République Algérienne Démocratique et Populaire
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique
Université Abbes Laghrour Khenchela
Faculté des Sciences et de la Technologie
Département de Mathématiques et Informatique

Mémoire de fin d'études

pour l'obtention du diplôme de Master en Informatique (L.M.D).
Spécialité : Génie logiciel et systèmes distribués

Une Approche Dirigée par les Modèles pour la transformation de Modèle BPMN en DD-LOTOS

Présenté par :
Nihed Rahabi.

Dirigé par :
Dr Elkamel Merah.

Promotion : 2016/2017.

Dédicace

Je dédie ce mémoire :

A mes chers parents source de la tendresse et de noblesse, qui font le maximum pour me soutenir et m'encourager durant tout mon cursus .

A Mes frères et mes sœurs à qui je souhaite beaucoup de réussite et de bonheur. à mon amie et ma sœur Hana qui ma aider dans ce mémoire.

A tout mes amis et ma famille qui sont chers, à tous ceux que j'aime et m'aiment,et à Edward Willink qui ma beaucoup aider et répondu à toutes mes questions concernant les outils que j'ai utilisé dans mon travaillé.

A tout mes professeurs qui m'ont enseignée, et spécialement Dr Houassi qui m'a donné les principe de base dans l'informatique, et aussi à Dr Maarouk qui m'a appris de voir dans le haut horizon, et à Dr Merah, Dr Siam, Dr Mahdaoui et sa femme Dr chouhal, Dr Malik.

Aussi à tout les personnes qui m'ont aidée de pré ou de loin à la réalisation de ce travail.

Remerciements

Je tiens tout d'abord à remercier Dieu le Tout Puissant et Miséricordieux qui m'a donné la force et la volonté d'accomplir ce modeste travail. Mes remerciements vont ensuite à mon encadreur Dr Merah Elkamel et à Dr Maarouk Toufik Messaoud qui étaient toujours disponible pour rependre à mes questions et leur précieux conseils.

Résumé

Le langage BPMN (Business Process Modeling Notation) représente un nouveau standard industriel créé pour fournir une notation graphique accessible aux utilisateurs engagés dans un processus de modélisation de processus métier (business process). Cependant, le langage BPMN est semi-formel comme c'est le cas pour UML, et a donc besoin d'une sémantique formelle. Cette formalisation permet l'analyse et la vérification. Peu de travaux ont été proposés pour transformer les modèles BPMN vers des méthodes formelles. En se basant sur une sémantique de ces modèles en DD-LOTOS, nous proposons une approche qui consiste à transformer automatiquement un sous-ensemble des notations BPMN en code DD-LOTOS. Cette approche a été implémentée en utilisant principalement Acceleo un outil sous Eclipse de génération de code.

Mots-Clés

BPM, BPMN, DD-LOTOS, Transformations de modèles, Méta-modélisation, Règles de transformation, Model To Tex, Eclipse, Acceleo

Abstract

The language BPMN(Business Process Modeling Notation) represent a new industry standard created to supply a graphic notation accessible to the users launched in a process of modeling of the business process. However, the language BPMN is semi-formal as it is the case for UML, and therefore needs a formal semantics. This formalization allows the analysis and check. Few works were proposed to transform the models BPMN towards formal methods. By being based on a semantics of these models in DD-LOTOS, We propose an approach which consists in automatically transforming a subset of the BPMN notations into code DD-LOTOS. This approach was implemented by using mainly Acceleo a tool under Eclipse of generation.

Key-Words

BPM, BPMN, DD-LOTOS,Transformations of models, Meta Modelling, Rules of transformation, Model To Tex, Eclipse, Acceleo

Table des matières

Liste des tableaux

Table des figures

Listings

Introduction générale

Dans la vie quotidienne et à tout moment nous faisons partie d'un ou de plusieurs processus, parfois en tant que ressources, ou en tant que clients ou parfois en tant que données simples, parmi les types de processus qui existent on a le processus métier. le processus métier est défini par Adam Smith(le père des sciences économiques modernes) en 1776 comme un ensemble des évènements, des activités ou l'une des deux, qui s'exécutent d'une manière séquentielle ou parallèle pour arriver à la fin à un service ou à un produit livré aux clients, ces services fourni par les entreprises

D'autre part, les entreprises sont devenues plus intéressées par l'étude et la compréhension des processus internes, sur lesquels ils comptent pour leurs affaires. En effet, ces processus dépendent de la qualité de leurs services, leurs revenus, leurs coûts, et la satisfaction de leurs clients et par conséquent leur succès. En 1999 David McCoy a commencé à parler de la gestion des processus métier (Business Process Management), ce dernier est basé sur l'idée de conciliation entre les objectifs de l'organisation et les besoins de client. La modélisation des processus métier est une des concepts qui ont été abordé par cette approche.

En effet, la modélisation d'un processus consiste à la représentation visuel des activités de l'entreprise d'une manière abstraite, la notation BPMN(Business Process Management Notation)est parmi les notations les plus utilisées dans ce type de modélisation.Cette notation est facile à comprendre et capable de modéliser les processus métiers complexes. La diversité des composants proposés par BPMN et la capacité de représenter des comportements comme la concurrence et la synchronisation des processus augmentent la possibilité d'établir des modèles logiquement incorrects avec des erreurs de contrôle de flux (interblocage, etc.) , De plus, BPMN a été créer comme un langage de modélisation semi-formel, il n'inclut pas une sémantique formelle. Cette limitation rend l'analyse rigoureuse difficile, menant à des modèles ambiguës. La conversion de BPMN vers des modèles mathématiques, afin de les formaliser et de les valider peut être une tâche importante. Beaucoup de chercheurs ont été monopolisés unique-

ment pour transformer les modèles BPMN en des modèles formels. Dans ce mémoire, le modèle formel choisi est DD-LOTOS. DD-LOTOS est un langage formel qui prend en charge le temps et la distribution.

L'objectif que nous voulons atteindre est l'automatisation de la transformation de la notation BPMN vers DD-LOTOS, nous utiliserons l'approche IDM (L'Ingénierie Dirigée par les Modèles). Le principe de L'approche IDM est "tout est modèle", il permet de réutiliser des modèles de formalisme appelés méta-modèles, qui sont adaptables à toutes les plateformes, et il permet aussi de manipuler les modèles à travers les transformations, il existe deux types de transformations : Modele-to-Modele (M2M) et Modele-to-Text(M2T). La transformation rend les modèles opérationnels dans une approche IDM, ce qui augmente considérablement la productivité des applications.

Pour atteindre notre objectif d'automatisation de la transformation nous allons proposé un méta-modèle pour BPMN, Ensuite, nous efforcerons de réaliser un éditeur graphique pour créer des modèles d'une manière facile, et on fin, nous allons chercher une transformation.

Ce mémoire est composé de quatre chapitre, le premier est intitulé Contexte, dans ce chapitre nous allons abordé les concepts clés IDM, BPMN et DD-LOTOS. Le deuxième chapitre tourne ou tour des outils dont nous allons utilisé dans la réalisation de notre travail. Le troisième chapitre sera réservé à l'explication de notre travail pratique et il est intitulé une approche automatique de transformation d'un modèle BPMN vers un code DD-LOTOS. Et enfin, le quatrième chapitre qui présente une étude de cas.

Chapitre 1

Contexte

1.1 Introduction

Ce chapitre a pour but de mettre la lumière sur les concepts clés de notre thème. Dans un premier temps nous allons parlé de L'Ingénierie Dirigée par les Modèles (L'IDM)qui est apparu récemment dans le domaine de l'informatique, elle se base sur l'utilisation intensive et systématique des modèles tout au long du processus de développement des logiciels et aussi basé sur la transformation des modèles. La transformation des modèles est considéré comme une phase cruciale puisqu'elle définit l'automatisation qui peut être utilisée dans le processus de développement des logiciels. En deuxième lieu nous allons parlé de Business Process Management Notation(BPMN) , il constitue une nouvelle standard dans la modélisation des processus métiers. Et en dernier lieu nous allons présenté le langage DD-LOTOS.

1.2 L'approche IDM

L'ingénierie Dirigée par les Modèles(IDM) et connu par l'appellation de MDE(Model-Driven Engineering) en anglais[Jézéquel et al., 2006], est une méthodologie spécifiée dans le développement des logiciels basés sur le principe de tout est modèle, son objectif consiste à l'offre des méthodes et des techniques qui réaliseront l'homogénéité des systèmes complexes, l'automatisation de la productivité ainsi que sa maintenance.
Parmi les principes IDM : la séparation des préoccupations (métier et logique)et chacune de ces préoccupations est modélisé par un modèle et ces derniers s'intègrent par la transformation automatique.

1.2.1 Principes de L'IDM

Modèle

Dans[Combemale, 2008] le modèle est une représentation abstraite du système selon certains points du vue pour un objectif picris. Le modèle doit être utilisé pour répondre à des questions relatif au système, on déduit la première relation dans IDM entre le modèle et le système celle qui est un système est représenté par un modèle (la relation EstRepresentéPar) voir figure1.1. Un système peut avoir déférents modèles où chaque modèle représente un aspect du système.

Méta-Modèle

Un méta-modèle est aussi un modèle, qui fait une description des éléments d'un modèle, c-à-d un langage de modélisation qui définit le langage d'expression d'un modèle. La notion de méta-modèle montre la deuxième relation dans IDM est :(Est ConformeA) et on dit un modèle est conforme à un méta-modèle voir1.1.

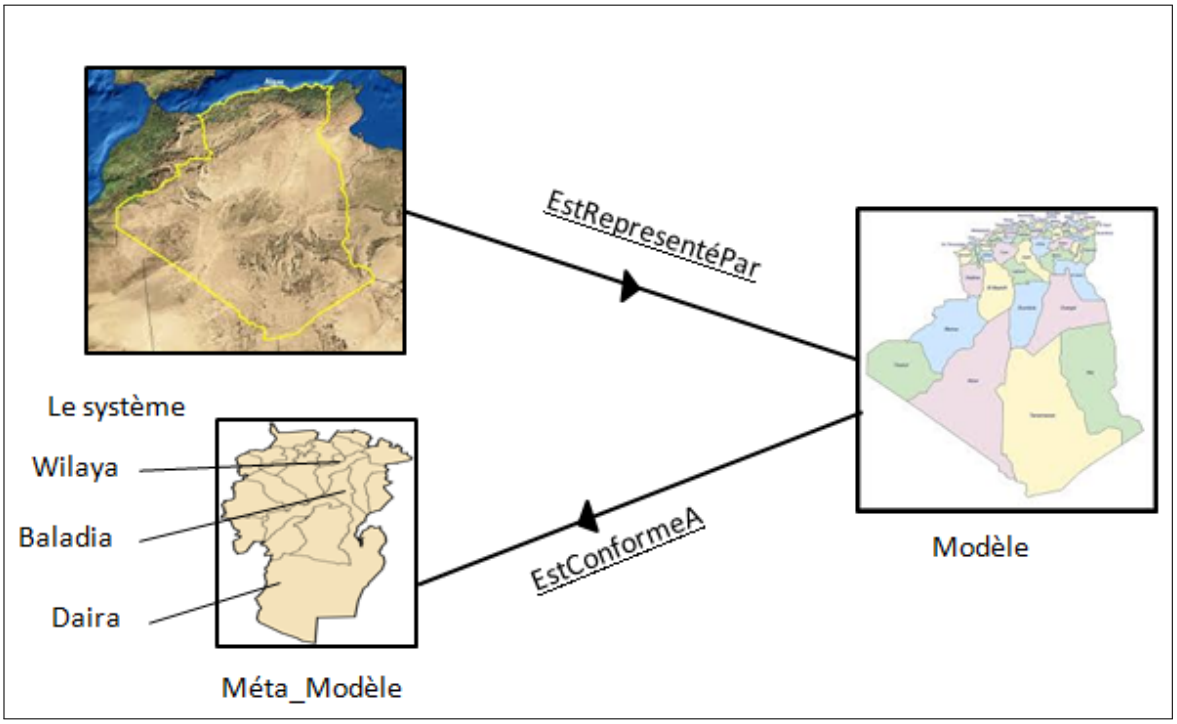


FIGURE 1.1 – Relation entre modèle et système

Méta-méta-modèle

Méta-méta-modèle est un modèle qui permet de décrire un Méta-modèle, donc est un modèle d'un langage de méta-modèlisation et on dit qu'un méta-modèle est conforme à un méta-méta-modèle (relation EstConformeA). Un méta-méta-modèle est conforme à lui même.

1.2.2 L'architecture dirigée par les modèles MDA

Auparavant, pour se bénéficier des avancées technologiques l'entreprise doit à chaque fois réécrire le code des applications avec l'architecture à base objet et l'architecture à base composant, parce que le développement repose généralement sur le code source, la séparation de préoccupation apparu comme une solution, pour cette raison l'OMG(Object Managment Group)en 2000 a proposée MDA [OMG, 2017](Model Driven Architecture).

MDA est une démarche de réalisation d'un logicielle, il est basé sur la séparation des préoccupations, ces dernières prennent en compte la séparation des aspects métiers et des aspects techniques d'une application. On peut réduire l'objectif de MDA en quelques mots : faire une conception une seule fois et construire sur n'importe quelle plateforme.
le MDA définit une architecture de spécification structurée en :

- **Modèle CIM** (Computation Independent Model) : le modèle d'exigence, décrivent les besoins fonctionnels de l'application.
- **Modèle indépendants des plateformes PIM**(Platform Independent Model) : le modèle d'analyse et de conception son rôle est donnée une vision structurelle et dynamique de l'application et indépendant de la conception technique .
- **Modèle spécifique aux plateformes PSM**(Platform Specific Model) : le modèle de code qui décrit l'implémentation d'une application sur une plateforme particulière.

La mise en oeuvre du MDA est entièrement basée sur : les modèles (méta-modélisation) et les transformations de ces modèles. MDA à quatre niveaux d'abstraction comme se présente dans la figure 1.2

- **Le niveaux M0** : Ce niveau représente le système réel à modéliser,par exemple jeu d'échecs.
- **Le niveaux M1** : Ce niveau est composé du modèle représentant le système réel à modéliser.

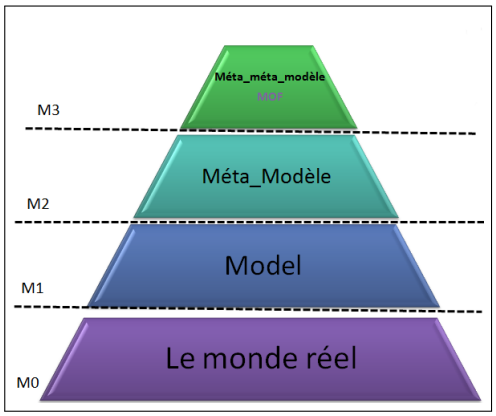


FIGURE 1.2 – Les quatre niveaux d'abstraction pour MDA

Exemple : modèle de jeu d'échecs via un DSL (Domain Specific Langage)

— **Le niveaux M2** :Dans ce niveau on trouve le méta-modèle décrivant le langage de modélisation.
Exemple : méta-modèle du DSL de jeu d'échecs.

— **Le niveaux M3** : Un niveau qui comporte le méta-méta-modèle décrivant le langage de méta-modélisation.
Exemple : Ecore, MOF.

1.2.3 Transformation des modèle

Une transformation est le traitement qui prend en entrés un modèle source(Ms) conforme à un méta modèle source(MMs) et produit un modèles cible(Mc) en sortant, qui se conforme à un méta modèle cible(MMc) voir figure 1.3 $M_s \rightarrow M_c$ selon une définition faite par le développeur. Ces définitions de transformation son des règles de transformation qui décrivent comment un modèle source peut se transformer en un modèle cible.

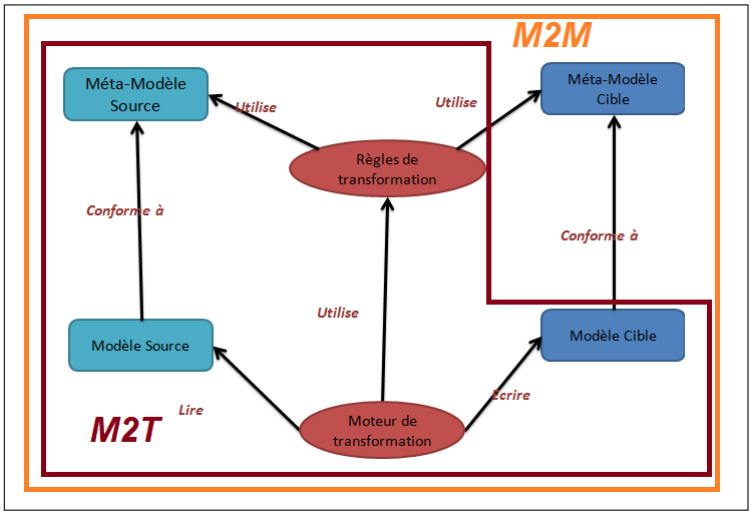


FIGURE 1.3 – Processus de transformation

on a deux Catégories de transformations de modèles :

1. Model-to-Model : est la transformation du modèle spécifique au modèle indépendant, c'est une transformation de syntaxe abstraite à une autre syntaxe abstraite. La technique de transformation est basé sur des langages de transformation tel que ATL,QTV(Query-View-Transformation) Operation et QTV Relation.
2. Model-to-text : cette transformation permet la génération automatique du code d'un modèle spécifique(PSM), c'est une transformation de syntaxe abstraite à une syntaxe concrète. il existe des techniques de transformation parmi les quelle ceux qui sont basés sur des templates tels que JET, Acceleo et Xpand.

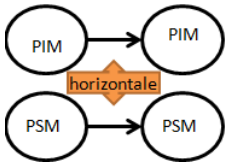
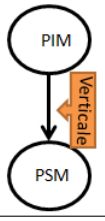
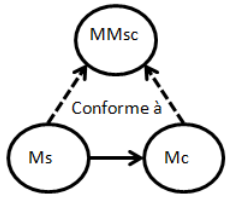
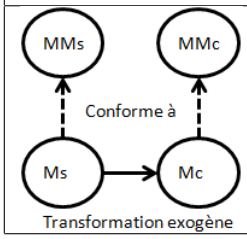
Et on a aussi des types de transformations tels que :

1. Endogène ou Exogène :
   — Endogène : lorsque le modèle source et le modèle cible conforme à un même méta-modèle(MMs = MMc)exemple : modèle UML en modèle UML.

— Exogène : lorsque le modèle source et le modèle cible conforme à des méta-modèle différents (MMs≠ MMc) exemple : modèle UML en modèle Java

2. verticale ou horizontal
   — Verticale : si le modèle source et le modèle cible sont définis à différents niveaux d'abstraction, et on dit raffinement. transformation peut minimiser les niveaux d'abstraction (PIM vers PSM) comme elle peut augmenter le niveaux d'abstraction (PSM vers PIM).
   — Horizontal : cette transformation permet la modification de la représentation et serve le niveau d'abstraction(PIM vers PIM ou PSM vers PSM)cette transformation peut être un ajout ou une suppression...

Ces différentes classes de transformation sont reprises dans le tableau :

		
	Restructuration Normalisation	Raffinement
	Migration de logicielle	Génération

1.3 Business Process Management

BPM(Business Process Management) [Palmer, 2014]est un ensemble de service et des outils qui prennent en charge la gestion des processus métier (par exemple, l'analyse des processus, la définition, le traitement, la surveillance et l'administration), a été défini en 2000 par David McCoy, ce dernier est basé sur l'idée de conciliation entre les objectifs de l'organisation et les besoins de client.

BPM est une unification de plusieurs disciplines parmi ces derniers, la modélisation de processus, la simulation, workflow et l'intégration d'application d'entreprise (EAI)[Owen and Raj, 2003], il permet de gérer et simuler les activités des processus métier dans les entreprises et voir leur interaction pour les optimiser et les automatiser autant que possible. chaque projet de BPM doit passé par un cycle de vie voir la figure 1.4

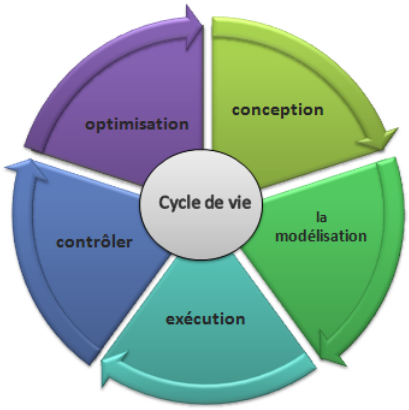


FIGURE 1.4 – Cycle de vie BPM

Les phases de cycle de vie de BPM :

1. La conception : on choisit la procédure ensuite on détermine les méthodes de travail, les acteurs et les personnes qui influentes dans cette mesure.
2. Modélisation : pour modéliser le processus il existe plusieurs approches parmi lesquelles UML(Unified Modeling Language), BPMN(business process management notation) et SLA(Service Level Agreements).
3. L'exécution : est souvent automatique mais parfois elle nécessite une aide humaine, dans cette phase l'utilisation des langages comme JAVA et NET sont satisfaisantes mais, ils sont couteux et la modification est difficile, ces problèmes ont conduit à l'apparition du langage de l'exécution automatique BPEL(Business Process Execution language)
4. Contrôler : vérifier si ce travail est convenable avec la conception, il y a des solutions techniques qui facilitent cette mission comme BAM(Business Activity Motitoring).
5. Optimisation :si on trouve des problèmes on optimise le processus.

1.3.1 Business Process Execution Language

BPEL(Business Process Execution language)[Naoufel, 2009] Standard de l'OASIS est un langage de balisage permettant la définition d'une collaboration entre les services web pour l'implémentation des processus métier. il a réussi lorsque le monde a passé à l'architecture orientée services (SOA).
BPEL est une fusion des deux langages WSFL(Web Services Flow Language) développés par IBM et XLANG de Microsoft[Naoufel, 2009]. L'objectif de cette langue est de permettre aux développeurs de combiner différents services Web ou d'autres composants pour obtenir des systèmes complexes. il y une possibilité de transformer un diagramme BPMN vers BPEL pour Attendre cet objectif.

1.3.2 Business Process Management Notation

BPMN(Business Process Management Notation)c'est une nouvelle standard pour la modélisation des processus métier, qui fournissent une notation graphique pour la spécification des processus dans un diagramme de processus métiers (BPD). Il est proposé en 2004 par BPMI(business process management Initiative), ensuite en 2008 il est repris par OMG(Object Management Group)après le fusionnement entre eux. L'objectif de BPMN est de fournir une notation standard facile et compréhensible, mais aussi pour modéliser les processus métier complexe. Les éléments de BPMN ou appelés aussi des pictogrammes, organisés en quatre catégories [BAUCHE, 2011], objet de flux, objet de connections, swimlane et les artefacts voir figure1.5, et aussi BPMN définit l'organisation de ces éléments soit par orchestration(qui se passe à l'intérieur de processus)soit par chorégraphies(les communications entre processus).

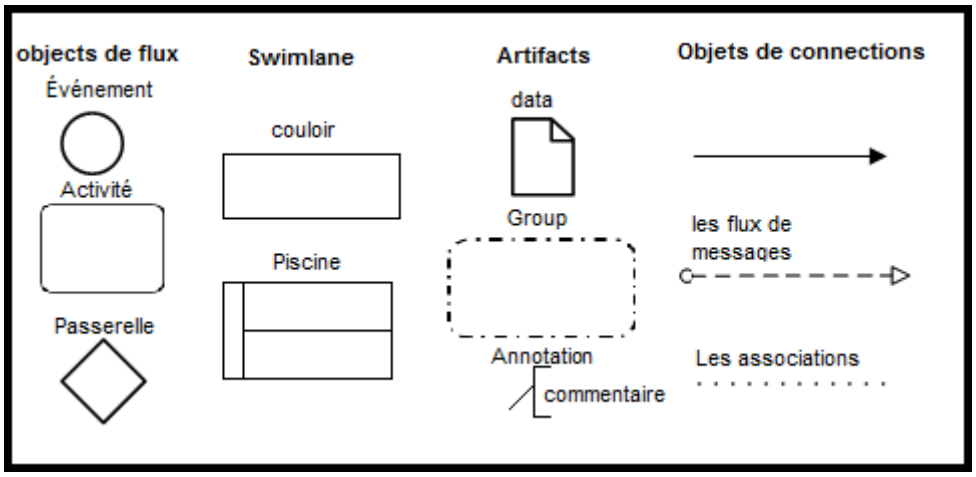


FIGURE 1.5 – Les pictogrammes de BPMN

1. **Object de flux**

   — **Événement**
   Le concept événement indispensable dans la notation BPMN, il identifie un état dans le processus, il existe trois catégories d'événements(départ ,intermédiaire, fin). Dans la modélisation d'un processus il est obligatoire de désigner au moins deux événements, l'événement de départ et l'événement de fin.
   Des événements de départ sont des événements qui déclenchent l'exécution de processus, dans quelque cas on peut trouver un ou plusieurs événements de départ pour déclencher un seul processus.
   Quand les processus deviennent plus complexes on aura besoin des événements plus complexes, tel que les messages, le temps, les règles des affaires, et en plus les conditions d'erreur. BPMN présente une collection des pictogrammes qui satisfaisaient la nécessiter voir tableau1.1.

   — **Activité** Une activité décrit le travail que l'entreprise effectue, et elle est présentée par un rectangle. Une activité prend normalement quelque temps pour l'exécution et ça impliquera le besoin d'un ou de plusieurs ressources(des entrées et des sortis).
   Les activités peuvent être des sous-processus(subprocess) ou des tâches(task), ce dernier type est atomic et n'inclue pas des détails internes et le sous-processus à une grande granularité et présenté par un rectangle avec un signe $\ll + \gg$ masquer les détails voir 1.6.

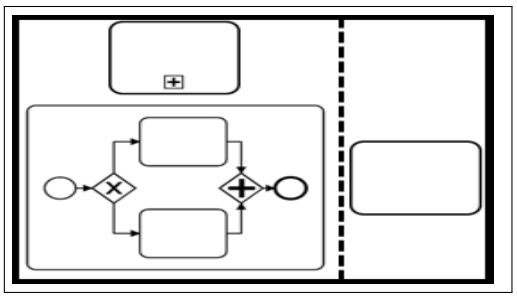


FIGURE 1.6 – Tache et sous processus

Il y des symboles qui indiquent le comportement de l'activité parmi ceux-ci le symbole de sous processus, la boucle(loop), AdHoc une exécution libre permet d'exécuter on ordre aléatoire et d'instance Multiple qui indique l'exécution d'une activité plusieurs fois en parallèle voir 1.7. et on peut combiner ces symboles dans la même activité mais il faut respecter certains réglés, il

cause	Capturer (catch)	Lancer (throw)	Pictogramme			
			Départ	intermédiaire		Fin
Basic	Événement simple aucun indication sur le type.					
Message	Recevoir des messages à partir d'un participant à travers les frontières d'un processus	Envoyer un message à l'un des participants à travers la frontière				
Temps	Utilisé dans une condition temporelle spécifique (par exemple : chaque lundi à 9 :00h faire...)					
Erreur	Un événement d'erreur est utilise pour définir une erreur reçu	l'envoie d'un rapport d'erreur	j		j	
Cancel	Les événements d'annulation utilises comme événements de frontière pour le sous-processus, et déclencher le rôle à retourner de la transaction.	Annulation une activité de type transaction				
Signal	Recevoir un signal énoncé par participant	Envoyer un signal				
Condition	commencer uniquement si certaines conditions est vrai					
Multiple	Plusieurs causes sont possible (une seule cause suffi)	Plusieurs résultats sont attendus (par exemple : envoyer plusieurs message)				
Parallèle	Plusieurs causes sont possible (si seulement si tous ces types sont satisfaits).					
Fin		Arrêt immédiat de toutes les activités du processus				

TABLE 1.1 – Liste des événements.

pictogramme	Description des activites
	tâche simple
	**tâche service** un service web ou application automatisée.
	**tâche d'envoi** envoyer un message à un participant externe(par rapport au processus)
	**tâche de réception :**attendre qu'un message arrive d'un participant externe (par rapport au processus)
	**tâche utilisateur :** une personne effectue la tâche à l'aide d'une application
	**tâche manuelle :**effectuée manuellement sans aide
	**Une tâche de règles métier** Envoie une entrée dans un moteur de règles de gestion et reçoit le résultat du calcul effectué par ce moteur.
	**tâche Script :**est exécutée par un moteur de processus

TABLE 1.2 – Liste des tâches.

faut que l'activité contient de 0 à 3 symboles, ne pas attacher le symbole AdHoc avec une tâche et une activité contenant une boucle ou une instance Multiple[Naoufel, 2009][White, 2008].

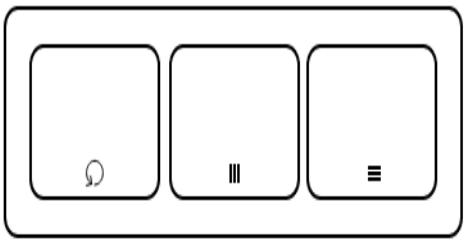


FIGURE 1.7 – a boucle b :instance multiple c : AdHoc

Et de plus à coté de ces type l'activité peut contenir d'autres types voir tableaux 1.2.
Il y une possibilité de découvrir un événement attaché à une activité est ici apparue une nouvelle notion de l'événement interrompt ou n'interrompe pas l'activité.

— **Passerelle** Les activités et les événements ne sont pas suffisants pour modéliser complètement un processus, il faut formaliser ce qu'on appelle le chemin parfait c'est à dire il faut découvrir tous les cas possibles, par exemple qui est

ce qui fait la machine à café dans le cas d'accepter ou refuser la demande ? C'est pour cette raison qu'on modélise les passerelles.
La passerelle(gateway) est présenter par un losange qui permet de contrôler le flux d'orchestration contenant deux ou plusieurs voix converges ou divergés selon une stratégie prédéfini. il existe différents type de passerelles voir le tableau1.3.

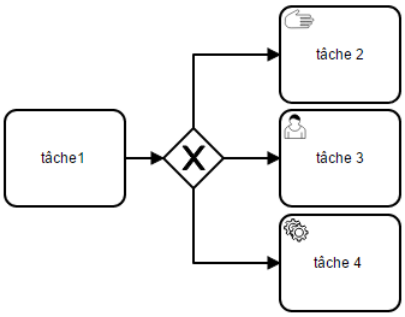
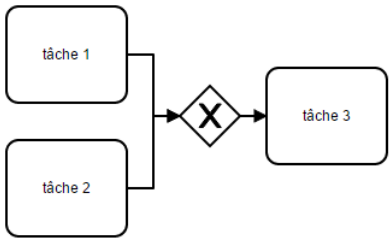
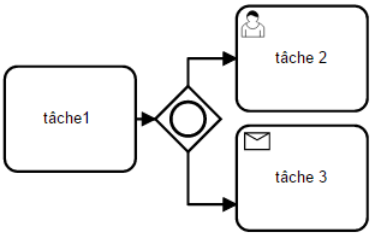
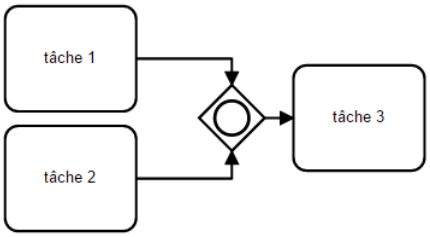
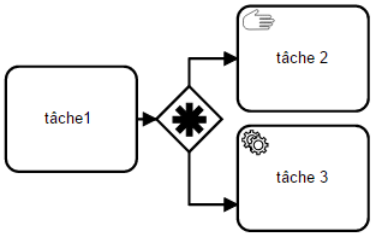
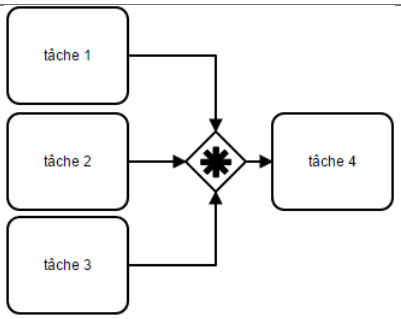
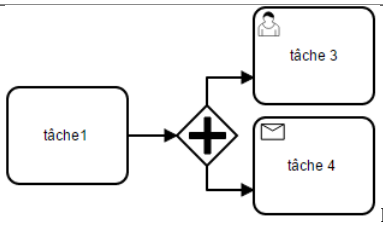
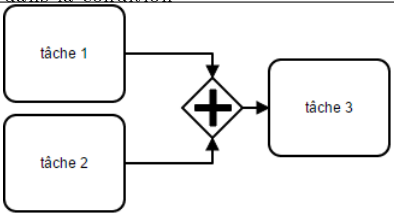
le passerelle	une décision	une fusion
Exclusive(XOR) :le passerelle par défaut	 permet de diviser le flux vers plusieurs chemins exclusives on fonction d'une condition basé sur une information	 permet de reconstituer un flux unique
Inclusive(OR)	 active une ou plusieurs branches tant-que la condition vérifiable et il faut spécifier le flux par défaut	 continue lorsque les premiers signaux d'entrée arrivent
Complexe basé sur condition précise par commentaire	 divisé le flux par rapport à la condition	 attend l'arrivant de tous les flux qui illustrer dans la condition
parallèle(AND)	 permet de Continuer le flux sur toutes les branches sortantes simultanément.	 attend que toutes les branches entrantes se terminent.

TABLE 1.3 – Liste des passerelles

2. **Objets de relation**

Les objets de relation reliés deux objets de BPMN digramme et il y a trois types de flux :

— **Flux de séquences** : qui se représente par une ligne plaine avec une flèche solide, et il est utilisé pour montrer l'ordre de l'exécution entre les événements, les activités et les passerelles.

— **Flux de messages :**il représente un envoi de message d'une entité à une autre, il connecte une piscine ou un objet dans la piscine à une autre piscine ou objet dans cette piscine et ne pas être utilisée pour connecter des activités ou des événements dans la même piscine .

— **Association :**est représenté par une ligne discontinu. Il est utilisée pour associer un artefact à un objet de flux. Aucune direction est utilisée lorsque l'artefact est associé à un flux de séquence(parce que le flux de séquence est déjà diriger).

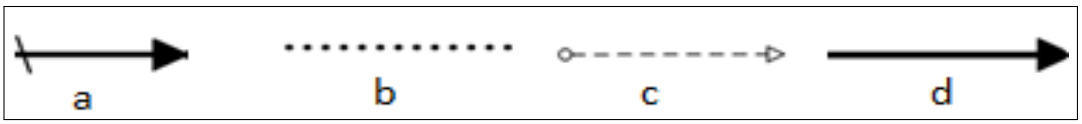


FIGURE 1.8 – (a) séquences par défaut,(b)Association ,(c)messages,(d) séquences

3. **Les Swimlanes :**

Les swimelane organisent et regroupent les éléments BPMN, ce moyen émerge la notion de collaboration entre les processus. ils se composent de deux types voir la figure 1.9.

— **piscine :**représente soit un processus, entité du métier ou le rôle l'interagissement entre ces processus. Une piscine peut contient plusieurs couloirs.

— **couloir :**une sous-ensemble dans une piscine, utiliser pour organiser et classer les activités selon leurs fonctions.

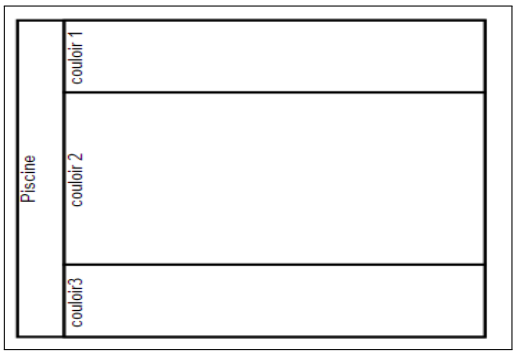


FIGURE 1.9 – Piscine et couloir

Pour montrer les échanges qui faites entre deux piscines on utilise des flux de messages, et entre les couloirs il suffit d'utiliser les flux de séquence.

4. **Artefacts**

Les artefacts sont utilisés pour ajouter plus d'information au diagramme ou pour effacer l'ambiguïté dans le diagramme.
Les artifices sont :

— **objets des données(Data) :**utilisés pour expliquer quelles données sont nécessaires dans le diagramme.
— **les groupes :** utilisés pour des activités de groupes différents, sans affecter le flux dans le diagramme.
— **annotations :** utilisées pour donner plus d'informations et des commentaires sur le diagramme.

1.3.3 Les différents diagrammes BPMN

1. **Diagramme de collaboration** :Représente les interactions entre deux ou plusieurs unités d'affaires représentées par des(Pool). Les pool sont définis comme étant les participants de cette collaboration.
2. **Le diagramme de conversation** : Est la description informelle d'un Diagramme de collaboration à haut niveau.
3. **Diagramme de chorégraphie** :est un type étendu de diagramme de collaboration BPMN, est utilisé pour analyser la façon dont les participants échangent des informations afin de coordonner leurs interactions.

1.4 Le langage formel DD-LOTOS

LOTOS c'est une langage formelle, normalisé par ISO(International Organization for Standardization) en 1988. Il repose sur deux concepts principaux : la premiére est la déclaration de types abstraits de données (ADTs) avec langage algébrique ActOne et l'ordonnancement temporel avec CCS(Calculus of communicating systems) et CSP(Communicating sequential processes).[Garavel et al., 2013]

DD-LOTOS [Messaoud, 2012] c'est une extension de D-Lotos(Duration LOTOS) supporte la distribution et la communication entre les localités, une localité est un environnement d'activité de calcul, et elle est nommée et structurée selon une structure plate, cette dernière est une manière de programmer les localités explicitement voir la figure 1.10.

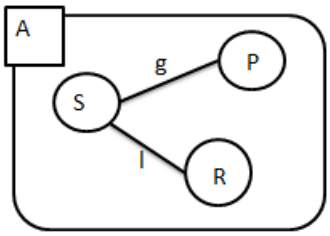


FIGURE 1.10 – Localité

P, R et **S**sont des processus qui résident dans la localité **A** et peuvent communiquer entre eux par des canaux de communication **l** et **g**. la notion de la distribution est incarné dans une collection des localités indépendantes voir la figure 1.11, cette dernière est présente trois localités **A, B** et **C** communiquent entre eux par des canaux de communication **k et d**, selon le principe d'échange de message. Cette communication est à distance et asynchrone (non-bloquante).

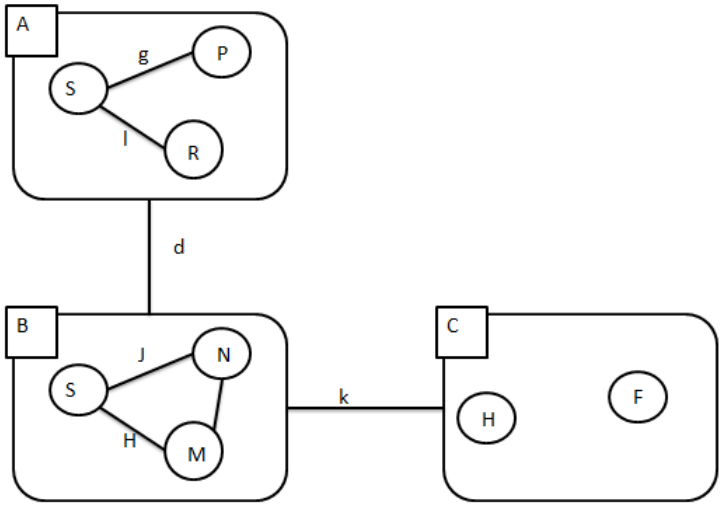


FIGURE 1.11 – Système distribué DD-LOTOS

1.4.1 Syntaxe de DD-LOTOS

La syntaxe DD-Lotos est défini comme suit :

$E ::=$	stop |exit | Δ^d | X[L]|	**Comportements**
	g@t[SP] ;E | i@td ;E | hide L in E |	
	E[]E | E|[L]|E | E $\gg$ E | E[>E |	
	$\mid a!v\{d\}; E$	**Envoyer un message**
	$\mid a?x; E$	**Recevoir un message**
$S ::=$	**Systèmes**	
	ϕ	**Vide**
	$\mid S \mid S$	**Composition**
	$\mid l(E)$	**Activité *E* dans la localité** l

1.4.2 Sémantique opérationnelle structurée

[Messaoud, 2012]

La sémantique opérationnelle des comportements est donnée par la sémantique opérationnelle de D-LOTOS. Cette sémantique est étendue à DD-LOTOS en donnant les règles sémantiques pour les systèmes communicants comme suit :

— **Processus** [Messaoud, 2012]

$a!v\{d\};E$: Prenons la configuration ${}_M[a!v\{d\};E]$, l'émission du message v commence une fois que toutes les actions indexées par l'ensemble M ont terminé leur exécution, ce qui est conditionné par le prédicats $Wait(M)$ qui doit être égal à $false$ dans la règle 1. Les règles 2 et 3 expriment le fait que la prise en compte de l'écoulement du temps dans $a!v\{d\};E$ commence uniquement si toutes les actions référencées par M terminent leur exécution. La règle 4 impose que l'occurrence de l'action d'envoi ait lieu dans la période d, dans le cas contraire le processus est transformé en *Stop*.

1. $\dfrac{\neg Wait(M)}{{}_M[a!v\{d\};E] \xrightarrow{M^{a!v}x} {}_{\{x:a!v:t\}}[E]} \qquad x = get(\mathcal{M})$
2. $\dfrac{Wait(M^{d'}) \text{ or } (\neg Wait(M^{d'}) \text{ and } \forall \varepsilon{>}0.\ Wait(M^{d'-\varepsilon})) \qquad d'{>}0}{{}_M[a!v\{d\};E] \xrightarrow{d'} {}_{M^{d'}}[a!v\{d\};E]}$
3. $\dfrac{\neg Wait(M)}{{}_M[a!v\{d'+d\};E] \xrightarrow{d} {}_M[a!v\{d'\};E]}$
4. $\dfrac{\neg Wait(M) \text{ and } d'{>}d}{{}_M[a!v\{d\};E] \xrightarrow{d'} {}_M[stop]}$

Processus $a?x;E$: Prenons la configuration ${}_M[a?x;E]$, la règle suivante exprime que la réception commence une fois que les actions indexées par l'ensemble M terminent leur exécution.

$$\frac{\neg Wait(M)}{{}_M[a?x;E] \xrightarrow{M^{a?x}y} {}_{\{y:a?x:0\}}[E]}$$

— **Communication à distance**[Messaoud, 2012]
Les activités distribuées échangent les messages entre eux, l'expression de comportement $l(a!v\{d\})$, exprime que le message v est offert pour une durée d unités de temps, cette activité évolue dans la localité l, ce message doit être envoyé via le canal a. De l'autre côté, $k(a?xE)$ spécifie que l'activité E évolue dans la localité k, est prête à recevoir un message via le canal a. La règle suivante définit la communication à distance entre les deux activités distribuées via le canal a. Dans ce cas, la communication est traduite par une évolution interne (action silencieuse τ) :

$$\frac{-}{{}_M[l(a!v\{d\};E1)] \mid {}_{M'}[k(a?xE2)] \xrightarrow{\tau} {}_M[l(E1)] \mid {}_{M'}[k(E2\{v/x\})]}$$

— **L'évolution temporelle du système**[Messaoud, 2012]

$$\frac{E \xrightarrow{d} E'}{l(E) \xrightarrow{d} l(E')}$$

$$\frac{S_1 \xrightarrow{d} S_1' \qquad S_2 \xrightarrow{d} S_2'}{S_1 \mid S_2 \xrightarrow{d} S_1' \mid \ S_2'}$$

Le temps agit sur tout le système.

1.5 Conclusion

Dans ce chapitre nous avons présenté la démarche de MDE et on a focalisé notre travail sur les différents types de la transformation telle que M2T|M2M ou endegène|exogène etc. Ensuit nous avons parler de BPM et on à présenté les concepts principaux BPEL et BPMN et on s'est détaillé dans le concept BPMN-collaboratif, et enfin nous avons donné une petite introduction sur le langage formel DD-LOTOS.

Chapitre 2

Les Outils

2.1 Introduction

Dans ce chapitre nous allons présenter une collection d'outils qui nous aide dans le processus de la transformation, commençons par les outils de modélisation des méta-modèles tel-que MOF(Meta Object Facility) et EMF(Eclipse Modeling Framework), Ensuite, les outils de création des éditeurs graphic nous allons présenter Sirius et GMF(Graphical Modeling Framework). Enfin, les outils de transformation qui sont basé sur les templates Xpand Acceleo.

2.2 Outils de création des méta-modèles

2.2.1 Meta Object Facility(MOF)

C'est un modèle standard de l'OMG existe depuis 1997, est une framwork faite pour définir, manipuler et intégrer des méta-données d'une manière indépendante dans la plate-forme. Il devient un pilier de l'architecture MDA, et un sous-ensemble d'UML.

2.2.2 Eclipse Modeling Framework(EMF) :

EMF est un framework d'eclipse développé par IBM pour la modélisation et la génération de codes pour faciliter le construction des applications, et aussi a été conçu pour emmener l'Eclipse au monde de développement dirigé par les modèles, et il se compose de trois éléments fondamentaux éclairés dans la figure 2.1

- **Ecore (EMF) :** framework contient un méta modèle pour décrire un modèle.
- **L'éditeur EMF.Edit :** framework contient des classes génériques réutilisables pour construire un éditeur d'un modèle

— **Le modèle de génération EMF.Codegen :**Capable de générer tout ce qui construit un éditeur complet pour un modèle EMF.

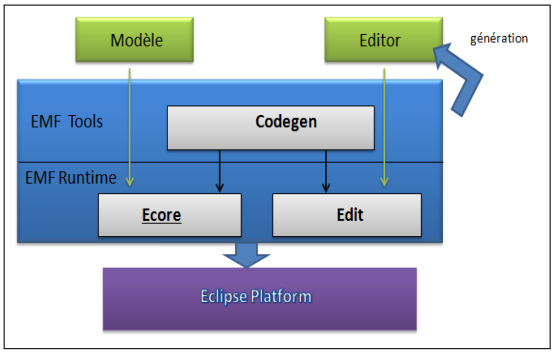


FIGURE 2.1 – L'architecture EMF

1. **La modélisation avec EMF :Modèle Ecore**

   Eclass : Signifie des classes d'un modèle contient un nom, peut être une interface ou une abstrait et peut aussi contenir des StructuralFeatures (attributs ou références), et il supporte l'héritage.
   EAttribute : Identifier par un nom, un type, et une valeur min et valeur max.
   EReference : Association entre deux classes.
   EDataType : Type primitif ou type objet défini par java.
   EPackage : Contient un ensemble de classe et des attributs,il est défini par un nom et URI(Uniform Resource Identifier) pour l'identification lors de la sérialisation.
   EOperation : Les opérations d'une classe identifiée par un nom, un type de retour et des paramètres.

2. **Exemple**

   L'exemple réseaux de Petri[Nadia, 2012] modifier , possède trois EClasse, trois relations(Composition) et chaque EClasse contient des attributs simples voir figure 2.2.

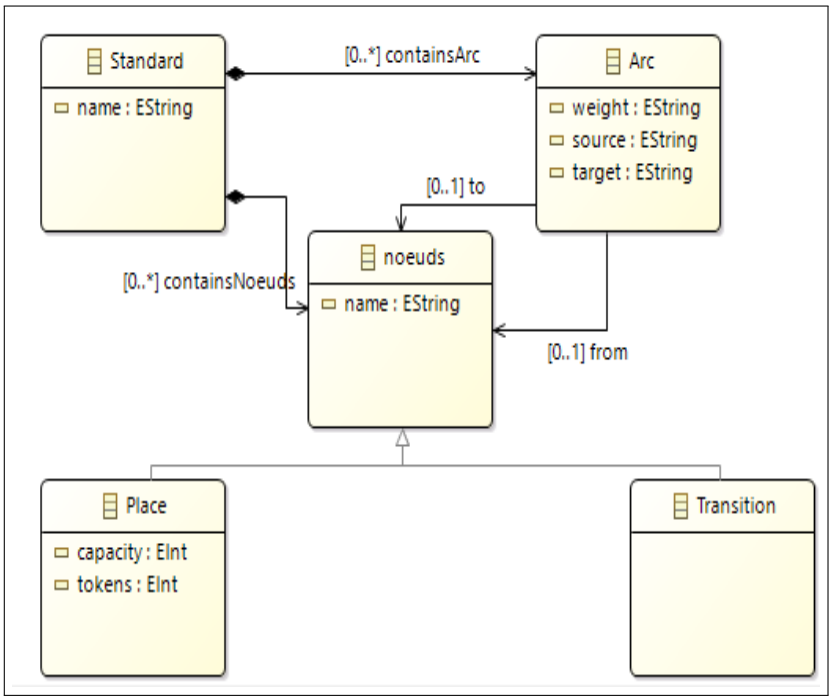


FIGURE 2.2 – Méta-modèle des réseaux de Petri

2.3 Outils de création d'éditeurs

2.3.1 Graphical Modeling Framework(GMF)

GMF(Graphical Modeling Framework)[24,] est un framworke sous eclipse basé sur **EMF**(Eclipse Modeling Framework) et **GEF**(Graphical Editing Framework), il est développé dans le but de créer un éditeur graphique conforme à un modèle.
Pour guider l'utilisateur vers la génération de son modèle, GMF fournit un tableau de bord voir figure 2.3. On voit dans l'image que tout commence à partir du domain model(méta-model)on peut dériver trois autres modèles :

1. **Domain Gen Model :** Ce modèle décrit la génération de code.
2. **Graphical Def Model :** Ce modèle décrit les éléments graphiques que vous trouverez dans votre diagramme.
3. **Tooling Def Model :** Ce modèle décrit la palette et les outils qui serons utilisés pour la création des éléments du diagramme.

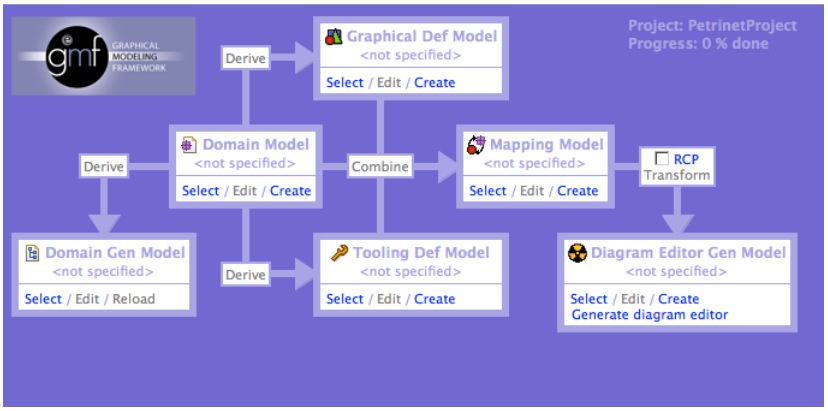


FIGURE 2.3 – Table de bord fournit par GMF

Après la dérivation de ces modèles on les combinent pour obtenir Mapping Model, en réalité les trois modèles sont complètement orthogonaux et Mapping Model reliés entre eux. Par exemple dans les réseaux de Petri on dit : Une ellipse dans Graphical Model est la représentation d'une instance d'une place(class dans domain model) créer à l'aide d'un outil de création décrit dans Tooling Def Model.

2.3.2 Sirius

Sirius est un framwork dans eclipse développé par la société **Obeo** partenariat avec **Thales**[BERNARD, 2014] en 2007 et en 2013 relâche comme open source. Il est développé à base de EMF et GMF pour créer facilement et rapidement des éditeurs graphiques à partir d'une représentation DSL au sein d'un modèle EMF sans avoir écrire le code.

La caractérisation de Sirius est la création d'un éditeur en parallèle avec la spécification et après il peut rendre l'observation en temps réel à partir de l'instanciation du modèle qu'on veut représenter. Pour lancer l'exécution de project de type basé sur le méta modèle créer, en fait (Run > Run configurations), puis créer une nouvelle configuration de type(Eclipse Application) on leur donne un nom et on ajoute dans la zone(VM arguments) de l'angle(Argument) le code qui apparu dans la figure2.4 et appuyer sur finish. Puis quand on lance l'application avec la nouvelle instanciation du modèle, apparu une nouvelle platforme eclipse se base sur le méta-modèle que nous avons créer.
un unique modèle nommé VSM permet de spécifier les représentations graphiques d'éléments sémantiques et de concevoir les fonctionnalités de l'éditeur à l'aide d'outils. Le

project Siruis est composé d'un ensemble de points de vue(Viewpoint), le point de vue contient des représentations(diagram, sequence diagram,table, tree) ces représentations contiennent des Nœuds, Edge et des pallètes. Un Nœud est une EClass composée d'une représentation(un schéma| une image) et un pré-condition, le Edge est une Ereferance ou une EClass, une pallete contient la création des nœuds et Edge. Les autres étapes pour créer un éditeur sont montrées dans l'exemple.

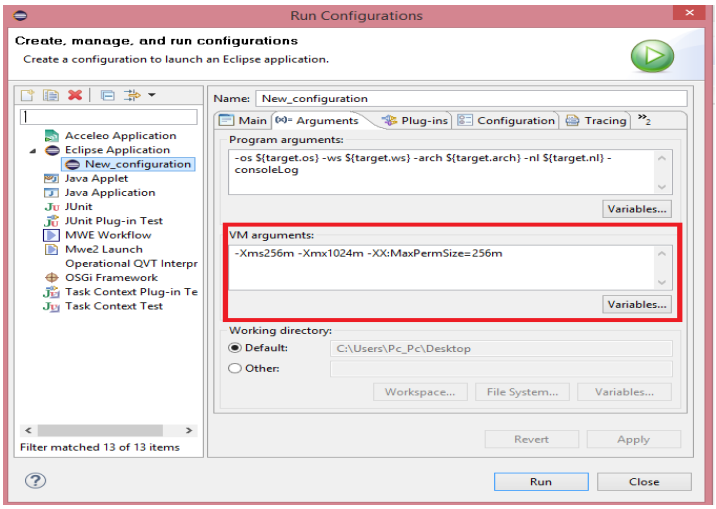


FIGURE 2.4 – L'instanciation du modèle

— **Exemple**

L'exemple qu'on va traiter est l'exemple de réseau de Petri qu'on a déjà identifié son modèle EMF dans la section précédente voir figure2.2 pour cela on fait :

1. Créer un projet(Modeling project) sous le nom de **Reseau-petri** dans la nouvelle plateforme. Créer ensuite une instance du modèle (new->Exemple EMF Model Creation Wizards)on donne un nom à notre exemple, puis on sélectionne l'objet racine **Standard**.
2. Créer un project(new->Sirius->Viewpoint Specification Model) pour construire notre éditeur.
   et ensuite on crée Viewpoint(1), et on remplit les champs dans propriété(2), on crée une représentation de type digramme(3) et remplit les champs de propriété(4) voir figure 2.5

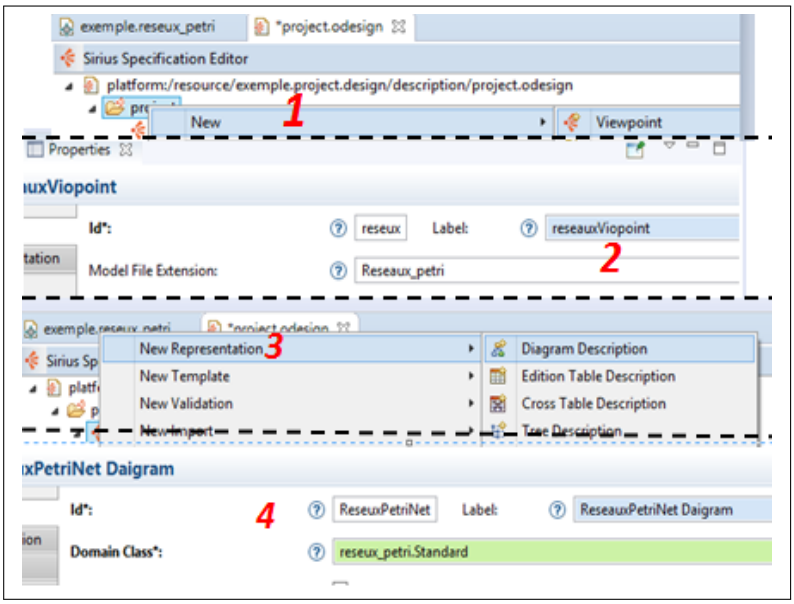


FIGURE 2.5 – Étape pour créer viewpoint

3. Pour spécifier les éléments il faut spécifier un Layer qui permettra d'afficher ou non certains éléments, pour le construire on fait un clic droit sur **ReseauxPetri Daigram** et on sélectionne(New Diagram Element > Default Layer Layer) et on donne à cet élément un ID et Label.

   — **spécification des nœuds :** Dans notre exemple on a deux nœuds(Place et Transition) pour réaliser un nœud on fait un clic sur layer qu'on a déjà crée(New Diagram Element->Node) et dans le champ classe domain on spécifie la classes de nœud voir figure2.6. Le champs Semantic Condidates Expression est optionnel mais cette syntaxe correspond au langage Acceleo qui nous aide dans l'étape suivante pour la transformation. Il faut aussi définir la représentation du nœud par un clic sur le nœud et spécifier le style pour plus des options comme la couleur le size etc, accéder à Properties.
   — **Spécification des relations :** Dans notre exemple on a une relation(Arc) donc on fait un clic droit sur layer(New Diagram Element->Element Based Edge), puis remplir les champs voir 2.7.

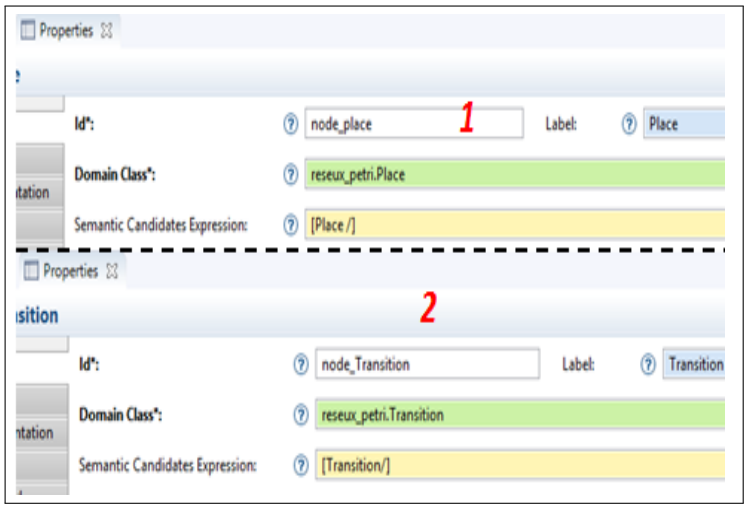


FIGURE 2.6 – Spécification nœuds

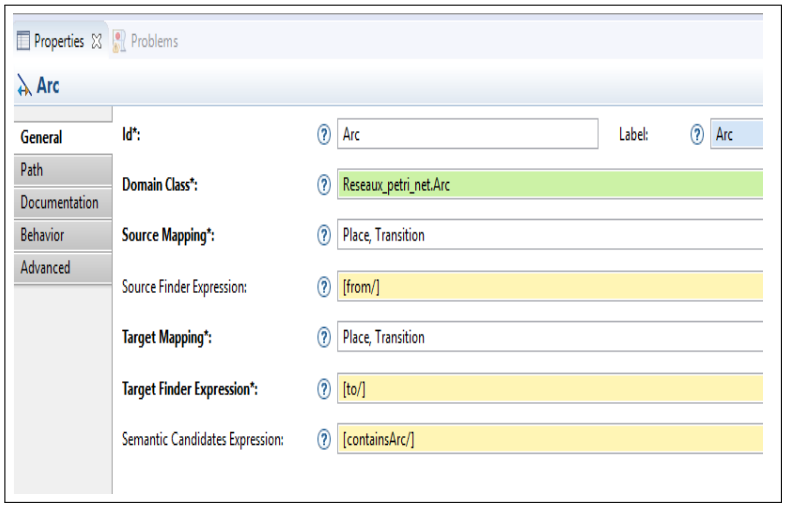


FIGURE 2.7 – Spécification de relation

4. **Création de la palette :** C'est la plus importantes dans l'éditeur. Dans le même project on fait un clic droit sur Layer(New Tool->section) et comme d'habitude, on donne à cet élément un ID et un label.
   — **Création des nœuds :** Clic droit sur section(New Element Creation->Node creation) et on définit le label, ID et quel est nœud qui va créer l'outil. L'étape la plus importante c'est indiquer à Sirius comment instancier l'élément et on le place où dans le modèle. C'est pour ça on clic a droite sur Begin(New Operation-> Change Context) et on définit le champs d'expression par[container /], ensuite créer une opération de type Instance, puis définir les opérations de type Set Value pour définir les attributs du nœud

par exemple attribuer name de la classe Place.

5. En fin, pour visualiser l'éditeur en temps réel, on fait un clic droit sur le projet(qui contient le modèle) et on choisit Viewpoints Selection, puis on fait une autre clic droite sur Standard dans notre exemple, puis on sélectionne(New Representation-> new Reseaux_petri daigrame)l'éditeur apparu voir figure2.8 .

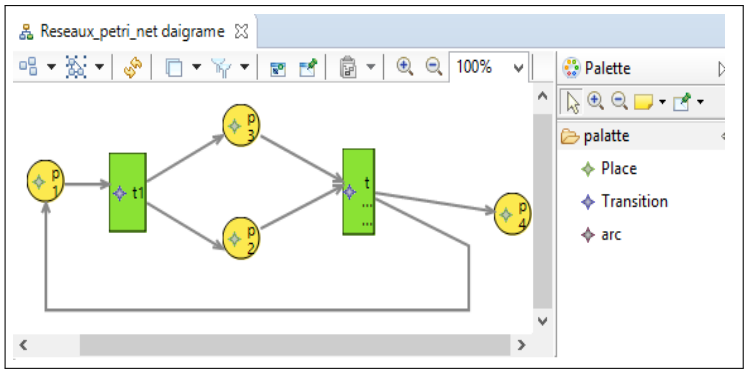


FIGURE 2.8 – Éditeur à base de notre DSL(réseaux de Perti)

2.3.3 Une comparaison GMF et Sirius

GMF et Sirius son tous les deux ont pour objectif de créer des éditeurs graphiques, pour créer l'éditeur dans GMF il faut passé par plusieurs étapes par rapport à Sirius. Le project GMF contient un buge, quand on combine les trois modèles voir figure2.3, il faut toujours vérifier le Mapping Model parce que les composants ne se relient pas avec leur outil correcte. Une autre difficulté qui nous rencontre quand on ajoute ou on modifie un composant graphique, il faut recommencer dès le début, imaginer que notre project est volumineux. Par contre dans Sirius toutes ces opérations son faite facilement et en temps réels.

2.4 Outils de transformations

2.4.1 Xpand

Xpand [12]c'est un langage de transformation à l'origine est une parti de project **O**pen **A**rchitecture **W**are platform (oAW), utilise les templates pour contrôler la génération de production et ces templates sont stockées dans des fichiers de l'extension .xpt.

Il comporte quelques propriétés très uniques telles que la sureté de typage (safty type) est un principe permettant d'améliorer la qualité dans la programmation et le polymorphisme dynamique(daynamic polymorphisme)c-à-d la décision sur la méthode à exécuter est faite au moment de l'exécution du code.

Pour générer un code avec Xpand il faut d'abord créer un project Xpand à partir du menu(File-> Xpand Project->Generate a simple EMF based Xpand project).

- **Expressions** :la syntaxe des expressions est mixte de Java et OCL .
- **Ckek** : contient une liste des contraintes que le modèle doit accomplir pour être correct. Et spécifie le type d'action qui sera prise en cas où la contrainte échouerai, il y ERROR(le message indiqué sera imprimé et tout le nouveau traitement s'arrête) et WARNING(le message indiqué et imprimée, mais l'exécution de workflow n'est pas arrêtée)
- **generator.mwe** : Le moteur de workflow de modélisation (MWE)est un moteur générateur configurable déclaratif. Il fournit un langage de configuration simple basé sur XML. Generator workflowse contient un certain nombre de composants de flux de travail qui sont exécutés séquentielle-ment dans une JVM unique.

2.4.2 Acceleo

Acceleo a été développé par la société française Obeo depuis 2009[10], est un langage de transformation qui implémente le standard MOFM2T (MOF Model To Text Transformation Langage) de l'OMG[11], et utilise un plugin compatible à EMF. Il utilisé l'approche par template.

1. **création d'un project Acceleo**

   Pour crée un project Acceleo ouvrer un nouveaux project, dans la catégorie Model to text Transformation appuyer sur Acceleo Project, choisir un nom et passé a l'autre page. Dans la deuxième page on met les autres informations voir la figure 2.9 .

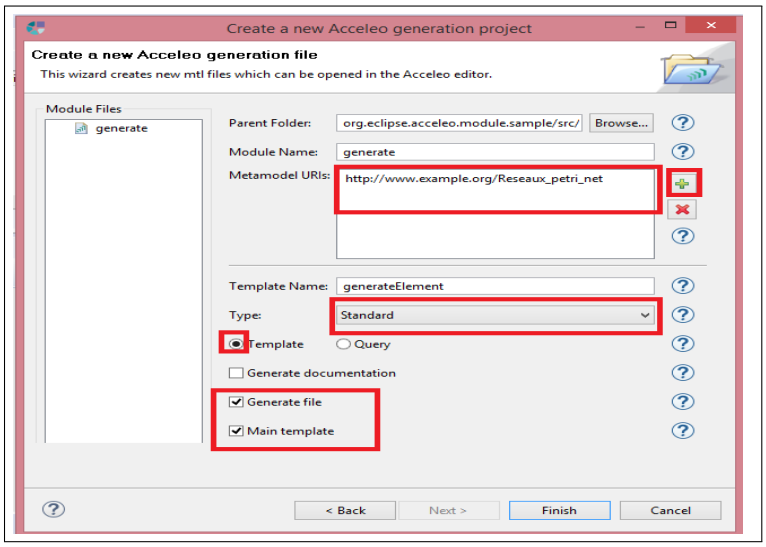


FIGURE 2.9 – Créer un project Acceleo

2. **Syntaxe de Acceleo**

— **Module :**Est un ficher contient des templates et des Queries de l'extension ".mtl", Il contient aussi le méta modèle qui sera utilisé pour générer de code. un module d'Acceleo peut hériter un seul module pour accéder à des éléments publics ou privés, et aussi peut importer d'autre module.

— **Template(patron) :**C'est la structure la plus utilisée dans Acceleo, elle contient le text à générer . la template contient une visibilité(public/protected/private), un nom et des paramètres. paramètre est déclaré comme suite <name> :<type>,le type doit être l'un des suivants : type fourni par le méta-modéle ou type de OCL comme boolean ou OclAny etc.

À l'intérieur d'un template on peut utiliser deux types d'expression pour générer le code, le premier static expression est générer sans aucune transformation (juste un text), et le deuxième utilise les éléments de modèle pour produire le text générer .
Un template peut avoir une pré_condition qui assure que le modèle ne soit exécuter que par la validité, et aussi peut avoir un post_condition qui exécute après l'exécution de template ,une autre caractéristique que temple peut contenir des appels des autres templates .

— **Main Template :** C'est la template principale et le point d'entrer de la génération, elle se caractérise par la présence du "@main".

— **Queries :** Sont utilisés pour encapsuler des expressions complexes .

— **File :** Est utilisé dans le template pour spécifier le ficher de sortie, contient trois paramètres : une expression renvoie le nom de fichier sortant, un boolean qui indique si le ficher existant doit être écrasée ou juste ajouter les modification set le dernier contient l'encodage de fichier.
— **Structures :** Un template peut contenir plusieurs types de structure tel que :**IF** : Est un bloc de code qui ne doit pas être exécuté sauf si une condition donnée est remplie, **For :** et aussi spécifier une condition sur une collection et **Let :** Définie une variable.

Le code 2.1montre la transformation de réseau de Petri de vers PNML en langage Acceleo, on a utilisé deux templates la première est le main template(aStandard), et cette dernière fait appel au deuxième template(Arc).

```
1  [template public reseaux_petri(aStandard : Standard)]
2  [comment @main/]
3  [file (aStandard.name+'pnml', false, 'UTF-8')]
4  <?xml version="1.0">
5  <pnml xmlns="http://www.pnml.org/version-2009/grammer/pnml
       ">
6  <net id="[aStandard.name/]"type=http://www.pnml.org/version
       -2009/grammer/ptnet">
7  [for (a : Arc | containsArc)]
8    [Arc(a)/]
9  [/for]
10 [for (N : noeuds | containsNoeuds)]
11  [if (N.oclIsKindOf(Place))]
12   [let P : Place = N]
13  <place id="[N.name/]"><initialMarking><text><[P.tokens/]</
       text></initialMarking></place>
14   [/let]
15 [else]
16   <transition  id=[N.name/]></transition>
17  [/if][/for]
18 </net></pnml>
19 [/file]
20 [/template]
21 [template public Arc(A : Arc)]
22   <arc id="[A.name/]" source="[for (N1 : noeuds | A.From)][
       N1.name/][/for]
23 " target="[for (N2 : noeuds | A.to)] [N2.name/][/for]"></
       arc>
24 [/template]
```

Listing 2.1 – Template réseau de Petri de vers PNML

3. Exécution d'un project

L'exécution d'un project Acceleo est hors ordinaire, c'est pour sa qu'il faut accéder à la barre d'outils (Run->Run configuration) et créer une nouvelle configuration pour le projet Acceleo. Ensuite il faut sélectionner le project qui contient le ficher Acceleo, et puis on définit la classe main, ensuite on sélectionne le modèle qu'on souhaite générer et enfin on choisit le ficher dont on mettra le modèle

générer voir figure 2.10.

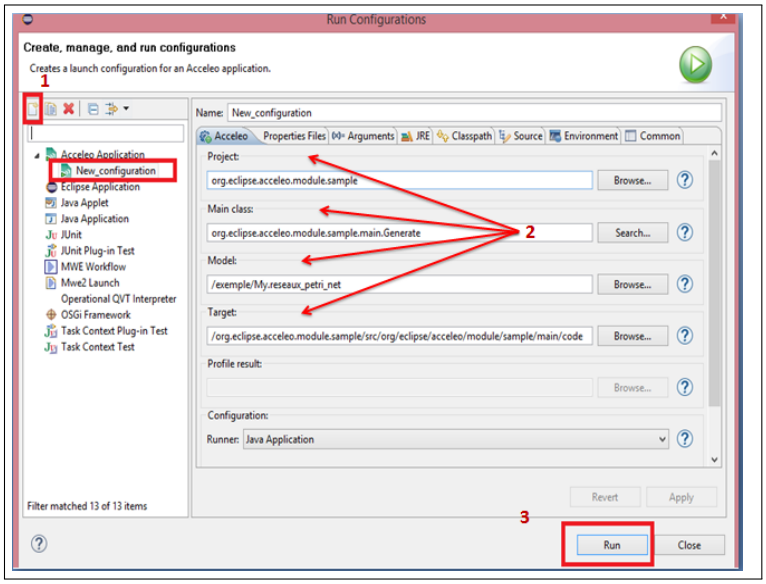


FIGURE 2.10 – Configuration

2.4.3 Une comparaison entre Acceleo et Xpand

Acceleo et Xpand ont le même objectif c'est génération du code à partir d'un modèle, et ils ont utilisé la même approche template, mais Xpand plus de que sa pauvre documentation sa syntaxe, il est ambiguë et il est difficile de travail avec lui. mais,n'est pas impossible, autre inconvénient de Xpand, il accepte que la génération des fichiers de l'extension XMI a l'inverse d'Acceleo qui a une interopérabilité avec toutes extensions (XMI ou extension de notre DSL). Acceleo contient une syntaxe colorie, un détecteur d'erreurs, et une complétion de code. Il est basé sur OCL et ça donne une caractéristique de vérification des contraintes local par contre Xpand a besoin d'un fichier Chek et extensions pour vérifier.

2.5 Conclusion

Après l'étude et l'analyse faite des outils on procéder d'une comparaison entre les outils qui ont les mêmes caractéristiques, il apparu que EMF(Eclipse Modeling Framework), Sirius et Acceleo son les outils convenables à notre étude.

Chapitre 3

Une approche automatique de transformation d'un modèle BPMN vers un code DD-LOTOS

3.1 Introduction

La transformation d'un modèle vers un text constitue un type de transformation spécifique par L'OMG dans un cadre de développement à base de modèles, il suit certains étapes pour décrire le processus de transformation d'un modèle en un text, dans ce chapitre nous allons proposer un processus de transformation basé sur les outils que nous avons présenté si dessus, pour automatiser la transformation de BPMN-collaboration vers DD-LOTOS.

3.2 Travaux correspondants

comme nous le connaissons, il y une tentative précédente de donner une définition formelle d'un daigrame BPMN comme dans [Dijkman et al., 2008] A proposé une sémantique formelle de BPMN par l'utilisation de réseaux de Petri. Ce travail est conduit à l'identification d'un nombre Des lacunesdans la spécification standard BPMN comme par exemple l'inter-blocage. Le travail se concentre sur le flux du contrôle alors il défini un sous-ensemble de la notation qui manque les concepts de haut niveaux par exemple les échanges de message entre les pool. Dans [Lyazidi and Mouline, 2013] les auteurs font une transformation du BPMN vers réseaux de Petri dans un cadre MDE utlisons la spécification dans [Dijkman et al., 2008] avec l'outil ATL(Langage pour spécifier les règles de transformation du modèle).

Dans [Wong and Gibbons, 2008] ce travail il propose une sémantique de processus dans langage de CSP pour un sous-ensemble de BPMN, qui peut être construite automatiquement à partir d'une représentation syntaxique simple du diagramme, les auteurs utilisent le langage **Z**. Dans ce travail montre comment on peut utiliser la sémantique pour vérifier si un diagramme BPMN est compatible avec les autre.
il y a d'autre travaux comme [Poizat et al., 2016] propose un modele de transformation de BPMN vers LNT(Algèbre de processus) et[Poizat et al., 2016] il représente des spécifications formelles d'un BPMN type chronologique. Tout ces travaux traitent un sous-ensemble de BPMN et de plus ils manquent des concepts essentiels comme le temps et la distribution.

3.3 Spécification formelle de BPMN

Pour la spécification formelle de BPMN on a utilisé le langage formelle DD-LOTOS, pour effectuer ce travail nous avons fait recours à la correspondance faite par Mlle Sara [Ghaoui, 2017] le tableau est figuré dans les annexes. On a utilisé ce tableaux pour extraire les règles de transformation.

3.4 Présentation générale de l'approche proposé

L'approche que nous avons proposée se compose de trois étapes que nous avons illustré dans la figure3.1

La première étape consiste à définir un méta-modèle pour BPMN-Collaboration, ensuite nous avons proposé un éditeur graphic de BPMN par le framework Sirius qui utilise un méta modèle que nous avons proposé pour créer des modèles, ces derniers sont conformes au méta modèle. En fin, la dernière étape est généré le code DD-LOTOS via un modèle par l'utilisation des template pré-défini par le langage de transformation Acceleo.

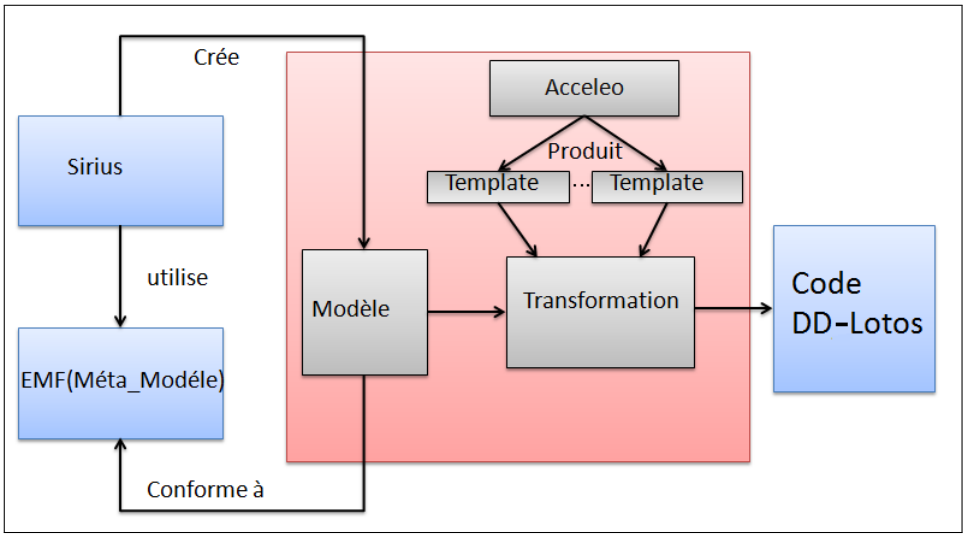


FIGURE 3.1 – Processus de transformation

3.5 Méta-modèle de BPMN

Dans cette section nous avons proposé un méta-modèle pour BPMN et on a spécifié le méta modèle pour BPMN-Collaboratif.

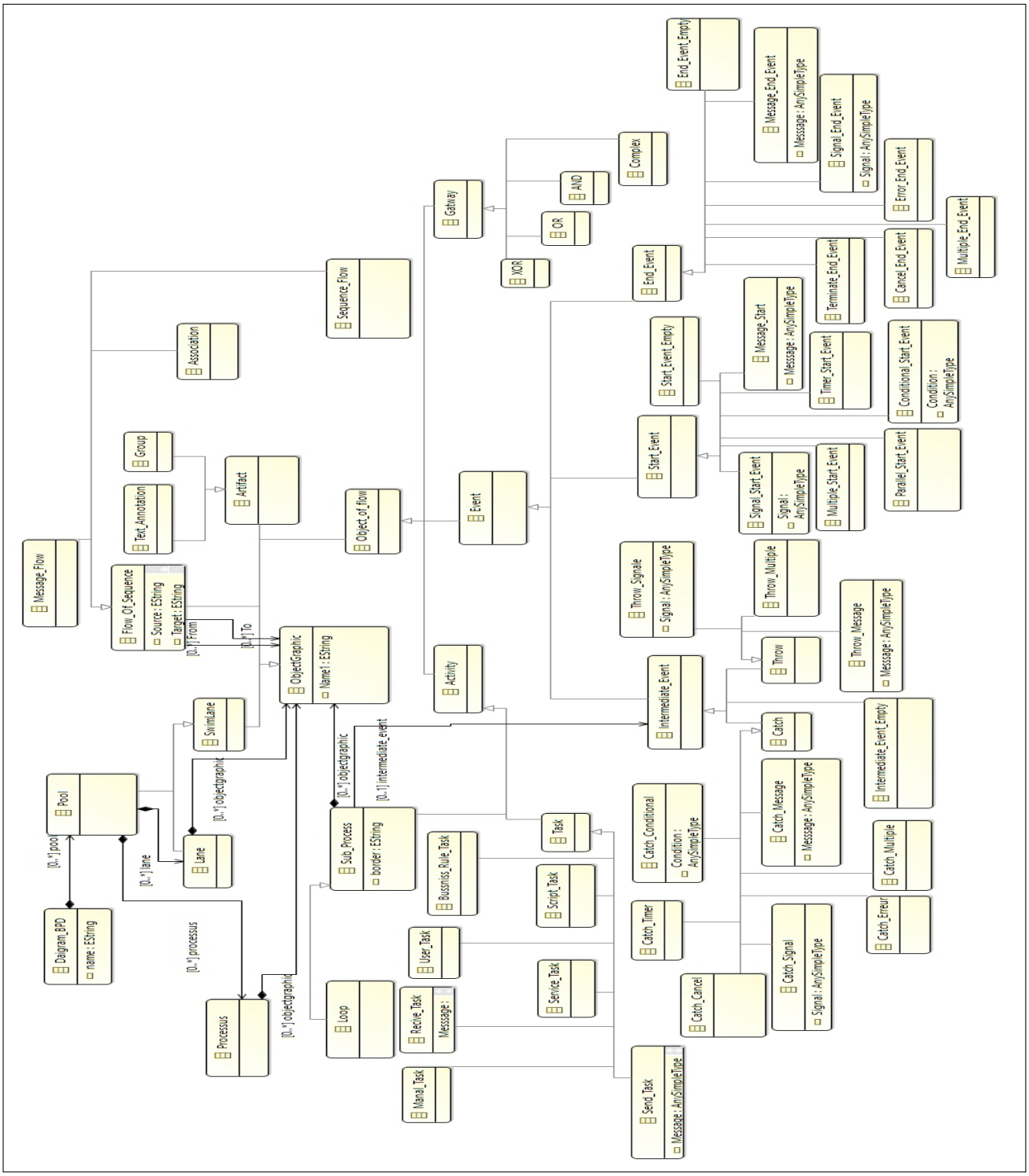


FIGURE 3.2 – Méta-modèle de BPMN

3.6 Éditeur BPMN sous Sirius

Le but de cette section est de réaliser un éditeur graphic BPMN, pour faciliter l'opération de dessin d'un modèle visuel avant de passer à l'étape de transformation voir figure3.3

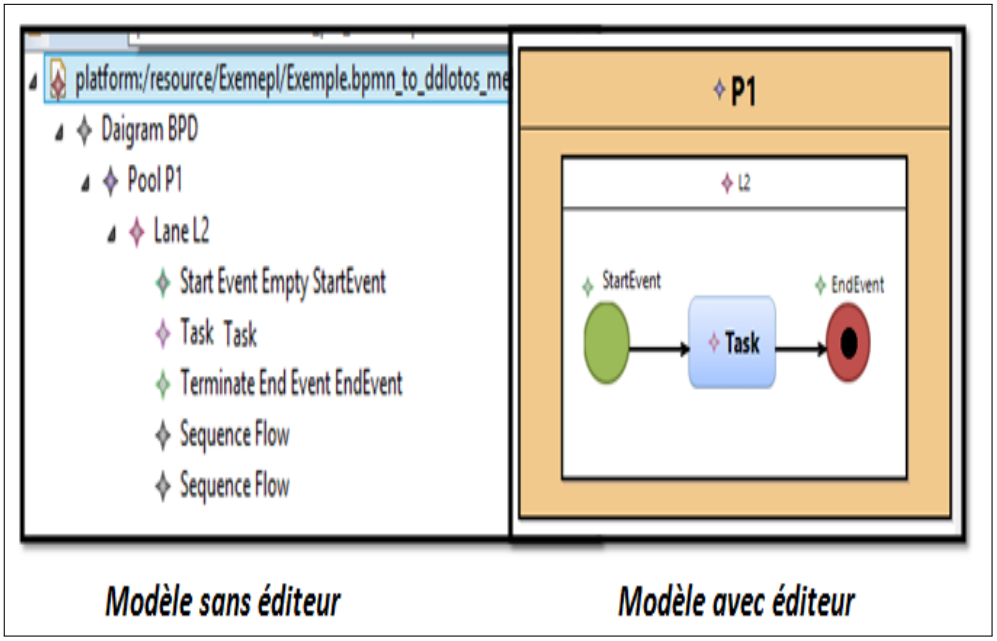


FIGURE 3.3 – Un modèle modélise sans/avec éditeur

Les étapes de création d'un éditeur par le plugin Sirius ont été déjà expliquer dans le chapitre précédant, après la création de l'éditeur nous obtiendrons l'éditeur comme il est apparu dans la figure 3.4.

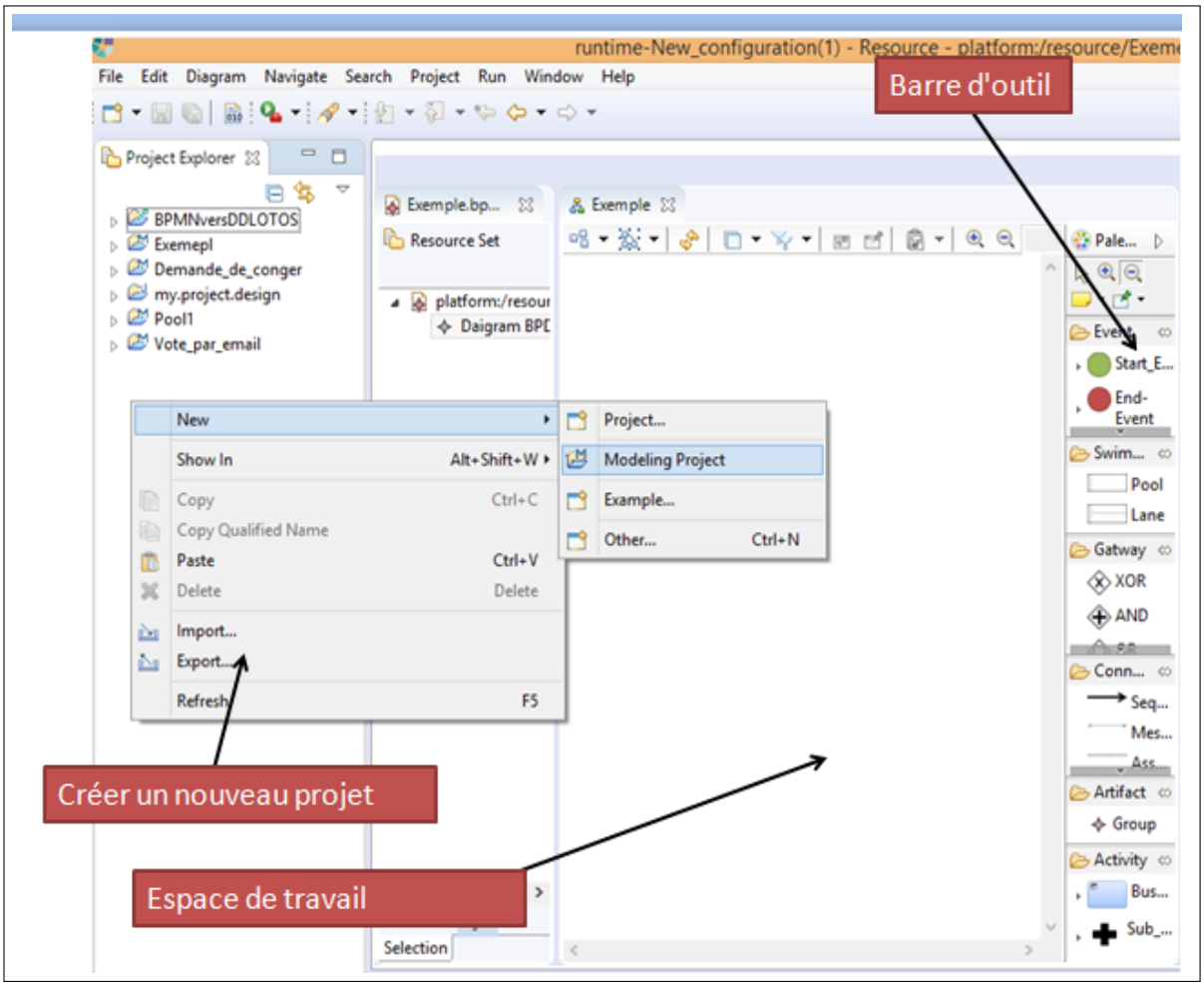


FIGURE 3.4 – Capture d'écran de project Sirius

La barre d'outil est organisée comme suit :

— **Pool et Lane**
Pool contient des lanes et une lane contient des objets c'est pour ça on modélisé les deux comme eContainer, cette spécification nous permet d'attribuer certains caractéristiques à l'éditeur comme :
Un pool contient au mois un lane, et ne permet pas de placer les objets que dans le lane.

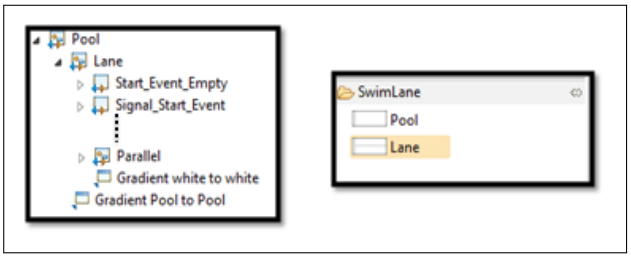


FIGURE 3.5 – Pool et lane

— **Événement**
Les événements sont des objets simples on les modélise comme des nœuds (Node).

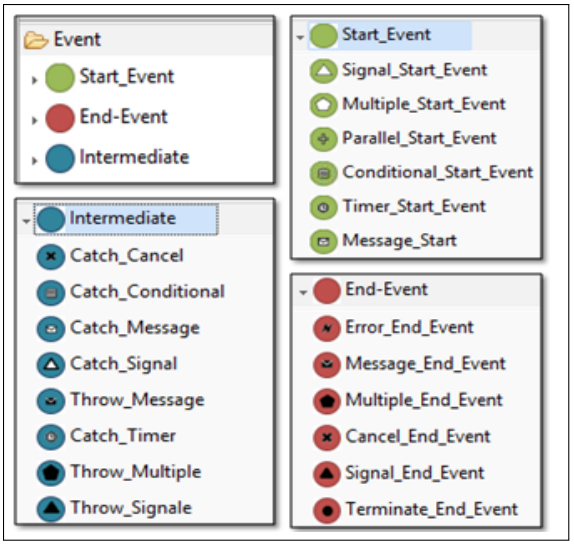


FIGURE 3.6 – Barre d'outil d'événement

— **Activité** on a deux types d'activités, les tâches sont des objets simples qui sont modélisé comme des nœuds, et les sous-processus qui ont une sémantique comme un processus. Alors, on la modélise comme eContainer qui hérite les caractéristiques d'un lane.

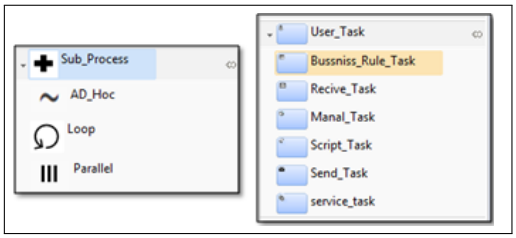


FIGURE 3.7 – barre d'outil de tâche et sous processus

— **Objet de relation**

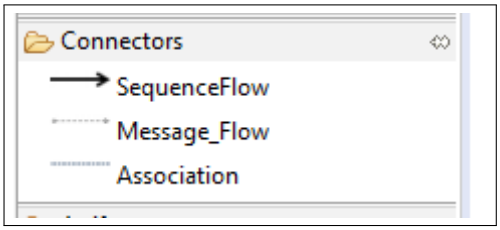


FIGURE 3.8 – Barre d'outil des objets de relation

— **Passerelle** Les passerelles sont des objets simples on les modélise comme des nœuds (Node).

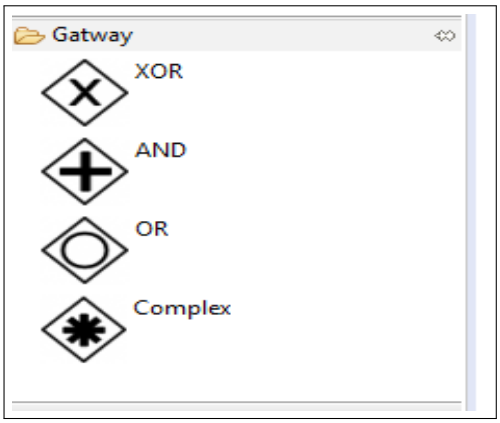


FIGURE 3.9 – Barre d'outil de passerelle

La modélisation par cet éditeur suit certains réglés dont nous avons résumé dans les points suivants :

— Une Piscine contient un seul événement de départ qui déclenche le processus.
— Pour dessiner les branches nous utiliserons seulement les passerelles et on ne permet pas qu'un événement, ou une tâche ou sous processus fait plusieurs sorties des séquences flux.

3.7 Un algorithme pour la transformation automatique de BPMN vers DD-LOTOS

Le BPMN utilise le BPD(Business Process Diagram) pour décrire le processus mitiè, et en même temps BPD respecte quelque structure qui il faut utilisé en consédiration durant de la transformation.

Les structures de processus principales de BPD sont (le modèle respectivement séquentiel, le modèle exclusif, la branche parallèle simple, le modèle de synchronisation, le modèle de fusion, le modèle de circulation) qui se présente dans la figure 3.10. Dans la phase de la transformation nous avons proposé quelques algorithmes qui assurent une transformation correcte des structures, et à partir de ces algorithmes nous produirons les templates d'Acceleo, la transformation est tirée à partir d'un tableaux qui figure dans [Dijkman et al., 2008]

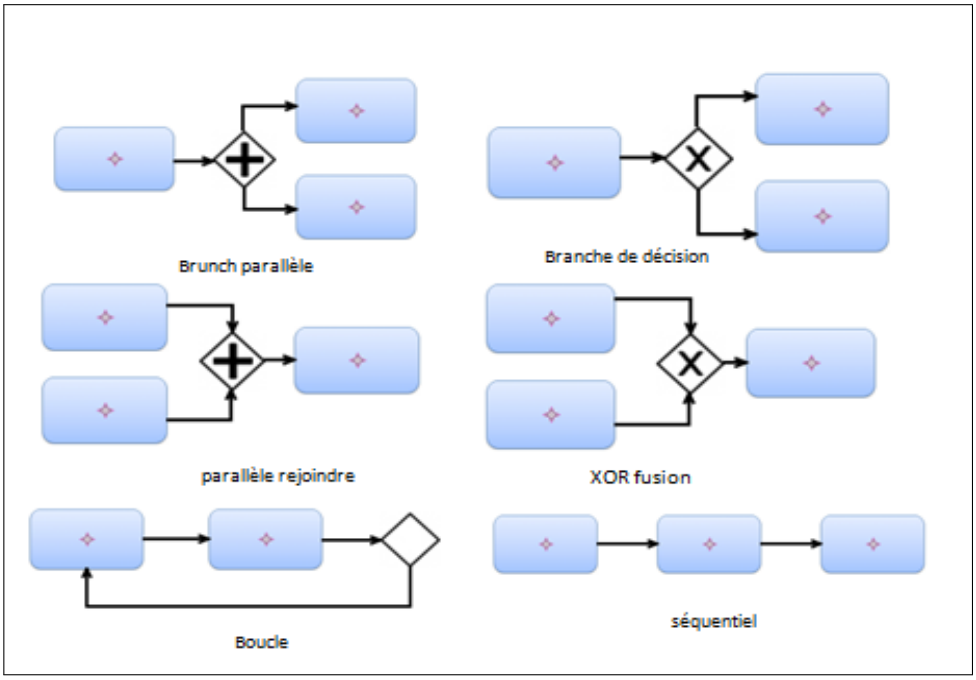


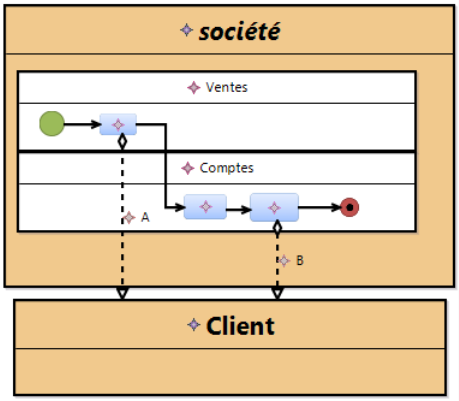
FIGURE 3.10 – Les structures de processus principales de BPD

3.7.1 La forme générale de DD-LOTOS

Le premier algorithme que nous proposons assure une interprétation de la forme générale de DD-LOTOS

Algorithm 1

```
1: Name ← Name_of_daigram;
2: Write("Specification"+ Name+" :=");
3:   foreach (Pool)  Do{ Write (L(Name_Of_Pool)}        ▷ separator(|[Canal]|)
4:          Write ("Where Process ")
5:   foreach (Pool)  Do{
6:       Write(Name Of Pool+" :=")
7:         Contents_of_Pool() }                          ▷ Call a function
8:    Write ( "EndProc")
9:        IF( Exist SubProcess) then
10:       Write ( Process )
11:       Sub_Process()                                  ▷ Call a function
12:       EndIF
13:       Write ( "EndProc ")
14: Write ( "EndSpec ")
```

BPMN	Code DD-LOTOS
	Specification Service := L(société)|[Canal]|L(Client) Where Process société := EndProc Process Client := EndProc EndSpec

3.7.2 Transformer le contenu de piscine

Pour transformer le contenu de piscine, nous suivons la procédure suivante :on trouve deux listes l'une des arc et l'autre des objects(event,task, sub process etc) pour chaque object et arc traité on le supprimer à partir des listes et tant que la liste des objets n'est pas vide on cherche l'objet suivant, pour assurer qu'il n'y a pas de répétition on ne transforme pas un object qui contient des flux de sequence entrant, juste dans le cas de tache ou d'événement qui contient un entrant à partir d'objet qui ne sont pas traités , alors on conclu que c'est une boucle et nous la traitons comme un

sous processus. si on trouve un sous processus il suffi de mettre son nom suivi par des parenthèses.

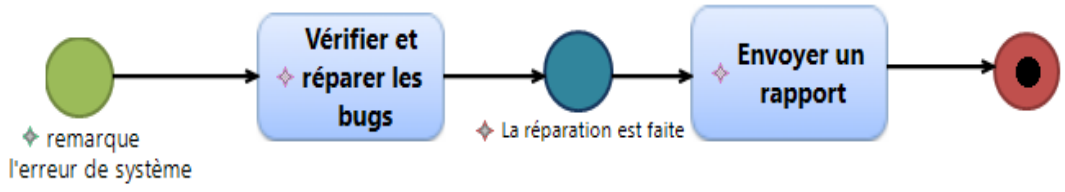
Algorithm 2 algorithm Contents of Pool()

```
1: Begin
2:    List Container ← Pool.Allcontents(Object) ;
3:    ListSequence ←Pool.Allcontents(Sequence_Flow) ;
4:    E ← Star Event ;
5:          Event(E)                ▷ Call a function that gives the corresponding code
6:          newListContainer←Remove E from List Container
7:          newListSequence←Remove edge(Edge.from=E)from ListSequence
8:          Next(aPool,newListSequence,newListContainer,E)
9: End
10: Procedure Next{P :Pool,L1 :ListSequence, L2 :ListObject,Obj :Object}
11:   IF (L2 is not empty) Do
12:    S←Pool.eContenet(Sequence.From=Obj) ;
13:    foreach (outEdge of Obj )Do
14:    next← S.To ;
15:   IF incomingEdge(next)=0 Do        ▷ Procedure return the nbr of enter link in object
16:    cas(Event) : Event(next) call Next(P,L1, L2-next,next)
17:     cas(Task) : Task(next) call Next(P,L1, L2-next,next)
18:     cas(Sub_Process) : next.name() call Next(P,L1, L2-next,next)
19:    /*And test if an event interrupts or no the subprocess
20:     cas(Gatway) : Gatway(next)
21:    ElSEIf(next is kind of(Event or Task))          ▷ Detect the circulation
22:      Sequence←—(Sequence.To=next) ;
23:      object←— Sequence.From ;
24: call Next(P,L1, L2-next,next) until object ;
25:     Circulation(next,object)
26:     Next(P,L1, L2-next,next)
27:    EndIF
28:    Endfor EndIF
29:  EndProcedure
```

Algorithm 3 Suivant

```
1: procedure EVENT(E :Event)
2:     IF(StartEvent)THEN
3:     cas :(Simple)DoWrite('[('True')]->')
4:     cas :(Timer)DoWrite('[('Event.time')]->')
5:     ...
6:     ENDIF
7:     IF(IntermedaiteEvent)THEN
8:     cas :(Conditional)DoWrite(Event.Conditional)
9:     cas :(Signal)DoWrite('[(Canal'?x'+ :Event.Messsage')]->')
10:    ...
11:    ENDIF
12:    IF(EndEvent)THEN
13:    cas :(Error)DoWrite('Stop')
14:    cas :(Cancel)DoWrite('[>Exit')
15:    ...
16:    ENDIF
17: end procedure
18: procedure TASK(T :Task)
19:    cas :(Script_Task)DoWrite('Task.Name')
20:    cas :(Send_Task)DoWrite(Canal'!x'+ :Task.Messsage)
21:    ...
22: end procedure
```

BPMN



Code DD-LOTOS

```
[remarque l'erreur de système]->
Vérifier et réparer les bugs; La réparation est faite; Envoyer un rapport;
Exit
```

— **Sous processus**

Il existe plusieurs types de sous processus dont chacun a ses spécificité et ses règles, le sous processus "type simple" nous le traitons comme un processus, la même procédure est suivit lors de "type boucle" mais après le traitement on leur fait une appelle récursive. Dans le type "parallèle" et "Ad-Hoc" le sous processus possède un ensembles de taches qui s'exécutent en parallèle mais dans le cas de "Ad-Hoc" le processus s'interrompre par Exit. Dans tous les cas il y une possibilité d'existence d'un événement intermédiaire sur les frontières qui peut interrompre ou ne pas interrompre le processus.

Algorithm 4 Suivant

```
1: procedure SUBPROCESS(p :subProcess)
2:     IF p= (simple_SubProcess) or (Loop) Then
3:             Process(p)
4:             IF p=Loop Then Write(p.name(new param))ENDIF
5:     ELSE                                          ▷ AD-HOC or Parallel
6:         ForEach task DO write the name and separate them with |||;
7:         IF p=Ad-Hoc Then Write(Exit)ENDIF
8:      ENDIF
9: end procedure
```

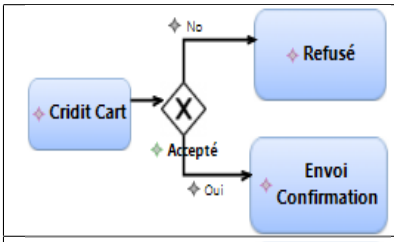
— **Transformation de passerelle**

Chaque passerelle a sa propre sémantique, elles sont toutes régit par la procédure suivante :

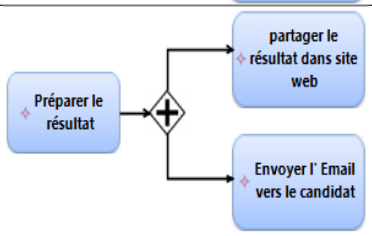
- Pour chaque sortie de sequence de flux on interprète les objets qui suit comme un chemin, les chemins sont séparés par(choix [], parallèle |||) selon le type de passerelle.
- le chemin se termine en deux cas si on détecte l'événement de fin ou une autre passerelle.

Algorithm 5 Gatway

```
1: procedure GATWAY(G :Gatway)
2:     for each kind of Gatway Do
3:      ForEach out_Edge Do Paths(G,listSequence,ListObject))
4: end procedure
5: procedure PATHS(o :Object,listSequence,ListObject)
6:     IF o.next= End_Event DO Event(o.next)
7:     ELSE IF o.next= Gatway DO  Gatway(o.next)
8:      ELSE
9:     o.next                          ▷ call code according to kind of object
10:    o← o.next ;
11:    PATHS (o,listSequence,ListObjects)
12: end procedure
```



Cridit Cart
[Accepté='No'] ⟶(Refusé)
[]
[Accepté='Oui'] ⟶(EnvoiConfirmation)



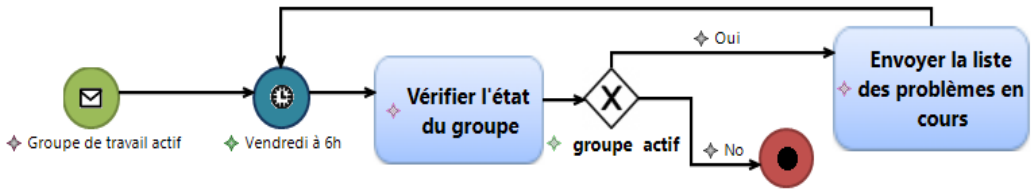
Préparer le résultat
(partager le résultat dans site web)
|||
(Envoyer l' Email vers le candidat)

— **Boucle**

Algorithm 6 Circulation

```
1: procedure CIRCULATION(object1 :Object,object2 :Object)
2:     (object())Where
3:     Process object :=
4:     call Next(object1) Until object2 ;
5:     EndProc
6: end procedure
```

BPMN



Code DD-LOTOS

```
[Canal?x Groupe de travail actif]⟶
[Vendredi à 6h]⟶ Vérifier l'état du groupe;
[ groupe actif=Non]⟶ (Exit)
[groupe actif=Oui]⟶(Envoyer la liste des problèmes en cours);object()
Where
Process object() :=
[Vendredi à 6h]⟶Vérifier l'état du groupe;
[ groupe actif=Non]⟶(Exit)
[groupe actif=Oui]⟶(Envoyer la liste des problèmes en cours);object();
```

3.8 Conclusion

Dans ce chapitre nous avons proposé une approche automatique basée sur la méta-modélisation et la correspondance a fin de générer le code DD-LOTOS en utilisons Acceleo. Nous avons proposé quelques algorithmes qui assurent la transformation correcte. il y a quelques objets qui sont difficile à transformer comme l'événement complexe et le passerelle complexe parce qu'il comporte à chaque fois une sémantique différente.

Chapitre 4

Étude de cas : Vote par Email

4.1 Introduction

Dans ce chapitre on va présenter un exemple de vote par mail, un exemple qui traite un ensemble des concepts importants dans BPMN. qui nous aide de tester la fonctionnalité de notre application. on va faire une comparaisons entre la transformation théorique et la transformation automatique (par l'application).

4.2 explication de l'exemple

Cet exemple montre les concepts principales utilisés dans BPMN, l'exemple est composé de deux piscines(se qui lui correspond une localité)**la piscine**(Système de vote)contient un processus qui déclenche tous les vendredis par un **événement de de type** TIME. Ensuite**un gestionnaire** des listes des sujets (Tâches)traitent les sujets et choisi ceux qui pourront rentrer dans **le cycle de discussion**(sous processus) La décision(avec une passerelle inclusive) doit être prise(décider si les sujets sont prêts ou non) soit pour enclencher le cycle de discussion et de vote soit pour on arrête là et attendre la semaine prochaine. Le sous-processus cycle de discussion possède deux flux de séquences entrants dans cette étape nous détectons une boucle qui traite en cas de présence de sujets à traiter ou suite à un besoin de rediscuter un sujet etc. Le processus se poursuit jusqu'à la fin.

4.2.1 Le daigrame de BPMN d'un processus vote par email

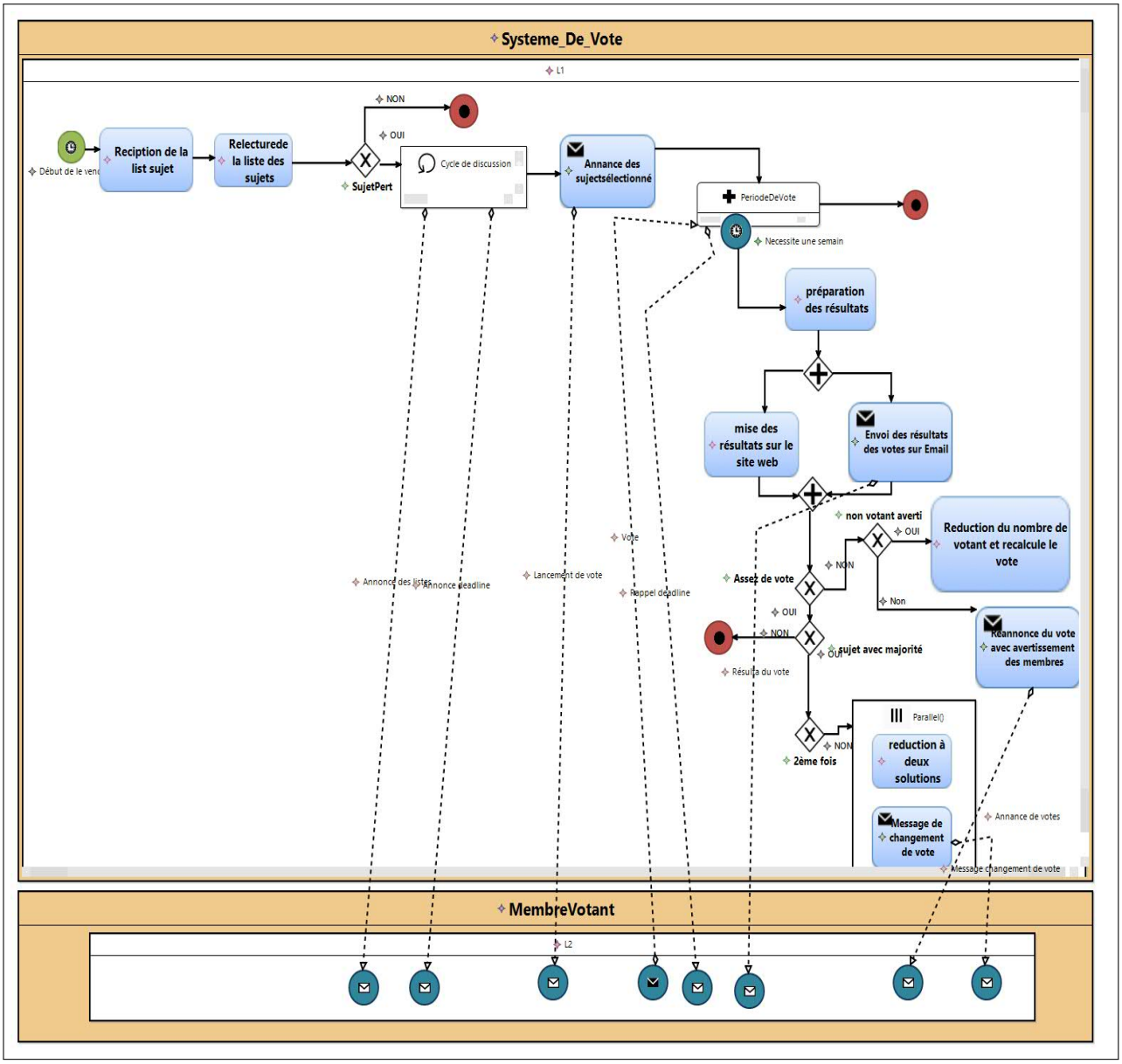


FIGURE 4.1 – Processus de vote par Email

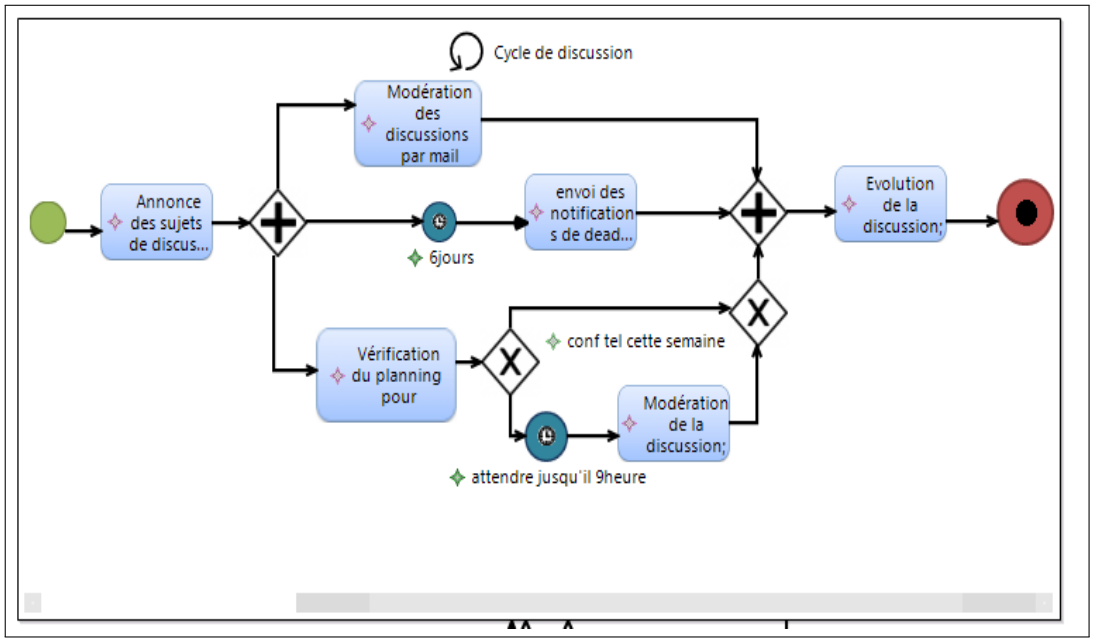


FIGURE 4.2 – Détail du cycle de discussion

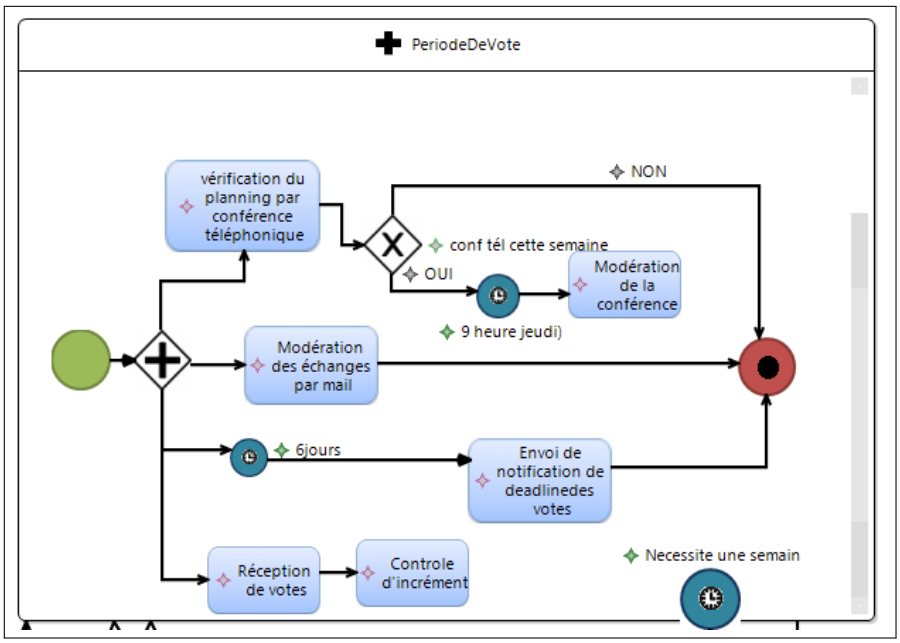


FIGURE 4.3 – Détail du sous-processus : période de vote

4.3 Le code manuel de DD-LOTOS

```
Specification VoteParEmail::=
L(systemeVote)|[Email]|L(MembreVotant)
process SystemeVote(Email)::=
g@t[jour= vendredi]-> receptionde la liste des sujets;
Relecturede la liste des sujets
[sujetPret='Non']->exit
[sujetPret='Oui']->cycle de discussion();
Email!v: annnonce de liste{2}
Email!v: annnonce deadline{3};
(periodeDeVote(email);exit)
            ]>g@x[x>=7 jours]->
                 preparation des resultats;
                 [mise des resultats sur le site web
                 |||
                  Envoi des resultats des votes sur Email;
            Email! v : Resultat de vote ]
[assez de vote= 'Non'] ->
           [ non votant averti = 'oui' ]->
            Reduction du nombre de votant et recalcule le vote
       [ non votant averti = 'non'] ->
            Reannonce du vote avec avertissement des membres
            Email?x : annonce de vote avec avertissement

[assez de vote= 'Non'] ->
            [sujet avec majorite='non']-> exit
      [sujet avec majorite='Oui']->
                [2eme fois ='non']->
                    reduction a deux solutions|||
                    [faire un email au votant devant changer le
   vote;
                    Email!x:Message de changement de vote]
                    where
                    process CycleDeDiscussion()::=
                    Annonce des sujets de discussion;
                    [g@{t=6jours} ->
                   envoi des notifications de deadline de la
   discussion]
                    |||
```

```
37                        Moderation des discussions par mail
38                                ]>g@t{t=6jours}
39                                |||
40                               Verification du planning pour
41                                conference telephonique;
42                        [conf tel cette semaine='Oui']->
43                        delay(x=9heure)Moderation de la discussion;
44                        Evolution de la discussion;Exit
45      EndProc
46      Process periode de vote()::=
47      (verification du planning par conference telephonique;
48      [conf tel cette semaine='Non']->exit
49      [conf tel cette semaine='Oui']->
50     delay(x=9 heure jeudi)Moderation de la conference;exit)
51      |||
52      (Moderation des echanges par mail; exit)
53      |||
54      delay(x=6jours)Envoi de notification de deadlinedes votes;
55      exit
56      |||Reception de votes;
57      Controle d''increment;
58      reception des votes"BBoucle "
59      EndProc
60
61
62  Process Membrevotant()::=
63  Email?x:Vote
64  Email?x:Vote
65  Email?x:Vote
66  Email!x:Vote
67  Email?x:Vote
68  Email?x:Vote
69  Email?x:Vote
70  Email?x:Vote
71  Endproc
72  EndSpec
```

Listing 4.1 – Le code manuel de DD-LOTOS

4.4 Le code généré par la transformation automatique

```
1  Specification  VoteParEmail:=
2  L(Systeme_De_Vote)|[Canal]|L(MembreVotant)
3  Where
4  Process Systeme_De_Vote(Canal):=
5  [g@t[ Debut de le vendredi]] ->Reciption de la liste sujets
6  ;Relecturede la liste des sujets
7  [SujetPert='NON']->(Exit)
8  []
9  [SujetPert='OUI']->(Cycle de discussion());
10  Canal!x:Annance des sujects selectionne;
11  PeriodeDeVote()(Exit)]> g@t[ 7jour]->
12         preparation des resultats
13             (mise des resultats sur le site web)
14            ||| (Canal!x: Envoi des resultats des votes sur Email
      );
15
16              [Assez de vote='NON']->(
17           [non votant averti='OUI']->(
18                      Reduction du nombre de votant et recalcule
      le vote)[]
19           [non votant averti='NON']->(Canal!x:Reannonce du vote
      avec avertissement des membres);
20         [] [Assez de vote='OUI']->(
21                    [sujet avec majorite='OUI']->(
22                       [2eme fois ='NON']->( taches())[]
23                  [sujet avec majorite='NON']->(Exit)
24
25    Where
26    Processus CycleDeDiscussion()::=
27                       Annonce des sujets de discussion;
28                  ( Verification du planning pour conference
      telephonique;
29                    ( [Conf tel cette semaine='Oui']->
30                     [x=attendre jusqu il 9heur]Moderation de la
      discussion;)
31                     |||
```

```
32                        ([6jours] ->
33                      (envoi des notifications de deadline de la
    discussion]))
34                   ; Evolution de la discussion;Exit
35     EndProc
36
37     Process periode de vote()::=
38 (Reception de votes; Controle d increment)
39     |||
40      [x=6jours]Envoi de notification de deadlinedes votes;Exit)
41      |||
42      (Moderation des echanges par mail;Exit)
43     (verification du planning par conference telephonique(
44     [conf tel cette semaine='Non']->exit
45     [conf tel cette semaine='Oui']->[x=9heur jeudi)]Moderation
    de la conference;Exit)))
46     EndProc
47
48 Process taches():=(reduction a deux solutions
49                         |||
50                      Canal!x:Message de changement de vote)
51 Process Membrevotant()::=
52 Canal?x:Annonce des listes
53 Canal?x:Annonce deadline
54 Canal?x:Lancement de vote
55 Canal!x:Vote
56 Canal?x:Rappel deadline
57 Canal?x:Resulta du vote
58 Canal?x:Annance de votes
59 Canal?x:Message changement de vote
60 Endproc
61 EndSpec
```

Listing 4.2 – Le code généré par la transformation automatique de DD-LOTOS

4.5 Une comparaison entre les deux codes

Après la comparaison entre les deux codes(le code manuel et le code générer par l'application). On a remarqué que les deux codes ont la même sémantique mais la

différence qui réside dans la syntaxe du code parmi ses différences on cite : la différence de nomination des ports de communication dans le cas de notre application nous avons lui accordé le mot clé canal par défaut. En revanche, dans le premier code ils ont attribué l'appellation Email. La deuxième différence, pour la transformation des sous processus est laissé a la fin dans notre travail par contre dans le premier code elle transforme le sous processus(de type parallèle)directement dans sa position. Et en fin, dans la deuxième pool(membre votant)l'objet évènement (send/recive message) prend l'appellation de l'arc.

4.6 Conclusion

à partir de l'étude du cas faite et après la comparaison entre les deux codes, on conclu que notre application fonctionne correctement.

Conclusion générale

Dans ce mémoire nous avons proposé la conversion de modèle de BPMN vers DD-LOTOS dans un objectif de vérification et de validation formelle. L'approche utilisée est celle basée sur les transformations des modèles.

nous avons commencé par des définitions des mots clés de notre thème tel que BPMN, DD-LOTOS et L'IDM et nous avons trouvé qu'il y a deux types de transformation M2T et M2M. Ensuite nous avons discuté sur les outils de la modélisation des méta modèle, de création des éditeurs et de transformation et après les études et les comparaisons faites nous sommes arrivé au résultat suivant : EMF, Sirius et acceleo sont les moyens les plus convenables à notre démarche. et à la fin de ce travail on peut dire qu'on a atteint les objectifs fixés au départ :

on a proposé un méta-modèle pour BPMN et on a spécifié le BPMN-Collaboratif, nous avons réalisé un éditeur BPMN pour créer les modèles d'une façon simple et il fonctionne correctement et en dernier lieu nous avons réaliser la transformation automatique par Acceleo mais il y a quelques objets qu'on n'a pas arrivé à les transformer comme l'événement complexe et le passerelle complexe à cause de sa sémantique ambiguë.

Nous espérons de découvrir au futur, un sous-ensemble plus large d'éléments BPMN tel que BPMN-Conversation et BPMN-Choreographie et chercher leur spécification formelle.

Bibliographie

[24,] *Tutoriel GMF : Ou comment créer un éditeur graphique à partir d'un modèle.*

[BAUCHE, 2011] BAUCHE, S. (2010-2011). la modélisation des processus métiers. Master's thesis, Université Paris Ouest Nanterre La défense.

[BERNARD, 2014] BERNARD, A. (2014). créer un éditeur de diagramme facilement avec eclipse sirius.

[Combemale, 2008] Combemale, B. (2008). *Approche de métamodélisation pour la simulation et la vérification de modèle–Application à l'ingénierie des procédés.* PhD thesis, Institut National Polytechnique de Toulouse-INPT.

[Dijkman et al., 2008] Dijkman, R. M., Dumas, M., and Ouyang, C. (2008). Semantics and analysis of business process models in bpmn. *Information and Software technology*, 50(12) :1281–1294.

[Garavel et al., 2013] Garavel, H., Lang, F., Mateescu, R., and Serwe, W. (2013). Cadp 2011 : a toolbox for the construction and analysis of distributed processes. *International Journal on Software Tools for Technology Transfer*, 15(2) :89–107.

[Ghaoui, 2017] Ghaoui, S. (2017). Une sémantique formelle pour le processus métiers bmpn en utilisant les algèbres de processus. Master's thesis, Université Abbes Laghrour Khenchela.

[Jézéquel et al., 2006] Jézéquel, J.-M., Gérard, S., and Baudry, B. (2006). Le génie logiciel et l'idm : une approche unificatrice par les modèles.

[Klatt, 2007] Klatt, B. (2007). Xpand : A closer look at the model2text transformation language. *Language*, 10(16) :2008.

[Lyazidi and Mouline, 2013] Lyazidi, A. and Mouline, S. (2013). Formal verification of bpmn models using petri nets.

[Messaoud, 2012] Messaoud, M. T. (2012). *Modèles formels pour la conception des systèmes temps réel.* PhD thesis, Université Mentouri – Constantine.

[Nadia, 2012] Nadia, M. (2012). Génération automatiquede code pnml à partir de réseaux de petri en utilisant xpand. Master's thesis, Université Abbas Laghrour-Khanchela.

[Naoufel, 2009] Naoufel, K. M. (2009). Une approche de transformation de la notation bpmn vers bpel basée sur la transformation de graphe. Master's thesis, Université Mentouri Constantine.

[OMG, 2016] OMG (1997-2016). Business process model and notation.

[OMG, 2017] OMG (1997-2017). Mda - the architecture of choice for a changing world.

[Owen and Raj, 2003] Owen, M. and Raj, J. (2003). Bpmn and business process management. *Introduction to the new business process modeling standard.*

[Palmer, 2014] Palmer, N. (2March 26, 2014). What is bpm ?

[Poizat et al., 2016] Poizat, P., Salaün, G., and Krishna, A. (2016). Checking business process evolution. In *13th International Conference on Formal Aspects of Component Software (FACS).*

[Rottenberg, 2015] Rottenberg, S. (2015). *Modèles, méthodes et outils pour les systèmes répartis multiéchelles.* PhD thesis, Institut National des Télécommunications.

[Thivolle, 2011] Thivolle, D. (2011). *Langages modernes pour la modélisation et la vérification des systèmes asynchrones.* PhD thesis, Université de Grenoble.

[von Rosing et al., 2015] von Rosing, M., White, S., Cummins, F., and de Man, H. (2015). Business process model and notation—bpmn. *The Complete Business Process Handbook*, pages 429–453.

[White, 2008] White, S. A. (2008). *BPMN modeling and reference guide : understanding and using BPMN.* Future Strategies Inc.

[Wong and Gibbons, 2008] Wong, P. Y. and Gibbons, J. (2008). A process semantics for bpmn. In *International Conference on Formal Engineering Methods*, pages 355–374. Springer.

Annexes

Tableau de correspondance

Elément BPMN		Interprétation formelle
Les objets de flux		
Activité		Selon son type.
Tache "tache"		tache
Types	abstraite"tache"	tache
	d'envoi"tache"	a !x : Message
	réception "tache"	a ?x : Message
	utilisateur "tache"	tache
	manuelle "tache"	tache
	de Règle de gestion "tache"	a !x : entrée a ?x : résultat
	Service "service"	service(paramètres)
	tache d'appel "tache"	tache
	Script"script"	script
Sous processus "proc"		processus proc(porte) := endproc ;
Types	à usage unique "proc"	processus proc(porte) := endproc ;
	réutilisable "proc"	processus proc(porte) := endproc ;
	parallèle"composé de : tache1,tache2 et tache3"	processus proc(porte) := tache1|||tache2|||tache3 endproc ;
	AD-HOC "composé de : tache1,tache2 et tache3"	processus proc(porte) := tache1|||tache2|||tache3]>exit endproc ;
	événementiel	[Ev]-> sousProcessus()

Evènement		Selon son type
	début	selon son type
	fin	Selon son type
	Intermédiaire	Selon son type
Evènement frontière		Selon son type
	Intérromptif	(tache]>[Ev]->tache|||E)
	Non intérromptif	(tache]>[Ev]->E)
Les passerelles 'les branchements'		Selon son type
	Passerelle exclusive	([p(x)]->chemin1 En DDLotos) [] [not(p(x))]->chemin2 En DDLotos [] chemin de sortie obligatoire
	Passerelle parallèle pour la synchronisation de plusieurs chemins parallèles	chemin1 En DDLotos|||chemin2 En DDLotos ||| ...||| cheminN En DDLotos
	Passerelle parallèle pour la synchronisation de plusieurs chemins parallèles	;
	Passerelle inclusive	([p(x)]-> chemin1 En DDLotos) [] ([G(x)]->chemin2 En DDLotos)
	Passerelle évènementielle exclusive	(([Ev1]-> chemin1)[ ]([Ev2]-> chemin2)[ ]....[ ]([EvN]-> cheminN))
	Passerelle complexe	Selon la spécification
Reconstitution des chemins pour tous types de passerelle		;
Les objets de connexion		
Message		Selon son type
	Envoi	a !v :Messaged
	Réception	a ? v : Messaged
Les flux de séquence		;
Les associations		Rien à ajouter
Les swimlanes		
Piscine contient un processus p		L(p)

Couloirs		Rien à ajouter
Diagramme de collaboration 'Spec'		Specification Spec(Porte) : := EndSpec
Processus proc		Processus proc(Porte) : := Endproc
Les éléments annexes		
Artéfacts		/* Contenu d'artéfact */
Les objets de données		//utiliser les types abstraits de données
Evènement de départ		
	Simple	[true]->
	Réception de message	[a ?x : Message]->
	Timer	[p(timer)]->
	Réception d'un signal	[a ? x : Signal]->
	Multiple	(Ev1[]Ev2[]...[]Ev3)->
	Conditionnel	[prédicat]->
	parallèle	(Ev1|||Ev2|||Ev3)->
Evènement de fin		
	Simple	exit
	Envoi d'un message	a !Message ; exit
	Envoi d'un signal	a ! Signal ;exit
	Erreur	Stop
	Cancel	[>exit
	Multiple	(Ev1[]Ev2[]...[]Ev3)->exit
	Parallèle	(Ev1|||Ev2|||Ev3)-> exit
Evènement intermédiaire		
	Envoi d'un message	a !message
	Réception d'un message	a ?x :message
	Timer	[p(timer)]
	Envoi d'un signal	a !Signal
	Réception d'un signal	a ?x :Signal
	Lien	Rien à ajouter
	Conditionnel	[prédicat]
	Multiple	(Ev1[]Ev2[]...[]Ev3)
	Parallèle	(Ev1|||Ev2|||Ev3)