression OCL qui fait référence à l'objet auquel elle est attachée. la limite est Juste

besoin de mettre en œuvre leur conversion ou synchronisation plus efficace. Contraintes OCL en jaune et doit être lié avec un autre nœud.

### 5.3.4 Architecture d'implémentation de transformation de modèles

Les transformations de modèle définies dans IDM nécessitent un environnement d'exploitation tel qu'EMF. Pour effectuer des transformations de modèles basées sur des graphes, nous avons dû utiliser l'outil TGG Interpreter. Notre application est responsable de la construction de la grammaire du TGG (source, cible et correspondance) à partir du métamodèle EMF, et utilise le moteur de recherche TGG pour construire le modèle EMF cible à partir du modèle EMF source. Le mécanisme est présenté dans la Figure 38 sous la forme de deux champs EMF et TGG spatiaux. Dans l'espace EMF nous avons des métamodèles sous forme de fichiers .ecore et des modèles sous forme de fichiers .xmi. Les métamodèles seront traduits en grammaires et donc en règles de production dans l'environnement TGG. A partir de ces règles de production, un moteur de transformation est dérivé.

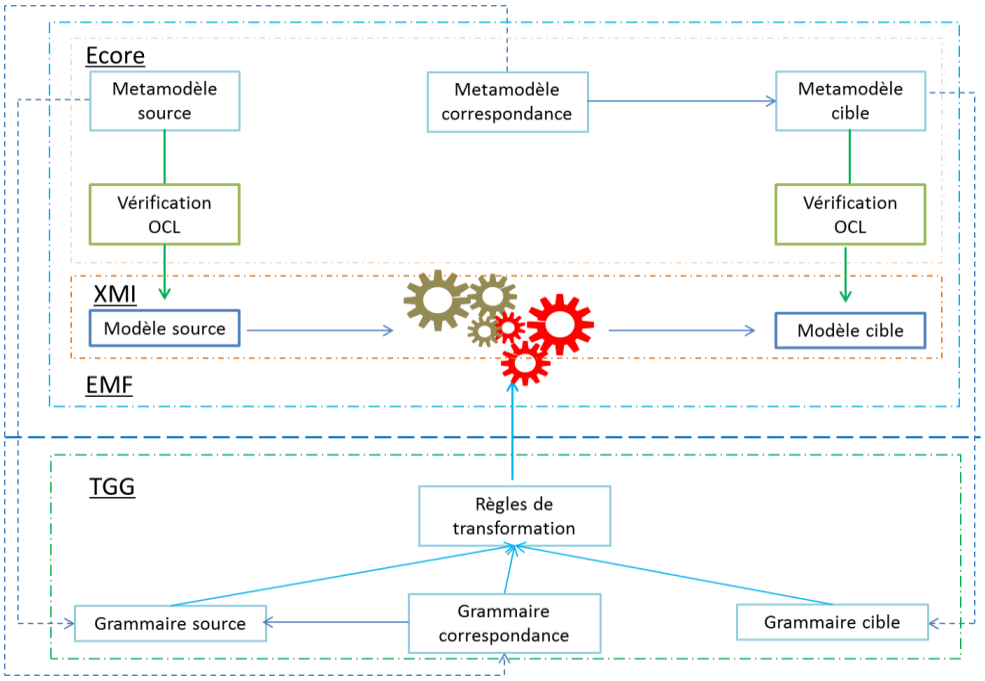


**Figure 5.4 :** L'architecture de TGG Interpréter.

## 5.4 Études de cas

Pour bien comprendre notre contribution aux transformations de modèles basées sur la grammaire des graphes, nous fournissons quelques études de cas pour illustrer le processus de spécification des transformations et leur mise en œuvre.

### 5.4.1 Les méta modèle

Dans ce qui suit, nous présentons les métamodèles source et cible utilisés ainsi celui des correspondances.

- **Méta modèle de system de production :** La figure 5.5 montre le méta modèle d'un system de production. Le méta modèle indique qu'un système de production se compose :

**machine :** cette class représenter un machine elle possède attributs < *name* >. De type String et les attributs <*jobtime*>et <*long*>de type double et <*Ability*> de type entier.
**convoyer :** cette class représenter un convoyer elle possède attributs < *name* > msource et *mdistination* de type String, les attributs <*jobtime*>et <long> de type réel, et <*Ability*> de type entier.
**Opération :** cette class représenter un Opération elle possède attributs < *name* > de type String
**Operator :** cette class représenter un Opérateur elle possède attributs < *name* > de type String.
**Pièce :** cette class représenter un Pièce elle possède attributs < *nbr* > de type entier .
**Stock :** cette class représenter un stock elle possède attributs < *name* > de type String, <*jobtime*> de type réel et <*Ability*> de type entier.

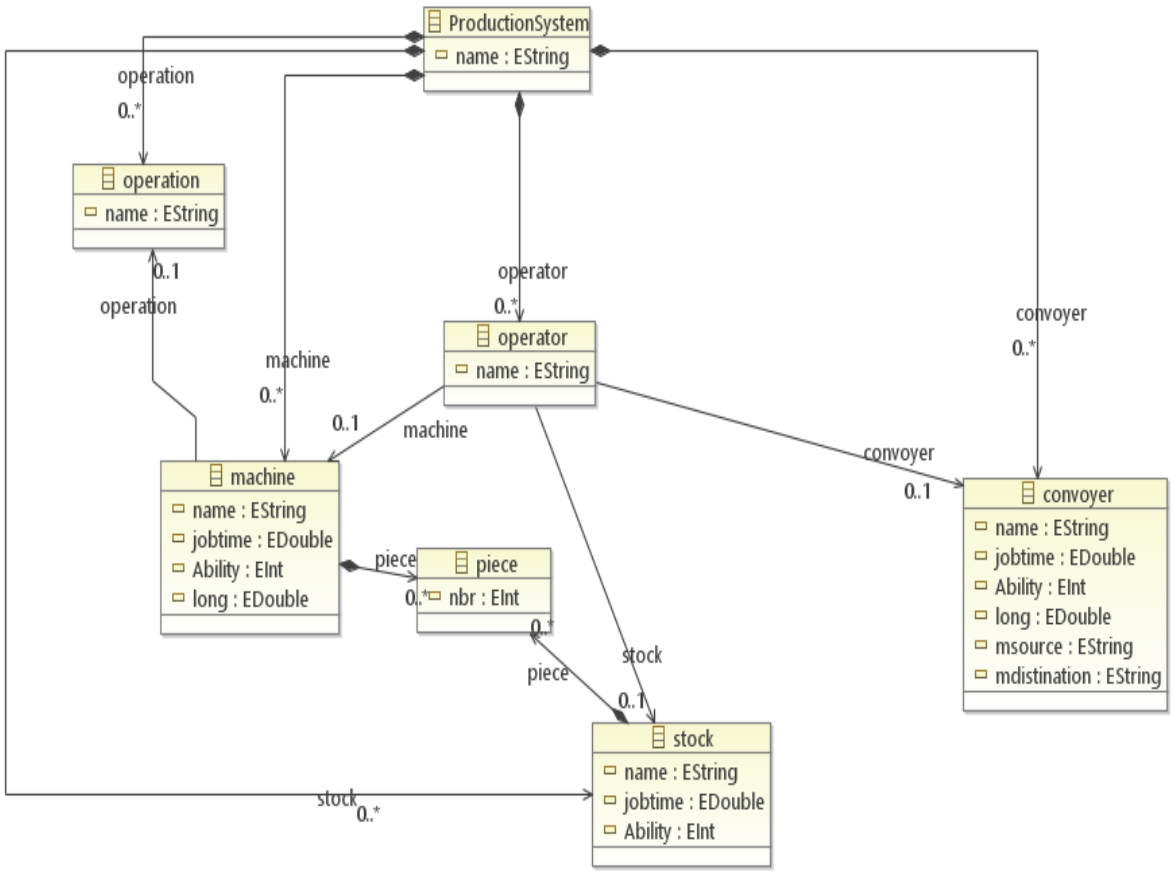


**Figure 5.5**: Méta modèle de system de production

- **Méta-modèle du réseaux de pétri :** Le méta-modèle des réseaux de pétri présenter dans la Figure 5.6 contient quater Classes :

**place :** cette classe représente les places. Elle possède attributs < *name* >. de type String, <*jobtime*> et <long> de type Double et l'attribut <*Ability*> de type entier.

**transition :** cette classe représente les transitions, elle possède l'attribut < *name* ><*psource*>et <*pdistination*>de type String, et les attributs <*jobtime*> et <long> de type réel, et l'attribut <*Ability*> de type entier.
**token :** cette classe représente les jeton. Elle possède attributs < *nbr*> de type entier.

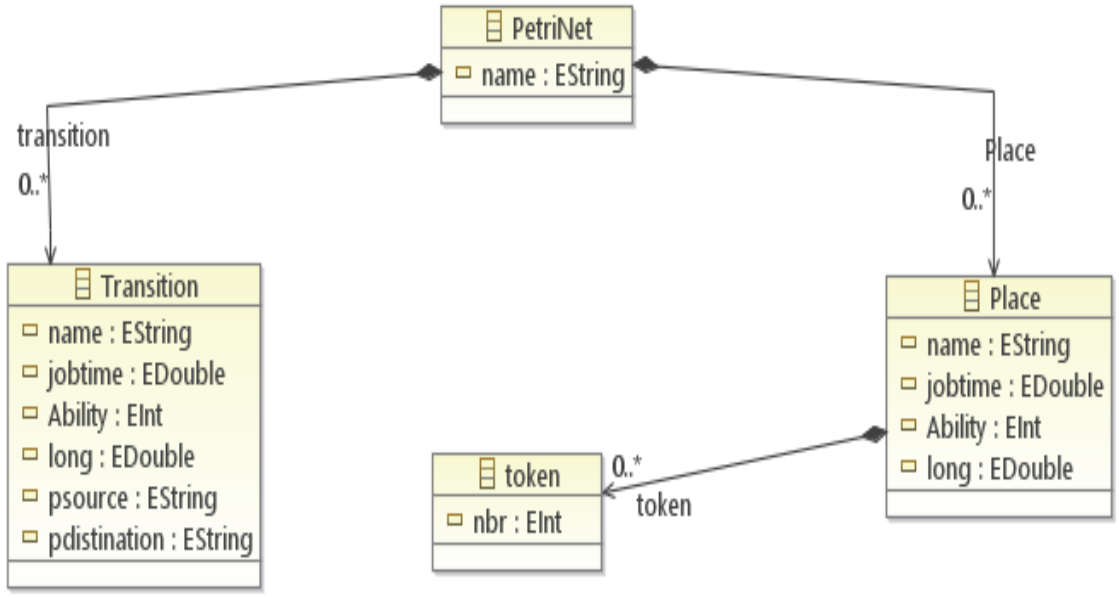


**Figure 5.6 :** Métamodèle de réseaux de pétri .

- **Métamodèle des correspondances :** La figure 43 illustre le métamodèle de correspondances de l'exemple étudié où *production2petrinet* présente la correspondance entre un system de production. *machin2place* représente la correspondance entre un machin et une place, et *stock2place* présente la correspondance entre un stock et un place

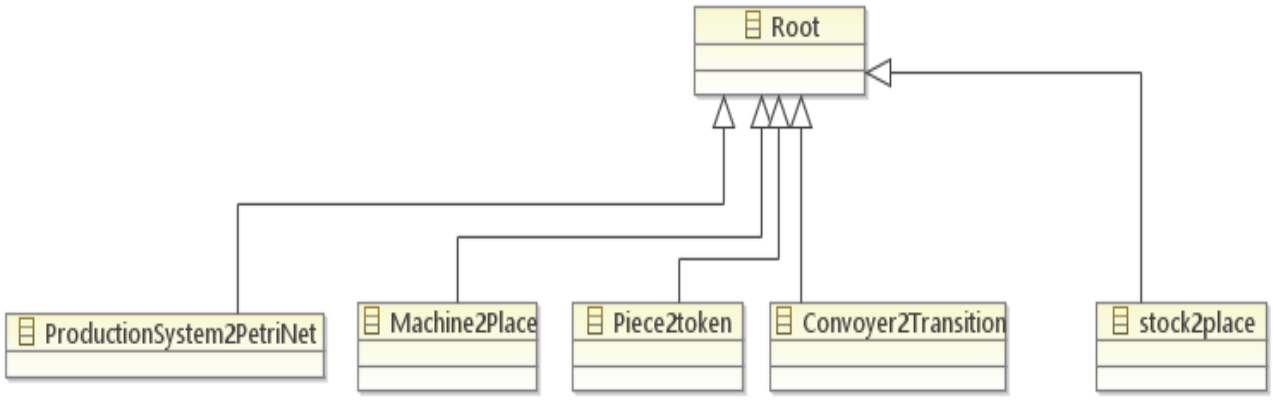


**Figure 5.7 :** Méta modèle de correspondances.

## 5.4.2. Génération des outils pour la transformation

En se basant sur le méta-modèle on peut générer un outil qui nous permettra de créer des exemples de system de production (des instances) avec les étapes suivantes :

Nous allons créer un projet EMF (*RDp*), puis un diagramme Ecore avec leur nom (Rdp) dans le dossier (Model).

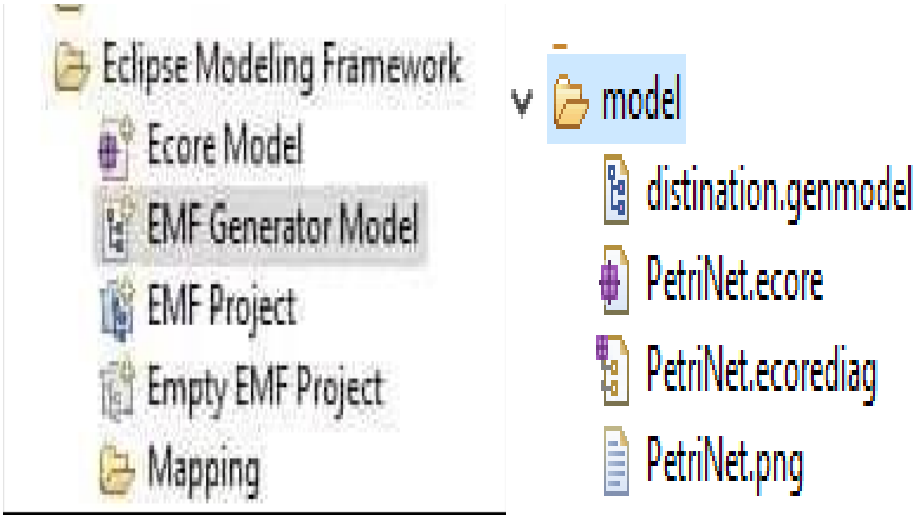


**Figure 5.8 :** Création d'un projet EMF et un diagramme Ecore

Les éléments de notre méta-modèle de réaux de pétri sont affichés dans le fichier (pétrinet .ecore) (figure 5.8).

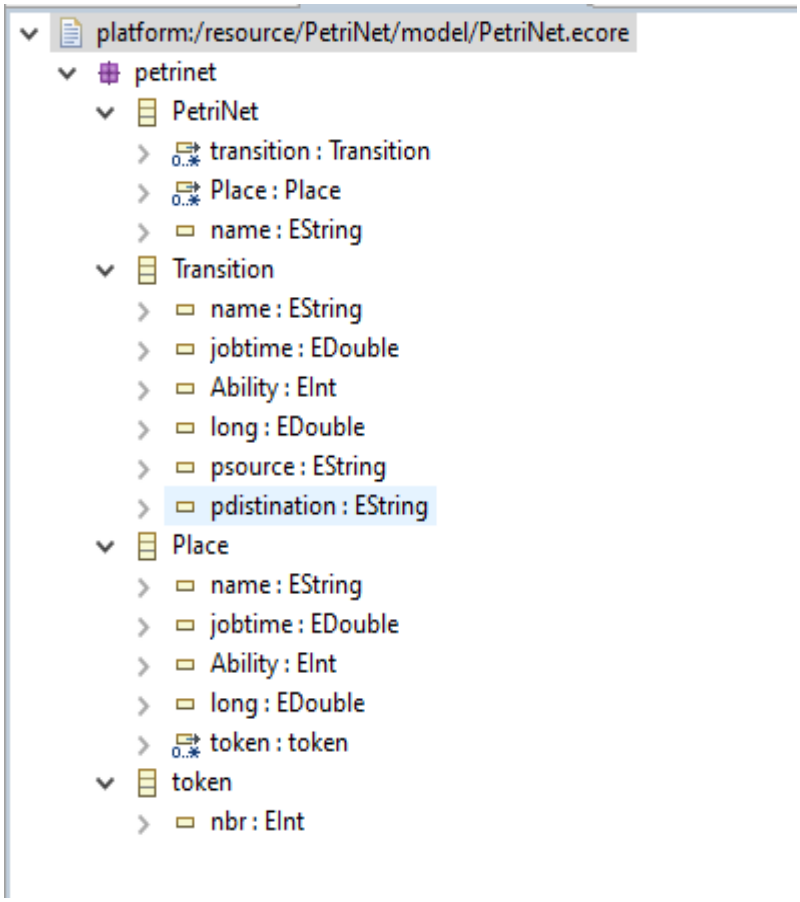


**Figure 5.9:** Les éléments de reseau de petri .

Nous ferons de même pour le réseaux de pétri, nous créerons un projet (Empty EMF Project) nommé (*petrinet*), dans le dossier model nous représenterons le schéma de méta-modele cible (*petrinet.ecorediag*), puis le fichier (*petrinet.ecore*) sera automatiquement affiché, et qui contient les éléments de réseaux pétri.

Dans le projet EMF nommé (*Correspondence*), dans le dossier model nous allons présenter le schéma de méta-model de correspondante (*Correspondence.ecorediag*),

le fichier (*Correspondence.ecore*) s'affichera automatiquement, mais nous devrons saisir les attributs et leur type pour chaque règle de transformation.

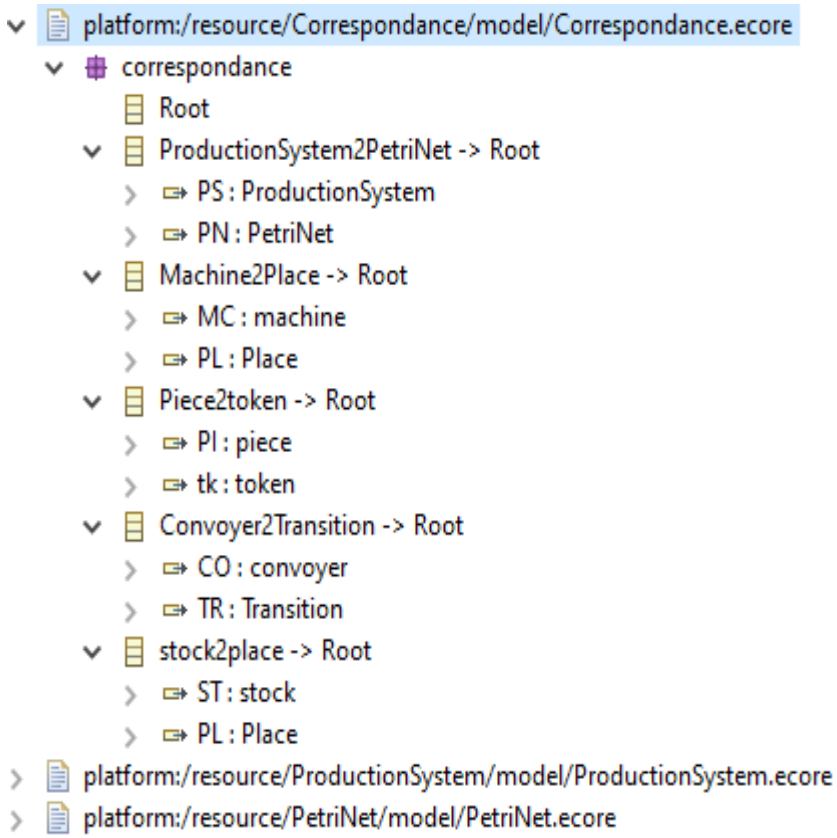


**Figure 5.10 :** Correspondence.ecore.

### 5.4.3 Définition de règles de TGG"

L'idée générale dans la définition des règles de transformation est de spécifier des règles TGG qui transforment chaque classe du métamodèle source en une classe du métamodèle cible.

- **L'axiome :** La plus insignifiante relation entre un system de production et une réseaux de pétri, ça veut dire les deux objets racine correspondent les uns aux autres. Ceci est aussi le point de départ de toute transformation et qui se nomme l'axiome dans TGG. Cet axiome est illustré dans la figure 5.11. Sur le côté gauche, il montre un objet racine system de production, sur le coté droit, il montre un objet racine d'un réseau de pétri, et dans la partie médiane, il montre un nœud de correspondance relative des deux. Comme les règles TGG

n'ont pas de sens et car ils sont graphique donc on est obligé de préciser les domaines ou les nœuds de chaque méta model soient relié, pour system de production et pour réseaux de pétri .

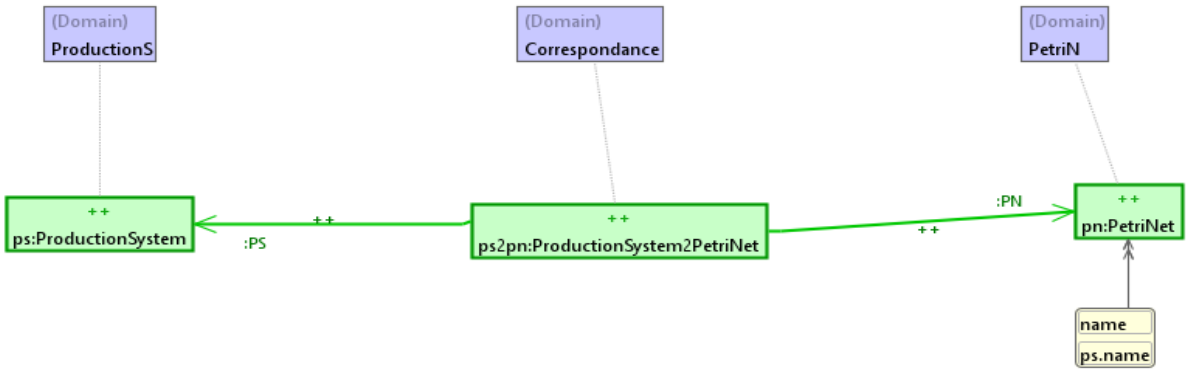


**Figure 5.11:** Présentation graphique de l'axiome.

A partir de cet axiome, nous discutons maintenant les autres constructions qui se produisent dans les system de la production. Nous montrons comment les états correspondants sont créés par conséquence dans les réseaux de pétri.

- **Transformation de "machine" en "place"** L'idée de base de la transformation est que chaque machine d'un system de production se traduit à une place de réseau de pétri. Maintenant, on suppose que cette machine vient d'être ajouté au pièce, qui est indiqué par la couleur verte et l'étiquette supplémentaire ++, on indique également, sur le côté droit, que l'équation correspondante doit être ajouté à la partie de réseau de pétri.

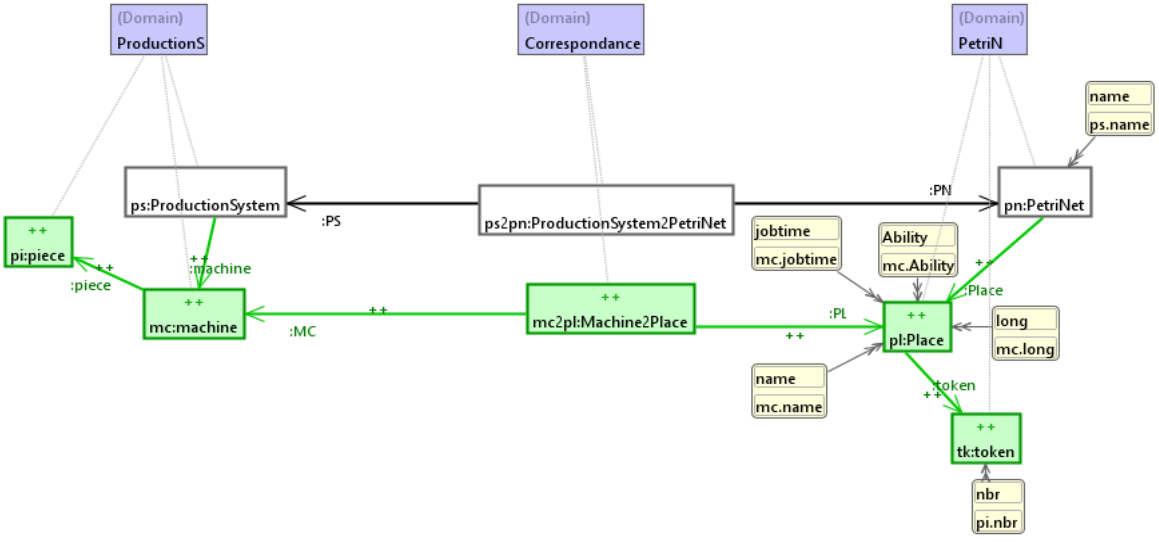


**Figure 5.12:** Présentation graphique de la règle machine2place.

Cette idée peut être exprimée par une règle TGG qui est montré dans la figure 5.12, la conversion de la syntaxe concrète et la syntaxe abstraite selon les métamodèles pour le system de production et le réseau de pétri . Dans la figure12 , il y a un pair de correspondance d'un système production et un réseau de pétri , qu'il suppose qu'ils existent déjà. Par conséquent, ils sont présentés comme des nœuds de contexte noirs (et ne sont pas étiquetés avec ++). En dessous, il montre les pièces nouvellement ajoutées, indiquées en vert et marquées avec ++: Pour le Domain System de production, cela est machine, pour le domaine de réseau de pétri,il est le nouveau place. En outre, il y a un nouveau nœud de correspondance machine2place comme une manifestation de la relation entre ces concepts. Le nœud avec les bords arrondis est une contrainte supplémentaire qui garantit que les valeurs pour les noms de la machine et la place sont les mêmes.

- **Transformation des pièce en token :** Chaque pièce d'une système de production est transformée en une token de réseau de pétri , nous allons créer un nœud de production (*pièce*) et un nœud (*token*) pour la cible, ensuite un nœud de correspondance piece2token qui va correspondre une pièce à une token.

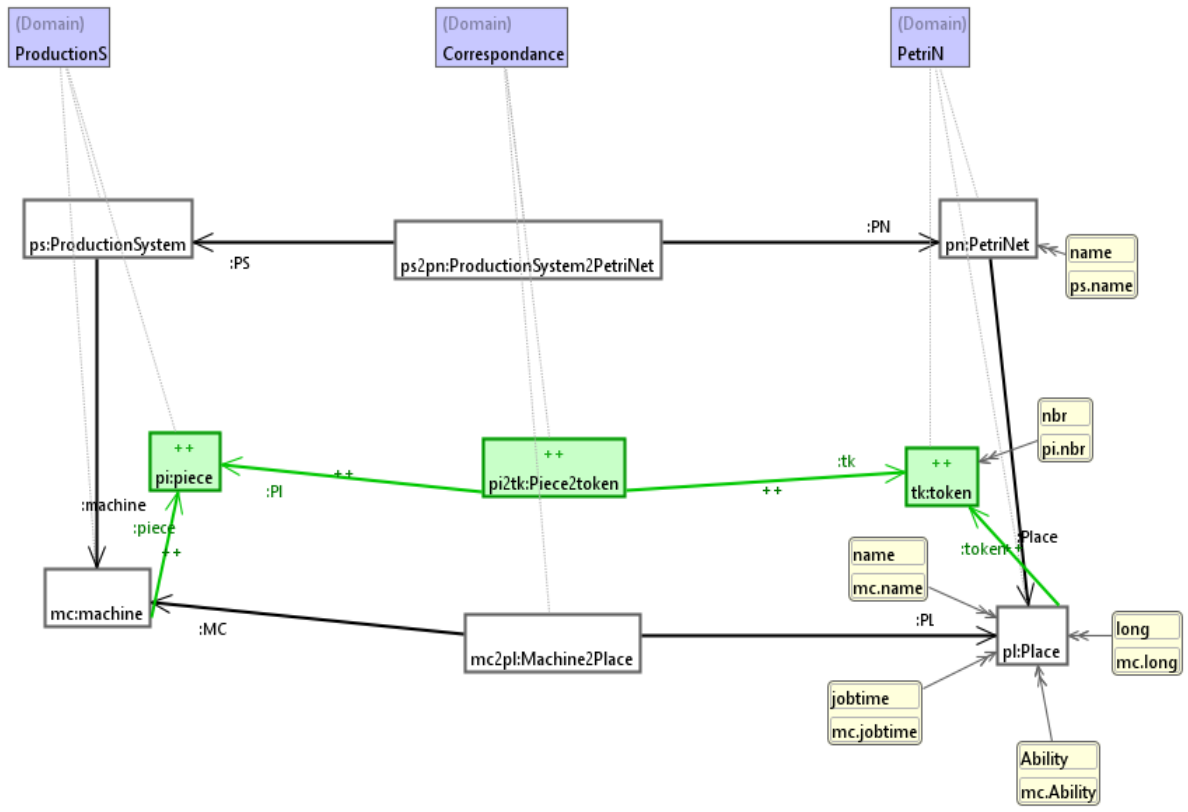


**Figure 5.13:** Présentation graphique de la règle piece2token.

- **Transformation de convoyeur en transition :** La figure 5.14 présente la règle convoyer2transition qui transforme un convoyeur à un transition avec le même nom., dans cette règle on va crée un nouveau nœud de correspondance convoyer2transition qui va correspondre un convoyer à une transition.

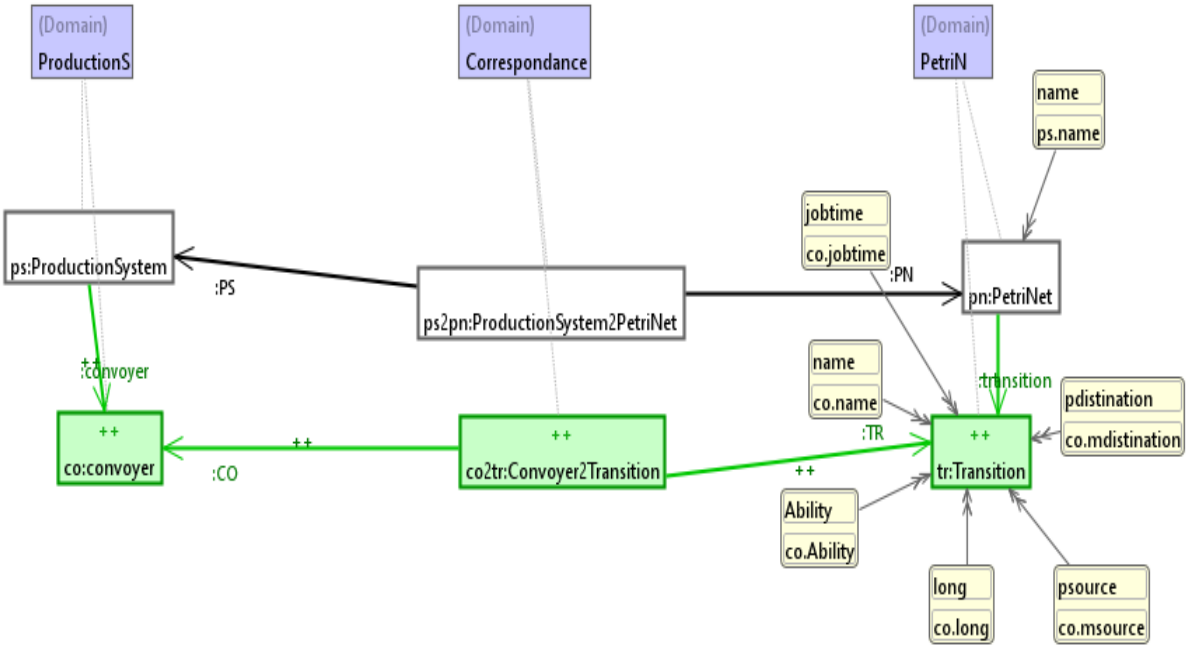


**Figure 5.14 :** Présentation graphique de la règle convoyer2transition.

- **Transformation de stock en place :** La figure 5.15 présente la règle stock2place qui transforme un stock à un place avec le même nom., dans cette règle on va crée un nouveau nœud de correspondance stock2place qui va correspondre un stock à une place.

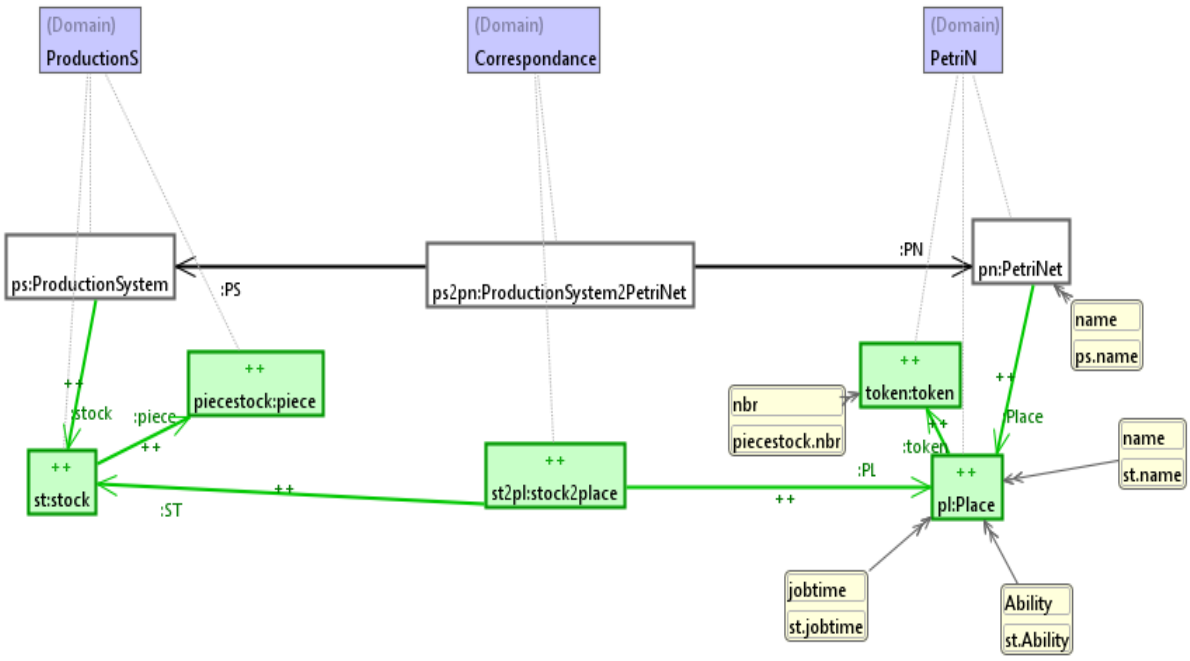


**Figure 5.15:** Présentation graphique de la règle stock2place.

### 5.4.4 Création de genmodel

On passe a l'étape de création de genmodel a partir du modèle défini précédemment, il est possible de générer du code Java dédié à la création des instances de ce *modèle.Nous* allons créer un fichier EMF Generator Model avec un nom (*source.genmodel*) dans le dossier model de projet (*production2system*), ensuite nous allons générer les projets(*production2system.edit/ production2system.editor/ production2system.tests*).

Nous ferons de même pour l*e petrNiet* : génération des projets *.edit/.editor/.tests*

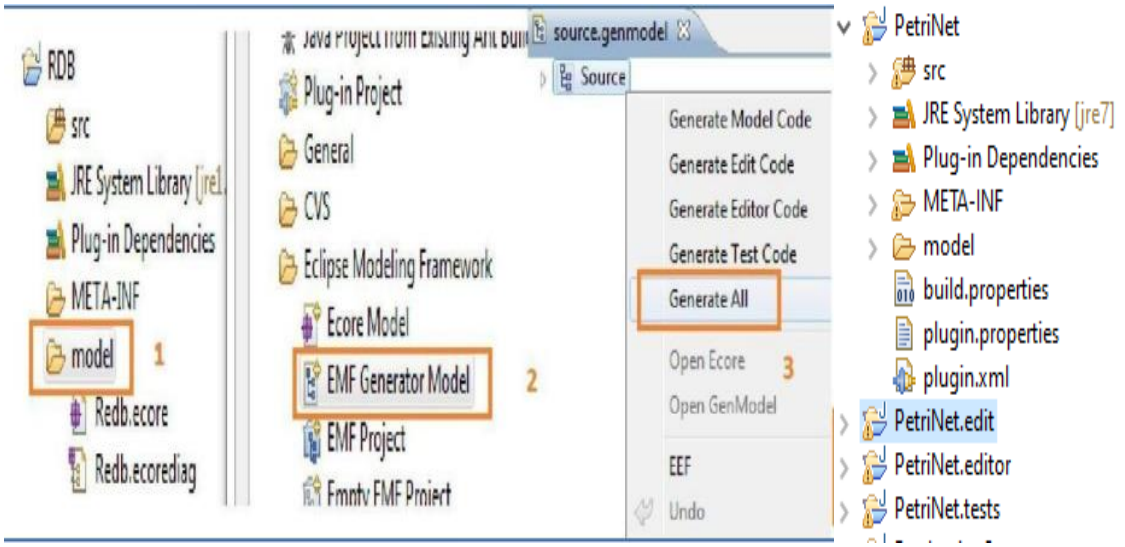


**Figure 5.16:** génération des projets .edit/.editor/.tests

La génération de code nécessite la création d'un modèle de génération, Ce modèle contient des informations dédiées uniquement à la génération et qui ne pourraient pas être intégrées au modèle Les éléments de *source.genmodel / destination.genmodel*.

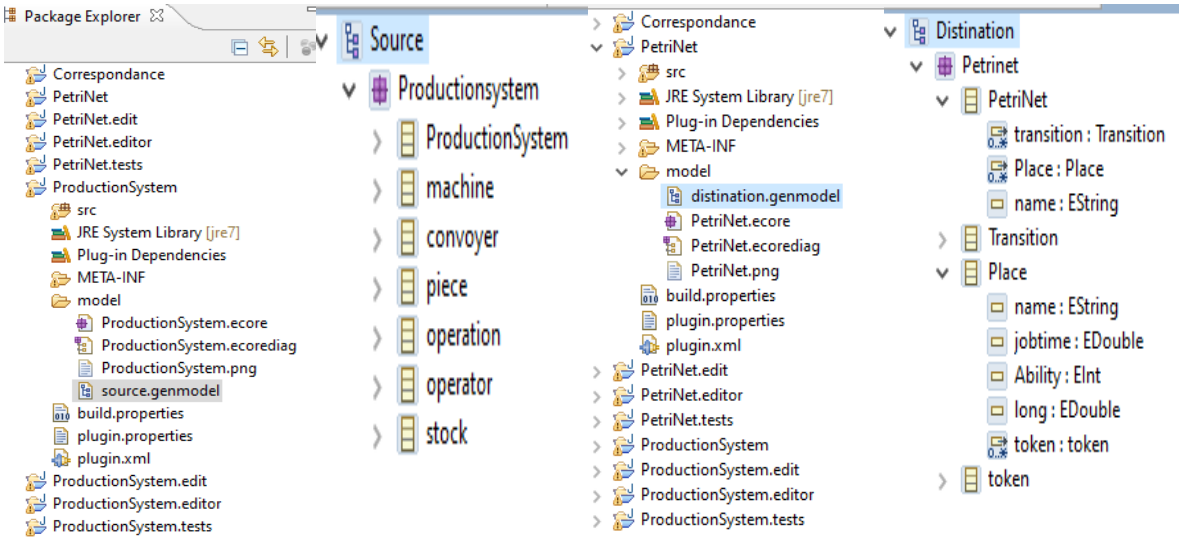


**Figure 5.17:** Les éléments de source.genmodel / destination.genmodel

## 5.5 L'exécution

La phase d'exécution dans TGG Interpréter se résume à crée une configuration du programme de transformation. Une fois ce dernier est créé, il est exécuté sur le modèle source afin d'obtenir le modèle cible et une trace de l'exécution des règles est générée.

### 5.5.1 Le modèle d'entrée

on a prie un exemple simple d'un system de production ou on a une machin qui a un attributs: name qui sont de type String et elle a une a association avec opération, le convoyer a un attributs: name qui sont de type String, le pièce qui a un attributs: nbr qui sont de type entier la figure 5.18 présente la syntaxe concrète de cet exemple.

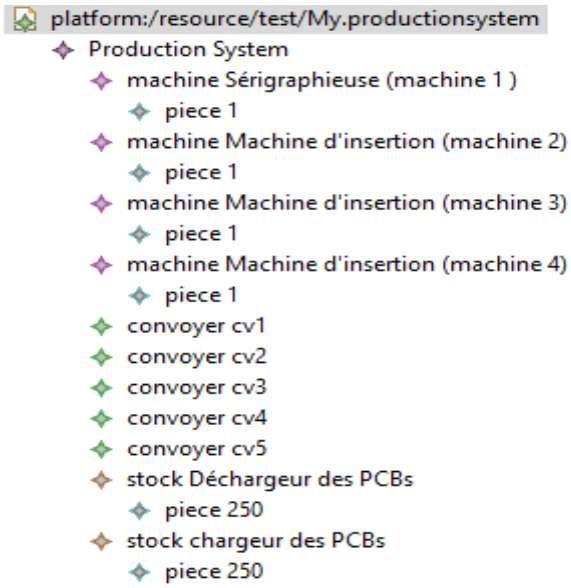


**Figure 5.18 :** Exemple de modèle de system de production.

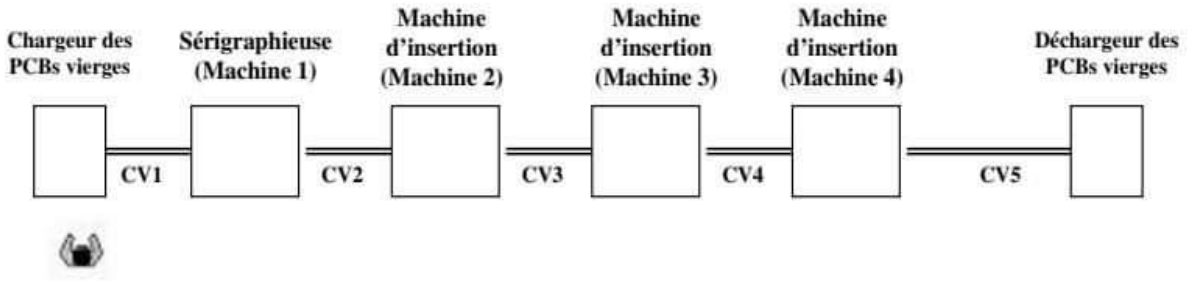


**Figure 5.19 :** modèle de system de production.

### 5.5.2 Appliquer les règles

On va lancer une seconde instance d'Eclipse appliquée au projet source (*prodectionsystem*), puis on va créer un projet appelé (test) contenant le fichier (exemple1) pour réaliser un

exemple du modèle source qui à son tour se transforme en modèle cible (voir figure 5.19), dans le dossier (test) on va créer un fichier (*exemple1.interpreterconfiguration*) avec lequel nous faisons le processus de transformation et une fenêtre de fin de transformation sera afficher (voir figure 5.20).

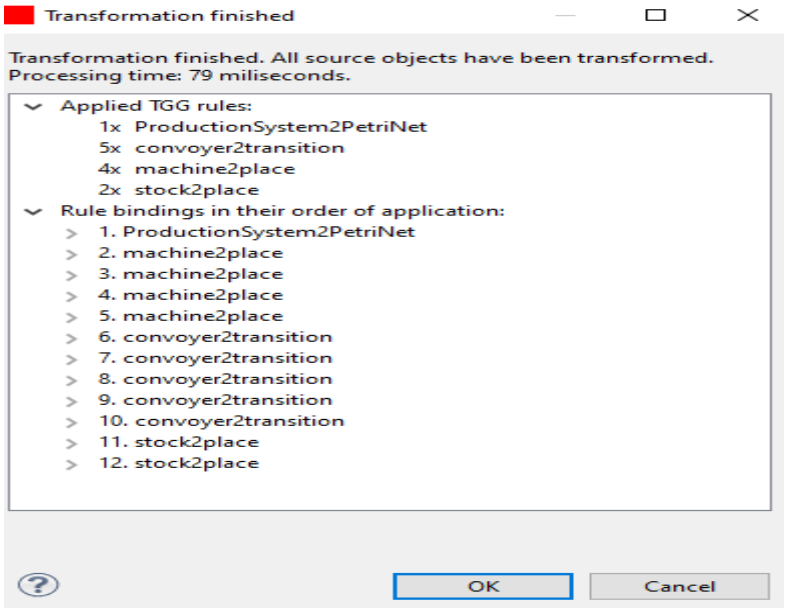


**Figure 5.20** : Fin de transformation.

## 5.5. 3 Le modèle de sortie

Après avoir exécuter notre moteur de transformation on a eu un réseaux de pétri qui contient un place qui a un attributs: name qui sont de type String et un transition qui a un attributs: name qui sont de type String et un arc qui a un attributs: nbr qui sont de type entier la figure 53 présente le modèle cible de la transformation.

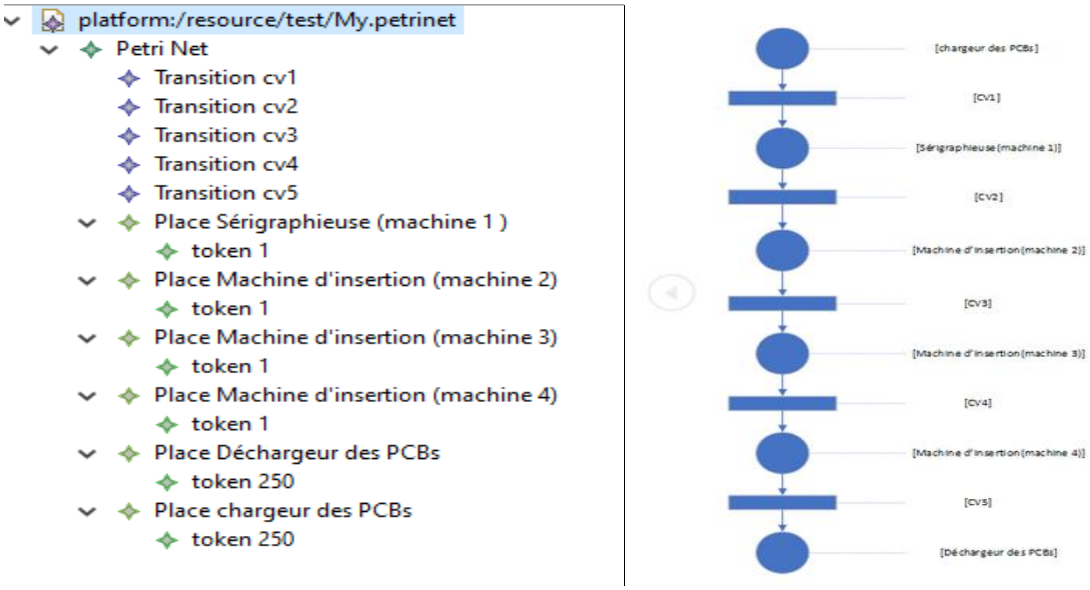


**Figure 5.21:** Exemple de modèle réseaux de pétri.

## 5.6 Le langage de génération de code Xpand

Dans cette section nous allons présenter l‟outil Xpand (Xpand) du projet oAW (oAW) intégré dans le framework de modélisation d‟eclipse EMF.

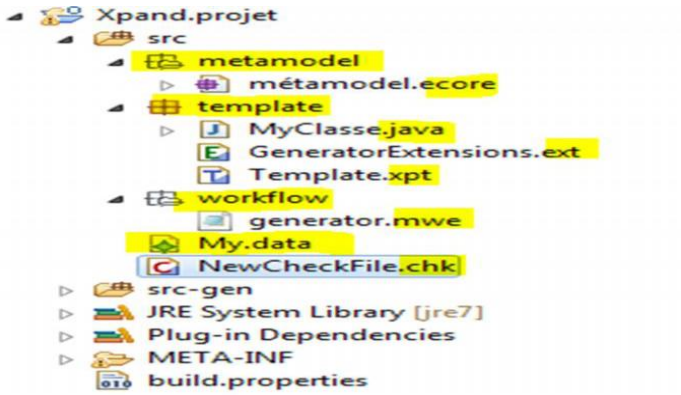


**Figure 5 .22 :** Structure d'un projet Xpand.

**le package méta model** : contient un méta modèle de sortie (ex le méta modèle d'un langage).
**le package Template** : contient un fichier java qui a l'extension « **.java** », un fichier xpand qui a l'extension « **.xpt** » et un fichier xtend qui a l'extension « **.ext** ».
**le package Workflow** : dont l'extension « **.mwe** », contient le workflow qui permet D'appliquer le Template sur un modèle pour engendrer un ou plusieurs fichiers textes.

### 5.6.1 Structure générale d'un template Xpand

Le template Xpand (Klatt, 2007) permet le contôle de la génération de code correspondant à un modèle. Le modèle doit être conforme à un méta-modèle donnée. Le template est stocké dans un fichier ayant l‟extension « .xpt ».

Un fichier template se compose d‟une ou plusieurs instructions IMPORT afin d‟importer les méta-modèles, de zéro ou plusieurs EXTENSION avec le langage Xtend et d‟un ou plusieurs blocks DEFINE.

**Exemple:**

«IMPORT meta::model»

«EXTENSION my::ExtensionFile»

«DEFINE javaClass FOR Entity»

«FILE fileName()»

package «javaPackage()»;

public class «name» {

// implementation

}

«ENDFILE»

«ENDDEFINE»

Le template de cet exemple importe la définition du méta-modèle, charge une extension Xtend et définit un template pour des simples classes java appliquées à un élément de la méta-classe Entity. fileName est une fonction Xtend appelée pour retourner le nom du fichier de sortie. Name est un méta-attribut de la méta-classe Entity.

### 5 .6.2 Le bloc DEFINE

Les blocs DEFINE, aussi appelés templates, constituent le concept central du langage Xpand. C‟est la plus petite unité du fichier template. La balise DEFINE se compose d'un nom, une liste optionnelle de paramètres et du nom de la méta-classe pour laquelle le template est défini. Les templates peuvent être polymorphes, ils ont le format suivant :

«DEFINE templateName(formalParameterList) FOR MetaClass»

a sequence of statements

«ENDDEFINE»

### 5.6.3 L'instruction FILE

L'instruction FILE redirige la sortie produite, à partir des instructions de son corps, vers la cible spécifiée. La cible est un fichier dont le nom est spécifié par expression. Le format de l‟instruction FILE se présente comme suit:

«FILE expression »

a sequence of statements

«ENDFILE»

## 5.6.4 L'instruction EXPAND

L'instruction EXPAND appelle un bloc DEFINE et insère sa production "output" à son emplacement. Il s'agit d'un concept similaire à un appel de sous-routine (méthode). Le format de l‟instruction EXPAND se présente comme suit :

«EXPAND definitionName [(parameterList)]

[FOR expression | FOREACH expression [SEPARATOR expression] ]»

definitionName est le nom du bloc DEFINE appelé. Si FOR ou FOREACH est omise l'autre template est appelé pour l'instance courante this. Si FOR est spécifié, la définition est exécutée pour le résultat d'une expression cible. Si FOREACH est spécifiée, l'expression cible doit être évaluée à un type collection. Dans ce cas, la définition spécifiée est exécutée pour

chaque élément de cette collection. Il est possible de spécifier un séparateur pour les éléments générés de la collection.

«DEFINE javaClass FOR Entity»

..

«EXPAND methodSignature FOR this.methods»

..

«ENDDEFINE»

«DEFINE methodSignature FOR Method»

...

«ENDDEFINE»

## 5 .7 Transformations de modèle à texte (M2T)

Le projet Model-to-Texte (M2T) se concentre sur modèle [65]. Une grande classe de transformations convertit les modèles en texte. Le texte peut être du code généré, d'autres modèles dans la grammaire du texte ou d'autres objets texte, tels que des rapports ou des documents. Le texte est généralement généré à l'aide de modèles, qui est une technologie très mature (par exemple dans le développement Web). Les modèles peuvent être considérés comme du texte cible avec des trous pour les parties variables. Ces trous contiennent du méta-code (et donc du code de création de code), qui est exécuté à l'instanciation du modèle pour calculer les parties variables.

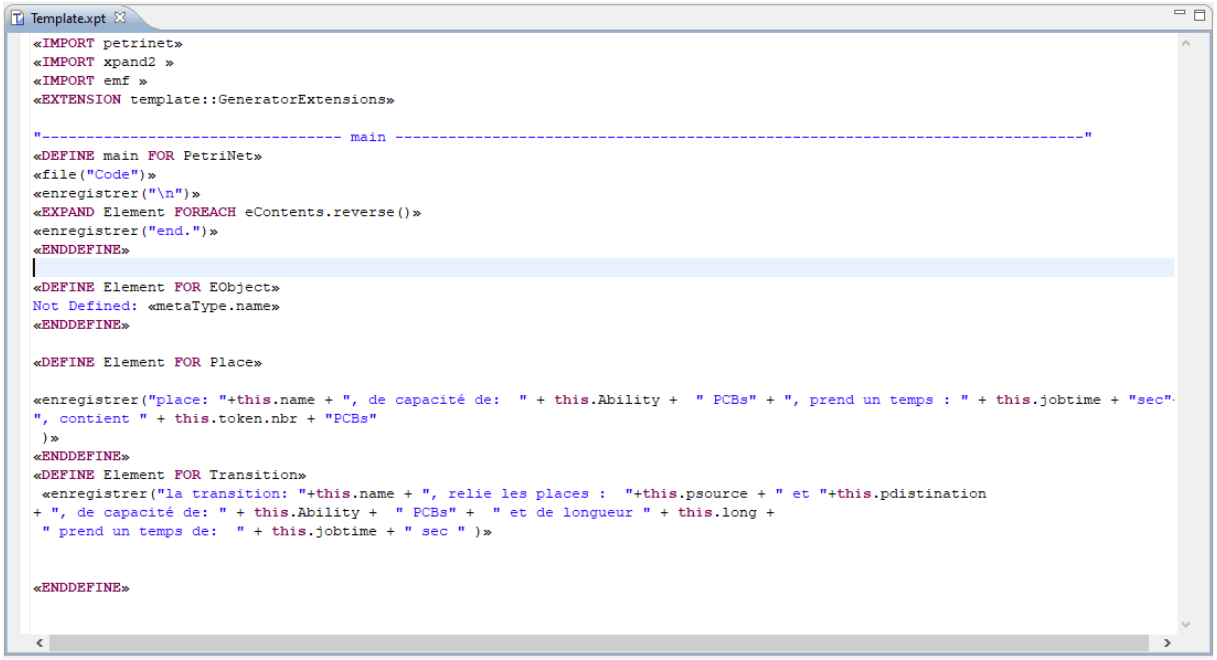


**Figure 5 .23 :** Le template de génération de code.

```
package template;
import java.io.*;
public class MyClasse {
    private static  String filename = "c://test/code.petrinet";
    private static boolean existed = true;
    public static PrintWriter print1;


        public static void enregistrer(String aString) {
            try {
                 print1 = new PrintWriter( new BufferedWriter ( new FileWriter  (filename,existed)));
                 print1.println();
                 print1.println(aString);


                 print1.close();

            } catch (IOException e) {
                // TODO Auto-generated catch block
                e.printStackTrace();}
            System.out.println(aString);
          }

        public static void enregistrer1(String aString) {
            try {
                 print1 = new PrintWriter( new BufferedWriter ( new FileWriter  (filename,existed)));
                 print1.println("  "+aString);

                 print1.close();

            } catch (IOException e) {
```

```
            } catch (IOException e) {
                // TODO Auto-generated catch block
                e.printStackTrace();}
            System.out.println(aString);
          }

        public static void enregistrer2(String aString) {
            try {
                 print1 = new PrintWriter( new BufferedWriter ( new FileWriter  (filename,existed)));
                 print1.println("    "+aString);

                 print1.close();

            } catch (IOException e) {
                // TODO Auto-generated catch block
                e.printStackTrace();}
            System.out.println(aString);
          }


        public static void enregistrer3(String aString) {
            try {
                 print1 = new PrintWriter( new BufferedWriter ( new FileWriter  (filename,existed)));
                 print1.println("      "+aString);

                 print1.close();

            } catch (IOException e) {
                // TODO Auto-generated catch block
                e.printStackTrace();}
            System.out.println(aString);
          }
```

**Figure 5.24 :** Template Xpand pour la génération de code petrinet.

La prochaine étape de ce travail consiste à illustrer la génération de code dans le langage Xpand Du modèle EMF [65]. Le processus que nous allons suivre commencera par l'utilisation Métamodèle petrinet, en écrivant un modèle Génération de code dans les fichiers ".xpt", en exécutant le générateur Enfin en ajoutant quelques contraintes (vérifications). Dans notre cas, nous voulons appeler Méthodes Java dans les expressions (SaveString et file). Cela peut être fait en fournissant Extensions Java via les classes Java. A la fin de cette étape, nous

devrions Obtenez une génération de code qui fonctionnera avec n'importe quel modèle de sortie petrinet La transformation des règles TGG est effectuée par l'interpréteur TGG à l'étape précédente. une fois que nous sommes dans module de temporisation petrinet la dernière étape consistera à tester l'exactitude (correctness) du système concerné en utilisant une vérification formelle. Cette dernière étape sera réalisée en utilisant l'outil de vérification de modèle Maude LTL model-checker.

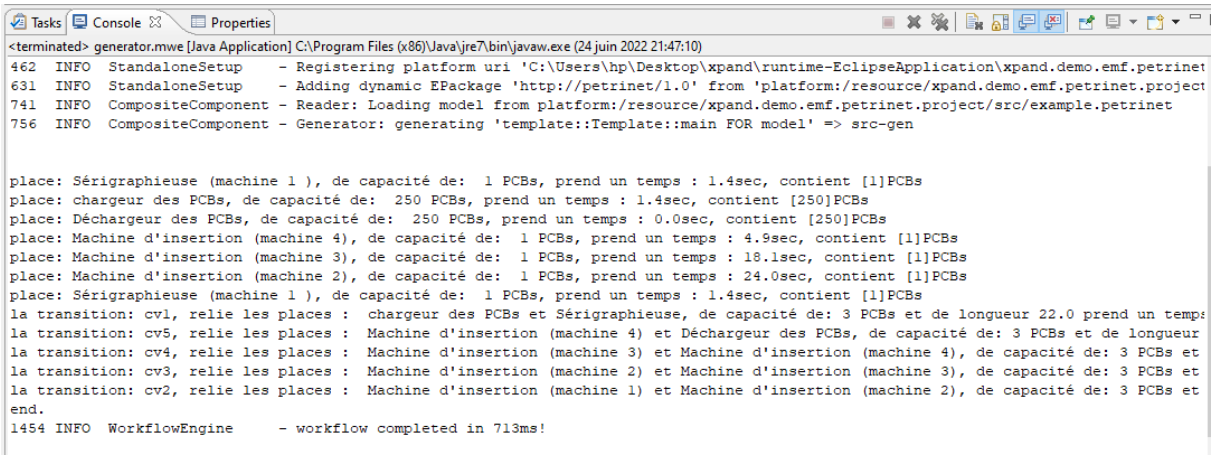


**Figure 5 .24 :** les Résulta de Cod

## 5.8 Conclusion

Dans ce chapitre, nous introduisons l'implémentation de la transformation de modèle basée sur la grammaire des graphes. Notre travail se concentre spécifiquement sur l'utilisation des spécifications de transformation sous forme TGG et la mise en œuvre des transformations à l'aide d'outils d'interprétation TGG intégrés dans l'environnement EMF.

# 6 .Conclusion générale

Le travail présenté dans ce mémoire s'inscrit dans le domaine de l'ingénierie dirigée par les modèles. Il se base essentiellement sur l'utilisation combinée de méta-modélisation et de transformation de modèle. Plus précisément, la transformation de graphe est utilisée comme outil de transformation de modèles

L'ingénierie dirigée par des modèles (IDM), qui prend en charge le développement de logiciels, a Cela apporte certains avantages dans la production et la maintenance du système. IDM n'a de sens que lorsque la transformation du modèle s'installe Très important pour automatiser partiellement ou entièrement le processus développer. Le concept de transformation repose sur un ensemble de techniques, Langages et outils qui permettent l'automatisation de la conversion et Générez automatiquement du code à partir de modèles dans le but ultime de Promouvoir les concepts de programmation et minimiser les coûts de développement. Dans cet article, nous proposons une approche de méthode d'ensemble Méthodes de développement formelles basées sur des fondations d'ingénierie directionnelle Via des modèles, y compris les grammaires de graphes et la métamodélisation Transformez le modèle du point de vue d'IDM. Plus précisément, nous Un pont technique entre MDA et les grammaires de graphes triplés. Les travaux présentés dans cet article appartiennent au domaine de l'ingénierie. Piloté par le modèle. Il repose essentiellement sur une utilisation combinée Métamodélisation et transformation de modèles. Plus précisément, la métamodélisation et transformation de diagramme de classes, rendu à l'aide d'EMF Un cadre complet et extensible pour les outils de développement et d'utilisation MDA Interpréteur TGG intégré à la plateforme EMF/Eclipse.

Le résultat de notre travail est une approche automatique pour transformer Le system de production vers Rdp « Réseaux de pétri ». L'approche proposée est basée sur la transformation de graphes, et elle est réalisée à l'aide de l'outil éclipse Le travail est réalisé dans deux étapes :

- La première étape consiste à proposer deux méta-modèles :system de production et réseaux de pétri.
- La deuxième étape : propose une grammaire de graphe permettant de transformer le system de production vers une reseau de pétri.

Le sujet une fois traité, ne s'est pas refermé, il peut être amélioré et perfectionné, dans de différentes vues possibles.

Finalement, afin d'arriver à développer une approche totalement automatique, incluant tous les system de production, nous proposerons de transformation des System de production vers les Rdp en utilisant toujours la transformation de graphes et l'outil éclipse.

# 7. Bibliographie

[1] Ingénierie dirigée par les modèles. Homepage. https://ics.utc.fr/ ~tha/co/IDM.html .
[2] H. Ehrig, K. Ehrig, C. Ermel, F. Hermann, and G. Taentzer ,(2007).“Information preserving bidirectional model transformations,” in FASE (M.B. Dwyer and A. Lopes, eds.), vol. 4422 of Lecture Notes in Computer Science, pp.72–86, Springer.
[3] L.Menet ,(2010). “Formalisation d’une approche d’Ingénierie Dirigéepar les Modèles appliquée au domaine de la Gestion des Données de Référence(in french)”,these de doctorat, Université PARIS VIII.
[4] F.Guerrouf ,(2009). “Une approche de transformation des Diagrammesd’Activités d’UML Mobile 2.0 vers les Réseaux de Petri,memoire de magistere,UniversitéEl Hadj Lakhdar BATNA.
[5] Mohamed Yassine Haouam, Préservation de la traçabilité des préoccupations dans un contexte d'ingénierie basée sur les modèles. Thèse de Doctorat. Informatique.Université Badji Mokhtar-Annaba, 2015.
[7] A.Hettab ,(2009). De M-Ml vers les réseaux de pétri « Nested Nets» :une approche basée sur la transformation de graphes. Thèse de doctorat. Universitéde mentouri Constantine
[8] : M.Amroune ,(2014).vers une approche orientée aspect d’ingénieriedes besoins dans les organisations multi-entreprise ,these de doctorat,UniversitéToulouse 2 Jean Jaurès .
[9] :M.AOUAG ,(2014). Des diagrammes UML 2.0 vers les diagrammes orientés aspect a laid de transformation de graphes,these de doctorat,Université de Constantine 2.
[10] Chris Raistrick, Paul Francis, John Wright, Colin Carter et Ian Wilkie. 2004. Model Driven Architecture with Executable UML. Cambridge.
[11] Mellor Stephen J, Scott Kendall, Uhl Axel et Weise Dirk. 2004. MDA distilled: principles of model-driven architecture. Addison-Wesley Professional.
[12] Bran Selic. The pragmatics of model-driven development. Software, IEEE, 20(5) :19–25,2003.
[13] Ekkart Kindler, Vladimir Rubin, and Robert Wagner. An adaptable tgg interpreter for in-memory model transformation. Proc. of the FUJABA Days, pages 35–38, 2004.
[14] Krzysztof Czarnecki and Simon Helsen Classification of Model Transformation Approaches Krzysztof Czarnecki and Simon Helsen University of Waterloo, Canada. czarnecki@acm.org, shelsen@computer.org.

[15] Ekkart Kindler and Robert Wagner. Triple graph grammars: Concepts, exten- sions, implementations, and application scenarios. University of Paderborn, 2007.
[16] Sana Mellouli ,2014,Méta-modélisation du Comportement d’un Modèlede Processus : Une Démarche de Construction d’un Moteur d’Exécution ”.thèsede doctorat,université de paris.
[17] Object Management Group (OMG), Model Driven Architecture (MDA), site Internet, http://www.omg.org/mda,2004.
[18] Sana Fathallah. Résolution des interférences pour la composition dynamique de

services en informatique ambiante. Thèse de doctorat, Université Nice Sophia Antipolis, 2013.
[19] Michael Löwe. Algebraic approach to single-pushout graph transformation.
[20] Andy Schürr. Specification of graph translators with triple graph grammars. In Graph-Theoretic Concepts in Computer Science, pages 151–163. Springer, 1995.
[21] [Matthieu ROQUE] ; « Contribution à la définition d'un langage générique de modélisation d'entreprise ». Thèse de doctorat, Spécialité productique : université Bordeaux1, 2005.
[21] [Xiaojun YE]; « Modélisation et Simulation des Systèmes de Production: une Approche Orientée-objets ». Thèse de doctorat, spécialité : ingénierie informatique : institut national des sciences appliquées de LYON, 1994.
[23] [Abdennebi TALBI] ; « Modélisation des systèmes industriels ». Editions universitaires européennes. ISBN: 978- 613-1-59806-7, Ecole Supérieure de Technologie, Fès, 2012.
[24] [Naoufel CHEIKHROUHOU]; « Modélisation et Simulation de Systèmes de Production » ; cours, Dr-Ing, Laboratoire de Gestion et Procédés de Production, MOSISP 2007.
[25] [Georges HABCHI]; « Conceptualisation et modélisation pour la simulation des systèmes de production ». Document de Synthèse, Pour l'obtention de l'habilitation a dirigé des recherches, Université de SAVOIE, 2001.
[26] [Karim TAMANI]; « Développement d'une méthodologie de pilotage intelligent par régulation de flux adaptée aux systèmes de production ». Thèse de doctorat. Université de SAVOIE, Spécialité : Electronique -Electrotechnique – Automatique, 2008.
[27] [Paul-Marie BOULANGER, Thierry BRECHET], « Une analyse comparative des classes de modèles », rapport de recherche, Action de support PADD I, SSTC. Institut pour un développement durable ; Chaire Lhoist Berghmans, Core – UCL, Bruxelles 2003.
[28] [[Lazhar BENOUDINA]; « Modélisation et simulation basées multi-agents du contrôle de processus industriel». Mémoire de magistère : Ecole Doctorale de l'informatique INI, Option Sciences et Technologie de l'Information et de la communication. Alger 2009
[29] [Timothée KOMBE]; « Modélisation de la propagation des fautes dans les systèmes de production » ; thèse de doctorat, Électronique, Électrotechnique, Automatique, Spécialité : Automatique Industrielle : institut national des sciences appliquées de LYON, 2011.
[30] [Séverine SPERANDIO], « Usage de la modélisation multi-vue d'entreprise pour la conduite des systèmes de production ». Thèse de doctorat, Spécialité productique. Université Bordeaux 1, 2005.
[31] [Xiaolan XIE], « Evaluation des Performances des systèmes de production ».cours, Ecole Nationale Supérieure des Maines, SAINT-ETIENNE, Département Génie Industriel Hospitalier, Centre Ingénierie et Santé ; Master GI 2007.
[32] [P NAIM, P HENRI, K WUILLEMIN, P LERAY, O POURRET, A BECKER], « Réseaux Bayésiens ». 3ème Edition, Eyrolles. 2007.
[33] Benmoussa, Fatima Zohra.et Houidi,Farida. Une approche basé sur la transformation du graphe sous ATom3 pour l'intégration des deux outils de réseau de petri.Mémoire de master. Université Echahide Hamma Lakhdar,EL-OUED.pp :4-6.

[34] :E.Kerkouche ,(2011). Modélisation Multi-Paradigme : Une Approche Basée sur laTransformation de Graphes,these de doctorat ,universite de mentouri constatine.
[35] T.Murata ,(1989) .Petri Nets : Propretés, analysis and applications, proceeding of the IEEE.
[36] Bahri, Mohamed Rédha .(2011).Une approche intégrée Mobile-UML/Réseaux de Petri pour l'Analyse des systèmes distribués à base d'agents mobiles. Thèse de doctorat, Université de Mentouri, Constantine.pp : 55-71.
[37] N.Dehimi, (2014). Un cadre formel pour la modélisation et l'analyse des agents mobiles.Thèse de doctorat. Université de mentouri Constantine.
[38] Elmansourie ,(2009) .Modélisation et vérification des processus métiers dans les entreprises virtuelles :Une Approche basée sur la transformation de graphes, thèse de doctorat, université de mentouris Constantine.
[39] R,Torchi, B,Mezahi ,(2018). Le développement d'un outil de transformation des modèles orientes aspect vers les réseaux de pétri, base sur la transformation de graphes.Mémoire de master. université de Mila.
[40] A.Laouar,(2013). Une approche de transformation de diagrammes d'activités oriente aspects vers les réseaux de pétri colores basée sur la transformation de graphes.Mémoire de master. Université de constantine2.
[41] Saggadi,Samira.(2007).Optimisation des temps d'attente des système flexibles de production basée sur les réseaux de pétri . Mémoire de magis- ter. Université M'hamed bougara,Boumerdése .pp :24-25.
[42]https://dept-info.labri.fr/~griffaul/Enseignement/MasterInfo-04-05/FDS/vademecum-petri.pdf
[43] Haiouni,Houda.(2010). Approche mixte de modélisation par Réseaux de Petri et SMA. Mémoire de magister. Université de Mentouri, Constantine. pp :51-52
[44] Belounnar ,Saliha.Une approche formelle pour l'adaptabilit d'un agent.Mémoire de Magister . Université Mohamed Khider , Biskra.pp
[45] :J.L.Peterson (1981) ,Petri Net Theory and The Modeling of System,Practice-hall,Englewood Chiffe.
[46]V. Murilo Savi ,(1994) .Conception Préliminaire des systèmes de production à l'aide des réseaux de pétri,these de doctorat,université de metz.

[47] Bouarioua ,Mouna.(2013). Une approche basée transformation de graphes pour la génération de modèles de réseaux de Petri analysables à partir de diagrammes UML. Thèse de doctorat .Université de Mentouri,Constantine.pp :71-81
[48] TGG interpreter. Home Page. http://www.cs.uni-paderborn.de/index.php?id=tgg-interpre
[49] Voormann ,H.Luna'Rising [Online]. 2014. Disponible http://eclipsehowl.wordpress.com/2014/06/25/luna-rising/, (Consulté mai 2022 ).
[50] Slimane HAMMOUDI. *Contribution à l'étude et à l'application de l'ingénierie dirigée par les modèles.* Thèse de doctorat, université de DANGERS, 2010.
[51] Dave Steinberg, Frank Budinsky, Ed Merks et Marcelo Paternostro. Emf : eclipse Mödling Framework. Pearson Education, 2008. (Cité en page 124.)

[52] Andy Schürr. Specification of graph translators with triple graph grammars. In Graph-Théorétique Concepts in Computer Science, pages 151–163. Springer, 1995.
[53] Alexander Konigs. Model transformation with triple graph grammar. Septembre, 2005.
[54] TGG interpreter. Site web : http://www-old.cs.uni-paderborn.de/en/research-group/softwareengineering/ research/projects/tgg-interpreter.html, (Consulté mai 2022).
[55] M. Bendiaf, (2018).Spécification et vérification des systèmes embarqués temps réel en utilisant la logique de réécriture.Thèse de doctorat.Université de Mohamed Khider, Biskra.