الجمهورية الجزائرية الديمقراطية الشعبية
République Algérienne Démocratique et Populaire
وزارة التعليم العالي والبحث العلمي
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

N° Réf :……

Centre Universitaire
Abd Elhafid Boussouf Mila

Institut des Sciences et Technologie Département de Mathématiques et Informatique

Mémoire préparé en vue de l'obtention du diplôme de Master

En : Informatique

Spécialité: Sciences et Technologies de l'Information et de la Communication (STIC)

La génération d'un outil de transformation des modèles orientés aspect vers des modèles formèls. Une approche basée sur la transformation de graphes

Préparé par : Boualita Salim
Laggoune Fouad

Devant le jury :

Boumassata Meriem	(M.A.A)	C.U.Abd Elhafid Boussouf	Président
Aouag Mouna	(M.C.B)	C.U.Abd Elhafid Boussouf	Rapporteur
Abderrezek Samira	(M.A.A)	C.U.Abd Elhafid Boussouf	Examinateur

Année Universitaire : 2020/2021

Remerciements

Tout d'abord, nous remercions Allah le tout puissant de nous avoir donné le courage et la patience nécessaires à mener ce travail à son terme.

Nous tenons à remercier tout particulièrement notre encadrante **Mme. Aouag Mouna** pour l'aide compétente qu'elle nous a apportée, pour sa patience et son encouragement. Son œil critique m'a été très précieux pour structurer le travail et pour améliorer la qualité des différentes sections.

Nous souhaitons remercier également les membres de jury qui trouvent, ici, l'expression de mes sincères remerciements pour l'honneur qu'ils me font en prenant le temps de lire et d'évaluer ce travail.

Nous souhaitons aussi remercier l'équipe pédagogique et administrative du centre universitaire mila pour leurs efforts dans le but de nous offrir une excellente formation.

Nous tenons à remercier mes parents et mes amis.

Sans oublier de remercier tous nos amis et nos collègues et à toutes les personnes qui nous ont aidé dans la réalisation de ce mémoire.

Pour finir, nous souhaitons remercier toute personne ayant contribué de près ou de loin à la réalisation de ce travail.

Résumé

Dans le monde de génie logiciel, UML est le langage de modélisation orientée objet le plus utilisé. Il offre au développeur un ensemble de diagrammes qui donne les différentes vues d'un système à réaliser. Mais avec l'évolution de l'informatique, les problèmes deviennent plus complexes, pour cela l'orienté objet a montré ses limites pour les qu'elles ce paradigme n'a donné aucune solution à ce type de problème. Donc, La modélisation orientée aspect est un nouveau paradigme qui apparut pour améliorer et résoudre les limites de la modélisation orientée objet.

Les diagrammes UML2.0 orientés aspects ne possèdent pas de sémantique. Malgré son succès, ce langage souffre du manque de capacités de vérification, validation et d'analyse.

L'objectif visé dans ce mémoire est de proposer une approche totalement automatique qui permet de transformer les diagrammes UML 2.0 (états transitions) orientés aspect vers les réseaux de pétri. Cette approche est basée sur la transformation de graphes et réalisée à l'aide de l'outil AToM^3. On va proposer des méta-modèles et une grammaire de graphes pour effectuer la transformation. Finalement on va argumenter notre proposition avec des études de cas bien illustrées.

Mots clés : UML 2.0, Les Diagrammes d'états transitions Orientés Aspect, Les Réseau de pétri, Grammaire de graphes, AToM^3, IDM, Transformation de graphes.

Abstract

In the software engineering world, UML is the most widely used object-oriented modeling language.
object-oriented modeling language. It offers the developer a set of diagrams that give the different views of a system to be realized. But with the evolution of computer science, the problems become more complex, for that the object oriented has shown its limits for them this paradigm has not given any solution to this type of problem. Therefore, the aspect-oriented modeling is a new paradigm that appeared to improve and solve the limits of the object-oriented modeling.

UML2.0 aspect-oriented diagrams do not have semantics. Despite its success, this language suffers from a lack of verification, validation and analysis capabilities.

The goal of this mémory is to propose a fully automatic approach to transform aspect-oriented UML 2.0 diagrams (state chart) into petri nets. This approach is based on graph transformation and realized with the AToM^3 tool. We will propose meta-models and a graph grammar to perform the transformation. the transformation. Finally we will argue our proposition with well illustrated case studies.

Keywords : UML 2.0, Aspect Oriented state chart Diagrams, Pétri nets, Graph grammar, AToM^3, IDM, Graph transformation of graphs.

ملخص

في عالم هندسة البرمجيات لغة النمدجةUML هي اللغة الأكثر استخداما الموجهة للأشياء وهذه الأخيرة بدورها توفر مجموعة من الرسوم البيانية التي تعطي وجهات نظر مختلفة للنظام لتم تحقيقه. ولكن مع تطور علوم الحاسوب أصبحت المشكلات أكثر تعقيدا وبدورها لغة النمدجة أظهرت حدودها في حل هذه المشاكل ولم تقدم أي حلول ,وهدا السبب أدى الى ظهور نمذجة جديدة تسمى النمذجة الموجهة الى الجانب aspect orienté

لا تحتوي المخططات الموجهة للعرض في UML2.0 على معنى . على الرغم من نجاحها ، إلا أن هذه اللغة تعاني من نقص في قدرات التحقق ,المصادقة والتحليل

لهدف من هذه المذكرة هو توفير نهج آلي بالكامل يحول مخططات (etat-transition 2.0UML) الموجهة إلى الجوانب إلى شبكات بتري. يعتمد هذا النهج على تحويل الرسوم البيانية ويتم تنفيذه باستخدام أداةAToM³ .سنقترح نماذج وصفية وقواعد بيانية لاجراء التحويل .

كلمات مفتاحية : réseau de pétri , Les diagrammes état-transition orienté-aspect , UML 2.0 Transformation de graphes ,IDM. ,AToM³ Grammaire de ,Graphes

---

Table des matières

Remerciements . . . . . **I**

Résumé . . . . . **II**

Abstract . . . . . **III**

ملخص . . . . . **IV**

1 Introduction Générale **1**
1.1 Introduction 1
1.2 Problématique 1
1.3 Contribution 2
1.4 Organisation du mémoire 2

2 Approche Orientée Aspect **4**
2.1 Introduction 5
2.2 Le paradigme orienté aspect 5
2.3 Les concepts de base du paradigme orienté aspect 5
2.3.1 Préoccupation 5
2.3.2 Aspect 6
2.3.3 Point de jonction (join point en anglais) 6
2.3.4 Point de Coupe (pointcut en anglais) 6
2.3.5 Conseil (advice en anglais) 6
2.3.6 Tisseur (Weaver en anglais) 6
2.4 Les techniques et les technologies orientée aspect 7
2.4.1 La programmation orientée aspect 7
2.4.2 L'ingénierie des exigences orienté aspect 7
2.4.3 Les approches d'architecture orientée aspect 7
2.4.4 Les approches de conception orientée aspect 8
2.4.5 Les approches de vérification et de tester des programmes orientés aspect 8
2.5 Modélisation orientée aspect 8
2.5.1 Définition de la modélisation orientée aspect 8
2.5.2 La modélisation graphique en aspect 10
2.6 Les approches de la modélisation orientée aspect 10
2.6.1 Extension UML léger 10
2.6.2 Extension UML lourd 11
2.6.3 Approches de la transformation des modèles 11

2.7 L'inconvénient de la modélisation orientée aspect . . . 11
2.8 Conclusion . . . 12

3 Réseau De Pétri . . . 13
3.1 Introduction . . . 14
3.2 Historique . . . 14
3.3 Réseau de Pétri . . . 14
3.4 Concepts de bases de réseau de pétri . . . 15
3.4.1 Définition graphique . . . 15
3.4.2 Définition formelle . . . 16
3.4.3 Représentation Matricielle . . . 16
3.4.4 L'arbre de recouvrement (le marquage d'un Rdp) . . . 17
3.5 Évolution d'un Rdp . . . 19
3.5.1 Transition validée . . . 20
3.5.2 Franchissement . . . 20
3.5.3 Règle de franchissement . . . 20
3.6 Sous-Classes des Rdp . . . 21
3.6.1 Machine á état et graphes d'événements . . . 21
3.6.2 Rdp sans conflit . . . 22
3.6.3 Rdp pur . . . 22
3.6.4 Rdp choix libre . . . 22
3.6.5 Rdp simple . . . 23
3.6.6 Rdp généralisés . . . 23
3.6.7 Rdp á capacité . . . 23
3.6.8 Rdp á priorités . . . 24
3.7 Extension des réseaux de pétri . . . 24
3.7.1 Les Rdp Colorés . . . 25
3.7.2 Rdp temporisés . . . 25
3.7.3 Rdp synchronisés . . . 26
3.7.4 Rdp stochastique . . . 27
3.7.5 Rdp avec un arc inhibiteur . . . 27
3.8 Methods d'analyses . . . 28
3.8.1 Arbre de recouvrement . . . 28
3.8.2 Algèbre linéaire . . . 29
3.8.3 Réduction du réseau . . . 29
3.9 Modélisation Avec les Rdp . . . 29
3.9.1 Parallélisme . . . 30
3.9.2 Synchronisation . . . 30
3.9.3 Calcule de flux de données . . . 31
3.10 Principales propriétés des Rdp . . . 32
3.10.1 Vivacité . . . 32
3.10.2 Réseau borné . . . 32
3.10.3 Consistance et réversibilité . . . 32
3.10.4 Persistance . . . 32
3.10.5 Invariants . . . 32
3.11 Utilisation des Rdp . . . 33

3.12 Conclusion 33

4 La transformation de modèles 34
4.1 Introduction 35
4.2 Les Concepts de base de l'ingénierie dirigée par les modèles 35
4.2.1 Système 36
4.2.2 Modèle 36
4.2.3 Méta-Modèle 36
4.2.4 Méta-Méta-Modèle 36
4.2.5 Conforme a 37
4.3 L'approche MDA 37
4.3.1 CIM(Computer Independent Model) 37
4.3.2 PIM(Plateforme independent Model) 37
4.3.3 PSM(Plateforme Specific Model) 37
4.3.4 PDM(Plateforme Description Modèle) 37
4.4 L'architecture MDA 39
4.5 Transformation de Modèles 40
4.5.1 Pourquoi la transformation de modèles 40
4.5.2 Les étapes de la transformation de modèles en MDA 40
4.5.3 Les types de transformation de modèles 41
4.6 Les approches de transformation 42
4.6.1 Modèle vers Texte (Modèle vers Code) 42
4.6.2 Modèle vers Modèle 43
4.7 Transformation de graphes 45
4.7.1 Notion de graphe 45
4.7.2 graphe orienté(digraphe) 46
4.7.3 Grammaire de graphe 47
4.7.4 Outils de transformation de graphes 49
4.7.5 les avantages de l outil AToM^3 50
4.7.6 outil d'implémentation 51
4.8 conclusion 52

5 L'approche De Transformation Proposée 53
5.1 Introduction 54
5.2 L'approche proposée 54
5.3 Méta-Modélisation 54
5.3.1 Spécification de méta-Modèle source 54
5.3.2 Specification de méta-Modèle cible 55
5.4 Processus de transformation de modèles 56
5.4.1 Etat-Simple 56
5.4.2 Etat-Final 57
5.4.3 Etat-Composite 57
5.4.4 Pseudo-État 57
5.4.5 Etat Aspect 58
5.4.6 Transition Simple 58
5.4.7 Transition de bifurcation 59

5.4.8 Transition de jointure . . . 59
5.4.9 Transition de jonction . . . 60
5.4.10 Transition de choix . . . 60
5.4.11 Transition de point d'entrée . . . 60
5.4.12 Transition de point de sortie . . . 61
5.4.13 Transition d'un état-initial . . . 61
5.4.14 Transition d'un état final . . . 61
5.4.15 Transition interne . . . 62
5.5 Grammaire de graphes pour la transformation des diagrammes etat-transition orientés aspect vers les réseaux de pétri . . . 62
5.5.1 Règle 1 . . . 63
5.5.2 Règle 2 . . . 63
5.5.3 Règle 3 . . . 64
5.5.4 Règle 4 . . . 64
5.5.5 Règle 5 . . . 65
5.5.6 Règle 6 . . . 65
5.5.7 Règle 7 . . . 66
5.5.8 Règle 8 . . . 66
5.5.9 Règle 9 . . . 67
5.5.10 Règle 10 . . . 67
5.5.11 Règle 11 . . . 68
5.5.12 Règle 12 . . . 68
5.5.13 Règle 13 . . . 69
5.5.14 Règle 14 . . . 69
5.5.15 Règle 15 . . . 70
5.5.16 Règle 16 . . . 70
5.5.17 Règle 17 . . . 71
5.5.18 Règle 18 . . . 71
5.5.19 Règle 19 . . . 72
5.5.20 Règle 20 . . . 72
5.5.21 Règle 21 . . . 73
5.5.22 Règle 22 . . . 73
5.5.23 Règle 23 . . . 74
5.5.24 Règle 24 . . . 74
5.5.25 Règle 25 . . . 75
5.5.26 Règle 26 . . . 75
5.5.27 Règle 27 . . . 76
5.5.28 Règle 28 . . . 76
5.5.29 Règle 29 . . . 77
5.5.30 Règle 30 . . . 77
5.5.31 Règle 31 . . . 78
5.6 Conclusion . . . 78

6 Etude de cas sur la transformation de diagramme état-transition orienté-aspect vers réseau de pétri . . . 79
6.1 Introduction . . . 80

6.2 Etude de cas 80
6.2.1 Exemple1: etude de cas sur le comportement d'un distributeur automatique d'argent 80
6.2.2 Exemple 2: etude de cas sur le comportement d'une fenêtre principale d'un serveur de messagerie 81
6.3 conclusion 83

7 Conclusion générale 84

Conclusion générale 84

Bibliographie 86

Table des figures

2.1 Le processus de la modélisation orientée aspect 9
2.2 Notion Complet 10

3.1 Méthodes générale de modélisation et d'analyse basée sur les Rdp 15
3.2 Exemple d'un Résau de Pétri 16
3.3 Un Rdp Marqué. 17
3.4 Matrice d'incidence. 17
3.5 un exemple d'arbre des marquages atteignables. 18
3.6 un exemple d'arbre de recouvrement. 19
3.7 Transition validée. 20
3.8 franchissement d'un Rdp. 20
3.9 exemple de règle de franchissable de transition. 21
3.10 exemple d'une machine à états et d'un graphe d'événement. 21
3.11 Rdp Pur 22
3.12 réseau à choix libre, réseau à choix libre étendu et réseau à choix non libre. 23
3.13 Rdp simple. 23
3.14 Rdp généralisé. 23
3.15 Rdp á capacité 24
3.16 Rdp á priorité 24
3.17 Rdp(gauche) et Rdp coloré(droite). 25
3.18 Rdp stochastique. 27
3.19 Rdp inhibiteur. 28
3.20 parallélisme dans les Rdp. 30
3.21 Problème du producteur et consommateurs. 30
3.22 Exclusion mutuelle 31
3.23 Exemple d'un calculé de flux de données par un Rdp. 31
3.24 L'utilisation des Rdp 33

4.1 Notion de base en ingenierie des modeles. 36
4.2 illustration OMG du MDA 38
4.3 Processus en Y de transformation de modèle dans l'approche MDA 39
4.4 Pyramide de modélisation de l'OMG 39
4.5 Les activités de la transformation de modèles. 40
4.6 Concepts de base de la transformation de modèles. 41
4.7 Les types de transformation de modèles. 42
4.8 Les approches de transformation de modèles. 42
4.9 Principe de l'application d'une règle. 44
4.10 Les approches de transformations de modèles 45

4.11 Graphe non-orienté. 45
4.12 Graphe orienté 46
4.13 Graphe (A) et sous-graphe (B). 47
4.14 Graphe étiqueté. 47
4.15 Principe de mise en œuvre de transformation de graphes 49
4.16 Présentation de l'outil AToM³ 51

5.1 Diagramme état-transition orienté-aspect. 55
5.2 Méta-Modèle de réseau de petri. 56
5.3 Transformation d'une état-simple 56
5.4 Transformation d'un état-final 57
5.5 Transformation d'un état composite 57
5.6 Transformation d'un pseudo-état initial 58
5.7 Transformation d'un pseudo-etat terminal. 58
5.8 Transformation d'un état aspect. 58
5.9 Transformation d'une transition simple. 59
5.10 Transformation d'une transition de bifurcation. 59
5.11 Transformation d'une transition de jointure. 59
5.12 Transformation d'une transition de jonction. 60
5.13 Transformation d'une transition de choix. 60
5.14 Transformation d'une point d'entrer 61
5.15 Transformation d'un point de sortie 61
5.16 Transformation d'une transition d'un état initial 61
5.17 Transformation d'une transition d'un état final. 62
5.18 Transformation d'une transition interne. 62
5.19 Grammaire de graphes proposée. 63
5.20 Règle 1: Diagramme etat-transition orienté-aspect. 63
5.21 Règle 2: Création Etat Simple. 64
5.22 Règle 3: Création Etat-initial. 64
5.23 Règle 4: Création Etat-final. 65
5.24 Règle 5: Création Aspect. 65
5.25 Règle 6: Création Etat-Composite. 66
5.26 Règle 7: Création Relation entre etat-initial et un etat-simple. 66
5.27 Règle 8: Création Relation entre une etat-initial et un aspect-simple. . . . 67
5.28 Règle 9: Création Relation entre un etat-initial et un etat-composite. . . . 67
5.29 Règle 10: Création relation entre etat-initial et aspect-composite. 68
5.30 Règle 11: Création relation entre une etat-simple et une autre etat-simple. 68
5.31 Règle 12: Création relation entre une etat-simple et lui mème. 69
5.32 Règle 13: Création relation entre une etat-simple et lui mème. 69
5.33 Règle 14: Création relation entre une etat-simple et une aspect-simple. . . 70
5.34 Règle 15: Création relation entre une etat-simple et une etat-composite. . . 70
5.35 Règle 16: Création relation entre une etat-simple et une aspect-composite. 71
5.36 Règle 17: Création relation entre une etat-simple et une etat-final. 71
5.37 Règle 18: Création relation entre une etat-composite et une etat-simple. . . 72
5.38 Règle 19: Création relation entre une etat-composite et une aspect-simple. 72
5.39 Règle 20: Création relation entre une etat-composite et une etat-final. . . . 73

5.40 Règle 21: Création relation entre une aspect-simple et une etat-simple. . . 73
5.41 Règle 22: Création relation entre une aspect-simple et une etat-composite. 74
5.42 Règle 23: Création relation entre une aspect-simple et une etat-final. . . . 74
5.43 Règle 24: Création relation entre une aspect-composite et une etat-simple. 75
5.44 Règle 25: Création relation entre une aspect-composite et une etat-final. . 75
5.45 Règle 26: supprimer le diagramme d'etat-transition orienté aspect. 76
5.46 Règle 27: supprimer les etats-simples. 76
5.47 Règle 28: supprimer etat-initial. 77
5.48 Règle 29: supprimer etat-final. 77
5.49 Règle 30: supprimer aspect-simple. 78
5.50 Règle 31: supprimer aspect-composite. 78

6.1 Diagramme d'etat transition orienté aspect pour "distributeur automatique d'argent". 80
6.2 Réseaux de pétri pour"distributeur automatique d'argent". 81
6.3 Diagramme d'etat transition orienté aspect pour "fenêtre principale d'un serveur de messagerie". 82
6.4 Réseaux de pétri pour "fenêtre principale d'un serveur de messagerie". . . . 82

Chapitre 1

Introduction Générale

1.1 Introduction

De nos jours les systèmes sont souvent critiques et très complexes dans leur structure ainsi que leur composition. C'est pour cette raison que ces systèmes doivent être modélisés et vérifiés formellement avant leur mise en œuvre. Un des langages les plus utilisés dans la modélisation semi-formelle est UML (Unified Modelling Language), qui est un langage visuel (graphique) spécialement développé pour modéliser proprement un système orienté-objet (OO). Les diverses vues offertes par UML permettent de visualiser plusieurs aspects d'un même système. Ceci permet de mieux gérer la complexité du système. Cependant, UML étant un langage à caractère plutôt visuel, il souffre d'un manque de sémantique formelle. En effet, les notations semi formelles et visuelles d'UML peuvent provoquer des inconsistances au niveau des modèles développés, il est donc impossible d'établir des preuves sur la concordance entre la solution envisagée et celle qui a été élaborée d'une part .et d'autre part, la modélisation doit faire intervenir des mécanismes de gestion du temps et d'évènements où la notion d'évolutions simultanées d'actions est prise en compte. Par conséquent, nous avons besoin des modèles qui supportent l'expression d'actions non atomiques structurellement et temporellement, c'est-à-dire d'actions divisibles et non nécessairement de durée nulle

1.2 Problématique

Les diagrammes UML2.0 sont des modèles orientés objet, mais la modélisation orientée objet a plusieur problèmes et limites tel que : la duplication,l'enchaînement, la résolution et la réutilisation des modèles, pour cela les développeurs pense à définir un nouveau paradigme qui permet de résoudre ces problèmes,ce paradigme appelle "le paradigme orienté aspect". La Modélisation Orientée Aspect (MOA) est une nouvelle méthodologie qui permet la séparation entre les modèles fonctionnels (base) et les modèles non fonctionnels (aspect) pour obtenir des modèles composés (intégrés). Les diagrammes UML2.0 orientés aspect sont classés dans la catégorie semi formelles , ils permettent à travers différents diagrammes de mieux maîtriser le développement d'un système , l'inconvénient majeur des diagrammes UML2.0 orientés aspect est l'absence d'une sémantique formelle ,ce qui rend

impossibles de faire la vérification du comportement des systèmes d'écrit par plusieurs diagrammes.

1.3 Contribution

Pour résoudre ces problèmes ,l'idée générale est de transformer les diagrammes UML 2.0 orientés aspect 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) .

Donc notre travail se situe dans le contexte de la transformation des modèles .Plus précisément, il s'agit de transformation des diagrammes d'état-transition orientés aspect 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 partir des diagrammes d'état-transition orientés aspect 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 implémentés avec l'outil AToM3 et le langage de programmation Python.

1.4 Organisation du mémoire

Ce mémoire est organisé en cinque chapitres, une introduction générale et une conclusion générale :

Le premier chapitre "**L'approche Orientée Aspect**" présente le paradigme orienté aspect avec les différents conceptes de base, leurs technologies, ainsi que les avantages et les inconvénients des approches orientés aspect.

Le deuxième chapitre "**Réseau de Pétri**" Sera consacré pour le formalisme de réseau de pétri avec ses propriétés et sa structure fondamentale permettent la modélisation ..

Le troisième chapitre "**La transformation de modèles**" Sera dédie à la transformation de modèles à l'aide de transformation de graphes. Nous donnons un rappel sur l'architecture dirigée par les modèle ou MDA modèle driven architecture),avec une brève présentation de la théorie de transformation de graphes ,ainsi nous présentons l'environnement de travail "AToM3"

Le quatrième chapitre "**L'approche proposée**" Ce chapitre est organisé en deux grandes parties. La première est consacrée à la définition des règles de transformation qui constituant notre grammaire de graphes. La deuxième présenter les Grammaires de graphes pour la transformation des diagrammes etat-transition orientés aspect vers les réseaux de pétri.

Le cinquième chapitre "**Etudes de cas sur la transformation de diagramme etat-transition orientee aspect vers réseau de pétri**" présente en détail des études de cas illustrant notre approche de transformation à l'aide des exemples. Nous donnons un

diagramme d'états-transitions orienté aspect, et le modèle de reseau de pétri et exécuter l'outil.

Chapitre 2

Approche Orientée Aspect

2.1 Introduction

La modélisation en informatique est l'étape la plus importante dans le développement d'un logiciel. Elle facilite la compréhension du fonctionnement d'un système avant sa réalisation en produisant un modèle. Ils existent plusieurs méthodes de modélisation comme la Modélisation Orientée Objet(MOO), et la Modélisation Orientée Aspect(MOA) . C'est dans ce contexte qu'on a vu émerger les approches de développement basé sur le paradigme orienté aspect. Dans ce chapitre nous nous intéressons par la modélisation orientée aspect, donc dans un premier temps, nous présentons d'abord le paradigme aspect et nous définissons brièvement les concepts de base. Dans les titres qui suivent, nous résumons, les techniques et les technologies orientée aspect, la modélisation orientée aspect, en suit les approche de la modélisation orientée aspect, finalement nous discutons l'inconvénient majeur de la modélisation orientée aspect.

2.2 Le paradigme orienté aspect

Le paradigme aspect utilise le principe de la séparation des préoccupations afin d'offrir une meilleure organisation des programmes, par rapport aux approches classiques de développement. En effet, dans les approches classiques cette organisation est le plus souvent mal adaptée pour la représentation explicite de certaines préoccupations des systèmes dites préoccupations transverses. La mise en œuvre de cette préoccupation donne souvent lieu à une dispersion et un enchevêtrement de code au sein de l'application qui rendent difficile entre autres, l'évolution du système et la réutilisation de ses composants. Par essence, le paradigme aspect permet la programmation séparée et modulaire des préoccupations transverses. Il propose en effet de décomposer les programmes non seulement en unités modulaires proposés aux préoccupations de base, mais aussi en unités modulaires spécifiques aux préoccupations transverse, De plus ce paradigme offre des moyens facilitant la composition et l'intégration des diverses unités afin de produire des systèmes mieux organisés plus compréhensibles et offrant une meilleure réutilisabilité[**Mostefaoui**, **2008**].

2.3 Les concepts de base du paradigme orienté aspect

Chaque nouveau paradigme propose un ensemble de nouveaux concepts. Le paradigme procédural apporte les notions de module, fonction et de procédure .Le paradigme objet apporte les notions de classe et d'héritage, d'objet, de méthode. À son tour, le paradigme aspect apport aussi des nouveaux concepts Ces concepts sont[**Amroune** , **2015**].

2.3.1 Préoccupation

une préoccupation est définie comme étant l'abstraction d'un but particulier ou une unité modulaire d'intérêt [Hachani , 2006].Un système typique contient essentiellement

deux catégories de préoccupation : Les préoccupation fonctionnelles et des préoccupation non fonctionnelles.[**AOUAG , 2014**].

Préoccupation de base

Elle représente donc une préoccupation non transversale (fonctionnelle) qui peut être capturée de manière efficace avec des approches traditionnelles telles que l'approche orientée objet.[**AOUAG , 2014**].

Préoccupation d'aspect

Elle représente donc une préoccupation transversale (non fonctionnelle) qui peut être capturée de manière efficace avec l'approche orientée aspect[**AOUAG , 2014**].

2.3.2 Aspect

L'apport principal de l'orienté aspect est le concept d'aspect lui-même. Un Aspect est une entité logicielle qui capture une fonctionnalité transversale[**Kiczales et al , 1997**].

2.3.3 Point de jonction (join point en anglais)

Endroit dans le modèle ou les conseils devraient être insérés.

2.3.4 Point de Coupe (pointcut en anglais)

Un ensemble de points de jonction, souvent une expression régulière et signifie le choix des points de jonction pour l'application de conseils.

2.3.5 Conseil (advice en anglais)

Une partie du modèle qui contient la totalité ou une partie de l'aspect inséré avant ou après ou autour d'un point de jonction implique une préoccupation transverse.

2.3.6 Tisseur (Weaver en anglais)

Est un outil spécial permettant d'intégrer ou de composer les aspects au modèle de base pour obtenir un modèle final (base+aspect).

2.4 Les techniques et les technologies orientée aspect

Le domaine du développement de logiciels orientés aspect intervient d'abord au niveau de la programmation à travers la conception de langage de programmation orienté aspect, Récemment, des techniques d'analyse et de conception orienté aspect ont émergé et l'attention est donnée aux techniques de vérification formelle portant sur les interactions et les interférences aspect. Nous allons donner une aperçue rapide sur ces techniques et technologies[**AOUAG , 2014**].

2.4.1 La programmation orientée aspect

L'approche orientée aspect a commencé d'abord à la programmation à travers la conception des langages de programmation orientée aspect. De ce fait, la plupart des idées sur les aspects sont perçues à la phase d'implémentation, la plupart de ces langages orientés aspects étendent des langages orientés objet existant par des caractéristiques orientés aspects pour représenter : les aspects, points de coupure advice,. . . etc. Le langage orienté aspect le plus important et maturé est probablement AspectJ, sans négliger d'indiquer d'autres langages orientés aspect importants comme Jasco, CaesarJ, AspectS, Object Teams, HyperJ, JBOSS, Compose*[**zerara**, **megrous**, **2020**].

2.4.2 L'ingénierie des exigences orienté aspect

Ces approches fournissent une représentation des préoccupations transverses dans les artefacts d'exigences. Elles reconnaissent explicitement l'importance d'identifier et de traiter les préoccupations transversales d'une manière précoce, les préoccupations transversales peuvent être des exigences non fonctionnelles aussi bien que des exigences fonctionnelles, leurs identifications précoces permettent l'analyse précoce de leurs interactions. Ces approches se concentrent sur le principe de la composition de toutes les préoccupations pour avoir le système complet en cours de construction[**Brichau**, **Hondt**, **2005**].

2.4.3 Les approches d'architecture orientée aspect

Les approches de conception d'architecture orientées aspect donc décrit les étapes d'identification des aspects architecturaux et leurs composants enchevêtrés, cette information est utilisée pour refaire une conception d'architecture donnée tout en mettant les aspects architecturaux explicites. Ceci est différent des approches traditionnelles où les aspects architecturaux sont une information implicite dans la spécification de l'architecture[**Schauer Huber et al**, **2011**].

2.4.4 Les approches de conception orientée aspect

Ces approches de conception se concentrent sur la représentation explicite des préoccupations transversales en utilisant des langages de conception adéquates. Dans les premiers temps de modélisation, les concepteurs utilisent simplement des méthodes et des langages orientées objet (tel que UML) pour la conception de leurs aspects, c'est le travail que nous ferons[**Loingtier**, **Irving**, **1999**].

2.4.5 Les approches de vérification et de tester des programmes orientés aspect

L'approche orientée aspect lève de nouveaux défis dans les techniques de vérification et validation des logicielles afin de s'assurer que la fonctionnalité désirée est satisfaite par le système. Des aspects peuvent potentiellement endommager la fiabilité d'un système auquel ils sont tissés, et peuvent rendre invalide des propriétés essentielles du système qui étaient correctes avant le tissage d'aspect[**Brichau**, **Hondt**, **2005**].

2.5 Modélisation orientée aspect

La programmation par aspects est encore jeune et nouvelle, elle ne dispose pas encore d'outils ou de langages de modélisation matures à l'instar de la programmation orientée objet. Entre autres, plusieurs chercheurs travaillent à l'intégration du paradigme aspect au niveau de la modélisation. Bien que plusieurs se soient penchés sur le sujet, aucun langage de modélisation pour le paradigme aspect ne s'est clairement démarqué. La plupart des travaux sur la modélisation par aspects s'inspirent très fortement de la modélisation par objets. Ainsi, ce nouveau paradigme hérité de tous les acquis de la technologie objet. L'un des acquis incontestable de l'objet est le langage unifié de modélisation par objets UML[**Object Management Group**, **2005**]. L'utilisation d'UML comme assise à la modélisation par aspects est très naturelle, sachant que la programmation par aspects se veut une extension voir une amélioration de la programmation par objets. La plupart des langages de programmation par aspects proposés actuellement sont tous construits autour de l'orienté objet : ils sont soit des extensions soit des évolutions des langages orientés objet[**Object Management Group**, **2005**].

2.5.1 Définition de la modélisation orientée aspect

La modélisation orientée aspect est une approche permettant d'atteindre ce but. Autrement dit le processus de tissage (la composition) d'aspects se divise en deux phases[**Aouag**, **2014**] . Tout d'abord, une phase de détection permettant d'identifier des parties particulières d'un modèle de base, puis une phase de composition permettant de construire le modèle prenant en compte l' aspect. En général, le cycle de développement de la modélisation orientée aspect se fait en trois [**AOUAG**, **2014**] :

La décomposition de l'aspect

Elle consiste à décomposer les besoins afin d'identifier et séparer les modèles transverses encapsulés dans des modules aspects non fonctionnels et métiers qui peuvent être modularisés dans des modules de base tel que : classe.[**AOUAG**, **2014**].

Implantation des modèles séparés

Elle consiste à implanter chaque modèle séparément. Les modèles métiers sont implantés par les techniques conventionnelles de la modélisation orientée objet alors que les modèles transverses sont implantés par les techniques de la modélisation orientée aspect[**AOUAG**, **2014**].

Recomposition de l'aspect

Elle consiste à construire le système final en intégrant ou recoupant les modèles métiers avec les modèles transverses. Cette phase est appelée tissage (Weaving en anglais)[**AOUAG**, **2014**]. Dans la Figure suivante[**AOUAG**, **2014**], nous présentons le processus de la modélisation orientée aspect tels que : S1, S2 et S3 sont des états, vérification et sécurité sont des aspects.

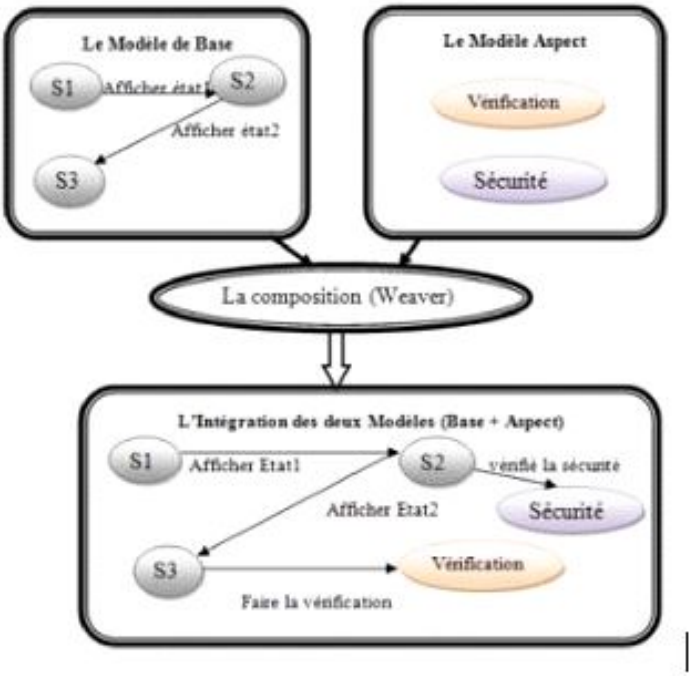

Fig. 2.1 : Le processus de la modélisation orientée aspect

2.5.2 La modélisation graphique en aspect

Dans le domaine de la modélisation orientée objet, UML fait quasiment l'unanimité aujourd'hui même si de nouveaux outils introduiront certainement leur propre notation d'ici quelque mois (Microsoft White House ,Certains outils MDA dédiés à un domaine particulier). C'est donc ce langage que nous avons choisi pour exprimer nos modèles de modélisation orientée aspect. [**Gil , 2006**]. Dans la figure suivante nous présentons la modélisation graphique d'un aspect à travers un exemple expliquer la notion complètement , qui permet de bien distinguer les classes cibles, les aspects et l'opération de tissage l'exemple suivant représente un tissage dans lequel :

1. un attribut est introduit dans plusieurs classes cibles.
2. le corps d'une méthode est inséré au début de toutes les méthodes des classes cibles[**Gil , 2006**].

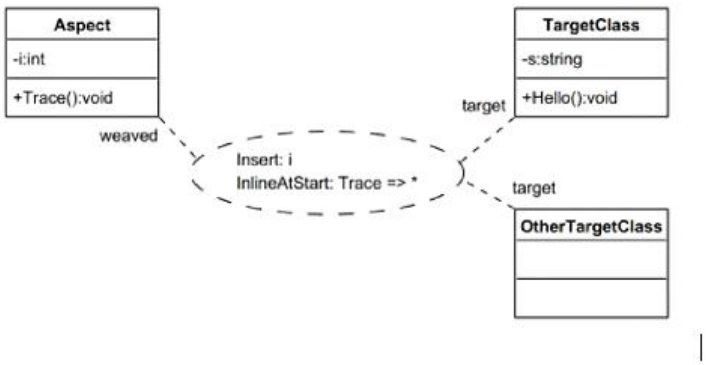

FIG. 2.2 : Notion Complet

2.6 Les approches de la modélisation orientée aspect

De manière générale, il existe trois approches de la modélisation orientée aspect : l'extension UML léger à travers des profils, l'extension UML lourd à travers la Méta modélisation et le tissage de modèles explicites. Les deux premières approches tentent de pousser la sémantique du tissage dans un Méta-modèle UML étendu. La troisième approche reconnaît le tissage comme un concept de Méta-modélisation. Enfin, la relation entre le tissage du modèle et l'architecture pilotée par le modèle est discutée[**Cottenier et al**, **2005**]

2.6.1 Extension UML léger

L'extension légère aux approches UML tire parti des mécanismes d'extension UML pour affiner le méta modèle UML comme pour soutenir les constructions de langage Aspect Oriented Programming[**Cottenier et al**, **2005**]. Ces approches fournissent des notations

graphiques pour les aspects, par exemple en définissant des relations de type point de jonction, qui relient les conseils aux points de jonction. Ces approches couplent étroitement les aspects au modèle de base. Pour les grands projets, les relations peuvent impliquer de nombreuses classes de l'application, et la représentation visuelle de la coupure transversale n'est pas bien adaptée.Le résultat de la composition est complexe et difficile à lire et à comprendre. Le modèle de conception ressemble à un modèle tissé[**Cottenier et al**, **2005**]. La position des auteurs est que les points de jonction sont un problème de validation et de vérification. Cela implique que pendant la phase de conception, nous ne voulons pas savoir où se trouvent les points de jointure, et par conséquent, nous n'avons pas besoin d'une construction de modélisations explicite pour eux[**Cottenier et al**, **2005**].

2.6.2 Extension UML lourd

Parce que les mécanismes d'extension légers ne sont pas assez expressifs pour capturer seuls la sémantique du tissage, les partisans de l'extension lourde de l'UML soutiennent que les aspects doivent entrer des éléments de première classe dans l'UML. Les approches d'extension UML lourdes définissent un méta modèle autonome qui s'adapte à Aspect Oriented Softwar Development(AOSD) .Le méta modèle est obtenu en adaptant le méta modèle du noyau UML. Les extensions lourdes fournissent des notations graphiques pour les aspects, mais ne définissent pas explicitement le concept de tissage[**Cottenier et al**, **2005**].

2.6.3 Approches de la transformation des modèles

Les approches de transformation de modèle reconnaissent le processus de tissage comme un concept dans le méta modèle étendu d'UML. Les approches de tissage de modèles réalisent la coordination de préoccupations transversales avec un modèle de base grâce à la transformation du modèle. Le tissage de modèles permet de décrire l'essence d'une préoccupation dans un modèle distinct et de transformer d'autres modèles en conséquence. Ces approches ont l'inconvénient que les constructions de langage Aspect Oriented Programming n'ont pas d'équivalent clair au niveau de la modélisation[**Cottenier et al**, **2005**].

2.7 L'inconvénient de la modélisation orientée aspect

Un inconvénient des mécanismes de composition flexibles dans la modélisation orientée aspect est que la compréhension locale est réduite. La compréhension détaillée des différentes facettes de l'être dans le système composé est difficile parce que les détails de la composition sont cachés du spectateur. Par conséquent, il est difficile d'assurer l'absence de conflits dans le système compose[**Zerara** , **Megrous**, **2020**]. Cet inconvénient réduit les avantages d'une structure modulaire améliorée facilitée par la modélisations orientée aspect

2.8 Conclusion

Dans ce chapitre nous avons présenté brièvement l'oriente aspect. Pour cela nous avons commencé par le paradigme orienté aspect, les concepts de base du paradigme aspect, par la suite nous avons vu les techniques et les technologies de développement oriente aspect à travers le cycle de vie de développement logiciel, puis la modélisation orientée aspect avec sa définition. les modèles en orienté aspect, elles manquent des outils de vérification de comportement, qui soient disponible dans les réseaux de petri "notre modèle cible", qui est défini dans le chapitre suivant.

Chapitre 3

Réseau De Pétri

3.1 Introduction

Du fait de la complexité de plus en plus forte des systèmes technologiques, il apparaît de plus en plus nécessaire de disposer des méthodes et d'outils de conception, de réalisation qui sont particulièrement efficaces, au centre de ces méthodes et de ces outils, se trouve en générale la modélisation des processus. Parmi le grand nombre des techniques formelles qui ont déjà été proposées, les réseaux de pétri. Dans ce chapitre nous intéressent par la présentation des Rdp , donc dans un premier temps nous présentant les concepts de base d'un Rdp, dans les titres qui suive nous parlons brièvement de l'évolution d'un Rdp, les sous-classes d'un Rdp, ensuite les extension des Rdp, les méthodes d'analyse, et enfin la modélisation avec Rdp, les principales propriétés des Rdp, et l'utilisation des Rdp .

3.2 Historique

Le concept de réseau de pétri a sous origine la thèse de Carl Adam Petri[**Pétri**, **1992**],présentée en 1962 a la faculté de mathématique et de physique à l'université technique de Darmstadtium ,en ouest Allemagne ,il est caractérisé par son aspect graphique qui nous aide à comprendre facilement le système modélise ,il permet de simuler les activités dynamique ,concurrent et parallèles,son aspect mathématique permet d'analyser le système modélisé grâce au outils d'analyse automatique[**David et al**, **2004**].

3.3 Réseau de Pétri

Il est clair que les systèmes dynamiques ne peuvent pas être décrits en se référant seulement à leurs états initiaux et finaux. Une description adéquate doit tenir compte de leur comportement permanent qui est une séquence (peut-être infinie) d'états. Plusieurs techniques formelles ont déjà été proposées pour spécifier, analyser et vérifier ce genre de systèmes. Les réseaux de Petri représentent une technique formelle largement utilisée. En tant qu'outil mathématique, ils permettent l'analyse des propriétés du système concernant sa structure et son comportement. Ce formalisme bénéficie d'une riche panoplie de techniques d'analyse et d'outils. Les résultats de cette analyse sont utilisés pour évaluer le système et en permettre la modification ou l'amélioration. La figure suivante montre les méthodes générales basées sur le formalisme des réseaux de pétri pour la modélisation et l'analyse des systèmes[**Kerkouche**, **2011**]. Grâce à leur généralité et à leur souplesse, 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[**Kerkouche**, **2011**].

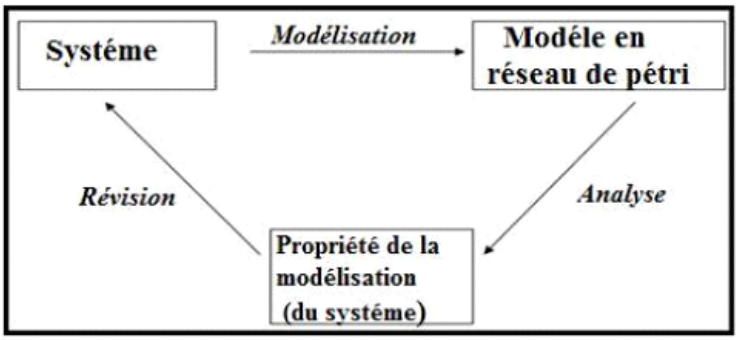

FIG. 3.1 : Méthodes générale de modélisation et d'analyse basée sur les Rdp

3.4 Concepts de bases de réseau de pétri

3.4.1 Définition graphique

Un réseau de Petri (RDP) est un graphe biparti orienté valué , il a deux types de nœuds[**Murata**, **1989**] :

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.

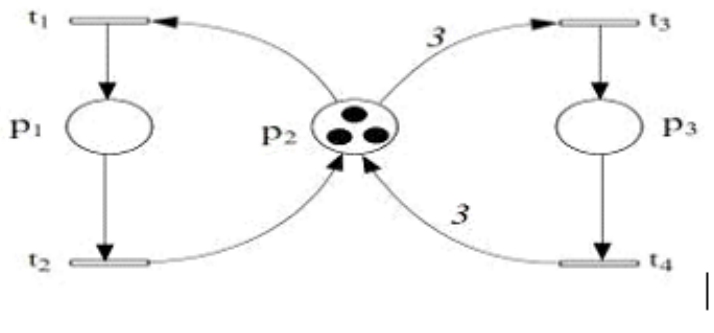

FIG. 3.2 : Exemple d'un Résau de Pétri

Les réseaux de pétri offrent une description d'un système en faisant une distinction claire entre l'aspect statique et l'aspect dynamique, telle que sa représentation graphique facile et compréhensible par les utilisateurs. L'analyse d'un réseau de pétri peut énoncer des caractéristiques importantes du système concernant sa structure et son comportement dynamique .Les résultats de cette analyse sont utilisé pour évaluer le système et en permettre la modification ou l'amélioration [**Aouag**, **2014**]

3.4.2 Définition formelle

Formellement, Un réseau de petri marqué est représenté par 5-uplet, Rdp = P, T, F, W, M0, ou [**Murata**, **1989**] :

1. **P**= p1, p2, p3,… pm, est un ensemble fini non vide de places p.
2. **T**= t1, t2, t3,… tn, est un ensemble d'arcs ou :
   (p * t) est l'arc allant de P à T.
   (t * p) est l'arc allant de T à P.
3. **W** : F → (1, 2,3,….) : est une fonction du poids ou :
   W (p, t) :''pré (p, t)'' : est le poids de l'arc allant de P à T.
   W(t, p) :''post (p, t)'' : est le poids de l'arc allant de T à P.
4. **M**0: p → (0,1,2,3,…)est le marquage initial.
5. **P** ∩ T = ∅ et P ∪ T = ∅

3.4.3 Représentation Matricielle

Une représentation matricielle d'un RdP est offerte afin de simplifier les tâches d'analyse et de vérification effectuées sur un modèle RdP .Agir sur une représentation graphique d'un modèle Rdp est une tâche délicate en la comparant avec une représentation matricielle. il est possible de représenter la fonction ϖ (fonction de poids) par des matrices[**El Mansouri**, **2009**],D'autre part un Rdp R=(P,T,W) avec P=p1,p2,…..,pn et T=t1,t2,..,tm,on appel matrice des pré-condition ,la matrice m*n a coefficients dans N telle que pré(i,j)= ϖ(i,j) indique le nombre de marques que doit contenir la place pi

pour la transition tj devient franchissable ,De la même manière on définit la matrice des post-condition ,la matrice n*m telle que post(i,j)= ϖ(tj,pi) contient le nombre de marque déposer dans pi lors du franchissement de la transition tj,la matrice C=post – pré ; est appelée matrice d'incidence du réseau(m représente le nombre de places d'un Rdp et n le nombre de transition).Le marquage d'un Rdp est représenté par un vecteur de dimension m a coefficients dans n , la règle de franchissement d'un Rdp est définie par : M'(p)=M(p)+C(p,t). La figure suivante représente un Rdp marqué avec un vecteur de marquage M telle que :M=(1,0,0,0).

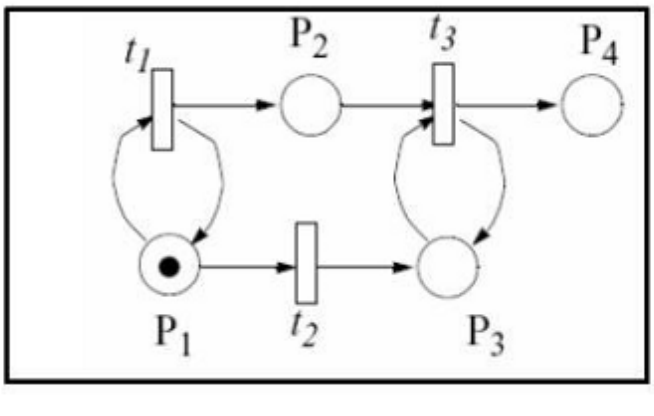

Fig. 3.3 : Un Rdp Marqué.

Pour le réseau ci-dessus P = p1, p2, p3, p4. T = t1, t2, t3 .la représentation matricielle est donné ci-dessus du Rdp de la figure 3

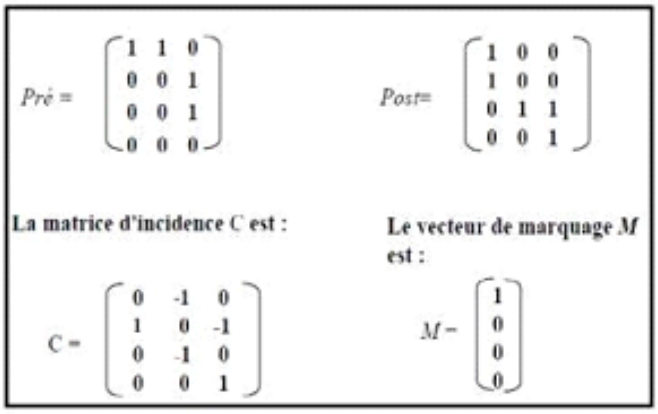

Fig. 3.4 : Matrice d'incidence.

3.4.4 L'arbre de recouvrement (le marquage d'un Rdp)

L'ensemble des marquages accessible à partir du marquage initial M0 peut être représenté sous la forme d'un arbre que nous appelons arbre des marquages atteignables. Dans cet arbre les nœuds représentent les marquages , les arcs et les transitions franchies. En

partant de M0, nous représentons tous les nouveaux marquages obtenus par le franchissement de chacune des transitions validées par M0 .On répète cette procédure à partir de chacun de ces marquages pour atteindre des nouveaux marquages, et ainsi de suite [**Savi**, **1994**]. La figure suivante donne un exemple d'un Rdp marqué et de son arbre de marquage atteignable.

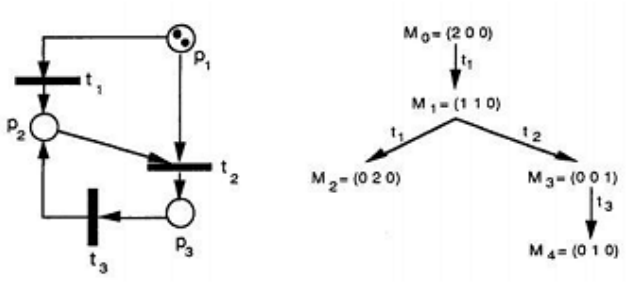

FIG. 3.5 : un exemple d'arbre des marquages atteignables.

Dans l'exemple ci-dessus, l'arbre obtenu est fini puisque le nombre de marquages atteignables est limité, mais ce n'est pas toujours le cas. Un Rdp avec un nombre infini de marquages atteignables aura un arbre fini, Même dans le cas où le nombre de marquages atteignable est limité l'arbre obtenu peut être infini. Pour résoudre ce problème nous allons construire un autre arbre, nommée arbre de recouvrement, lequel sera toujours fini , par construction l'algorithme pour l'obtention d'un arbre de recouvrement est présenté ci-dessus .Dans cet algorithme le symbole « ϖ » est utilisé pour représenter un marquage non borné[**Savi**, **1994**]

Algorithme 1

1. le marquage initial M0 comme la racine de l'arbre.
2. Tant que des marquages nouveaux existant faire :
3. choisir un nouveau marquage M.
4. Si M a été déjà trouvé sur le chemin conduisant de la racine M0 jusqu'à M, marque M par 'déjà trouvé' .on ne poursuivra pas les recherches à partir de ce nœud.
5. Si aucune transition ne peut être franchie à partir de M, marque M par 'blocage'.
6. Pour toute transition T franchissables à partir de M faire :
7. calculer le marquage Mt obtenu par franchissement de T.
8. si, sur le chemin reliant M0 a Mt, il existe un marquage M* tel que Mt(p)>= M*(p) quelle que soit la place p est si Mt(p)>M*(p), pour au moins une place p, alors marqué par « ϖ » tous les éléments de Mt correspondant aux places p telles que Mt(p)>M*(p)

9. conserver Mt comme nouveau sommet de l'arbre (i.e.: nouveau marquage) et établir l'arc (M, Mt) marque t.

Un exemple d'application de cet algorithme est présenté dans la figure suivante :

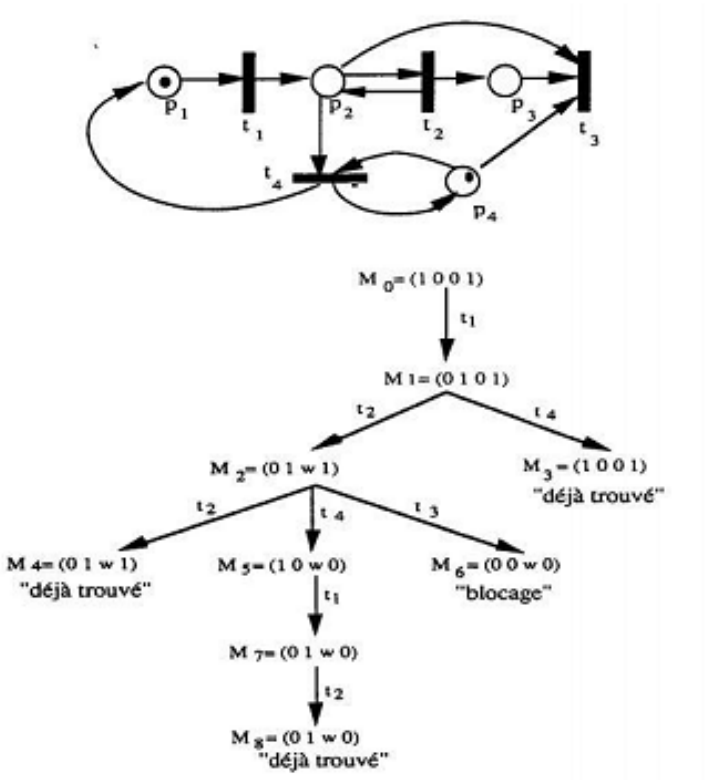

Fig. 3.6 : un exemple d'arbre de recouvrement.

Remarque

L'utilisation du symbole « ϖ » dans cet algorithme provoque une perte d'information du fait que l'arbre ne contient pas tous les marquages qu'il est possible d'atteindre à partir de M0. Une conséquence directe de ceci est que deux Rdp différents peuvent avoir le même arbre de recouvrement (voir l'exemple présent dans la figure 6). L'arbre de recouvrement est l'outil de base pour la vérification des propriétés d'un Rdp.

3.5 Évolution d'un Rdp

L'évolution d'un Rdp correspond à l'évolution du son marquage au fil du temps (évolution de l'état du système), il se traduit par un déplacement des jetons pour une transition t de l'ensemble des places d'entrée vers l'ensemble des places de sortie de cette transition ce déplacement s'effectue par le franchissement de la transition t selon des règles de franchissement [**El Mansouri**, **2009**] .

3.5.1 Transition validée

L'évolution de l'état du Rdp correspond à une évolution du marquage .Les jetons, qui matérialisent l'état de réseau à un instant donné, pouvant passer d'une place à l'autre par franchissement ou tir d'une transition. Une transition est validée (ou dit aussi sensibilisée ou franchissable ou encore tirable) si toutes ses places d'entrée contiennent au moins un jeton[**Torchi**, **Mezahi** , **2018**].

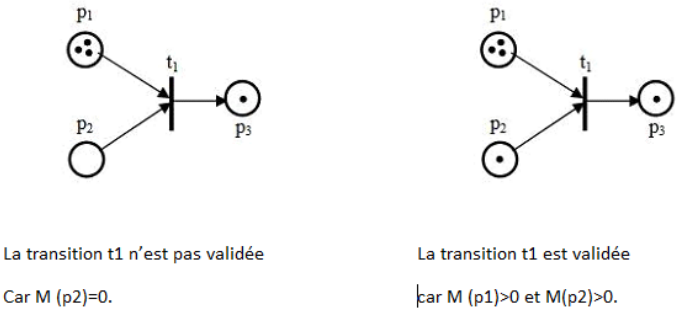

Fig. 3.7 : Transition validée.

3.5.2 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 [**Murata**, **1989**].

Fig. 3.8 : franchissement d'un Rdp.

« Le franchissement M1=[3 1 1], transition t1 à partir du marquage M1 conduit at : M2=[2 0 2] »,se note : M1(t1 →)M2.

3.5.3 Règle de franchissement

Le franchissement consiste à :

1. **R**etirer ϖ(p, t) jeton dans chacune des places en entrée P de la transition T.

2. **A**jouter ϖ(t, p) jetons à chacune des places en sortie P de la transition T.

La règle de franchissement est illustrée par la figure 3.9, en utilisant la réaction Chimique connue [**Murata**, **1989**]
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 franchisable, 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.

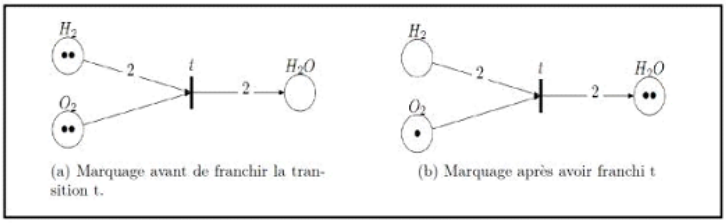

Fig. 3.9 : exemple de règle de franchissable de transition.

3.6 Sous-Classes des Rdp

Cette section est consacrée à une description succincte de quelque réseaux de pétri a structure particulière, d'autres sous-classes et des extensions des Rdp, peuvent être trouvées dans la bibliographie fournie au début de ce chapitre.

3.6.1 Machine á état et graphes d'événements

Une machine à état est un Rdp dans lequel chaque transition a exactement une place d'entrée et une place de sortie. Un graphe d'événements correspond au dual d'une machine à états : c'est un Rdp dans lequel chaque place a exactement une transition d'entrée et une transition de sortie. Un exemple de chacune de ces deux ces classes de Rdp est présenté dans la figure suivante[**Savi**, **1994**].

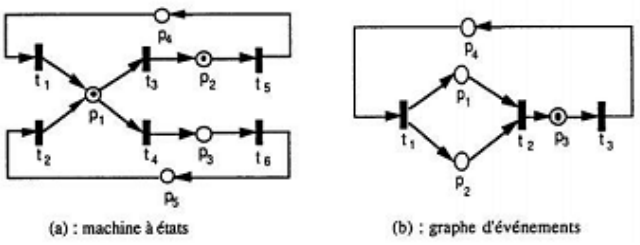

Fig. 3.10 : exemple d'une machine à états et d'un graphe d'événement.

Comme en peut le constater dans les exemples de la figure ci-dessus un réseau de type machine à états permet la représentation de décision (voir place p1 dans la figure 10) mais ne permet pas la représentation de la synchronisation des activités parallèles .Un graphe d'événements permet par contre de représenter la concurrence (voir place t2 dans la figure 9), mais non les décisions[**Savi**, **1994**].

3.6.2 Rdp sans conflit

Une place P ayant au moins deux transition de sortie constitue ce qu'on appelle un conflit structurel (aussi appelé décision ou choix).On représentera ceci par un doublet forme d'une place et de l'ensemble de ses transition de sortie <p1, t1, t2,….> Un conflit représente la possibilité d'une exclusion mutuelle entre déférent événement du système. Un conflit structurel devient un conflit effectif quand le marquage est tel que le nombre de marquages dans la place P est inférieur à la somme des marquages des arc conduisant aux transition de sortie de cette place, ceci implique donc la nécessité d'un choix des transitions à franchir. Dans la figure 10 le doublet<p1,t3,t4> est un conflit effectif. Un Rdp sans conflit, comme l'indique son nom : est un Rdp dans lequel ,il n'y a pas de conflit structurel[**Savi**, **1994**].

3.6.3 Rdp pur

Un circuit élémentaire constitué d'une seule transition et d'une seule place est appelée une boucle élémentaire ou plus simplement boucle, Une telle boucle se caractérise par le fait que la transition possède une place d'entrée qui est également une place de sortie de cette transition[**Savi**, **1994**]. Un Rdp est pur s'il ne contient pas de boucle élémentaire, une propriété intéressante des Rdp est qu'il est toujours possible de transformer un réseau impur en réseau pur en ajoutant dans chaque boucle élémentaire une transition et une place[**Savi**, **1994**].

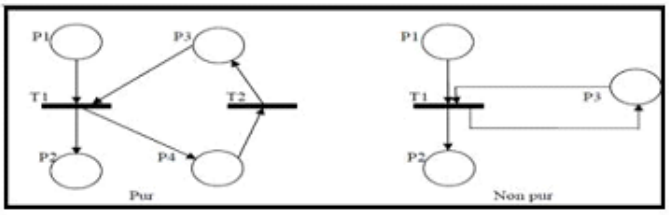

Fig. 3.11 : Rdp Pur

3.6.4 Rdp choix libre

On dira qu'un Rdp est à choix libre si pour tout conflit <p1, t1, t2, …>, P1 est la seule place d'entrée de l'ensemble de transition t1, t2,….,il s'agit donc d'un cas ou la présence d'une marque dans la place p1 autorise le franchissement de toutes ses transition de sortie (le choix de la transition à franchir rentablement)[**Savi**, **1994**]. Cette définition peut être généralisée à la classe des réseau à choix libre étendu de la façon suivante : un Rdp est à choix libre étendu si pour tout conflit <p1,t1,t2,…> tous les transitions t1,t2,… ont le même ensemble de places d'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[**Savi**, **1994**].

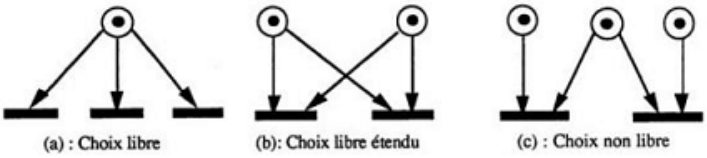

FIG. 3.12 : réseau à choix libre, réseau à choix libre étendu et réseau à choix non libre.

3.6.5 Rdp simple

Les Rdp simple sont des Rdp ordinaire tel que chaque transition a au plus une place d'entrée qui peut être reliée a d'autre transition (à tous transition appartient à un seul conflit au plus), tel qu'il est présenté dans la figure suivante[**Savi**, **1994**].

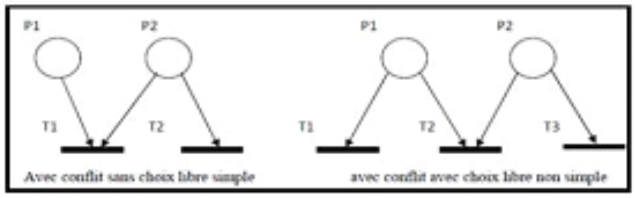

FIG. 3.13 : Rdp simple.

3.6.6 Rdp généralisés

Un Rdp généralisé est un Rdp dans lequel des poids (nombre entier strictement positif) 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 ajoute K jetons à la place Pi. Lorsque le poids n'est pas signalé, il est égale à un par défaut .Comme il est exprimé dans la figure suivante[**Laouar**, **2013**]

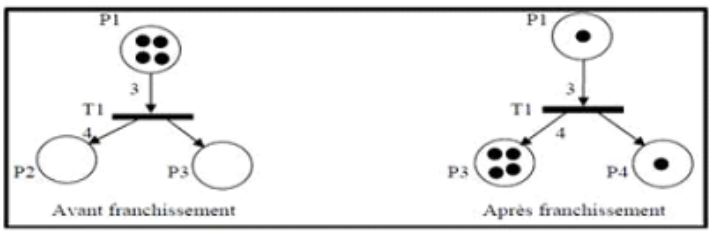

FIG. 3.14 : Rdp généralisé.

3.6.7 Rdp á capacité

Un Rdp á capacité 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 dans la capacité cap (pi) n'est pas possible que si le franchissement ne conduit pas à un nombre de jetons dans Pi qui est plus grand que cap(Pi)[**Laouar**, **2013**]. La figure suivante montre le franchissement de Ti conduit a 3 jetons dans P2 d'où T1 ne peut plus être franchie[**Laouar**, **2013**].

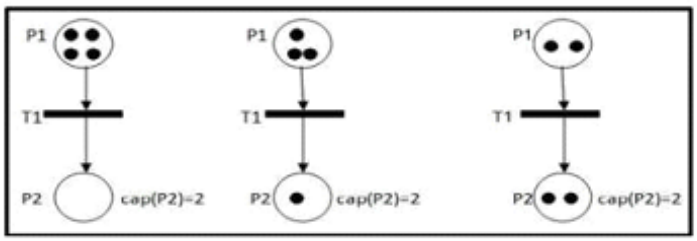

Fig. 3.15 : Rdp á capacité

3.6.8 Rdp 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[**Laouar**, **2013**].

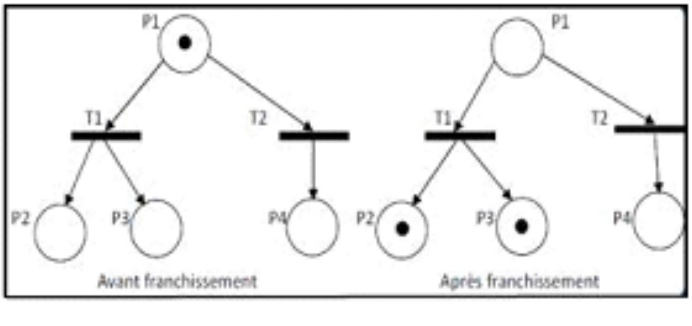

Fig. 3.16 : Rdp á priorité

3.7 Extension des réseaux de pétri

Les systèmes réels imposent beaucoup de contraintes, leur modélisation peut donc engendrer des réseaux de pétri de taille importante dont la manipulation et l'analyse est très difficile. En plus, les Rdp usuels ne permettent pas d'exprimer certaines propriétés, tel que la mobilité, rendant ainsi leur analyse assez difficile. Dans ce contexte, plusieurs extensions de réseaux de pétri ont vu le jour, entre autres on a : les réseaux de pétri colore, les réseaux de pétri temporisés etc. Dans la section suivante nous présentons un aperçu de quelques extensions des Rdps [**Dehimi**, **2014**]

3.7.1 Les Rdp Colorés

Lorsque le nombre d'entités du système à modéliser est important, la taille du réseau de pétri devient rapidement énorme, et si les entités présentent des comportements similaires, l'usage des réseaux colorés permet de condenser le modèles réseaux de petri colorés sont des réseaux de pétri dans lequel les jetons portent des couleurs. Une couleur est une information attachée à un jeton. Cette information permet de distinguer des jetons entre eux et peut être de type quelconque. Ainsi les arcs ne sont pas seulement étiquetés par le nombre de jetons mais par leurs couleurs. Le franchissement d'une transition est alors conditionné par la présence dans les places en entrée du nombre de jetons nécessaires, qui en plus satisfont les couleurs qui étiquettent les arcs. Après le franchissement d'une transition, les jetons qui étiquettent les arcs d'entrées des places en entrée tandis que ceux qui étiquettent les arcs de sortie sont ajoutés aux places en sortie de cette transition. Les réseaux colorés n'apportent pas de puissance de description supplémentaire par rapport aux réseaux de petri, ils permettent juste une condensation de l'information. A tout réseau de petri coloré marqué correspond un réseau de pétri qui lui est isomorphe. La relation entre le Rdp colore et le Rdp ordinaire est vu comme une relation entre le langage 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 rapport au langage assembleur, car il est bien structuré, bien typé et modulé[**Hettab**, **2009**]. Le passage de Rdp au Rdp colore est appelé pliage et le passage du Rdp colore au Rdp est dépliage.

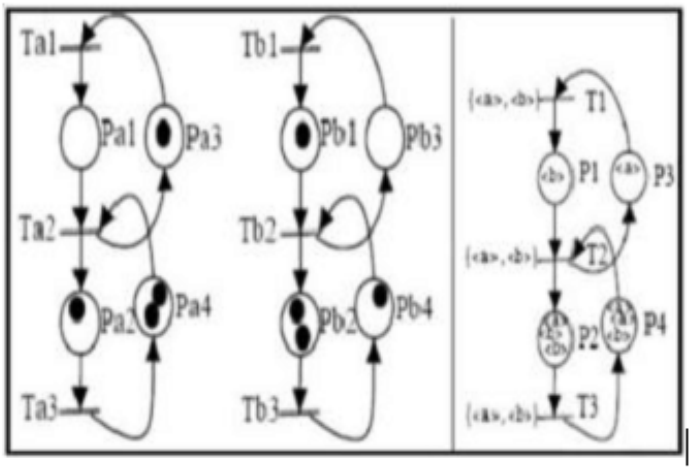

Fig. 3.17 : Rdp(gauche) et Rdp coloré(droite).

3.7.2 Rdp temporisés

Dans les Rdp que nous avons présentés jusqu'ici nous considérons que tous les franchissements sont exécutés immédiatement. Aucune durée n'est associée à un franchissement. Cependant le fonctionnement de certains systèmes dépend du temps. Par exemple, dans le cas d'un système de production, le modèle doit refléter la durée des opérations, des transports, des changements d'outils, etc. Pour pouvoir modéliser de tels systèmes nous allons utiliser les RDp temporisés. Deux types de modèles existent : celui dans lequel

les temporisations sont associées aux places, et celui dans lequel les temporisations sont associées aux transitions[**Savi**, **1994**].

Temporisations associées aux places

Dans ce type de modèles une temporisation θ, éventuellement nulle, est associée à chaque place P .Dès qu'une marque arrive dans la place P, elle reste indisponible pour un temps θ, à la fin duquel elle devient disponible et peut être utilisée dans le franchissement des transitions de sortie de cette place[**Savi**, **1994**]. A un instant τ quelconque le marquage M d'un Rdp avec temporisation associée aux places est l'union du marquage Md, correspondent aux marques disponibles, et du marquage Mi, correspondent aux marquages indisponibles. Une transition est validée pour un marquage M=Md ∪Mi, si elle est validée pour le marquage Md .Comme dans le cas des Rdp non-temporisés, le franchissement d'une transition t a une durée nulle. Cependant ,les conséquences du franchissement de t sont un peu différentes dans le cas des Rdp temporisés : celui-ci provoque l'enlèvement des marques disponibles de chacune des places d'entrée de t et le dépôt des marques indisponibles dans chacune des places de sortie de t. Ces dernières deviendront disponibles après l'écoulement du temps correspondant à la temporisation de chaque place de sortie et pourront, à ce moment l'être utilisées dans la validation des transitions[**Savi**, **1994**].

Temporisation associée aux transitions

Une autre forme de temporisation consiste à associer une durée positive ou nulle pour le franchissement de chaque transition. On peut montrer que ce modèle est équivalent au modèle précédent, c'est-à-dire qu'il est possible de passer de l'un à l'autre. Si θ est la temporisation liée à une transition validée t et τ0l'instant où son franchissement débute. Alors le franchissement de t se terminera à l' instant τ=τ0+0. Les instants τ0 et τ sont appelés respectivement, début de franchissement de t et fin de franchissement de t, la seul action réalisée au début de franchissement. La transition t est effectivement franchie quand on atteint l'instant de fin de franchissement, et ceci se traduit par l'enlèvement des marques réservées dans les places d'entrée et le dépôt de marques non réservées dans les places de sortie de t. Comme précédemment, on dira que le marquage M est composé de marques réservées et de marques non réservées. Une transition est validée pour un marquage M si et seulement si elle est validée pour le marquage correspondant aux marques non réservées[**Savi**, **1994**].

3.7.3 Rdp 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. Dans un Rdp synchronisé, une transition validée n'est pas forcément franchissable. 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[**Dehimi**, **2014**].

3.7.4 Rdp stochastique

Les Rdp stochastiques (Rdps) ont été développés. En effet, en introduisant une temporisation dans les Rdp, le but est de concevoir un modèle unique permettant à la fois une validation qualitative et quantitative de réseaux. Un Rdps est un Rdp temporisé doté d'une mesure de probabilité sur l'espace des trajectoires, c'est-à-dire que les séquences de franchissement sont mesurables en considérant un espace aléatoire. De façon formelle[**Hettab**, **2009**] : Un Rdps est un couple S=<R, >, tel que :

1. R=<P, T, Pre, Post, M0> est le réseau sous-jacent.
2. ϕ : T $\rightarrow$ R+, la fonction qui associe à chaque transition un taux de franchissement fixe.
3. M0: marquage initial du réseau.

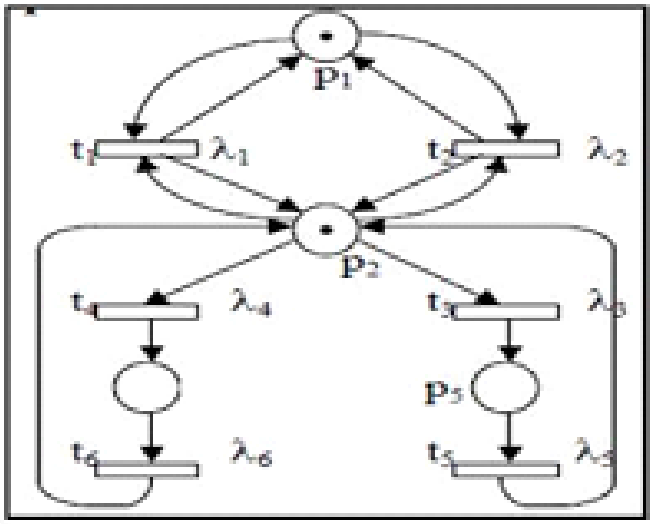

Fig. 3.18 : Rdp stochastique.

3.7.5 Rdp 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[**Torchi et al**, **2018**].

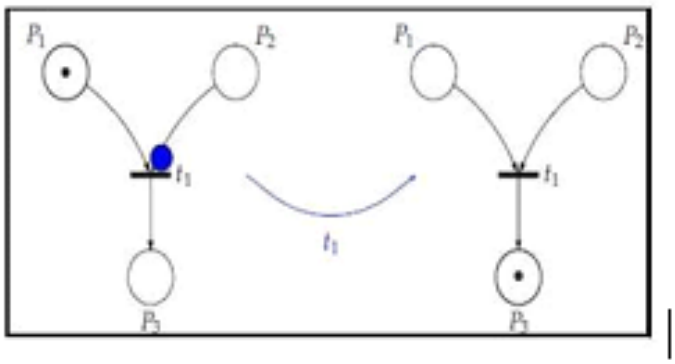

FIG. 3.19 : Rdp inhibiteur.

3.8 Methods d'analyses

Dans cette section nous faisons quelques commentaires sur des méthodes disponibles pour la vérification des propriétés d'un Rdp. Pour une description plus détaillée, le lecteur pourra consulter les références fournies au début de ce chapitre ou celles présentées tout au long de cette section[**Murata**, **1989**].

3.8.1 Arbre de recouvrement

L'arbre de recouvrement et la technique d'analyse des propriétés d'un Rdp la plus utilisée, l'inscription directe de l'arbre de recouvrement permet de mettre en évidence certaines propriétés d'un Rdp. Si on examine un arbre construit selon l'algorithme présenté dans la section 4, on peut dire entre autre que :

le Rdp est sauf si et seulement si tous les marquages ne contiennent que des 0 et des 1.

une transition t est morte (i.e. jamais franchie) si aucune des arcs de l'arbre de recouvrement n'est pas marqué.

le Rdp est bornée si et seulement si le symbole « ϖ » n'apparaît dans aucune des marquages correspondant à un nœud de l'arbre. L'inconvénient principal de cette technique réside dans le fait qu'elle est une méthode de recherche exhaustive et par conséquent limitée à des Rdp de petite taille.

3.8.2 Algèbre linéaire

Comme nous l'avons souligné dans les paragraphes précédents, un Rdp peut être représenté sous forme matricielle cette écriture permet d'utiliser des résultats de l'algèbre linéaire pour obtenir certains résultats sur les Rdp. Par exemple, pour vérifier si un réseau est vivant, consistant et borné nous utilisons les relations données dans la suite. Dans ce relation c'est la matrice d'incidence m est le nombre de transition et n le nombre de places du réseau .Nous considérons aussi qu'un vecteur x est supérieur à zéro si et seulement si toutes ses composantes sont supérieures à zéro :

1. vivacité :$\exists$ x $\in$ Rm, x> 0, tel que c*x >= 0
2. consistance :$\exists$ x $\in$ Rm, x>0, tel que c*x=0
3. réseau borne :$\exists$ y $\in$ Rn, y>0, tel que ytr *c<=0

Cependant dans le cas général deux problèmes majeurs se posant pour la résolution des équations associées a une certaine propriété :

1. le fait que l'on est contraint de trouver des solutions entières non nulles.
2. La possibilité d'obtenir des solutions non significatives pour la propriété recherchée car certaines relations donnent des conditions qui sont des conditions nécessaires mais non suffisantes.

Pour plus détaillés concernant l'utilisation de l'algèbre linéaire dans la vérification des propriété d'un Rdp, le lecteur peut consulter [**Silva**, **1980**][**Lautenbach**, **1986**][**Martinez**, **1985**].

3.8.3 Réduction du réseau

La réduction ne constitue pas une méthode de vérification des propriétés, mais une technique qui fournit un modèle de taille réduit laquelle préserve certaines propriétés qualitatives du réseau original, ceci rend l'analyse par les méthodes précédente plus facile[**Hala**, **2004**]

3.9 Modélisation Avec les Rdp

Les Rdp ont été conçus et utilisés principalement pour la modélisation. Plusieurs systèmes peuvent être modélisés par les Rdp, ces derniers peuvent être de natures très diverses : matériel informatique, logiciels informatiques, les systèmes physiques, les systèmes sociaux, etc[**Peterson**, **1981**]. Dans les sections suivantes nous donnerons quelques exemples de modélisation des problèmes informatiques et mathématiques avec les Rdp.

3.9.1 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)

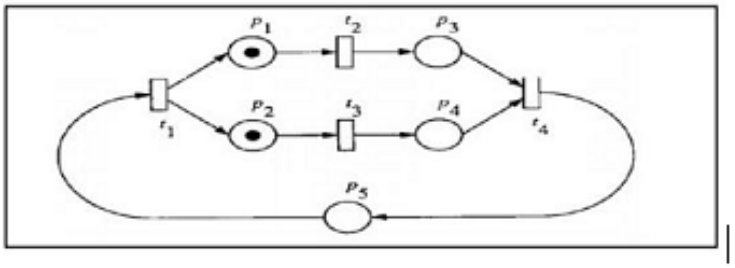

FIG. 3.20 : parallélisme dans les Rdp.

3.9.2 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[**Peterson**, **1981**].

Exemple 1: problème du producteur/consommateurs

Le problème de producteur/consommateurs implique un tampon partage. Le processus producteur crée des objets qui sont mis dans le tampon, le consommateur attend jusqu'à ce qu'un objet soit mis dans le tampon pour le consommer. Cela peut être modélisé comme le montre la figure suivante. La place B représenté le tampon, chaque jeton représente un élément qui a été produit mais pas encore consommé[**Peterson**, **1981**].

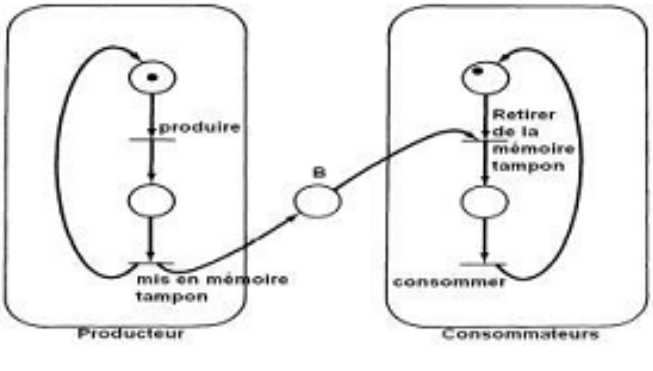

FIG. 3.21 : Problème du producteur et consommateurs.

Exemple 2: Exclusion mutuelle

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[**Peterson**, **1981**].

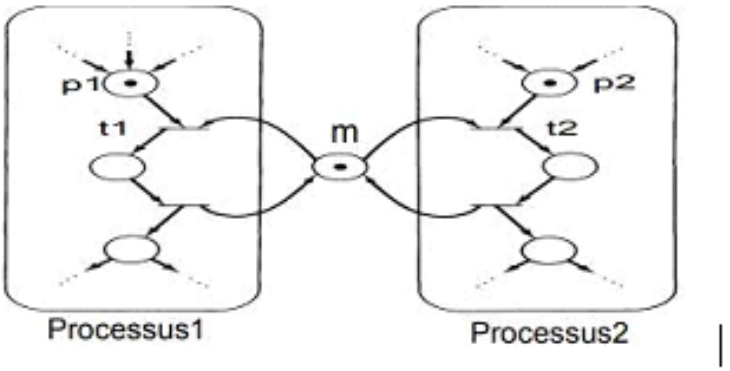

Fig. 3.22 : Exclusion mutuelle .

3.9.3 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 : [**Murata**, **1989**]

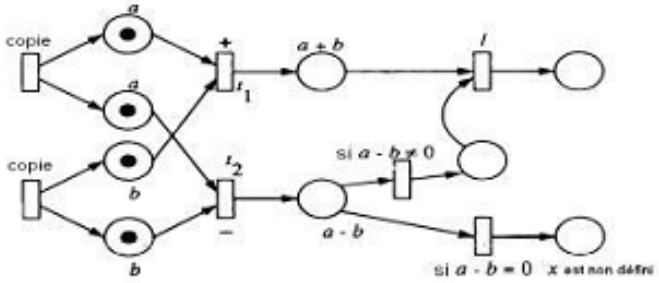

Fig. 3.23 : Exemple d'un calculé de flux de données par un Rdp.

3.10 Principales propriétés des Rdp

3.10.1 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[**V. Murilo Savi**, **1994**].

3.10.2 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[**Murilo Savi**, **1994**]

3.10.3 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 ∈ 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[**Savi**, **1994**]

3.10.4 Persistance

Un Rdp est persistant pour un marquage initial M0 si quel que soit le marquage accessible M R(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[**Savi**, **1994**].

3.10.5 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[**Savi**, **1994**].

3.11 Utilisation des Rdp

Dans certains cas, les Rdp ordinaires ne peuvent exprimer toutes les propriétés que nous voudrions modéliser, et de ce fait des extensions s'avèrent utiles afin de pallier ces insuffisances. Parmi les extensions les plus utilisées, nous citons :

1. Les Rdp généralisent : Un Rdp généralisé est un Rdp dans lequel des poids (nombres entiers strictement positifs) sont associés aux arcs.
2. Les Rdp temporisées : C'est une extension temporelle des Rdp.
3. Les Rdp colorées : Dans un Rdp on associe une valeur à chaque jeton.
4. Les Rdp continus : Le nombre de jetons dans un Rdp 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[**Torchi**, **Mezahi**, **2018**].

Quelque domaine d'utilisation des Rdp, non seulement informatiques, ont été résumées dans la table suivante :

Réseau de petri ordinaire	-Modélisation des systèmes. -Modélisation des processus d'affaires. -Gestion des flux. -Programmation concurrente. -Génie de la qualité. -Diagnostics.
Réseau de petri généralisé	-Gestion des flux complexes -Modélisation de chaîne logistique. -Utilisation pour les techniques qualitatives. -
Réseau de petri temporisé	-Gestion du temps. -Modélisation d'attentes.
Réseau de petri coloré	-Modélisation des systèmes de collaboration.
Réseau de petri continu	-Modélisation des réactions chimique.

Fig. 3.24 : L'utilisation des Rdp .

3.12 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'un Rdp ,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 suivant va se concentrer sur la transformation de graphes.

Chapitre 4

La transformation de modèles

4.1 Introduction

Le développement du logiciel fait face à une augmentation de la complexité des systèmes malgré l'utilisation des techniques spécifiques qu'on les appelle modèles de cycles de vie d'un logiciel comme le modèle en cascade, le modèle en V et le modèle en W. A l'instar d'autres sciences, la modélisation est de plus en plus utilisée pour maîtriser cette complexité des systèmes. L'ingénierie dirigée par les modèles(IDM), ou Model Driven Engineering(MDE) en anglais s'inscrit dans cette évolution en prônant l'utilisation systématique de modèles pour automatiser une partie des processus de développement suivis par les ingénieurs. Donc l'IDM propose de modéliser les applications à un haut niveau d'abstraction où elle place le modèle au cœur du processus de conception puis génère le code de l'application à partir des modèles. Ainsi, l'IDM est une approche générale et ouverte qui fait suite à la proposition du standard MDA (Model Driven Architecture) proposée par l'OMG en 2000. Elle a permis plusieurs améliorations significatives dans le développement des logiciels en permettant de se concentrer sur l'utilisation intensive des modèles et leur transformation.

Dans ce chapitre nous spécifions : la transformation de modèles est basée sur les techniques de modélisation, méta-modélisation et l'ingénierie dirigée par les modèles et nous finissons par la transformation de graphes .

4.2 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[**Etien et al**, **2007**].

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[**Bézivin**, **2005**].

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.

4.2.1 Système

Dans le cadre de l'ingénierie dirigée par les modèles, nous parlons souvent du système, quelle que soit sa nature (existant ou à réaliser). Un système est défini comme un ensemble organisé d'entité collaborant de manière unitaire, et en interaction permanent pour assurer une ou plusieurs fonctions dans les systèmes, on s'intéresse surtout aux logiciels[**Menet**, **2010**].

4.2.2 Modèle

Un modèle est une description abstraite d'un système construit dans un but donné. Donc il doit pouvoir être utilisé pour répondre à des questions sur le système. Les modèles doivent en général respecter des contraintes décrites par un méta modèle[**Menet**, **2010**].

4.2.3 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[**Guerrouf**, **2009**].

4.2.4 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[**Menet**, **2010**].

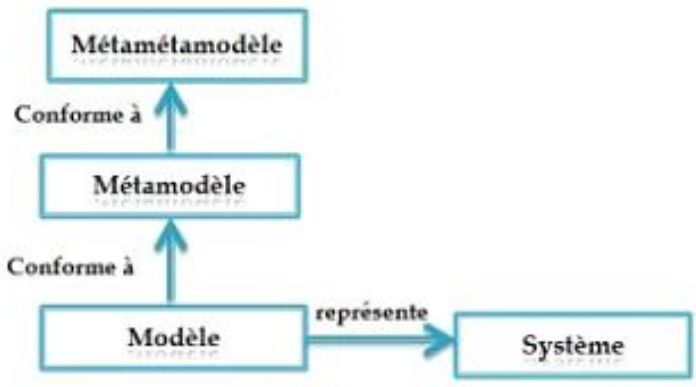

Fig. 4.1 : Notion de base en ingenierie des modeles.

4.2.5 Conforme a

Elle est définie par un ensemble de contraintes entre un modèle et son méta-modèle et qui expriment les relations entre eux[**Berramla**, **2014**].

4.3 L'approche MDA

MDA est une concrétisation de l'approche IDM. Elle se base sur la séparation des spécifications fonctionnelles d'un système, de son implémentation sur une plateforme spécifique, cette approche est basée sur le principe de la séparation des préoccupations. En partant de la phase de L'IE(ingénierie des exigences) jusqu'à la phase d'implémentation, selon le processus de développement orienté MDA,tout est considéré comme modèle[**Favre et al**, **2006**].

MDA identifie quatre types de modèles : CIM,PIM,PDM et PSM.

4.3.1 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[**Hettab**, **2009**].

4.3.2 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[**Hettab**, **2009**].

4.3.3 PSM(Plateforme Specific Model)

est le résultat de la transformation du Modèle PIM prenant en compte les spécificités de la plateforme d'exécution[**Amroune**, **2015**].

4.3.4 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[**Saidi**, **Bouanane**, **2018**].

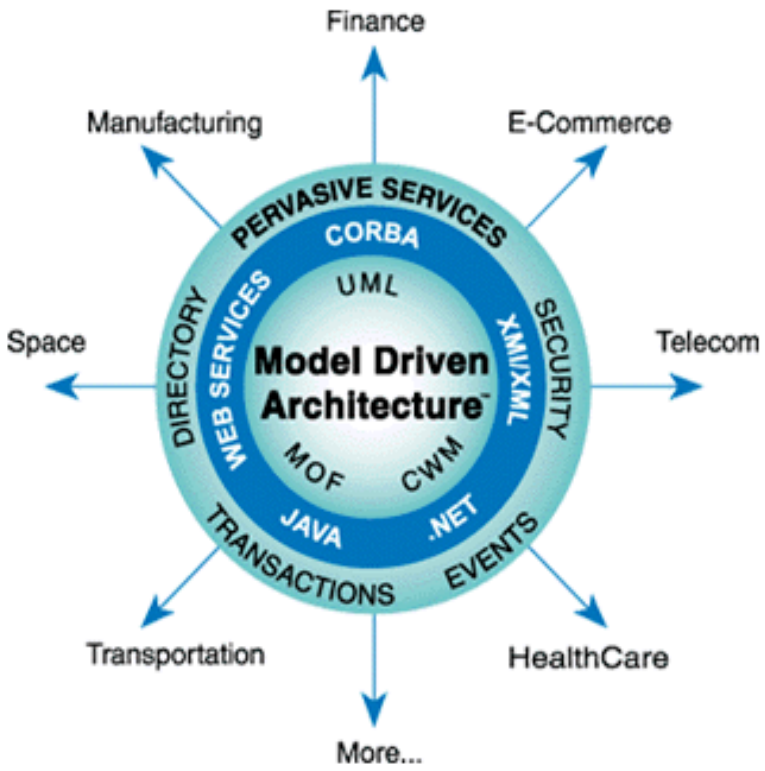

FIG. 4.2 : illustration OMG du MDA

Il existe des transformation de modèle du système vers d'autre modèle du même système, on peut citer[**Aouag**, **2014**]

1. **T**ransformation de Modèle CIM vers PSM.
2. **T**ransformation de Modèle CIM vers PIM.
3. **T**ransformation de Modèle PIM vers PSM.
4. **T**ransformation de diagramme UML 2.0 vers les réseaux de petri.

Dans la figure suivante, nous présentons le processus de transformation de l'approche MDA

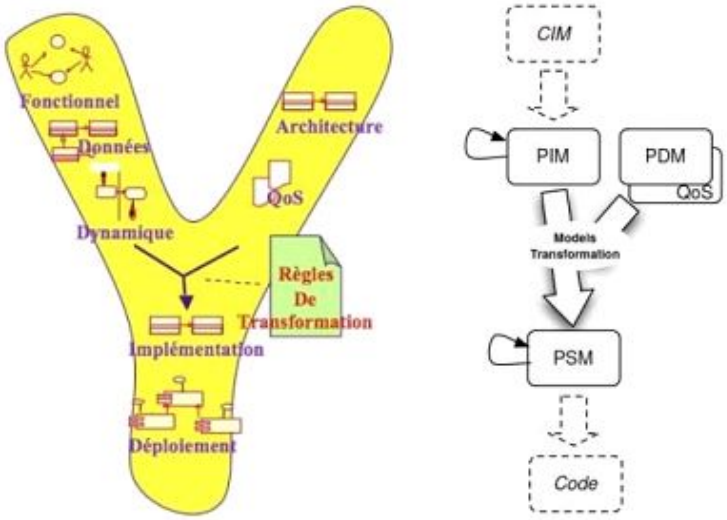

FIG. 4.3 : Processus en Y de transformation de modèle dans l'approche MDA

4.4 L'architecture MDA

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

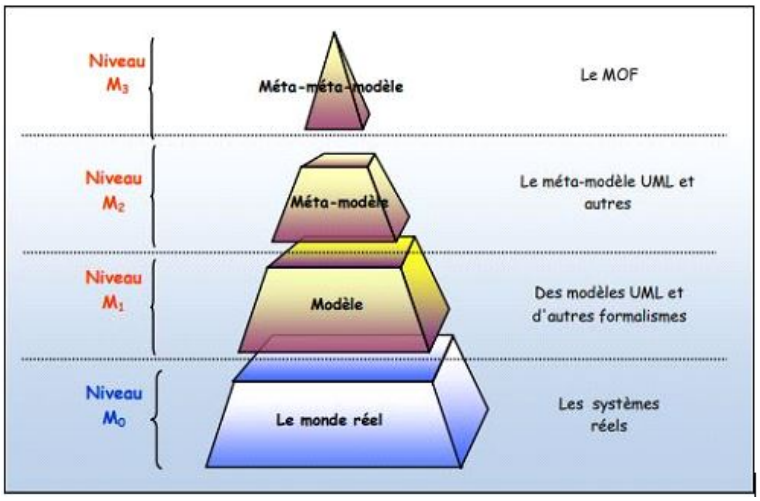

FIG. 4.4 : Pyramide de modélisation de l'OMG

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.

4.5 Transformation de Modèles

La transformation de modèle est une activité centrale dans le développement de logiciels dirigés par les modèles. Elle est utilisée aussi pour l'optimisation des modèles et d'autres formes d'évolutions des modèles. En outre, la transformation du modèle est utilisée pour le mappage des modèles entre les différents domaines pour les analyser ou pour la génération automatique de code[**Ehrig**, **et al**, **2007**].

4.5.1 Pourquoi la transformation de modèles

La transformation de modèles joue un rôle fondamental dans le développement des logiciels et vise des objectifs divers comme la translation de modèles, l'ingénierie inverse, la génération de code et la migration de données.

La figure suivante représente ce concept.

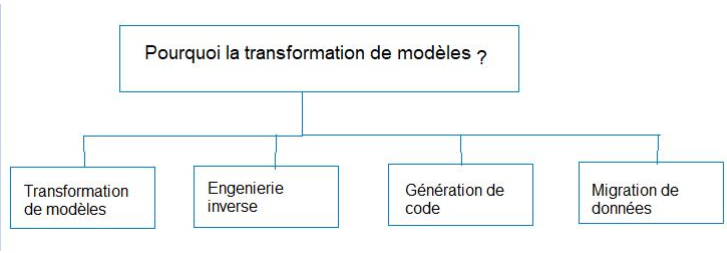

Fig. 4.5 : Les activités de la transformation de modèles.

4.5.2 Les étapes de la transformation de modèles en MDA

En MDA la transformation de modèles est basée sur les méta-modèle et contient deux étapes successive qui sont[**Aouag**, **2014**] :

1. **L**a spécification de règles de transformation permettant la définition de correspondance entre les concepts du méta-modèle du modèle source et les concepts du méta-modèle du modèle cible.

2. **L**'application des règles de transformation permettent de passer du modèle source au modèle cible automatiquement. Un outil de transformation est nécessaire pour l'exécution de ces modèles.

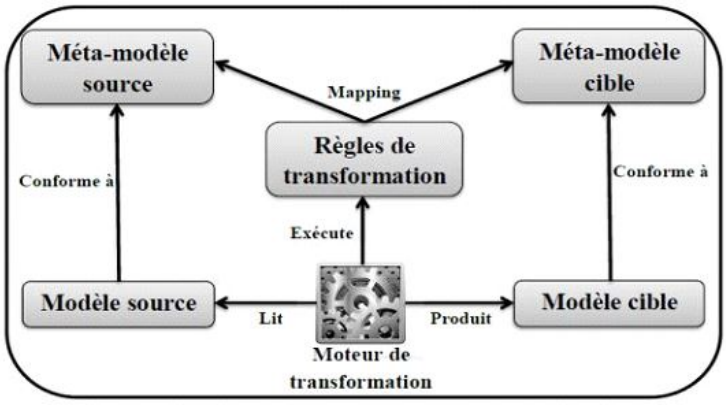

FIG. 4.6 : Concepts de base de la transformation de modèles.

4.5.3 Les types de transformation de modèles

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

Transformation horizontal

Ces transformations gardent le même niveau d'abstraction en modifiant les représentations du modèle source(ajout, modification, suppression[**Amroune**, **2015**].

Transformation verticale

La source et la cible d'une transformation verticale sont définies à différents niveaux d'abstraction. Un raffinement fait référence à une Ingénierie Dirigée par les Modèles (IDM) transformation qui baisse le niveau d'abstraction. Tandis qu'une abstraction désigne une transformation qui élève le niveau d'abstraction[**Amroune**, **2015**]

Transformation oblique

Ces transformations sont généralement utilisées par les compilateurs qui optimisent le code source avant la génération du code exécutable. Elles sont le résultat de la combinaison des deux premiers types de transformations[**Amroune**, **2015**]. On peut aussi distinguer deux type de transformation de modèles sur la base de méta-modèle dans lequel le modèle source et le modèle cible d'une transformation sont exprimés :

Endogène

C'est une transformation entre modèles exprimés dans le même méta-modèle[**Amroune**, **2015**].

Exogène

C'est une transformation entre modèles exprimés dans différents méta-modèles[**Amroune**, **2015**].

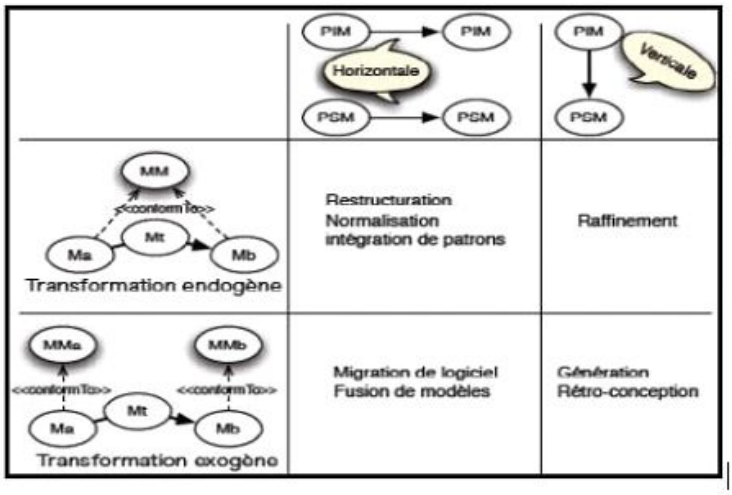

FIG. 4.7 : Les types de transformation de modèles.

4.6 Les approches de transformation

Les approches de transformation de modèles sont classées en deux grandes familles : les approches de modèle au code(Modèle to Text M2T), ou le code est considéré comme un modèle spécifique et les approches de modèle a modèle(Modèle to Modèle M2M). Le tableau ci-dessous illustre 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	

FIG. 4.8 : Les approches de transformation de modèles.

4.6.1 Modèle vers Texte (Modèle vers Code)

Dans cette catégorie, on distingue deux approches transformationnelles. La première basée sur la méthode de génération de code source qui se base soit sur le parcours du modèle, et la deuxième basée sur la génération de code par template[**Murata**, **1989**].

Génération de code par parcours de modèle

C'est une approche basée sur : en donnant un modèle en entrée et en écrivant un code en sortie. L'exemple le plus connu avec cette approche est l'environnement orienté objet dont les modèles UML sont représentés par des classes ou une API manipule les modèles et un outil est utilisé pour la génération de code [**Murata**, **1989**].

Génération de code par template

C'est l'une des approches la plus utilisée pour la génération de code. Cette approche consiste à définir des templates de modèle cible. Ces templates sont des modèles cibles paramètres ou modèles templates. L'exécution d'une transformation consiste a prendre un modèle template et à remplacer ses paramètres par les valeurs d'un modèle source. D'une autre façon, elle consiste a utiliser comme RHS Right Hand Side(le côté droit de la règle) un texte fixe dans lequel certaines parties sont variables : elles sont renseignées en fonction des informations récupérées dans le modèle source LHS Left Hand Side (le côté gauche de la règle), et peuvent notamment faire l'objet d'interactions. Parmi les langages les plus utilisés, on a Xpand, Acceleo,etc…[**Khalfaoui**, **2014**].

4.6.2 Modèle vers Modèle

Les transformations de type M2M consistent à transformer un modèle source en un modèle cible, ces modèles peuvent être des instances de différents méta-modèles. Elles offrent des transformations plus modulaires et faciles à maintenir, et permettent la génération de plusieurs modèles intermédiaires avant d'atteindre le modèle final. Ces modèles intermédiaires peuvent être utiles pour étudier les différentes vues du système, son optimisation, la vérification de ses propriétés et sa validation[**Khalfaoui**, **2014**].

Structure d'une transformation

Une transformation de modèle est principalement caractérisée par la combinaison des éléments suivants : des règles de transformation, une relation entre la source et la cible, un ordonnancement des règles, une organisation des règles, une traçabilité et une direction[**Elmansouri**, **2009**].

Règle de transformation

Une règle de transformation est composée de deux parties : un côté gauche (LHS, Left Hand Side) qui accède au modèle source, et un côté droit (RHS, Right Hand Side) qui accède au modèle cible. Une règle comporte également une logique, qui exprime des contraintes ou calculs sur les éléments des modèles source et cible. Une logique peut avoir la forme déclarative ou impérative. Une logique déclarative consiste à spécifier des relations entre les éléments du modèle source et des éléments du modèle cible. Une logique impérative correspond le plus souvent, mais pas nécessairement, à l'utilisation de langages

de programmation pour manipuler directement les éléments des modèles par le biais d'interfaces dédiées. Optionnellement, une règle, peut être bidirectionnelle, comporter des paramètres, ou encore nécessiter la construction de structures intermédiaires.Les règles de transformation nécessitent[**Aouag**, **2014**] :

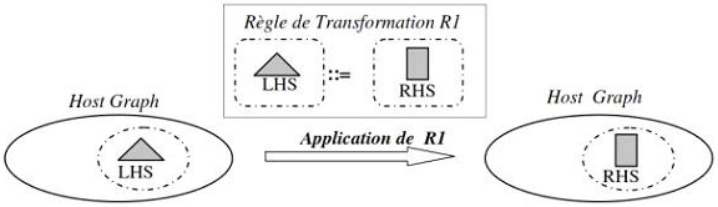

Fig. 4.9 : Principe de l'application d'une règle.

organisation des règles

il existe plusieurs façons pour organiser les règles de transformation , on peut les organiser de façon modulaire, d'un ordonnancement explicite ou d'une structure dépendante du modèle source ou du modèle cible[**Aouag**, **2014**].

ordonnancement et orientation

il y a deux types d'ordonnancement des règles ; l'ordonnancement implicite qui est défini par l'outil de transformation lui-même, et l'ordonnancement explicite. Il existe des mécanismes permettant de spécifier l'ordre d'exécution des règles[**Aouag**, **2014**].

traçabilité

la tracabilité présente le lien entre les éléments de modèle source et modèle cible[**Elmansouri**, **2009**].

Enfin, les techniques de transformations peuvent être classées en cinq catégories selon :

1. **A**pproches manipulant directement les modèles.
2. **A**pproches relationnelles .
3. **A**pproches guidées par la structure.
4. **A**pproches basées sur la transformation de graphes.
5. **A**pproches hybrides .

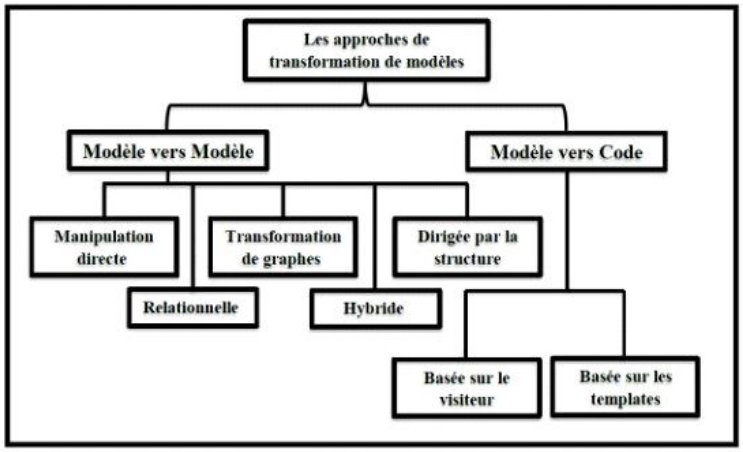

Fig. 4.10 : Les approches de transformations de modèles

4.7 Transformation de graphes

Avant d'expliciter les fondements de la transformation de graphe, nous devons présenter les notions de base d'un graphe : graphe, sous-graphe et morphisme de graphe ...

4.7.1 Notion de graphe

Un graphe est constitué de sommets qui sont reliés par des arêtes. Deux sommets reliés par une arête sont adjacents. Le nombre de sommets présents dans un graphe est appelé l'ordre du graphe. Le degré d'un sommet est le nombre d'arêtes dont ce sommet est une extrémité[**Kerkouche**, **2011**].

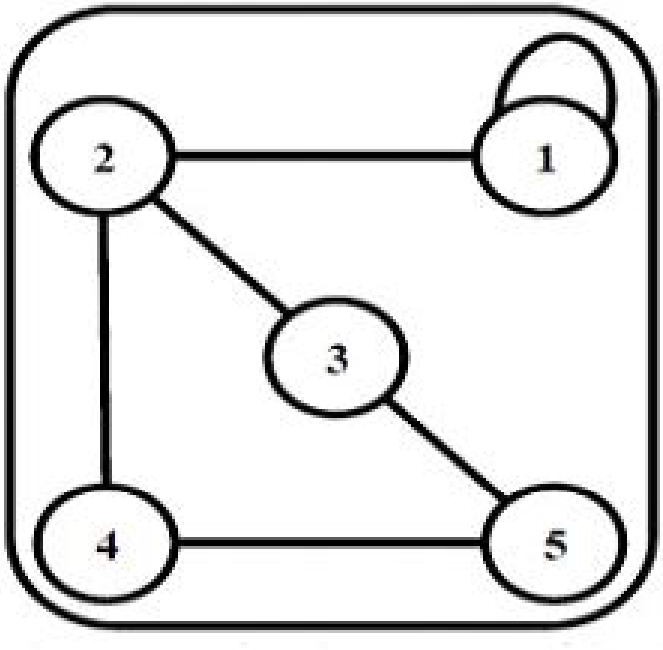

Fig. 4.11 : Graphe non-orienté.

4.7.2 graphe orienté(digraphe)

Un graphe orienté (digraphe) est un graphe dont les arêtes sont munies de directions : nous parlons alors de l'extrémité d'une arête. Dans un graphe orienté, une arête est appelée"arc"[**Kerkouche**, **2011**].

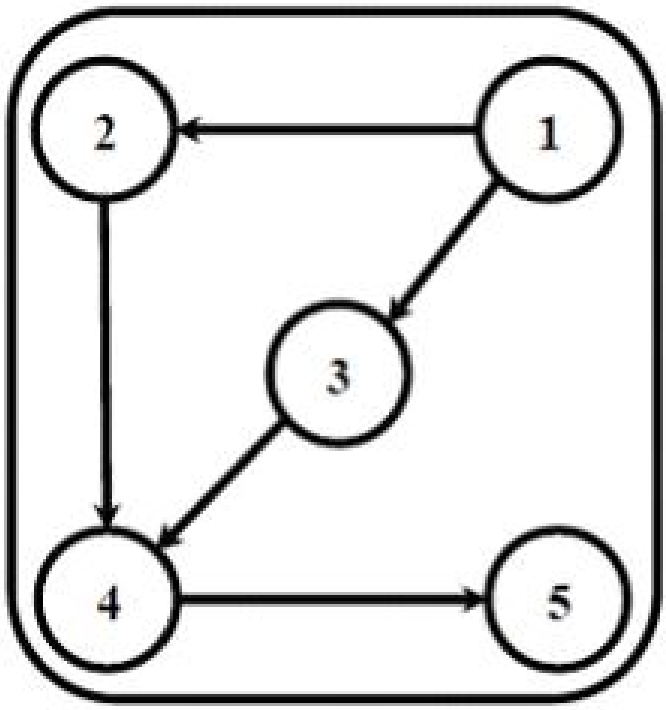

FIG. 4.12 : Graphe orienté

Propriétés d'un graphe

1. **graphe attribué** est un graphe qui peut contenir un ensemble prédéfini d'attributs[**Kerkouche**, **2011**].

2. **les sommets adjacents** Deux sommets sont adjacents s'ils sont reliés par une arête[**Kerkouche**, **2011**].

3. **sous-graphe** Un sous-graphe d'un graphe G est un graphe G' composé de certains sommets de G, ainsi que toutes les arêtes qui relient ces sommets[**Kerkouche**, **2011**].

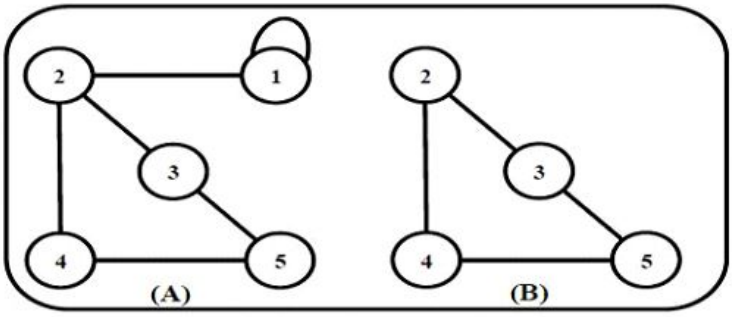

FIG. 4.13 : Graphe (A) et sous-graphe (B).

4. **graphe étiqueté** Un graphe étiqueté est un graphe orienté, dont les arcs possèdent des étiquettes. Si toutes les étiquettes sont des nombres positifs, on parle de graphe pondéré[**Kerkouche**, **2011**].

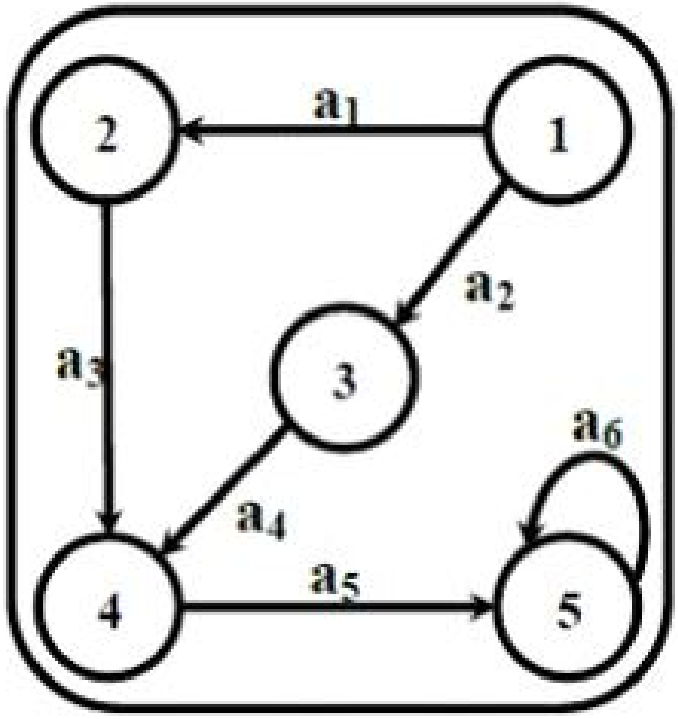

FIG. 4.14 : Graphe étiqueté.

4.7.3 Grammaire de graphe

Une grammaire de Graphe est généralement définie par un triplet : GG = (P, S, T) Où :

1. **P** : ensemble de règles,

2. **S** : un graphe initial .

3. **T** : ensemble de symboles terminaux.

Une grammaire de graphes distingue les graphes non terminaux, qui sont les résultats intermédiaires sur lesquels les règles sont appliquées, des graphes terminaux dont on ne peut plus appliquer de règles. Ces derniers sont dans le langage engendré par la grammaire de graphe. Pour vérifier si un graphe G est dans les langages engendrés par une grammaire de graphe, il doit être analysé. Le processus d'analyse va déterminer une séquence de règles dérivant G[**Andries et al , 1999**].

Le principe de règle

Une règle de transformation de graphe est définie par : r = (L, R, K, glue, emb, cond) . Elle consiste en :

1. **D**eux graphes : L graphe de côté gauche et R graphe de côté droit.

2. **U**n sous-graphe K de L.

3. **U**ne occurrence glue de K dans R qui relie le sous graphe avec le graphe de côté droit.

4. **U**ne relation d'enfoncement emb qui relie les sommets du graphe de côté gauche et ceux du graphe du côté droit.

5. **U**n ensemble cond qui spécifie les conditions d'application de la règle.

[**M.Bendiaf**, **2018**]

Application des règles

L'application d'une règle r = (L, R, K, glue, emb, cond) à un graphe G produit un graphe résultant H. Le graphe H produit peut être obtenu depuis le graphe d'origine G en passant par les cinq étapes suivantes :

1. **C**hoisir une occurrence du graphe de côté gauche L dans G.

2. **V**érifier les conditions d'application d'après cond.

3. **R**etirer l'occurrence de L (jusqu'à K) de G ainsi que les arcs pointillés, c'est-à-dire tous les arcs qui ont perdu leurs sources et/ou leurs destinations. Ce qui fournit le graphe de contexte D de L qui a laissé une occurrence de K.

4. **C**oller le graphe de contexte D et le graphe de côté droit R suivant l'occurrence de K dans D et dans R. C'est la construction de l'union de disjonction de D et R et, pour chaque point dans K, identifier le point correspondant dans D avec le point correspondant dans R.

5. **E**nfoncer le graphe de côté droit dans le graphe de contexte de L suivant la relation d'enfoncement emb.

Pour chaque arcs retiré incident avec un sommet v dans D et avec un sommet v' dans l'occurrence de L dans G et pour chaque sommet v" dans R, un nouvel arc est établi (avec la même étiquette) incident avec l'image de v et le sommet v" à condition que (v', v") appartient à emb[**Bendiaf**, **2018**].

Système de transformation de graphe

Un système de transformation de graphe est défini comme un système de réécriture de graphes qui applique les règles de la Grammaire de Graphes sur son graphe initial jusqu'à ce que plus aucune règle ne soit applicable[**Rozenberg**, **1999**]. Cette approche

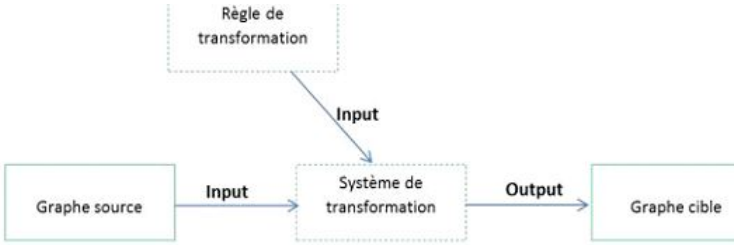

Fig. 4.15 : Principe de mise en œuvre de transformation de graphes

de transformations de modèles a plusieurs avantages par rapport aux autres approches :

1. **L**es grammaires de graphes sont un formalisme naturel, visuel, formel et de haut niveau pour décrire les transformations.

2. **L**es fondements théoriques des systèmes de la réécriture de graphes permettent d'aider à vérifier certaines propriétés des transformations telles que la terminaison ou la correction[**Rozenberg**, **1999**].

Langage engendré

Soit G0 un graphe initial,Gn est le graphe final et une séquence de transformations successives de graphes : G0 → G1 → ... → Gn est une dérivation successive à partir de G0 vers Gn en appliquant les règles de R qui sont étiquetées par les symboles de T, est dit langage engendré par R, G0 et T et on écrit L(R, G0, T)[**Aouag**, **2014**].

4.7.4 Outils de transformation de graphes

Il existe plusieurs outils implantant les systèmes de réécriture de graphes. Les plus intéressants sont :

Fujaba

un outil très puissant utilisé principalement pour la génération de code Java à partir de diagrammes UML. Ces dernières années, il est devenu une référence dans plusieurs axes de recherche. Particulièrement, il est très utilisé dans la modélisation et la simulation des systèmes mécaniques et électriques[**Burmester et al , 2004**].

AGG

un environnement développé à l'université technique de Berlin. C'est l'un des outils de transformation de graphes les plus cités dans la littérature. Il est implémenté en langage Java. Il est muni d'une interface permettant de spécifier graphiquement les règles de transformation. Il permet des simulations pas à pas ainsi que quelques vérifications élémentaires. Les graphes manipulés sont orientés. Les nœuds et les arcs peuvent être étiquetés par des objets Java[AGG].

GreAt

un outil permettant la définition des transformations unidirectionnelles de plusieurs modèles sources vers plusieurs modèles cibles. Il est basé sur une notation graphique pour la spécification des règles de transformations, mais les expressions d'initialisation des attributs et les conditions d'application sont éditées textuellement. Les modèles manipulés doivent être conformes à leurs méta-modèles[**Balasubramanian et al , 2006**].

AToM³

un outil de modélisation multi-paradigme développé par le laboratoire MSDL de l'université McGill Montréal au Canada. AToM³ est un outil visuel dédié à la transformation de graphes, implémenté en langage Python. Il possède une couche de Méta-Modélisation permettant une spécification syntaxique et graphique des formalismes utilisés. Les manipulations de modèles souhaitées sont définies sous forme de grammaires de graphes[**Vangheluwe et al , 2002**]. Il existe d'autres outils comme : Viatra, Eclipse Modeling Galileo, progrès, Groove, GrGen, EMF Tiger Dans le cadre de ce mémoire de mastère, nous nous intéresserons à l'utilisation de l'outil AToM³ pour l'implémentation des grammaire de graphes proposées.

4.7.5 les avantages de l outil AToM³

Ce choix est justifié par les nombreux avantages présentés par cet outil, parmi lesquels on peut citer : sa simplicité, sa disponibilité, il est multi paradigmes (la possibilité de méta-modéliser), et il s'adapte bien avec les types de transformations utilisés dans le cadre de ce mémoire, à savoir les transformations de type modèle vers modèle.

4.7.6 outil d'implémentation

AToM³ est un outil de modélisation multi-paradigmes en cours de développement au Laboratoire de modélisation, simulation et conception (MSDL) de l'Université McGill. AToM3 permet la description ou la méta-modélisation de différents types de formalismes (nous nous concentrons sur les formalismes pour la simulation de systèmes dynamiques), ainsi que de générer des outils personnalisés à manipuler (créer, éditer, transformer, simuler, optimiser, ...) modèles exprimés dans de tels formalismes. [**Université**, **2020**]. Les deux fonctionnalités principales d'AToM3 sont la méta-modélisation et la transformation de modèle.

La méta-modélisation

est la description ou la modélisation de différents types de formalismes en peut utiliser le modèle ER ou le modèle des diagrammes de classes. Dans ce mémoire nous avons choisi d'utiliser le modèle des diagrammes de classes et le code python pour spécifier les contraintes[**zerara et al**, **2020**].

La transformation de modèles

est une technique consistant à transformer, traduire ou modéliser automatiquement un modèle décrit dans un certain formalisme, vers un autre modèle qui peut être décrit soit dans le même formalisme soit dans un autre. L'utilisateur doit définir l'ensemble des règles (elles sont composées de : LHS et RHS), les priorités , les conditions qui doivent être satisfaites pour que la règle soit applicable (pré-condition, post-condition) et les actions à exécuter[**zerara et al**, **2020**].

AToM³ offre la possibilité à l'utilisateur de créer sa propre nouvelle grammaire, de charger une grammaire prédéfinie (modification, consultation ou exécution). L'exécution de la grammaire s'effectue sur un modèle en entrée et produit un modèle de sortie d'une manière automatique[**Aouag**, **2014**].

La figure suivante présente l'interface d'AToM³

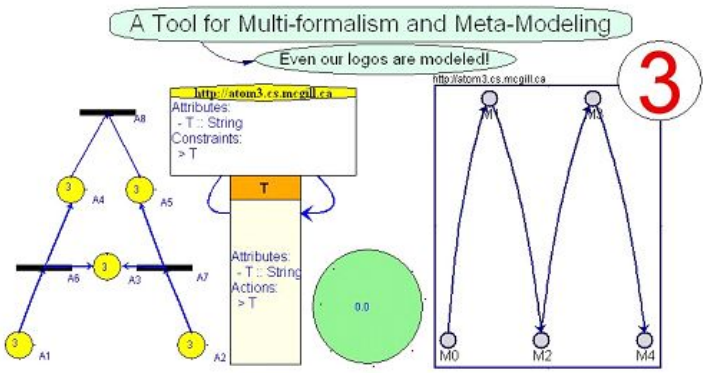

FIG. 4.16 : Présentation de l'outil AToM³

4.8 conclusion

Dans ce chapitre nous nous somme intéressé à l'ingénierie dirigée par les modèles et ces différent principes, Nous avons survolé également les technique de transformation de modèles , Après nous avons présenté un introduction à la transformation de graphes.Nous avons également citer quelques outils de transformation de graphes et présenter AToM^3 qui est utilisé dans l'implémentation de notre travail. Le chapitre suivant c'est l'approche proposée, qui est basée sur les concepts présentés précédemment.

Chapitre 5

L'approche De Transformation Proposée

5.1 Introduction

La transformation des modèles est l'un des principaux concepts de l'IDM , elle permet de translation des modèles vers d'autre modèle avec différent niveaux d'abstraction dans le but de réduire le coût et le temp de développement, la transformation des modèles, n'est bien sûr pas une tâche facile à implementé, il est donc nécessaire d'avoir des outiles puissant pour la gestion des modèles et des langage dédié pour leur transformation. Le travail présenté dans ce chapitre illustre la transformation des diagrammes état-transition orienté-aspect vers des réseaux de petri.

5.2 L'approche proposée

Dans ce chapitre nous présentons une démarche de transformation de modèle proposé dans le cadre de notre recherche. Il s'agira de transformer des diagramme etat-transition orienté-aspect vers des réseaux de petri.

5.3 Méta-Modélisation

Généralement , la pratique de la méta-modélisation consiste à définir les méta-modèles qui reflètent une partie de la sémantique des modèles . Les langages de méta-modélisation offrent les conceptions et les relations élémentaires en termes desquels il est possible de définir les méta-modélisation qui peuvent être utilisés dans la spécification des règles de transformation de modèle[**mellouli**, **2014**].

Dans ce qui suit nous présentons de manière graphique les méta-modèles source et cible de notre travail d'étude de cas.

5.3.1 Spécification de méta-Modèle source

Dans cette partie, nous spécifions notre méta-modèle de diagramme d'état-transition orienté-aspect. Le méta-modèle proposé est composé des classes et des associations.

Les classes

1. **Diagramme etat-transition** Cette classe a un attribut qui est le Nom, elle représente les états dans le diagramme.
2. **État-initial** Cette classe représenté l'état initial(l' initialisation du système).
3. **Etat-simple** Cette classe représenté les états simples (les étapes de la vie du système) .
4. **Etat-composite** Cette classe représenté les états composites (état regroupant un ensemble d'états.) .

5. **État-Final** Cette classe représente l'état final du diagramme d'états- transitions (fin de vie du système) .

6. **Aspect** Cette classe représente les aspects. Chaque aspect a un nom, les points d'activation et les conseils pour l'insérer, un événement ,une condition et une action pour spécifier aux nouvelles transition qui sera créé lors la création ou la liaison entre les états d'un aspect[**zerara et al**, **2020**].

Les Associations

Chaque association du méta-modèle possède des attributs de type String(evennement : c'est l'événement c'est un fait instantané venant de l'extérieur du système ,condition : des condition vérifiées,action : c'est la réaction du système a un événement), Les cardinalités pour chaque association sont :

1. **T**o target : 0 to N.

2. **F**rom source : 0 to N.

La définition des méta-modèles a été réalisée sous AToM3.

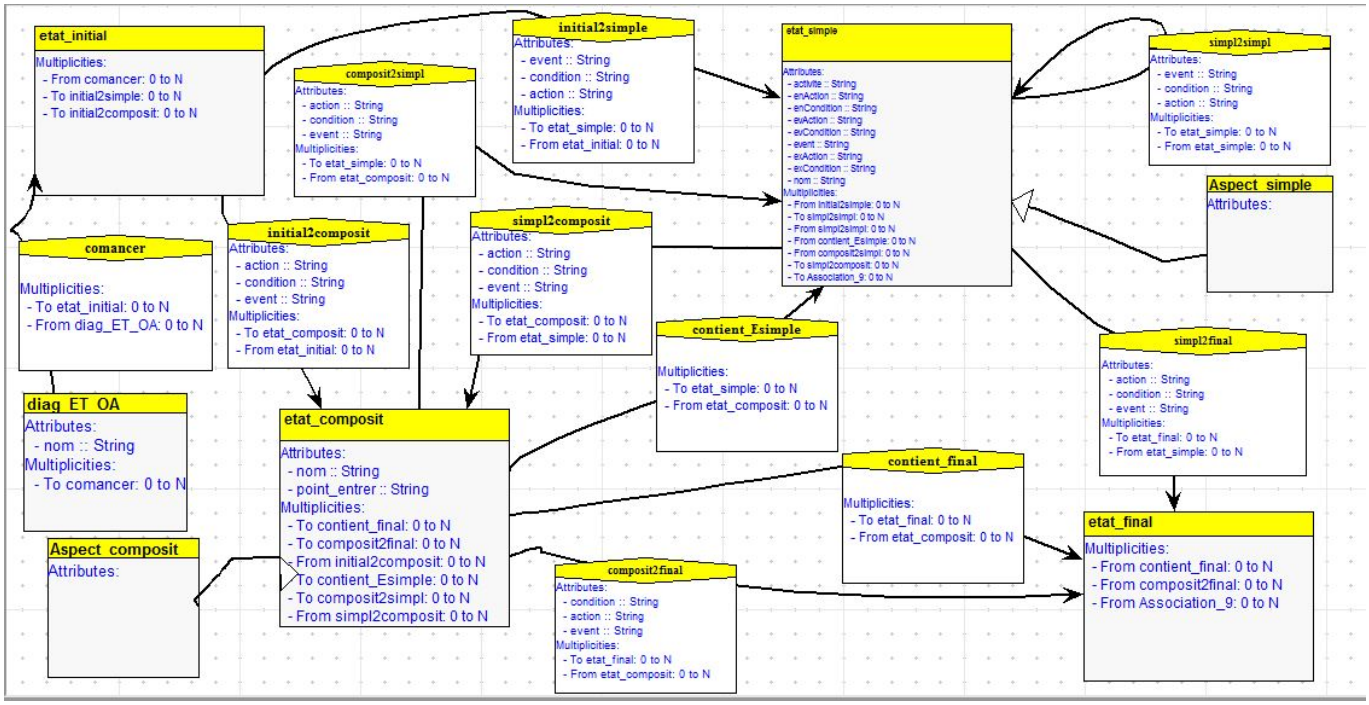

Fig. 5.1 : Diagramme état-transition orienté-aspect.

5.3.2 Specification de méta-Modèle cible

Nous présentons maintenant le méta-modèle cible qui définit la structure statique Des modèles de réseau de petri.

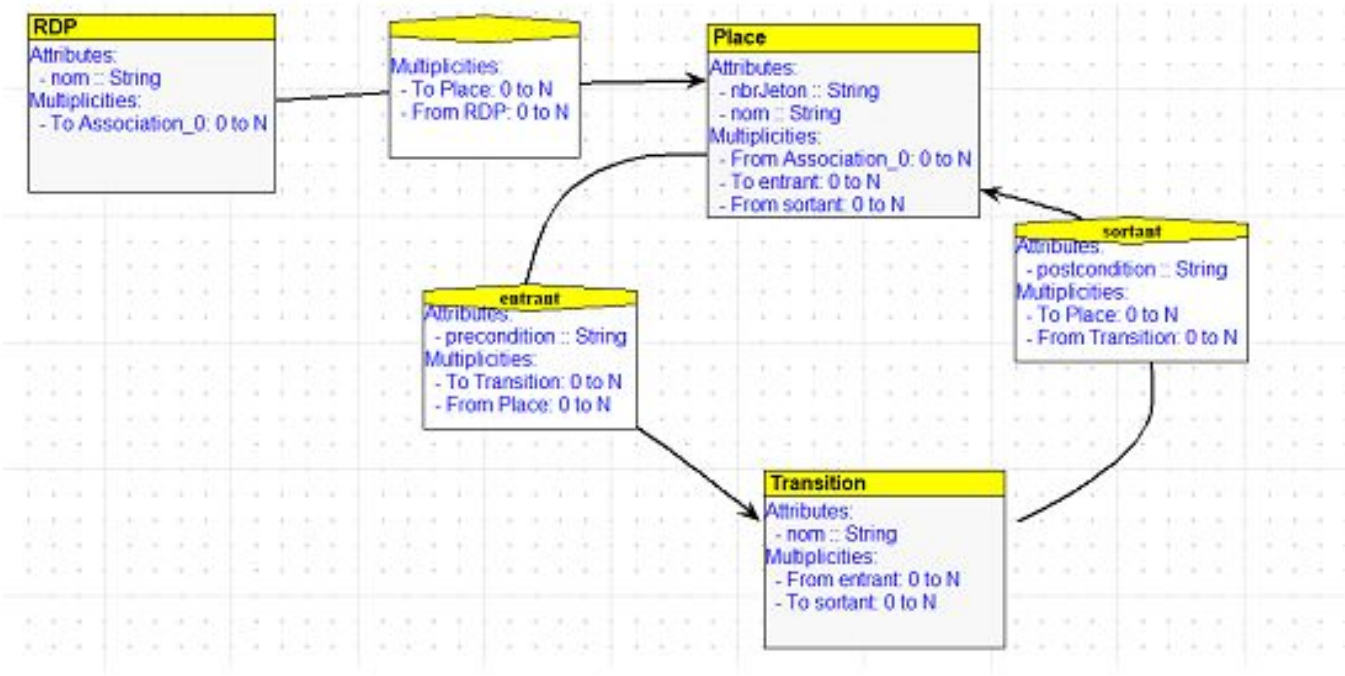

FIG. 5.2 : Méta-Modèle de réseau de petri.

5.4 Processus de transformation de modèles

La transformation de modèles se base sur la définition des règles de transformation ou une bonne transformation de modèles dépend d'une bonne définition des règles de transformation. Dans cette sous-partie, nous présentons un ensemble de règles de transformation que nous avons étendu et amélioré. Ces règles de correspondance exprime la sémantique suivante :

5.4.1 Etat-Simple

La transformation de l'état simple se fait directement vers une place. La figure suivante présente graphiquement cette transformation.

FIG. 5.3 : Transformation d'une état-simple

5.4.2 Etat-Final

La même chose se réalise pour l'état-final, donc il est translaté en une place. Nous présentons cette figure qui traduit cette transformation.

FIG. 5.4 : Transformation d'un état-final

5.4.3 Etat-Composite

L'état-composite se traduit vers les réseaux de petri par la transformation de ces états et de ces sous transitions. Cette figure présente les possibilités de transformer l'état composite dans le réseau de petri correspondant.

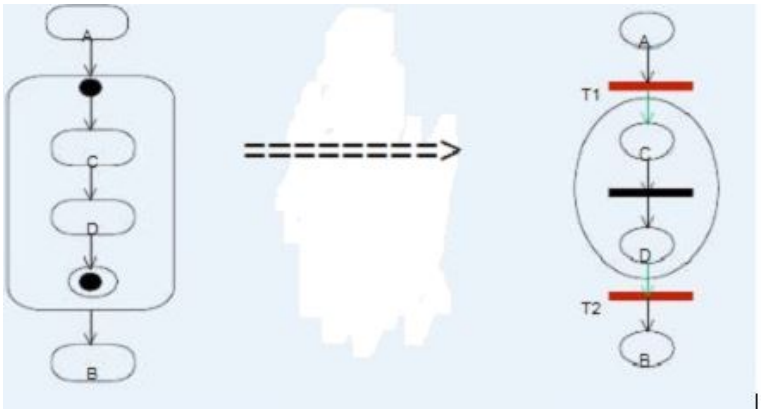

FIG. 5.5 : Transformation d'un état composite

5.4.4 Pseudo-État

La transformation d'un pseudo-état se change selon son propre type

pseudo-état initial

Un pseudo-état initial est transformé de la même manière que état simple, mais son avoir des transition interne.

FIG. 5.6 : Transformation d'un pseudo-état initial

Pseudo-État terminal

Un pseudo-état terminal est translaté en une place dans le réseau de petri correspondant.

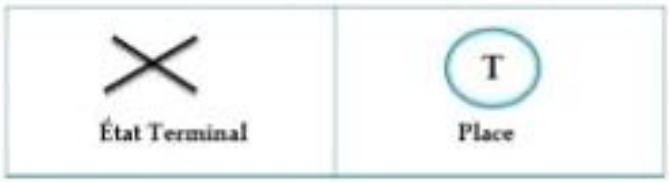

FIG. 5.7 : Transformation d'un pseudo-etat terminal.

Les pseudo-états de type : jointure, choix,point d'entrée, point de sortie, bifurcation et jonction sont supprimés en gardant leur sens par la transformation de leur transition

5.4.5 Etat Aspect

La même chose se réalise pour l'état aspect, donc il est translaté en une place. Nous présentons cette figure qui traduit cette transformation.

FIG. 5.8 : Transformation d'un état aspect.

5.4.6 Transition Simple

Tout d'abord, une transition simple possède comme une source et une destination des états simples. Donc elle peut être transformée vers une transition de réseaux de petri en gardant les mêmes sources et les mêmes destinations.

FIG. 5.9 : Transformation d'une transition simple.

5.4.7 Transition de bifurcation

une transition est dite de bifurcation si elle possède une destination de type pseudo-état de bifurcation. cette transition est transformée vers une transition dans le réseau de petri en gardant ces sources et en changeant ses destination par les destination de ses transition sortantes qui possèdent comme source le pseudo etat de bifurcation.

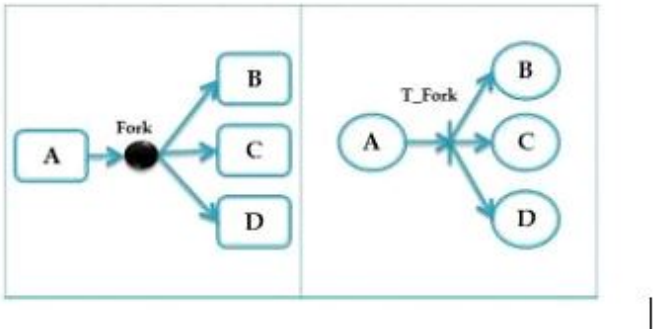

FIG. 5.10 : Transformation d'une transition de bifurcation.

5.4.8 Transition de jointure

une transition est dite de type jointure si elle contient une source de type pseudo-état de jointure. Cette transition est donc transformée vers une transition dans les réseaux de petri en gardant ses destinations et en changeant ses sources de ses transition.

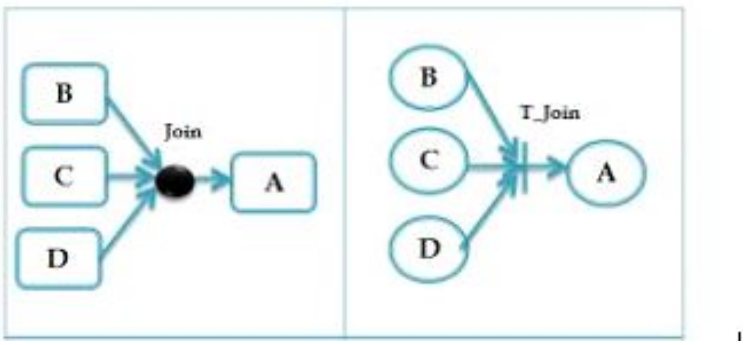

FIG. 5.11 : Transformation d'une transition de jointure.

5.4.9 Transition de jonction

Une transition est dite de jonction si elle a une source de type pseudo-état de jonction. Cette transition peut être transformée vers une transition dans le réseau de petri en gardant ses destinations et en changeant ses sources par les sources de ses transitions qui contient comme destination le pseudo état de jonction .

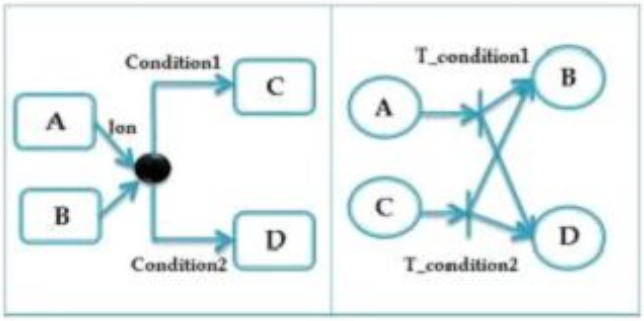

FIG. 5.12 : Transformation d'une transition de jonction.

5.4.10 Transition de choix

Un Choix est montré comme un diagramme avec une transition à l'arrivée et deux ou plusieurs transitions quitter

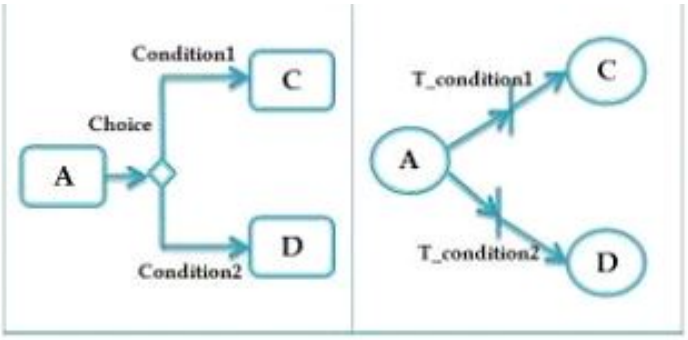

FIG. 5.13 : Transformation d'une transition de choix.

5.4.11 Transition de point d'entrée

Le point d'entrer permet de relier les transition entrants

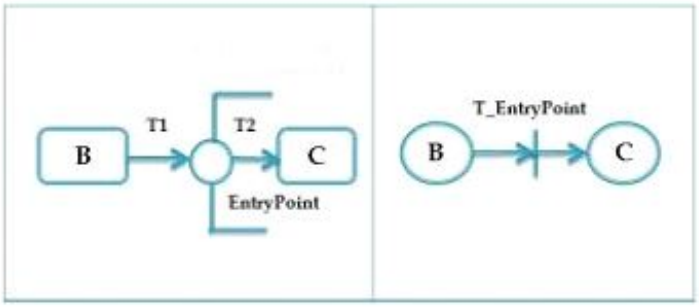

FIG. 5.14 : Transformation d'une point d'entrer

5.4.12 Transition de point de sortie

Le point de sortie permet de relier les transition sortant.

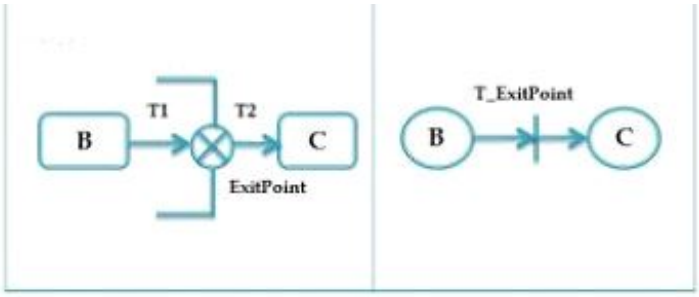

FIG. 5.15 : Transformation d'un point de sortie

5.4.13 Transition d'un état-initial

Une transition est dite d'un état initial si elle possède une source de type pseudo-état initial. Cette transition est donc transformée en une transition dans le réseau de petri en gardant ses destination et ses sources.

FIG. 5.16 : Transformation d'une transition d'un état initial

5.4.14 Transition d'un état final

Une transition est dite d'un état final si elle possède une destination de type pseudo-etat final. Cette transition est donc transformée en une transition dans le réseau de petri en gardant ses destination et ses sources.

FIG. 5.17 : Transformation d'une transition d'un état final.

5.4.15 Transition interne

Une transition interne est dite interne si un état simple possède une activité interne(Entry,Do,Exit).donc ses activites peuvent etre transformes en des transition dans le réseau de petri qui possedent la meme source et la même destination(était-simple).

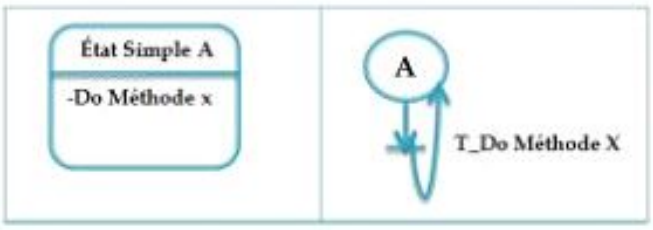

FIG. 5.18 : Transformation d'une transition interne.

5.5 Grammaire de graphes pour la transformation des diagrammes etat-transition orientés aspect vers les réseaux de pétri

Nous avons défini une grammaire de graphes composee par des règles qui seront exécutées dans un ordre ascendant selon leurs priorités. Notre technique de transformation est basée sur la transformation de graphes, alors chaque règle est composée de deux parties : une partie gauche (LHS) et une partie droite (RHS). Ces deux parties peuvent contenir des éléments des diagrammes etat-transition orientés aspect, des RdPs ou bien les deux formalismes ou même temps (DETOA/RdP). La figure suivante presenter notre grammaire.

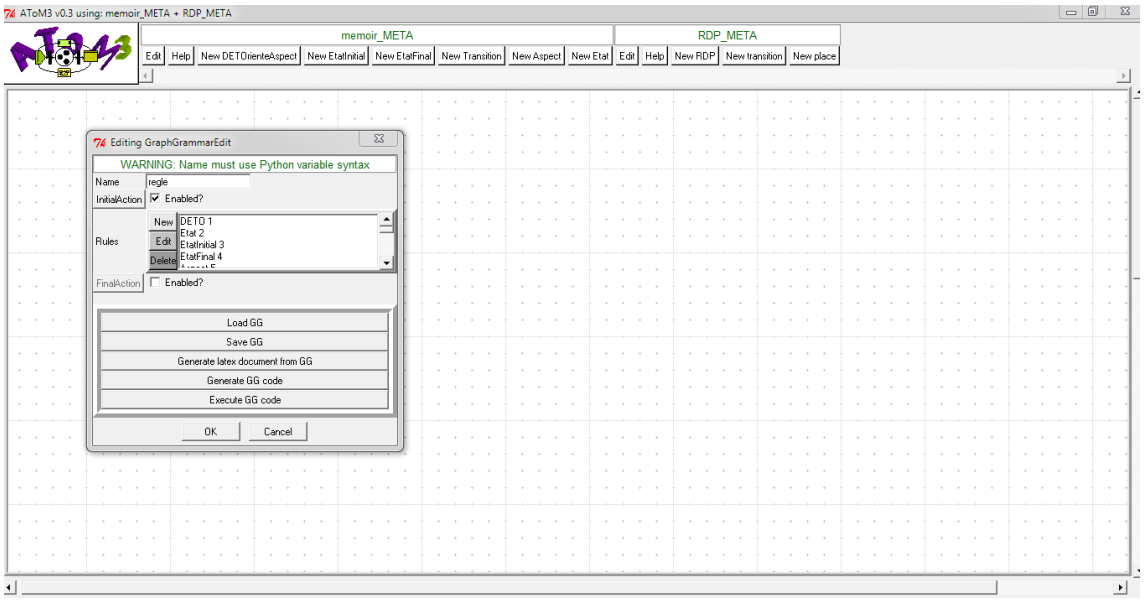

FIG. 5.19 : Grammaire de graphes proposée.

5.5.1 Règle 1

Cette règle permet de transformer le diagramme d'etat transition orienté aspect de la partie gauche LHS vers le réseau de pétri dans la partie droite RHS.

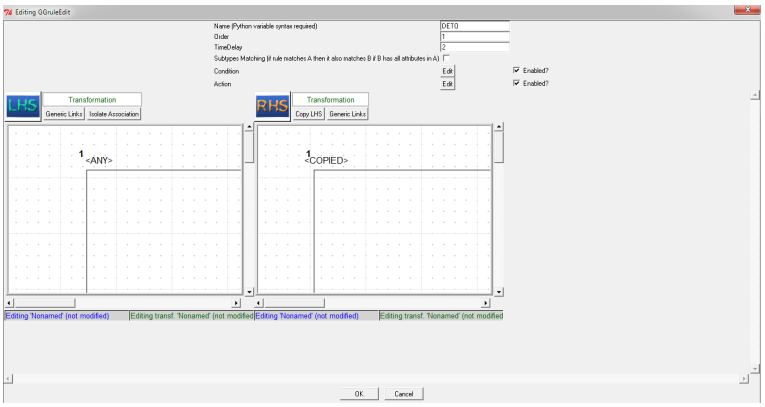

FIG. 5.20 : Règle 1: Diagramme etat-transition orienté-aspect.

5.5.2 Règle 2

Cette règle permet de transformer « un etat » de diagramme d'etat-transition orienté aspect de la partie gauche LHS en « une place » de réseau de pétri dans la partie droite RHS.

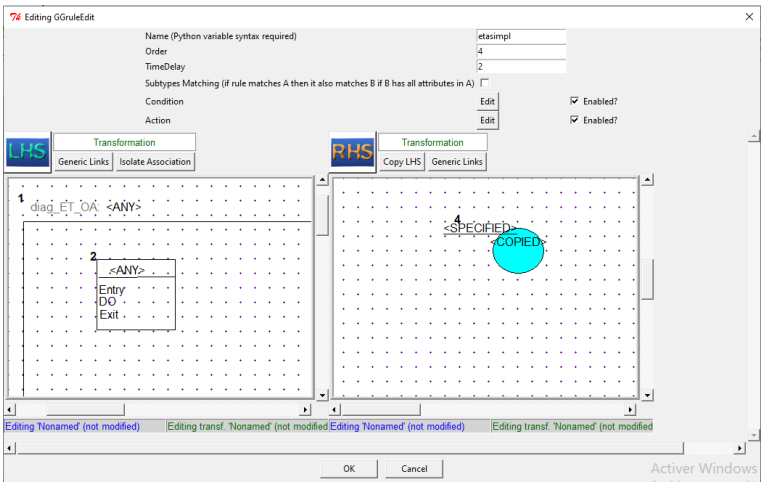

FIG. 5.21 : Règle 2: Création Etat Simple.

5.5.3 Règle 3

Cette règle permet de transformer « un etat-initial » de diagramme d'etat-transition orienté aspect de la partie gauche LHS en « une place » de réseau de pétri dans la partie droite RHS.

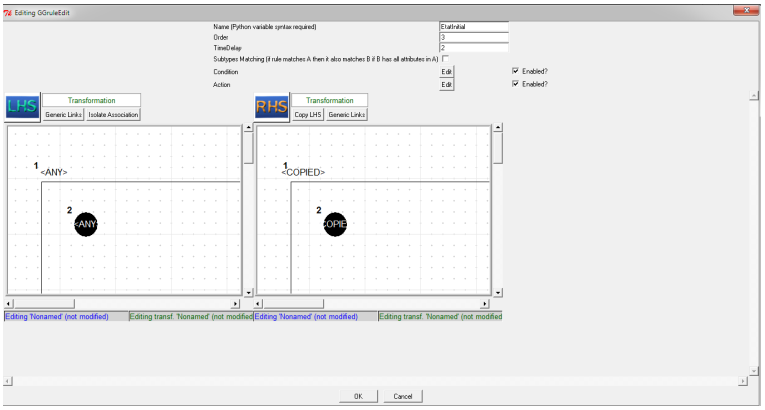

FIG. 5.22 : Règle 3: Création Etat-initial.

5.5.4 Règle 4

Cette règle permet de transformer « un etat-final » de diagramme d'etat-transition orienté aspect de la partie gauche LHS en « une place » de réseau de pétri dans la partie droite RHS.

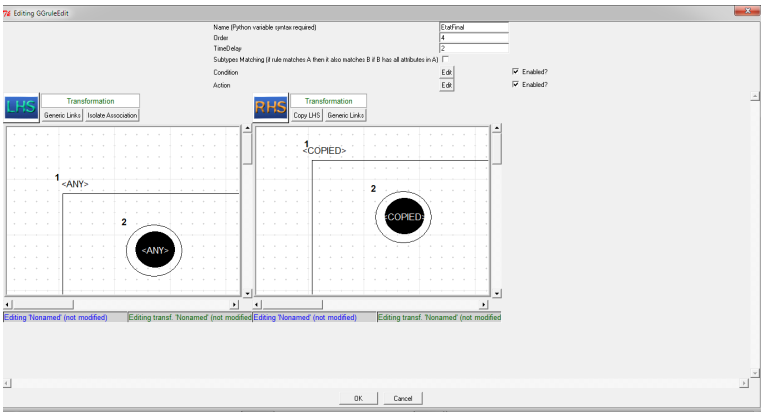

FIG. 5.23 : Règle 4: Création Etat-final.

5.5.5 Règle 5

Cette règle permet de transformer « un aspect » de diagramme d'etat-transition orienté aspect de la partie gauche LHS en « une place » de réseau de pétri dans la partie droite RHS.

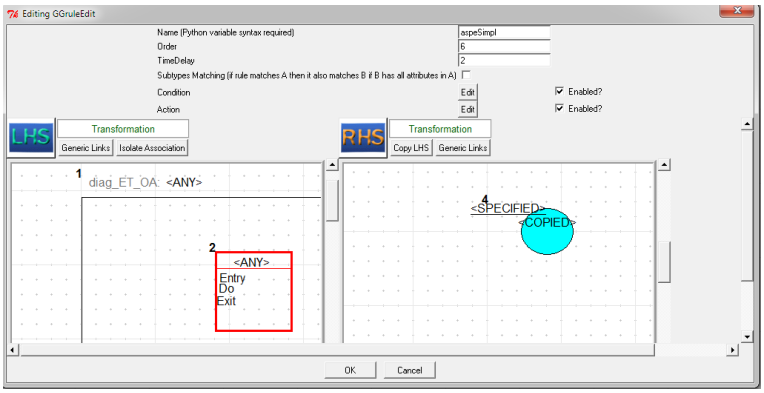

FIG. 5.24 : Règle 5: Création Aspect.

5.5.6 Règle 6

Cette règle permet de transformer « une etat-composite » de diagramme d'etat-transition orienté aspect de la partie gauche LHS en « une place » de réseau de pétri dans la partie droite RHS.

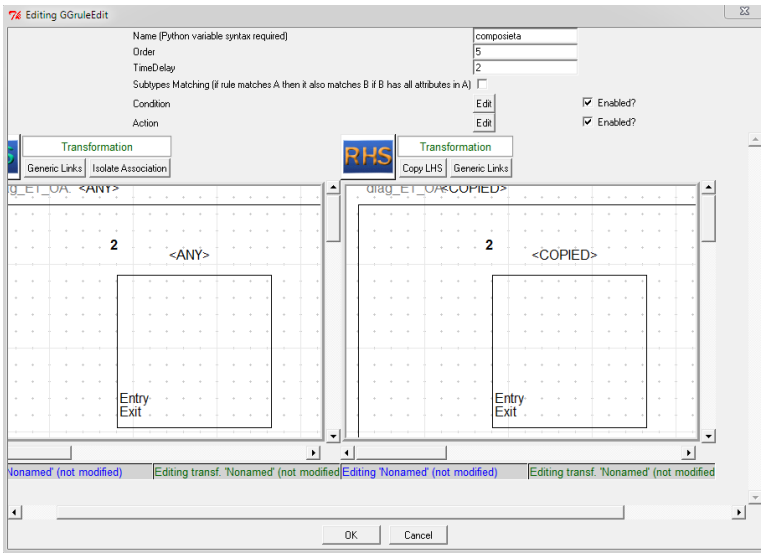

FIG. 5.25 : Règle 6: Création Etat-Composite.

5.5.7 Règle 7

Cette règle permet de transformer "une relation" qui relié etat-initial avec une etat-simple d'un diagramme d'etat-transition orienté aspect de la partie gauche LHS en « une relation » relie une place represente l'etat-initial et une autre place représente un etat-simple dans le réseau de pétri dans la partie droite RHS

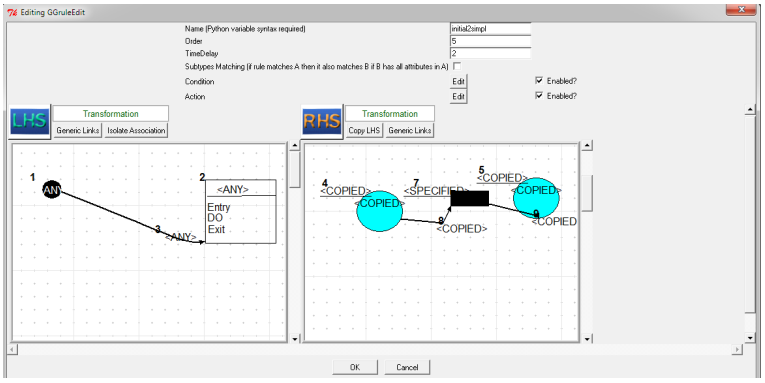

FIG. 5.26 : Règle 7: Création Relation entre etat-initial et un etat-simple.

5.5.8 Règle 8

Cette règle permet de transformer "une relation" qui relié etat-initial avec un aspect d'un diagramme d'etat-transition orienté aspect de la partie gauche LHS en « une relation » relie une place représente l'etat-initial et une autre place représente l'aspect-simple dans le réseau de pétri dans la partie droite RHS

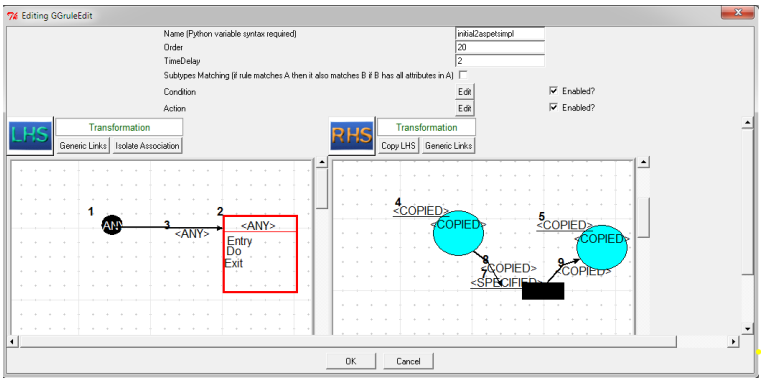

FIG. 5.27 : Règle 8: Création Relation entre une etat-initial et un aspect-simple.

5.5.9 Règle 9

Cette règle permet de transformer "une relation" qui relié un eta-initial avec un etat-composite d'un diagramme d'etat-transition orienté aspect de la partie gauche LHS en « une relation » relié une place qui représente l'etat-initial et un autre place qui représente un etat-composite dans le réseau de pétri dans la partie droite RHS

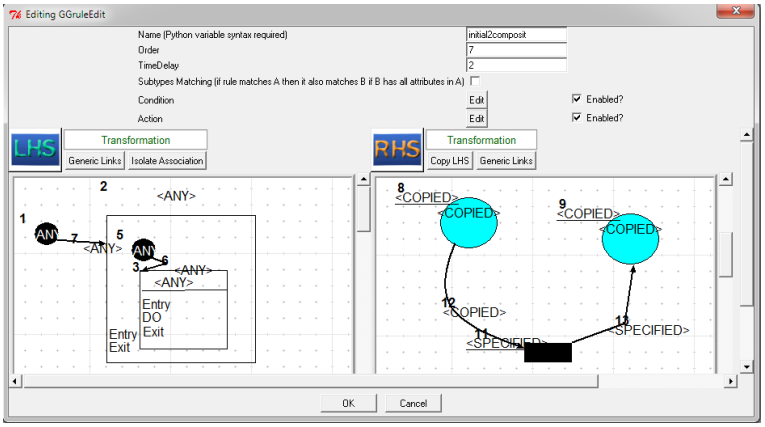

FIG. 5.28 : Règle 9: Création Relation entre un etat-initial et un etat-composite.

5.5.10 Règle 10

Cette règle permet de transformer "une relation" qui relié un etat-initial avec un aspect-composite d'un diagramme d'etat-transition orienté aspect de la partie gauche LHS en « une relation » relié une place qui représente etat-initial et une autre place qui représente un aspect-composite dans le réseau de pétri dans la partie droite RHS

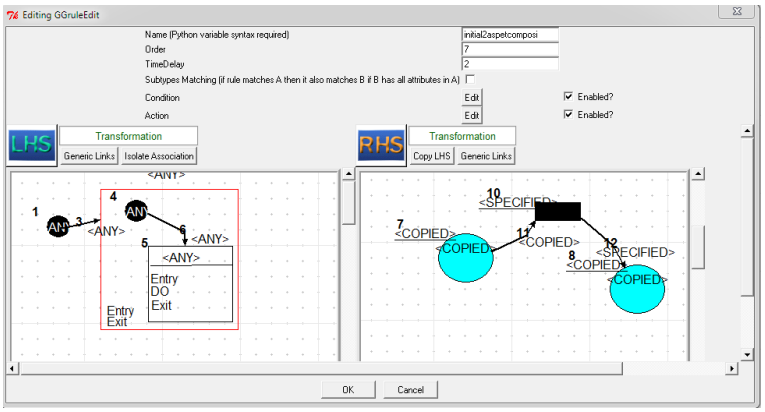

FIG. 5.29 : Règle 10: Création relation entre etat-initial et aspect-composite.

5.5.11 Règle 11

Cette règle permet de transformer "une relation" qui relié un etat-simple avec un autre etat-simple d'un diagramme d'etat-transition orienté aspect de la partie gauche LHS en « une relation » relie une place qui représente etat-simple et une autre place qui représente un etat-simple aussi dans le réseau de pétri dans la partie droite RHS

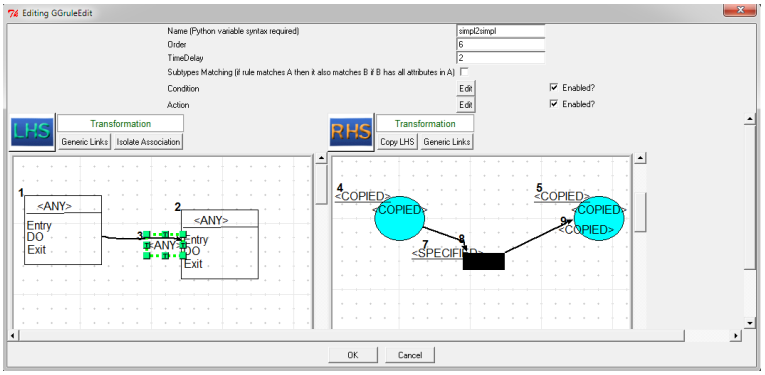

FIG. 5.30 : Règle 11: Création relation entre une etat-simple et une autre etat-simple.

5.5.12 Règle 12

Cette règle permet de transformer "une relation" qui relié un etat-simple à lui mème avec etat-simple d'un diagramme d'etat-transition orienté aspect de la partie gauche LHS en « une relation » relié une place à lui mème et une autre place qui représente etat-simple dans le réseau de pétri dans la partie droite RHS

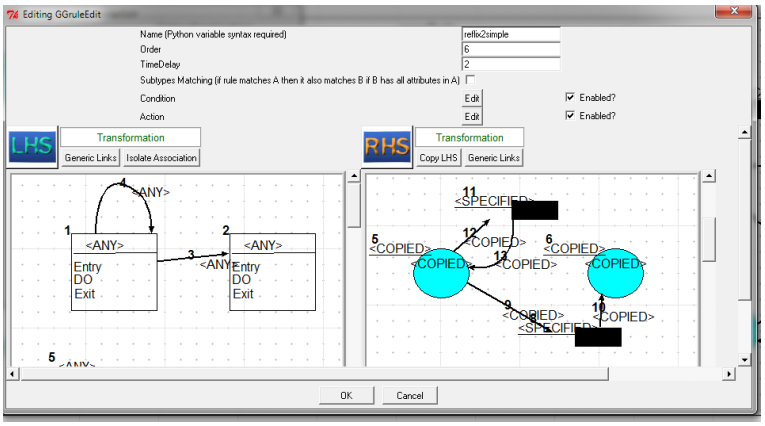

FIG. 5.31 : Règle 12: Création relation entre une etat-simple et lui mème.

5.5.13 Règle 13

Cette règle permet de transformer "une relation" qui relié un etat-simple à lui mème avec aspect-simple d'un diagramme d'etat-transition orienté aspect de la partie gauche LHS en « une relation » relié une place à lui mème et une autre place qui représente aspect-simple dans le réseau de pétri dans la partie droite RHS

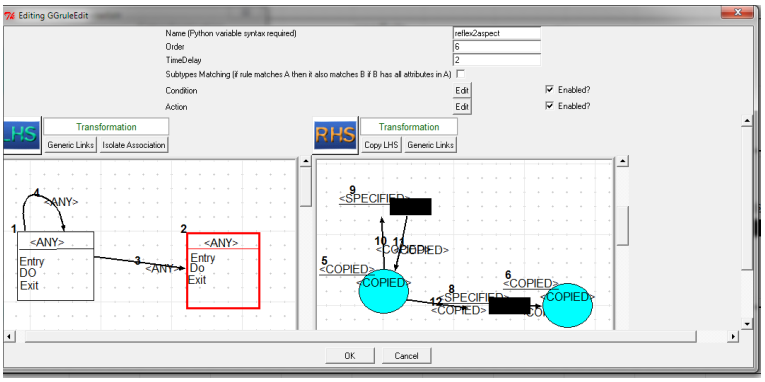

FIG. 5.32 : Règle 13: Création relation entre une etat-simple et lui mème.

5.5.14 Règle 14

Cette règle permet de transformer "une relation" qui relié un etat-simple avec un aspect-simple d'un diagramme d'etat-transition orienté aspect de la partie gauche LHS en « une relation » relie une place qui représente etat-simple et une autre place qui représente un aspect-simple dans le réseau de pétri dans la partie droite RHS

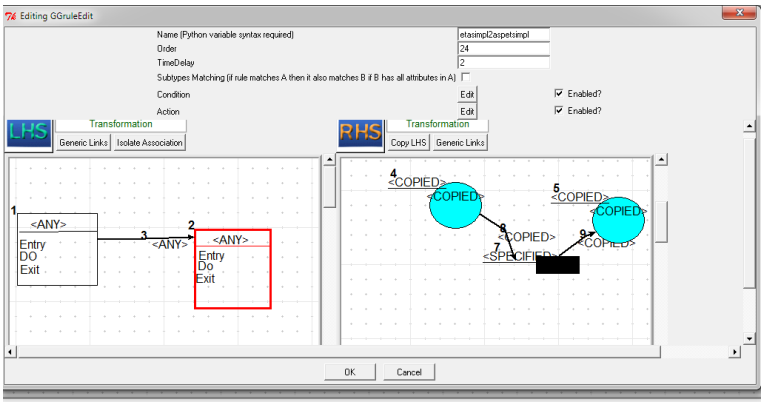

FIG. 5.33 : Règle 14: Création relation entre une etat-simple et une aspect-simple.

5.5.15 Règle 15

Cette règle permet de transformer "une relation" qui relié un etat-simple avec un etat-composite d'un diagramme d'etat-transition orienté aspect de la partie gauche LHS en « une relation » relie une place qui représente etat-simple et une autre place qui représente un etat-composite dans le réseau de pétri dans la partie droite RHS

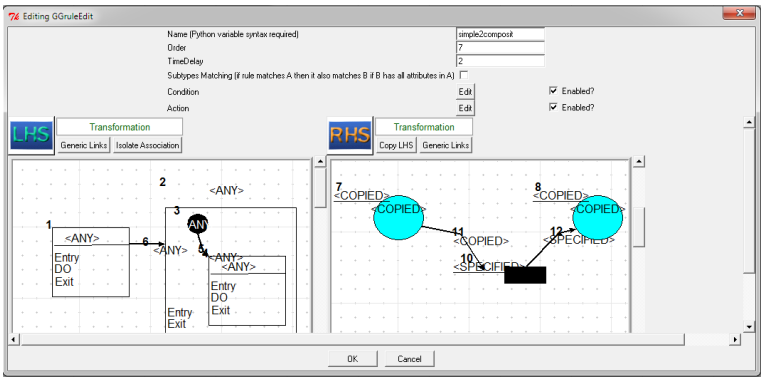

FIG. 5.34 : Règle 15: Création relation entre une etat-simple et une etat-composite.

5.5.16 Règle 16

Cette règle permet de transformer "une relation" qui relié un etat-simple avec un aspect-composite d'un diagramme d'etat-transition orienté aspect de la partie gauche LHS en « une relation » relie une place qui représente etat-simple et une autre place qui représente un aspect-composite dans le réseau de pétri dans la partie droite RHS

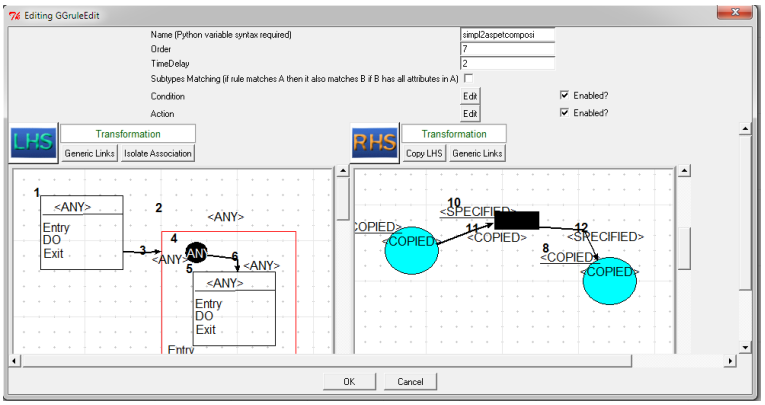

FIG. 5.35 : Règle 16: Création relation entre une etat-simple et une aspect-composite.

5.5.17 Règle 17

Cette règle permet de transformer "une relation" qui relié un etat-simple avec un etat-final d'un diagramme d'etat-transition orienté aspect de la partie gauche LHS en « une relation » relie une place qui représente etat-simple et une autre place qui représente un etat-final dans le réseau de pétri dans la partie droite RHS

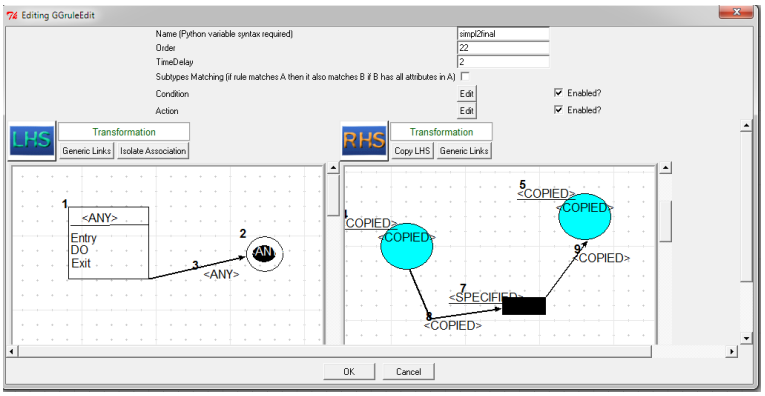

FIG. 5.36 : Règle 17: Création relation entre une etat-simple et une etat-final.

5.5.18 Règle 18

Cette règle permet de transformer "une relation" qui relié un etat-composite avec un etat-simple d'un diagramme d'etat-transition orienté aspect de la partie gauche LHS en « une relation » relie une place qui représente etat-composite et une autre place qui représente un etat-simple dans le réseau de pétri dans la partie droite RHS

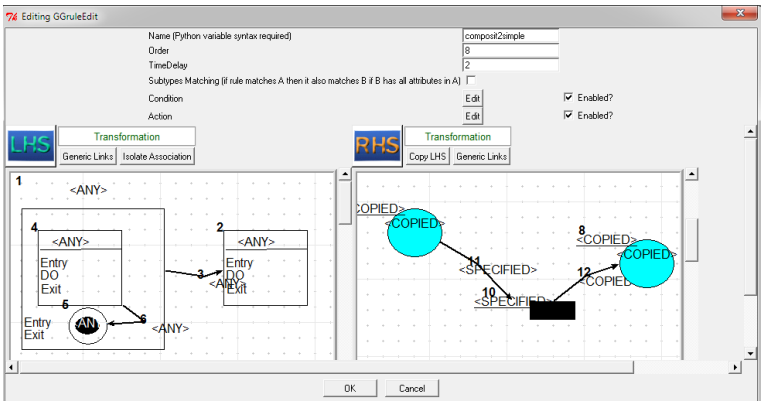

FIG. 5.37 : Règle 18: Création relation entre une etat-composite et une etat-simple.

5.5.19 Règle 19

Cette règle permet de transformer "une relation" qui relié un etat-composite avec un aspect-simple d'un diagramme d'etat-transition orienté aspect de la partie gauche LHS en « une relation » relie une place qui représente etat-composite et une autre place qui représente un aspect-simple dans le réseau de pétri dans la partie droite RHS

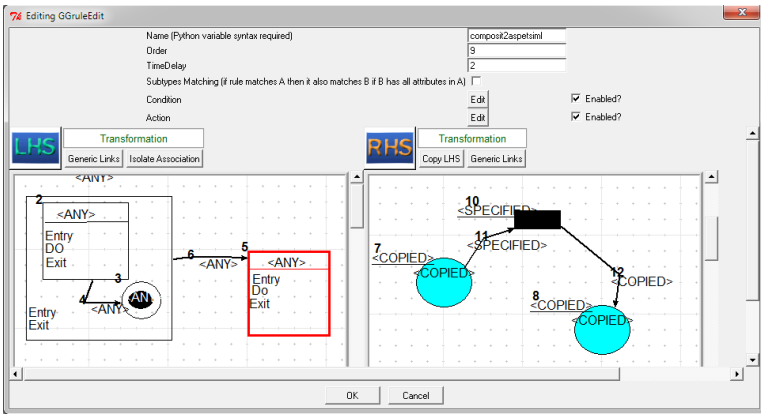

FIG. 5.38 : Règle 19: Création relation entre une etat-composite et une aspect-simple.

5.5.20 Règle 20

Cette règle permet de transformer "une relation" qui relié un etat-composite avec un etat-final d'un diagramme d'etat-transition orienté aspect de la partie gauche LHS en « une relation » relie une place qui représente etat-composite et une autre place qui représente un etat-final dans le réseau de pétri dans la partie droite RHS

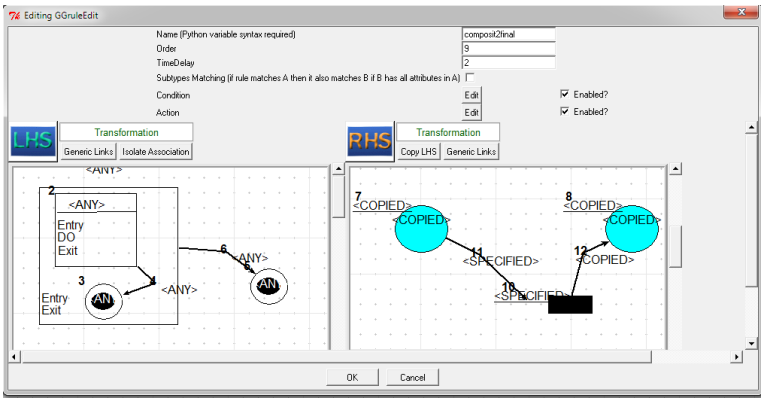

FIG. 5.39 : Règle 20: Création relation entre une etat-composite et une etat-final.

5.5.21 Règle 21

Cette règle permet de transformer "une relation" qui relié un aspect-simple avec un etat-simple d'un diagramme d'etat-transition orienté aspect de la partie gauche LHS en « une relation » relie une place qui représente aspect-simple et une autre place qui représente un etat-simple dans le réseau de pétri dans la partie droite RHS

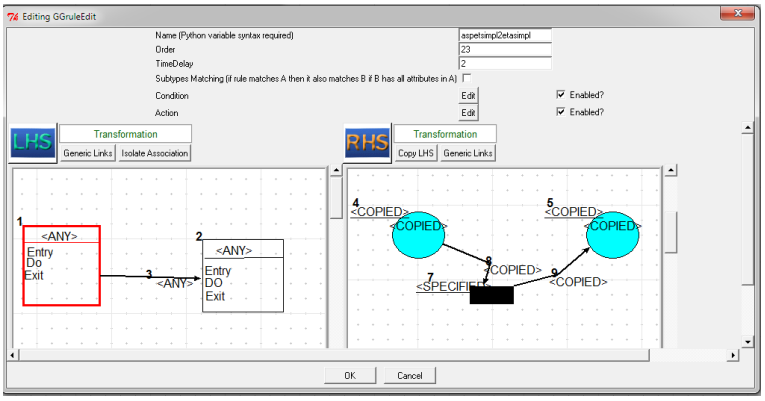

FIG. 5.40 : Règle 21: Création relation entre une aspect-simple et une etat-simple.

5.5.22 Règle 22

Cette règle permet de transformer "une relation" qui relié un aspect-simple avec un etat-composite d'un diagramme d'etat-transition orienté aspect de la partie gauche LHS en « une relation » relie une place qui représente aspect-simple et une autre place qui représente un etat-composite dans le réseau de pétri dans la partie droite RHS

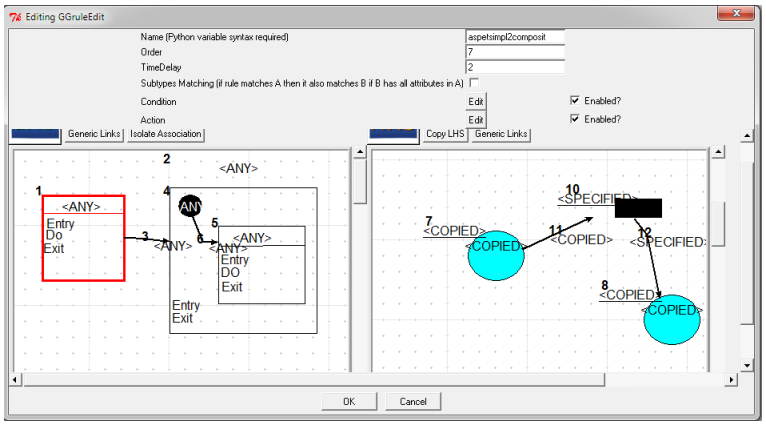

FIG. 5.41 : Règle 22: Création relation entre une aspect-simple et une etat-composite.

5.5.23 Règle 23

Cette règle permet de transformer "une relation" qui relié un aspect-simple avec un etat-final d'un diagramme d'etat-transition orienté aspect de la partie gauche LHS en « une relation » relie une place qui représente aspect-simple et une autre place qui représente un etat-final dans le réseau de pétri dans la partie droite RHS

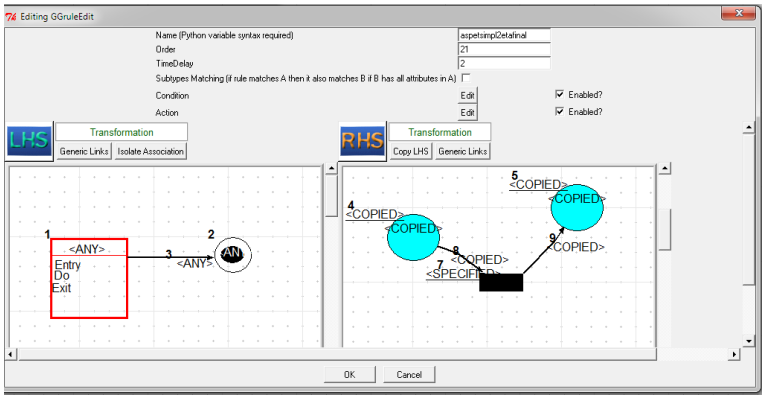

FIG. 5.42 : Règle 23: Création relation entre une aspect-simple et une etat-final.

5.5.24 Règle 24

Cette règle permet de transformer "une relation" qui relié un aspect-composite avec un etat-simple d'un diagramme d'etat-transition orienté aspect de la partie gauche LHS en « une relation » relie une place qui représente aspect-composite et une autre place qui représente un etat-simple dans le réseau de pétri dans la partie droite RHS

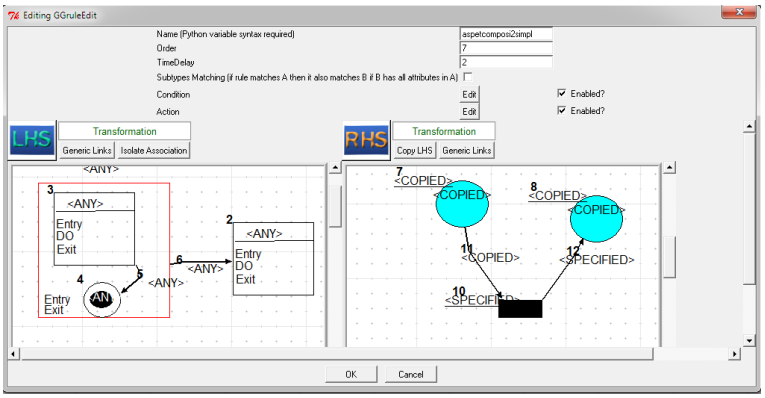

FIG. 5.43 : Règle 24: Création relation entre une aspect-composite et une etat-simple.

5.5.25 Règle 25

Cette règle permet de transformer "une relation" qui relié un aspect-composite avec un etat-final d'un diagramme d'etat-transition orienté aspect de la partie gauche LHS en « une relation » relie une place qui représente aspect-composite et une autre place qui représente un etat-final dans le réseau de pétri dans la partie droite RHS

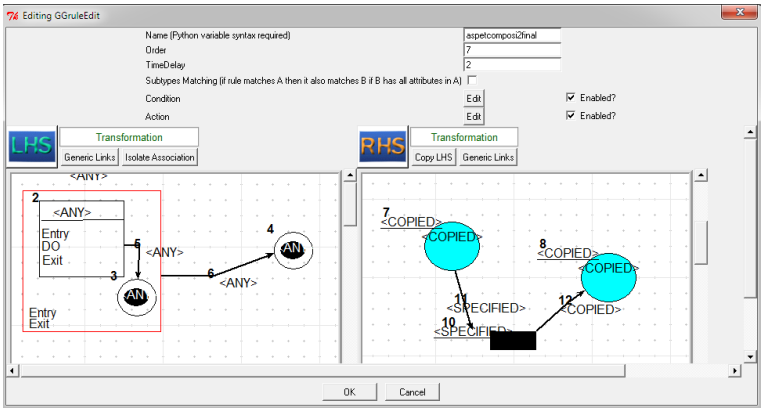

FIG. 5.44 : Règle 25: Création relation entre une aspect-composite et une etat-final.

5.5.26 Règle 26

Cette règle permet de supprimer le diagramme d'etat-transition orienté aspect.

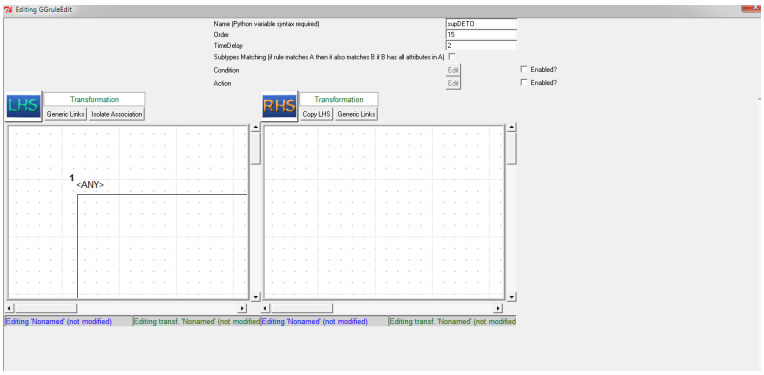

FIG. 5.45 : Règle 26: supprimer le diagramme d'etat-transition orienté aspect.

5.5.27 Règle 27

Cette règle permet de supprimer « les etats-simples » qui existe dans le diagramme d'etat-transition orienté aspect.

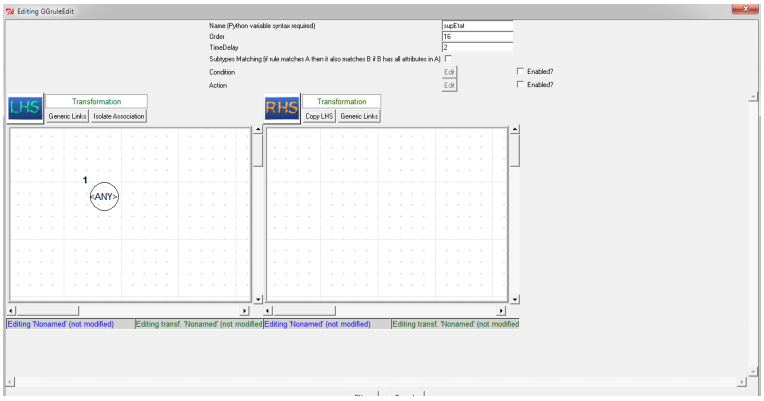

FIG. 5.46 : Règle 27: supprimer les etats-simples.

5.5.28 Règle 28

Cette règle permet de supprimer « etat-initial » qui existe dans le diagramme d'etat-transition orienté aspect.

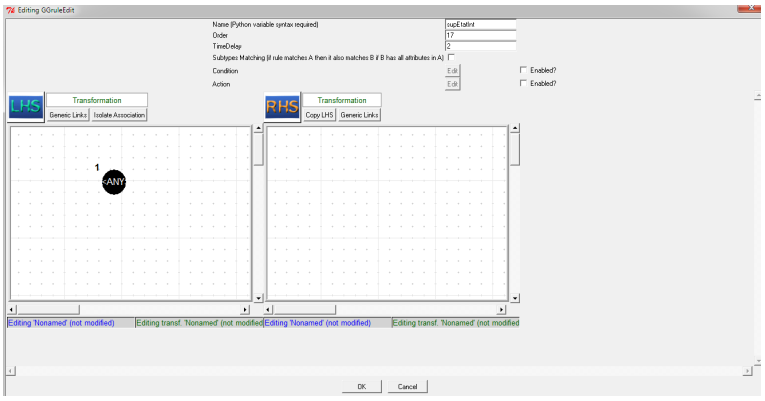

FIG. 5.47 : Règle 28: supprimer etat-initial.

5.5.29 Règle 29

Cette règle permet de supprimer « etat-final » qui existe dans le diagramme d'etat-transition orienté aspect.

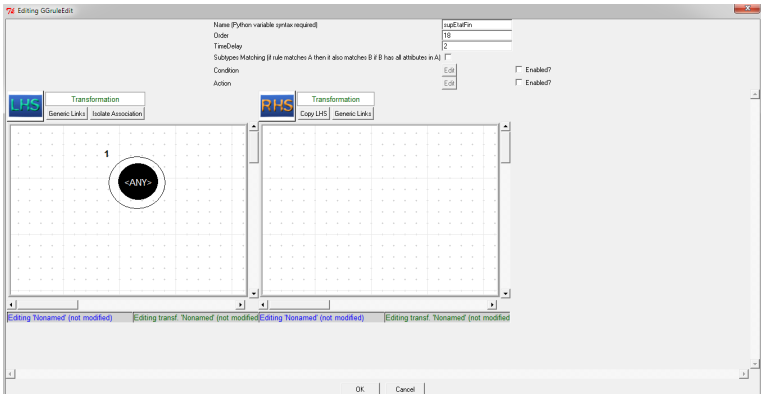

FIG. 5.48 : Règle 29: supprimer etat-final.

5.5.30 Règle 30

Cette règle permet de supprimer « aspect-simple » qui existe dans le diagramme d'etat-transition orienté aspect.

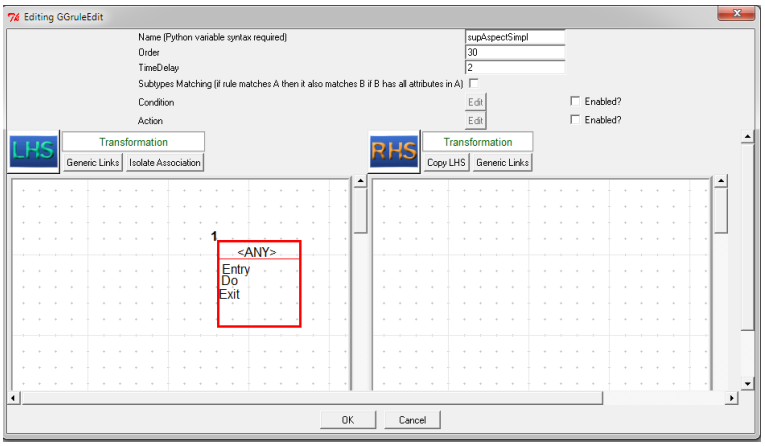

FIG. 5.49 : Règle 30: supprimer aspect-simple.

5.5.31 Règle 31

Cette règle permet de supprimer « aspect-composite » qui existe dans le diagramme d'etat-transition orienté aspect.

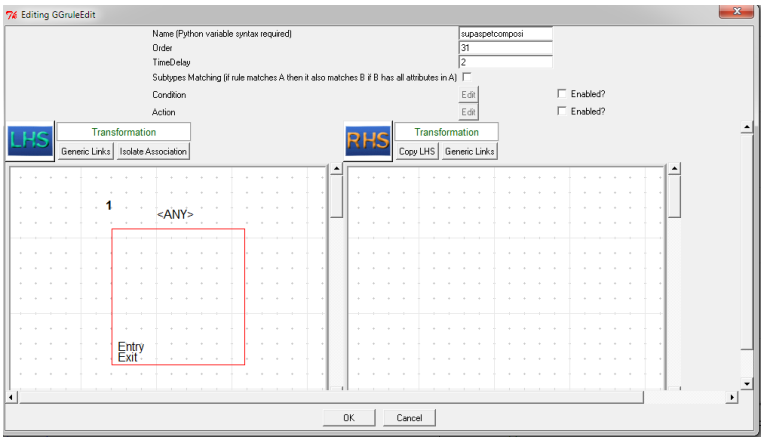

FIG. 5.50 : Règle 31: supprimer aspect-composite.

5.6 Conclusion

Dans ce chapitre, nous avons proposé une approche automatique basée sur la transformation de graphes pour générer un reseau de petri à partir d'un diagramme etat-transition orienté aspect. Cette approche est basée sur la méta-modélisation (les méta-modèles).

Donc, nous avons proposé dans chaque approche la défenition du méta-modèle pour le modèle d'entrée ainsi que le méta-modèle du modèle de sortie. Ensuite nous avons présenté une grammaire de graphes "ensemble de règles" permettant la réalisation de la transformation. Cette transformation a été efectuée à l'aide de l'outil de modélisation et de méta-modélisation AT oM3 et les contraintes ont été exprimées en python. dans le chapitre suivant, nous allons illustrer des exemples de transformation bien discuté dans la littérature.

Chapitre 6

Etude de cas sur la transformation de diagramme état-transition orienté-aspect vers réseau de pétri

6.1 Introduction

Dans ce chapitre, nous présentons des études de cas sur la transformation des diagrammes d'états-transitions oriente aspect vers reseau de petri ,nous commençons par l'application de notre approche de transformation sur deux études de cas. Le premier sur le comportement d'un distributeur automatique d'argent,le deuxième sur la fenêtre principale d'un serveur de messagerie.

6.2 Etude de cas

Afin de mettre en évidence notre approche nous avons opté pour l'étude de cas deux exemples pour montrer les étapes de transformation (les règles) en utilisant l'outil AToM3. Nous avons appliqué notre approche sur « le comportement d'un distributeur automatique d'argent » Ainsi sur le comportement d'une fenêtre principale d'un serveur de messagerie

6.2.1 Exemple1: etude de cas sur le comportement d'un distributeur automatique d'argent

Dans la Figure suivante, nous présentons la modélisation du comportement de ce processus avec le diagramme d'etat-transition oriente aspect qui est notre modèle de base.

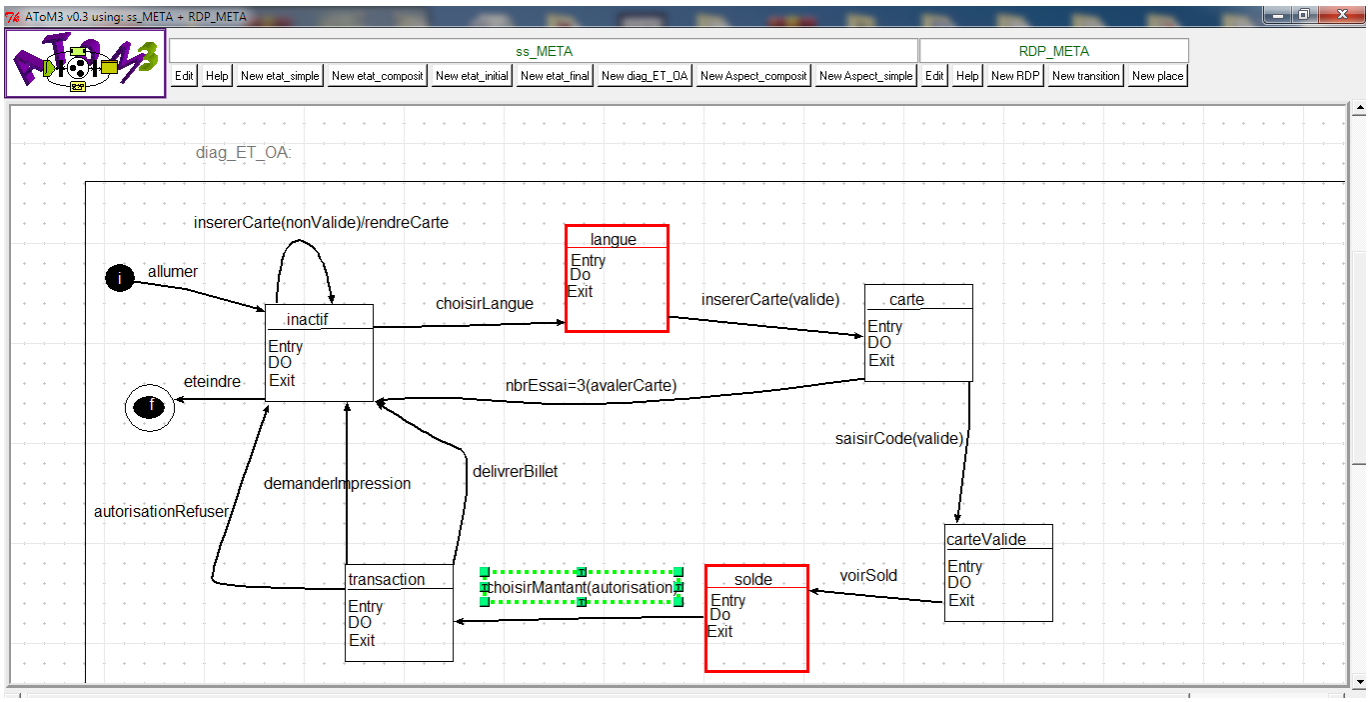

FIG. 6.1 : Diagramme d'etat transition orienté aspect pour "distributeur automatique d'argent".

Après l'application des règles de grammaire on obtient le modèle cible présenté dans la figure suivant.

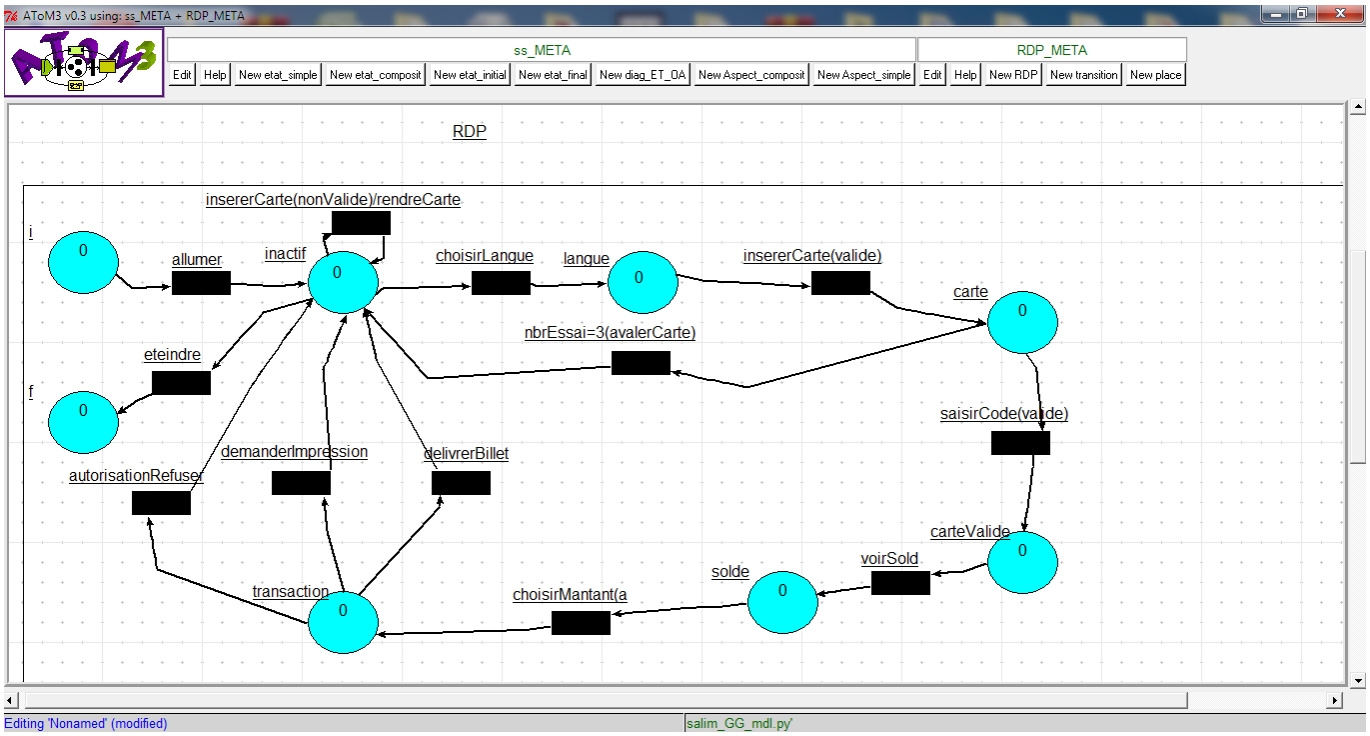

FIG. 6.2 : Réseaux de pétri pour"distributeur automatique d'argent".

6.2.2 Exemple 2: etude de cas sur le comportement d'une fenêtre principale d'un serveur de messagerie

Dans la Figure suivante, nous présentons la modélisation du comportement de ce processus avec le diagramme d'etat-transition orienté aspect qui est notre modèle de base.

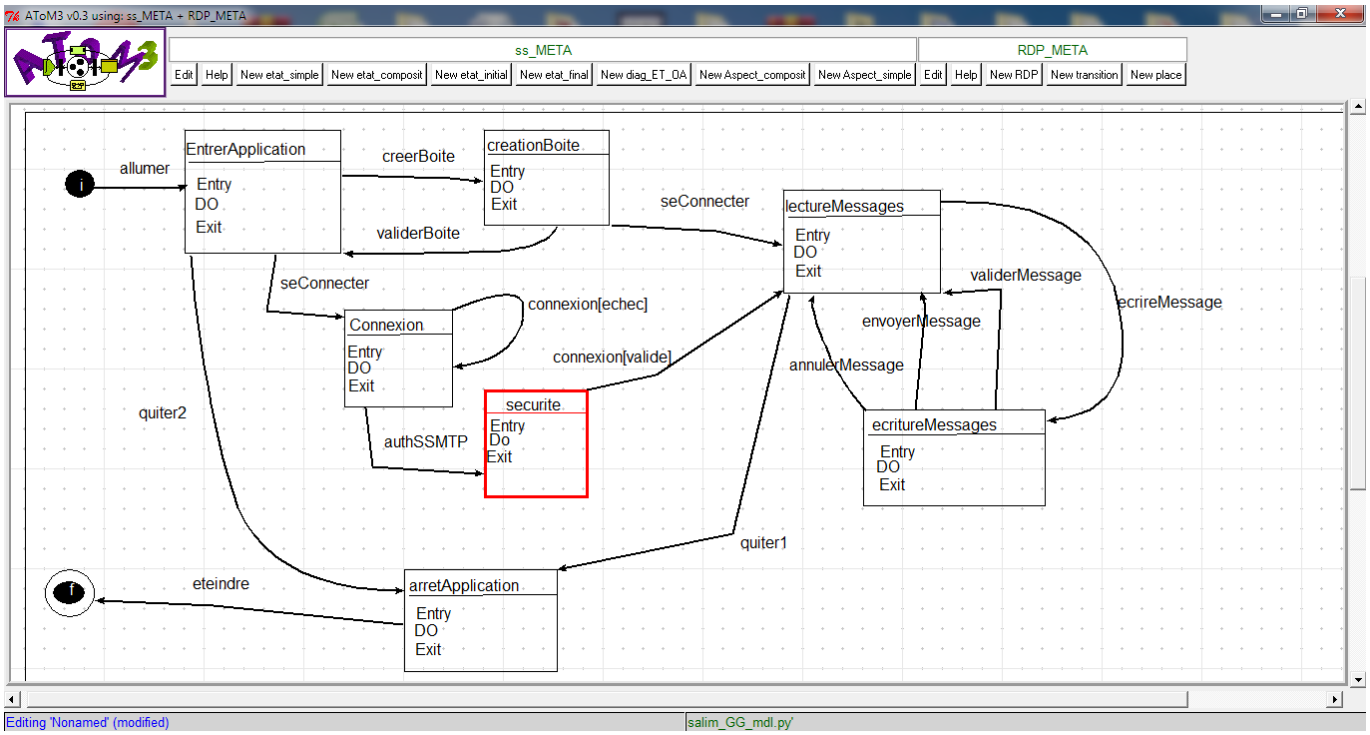

FIG. 6.3 : Diagramme d'etat transition orienté aspect pour "fenêtre principale d'un serveur de messagerie".

Après l'application des règles de grammaire on obtient le modèle cible présenté dans la figure suivant.

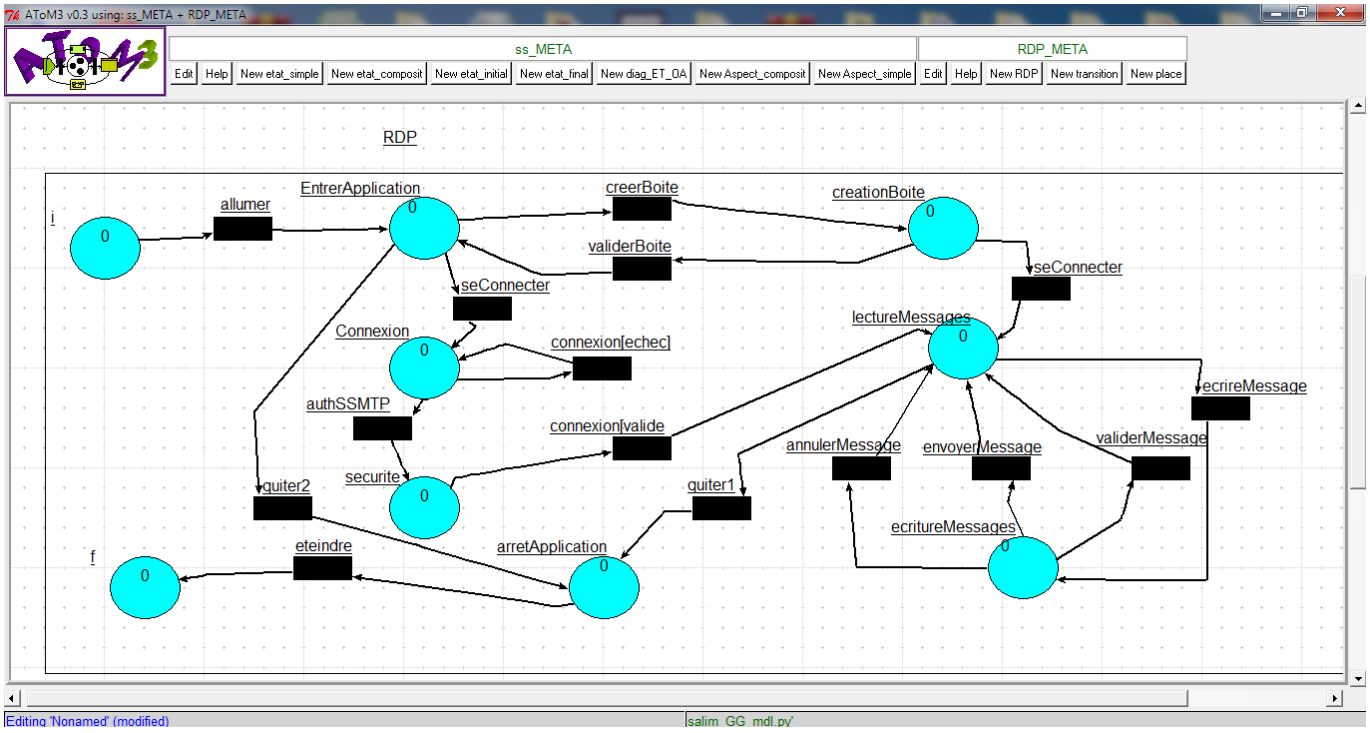

FIG. 6.4 : Réseaux de pétri pour "fenêtre principale d'un serveur de messagerie".

6.3 conclusion

Dans ce chapitre, nous avons présenté deux cas d'études afin d'illustrer notre approche de transformation. Le premier sur le comportement d'un distributeur automatique d'argent. Le deuxième sur le comportement d'une fenêtre principale d'un serveur de messagerie. Finalement nous avons montré l'efficacité de notre approche à travers les résultats obtenus.

Chapitre 7

Conclusion générale

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.

Le résultat de notre travail est une approche automatique pour transformer le diagramme d'etat-transition orienté aspect 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 ATOM³

Le travail est réalisé dans deux étapes :

La première étape consiste à proposer deux méta-modèles : diagramme d'etat-transition orienté aspect, et Rdp afin de générer un outil visuel permettant la modélisation de ce diagramme.

La deuxième étape propose une grammaire de graphe permettant de transformer le diagramme d'etat-transition orienté-aspect graphique vers une reseau de pétri.

Comme perspectives, nous comptons dans un premier lieu, de continuer la transformation des arcs et des etat non encore transformés dans le travail présenté , afin d'arriver à une transformation complète de diagramme d'etat-transition orienté-aspect.

Finalement, et afin d'arriver à développer une approche totalement automatique, incluant tous les diagrammes UML, nous proposerons de continuer la transformation des autres diagrammes UML orienté-aspect(diagramme de séquence, diagramme de cas d'utilisation, diagramme d'activite, etc …) vers les Rdp en utilisant toujours la transformation de graphes et l'outil ATOM³.

Bibliographie

Bibliographie

1. [Mostefaoui,2008] :Farida Mostefaoui ,(2008). Un cadre formel pour le développement orienté aspect :modélisation et vérification des interactions dues aux aspects,Thèse de doctorat,Université de Montréal.

2. [Brichau et al,2005] :J.Brichau,T.Hondt ,(2005). "Introduction to Aspect-Oriented Software Development",AOSD-Europe,these de doctorat,université Northeastern.

3. [Schauerhuber et al,2011] :A. Schauerhuber,W. Schwinger,E. Kapsammer, W. Retschitzegger, M. Wimmer and G. Kappel1 ,(2011).A Survey on Aspect-Oriented Modeling Approaches,informatics group,université vienna.

4. [Object Management Group,2005] :unified modeling language : Superstructure .version 2.0,August 2005.

5. [Cottenier et al ,2006] :T.Cottenier ,A.VanDenBerg ,T.Elard Mdeling, (2006). Aspect-Oriented Composition,université Linz austria.

6. [Zerara et al,2020] :A.Zerara,F.Megrous ,(2020). La génération d'un outil de transformation desdiagrammes UML2.0 vers les diagrammes orientés aspect ,basé sur la transformation de graphes,Memoire de master , centre universitaire mila.

7. [Amroune,2014] :M.Amroune ,(2014).vers une approche orientée aspect d'ingénierie des besoins dans les organisations multi-entreprise ,these de doctorat,Université Toulouse 2 Jean Jaurès .

8. [AOUAG ,2014] :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.

9. [Kiezales et al,1997] :Kiczales.G,lamping.J,Mendhekar.A,Maeda.C.Lopes ,(1997). Aspect-oriented programming, European Conference on Object-Oriented Programming.

10. [Loigtier,1999] :C.V.Loigtier ,(1999). Aspect oriented programming , european conference on object oriented programming.

11. [Gil ,2006] :T.Gil (2006),La conception oriente aspect ,Valtech Training.

12. [Kerkouche,2011] :E.Kerkouche ,(2011). Modélisation Multi-Paradigme : Une Approche Basée sur la Transformation de Graphes,these de doctorat ,universite de mentouri constatine.

13. [Pétri,1992] :C.A.Petri ,(1992).kommunikation mit automation (in Germany), phd thèse université of bomme.

14. [Hala ,2004] :Hala ,(2004) .Discrete, Continuous, and Hybrid petri Net, these de doctorat,Université Joseph-Fourier - Grenoble I.

15. [Murata,1989] :T.Murata ,(1989) .Petri Nets : Propretés, analysis and applications, proceeding of the IEEE.

16. [J.L.Peterson,1981] :J.L.Peterson (1981) ,Petri Net Theory and The Modeling of System,Practice-hall,Englewood Chiffe.

17. [V. Murilo Savi,1994]V. Murilo Savi ,(1994) .Conception Préliminaire des systèmes de production à l'aide des réseaux de pétri,these de doctorat,université de metz.

18. [Elmansouri,2009] :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.

19. [Memi et al,1980] :G.Memi,M.Silva ,(1980) .Linéaire algèbre in Net Théorie, Net Théorie and Application, lecteur Notes in capter science n=84 ,Springer verlag.

20. [Lautenbach,1986] : K.Lautenbach ,(1986).Linear algébrique techniques for place transition nets,Net Théorie and application, Lecture Notes in Computer Science n=254, Springer Verlag.

21. [Martinez et al,1985] :L.Martinez et M.Silva ,(1985).A simple and faste algorithme to obtain all invariants of a généralised pétri net,Lecture Note in Computer Science n=52, Springer-verlag,Berlin, FRG.

22. [Shib et al,1991] :H.Shib and T.sekiguchi ,(1991).A timed pétri net ans beam search based online FMS scheduling with routing flexibilité, Proceeding of the 1991 IEEE international conférence on robotiques and automation.

23. [Laouar,2013] :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.

24. [Dehimi,2014] :N.Dehimi, (2014). Un cadre formel pour la modélisation et l'analyse des agents mobiles.Thèse de doctorat. Université de mentouri Constantine.

25. [Hettab,2009] :A.Hettab ,(2009). De M-Ml vers les réseaux de pétri « Nested Nets » : une approche basée sur la transformation de graphes. Thèse de doctorat. Université de mentouri Constantine.

26. [Torchi et al,2018] :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.

27. [A.Etienne ,2007] :A.Etienne ,(2007). Anne Etien, Cedric Dumoulin,Emmanuel Renaux. Towards a Unified Notation to Represent Model Transformation. Research Report RR-6187, INRIA.

28. [J.Bézivin,2005] :J.Bezivin ,(2005). On the unification power of models. Software Systems,these de doctorat,université San Diego USA.

29. [L.Menet, 2010] : L.Menet ,(2010). “Formalisation d’une approche d’Ingénierie Dirigée par les Modèles appliquée au domaine de la Gestion des Données de Référence (in french)”,these de doctorat, Université PARIS VIII.

30. [Guerrouf] :F.Guerrouf ,(2009). “Une approche de transformation des Diagrammes d’Activités d’UML Mobile 2.0 vers les Réseaux de Petri,memoire de magistere,Université El Hadj Lakhdar BATNA.

31. [Berramla,2014] :K. Berramla ,(2014). Vérification et Validation des Transformation de Modèles,these de doctorat,Université El Hadj Lakhdar BATNA.

32. [Favre et al ,2006] : Jean-Marie Favre, Jacky Estublier, and Mireille Blay-Fornarino, (2006).editors. L’ingénierie dirigée par les modèles : au-delà du MDA. Hermès-Lavoisier, Cachan, France. ISBN 2-7462-1213-7.

33. [Ehrig et al,2007] : 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.

34. [Khalfaoui, 2014] : k.khalfaoui, (2014). Une Approche de Spécification des Changements des Besoins Basée Transformation de Graphes. Thèse de doctorat, UNIVERSITE MOHAMED KHIDER BISKRA.

35. [Rozenberg,1999] :G.Rozenberg, (1999). Handbook of graph grammars and computing by graph transformation. World Scientific, 1.

36. [Burmester et al ,2004] : S. Burmester, H. Giese, J. Niere, M. Tichy, J.P. Wadsack, R. Wagner, L.Wendehals, and A. Zündorf,(2004). “Tool Integration at the Meta-Model Level within the Fujaba Tool Suite”, International Journal on Software Tools for Technology Transfer, vol. 6(3), pp. 203-218.

37. [AGG, 2010] AGG (2010). http ://tfs.cs.tu-berlin.de/agg/.

38. [Konigs, 2005] : A. Konigs,(2005). “Model Transformations with Triple Graph Grammars”, in Model Transformations in Practice Workshop at MoDELS, Jamaica.

39. [Balasubramanian et al ,2006] : D. Balasubramanian, A. Narayanan, C. vanBuskirk and G. Karsai,(2006). “The Graph Rewriting and Transformation Language : GReAT”, In Proceedings of the 3rd International Workshop on Graph Based Tools, Brazil.

40. [Vangheluwe et al , 2002] : J. De Lara and H. Vangheluwe,(2002). “AToM3 : A Tool for Multi-Formalism Modelling and Meta-Modelling”, LNCS, Springer, vol. 2306, pp. 174-188.

41. [Bendiaf,2018] : 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.

42. [Saidi et al,2018] : B.Saidi,B.Razika, (2018).Une approche de Transformation Des Diagrammes D'activités UML Vers Les Réseaux De Pétri Temporellement Temporisés .Mémoire de master .Université de Djilali Bounaama Khemis Miliana.

43. [Andries et al ,1999] : M.Andries, Gregor Engels,(1999). Annegret Habel, Berthold Hoffmann, Hans-Jorg Kreowski, Sabine Kuske, Detlef Pump, Andy Schürr and Gabriele Taentzer, "Graph transformation for specification and programming", Science of Computer programming, vol 34, NO°1, pages 1-54. .

44. [Nacera, 2015] : M.Nacera, (2015). "Une approche hybride pour transformer les modèles",mémoire de master ,université d'oran .

45. [Steinberg et al ,2008] : D.Steinberg, F.Budinsky, E.Merks e tM.Paternostro,(2008). Emf : eclipse modeling framework. Pearson Education.

46. [Schurr,1995] : A.Schürr,(1995). Specification of graph translators with triple graph grammars. In Graph- Theoretic Concepts in Computer Science, pages 151–163. Springer, .

47. [Kindler et al ,2007] : E.Kindler and Robert Wagne,(2007). Triple graph grammars : Concepts, extensions, implementations, and application scenarios. Technical Report tr-ri-07-284, Technical Report, University of Paderborn, D-33098 Paderborn, Germany, .

48. [Konigs,2005] : A.Konigs,(2005). Model transformation with triple graph grammar. ,these de doctorat ,universite san diego usa.

49. [Berramla,2014] : Berramla,Karima ,2014,Vérification et Validation des transformations des modèles,mémoire de magistère,université d'oran.

50. [mellouli,2014] : Sana Mellouli ,2014,Méta-modélisation du Comportement d'un Modèle de Processus : Une Démarche de Construction d'un Moteur d'Exécution ".thèse de doctorat,université de paris.