الجمهورية الجزائرية الديمقراطية الشعبية

MINISTERE DE L'ENSEIGNEMENT SUPERIEUR
ET DE LA RECHERCHE SCIENTIFIQUE

UNIVERSITE DR YAHIA FARES DE MEDEA

FACULTE DES SCIENCES

Département du Mathématique et informatique



Mémoire de fin d'Études

Présenté pour l'obtention du Diplôme de Master en Informatique
Option : Ingénierie des Systèmes et Technologie de Web

THÈME

Une démarche MDA pour la composition des services web

Proposé et dirigé par,

Mr : Toubal El Bahi

Préparé et présenté par,

Mlle : Hamidia Karima

Mlle : Mokrane Semeia

Promotion 2017

Remerciment

Avant tous : Nous remercions

ALLAH de nous avoir aidé à faire notre travail.

Nous tenons tous d'abord à exprimer notre remerciement à notre parents pour leur courage durant toute la formation.

Nous exprimons notre chaleureux remerciement à notre promoteur Mr : TOUBAL El Bahi pour son temps qu'il a partagé avec nous et ses conseils durant la période de notre travail.

Aussi nous exprimons nos très sincères remerciements à Mr MAZARI Redha, Mr KHALDOUN et Mr Rahim qui nous aide à réaliser notre projet qui nous a été si précieuse.

Aussi nous remercions Madame BENILLES Ihsene pour son aide durant le travail.

Nous adressons nos remerciements à tous les enseignants d'université, et toutes les personnes qui nous aident à réaliser notre projet.

Enfin nous dédions les membres des jurys pour accepter de corrigé notre projet de fin d'étude.

Dédicace

Je dédié ce travail

Avant tous le monde à mes parents Vous êtes pour moi une source de vie car sans vos sacrifices, votre tendresse et votre affection je ne pourrais arriver jusqu'au bout. Je me réjouis de cet amour filial. Que Dieu vous garde afin que votre regard puisse suivre ma destinée.

A mes frères : Hicham, Mohamed, Bilal et Abd el basset & mes chères sœurs : Fatma, Asma et son marié Abd El Nacer et leur fils Kinan et toute ma famille qui ont été toujours présent pour moi.

A mon chère ami Ali.

A mes amis avec qui j'ai partagé des moments des plus agréables : Sara, Lamia, Samira, Imene,Abir, Wafia, Nor et Mahdi.

A ma très chère amie et consort dans ce projet : Karima.

A tous ceux qui sont chères, proches de mon cœur, et à tous ceux qui m'aiment et qui aurait voulu partager ma joie…

Soumia

Dédicace

Je dédié ce travail

Avant tous le monde à mes parents Vous êtes pour moi une source de volonté pour récusé dans ma formation, c'est grâce à votre tendresse et votre affection j'ai arrivé jusqu'au bout. Je me réjouis de cet amour filial. Que Dieu vous protège afin de remplir ma vie de joie et de bonheur.

A mes frères : Hakim et Abd el karim et mes chères sœurs : Faiza, Djamila et son marié Mohamed

Et leur mignon fils Younes qui ont été toujours présent pour moi.

A mes amis avec qui j'ai partagé des moments des plus agréables : Abir, Wafia, Nor, Fatima zahra , fathia ,Nora.

A mon oncle Tarik et sa femme et tout la famille.

A ma belle et très chère amie, ma binôme dans ce projet : Soumia.

A tous ceux qui sont chères, proches de mon cœur, et à tous ceux qui m'aiment et qui aurait voulu partager ma joie…

Karima

Sommaire

Liste des figures…….....I

LISTE DES TABLEAUX…….....III

LISTE D'ACRONYME……...……IV

Résumé…….……...VII

Introduction générale. …….……..…….....1

Chapitre 1 Les services web……..……..3

I Introduction……..……4

II. Services web……...4

1.Définition…….…….....4

2. Architecture des services web……5

3. Protocoles utilisées par les services web…….....6

3.1 Couche de transport……7

3.2 Couche de communication (SOAP) …….....7

3.3 Couche de description. …….....7

3.5 Couche de découverte et de publication (UDDI) …….....8

III.Composition de services web……...…….....8

1.Définition. ……...…8

2.Description et fonctionnement…….....8

A.Orchestration……9

B.Chorégraphie……..11

3.Types de composition …….....12

4. Langages utilisés pour la composition……...12

VI.Conclusion……...……14

Chapitre 2 Présentation du langage BPMN …….……..…….……..…….……..……15

I. Introduction…….……..…….……..…….……..…….……..……...16

II.Processus métier…….……..…….……..…….……..…….……..…16

1.Définition…….……..…….……..…….……..…….……..……16

III.Le BPM et BPMN…….……..…….……..…….……..……17

1.Historique…….……..…….……..…….……..……...17

2.Définition…….……..…….……..…….……..…….……..17

Sommaire

3.Composants d'un diagramme BPMN……..……..……..……..……...18

a) Activité. ….……..……..……..……..……..……..……18

b) Evènement…….……...……...……...……...……...…19

c) Porte décisionnelle…..……..……..……..……..……..19

d) Lignes……..……..……..……..……..……..……....21

e) Connecteurs…..……..……..……..……..……..……21

IV.Conclusion…..……..……..……..……..……..……..…22

Chapitre 3 Approche dirigées par les modèles……..23

I. Introduction……24

1.L'ingéniere dirigée par les modèles (IDM) ……...24

2.Concept fondamentaux de l'ingénierie dirigée par les modèles……25

A.Modèle……..…….25

B.Méta-modèles et méta-modélisation……..……..…25

C. Transformation de modèles ……..……..……..……....26

C.1 Principe général de transformation des modèles……..……....26

C.2 Types de transformation des modèles.…….……...27

C.3 Paradigmes de transformation des modèles …… ……28

C.3.1 Approches modèle vers modèle (M2M : Modèle-to-modèle) …… ……28

C.3.2 Approches modèle vers code…… …….……..……..…..…...29

3. Approche MDA (Model driven Architecture) …….……..……..…….30

3.1.Types de modèles dans MDA…….……..……..30

4. Niveaux d'abstraction du MDA.……..……..……..……..……32

5. Approches dirigées par les modèles pour la composition des services…….....33

II. Conclusion…..……..……..……..……..……..……..……34

Chapitre 4 Diagramme d'activité…..……..……..……..……..……..……..35

I.Introduction…….……..……..……..……..……..……36

Définition ……..……..……..……..……..……..……36

1. Diagrammes de Structure Composites. ……..……..……..……..……37

2.Diagramme de comportement ou diagrammes dynamiques (BehaviorDiagram)……..37

2.1 Digramme d'activité…….……...…..……36

Sommaire

2.1.1 Définition……..……..……...…...……..…37

2.1.2 Intérêts des diagrammes d'activité…….……..……...……..……37

2.1.3 Element d'un diagramme d'activité…….38

2.1.3.1 Nœuds (nodes) ……..……..……...……..……..38

2.1.3.1.1 Nœud d'activité (ActivityNode) ….……..……..……..…38

Définition 1. Une activité….……..……...……..……..…38

Définition2. Un nœud d'activité….……..……...……..……..…38

B. Nœud de contrôle (control node) ……...……..……..……..39

B.1.Noeud initial (initial node) ……...……..……..……..39

B.2.Nœud final (final node) ……...……..……..……...39

Nœuds finaux d'activité (activity final node) ……...……..…….....40

Nœuds finaux de flux (flow final node) ……...……..…….....40

B.3. Nœud de fusion (mergenode) ……...……...…….....40

B.4. Nœud de décision (decisionnode) ……...……...……40

B.5. Nœud de bifurcation (forknode) ……...……...……..41

B.6. Nœud d'union (joinnode) ……...……...……...……..41

C .Nœud exécutable (executableNode) ……...……...……..……...…41

Action…...……...……...……...…...41

2.1.3.2 Acs (edges) …...……...……...…...……..42

Définition d'arc d'activité (ActivityEdge) …...……...……...…...….42

Flux d'objet (Object Flow) …...……...……...…...….42

Flux de contrôle (Control Flow) …...……...……...….....42

Exemple……43

II.Conclusion…...……...……...….....44

Chapitre 5 Conception…...…….…...……...….....45

I.Introduction …...……...……...….....46

1.Approche proposée……...……...….....46

1.1 Shéma de transformation d'un service composite……..46

1.2 Schéma globale de transformation ……...…….....47

1.3 Méta-modèles utilisés pour la transformation……...…….....48

Sommaire

A.Méta-modèle de diagramme d'activité……...……..…….....48

A.1 Description de méta-modèle de diagramme d'activité……...……..…..…48

B.Méta-modèle de BPMN……...……...……...……..…..……...50

B.1 Description de méta-modèle de BPMN……...……...……...…..…..50

1.4 Règles transformations……...……...……...……...……...52

1.5 Exemples de scenario algorithmique de transformation des règles……...……56

II.Conclusion ……...……...…..…….....57

Chapitre 6 Implémentation et étude de cas…….……..…58

I.Introduction……...……..…59

1.Schéma globale de transformation…….……59

2.Outils et langages utilisées…….……..……..…59

3.Méthodes utilisées dans la transformation……....60

4.Etude de cas……...61

4.1 Scenario de la composition……61

4.2 Représentation de scénario à l'aide de diagramme d'activité..……..62

5.Les interfaces……63

Page d'accueil de l'application……...……..63

Première interface de notre application……64

Deuxième interface de notre application……...65

Troisième interface de notre application……..……66

I.Conclusion……..……...……67

Conclusion générale……...……...67

Bibliographie……...…….…68

Webographie……...…….…70

Liste des figures

Figure1.1 l'architecture de base des services web 5
Figure 1.2 La pile Protocolaire simplifiée des services web 7
Figure 1.3 Comportement d'un service composite 9
Figure 1.4 Vue générale de l'orchestration 10
Figure 1.5 Exemple de scénario d'un service composite 10
Figure 1.6 Vue générale de la chorégraphie 11
Figure 2 .1 Axe de développement de BPMN 17
Figure 2.2 Processus de vente aux enchères modélisées par la notation BPMN 18
Figure 2.3 Branchement exclusif en entrée 19
Figure 2.4 Branchement exclusif en sortie 20
Figure 2.5 Branchement parallèle en entré 20
Figure 2.6 Branchement parallèle en sortie 21
Figure 2.7 Flux de séquence 21
Figure 3.1 Notion de base de l'ingénierie des modèles 26
Figure 3.2 Transformation endogène 27
Figure 3.3 Transformation exogène 27
Figure 3.4 Les approches de transformation de modèles 29
Figure 3.5 Les différents modèles dans MDA 31
Figure 3.6 Pyramide de modélisation de l'OMG 33
Figure 3.7 Classification des approches dirigées par les modèles 33
Figure 4.1 Les diagrammes d'UML 36
Figure 4.2 Notation nœuds d'activité 40
Figure 4.3 : Arbre de spécialisation des nœuds de contrôle 40
Figure 4.4 Notation de nœud initial 40
Figure 4.5 Notation de nœud final 41
Figure 4.6 Notation de nœud de fusion 41
Figure 4.7 Notation de nœud de décision 42
Figure 4.8 Notation de nœud de bifurcation 42
Figure 4.9 Notation de nœud d'union 42
Figure 4.10 Exemple illustre l'utilisation des nœuds de contrôle 43
Figure 4.11 Notation d'Action 44
Figure 4.12 Notation flux de contrôle 44
Figure 5.1 Schéma de transformation d'un service composite 46

Liste des figures

Figure 5.2 Schéma globale de transformation 47
Figure 5.3 Méta-modèle de diagramme d'activité 49
Figure 5.4 Méta-modèle de BPMN 51
Figure 6.1 Schéma global de transformation 59
Figure 6.2 Syntaxe de déclaration des helpers 60
Figure 6.3 Syntaxe de déclaration d'une règle 60
Figure 6.4 Schéma explicatifs de notre cas d'étude 61
Figure 6.5: Diagramme d'activité pour l'orchestration des services 62
Figure 6.6 Représentation de la page d'accueil 63
Figure 6.7 Importation de modèle de diagramme d'activité UML 64
Figure 6.8 Fichier (.xmi) générer a partir de fichier (.uml**)** 65
Figure 6.9 Graphe BPMN générer 66
Figure 6.10 Graphe BPMN générer à l'aide de BPMN Modeler 67

Liste des tableaux

Tableau 2.1 Les composantes des activités..... 18

Tableau 2.2 Les évènement……..19

Tableau 5.1 Règle 1 InitialNode vers StartEvent..... 52

Tableau 5.2 Règle 2 Activité vers Task 52

Tableau 5.3 Règle3 Fork Node vers And Splite. 53

Tableau 5.4 Règle 4 Join Node vers And Join 53

Tableau 5.5 Règle 5 Decision Node vers Xor Splite. 54

Tableau 5.6 Règle 6 Merge Node vers Xor Join. 54

Tableau 5.7 Règle 7 Send Signal Action vers intermidare Event « Message request ». 55

Tableau 5.8 Règle 8 Accepte Event Action vers Intermidiare Event « Message responce...55

Tableau 5.9 Règle 9 Final Node vers End Event. 56

Tableau 5.10 Règle10 Control Flow vers Flow…….56

Liste d’acronymes

Mot	Description
ADM	Architecture Dirigée par les Modèles
API	Application Programming Interface
AS	Action Semantic
ATL	Atlas Transformation Language
BPD	BPMN Diagram
BPDM	Business Process Definition Méta_model
BPEL	Business Process Exécution Language
BPM	Business Process Management
BPMI	Business Process Management Initiative
BPMN	Business Process Model and Notation
BPMS	Business Process Management Suite
CIM	Computation Independent Model
CWM	Common Warehouse Metamodel
EMF	Eclipse Modeling Framework
EPC	Event Process Chain
HTTP	HyperText Transfer Protocol
IBM	International Business Machines Corporation
IDEF	Integration DEFinition Methods
IDM	Ingénierie Dirigée par les Modèles
JMI	Java Metadata Interface
MDE	Model Driven Engineering
MOA	Maitrise d’ouvrage
MOF	Meta Object Facility
OCL	Object Constraint Language
OMG	Object Management Group
OMT	Object Modeling Technique
OOSE	Object Oriented Software Engineering
OSSAD	Office Support System Analysis and Design
PIM	Platform Independent Model

Liste d'acronymes

PSM	Platform Specific Model
RPC	Remote Procedure Call
SOA	Service Oriented Architecture
SOAP	Simple Object Access Protocol
UDDI	Universal Description Discovery and Integration
UML	Unified Modeling Language
URI	Uniform Resource Identifier
W3C	Consortium du World Wide Web
WfMC	Workflow Management Coalition
WSCI	Services Web Chorégraphie Interface
WSCL	Web Services Conversation Lagunage
WSDL	Web Services Description Language
WSFL	Web Service Flow Langage
XLANG	Web Services for Business Process Design
XML	Extensible Markup Language
XMI	XML Metadata Interchange

ملـــــخـص:

الأساليب الحالية لتحليل أنظمة العمل، تقوم على تمثيل الأعمال في شكل نماذج تطبيقية. وهناك معيار لتمثيل هذه النماذج هو التمثيل على شكل BPMN. و هذا يهدف إلى الجمع بين محللي الميدان. ولذلك فهو يعتبر مدخل لمشاريع تطوير البرمجيات.

وفي الوقت نفسه، أساليب النظم الموجهة للعملية تعتمد على تعريفات تفصيلية من العمليات التي يتم تنفيذها بواسطة محرك العملية.

و هذه التعريفات الخاصة بالعملية تقوم بصقل نماذج BPMN بإضافة تفاصيل التنفيذ.

في عملنا هذا فإننا نقترح تقنية اوتوماتكية على أساس التحويل من Diagramme d'activité UML للرسم البياني BPMN باستخدام لغة التحويل ATL و تطوير أداة لتوضيح الرسم البياني.

الكلمات المفتاحية: التحويل، أداة ، رسم بياني, ATL, BPMN, Diagramme d'activité

Résumé :

Les méthodes actuelles pour l'analyse des systèmes d'entreprise, se basent sur la représentation des travaux pratiques sous forme de modèles de processus d'entreprise. Un standard pour représenter de tels modèles est la représentation BPMN. Cette dernière est destinée à mettre en collaboration les analystes du domaine. De ce fait, il est considéré comme une entrée pour les projets de développement de logiciel. En même temps, les méthodes pour les systèmes orientés processus comptent sur les définitions détaillées des processus qui sont exécutés par des moteurs de processus.

Ces définitions de processus raffinent les modèles BPMN par le rajout des détails d'implémentation.

Dans notre travail nous proposons une technique automatique basée sur la transformation de Diagramme d'activité d'UML vers un graphe BPMN en utilisons le langage de transformation ATL et en développant un outil pour illustrer le graphe.

Mot clé : Diagramme d'activité d'UML, ATL, BPMN, graphe.

Abstract:

The current methods for the analysis of enterprise systems are based on the representation of practical work in the form of business process models. A standard to represent such models is the BPMN. The latter is intended to bring together analysts in the field. Because of this, it is considered as an entry for software development projects. At the same time, methods for process-oriented systems rely on detailed definitions of processes that are executed by process engines.

These process definitions refine BPMN models with the addition of implementation details.

In our work, we propose an automatic technic based on the transformation of UML activity diagram to a BPMN graph using the ATL transformation language and develope a tool to illustrate the graph.

Key word: BPMN, activity diagram, ATL.

Introduction générale

Avec l'évolution technologique et le flux d'informations de plus en plus difficile à maitriser, la dématérialisation de la gestion des processus métier devient une approche ingénieuse qui gouverne le changement pour engendrer l'optimisation de l'activité.

Le concept de la gestion des processus métier au sein de l'entreprise vise la standardisation des actions afin de pouvoir orchestrer des appels de systèmes externes de façon cohérente et ainsi fournir des services à valeur ajoutée dans un contexte en perpétuelle évolution le plus rapidement possible.

Le langage UML permet de modéliser des processus métier en utilisant l'un de ses diagrammes (diagrammes d'Activité), cependant UML est un langage semi formel qui n'a pas des constructions définis rigoureusement.

Pendant très longtemps on a voulu contraindre les MOA (Maitrise d'ouvrage) à apprendre UML avec des concepts techniques (classes, objets, ...) sans se soucier du vocabulaire des experts métiers. Ceux-ci ont alors refusé d'utiliser UML en disant que c'est trop compliqué.

L'Ingénierie Dirigée par les Modèles (IDM) ou Model Driven Engineering (MDE) est devenue largement utilisée pour le développement des applications complexes distribuées. Par l'IDM, nous faisons référence à une approche de développement mettant à disposition un ensemble d'outils et de langages pour la manipulation des modèles tel que l'UML.

Ces modèle sont une manière intuitive et naturelle permettant de représenter l'état et le comportement d'un système, quel que soit son degré de complexité et à chaque étape de son cycle de vie.

La norme BPMN (Business Process Modeling Notation) a été conçue spécialement pour eux en répondant à leurs besoins. Pas besoin de faire des stéréotypes, tout ce dont l'expert métier à besoin se trouve déjà dans BPMN sous forme de symboles comme par exemple les différents types d'évènements (date, message, ...). Donc BPMN est plus convivial.

L'objectif visé dans ce mémoire est de proposer une démarche MDA pour la composition des services web en se basant sur une transformation des diagrammes d'activité d'UML en BPMN. La composition des services web est modélisée par le diagramme d'activité et le passage des diagrammes d'activité vers BPMN permet l'exploitation des services web composites.

Ce mémoire est organisé en six chapitres. Nous exposons ci-dessous le contenu de ces chapitres.

Chapitre 1 : « Les services web »

Dans ce chapitre, nous introduisons des définitions générales sur la concept de « Service web» et sa notion. Nous y présentons leur origine, l'architecture ainsi que les langages et les protocoles utilisés pour leur déploiement.

Chapitre 2 : « Présentation du langage BPMN »

Ce chapitre est consacré à la présentation du langage BPMN une définition ainsi que le rôle de ce langage au niveau de la composition des services web. Enfin nous présentons un aperçu sur la structure du fichier BPMN.

Chapitre 3 : « Approche dirigée par les modèles »

Dans ce chapitre, nous présentons les concepts de base de l'Ingénierie Dirigée par les Modèles (IDM) avec ses différentes variantes, nous nous intéressons en particulier à l'Architecture Dirigée par les Modèles (MDA). Enfin, nous présentons un aperçu sur les différentes approches de modélisation pour la composition des services web.

Chapitre 4 : « Présentation du diagramme d'activité »

Dans ce chapitre, nous allons présenter le diagramme d'activité d'UML par une définition et la structure de diagramme.

Chapitre 5 : « Conception »

Dans ce chapitre nous allons proposer une approche MDA basée sur la méta-modélisation et les règles de transformation.

Chapitre 6 : « Implémentation et étude de cas »

Ce chapitre est consacré à la mise en œuvre de notre approche. Nous proposons également une étude de cas pour permettre de bien illustrer cette approche.

Enfin nous terminons par une conclusion générale qui donne un aperçu du travail que nous avons accompli. Nous y discutons également de quelques perspectives pour notre travail.

Chapitre 1

Les services web

I. Introduction :

Au début de l'année 2000, le paradigme SOA (Service Oriented Architecture) a vu le jour dans le but de résoudre les problématiques d'interopérabilité entre les différentes technologies informatiques distribuées utilisées en entreprise. Ce paradigme est l'un des modèles les plus prometteurs, en effet il adopte les avantages des autres paradigmes (orienté objet et orienté composant), tels que la modularité et l'encapsulation, et ajoute d'autres concepts, à savoir le couplage faible et l'interopérabilité.

En plus du respect des principes du SOA, les services web présentent plusieurs avantages tels que l'adaptation aux caractéristiques du web, et le bénéfice d'un large support de la part des industriels.

Dans ce chapitre, nous allons présenter quelques concepts de base qui entourent la technologie des services web.

II. Services web

1. Définition

Depuis les dernières années, l'utilisation de Services web a connu une popularité grandissante. Ces services sont très utilisés notamment par les entreprises pour rendre accessible leurs métiers ou leurs données via le web. Les services web ont été proposés initialement par IBM (International Business Machines Corporation) et Microsoft, puis en partie standardisés par le W3C (Consortium du World Wide Web) [1].

Plusieurs définitions des services web ont été proposées dans la littérature.

Selon IBM :

Les services web sont la nouvelle vague des applications web. Ce sont des applications modulaires, auto-contenues et auto-descriptives qui peuvent être publiées, localisées et invoquées depuis le web. Les services web effectuent des actions allant de simples requêtes à des processus métiers complexes. Une fois qu'un service web est déployé, d'autres applications (y compris des services web) peuvent le découvrir et l'invoquer [W1].

Selon le W3C :

Un service web est un système logiciel identifié par une URI (Uniform Resource Identifier) et conçu pour supporter l'interaction interopérable de machine à machine sur un réseau. Il possède une interface décrite dans un format exploitable par la machine, c.à.d. décrite en WSDL (Web Services Description Language).

D'autres systèmes interagissent avec le service web d'une façon prescrite par sa description en utilisant des messages SOAP (Simple Object Access Protocol), typiquement en utilisant HTTP (HyperText Transfer Protocol) avec une sérialisation XML (Extensible Markup Language) en même temps que d'autres normes du Web [W2].

2. Architecture des services web :

Dans cette section, nous allons présenter l'architecture de base des services web tel que proposée par IBM [W1]. Cette architecture comporte trois entités : le fournisseur de service, l'annuaire de services et l'utilisateur du service « client » (figure 1.1).

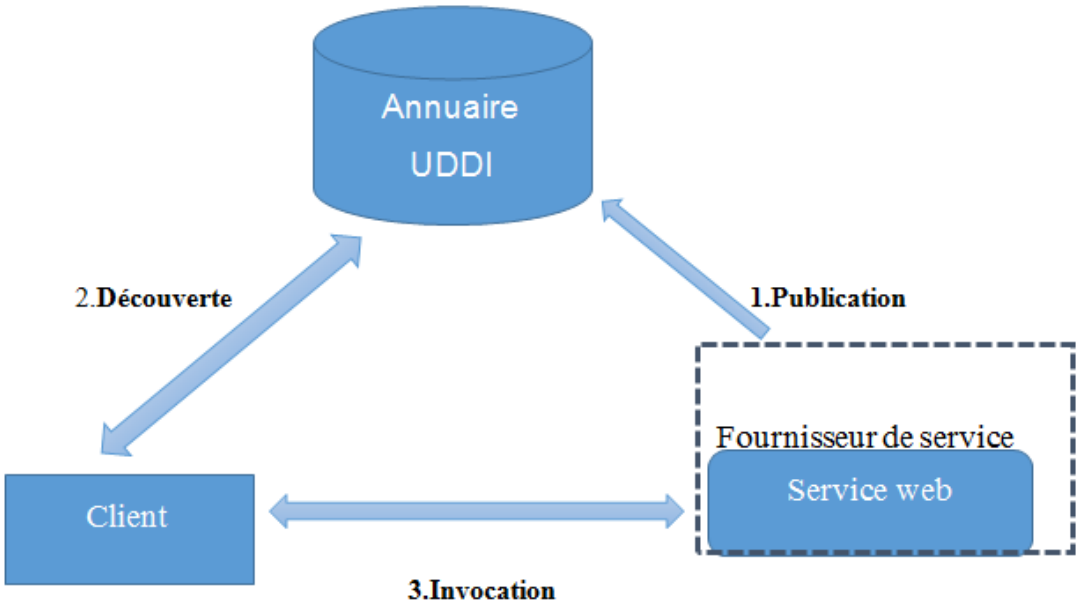


Figure1.1 l'architecture de base des services web [1].

- **Le Fournisseur du service :** qui publie son service en fournissant la description au format WSDL dans l'annuaire de services et réalise les opérations propres au service. C'est donc le propriétaire du service web [1]. Il représente l'environnement d'hébergement et d'exécution du service. Il est constitué de trois couches de bases : [1] :
  - ✓ ***La couche de données*** **:** contient les différentes bases de données utilisées par le service.
  - ✓ ***La couche applicative*** **:** c'est la plateforme de développement qui assure l'exécution du service web.
  - ✓ ***La couche de description*** : elle expose les fonctionnalités du service via un fichier WSDL.

- **Client** : qui obtient la description du service grâce à l'annuaire et utilise le service [1].

- **Annuaire :** L'annuaire de service est un annuaire qui contient les services disponibles [2].L'annuaire contient des publications des services web et permet à ses clients de les localiser et de découvrir leurs détails [1].

3. Protocoles utilisés par les services web :

Les services web utilisent de nombreux langages et protocoles durant le déploiement et l'invocation des services web. L'atout principal des protocoles SOAP (Simple Object Access Protocol) et UDDI (Universal Description Discovery and Integration) ainsi que du langage WSDL est de se reposer sur le langage XML.

Les services web se basent sur l'utilisation des normes actuelles d'Internet comme le protocole HTTP.

Dans cette section, nous allons présenter les déférentes couches horizontales qui décrivent la pile de protocoles et de modèles des services web.

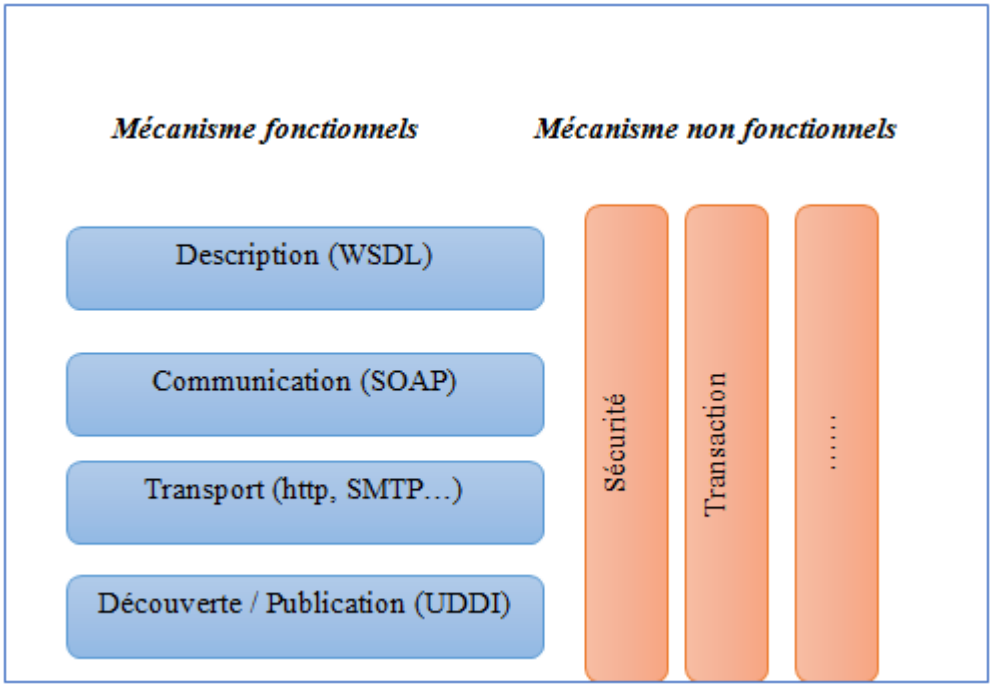


Figure 1.2 La pile Protocolaire simplifiée des services web [1].

3.1 Couche de Transport :

Cette couche s'intéresse aux protocoles de transport de bas niveau, ces derniers vont transporter les requêtes et les réponses échangées entre services. Le protocole le plus utilisé est le HTTP [1].

3.2 Couche de Communication (SOAP) :

Cette couche spécifie les protocoles d'échanges de documents XML entre le service web et ses clients. Le protocole SOAP est adopté comme un standard pour la messagerie entre les services web.

3.4 Couche de Description (WSDL) :

WSDL est un standard du W3C qui permet de définir la structure abstraite et c&&oncrète d'un service.

Un document WSDL se compose d'un ensemble d'éléments décrivant les types de données utilisés par le service, les messages que le service peut recevoir, ainsi que les liaisons SOAP associées à chaque message [1].

3.5 Couche de Découverte et de Publication (UDDI) :

UDDI est une spécification d'annuaires de services web, cette norme W3C propose un ensemble de structures à publier par les fournisseurs de services. Ces structures sont formalisées en XML [1].

III. Composition de services Web

La composition de services web peut être vue comme étant un moyen efficace pour créer, exécuter, et maintenir des services qui dépendent d'autres.

1. Définition :

Une composition de services Web est constituée de plusieurs services qui interagissent les uns avec les autres, afin d'offrir de nouvelles fonctionnalités qu'un seul service ne pourrait pas les offrir [3].

La composition permet de combiner des services pour former un nouveau service dit composé ou composite. L'exécution d'un service composé implique des interactions avec des services partenaires en faisant appel à leurs fonctionnalités.

Le but de la composition est avant tout la réutilisation de services (simples ou composés) et de préférence sans aucune modification de ces derniers [4].

Une autre définition plus raffinée est celle proposée par *Georges Gardarin,* qui définit la composition des services web comme étant : « une technique permettant d'assembler des services Web afin d'atteindre un objectif particulier, par l'intermédiaire de primitives de contrôle (boucle, test, traitement d'exception, etc.) et d'échange (envoi et réception de messages) [5].

2. Description et fonctionnement :

Un service web est dit composé ou composite lorsque son exécution implique des interactions avec d'autres services web afin de faire appel à leurs fonctionnalités. La composition de services web spécifie quels services ont besoin d'être invoqués, dans quel ordre et comment gérer les conditions d'exception. La figure suivante montre le comportement d'un service composite (figure 1.3).

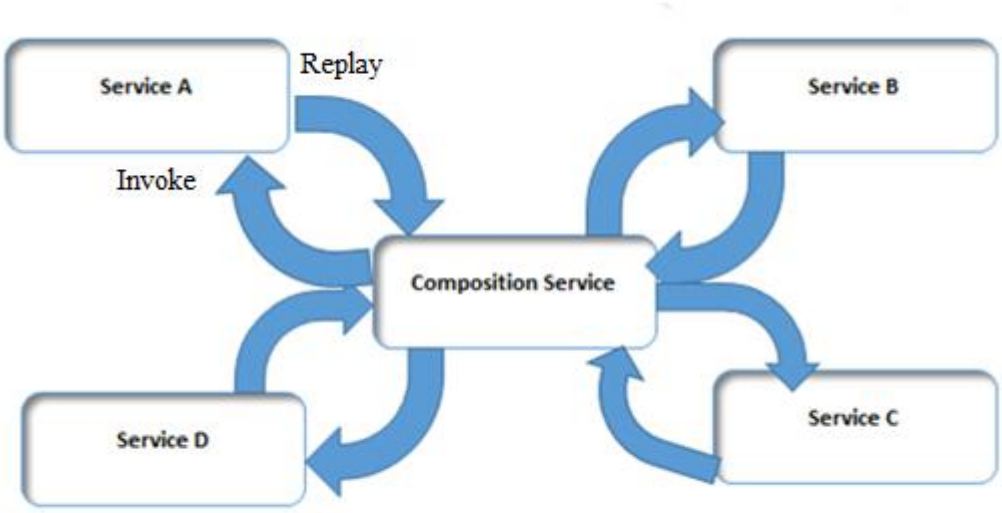


Figure 1.3 Comportement d'un service composite [4].

La composition des services web est classifiée en deux catégories dynamique et statique dont cette dernière peut se faire de deux manières : ***orchestration*** et ***chorégraphie.***

A. Orchestration :

L'orchestration décrit l'interaction des services au niveau de messages, incluant la logique métier et l'ordre d'exécution des interactions.

Les services web n'ont pas de connaissance (et n'ont pas besoin de l'avoir) d´être mêlées dans une composition et d'être partie d'un processus métier. Seulement le coordinateur de l'orchestration a besoin de cette connaissance [3].

La figure suivante montre le workflow dans l'orchestration des services web (figure 1.4). Un coordinateur prend le control de tous les services web impliqués et coordonne l'exécution des différentes opérations des services web qui participent dans le processus.

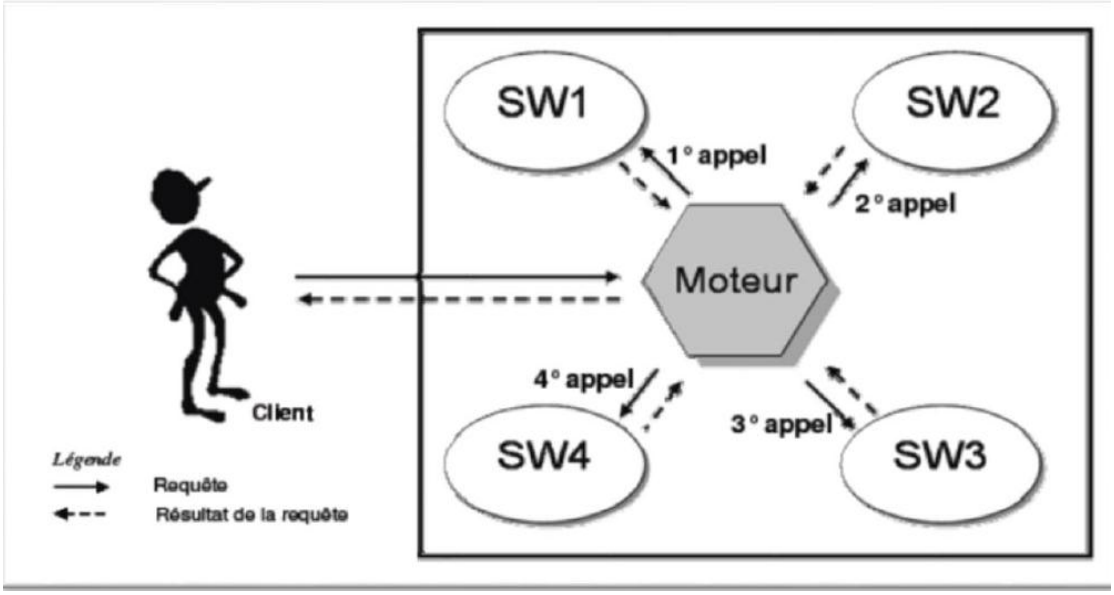


Figure 1.4 Vue générale de l'orchestration [4].

Dans ce type, seul le coordinateur (le chef d'orchestre) connaît la façon d'exécution de ses services en parallèle ou séquentiel.

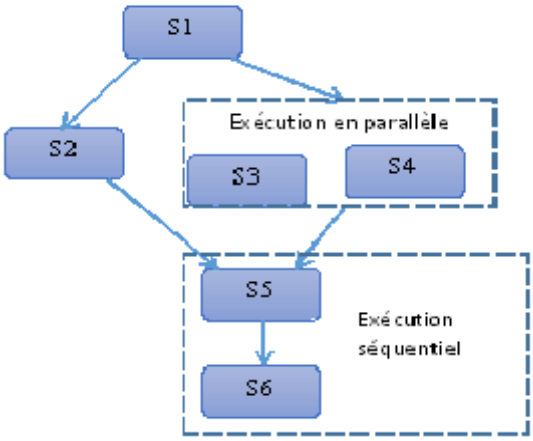


Figure 1.5 Exemple de scénario d'un service composite

B. Chorégraphie

Contrairement à l'orchestration, la chorégraphie n'a pas un coordinateur central.

Chaque service web mêlée dans la chorégraphie connaît exactement quand ses opérations doivent être exécutées et avec qui l'interaction doit avoir lieu.

La chorégraphie est un effort de collaboration dans lequel chaque participant du processus décrit l'itération qui l'appartient. Elle trace la séquence des messages qui peut impliquer plusieurs Services Web [3].

La collaboration dans la chorégraphie des services web peut être représentée de la manière suivant (figure 1.5).

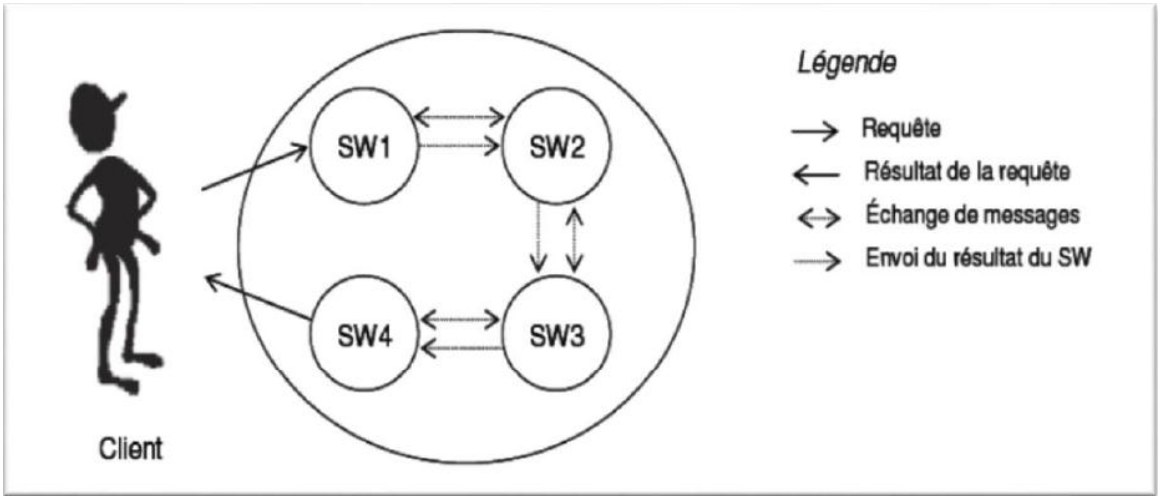


Figure 1.6 Vue générale de la chorégraphie [4].

3. Types de composition

Nous avons déjà expliqué qu'est-ce qu'un service Web et comment fonctionnent les services web composites. Mais, il nous faudra encore comprendre comment cette composition peut s'accomplir depuis la perspective du dégrée d'automatisation. La composition peut être classifiée dans trois catégories [3] :

- **Composition manuelle**

La composition manuelle des services Web suppose que l'utilisateur génère la composition à la main via un éditeur de texte et sans l'aide d'outils dédiés [3].

- **Composition semi-automatique**

Les techniques de composition semi-automatiques sont un pas en avant en comparaison avec la composition manuelle, dans le sens qu'ils font des suggestions sémantiques pour aider à la sélection des services Web dans le processus de composition [3].

- **Composition automatique**

La composition totalement automatisée prend en charge tout le processus de composition et le réalise automatiquement, sans qu'aucune intervention de l'utilisateur ne soit requise [3].

4. Langages utilisés pour la composition des services web

Ils existent plusieurs langages de définition pour la composition de services web.

Dans cette section nous allons présenter brièvement quelques-uns d'entre eux, notamment les plus connus et acceptés afin de faire le choix le plus convenant pour notre projet [3].

Dans cette section, nous allons présenter des langages d'orchestration et de chorégraphie de services web, que nous avons introduit précédemment. Ces langages sont :

- **Langage XLANG:** (Web Services for Business Process Design). Créé par Microsoft, le XLANG est une extension de WSDL. Elle fournit en même temps un modèle pour une orchestration des services et des contrats de collaboration entre celles-ci. XLANG a été conçus avec une base explicite de théorie de calcul [5].

Langage BPEL : (Business Process Exécution Language) représente un langage de programmation standard aidant les entreprises à définir la manière de combiner des services web.

BPEL est un langage de modélisation des processus métiers. Il permet de définir un modèle abstrait d'interaction entre collaborateurs participant à une activité de l'entreprise, voire entre une organisation et ses partenaires [5].

- **Langage WSFL : (***Web Service Flow Langage***)** C'est une approche IBM qui permet d'intégrer la composition de services Web comme un processus métier statique. C'est un langage basé XML permettant de décrire des orchestrations des services Web [5].

- **Langage WSCL :** [4] (Web Services Conversation Lagunage) permet de décrire les conversations du niveau de la logique du métier ou les processus publics basé sur la définition d'un service web.

Les définitions de conversation en WSCL sont elles-mêmes des documents XML et peuvent donc être interprétées par des infrastructures de services web et des outils de développement.

WSCL a été proposé par le consortium W3C, il permet de définir les interfaces abstraites des services Web.

- **Langage WSCI :** (Services Web Chorégraphie Interface) est un langage de description basé sur XML dont l'objectif est de décrire les messages échangés entre le service web participant à des interactions de chorégraphie et les autres services de cette chorégraphie [4].

WSCI propose d'exprimer les dépendances logiques et temporelles entre les messages échangés à l'aide de contrôles de séquences, corrélation, gestion de fautes et transactions [3].

- **Langage BPMN :** (*Business Process Model and Notation* (BPMN en anglais), est une notation graphique développée par BPMI (Business Process Management Initiative) en Mai 2004. Elle a été reconnue comme norme par l'OMG (Object Management Groupe) en 2005 [6].

Dans notre projet on s'intéresse à ce langage de modélisation (BPMN).

IV. Conclusion

Dans ce chapitre, nous avons présenté un état de l'art sur les services web, leur composition, sa découverte et les langages de modélisation les plus connus.

Dans le chapitre suivant, Nous détaillions le langage de modélisation de processus métier BPMN, dont nous allons utiliser un ensemble d'éléments qu'on a besoin dans notre démarche.

Chapitre 2

Présentation du langage BPMN

I. Introduction

Un processus d'entreprise est considéré comme une suite d'étapes. Une ou plusieurs personnes se partagent des tâches en vue d'atteindre un certain objectif.

De nos jours, beaucoup de sociétés esquissent des représentations graphiques pour leur processus d'entreprise grâce à des techniques de dessin que les programmateurs ont presque toujours utilisé pour illustrer le fonctionnement de leurs programmes.

La BPMN (*Business Process Modeling Notation*), une nouvelle norme, fait son apparition en proposant un ensemble d'éléments graphiques normalisés et une grammaire permettant leur manipulation. La BPMN doit à la fois être simple et capable de gérer les complexités inhérentes au processus métier, elle doit aussi être facilement traduite en langage exécutable.

Le but de ce chapitre est d'introduire les objets graphiques de la BPMN (BPD *Business Process Diagrams*), en se basant sur le document de travail de l'OMG (*Object Management Group*) : « Business Process Modeling Notation » du 03 janvier 2009, Avec une manière souple allant des objets les plus simples (les plus utilisés aussi) aux objets plus complexes.

Dans notre projet, on a utilisé des simples objets pour la modélisation des services web composites.

II. Processus métier

1. Définition

Un processus métier ou "Business Process" en anglais est défini par le Workflow Management Coalition (WfMC) comme : "un ensemble de procédures ou d'activités liées les unes aux autres pour atteindre collectivement un objectif métier en définissant les rôles et les interactions fonctionnelles au sein d'une structure organisationnelle" [5].

III. Le BPM et BPMN

1. Historique

À l'origine BPMN est un projet initié par Business Process Management Initiative (BPMI) qui fusionna en 2005 avec l'Object Management Group (OMG) [9].

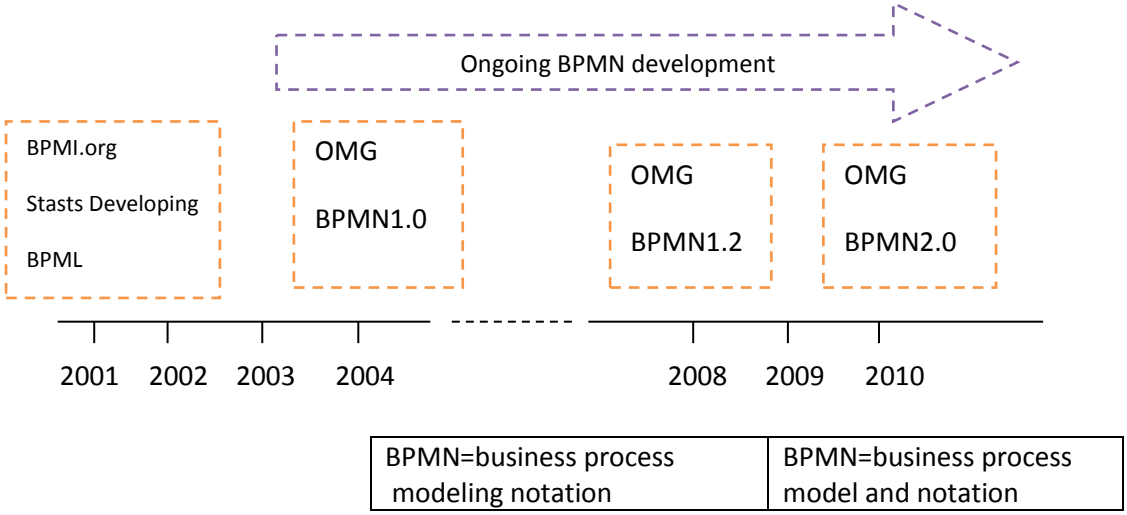


Figure 2.1 Axe de développement de BPMN [10]**.**

2. Définition

BPM (Business Process Management) est la discipline qui consiste à considérer la gestion des processus comme un moyen d'améliorer la performance opérationnelle.

BPMN (Business Process Model and Notation) est l'ensemble de convention graphique permettant de représenter les processus métier sous forme de modèle. BPMS (Business Process Management Suite) est une plate-forme logicielle permettant d'outiller la démarche BPM depuis la découverte du processus jusqu'à son optimisation via sa définition, son automatisation et analyse.

BPMN est un langage récent conçu pour être au cœur d'une approche de modélisation et d'implémentation utilisant les cadres conceptuels de l'ingénierie dirigée par les modèles (IDM), de l'architecture orientée services (SOA) et de la gestion des processus métier [10].

3. Composants d'un diagramme BPMN

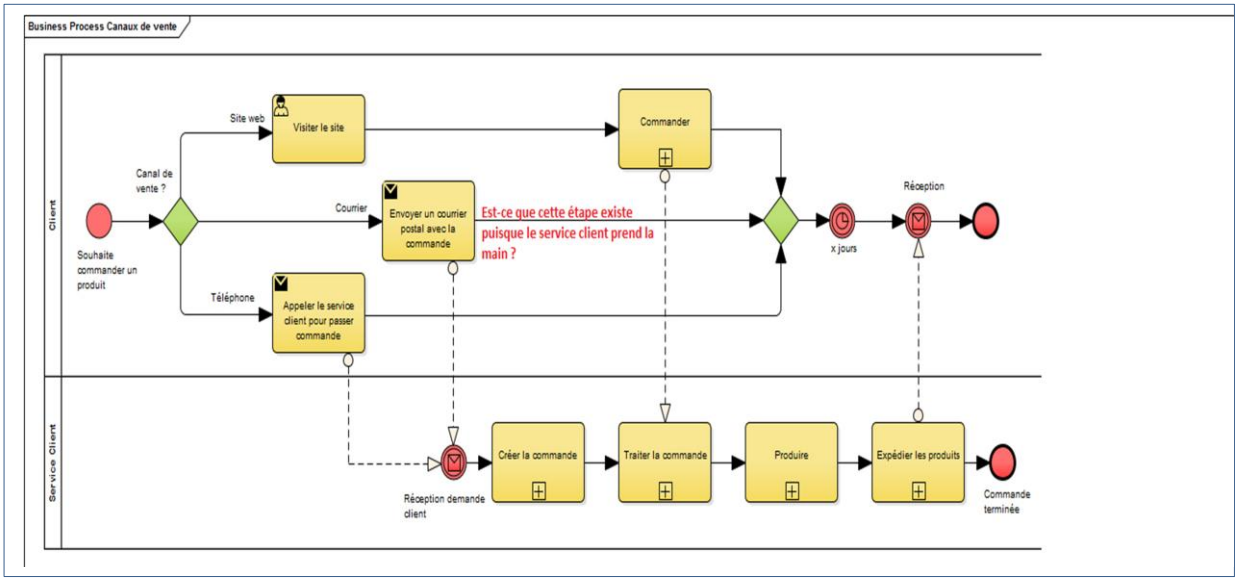


Figure 2.2 Processus de vente aux enchères modélisées par la notation BPMN [W3].

Un diagramme BPMN est composé d'un ensemble d'éléments graphiques qui permettent de modéliser les activités, les flux et les relations.

- Les objets de flux :

a) ***Activité*** : Réalisation d'une action par un utilisateur ou un système informatique. Les activités peuvent être des processus (Process), des sous-processus (Sub-Process) ou des tâches (Task) [11].

Symbole	Nom	Description
	Tâche	Une tâche est un élément indivisible. Elle représente une action.
	sous processus	Un sous processus est une tâche qui est décomposée.
	Activité d'appel	Une activité d'appel est une tâche utilisée pour appeler un deuxième processus lors de l'exécution du processus en cours.

Tableau 2.1 les composantes des activités [11].

b) **Evènement** : Utilisée pour débuter ou finir un processus et gérer des actions spécifiques au cours de celui-là [11].

Les événements peuvent être de trois types différents : *Début* (Start), *intermédiaire* (Intermediate), *fin* (End) et pour l'évènement message dont le déclanchement est représenté par une enveloppe vide (message reçu) et le résultat par une enveloppe plein (message d'envoi).

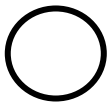
Nom d'Evènement	Symbole		
Aucun	Début	Intermédiaire	Fin
			
Message			 

Tableau 2.2 Les évènement [12].

c) **Porte décisionnelle :** Le branchement est un objet essentiel dans la norme BPMN. Il sert à représenter la condition de routage entre le(s) flux en entrée et le(s) flux en sortie.

c.1 Les branchements exclusifs en entrée / sortie

C.1.1 En entrée d'une étape, un branchement représente un point de convergence des enchaînements en entrée d'un processus. L'étape du processus attend qu'une branche entrante se termine avant de déclencher le flux de sortie.[11][5].

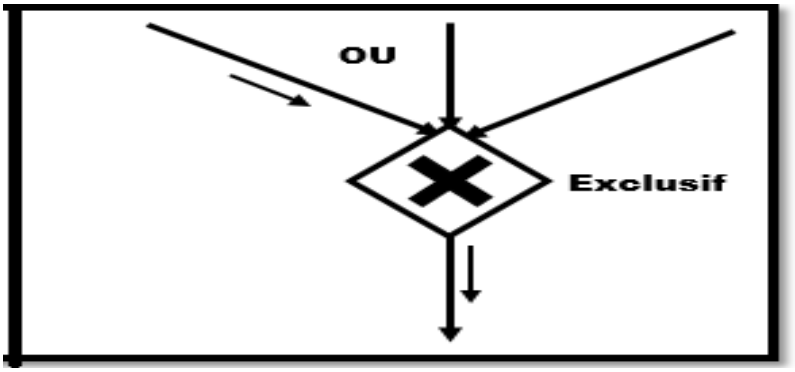


Figure 2.3 branchement exclusif en entrée.

C .1.2 En sortie d'une étape, un branchement représente un point de divergence des enchaînements en sortie d'un processus, seule une branche sortante peut être choisie parmi les différents possibles [11] [5].

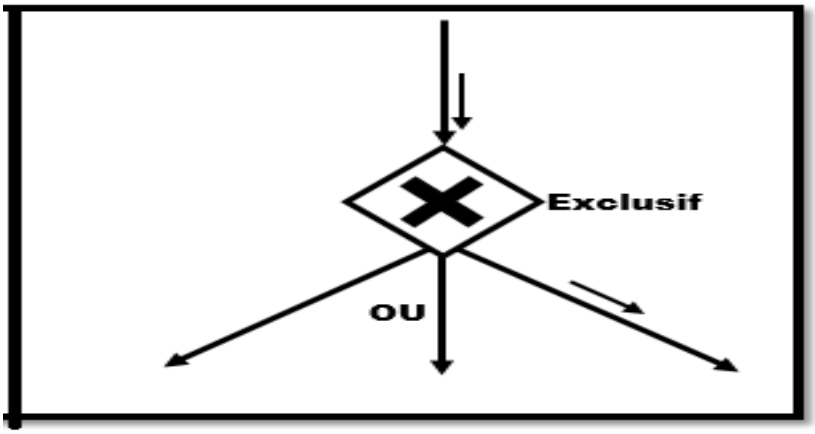


Figure 2.4 branchement exclusif en sortie.

***c.2* Les branchements parallèles en entrée et en sortie**

c.2.1 En entrée d'une étape, un branchement représente un point de convergence des enchaînements en entrée d'un processus. L'étape du processus attend que toutes les branches entrantes se terminent [11] [5].

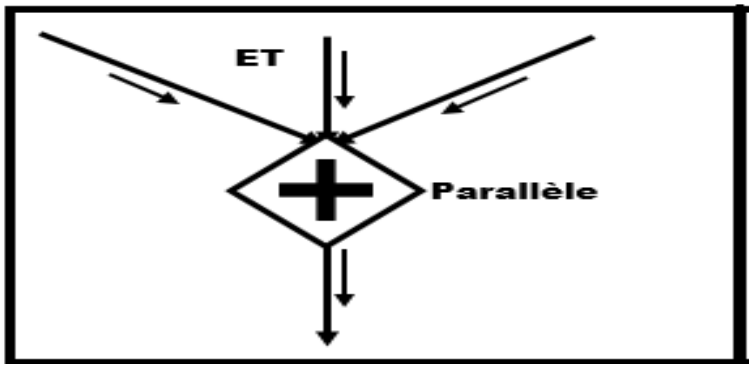


Figure 2.5 branchement parallèle en entré.

c.2.2 En sortie d'une étape, un branchement représente un point de divergence des enchaînements en sortie d'un processus, active toutes les branches sortantes simultanément [11] [5].

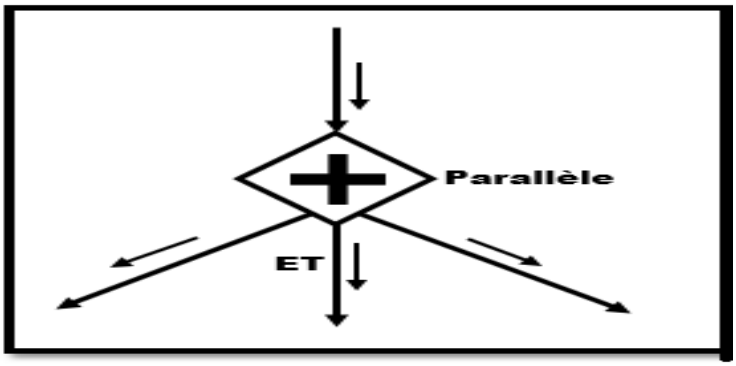


Figure 2.6 branchement parallèle en sortie.

d) Les lignes

Parmi les « lignes » il est possible de grouper de deux manières différentes les éléments du modèle :

- Piste (ou « Pool » en anglais) ;
- Ligne (ou « Lanes » en anglais).

e) Les connecteurs

Il y a trois connecteurs possibles pour relier les objets de flux entre eux ou avec d'autres informations, parmi les trois [13] :

- Flux de séquence : Connecte les événements, les activités et les branchements dans des processus dans les diagrammes de collaboration, de processus et de chorégraphie.



Figure 2.7 Flux de séquence.

.

IV. Conclusion

Dans ce chapitre nous avons présenté les concepts de bases sur BPMN qui permet de créer un mécanisme simple pour modéliser la complexité inhérente à un processus. BPMN fournit une notation commune de sorte que les personnes liées à des processus peuvent les exprimer graphiquement d'une manière claire, standardisée et complète.

Le chapitre suivant est consacré à une présentation synthétique de l'IDM (Ingénierie Dirigée par les Modèles). Nous analysons l'existant dans ce domaine afin de déterminer dans quelle mesure nous pouvons intégrer la composition de modèles dans un cycle de développement dirigé par les modèles, et comment cette opération peut profiter pleinement des techniques et résultats obtenus dans le domaine de la transformation de modèle.

Chapitre 3

Approches dirigées par les modèles

I. Introduction

Dernièrement, l'Ingénierie Dirigée par les Modèles (IDM, ou MDE : Model Driven Engineering) est devenue largement utilisée pour le développement des applications complexes distribuées. Par l'IDM, nous faisons référence à une approche de développement mettant à disposition un ensemble d'outils et de langages pour la manipulation des modèles.

Ces derniers permettant de représenter l'état et le comportement d'un système, quel que soit son degré de complexité et à chaque étape de son cycle de vie.

L'IDM exploite les modèles en les détaillant en fonction du besoin, jusqu'à obtention d'un ou plusieurs squelettes de codes générés de manière systématique. L'intérêt principal de cette approche réside dans le fait qu'elle permet d'élever la productivité du développement en séparant la « logique métier » des plateformes utilisées.

Dans ce chapitre, nous allons présenter les concepts de base de l'Ingénierie Dirigée par les Modèles qui couvre différentes disciplines dans lesquelles les modèles jouent un rôle principal. Nous nous intéresserons également à l'Architecture Dirigée par les Modèles (MDA) et à son utilisation pour le développement des systèmes complexes.

1. L'Ingénierie dirigée par les modèles (IDM) :

Ce que propose l'approche de l'ingénierie des modèles (IDM, ou MDE en anglais pour Model Driven Engineering) est simplement de mécaniser le processus que les ingénieurs expérimentés suivent à la main. L'intérêt pour l'IDM a été fortement amplifié à la fin du XX^{e} siècle lorsque l'organisme de standardisation OMG (Object Management Group) a rendu publique son initiative MDA (Model Driven Architecture).

IDM est une pratique d'ingénierie des systèmes utilisant les capacités des technologies informatiques pour décrire au travers de modèles, concepts, et langages, à la fois le problème posé (besoin) et sa solution [1].

2. Concepts fondamentaux de l'ingénierie dirigée par les modèles :

Dans cette section, nous nous proposons de définir les concepts de base de l'Ingénierie Dirigée par les Modèles afin d'appréhender de manière optimale cette approche :

A. Modèle :

Le modèle est une représentation simplifiée du système. Il permet de spécifier formellement les fonctions, la structure et/ou le comportement de ce système, dans un certain but. Un modèle est souvent représenté par des schémas et par des textes. Ces derniers peuvent être exprimés dans un langage de modélisation ou en langage naturel [1].

Un modèle doit avoir les propriétés suivantes [1]:

- ✓ ***Abstrait :*** Un modèle doit permettre de cacher certains détails inutiles à un moment donné de la conception.
- ✓ ***Compréhensible :*** Un modèle doit être facile à assimiler et à manipuler par tous les acteurs entrant dans l'élaboration du système.
- ✓ ***Fidèle et précis :*** Un modèle doit représenter fidèlement les propriétés et les caractéristiques du système.
- ✓ ***Prédictif :*** Un modèle doit fournir assez d'informations pour permettre de prédire les propriétés du système.
- ✓ ***Economique :*** Les coûts de construction et de tests sur le modèle doivent être limités et ne doivent pas dépasser les coûts de construction d'un prototype du système.

B. Méta-modèles et méta-modélisation :

L'ingénierie dirigée par les modèles préconise l'utilisation d'un mécanisme standard et abstrait pour définir des modèles. Ce mécanisme abstrait est dénoté par le terme méta-modèle [1]. Un méta-modèle est encore une autre abstraction soulignant les propriétés du modèle lui-même. Un modèle est dit "conforme à" son méta-modèle comme un programme conforme à la grammaire du langage de programmation selon lequel il est écrit. Cela signifie qu'un méta-modèle décrit les différents types d'éléments de modèle et la façon dont ils sont disposés, reliés et contraints.

Les méta-modèles nous mènent à la notion de méta-modélisation qui consiste à créer un méta-modèle afin de simplifier et ainsi de pouvoir exploiter plus facilement le modèle initial. En effet, avec l'apparition de l'IDM, l'espace des modèles devient de plus en plus large et occupe toutes les dimensions de l'application (métier, technique, niveau d'abstraction, etc.).

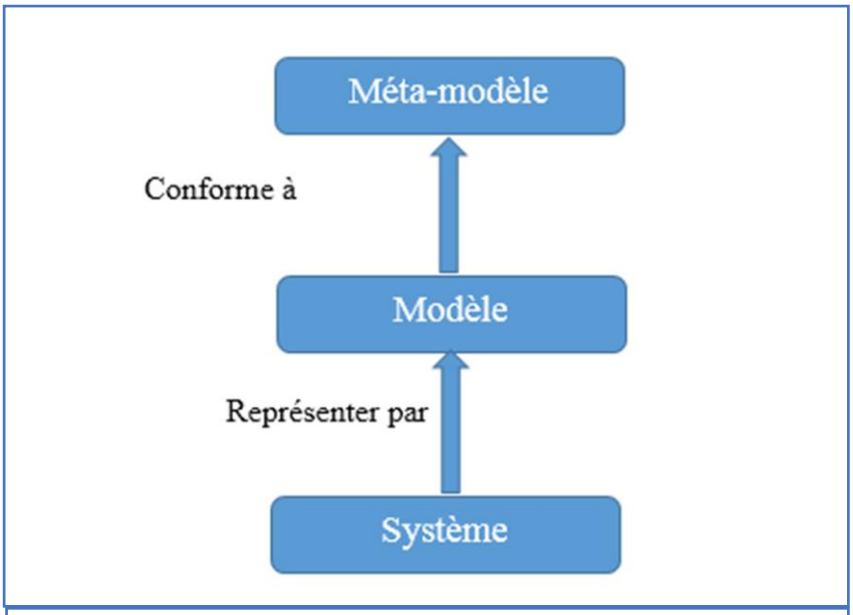


Figure 3.1 Notion de base de l'ingénierie des modèles [7].

C. La transformation de modèles :

La transformation de modèle est une opération fondamentale dans toute approche orientée modèle. La définition et l'automatisation des transformations a pour objectif de rendre les modèles plus opérationnels et d'augmenter la productivité du développement dans une approche IDM.

C.1 Le principe général de transformation des modèles :

Une transformation de modèle se définit par l'opération de génération d'un ou plusieurs modèles cible à partir d'un ou de plusieurs modèles source, conformément à une définition de transformation. Conscient de l'importance des transformations dans l'approche MDA, l'OMG a lancé en 2002 un appel à proposition industriel RFP MOF/QVT dont le but était de définir un langage unifié de transformation. L'idée est de modéliser la transformation elle-même en lui appliquant les principes de l'IDM, et donc de la considérer comme une application définie par un modèle de transformation (Mt) lui-même conforme à un méta-modèle (MMt), ce dernier représentant la définition abstraite du langage de transformation [6].

C.2 Les types de transformation des modèles :

- **Transformation endogène (raffinement) :**
  - ✓ Un modèle M1 devient un modèle M2 ;
  - ✓ Même méta-modèle MM pour les deux ;
  - ✓ Règle de transformation T.

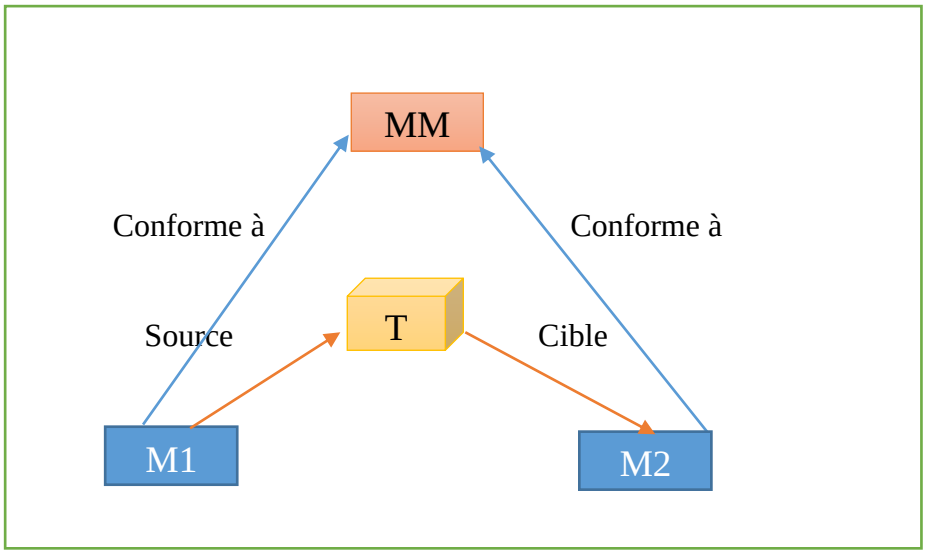


Figure 3.2 Transformation endogène [7].

- **Transformation exogène (projection) :**
  - ✓ Un modèle M1 devient un modèle M2
  - ✓ Méta-modèles distincts, MM1, MM2.
  - ✓ Règle de transformation T.

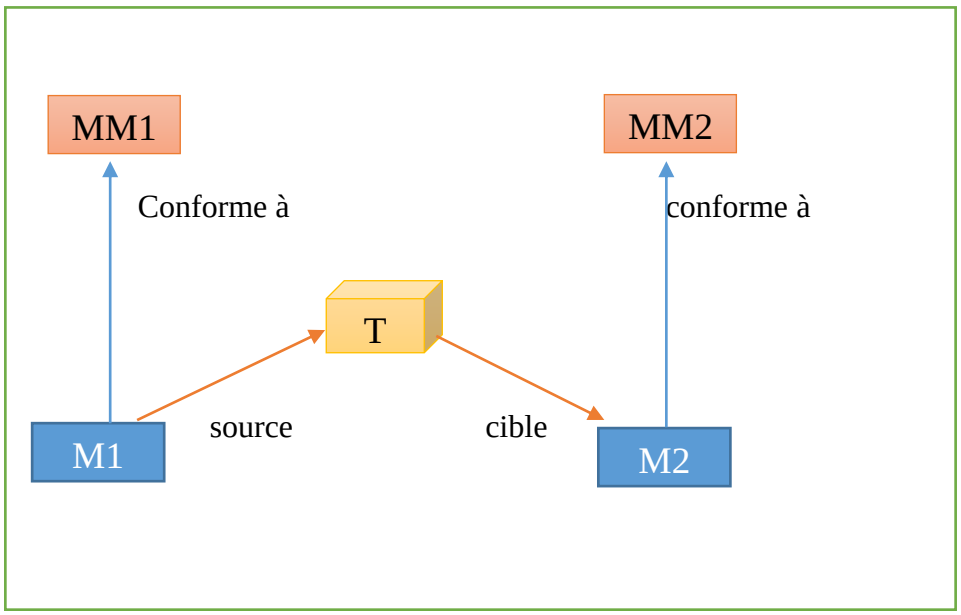


Figure 3.3 Transformation exogène [7].

C.3 Les paradigmes de transformation des modèles :

De nombreuses techniques sont utilisées pour élaborer des transformations de modèle, deux grandes familles d'approches sont répertoriées : les approches de transformations « modèle vers modèle », et les approches de transformations « modèle vers texte ».

C.3.1. Approches Modèle vers Modèle (M2M : Model-to-Model)

✓ **Approche de manipulation directe :** utilisent en général un framework orienté objet comme JMI (Java Metadata Interface) ou EMF (Eclipse Modeling Framework). Ces API (Application Programming Interface) sont souvent couplées avec un langage de programmation impératif (tel que Java). Cette approche est la plus utilisée, car elle est bien supportée par les outils de développement [6] [1].

✓ **Approche relationnelle** : Cette approche utilise les contraintes pour spécifier les relations entre les éléments du modèle source et ceux du modèle cible en utilisant une logique déclarative basée sur des relations mathématiques [1].

✓ **Approche basée sur les transformations de graphes :** s'appuient sur les notions théoriques du domaine de la réécriture de graphes. Elle exprime les transformations sous une forme déclarative. Les règles de transformation sont définies sur des parties de modèle et non pas sur des éléments basiques [6].

✓ **Approche dirigée par la structure** : est caractérisé par un processus de transformation de modèle composé de deux étapes. La première étape permet de créer la structure hiérarchique du modèle cible (PSM) [1], alors que durant la seconde étape, les attributs et les références sont mis en place afin de compléter le modèle produit. Dans cette approche, l'environnement de transformation définit l'ordonnancement et la stratégie d'application des règles [6].

Enfin, les approches hybrides combinent les différentes techniques décrites précédemment. Le style d'écriture des règles compose des structures déclaratives et impératives. L'enchaînement des règles est spécifié soit explicitement soit implicitement. ATL est un exemple de cette approche [6].

C.3.2. Approches Modèle vers Code

La génération de code peut être considérée comme un cas particulier de transformation de type modèle vers texte. Pour cela, il faut définir un méta-modèle correspondant au langage de programmation cible. Pour des raisons pratiques liées à la réutilisation des compilateurs existants, le code produit est souvent donné sous format texte [6]. Ceci conduit à considérer deux types d'approche :

1) Le premier type, on range les approches basées sur un mécanisme de visiteur dont le principe repose sur le balayage de la représentation interne du modèle et l'écriture du code dans un fichier de sortie sous format texte comme exemple le Framework Jamda [6].

2) Le deuxième type rassemble les approches qui sont basées sur des canevas ou 'templates' de transformation des modèles vers du code. Dans cette catégorie on trouve plusieurs outils MDA comme exemple Acceleo [6].

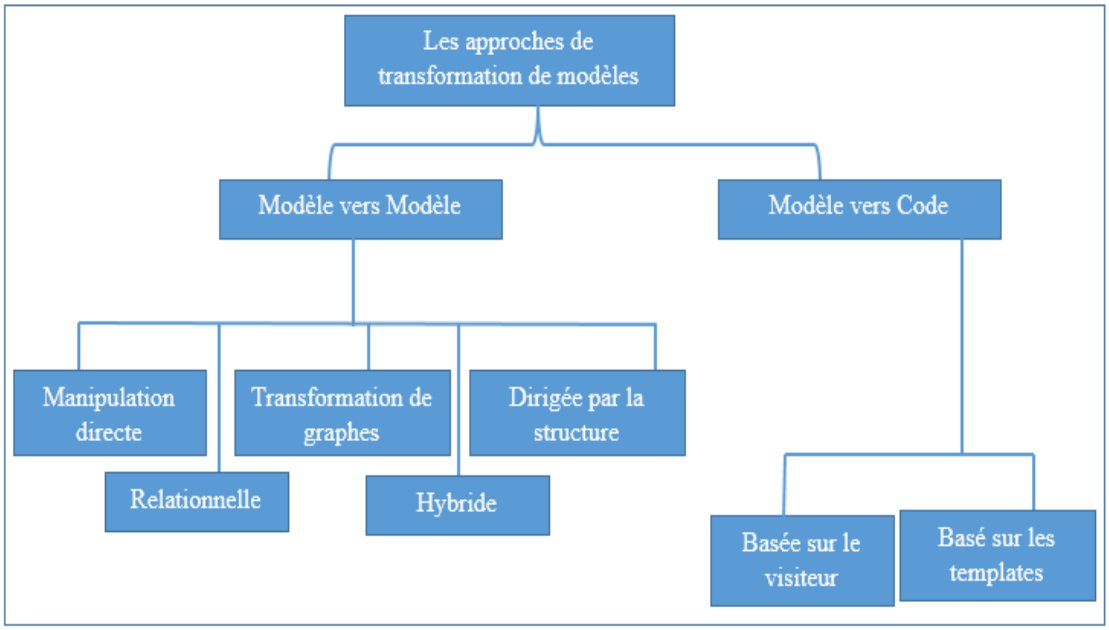


Figure 3.4 Les approches de transformation de modèles [1].

3. L'approche MDA (Model driven Architecture) [6]:

L'OMG (Object Management Group) a proposé l'approche Model Driven Architecture (MDA) en 2000 pour décrire séparément les spécifications fonctionnelles d'un système et son implémentation sur une plate-forme donnée. Partant de l'idée de séparation des préoccupations, MDA vise trois objectifs fondamentaux : la portabilité, l'interopérabilité et la réutilisabilité. MDA a fait l'objet d'un grand intérêt dans la littérature spécialisée.

L'approche MDA est basée sur la production de modèles indépendants des plates-formes (Platform Independent Model ou PIM) et la transformation de ces modèles en modèles spécifiques aux plates-formes (Platform Specific Model ou PSM). L'idée est d'automatiser la projection d'un PIM vers un PSM, pour qu'en cas de changement d'architecture technique, il suffise de régénérer un autre PSM à partir du même PIM. Cette séparation entre les spécifications fonctionnelles et techniques permet d'assurer la réutilisation des PIM, leur portabilité, et aussi l'interopérabilité.

3.1. Types de modèles dans MDA :

Dans un processus de développement orienté MDA, tout est considéré comme modèle. En partant de la phase de spécification des exigences jusqu'à la phase d'implémentation, les modèles sont utilisés pour représenter les artefacts logiciels manipulés durant le processus de construction de l'application. Ainsi, MDA identifie quatre types de modèles : CIM (*Computation Independent Model*), PIM (*Platform Independent Model*), PDM (*Platform Description Platform*) et PSM (*Platform Specific Model*) (Figure 3.5) [6].

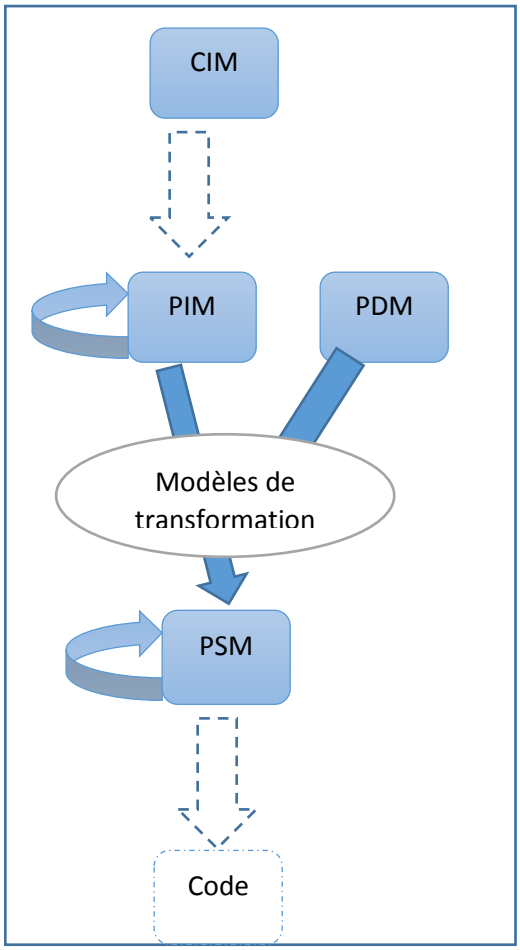


Figure 3.5 Les différents modèles dans MDA [1].

▪ **Le CIM** (*Computation Independent Model*) : La réalisation de ce modèle est la première étape dans le développement d'un système puisqu'elle va permettre de modéliser toutes les exigences du client et définir les différentes interactions qui impliqueront le système dans ses environnements, interne ou externe [1].

De ce fait, le CIM sera considéré comme référence pour vérifier la conformité du système avec les exigences du client [1].

Le CIM ne donne aucun détail sur la façon de la réalisation ou le fonctionnement du système mais exprime clairement des liens de traçabilité avec les modèles futurs.

- **Le PIM** (*Platform Independent Model*) **:** décrit le système sans montrer les détails de son utilisation sur une plate-forme particulière. Dans MDA, il est possible d'élaborer plusieurs modèles PIM indépendants de la plate-forme cible. Le PIM de *base* représente uniquement les capacités fonctionnelles métier et le comportement de l'application.

Les PIM suivants ajoutent des aspects technologiques et architecturaux mais toujours sans détails spécifiques d'une plate-forme. Un PIM doit être raffiné avec les détails d'une ou plusieurs architecture(s) particulière(s) pour produire un PSM [6].

- **Le PSM** (*Platform Specific Model*) est le modèle produit par la transformation d'un PIM pour prendre en compte la plate-forme d'exécution cible choisie. Les PSM servent principalement à faciliter la génération de code à partir d'un modèle d'analyse et de conception. Ces modèles contiennent les informations nécessaires à l'exploitation d'une plate-forme d'exécution [6].

- **Le PDM** : (*Platform Description Platform*) : 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 [1].

4. Niveaux d'abstraction du MDA

Depuis son apparition, MDA préconise l'utilisation du langage UML pour réaliser les modèles, mais ceci n'exclut pas que d'autres langages puissent être utilisés. Cette flexibilité de choix a eu comme conséquence l'apparition de nombreux formalismes de modélisation ou méta-modèle, chacun apportant sa spécificité pour un domaine particulier (développement logiciel, entrepôt de données, etc. Afin de limiter la divergence entre les formalismes de définition de ces méta-modèles, l'OMG a conçu le standard MOF (Meta Object Facility).

Le MOF est un formalisme de modélisation de méta-modèle, appelé *méta-méta-modèle*. Pour ne pas ajouter de niveaux d'abstraction supplémentaires, le MOF a la particularité de se définir lui-même. Ce principe est connu sous le nom de méta-circularité [6].

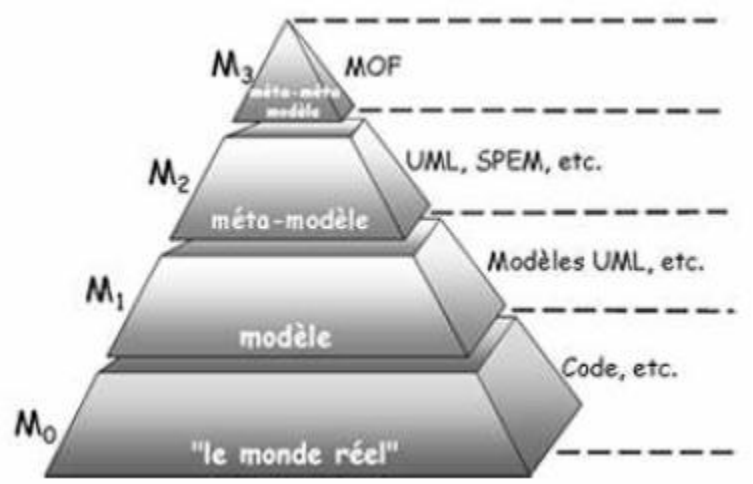


Figure 3.6 Pyramide de modélisation de l'OMG [6].

5. Approches dirigées par les modèles pour la composition des services :

Nous proposons une classification des approches dirigées par les modèles pour la composition de services basée sur le type de langage de modélisation utilisé. Cette classification est présentée sous forme d'une arborescence dans la Figure 3.7. Nous distinguons trois principales catégories de modèles : les systèmes concurrentiels, les systèmes de transition d'états et les processus métier (business process) [6] :

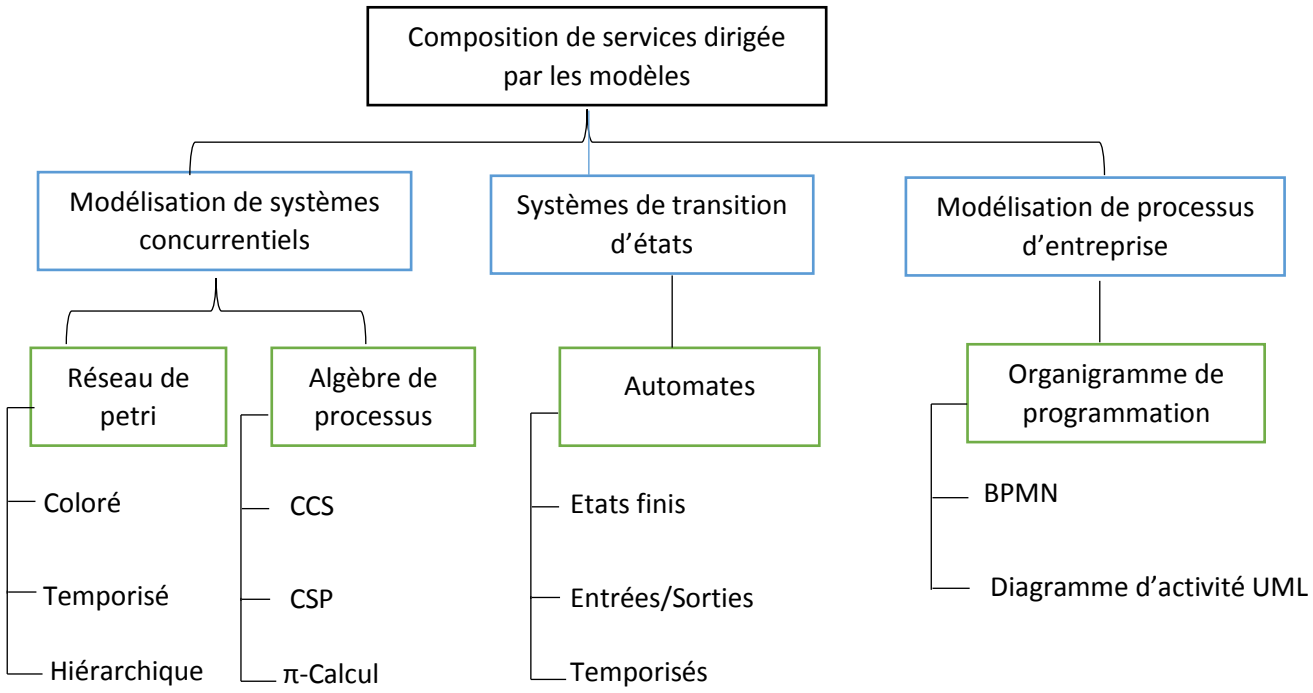


Figure 3.7 classification des approches dirigées par les modèles [6].

➢ **Les systèmes concurrentiels** : Parmi les langages de modélisation de ces systèmes, nous classons également :

- Les réseaux de Petri ainsi que leurs diverses extensions (couleur, temps, hiérarchie) sont très utilisés du fait de leur formalisme et de leur apparente simplicité.
- Les algèbres de processus telles que CCS, π- calculus ou CSP. Ces algèbres de processus sont particulièrement utiles pour la vérification de propriétés exprimées à l'aide de la logique temporelle.

➢ **les systèmes de transition d'états :** particulièrement les automates tels que les automates à états finis, à entrées/sorties ou temporisés. Les automates possèdent l'avantage de pouvoir être représentés graphiquement sous forme de graphes dirigés.

➢ **modélisation de processus métier :** dans cette catégorie on distingue les langages de modélisation de processus d'entreprise tels que BPMN ou les diagrammes d'activité UML sont connus pour la simplicité et la lisibilité de leurs modèles. Cependant, leur manque de formalisme ne permet pas l'analyse formelle.

II. Conclusion :

Cet état de l'art a permis de présenter une étude des approches de composition de modèles représentatives du domaine.

Nous avons présenté dans ce chapitre les paradigmes de base de l'IDM. Nous avons aussi décrit l'approche MDA de l'OMG, en montrant comment cette approche permet de pérenniser le savoir-faire grâce à l'élaboration de modèles tout au long du processus de développement, d'augmenter la productivité grâce à l'opérationnalisation de ces modèles en définissant des transformations, et finalement de prendre en compte les plates-formes d'exécution en les intégrant directement dans les modèles.

Dans le chapitre suivant, nous allons présenter le diagramme d'activité d'UML.

Chapitre 4

Diagramme d'activité

I. Introduction :

Aujourd'hui, l'UML est devenu un standard largement accepté dans l'industrie de développement de logiciels orienté objet. On distingue plusieurs diagrammes Certains sont utilisés pour modéliser la structure d'un système, d'autres sont utilisés pour modéliser son comportement.

Les diagrammes d'activité sont utilisés pour modéliser les systèmes workflow, les systèmes orientés services et les processus métiers.

Le but de ce chapitre est de donner une description détaillée de l'un des diagrammes fondamentaux d'UML, qui est le diagramme d'activité.

Définition

UML (en anglais Unified Modeling Language ou «langage de modélisation unifié») est un langage de modélisation graphique à base de pictogrammes. Il est apparu dans le monde du génie logiciel, dans le cadre de la « conception orientée objet ». Couramment utilisé dans les projets logiciels, il peut être appliqué à toutes sortes de systèmes ne se limitant pas au domaine informatique.

Le langage de modélisation objet unifié UML est né de la fusion des trois méthodes objet : OMT (Object Modeling Technique) de James Rumbaugh, OOSE (Object Oriented Software Engineering) d'Ivar Jacobson, et la méthode de GradyBooch., puis il est normalisé par l'Object Management Group (OMG) en 1997 dans sa version 1.1, La dernière version diffusée par l'OMG est UML 2.4.1 depuis aout 2012. » [16].

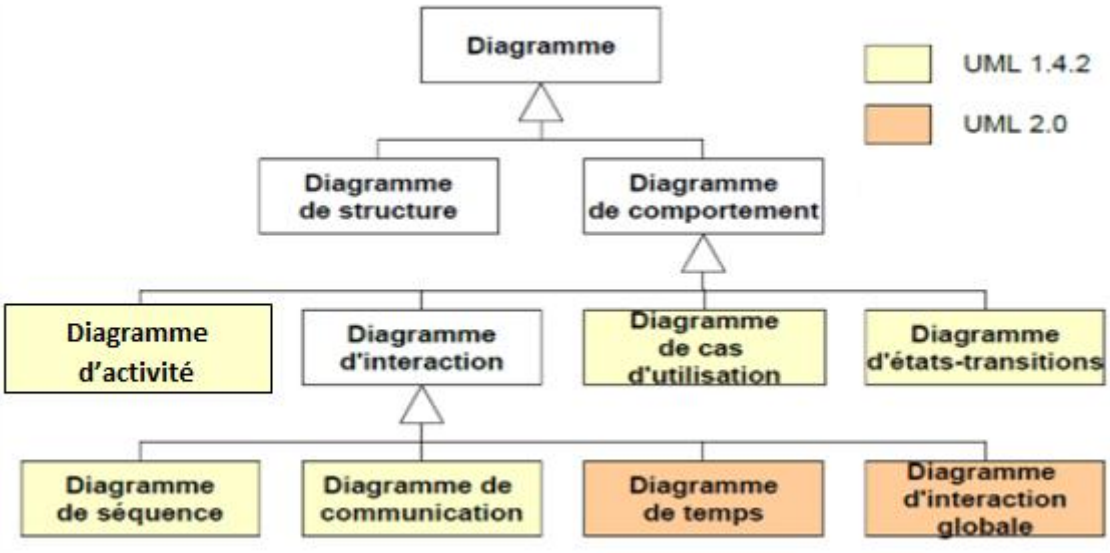


Figure 4.1 les diagrammes d'UML [21].

1. Diagrammes de Structure Composites

Un diagramme de structure composite est un diagramme, qui affiche la structure interne d'un classificateur. Y compris ses points d'interaction avec d'autres parties du système. Elle affiche la configuration et la relation de parties, qui ensemble, effectuer le comportement du classificateur contenant [13].

2. Diagramme de comportement ou diagrammes dynamiques (Behavior Diagram)

Les diagrammes comportementaux modélisent les aspects dynamiques du système. Ces aspects incluent les interactions entre le système et ses différents acteurs, ainsi que la façon dont les différents objets contenus dans le système communiquent entre eux [16].

2.1 Diagramme de cas d'utilisation

2.2 Diagramme d'états-transitions

2.3 Diagramme d'interaction

2.1 Digramme d'activité :

2.1.1 Définition

UML permet de représenter graphiquement les aspects dynamiques des systèmes à l'aide de plusieurs diagrammes comportementaux. Parmi eux, on distingue les diagrammes d'activité qui permettent de mettre l'accent sur les traitements. Ils sont utilisés pour représenter le comportement d'une méthode ou le déroulement d'un cas d'utilisation [16].

Les diagrammes d'activité d'UML constituent un outil de modélisation des systèmes workflows, des modèles orientés service et des processus métiers (ils montrent l'enchaînement des activités qui concourent au processus). Un modèle d'activité consiste en activités liées par des flux de données et de contrôle. Une activité peut varier d'une tâche humaine à une tâche complètement automatisée [16].

2.1.2 Intérêts des diagrammes d'activité

- ✓ Représenter graphiquement le comportement interne d'une opération, d'une classe ou d'un cas d'utilisation sous forme d'une suite d'actions.
- ✓ Modéliser un workflow dans un cas d'utilisation, ou entre plusieurs cas d'utilisations.
- ✓ Le diagramme d'activité est le plus approprié pour modéliser la dynamique d'une tâche ou d'un cas d'utilisation, lorsque le diagramme de classe n'est pas encore stabilisé.

2.1.3 Elément d'un diagramme d'activité

Les définitions de cette partie ont été inspirées essentiellement de l'OMG09.

Les éléments qui peuvent être contenus dans un diagramme d'activité sont :

2.1.3.1 Les nœuds (nodes)

2.1.3.1.1 Nœud d'activité (Activity Node)

Définition 1. ***Une activité***

Une activité est la spécification du comportement paramétré par un séquencement organisé d'unités subordonnées, dont les éléments simples sont les actions. Le flux de contrôle reste dans l'activité jusqu'à ce que les traitements soient terminés, et le flot d'exécution est modélisé par des nœuds reliés par des arcs [16].

Définition2. Un nœud d'activité

Un nœud d'activité est une classe abstraite permettant de représenter les étapes le long du flux d'une activité. Un nœud d'activité peut être l'exécution d'un comportement subordonné, comme un calcul arithmétique, un appel à une opération, ou la manipulation du contenu d'un objet. Les nœuds d'activité comprennent également le flux de contrôle des constructions, tel que la synchronisation, la décision et la concurrence.

Il existe trois types de nœuds d'activités :

- les nœuds de contrôle (control nodes).
- les nœuds d'exécutions (executable node).
- les nœuds objets (object node).

La figure ci-dessus représente graphiquement les nœuds d'activité.

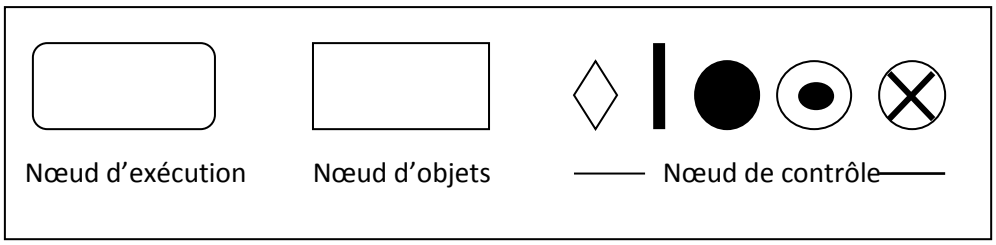


Figure 4.2 Notation nœuds d'activité [16].

B. Nœud de contrôle *(control node)*

On utilise les nœuds de contrôle pour guider le flux de contrôle (et le flux d'objets) à travers un groupe d'activités et d'actions. Les nœuds de contrôle viennent dans une variété de formes, en fonction de ce qu'on a besoin, ils servent comme un agent de circulation pour le flux de contrôle et les flux d'objets [2].

Les nœuds de contrôle sont les suivants

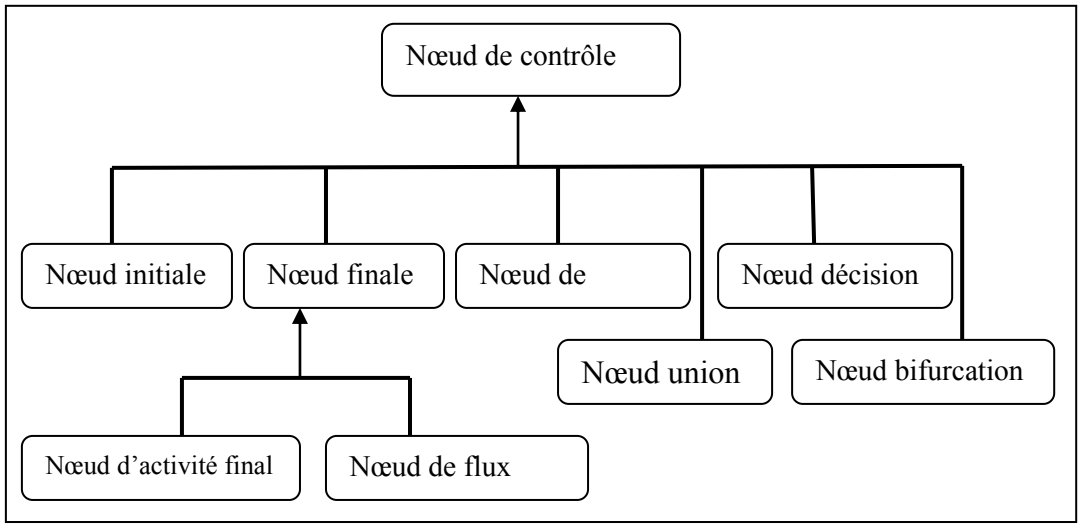


Figure 4.3 : Arbre de spécialisation des nœuds de contrôle [16].

B.1.Noeud initial (initial node)

Un nœud initial est un nœud de contrôle à partir duquel le flot débute. Il possède un arc sortant et pas d'arc entrant. Dans une activité on peut avoir plusieurs nœuds initiaux [16].



Figure 4.4 Notation de nœud initial [2].

B.2.Nœud final (final node)

Un nœud final est un nœud de contrôle dans lequel le flux d'activité s'arrête. Un nœud final peut avoir un ou plusieurs arcs entrants et aucun arc sortant. On peut distinguer deux types de nœuds finaux :

- ✓ **Les nœuds finaux d'activité (activity final node) :** Dans un nœud final d'activité, Lorsque l'un de ses arcs entrants est activé, l'exécution de l'activité encour s'achève, et tout nœud ou flux actif au sein de cette activité est abandonné.

- ✓ **Les nœuds finaux de flux (flow final node) :** L'arrivé du flux d'exécution à un nœud final de flux, implique la terminaison du flux de ce dernier. Mais cette fin n'a aucun effet sur les autres flux actifs de l'activité [2][16].

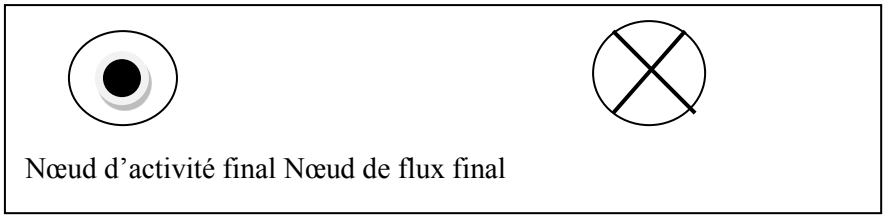


Figure 4.5 Notation de nœud final [2].

B.3.Nœud de fusion (merge node)

Un nœud de fusion est un nœud de contrôle, il rassemble plusieurs flots alternatifs entrants en un seul flot sortant. L'utilité de ce nœud n'est pas pour synchroniser des flux concurrents mais pour accepter un flux (en sortie) parmi plusieurs flux entrants [16].

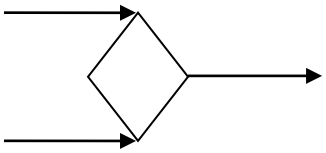


Figure 4.6 Notation de nœud de fusion [2].

B.4. Nœud de décision (decision node)

Un nœud de décision est un nœud de contrôle, il permet de faire un choix entre plusieurs flux sortants. Les flux sortants sont sélectionnés en fonction de la condition de garde qui est associée à chaque arc sortant [16].

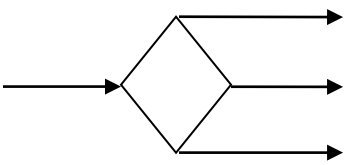


Figure 4.7 Notation de nœud de décision [2].

B.5. Nœud de bifurcation (fork node)

Un nœud de bifurcation est un nœud de contrôle qui sépare un flux d'entrée en plusieurs flots *concurrents* en sortie [2].

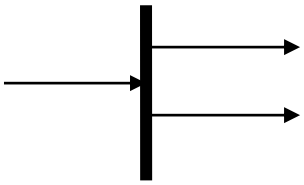


Figure 4.8 Notation de nœud de bifurcation [2].

B.6. Nœud d'union (join node)

Un nœud d'union (nœud de jointure) est un nœud de contrôle qui synchronise des flots multiples. Il possède plusieurs arcs entrants et un seul arc sortant. Ce dernier ne peut être activé que lorsque tous les arcs entrants sont activés.

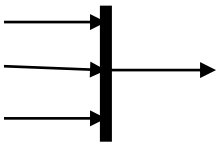


Figure 4.9 Notation de nœud d'union [2].

C .Nœud exécutable (executable Node)

Un nœud exécutable est une classe abstraite pour les nœuds d'activité qui peuvent être exécutés.

Action

Une action est un nœud d'activité exécutable, c'est le plus petit traitement qui puisse être exprimé en UML. L'exécution d'une action peut être une transformation ou un calcul dans le système modélisé (affectation de valeur à des attributs, création d'un nouvel objet, calcul arithmétique, émission ou réception d'un signal,…) [2].

La notation d'une action est un rectangle avec des coins arrondis comme illustre la figure 4.12



Figure 4.11 Notation d'Action.

Nous citons ci-dessous quelques types d'actions prédéfinis dans la notation UML

- Action appeler (*call operation*).
- Action comportement (*call behavior*).
- Action envoyé (*send*).
- Action accepter événement (*accept event*).
- Action accepter appel (*accept call*).
- Action créé (*creates*).

2.1.3.2 Les arcs (edges)

Définition d'arc d'activité *(Activity Edge)*

Un arc d'activité est une connexion dirigée entre deux nœuds d'activité. Si l'arc a un nom, il est noté près de la flèche.

- **Flux d'objet (Object Flow):** Un arc de flux d'objets est un arc qui permet de transmettre des données entre des nœuds d'objet.
- **Flux de contrôle (Control Flow) :** Un flux de contrôle est un arc qui permet de décrire le séquencement de deux nœuds d'activité (un flux de contrôle démarre un nœud d'activité, après la terminaison d'une activité précédente). Il ne transmet pas des données [16].

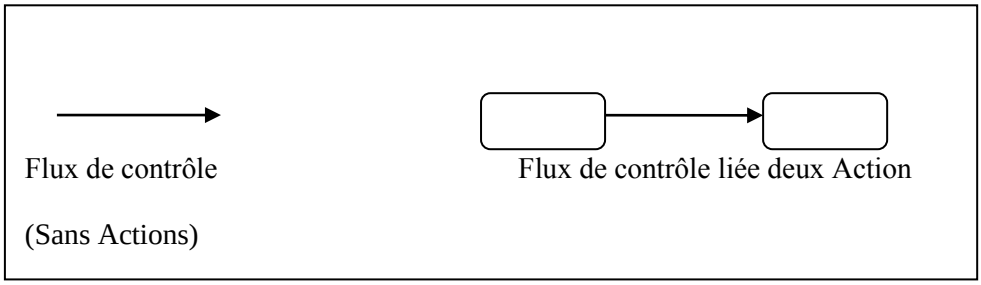


Figure 4.12 : Notation flux de contrôle [16].

Exemple

L'exemple suivant illustre l'utilisation des nœuds de contrôle.

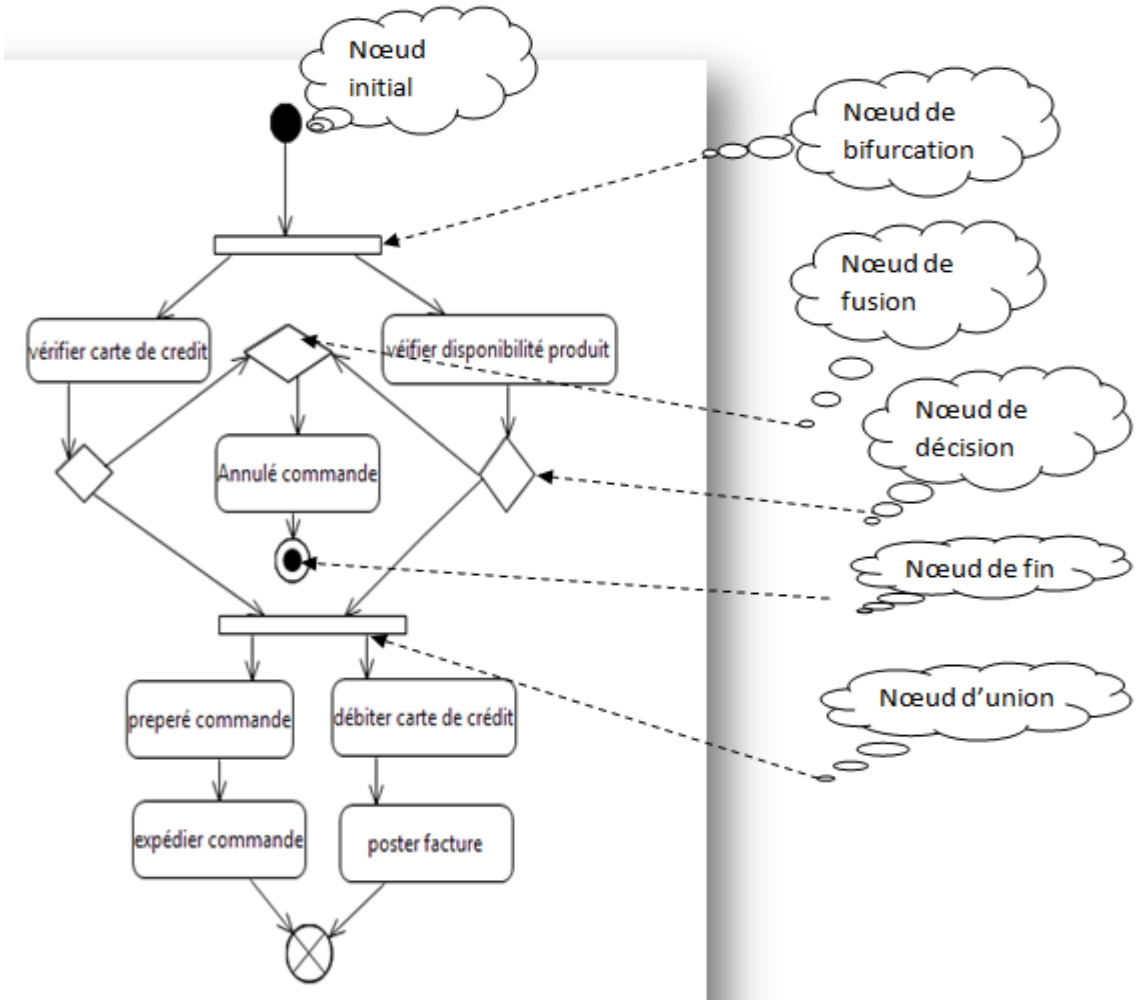


Figure 4.10 Exemple illustre l'utilisation des nœuds de contrôle [2].

II. Conclusion

Dans ce chapitre nous allons présenter brièvement le langage de modélisation unifié UML, son évolution et ses diagrammes. Ensuite, nous allons donner une description générale des diagrammes d'activité d'UML 2.0 et qui constituent un outil de modélisation des systèmes workflows, des modèles orientés service et des processus métiers. Nous allons insister sur les nœuds d'activité et les arcs, qui seront modélisés dans le formalisme source de la transformation constituant l'objet de notre travail.

Les diagrammes d'activité ont des mécanismes pour exprimer le séquencement, le choix et le parallélisme.

Dans le prochain chapitre on va présenter notre proposition pour la conception d'un modèle pour faire la transformation des diagrammes d'activité vers le langage de modélisation des processus métier BPMN.

Chapitre 5

Conception

I. Introduction :

Les diagrammes d'activité permettent la modélisation des systèmes workflows, des modèles orientés service et des processus métiers. Ils sont adaptés à la modélisation du cheminement de flux de données et de flux de contrôle. Ils incluent des mécanismes pour exprimer le parallélisme, le séquencement, le choix et les événements.

Le but de ce chapitre est de présenter une démarche pour la composition des services web basé sur la transformation des diagrammes d'activité d'UML vers BPMN, réalisée à l'aide de l'outil Eclipse Polar Sys avec le langage ATL (Atlas Transformation Langages).

II. L'approche proposée

Dans notre approche de composition des services web, nous proposons de décrire le service composite par un diagramme d'activité.

En suite, nous transformons ce diagramme en BPMN en se basant sur une démarche MDA. La figure 5.1 présente l'idée de notre démarche.

1.1 Conception de transformation d'un service composite

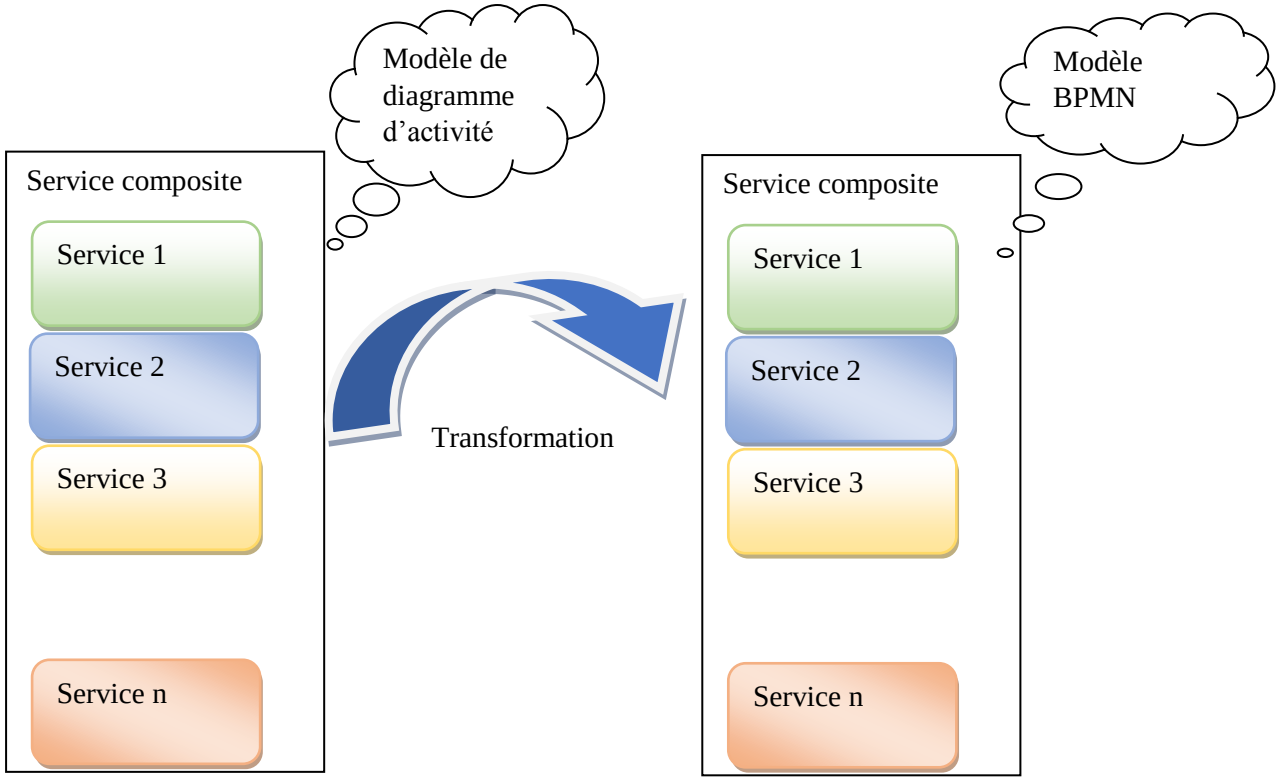


Figure 5.1 Schéma de transformation d'un service composite**.**

1.2 Schéma globale de transformation

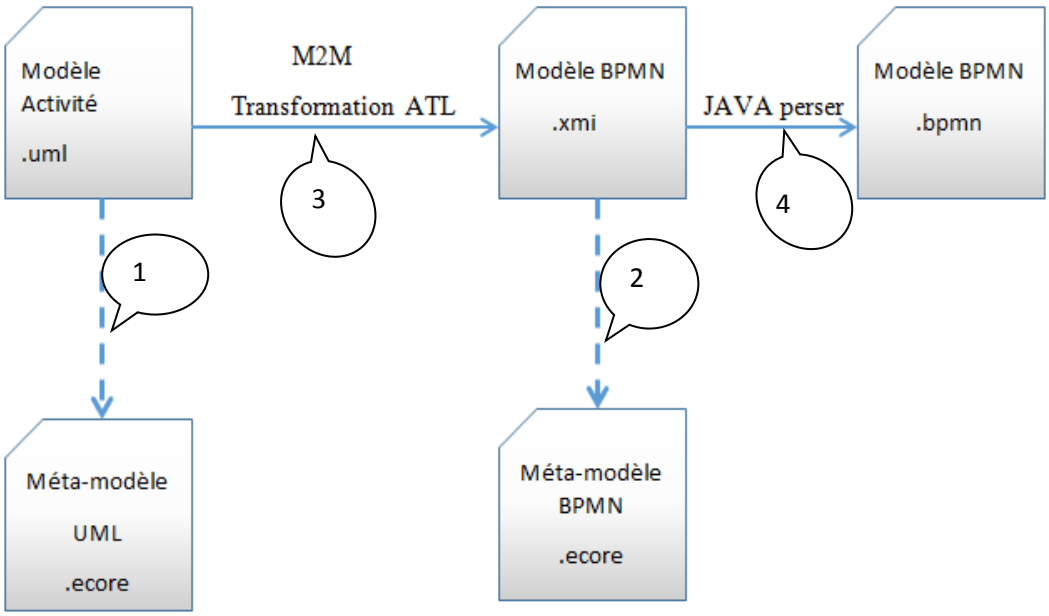


Figure 5.2 Schéma globale de transformation**.**

La figure 5.2 : représente les différentes étapes de la transformation du diagramme d'activité vers BPMN dont on a besoin

(1) un Méta-modèle source d'UML

(2) Un Méta-modèle cible de BPMN

(3) Après on a proposé un modèle de diagramme d'activité qui est conforme au Meta modèle d'UML.

(4) En suite grâce au langage ATL on a fait la transformation de fichier (.uml) vers un fichier (.xmi) ce qui illustre le Schéma si dessous, A la fin, on va générer le graphe BPMN à travers le fichier (.xmi) résultant.

1.3 Les Méta-modèles utilisés pour la transformation

Dans ce qui suite, nous présentons les méta-modèles de diagramme d'activité et de BPMN.

A. Méta-modèle de diagramme d'activité :

Nous avons présenté ce dessus le méta-modèle de diagramme d'activité proposé par l'OMG.

A.1 Description de Meta modèle de diagramme d'activité

La class **Node** : c'est la classe principale de méta-modèle de diagramme d'activité, la classe **Node hérite** Par 2 classes :

1) **ActivityEdge** : c'est une classe abstraite pour la connexion entre les activity Node, il couvre deux sous classe **Object flow** and **Contrôle Flow**.

2) **Activitynode** : c'est une classe abstraite qui représente les étapes d'une activité, elle hérite par 3 classes :

- La classe **exécutable Node** : est un nœud d'activité qui peut être exécuté, elle hérite par la classe :

Action : est une classe qui représente soit :

- ✓ **Send Signal Action.**
- ✓ **Accept Event Action.**
- ✓ **Call Behavior Action.**

- La classe **Contrôle Node** : On utilise les nœuds de contrôle pour guider le flux de contrôle (et le flux d'objets) à travers un groupe d'activités et d'actions.

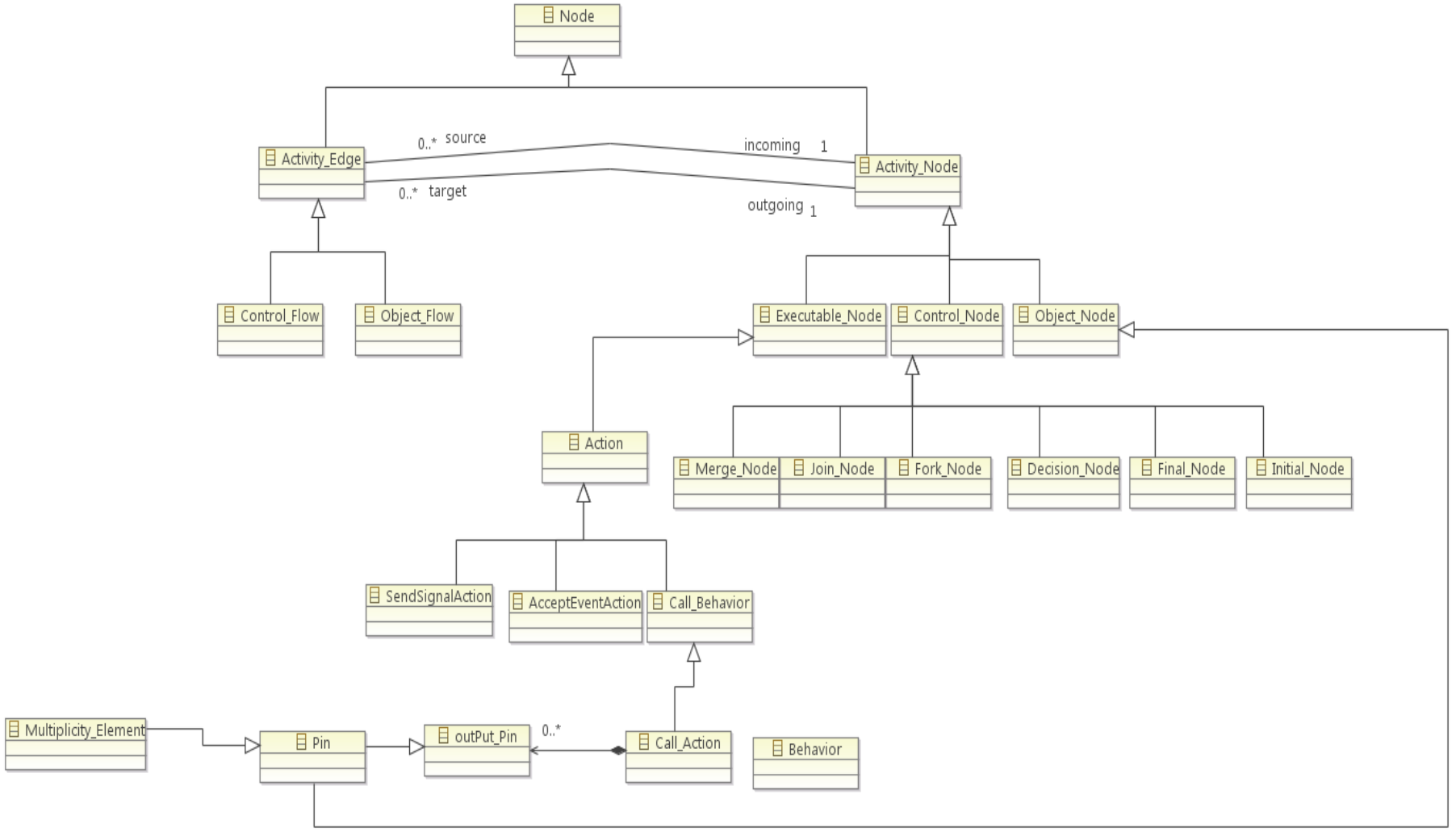


Figure 5.3 méta-modèle de diagramme d'activité [10] [18].

- **B .Le Méta-modèle de BPMN *:***

B.1 Description de Meta modèle de BPMN

Dans la **figure 5.4** qui représente le méta-modèle de BPMN, nous avons défini tous les éléments nécessaires pour construire un modèle BPMN, le processus BPMN contient deux classes principales.

- La classe "**Object**" qui est hérité par trois classes :

1) **Gateway** (Branchements) qui peut être de type AND-Split, AND-Join, XOR-Split ou XOR-Join,
2) **Event** (Evènements) qui peut être soit StartEvent, IntermEvent ou bien EndEvent, cette classe a un attribut nommé *Trigger* qui veut dire le type de l'évènement (soit None (rien), ErrorTrigger, TimerTrigger ou CancelTrigger).
3) **Activity**(Activité) qui peut être une tâche (**task**) ou un sous-processus (**subProcess**). La classe **subProcess** (sou-processus) est composé d'autre Flows et Objects.

- La classe "**Flow**" (c-à-d flux) a deux relations avec les objets (fromObject et toObject), cette classe fait la relation entre les objets de BPMN (la relation de l'objet précédent « fromObject » avec l'objet suivant « toObject »).L'attribut **FlowType** veut dire qu'il y a deux types de flux, soit un flux de séquence ou un flux de message.

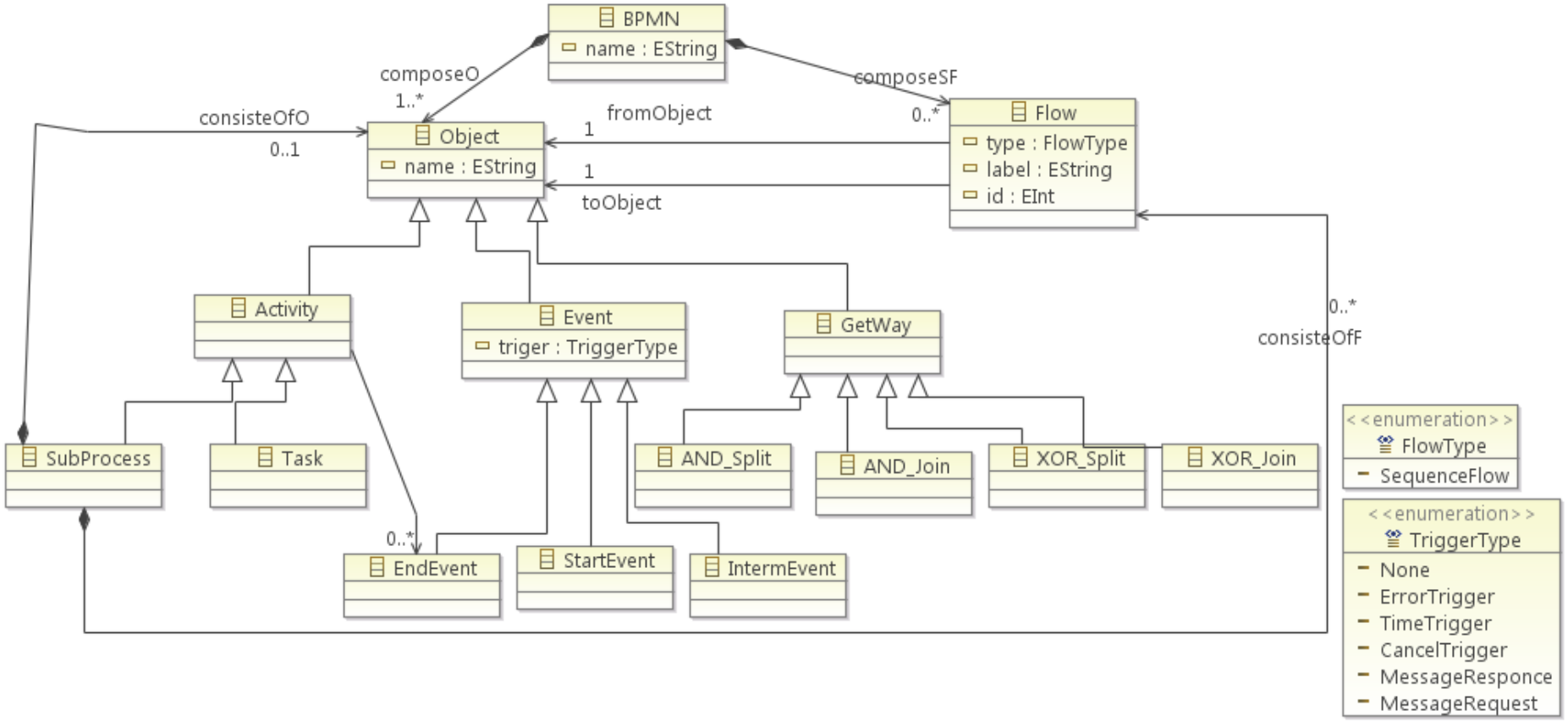


Figure 5.4 méta-modèle de BPMN [5].

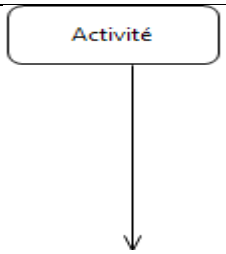
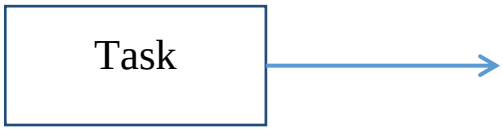
1.4Les règles transformations :

La règle 1 : **IntialeNod**e dans diagramme d'activité transformer en **Start Event**«None » dans **BPMN**.

Activité	BPMN
	

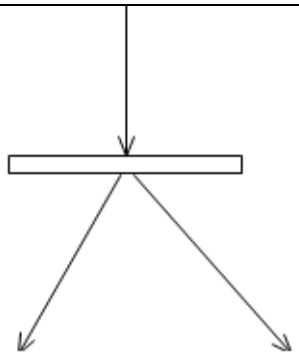
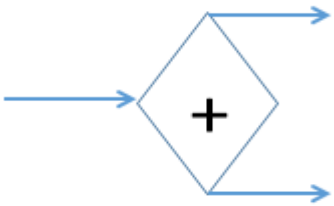
Tableaux 5.1 Initial Node vers Strat Event.

La règle 2 : Activité dans diagramme d'activité transformé en **Task** dans **BPMN.**

Activité	BPMN
	

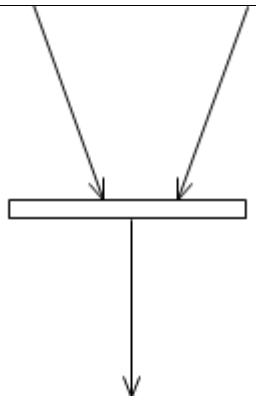
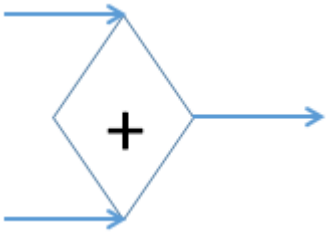
Tableaux 5.2 Activité vers Task.

La règle 3 : ForkNode dans diagramme d'activité transformé en **And splite** dans **BPMN.**

Activité	BPMN
	

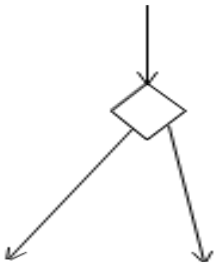
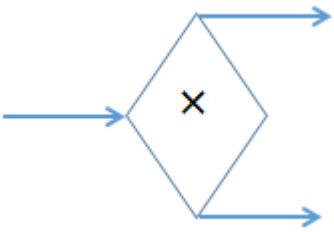
Tableaux 5.3 Fork Node vers And Split.

La règle 4 : JoinNode dans diagramme d'activité transformé en **And Join** dans **BPMN.**

Activité	BPMN
	

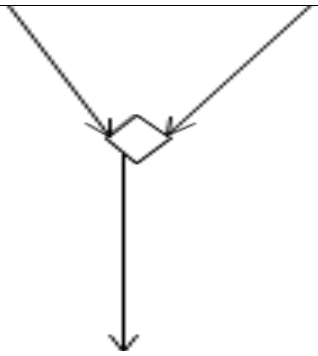
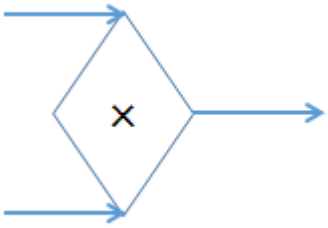
Tableaux 5.4 Join Node vers And Join.

La règle 5 : **DecisionNode** dans diagramme d'activité transformé en **Xor Split** dans **BPMN.**

Activité	BPMN
	

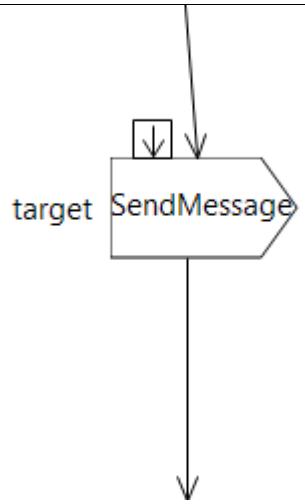
Tableaux 5.5 Decision Node vers Xor Split.

La règle 6 : MergeNode dans diagramme d'activité transformé en **XorJoin** dans **BPMN.**

Activité	BPMN
	

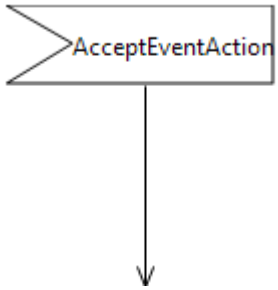
Tableaux 5.6 Merge Node vers Xor Join.

La règle 7 : **Send Signal Action** dans diagramme d'activité transformé en **Intermédiaire Event « Message request »**dans **BPMN.**

Activité	BPMN
	

Tableaux 5.7 Send Signal Action vers Intermédiaire Event « Message request ».

Règle 8 : Accept Event Action dans diagramme d'activité transformé en **Intermédiare Event « Message responce »** dans **BPMN.**

Activité	BPMN
	

Tableaux 5.8 Accept Event Action vers Intermédiare Event « Message responce ».

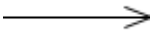
Règle 9 : Final Node dans diagramme d'activité transformé en **End Event** dans **BPMN.**

Activité	BPMN
	

Tableaux 5.9 Final Node vers End Event.

Règle 10 : Control Flow dans diagramme d'activité transformé en **Flow** dans **BPMN.**

La règle 10 qui transformé Control Flow permet de mettre les liaisons entre les éléments (in and out).

Activité	BPMN
	

Tableaux 5.10 Control Flow vers Flow.

1.5 Exemples de scenario Algorithmique de transformation des règles

- **Règle *d'initiale noude* vers *Strat Event* et *final noude* vers *End Event***

```
Algorithme Transforme_Regle1et 9
 Debut
   Si   node = initial node  alors
      Start_Event.name == initial.name  ;
       Si non
          Si (node = final node) alors
                    End_Event.name==fianl.name
Fin.
```

- **Règle *Send Signal Action* vers *Intermédiaire Event* « Message Request »**

```
Algorithme Transforme_Regle 7
Debut
  Si   node = Action node  alors
     Si Action = Send Signal Action  alors
          Intermédiaire Event.name == Send Signal Action .name ;
Fin.
```

III. Conclusion :

Dans ce chapitre nous avons proposé une approche MDA basée sur la méta-modélisation et les règles de transformation, nous avons présenté les méta-modèles utilisés des diagrammes d'activités et BPMN ensuite, nous avons définit les règles de transformations dont chaque composante de diagramme d'activité à gauche est transformée en un composant BPMN à droite.

Dans le chapitre suivant nous allons implémenter cette approche avec un cas d'étude.

Chapitre 6

Implémentation et étude de cas

Introduction

Après avoir exposé l'approche de transformation de diagramme d'activité vers BPMN pour la composition des services web, nous allons à présent décrire la phase de réalisation relative à cette dernière. Pour illustrer les différentes idées et concepts inclus dans notre approche, nous allons présenter dans ce chapitre un exemple sous la forme d'une étude de cas. Nous allons choisir « le dépôt de dossier de **recrutement** à l'aide des services web » comme un domaine d'application.

1. **Schéma globale de transformation**

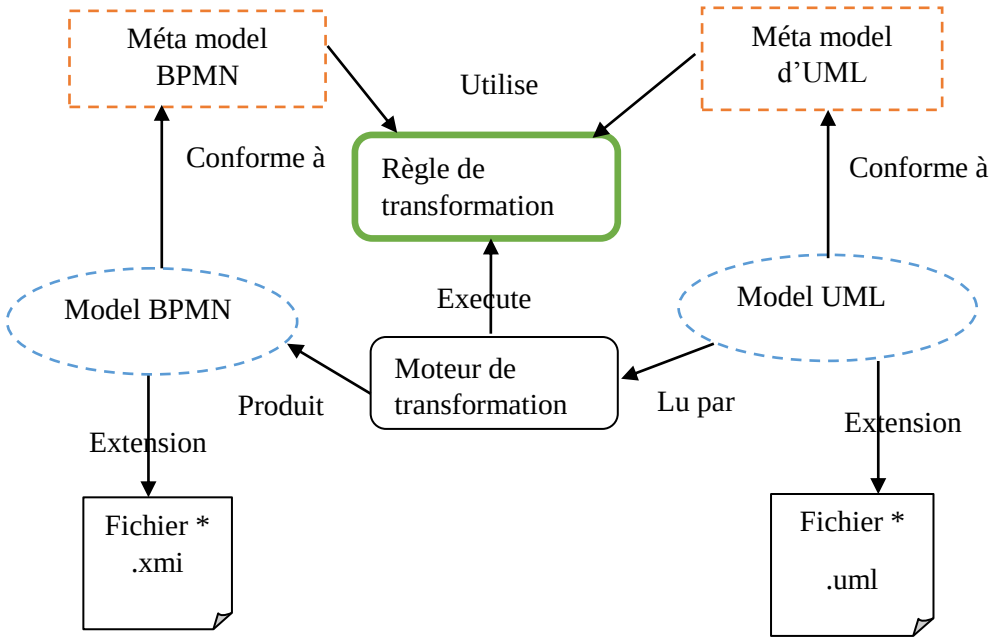


Figure 6.1 Schéma global de transformation.

2. **Outils et langages utilisées**

Les outils utilisés pour la création des diagrammes d'activités et la transformation vers BPMN sont :

- **Eclipse Papyrus** est un outil de modélisation graphique et d'ingénierie à base d'architecture open source.
- **ATL** (ATLas Transformation Language) est un langage de transformation de modèle et une trousse à outils, ATL fournit des moyens de produire un ensemble de modèles cibles à partir d'un ensemble de modèles sources,ATL se compose :

  - un compilateur et d'une machine virtuelle.
  - un IDE s'appuyant sur Eclipse.

- **Eclipse IDE** est un environnement de développement intégré libre (le terme Eclipse désigne également le projet correspondant, lancé par IBM) extensible, universel et polyvalent, permettant potentiellement de créer des projets de développement mettant en œuvre n'importe quel langage de programmation. Eclipse IDE est principalement écrit en Java grâce à des bibliothèques spécifiques, est également utilisé pour écrire des extensions. Nous avons utilisé cet environnement pour intégrer des plugins pour ***ATL, WindowBuilder***.
- **Java** est un langage de programmation à usage général, Java est notamment largement utilisée pour le développement d'applications d'entreprises et mobiles.

3. Méthodes utilisées dans la transformation

➢ **Helpers**

Les "**helpers**" ATL permettent aux développeurs de définir leurs propres méthodes dans les différentes unités d'ATL. Dans le contexte d'ATL les « helpers » se sont des méthodes. Ils permettent de factoriser du code ATL que l'on peut appeler à différents endroits du programme. Il existe deux types de « helpers » : fonctionnel et attribut. Un « attribut helper » est utilisé comme un attribut alors qu'un « fonctionnel helper » accepte des paramètres.

```
**********helpers**********
Helper def : name() : OclAny = OclUndefined ;
```

Figure 6.2 Syntaxe de déclaration des helpers.

➢ **Règles**

Une règle est caractérisée par deux éléments obligatoires un motif sur le modèle source (from) avec une éventuelle contrainte un ou plusieurs motifs sur le modèle cible (to) qui expliquent comment les éléments cibles sont initialises à partir des éléments sources correspondant.

```
**********rules**********
Rule name {
From Input_name : input_element
        To Output_name   : output_element( )
}
```

Figure 6.3 Syntaxe de déclaration d'une règle.

4. Etude de cas

Dans cette section nous allons présenter un scénario de composition des services web pour notre étude de cas.

4.1 Scenario de la composition

L'objectif principal de ce scénario est de facilité a un demandeur de travail de déposé un dossier de recrutement dans une établissement, le candidat donne son numéro d'identité nationale a l'établissement, cette dernière utilise un service web de vérification (ministère de l'intérieur) pour vérifier si ce numéro correspond à cet demandeur réellement, si le numéro est valide alors trois autres services vont se déclencher en parallèle (service universitaire, service santé et service scolarité) pour obtenir les diplômes universitaires, les conditions de santé et les diplômes scolaires dans ce cas-là, le processus se termine avec succès ,dans le cas où le numéro ne correspond pas au demandeur alors le processus s'arrête et l'établissement renvoyez le numéro ID(identité nationale) au demandeur pour le vérifier .

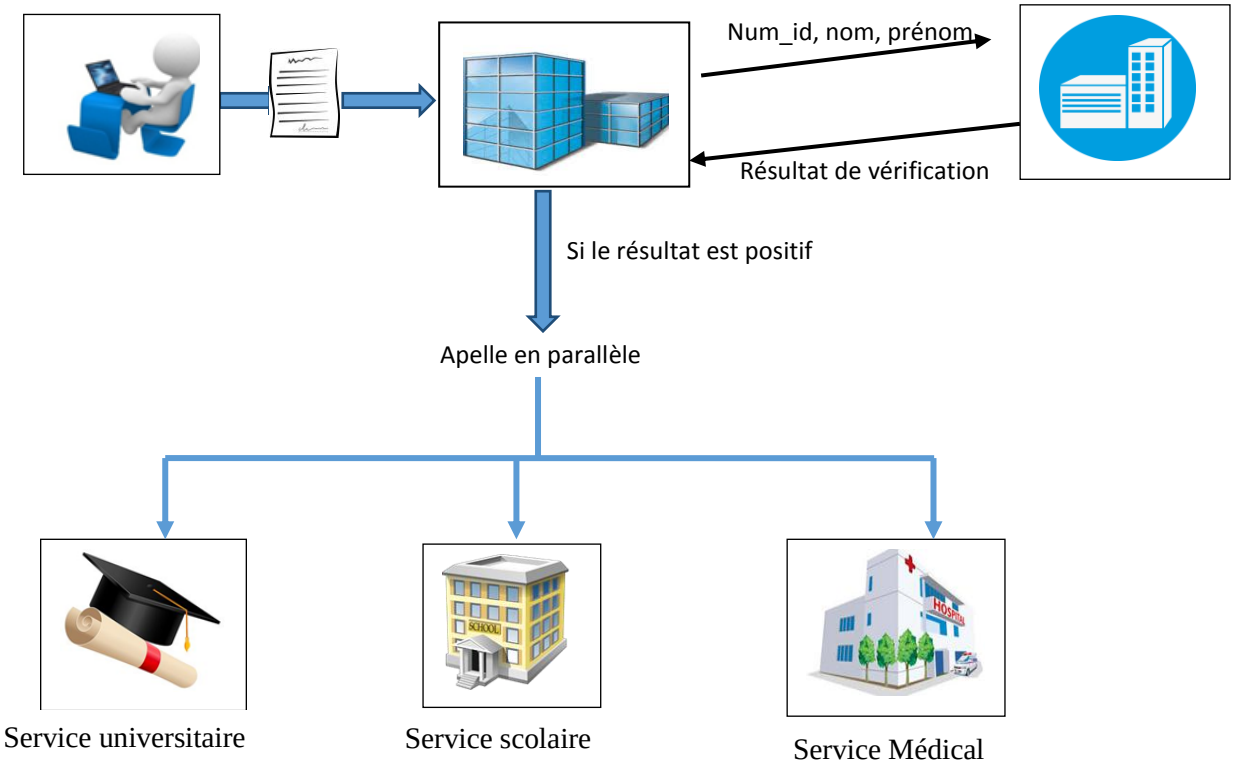


Figure 6.4: Schéma explicatif de notre cas d'étude

4.2 Représentation de scénario à l'aide de diagramme d'activité

Nous allons faire la modélisation de l'orchestration pour ce scénario avec le diagramme d'activité sous Eclipse Papyrus Modeler pour obtenir le modèle source.

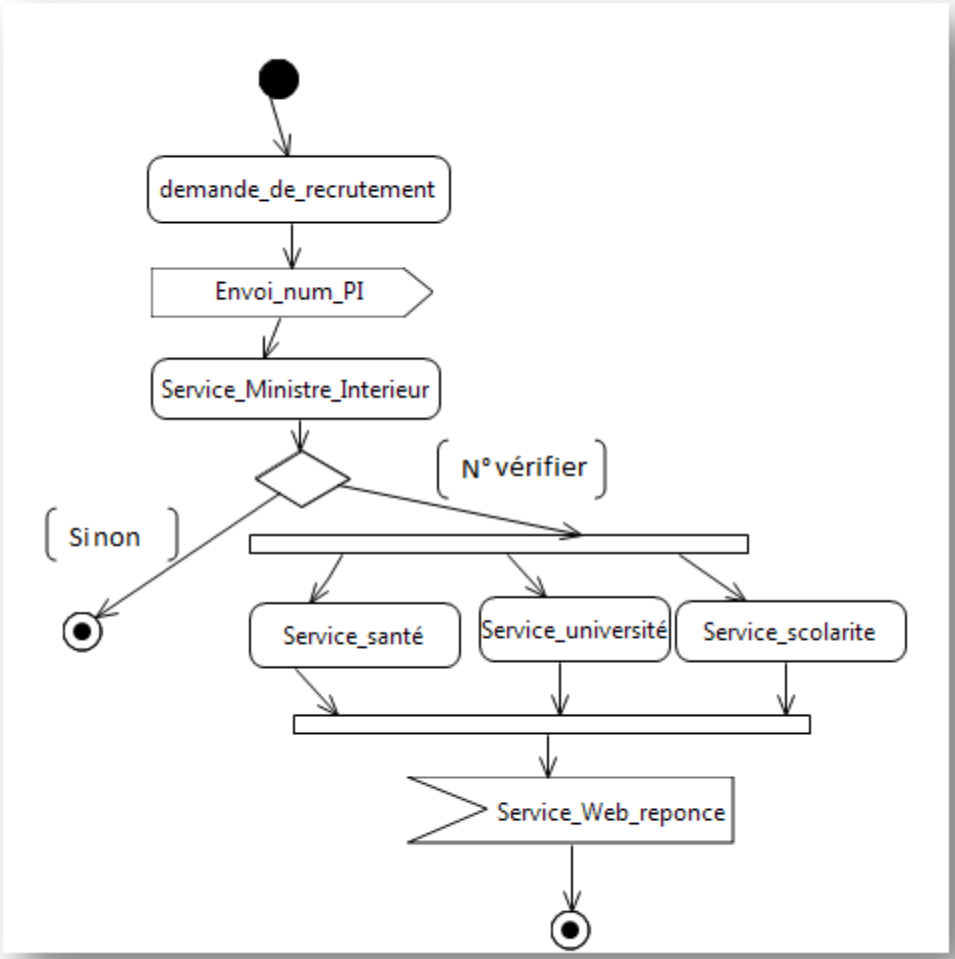


Figure 6.5: Diagramme d'activité pour l'orchestration des services.

5. Les interfaces :

Nous avons présenté notre application par des interfaces qui nous avons la programmé avec « EclipseJeeNeon », La figure ci-dessous (**Figure 6.6**) montre la page d'accueil de l'application.

- **Page d'accueil de l'application**

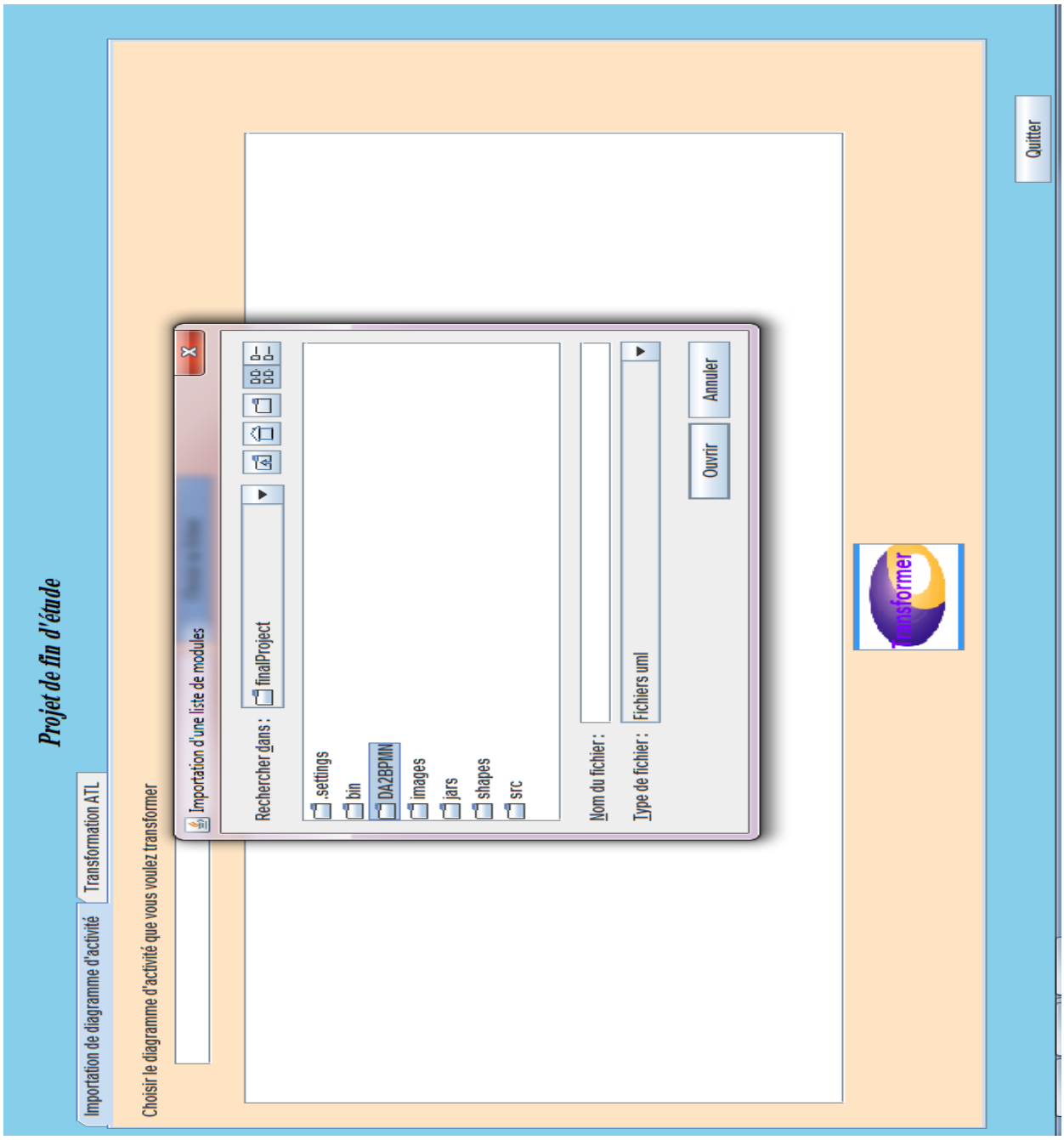


Figure 6.6 Représentation de la page d'accueil.

Première interface de notre application

La figure 6.7 représente la première interface de notre application qui permet d'importer un modèle de diagramme d'activité (.uml) pour le transformer vers un fichier XMI(.xmi), voici l'affichage de fichier (.uml) (**Figure 6.7**).

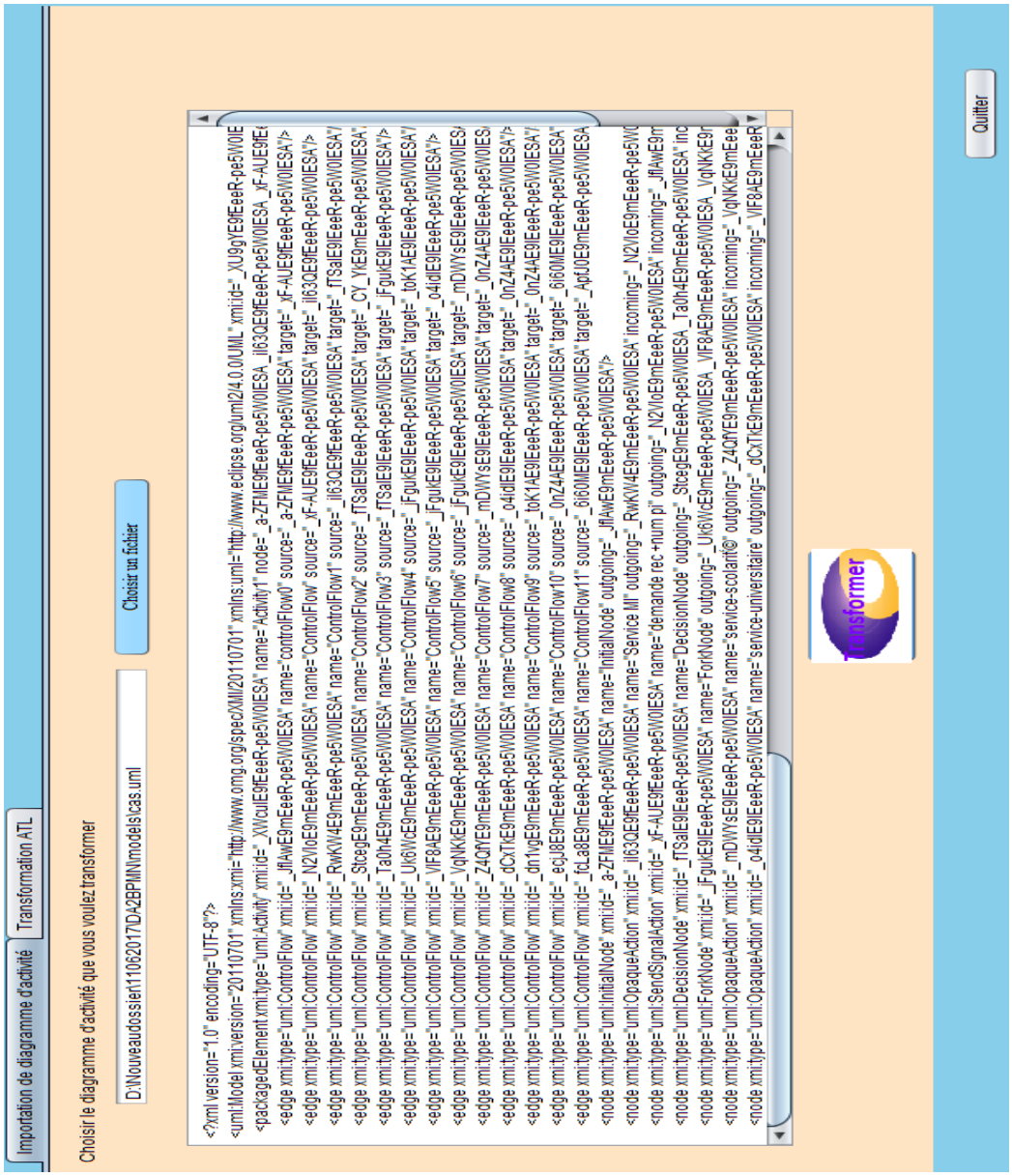


Figure 6.7 Importation de modèle de diagramme d'activité UML.

Deuxième interface de notre application

La figure 6.8 représente la deuxième interface de notre application qui démontre le fichier résultat d'extension (.xmi) après la transformation de fichier source (.uml) et l'extraction de fichier d'extension (.bpmn) de cette résultat.

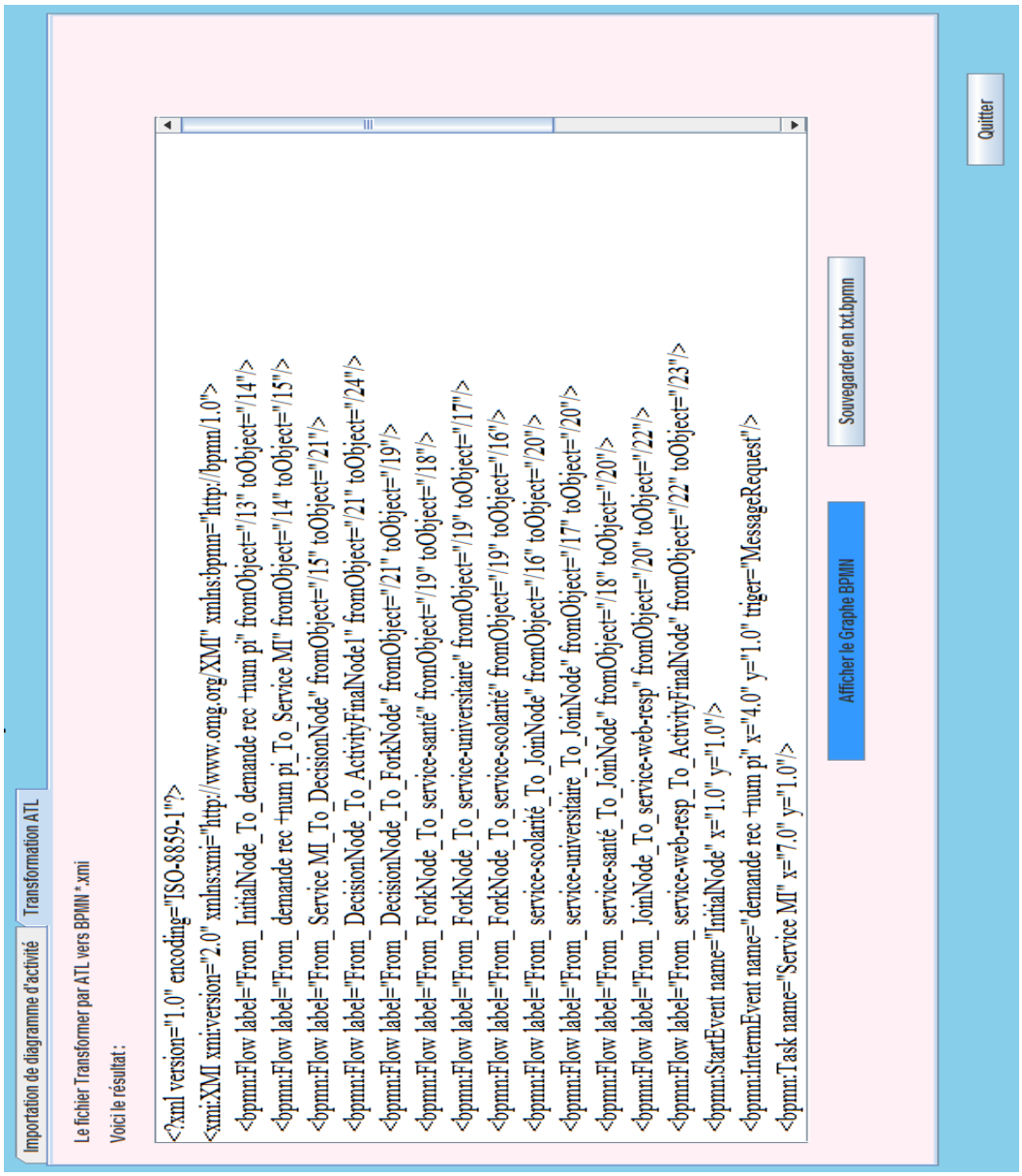


Figure 6.8 Fichier (.xmi) générer à partir de fichier (.uml).

- **Troisième interface de notre application**
  - ✓ Cette figure représente le graphe BPMN générer à partir de fichier XMI (.xmi)

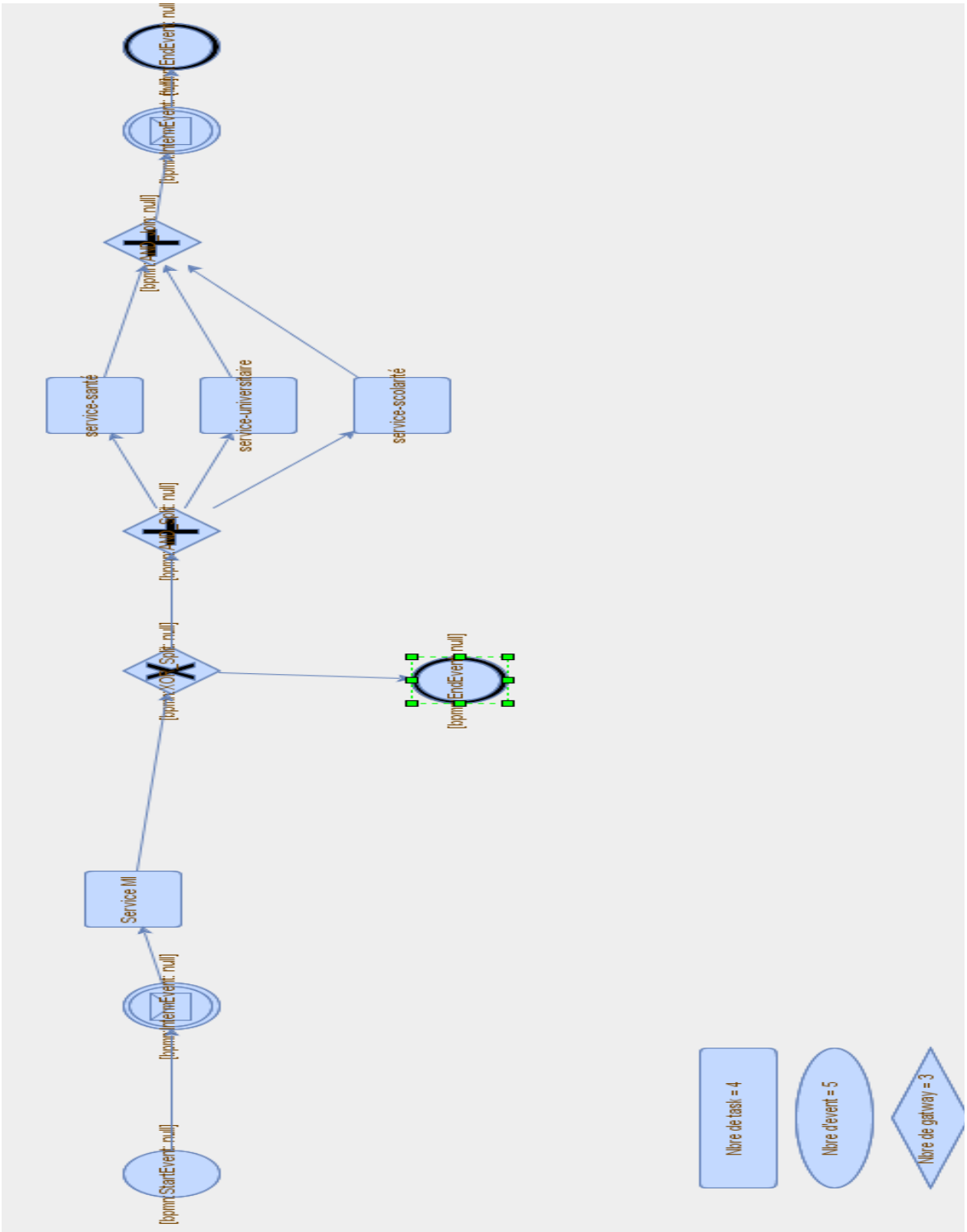


Figure 6.9 Graphe BPMN générer.

- ✓ L'ouverture de fichier (.bpmn) a l'aide de l'outil de modélisation graphique BPMN Modeler.

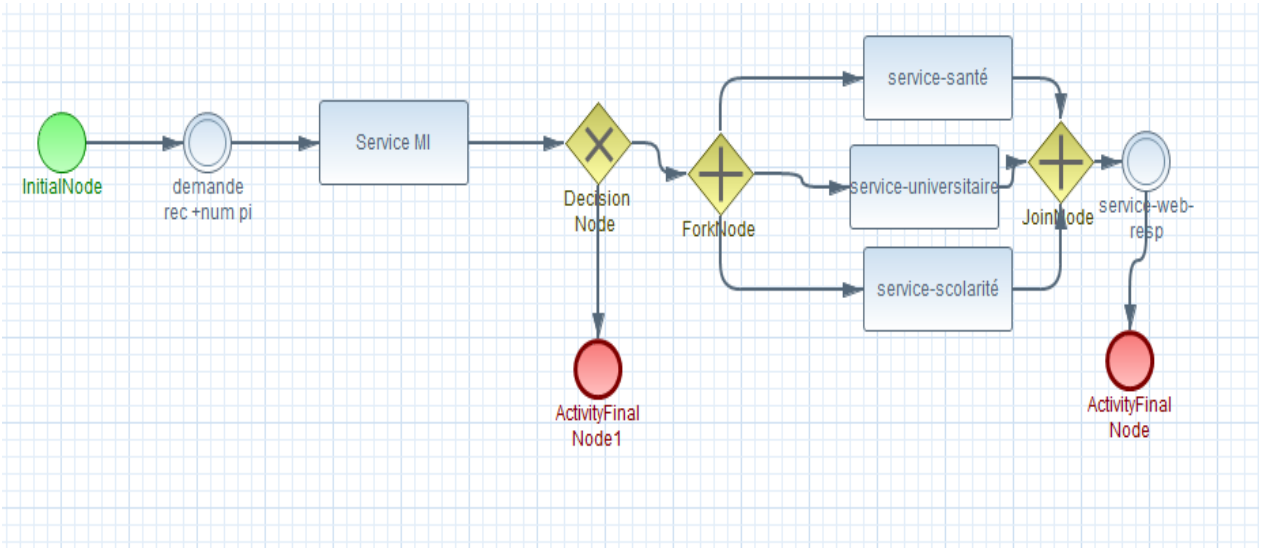


Figure 6.10 Graphe BPMN générer à l'aide de BPMN Modeler.

Conclusion

Dans ce chapitre, nous allons présenter les différant étapes pour réaliser notre application. Tous d'abord, nous allons commencer par une explication sur l'approche proposée, en suite nous allons présenter les outils, les langages et les méthodes utilisés durant la réalisation, cette étape a été suivie par une présentation de notre cas d'étude a l'aide d'un diagramme d'activité. En fin nous allons présenter les étapes d'exécution de notre application.

Conclusion générale

Dans ce mémoire nous sommes intéressés à la transformation des diagrammes d'activité vers des spécifications BPMN. Nous avons commencé avec une introduction des deux langages suivie d'une représentation des approches de transformation de graphes pour finir avec une technique permettant la génération d'un graphe BPMN à partir d'un diagramme d'activité.

Nous avons proposé une approche simple pour la modélisation de la composition des services web. Le travail que nous avons présenté se situe à l'intersection de plusieurs axes de recherche dont nous pouvons citer : la modélisation de la composition des services web, la méta-modélisation de diagramme d'activité et BPMN et la transformation des modèles.

La technique proposée permet de générer automatiquement un fichier BPMN lisible après l'exploration des composants intégrants le diagramme d'activité qui peuvent être convertis en fichier BPMN. En effet, une série des règles de transformation sont établies pour traiter les composants de diagramme d'activité considérés comme bien structurés, tel que ces règles, permettent de traités certains modèles de diagramme d'activité. Pour cela, nous avons implémenté notre approche proposée dans un outil JAVA qui permet de faire une transformation automatique d'un modèle de diagramme d'activité vers un modèle BPMN pour générer le fichier BPMN.

Dans un travail futur, nous comptons exploiter notre approche pours tous les diagrammes d'UML. Nous comptons également étendre la technique pour couvrir un sous-ensemble plus large d'éléments BPMN.

Bibliographie

[1] Sofiane Chemaa Thèse de doctorat : « Une approche de composition de services Web à l'aide des Réseaux de Petri orientés objet » Département d'informatique fondamentale et ses applications, Université ABDELHAMID MEHRI, CONSTANTINE 2, 2014.

[2] Klalib Abdelkader " Proposition d'une approche pour la génération automatique d'un fichier BPEL" Université Medea, promotion 2016.

[3] Nerea Arenaza projet de Master « Composition semi-automatique de Services Web » Ecole polytechnique fédérale de Lausanne 2006.

[4] DAHA Hanane, LIFA Siham : « Une approche formelle pour la composition des services web », Université ECHAHID HAMMA LAKHDAR - EL-OUED 2015.

[5] Bouteldja Radia, Projet de fin d'étude « Approche pour la génération automatique d'un fichier BPEL», Université Yahia Fares Médéa 2016.

[6] Mohamed Yassine KATIRI, « La modélisation des processus d'affaires en utilisant BPMN ainsi que qualigramme », Université du Québec 2010.

[7] Frédéric Jouault, Brahim-Khalil et William Piers LOUKIL « Model-to-model Transformation with ATL» EPL 2008.

[8] Adil Anwar « formalisation par une approche IDM de la composition de modèle dans le profil VUML » THESE Soutenue en vue de l'obtention du titre de DOCTEUR EN INFORMATIQUE DE L'UNIVERSITE DE TOULOUSE, 2009.

[9] Mireille Blay-Fornarino : « Gestion de processus » Licence professionnelle IDSE 2013.

[10] Julien Da Costa, « BPMN 2.0 pour la modélisation et l'implémentation de dispositifs pédagogiques orientés processus » Université de Genève 2014.

[11] BENILLES et BabaChik « Méthode Formelle pour la Modélisation et la Vérification des Processus Métiers » Université de Medea 2016.

[12] Kholladi Mohamed Naoufel « une Approche de transformation de la notation BPMN vers BPEL basée sur la transformation de graphe », Université Mentouri Constantine 2009.

Bibliographie

[13] KHERBOUCHE Mohammed Oussama « Contributions à la gestion de l'évolution des processus métier » Présentée envue d'obtenir le grade de DOCTEUR de l'Université du Littoral Côte d'Opale 2013.

[14] Houda HAMROUCHE «Une Approche de transformation des diagrammes d'activité d'UML vers CSP basée sur la transformation de graphes ». Ecole Doctorale en Informatique de l'Est – EDI Est 2012.

[15] Livre : Jean-Yves, « Didier Introduction à UML », UML™ (Unified Modeling Language) University D'EVRY, 2006.

[16] Coalition, T. W: «Workflow Management Coalition Terminology & Glossary », Rapport numéro WFMC-TC-1011 Issue 3.0, 1999.

[17] Guerrouf Faycel mémoire de Magister : « Une Approche de Transformation des Diagrammes d'Activités d'UML Mobile 2.0 vers les Réseaux de Petri ». Université El Hadj Lakhdar – BATNA 2011.

[18] OMG **«** UML 2.0 Superstructure Specification 2004 »

Webographie

[W1] IBM Web services tutorial.

Online: http://www-106.ibm.com/developerworks/webservices/

[W2] www.w3c.org

[W3] http://www.x2i.fr/2016/bpmn-sauver-projet-magento/