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

Mémoire de fin d'études

pour l'obtention du diplôme de Master en Informatique
Spécialité : Génie logiciel et systèmes distribués.

Transformation des diagrammes de collaboration BPMN en langage PROMELA

Présenté par :

OUADAOUI Souhila

Dirigé par :

Dr MAAROUK Toufik Messaoud

Promotion : 2020/2021

Remerciement

Remercie ALLAH de m'avoir donné le courage et la volonté ainsi que la conscience de terminer mes études.

Puis-je d'abord adresser mes sincères remerciements à mon père défunt Lhaj Mekki, qui m'a toujours poussé et motivé dans mes études. Puisse Dieu, le tout puissant, l'avoir en sa sainte miséricorde.

Je tiens à exprimer toute ma reconnaissance à mon directeur de mémoire,
Dr MAAROUK Toufik Messaoud. Je le remercie de m'avoir encadré, orienté, aidé et conseillé.

Je remercie ma trés chère maman Fatiha, elle a toujours été là pour moi. Je remercie mes sœurs , et mes frères , pour leurs encouragements.

Je désire aussi remercier mes enseignants de l'université Abbes Laghrour, qui m'ont fourni les outils nécessaires à la réussite de mes études universitaires.

À tous ces intervenants, je présente mes remerciements, mon respect et ma gratitude.

Résumé

Le modèle de processus métier et notation (BPMN) fournit un langage graphique standard qui peut être utilisé par les analystes métiers pour modéliser les collaborations des processus métiers.

L'objectif principal de ce travail est proposer une approche de transformation des processus métiers BPMN en termes de langage Promela. L'idée de prendre chaque élément BPMN et lui correspondre un code Promela équivalent. Promela est un langage qui est utilisé comme entrée pour le modèle checker SPIN. Cette étape est indisponsable pour une éventuelles vérification formelle des processus BPMN.

Abstract

Business Process Model and Notation (BPMN) provides a standard graphical language that can be used by business analysts to model business process collaborations.

The main objective of this work is to propose an approach for transforming BPMN business processes in terms of Promela language. The idea of taking each BPMN element and corresponding to it an equivalent Promela code. Promela is a language that is used as input for the SPIN checker pattern. This step is essential for any formal verification of the BPMN processes.

Table des figures

Liste des tableaux

Table des matières

Introduction générale

Le modèle de processus métier et notation (BPMN)[BPMN2] est le langague des flux de travail automatisés. Le langage universel BPMN fait le lien entre les développeurs, les exécuteurs et les analystes commerciaux, permettant à chacun de comprendre la séquence d'événements, d'informations et d'activités qui passe par une chaîne de tâches automatisées[Larissa20].

En effet, BPMN est capable de modéliser des diagrammes de processus complexes de manière intuitive[Pierre18]. BPMN est une notation semi-formelle, c'est-à-dire une conception basée sur BPMN peut conduire à des diagrammes incohérent. Pour remédier à cette insuffisance nous proposons de valider ces diagrammes en utilisant un langage formel.

De l'autre côté, le langage Promela[Promela] (Process Meta Language) est un langage de spécification des systèmes asynchrones, ce qui en d'autres termes veut dire que ce langage permet la description de systèmes concurrents, comme les protocoles de communication. Il autorise la création dynamique de processus. La communication entre ces différents processus peut se faire en partageant les variables globales ou alors en utilisant des canaux de communication. On peut ainsi simuler des communications synchrones ou asynchrones, Promela est associé à l'outil de vérification formelle Spin.

L'objectif de notre travail, est de proposer une approche de transformation des éléments BPMN en terme de code Promela, cette approche permet de traduire des diagrammes de collaboration BPMN en code Promela pour une éventuelle vérification formelle de ces diagrammes en utilisant le modèle checker SPIN[SVL].

Plan du mémoire

Le mémoire est structuré en trois chapitres de la manière suivante :

Une **Introduction générale** dans laquelle nous nous introduisons le contexte et

la problématique de ce travail.

Le **Chapitre 1** est consacré au modèle de processus métier et notation (BPMN).

Le **Chapitre 2** présente de manière détaillée le langage formel PROMELA.

Le **Chapitre 3** est dédié à notre contribution qui concerne la transformation des éléments BPMN en PROMELA.

Et nous terminerons par une conclusion et des perspectives.

Chapitre 1

Notation BPMN

1.1 Introduction

Un processus métier (Business process) BP est un ensemble de tâches liées les unes aux autres qui prennent fin à la livraison d'un service ou d'un produit à un client. Le processus métier a également été défini comme un ensemble d'activités et de tâches qui, une fois effectuées, rempliront l'un des objectifs de l'entreprise[BPMN2].

Le processus doit inclure des entrées clairement définies et une seule sortie. Ces entrées se composent de tous les facteurs qui contribuent (directement ou indirectement) à la valeur ajoutée d'un service ou d'un produit. Ces facteurs peuvent être classés en processus de gestion, processus opérationnels et processus métier d'accompagnement[WT2018].

La définition du terme « processus métier » et l'évolution de cette définition depuis sa conception par Adam Smith en 1776 ont mené à la création de domaines d'études tels que le développement d'activités et la gestion d'activités et au développement des divers systèmes de gestion opérationnelle[BPMN2].

1.2 Business Process Model and Notation

BPMN (Business Process Model and Notation)[BPMN] est une notation pour la représentation graphique des processus métiers dans un workflow. 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.

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

Dans ce chapitre nous aborderons premièrement l'approche BPM. Nous présenterons ensuite la représentation graphique utilisé par le standard BPMN 2.0, enfin nous étudierons les différents types de diagrammes.

BPMN s'articule autour de quatre catégories d'éléments. Les tableaux suivantes définit les différents éléments de BPMN [RG2017][JC2014] :

Tâche		
Manuelle	Permet de modéliser une action effectuée exclusivement par un acteur humain	
Utilisateur	Permet de modéliser une action effectuée par un acteur en interaction avec un service informatique	
Réception	La tâche de réception spécifie que l'on reçoit un message d'un utilisateur extérieur	
Envoi	La tâche d'envoi spécifie que l'on envoie un message à un utilisateur externe au processus	
Service	Une tâche automatisée, c'est-à-dire sans intervention humaine. L'application informatique déclenchée est vue comme un service demandé	
Script	Une tâche automatisée mais dont le comportement a été construit spécifiquement pour la gestion du processus	
Règle de gestion	Permet d'indiquer que l'action de la tâche est d'appliquer une règle métier pour prendre une décision.	
Objets de connexion		
Les messages	Servent à décrire les échanges entre processus	
Les flux de séquence	Représentent le flux entre deux tâches.	
Les associations	Support de rattachement entre une tâche et un objet de données ou avec une activité de compensation	

TABLE 1.1 – Tâches d'un BPMN

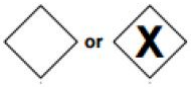
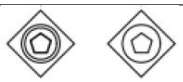
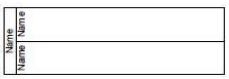
Les passerelles		
Passerelle exclusive	Un branchement exclusif évalue l'état du processus métier et, selon la condition définie, fait passer le flux sur un chemin en particulier, à l'exclusion du ou des autres.	
Passerelle parallèle	Les branchements parallèles sont utilisés pour représenter deux tâches simultanées dans un processus métier.	
Passerelle inclusif	Un branchement inclusif sépare le schéma de procédé en plusieurs flux	
Passerelle évènementielle	Dans le cas d'un branchement dépendant d'un événement vous évaluez l'événement qui s'est produit	
Passerelle complexe	Utilisée lorsque le comportement de flux ne peut pas être exprimé par un autre type de passerelle.	
Les swimlanes		
Bassin (POOL)	C'est un conteneur. Il représente les frontières d'un processus. Toutes les tâches se déroulent à l'intérieur du bassin. Seules les communications (flux de message) peuvent sortir d'un bassin vers un autre bassin.	
Couloir (Lane)	Un couloir représente un acteur, un rôle à l'intérieur processus d'un bassin. Le flux d'activité peut traverser les couloirs pour représenter l'enchainement des tâches entre les différents acteurs de notre processus.	

TABLE 1.2 – Type de passerelle d'un BPMN

Événement				
Evènement de départ/fin simple : aucune indication particulière n'est donnée				
Événement d'envoi et de réception d'un message ; Les messages s'adressent exclusivement à des receveurs uniques				
Événement temporel/timer : "Le processus démarre ou fait une action lorsque la condition temporelle est vérifiée".				
Événement d'envoi et évènement de réception d'un signal				
Évènements de type "lien" : permettent de découper un processus en plusieurs parties				
Évènement de type erreur : signale une erreur dans le processus et interrompt le processus en cours				
Plusieurs évènements peuvent déclencher/ mettre fin au processus				
Le processus sera déclenché ou terminé à faire quelque chose lorsque le prédicat soit vérifié				
Evènement d'annulation : signale la fin du processus et annule les transactions en cours.				
Évènement de terminaison : il ordonne la fin du sous-processus et de l'ensemble des tâches encours qui le compose				
Évènement parallèle				

TABLE 1.3 – Événements d'un BPMN

Activité en boucle		
Activité en boucle	Toutes les instances doivent se terminer , pour que l'activité soit terminée	
Sous processus en boucle	Un sous processus qui se répète tant qu'une condition n'est pas remplie.	
Activité multi-instance	**En parallèle**	**En séquence**
Activité MI		
Sous processus MI		

Table 1.4 – Tâches en boucle et MI d'un BPMN

1.2.1 Les activités de répétition

Activité en boucle

La boucle (équivalent du while en programmation) définit une tâche qui se répète tant qu'une condition n'est pas remplie. La condition est examinée à chaque itération de la boucle.

Activité multi-instance

Une activité à instance multiple (MI) permet de représenter plusieurs exécutions de l'activité.

1.3 Les niveaux de modélisation en BPMN

La modélisation des processus d'affaires se fait avec plusieurs niveaux selon la méthodologie suivie ainsi que la clientèle visée. La notation BPMN permet de faire des diagrammes de haut niveau comme de niveau très détaillé[KATIRI10].
La notation BPMN structure ses diagrammes nommés BPD (Business Process Diagram) en trois niveaux[Silver09] :

1.3.1 Niveau descriptif

Le but de ce premier niveau de modélisation, est de décrire le processus d'affaires d'une manière générale et simple. Il consiste à représenter le flux principal du processus.

1.3.2 Niveau analytique

L'analyse des diagrammes BPD du niveau descriptif ne permet pas d'améliorer la qualité et la performance du dit processus. Ceci nécessite d'en extraire des diagrammes plus détaillés qui décrivent tout les scénarios possibles du processus. Il s'agit, en effet, des diagrammes BPD du niveau analytique. Le deuxième niveau de modélisation, destiné aux architectes et analystes d'affaires, vise à décrire les détails du processus d'une manière précise. En effet, les flux d'exception sont aussi représentés.

1.3.3 Niveau exécutable

Ce niveau spécifique à la notation BPMN 2.0, est dédié spécialement aux développeurs informatiques. Il consiste à produire des modèles exécutables des processus d'affaires. Actuellement, il est supporté par les solutions BPMS d'IBM, ORACLE, et SAP.

1.4 Diagramme BPMN

Les diagrammes BPMN [BPMN][OMG] utilisent les notations graphiques pour la conception d'un processus métier, de sorte que tout le monde est capable de comprendre et de communiquer les procédures de manière standard. Au sein de la notation BPMN 2.0, on retrouve quatre catégories de diagrammes.

1.4.1 Le diagramme d'orchestration

Le processus d'orchestration est un processus standard, nous rencontrons le plus souvent dans BPMN. Il modélise généralement un seul point de vue de coordination. Un diagramme d'orchestration décrit la séquence d'un processus avec événements, activités, passerelles.

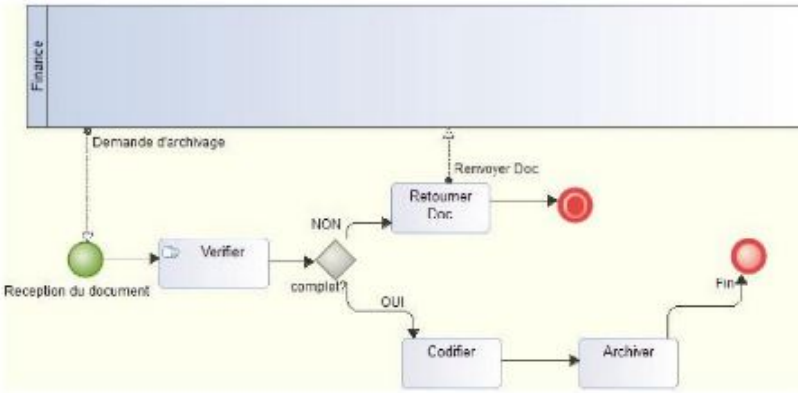


FIGURE 1.1 – Diagramme d'orchestration

1.4.2 Le diagramme de chorégraphie

Une chorégraphie est la modélisation d'un comportement attendu entre des participants qui interagissent les uns avec les autres et qui veulent coordonner leurs activités ou leurs tâches à l'aide de messages. Dans ce type de modélisation, la focalisation n'est pas sur l'orchestration (processus public ou privé), c'est-à-dire sur la manière dont est accompli le travail selon le point de vue des participants, mais sur les échanges de messages entre les participants. Le message est l'élément central de la chorégraphie.

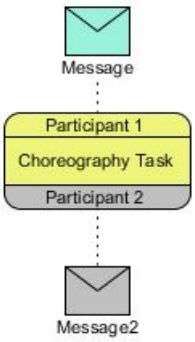


FIGURE 1.2 – Diagramme de chorégraphie

1.4.3 Le diagramme de conversation

Le diagramme de conversation est la description informelle d'un diagramme de collaboration à haut niveau. Il représente des échanges de messages (présentés sous la forme d'un regroupement de flux de messages), logiquement reliés entre les participants

et qui concernent un objet d'affaires d'intérêt, par exemple, un ordre, un envoi, une livraison ou une facture. Il représente un ensemble de flux de messages qui est regroupé ensemble. Une conversation peut impliquer deux ou plusieurs participants.

FIGURE 1.3 – Diagramme de conversation

1.4.4 Le diagramme de collaboration

Un diagramme de collaboration est tout type de diagramme qui permet de représenter les échanges et les interactions qui se nouent entre deux ou plusieurs unités d'affaires représenté par des bassins. Les bassins sont définis comme étant les participants de cette collaboration. Les messages échangés entre les participants du diagramme de collaboration sont présentés à l'aide du symbole flux de message. Ce symbole permet de connecter les bassins entre eux (ou les objets que l'on retrouve à l'intérieur de ces bassins).

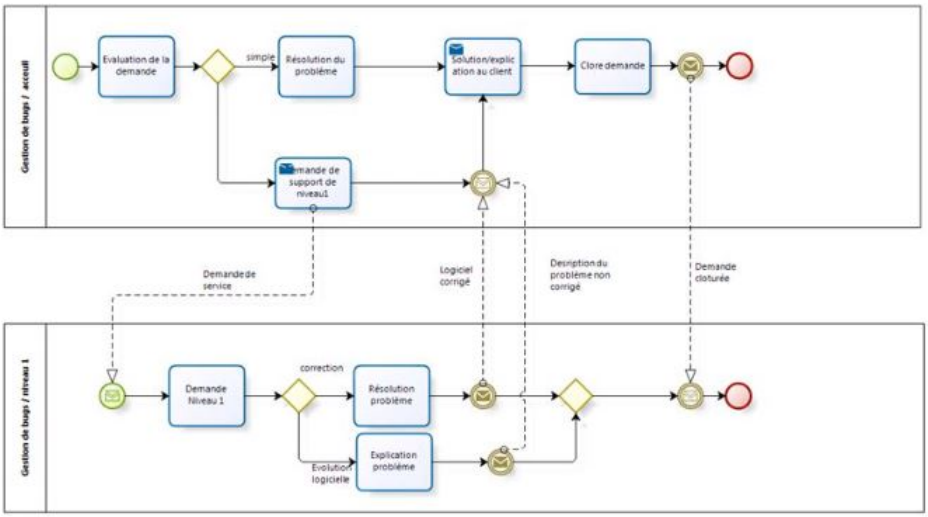


FIGURE 1.4 – Diagramme de Collaboration

1.5 Conclusion

BPMN est la notation standard la plus adaptée pour modéliser les processus d'entreprise. L'objective de la modélisation constitue la première brique sur laquelle la grammaire d'un processus métier est construite.

BPMN s'oriente plus vers la transformation d'un processus d'entreprise en processus exécutable et il en résulte une grammaire qui ne répond pas aux attentes des analystes et modélisateurs métier. De plus, l'absence flagrante des approches d'analyses telle que la gestion des risques impose la mise en œuvre des règles de validation et ainsi complique considérablement la tâche du staff métier.

Chapitre 2

Le langage Promela

2.1 Introduction

Promela (Protocol Meta Language), c'est un langage qui permet de modéliser des systèmes distribués, des processus qui se déroulent en parallèle et qui se communiquent à travers des canaux[Promela].

Un modèle de protocole écrit par Promela peut être simulé et vérifié par le l'outil SPIN et son interface graphique XSPIN[Promela].

Ces outils permettent d'étudier un protocole de façon exhaustive et de trouver toutes les possibilités d'erreur qu'il renferme, par exemple les deadlocks, starvation, et livelock etc.

2.2 Process Meta Language

PROMELA[Promela] est un langage de modélisation de processus dont l'utilisation prévue est la vérification de la logique des systèmes parallèles[FB2013]. Etant donné un programme en PROMELA, Spin peut vérifier l'exactitude du modèle en effectuant des simulations aléatoires ou itératives de l'exécution du système modélisé, ou il peut générer un programme C qui effectue une vérification exhaustive rapide de l'espace d'état du système. Pendant les simulations et les vérifications, SPIN vérifie l'absence de blocages, de réceptions non spécifiées et de code non exécutable. Le vérificateur peut également être utilisé pour prouver l'exactitude des invariants du système et il peut trouver des cycles d'exécution non progressifs. Enfin, il prend en charge la vérification

des contraintes temporelles en temps linéaire, soit avec les neverclaims Promela, soit en formulant directement les contraintes en logique temporelle[Promela].

2.3 Simple Promela Interpreter

SPIN(Simple Promela Interpreter)[SVL] est un outil général pour la vérification de l'exactitude des modèles de logiciels distribués d'une manière rigoureuse et surtout automatisée. Il a été écrit par Gerard J. Holzmann et d'autres membres du groupe Unix original du Computing Sciences Research Center des Bell Labs, à partir de 1980. Le logiciel est disponible gratuitement depuis 1991, et continue d'évoluer pour suivre les nouveaux développements dans le domaine. Depuis 1995, des ateliers SPIN (approximativement) annuels ont été organisés pour les utilisateurs de SPIN, les chercheurs et les personnes généralement intéressées par le model-checking[SVL].
Spin comporte essentiellement deux modes :
Simulation :le système est exécuté pas à pas, ce qui permet de se familiariser avec son comportement.
Vérification :les états du système sont explorés exhaustivement pour vérifier que le système satisfait bien certaines propriétés exprimées.

2.3.1 Spin en mode simulation

En mode simulation, spin simule une exécution possible du système.La simulation s'arrête en cas d'assertion non vérifiée ou deadlock.
Options de ligne commande importantes :

- -T : ne pas indenter la sortie des printfs.
- -p : afficher les opérations exécutées.
- -l : afficher la valeur des variables locales.
- -g : afficher la valeur des variables globales.
- -i : simulation interactive, l'utilisateur choisit la transition suivante.
- -c : afficher "graphiquement" les émissions et réceptions sur les canaux.

2.3.2 Spin en mode vérification

- En mode vérification, spin explore l'espace des états accessibles du programme (= simule toutes les exécutions possibles du programme).
- Les instructions printf sont ignorées.
- L'exploration s'arrête en cas d'assertion non vérifiée.

2.4 Structure d'un programme Promela

Un programme Promela est constitué d'un ensemble de processus, de canaux de communication utilisés pour l'échange des message entre processus, et de variables. Dans un processus on peut déclarer des variables et des canaux de communication de manière globale ou locale[FB2013]. Un processus est donné par une déclaration proctype. Un processus s'exécute en parallèle aux autres processus.
Un programme Promela est une liste de déclarations de type, constantes, variables (globale) et processus suivi d'un point d'entrée du programme (`init`).

2.4.1 Déclaration des types

Les types de données de base utilisés dans PROMELA
- Bit [0..1]
- Bool [0..1]
- Byte [0..255]
- Short
- Int
- Mtype [0..255]

Les variables déclarées dans ces types sont initialisées avec des valeurs par défaut. L'exemple suivant permet de déclarer plusieurs variables de différentes types.
Exemple de déclaration de variables :

```
bit g1; /*0*/
byte g2; /*0*/
int g3; /*0*/
int tab[0 1];/*\{0,0,...,0}*/
short x=123; /*123*/
active proctype proc(){
int loc1;
prinf("g1 %d loc1 %d/n, g1, loc1")
```

2.4.2 Déclaration des canaux de communication

La communication entre les processus se fait via des canaux par des messages donc on a un émetteur et un récepteur. Les canaux transmettent ces messages dans l'ordre premier entré, premier sorti (FIFO). Les messages dans les canaux doivent être typées.
Exemple de déclaration du canaux :

```
chan can1 = [N] of {type1 , . . . , typeN}
```

Les opérations sur les canaux peuvent bloquer l'exécution :

- can ! var ; bloque le processus si can est plein.
- can ? var ; bloque le processus si can est vide.
- can ? 5 ; bloque le processus si can ne contient pas l'entier 5.

2.4.3 Déclaration des processus

- On déclarer un processus par le mot proctype.
- Un processus peut contient des paramètres (arguments).
- Le " ;" pour terminer les instructions.
- La déclaration des variables comme en C.
- Les variables globales sont déclarées hors processus.
- Le printf pour afficher le texte ou les résultats sur le fichier de sortie.
- On n'a pas la possibilité de définir et appeler des fonctions.

La déclaration d'un processus :

```
Proctype proc (typeP1 P1 ; . . . ; typePn Pn) {
/* code du processus */
}
```

Création des processus statiquement

Sont définis par le mot clé active. On peut déclarer N processus.
L'exemple suivant permet de déclarer N processus de type statique.
La déclaration du processus statiquement :

```
active [N] proctype proc ( typeP1 P1 ; . . . ; typePn Pn) {
/* code du processus */
}
```

Création des processus dynamiquement

Ces processus sont éxecuter par l'instruction `run`.
L'exemple suivant permet de créer du processus de type dynamique.
La déclaration du processus dynamiquement :

```
proctype proc ( typeP1 P1 ; . . . ; typePn Pn) {
/* code du processus */
```

```
}
. . .
run proc ( x1 , . . . , xn)
```

Le processus init

Est un processus facultatif créé à l'initialisation du programme, pour initialiser des variables globales et lancer d'autres processus. Dans cet exemple suivant, nous avons créé deux processus et les avons exécutés par le processus `init`.

Exemple du processus init :

```
active [ 2 ] proctype proc1 ( ) {
printf ( " statique\n" ) ;
}
proctype proc2 ( int x ; int y ) {
printf ( " dynamique   :  (%d,%d )\ n" , x , y ) ;
}
init {
run proc2 ( 1 , 2 ) ;
run proc2 ( 3 , 4 ) ;
}
```

La simulation du programme :

```
$ spin prog .pml
dynamique : (1 ,2)
statique
statique
dynamique : (3 ,4)
```

2.4.4 Les instructions

L'instruction if

Est exécuté de manière non-déterministe, si plusieurs gardes sont exécutables.

— On utilise «->» au lieu de « ;». Par convention, il est utilisé dans des déclarations `if` pour séparer les gardes.

— Exécute `else` si aucune des gardes n'est exécutable.

— Si aucune des gardes n'est exécutable et qu'il n'y a pas de `else`, le processus est bloqué.

Exemple de l'instruction if :

```
mtype = { id msg };
chan name = [0] of { mtype, byte };
active proctype A(){
name!msg(100);
name!id(10);
}
active proctype B(){
byte var;
if
:: name?msg(var) -> printf("state = %d", var);
:: name?id(var) -> printf("value = %d", var);
else -> printf("");
fi
}
```

L'instruction do

Même sémantique que le `if` :

— il décrit une structure de répétition dans PROMELA.

— on utilise la instruction break pour terminer la structure de répétition.

Exemple de l'instruction do :

```
i = 0 ;
y = 0 ;
do
: : i < 10 ¡>
c ? x ;
y = y + x ;
i = i + 1
: : i == 10 ¡>
break
od
```

2.5 Conclusion

Dans ce chapitre nous avons introduit le langage Promela pour la modélisation des systèmes concurrents.
Ce langage présente plusieurs caractéristiques. Toute communication entre les processus se fait par le biais de messages ou de variables partagées.
Les communications synchrones et asynchrones sont modélisées comme deux cas particuliers d'un mécanisme général de passage de messages. Chaque instruction dans Promela peut potentiellement modéliser un retard : elle est exécutable ou non, dans la plupart des cas en fonction de l'état de l'environnement du processus en cours d'exécution.

Chapitre 3

Correspondance BPMN et Promela

3.1 Introduction

Dans ce chapitre, nous montrons comment traduire les éléments BPMN, dans un code PROMELA comportementalement équivalent. Pour ce faire, nous expliquons la sémantique de chaque opérateur, et nous décrivons de manière informelle comment la correspondence est réalisée, nous montrons en plus le résultat de l'application de ces idées à un exemple.

3.2 Tansformation des éléments BPMN en PROMELA

3.2.1 Événements

Événement initial

L'événement initial est un événement de contrôle auquel le flux démarre lorsque le diagramme est appelée et représente donc le début d'un diagramme de collaboration. La correspondence de cet événement est représentée par le processus initial appelé `init` en langage PROMELA (Figure 3.1).

Événement final

L'événement final représente la fin du comportement du système défini par son diagramme de collaboration donc c'est la fin du processus qui correspond à la fermeture du processus `}` (Figure 3.1).

Événement d'envoi/réception de message

Les deux événements d'envoi/réception de message représentent l'échange d'information entre deux processus, un émétteur et un récepteur. En Promela le code correspondant est dans la figure 3.1.

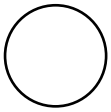
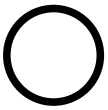
élément BPMN	*Code PROMELA*
"Événement initial" 	**init {**
"Événement final" 	**...}**
"Action d'envoi" 	**gate ! v**
"Action de réception" 	**gate ? x**

FIGURE 3.1 – Les évenements BPMN

3.2.2 Les activités

Les activités en BPMN 2.0 se décompose en deux types ; les tâches et les sous-processus. En générale les tâches peuvent être des actions à éxecuter. Les sous-processus peuvent être composé de plusieurs tâches ou des sous-processus parallèle ou des sous-processus événementiel c'est à dire un sous-processus instancié par un événement.

Activité	*Code PROMELA*
"Tâche simple" 	**proctype task()** **{ ... ... }**
"Tâche de réception" 	**proctype reception()** **{ ... gate ? x ... }**
"Tâche d'envoi" 	**proctype envoi()** **{ ... gate ! v ... }**
"Tâche service" 	**proctype service()** **{** /*code de l'activité*/ **... }**
"Tâche répétitive" 	**proctype boucle()** **{** do : : Condition -> corps de la boucle ; : : else -> break ; od ; **}**

FIGURE 3.2 – Les activités.

3.2.3 Les passerelles (gateway)

Les passerelles permet de contôler le flux d'orchestration. Les tableaux 3.1 et 3.2 présentent la transformation en code Promela des deux passerelles à savoir la passerelle exclusive et la passerelle parallèle.

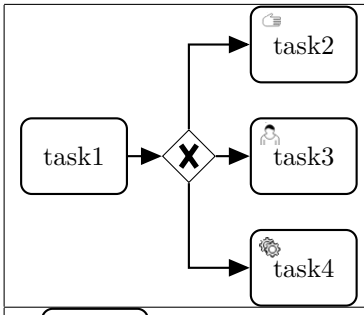
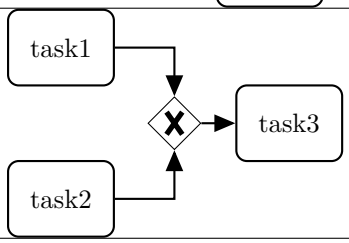
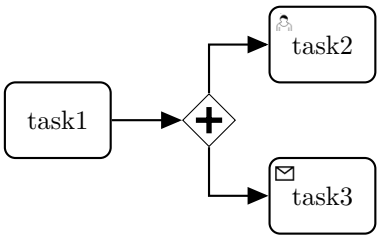
Passerelle Exclusive	Description	Code Promela
	Permet de diviser le flux en plusieurs chemins exclusifs en fonction d'une condition.	task1 ; if : : condition → task2 ; : : condition → task3 ; : : condition → task4 ; : : else → skip ; fi
	Permet de reconstruire un seul flux.	if : : cond → task1 ; goto E : : cond → task2 ; goto E : : else → skip ; fi E : task3 ;

TABLE 3.1 – Passerelle exclusive

Passerelle parallèle	Description	Code Promela

	Continuer le flux sur toutes les branches de sortie simultanément.	bool act2=false ;
		bool act3=false ;
		proctype activite2()
		{
		task2 ;
		act1=true
		}
		proctype activite3()
		{
		task3 ;
		act3=true
		}
		tast1 ;
		run activite2() ;
		run activite3() ;

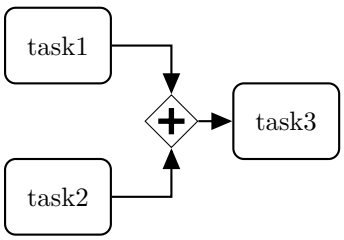
	Attendez que toutes les branches d'entrée se terminent.	bool act1=false, act2=false ;
		proctype activite1()
		{
		task1 ; act1=true ...
		}
		proctype activite2()
		{
		task2 ; act2=true ...
		}
		active proctype activite3() provided
		(act1==true)&&(act2==true)
		{ ...
		}

TABLE 3.2 – Passerelle parallèle

3.3 Exemple : Commande de pizza

La figure 3.3 montre le diagramme BPMN qui décrit le processus de commande d'une pizza[BPMN-Example], nous nous sommes limités au processus client. La spéci-

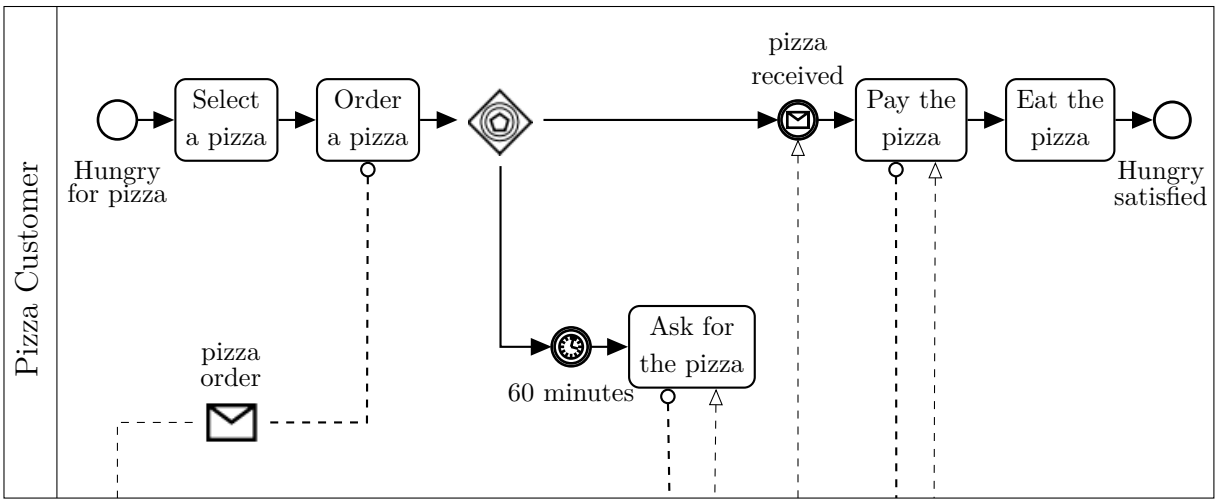


FIGURE 3.3 – Commande de pizza

fication Promela générée par l'approche proposée est la suivante :

```
mtype={temps}
mtype t= temps;
chan c=[1] of {mtype};
proctype Select_Pizza()
{
 run Order_Pizza();
}

proctype Order_Pizza()
{
 c!pizza;
 if
   :: assert(t>=60)-> run Ask_Pizza();
   :: assert(t<60) -> run Pizza_Received();
   else->skip;
 fi
}

proctype Ask_Pizza()
```

```
{
 run Order_Pizza();
}

proctype Pizza_Received()
{
 c?pizza;
 run Pay_Pizza();
 run Eat_Pizza();
}

init{
   run Select_Pizza()
}
```

Discussion :

Dans cet exemple de commande de pizza, on a déclaré un nouveau type nommé temps, puis on a créé une variable de même type. On a créé aussi un canal de communication pour l'échange des messages entre les processus.

Le processus *Select_Pizza* lance le processus *Order_Pizza*, dans lequel le client peut commander une pizza via le canal de communication *c*. Si le temps écoulé depuis le lancement de la commande (*c !pizza*) dépasse 60 unités de temps, le client recommande encore une fois sa pizza c'est-à-dire il reprend le processus *Order_Pizza*, sinon le processus *Pizza_Received* est lancé. Dans ce dernier le client reçoit la pizza demandée et exécute les deux tâches à savoir *Pay_Pizza* et *Eat_Pizza*.

3.4 Conclusion

Dans ce chapitre on a présenté une approche de transformation des éléments BPMN en Promela, cette transformation fait correspondre chaque éléments BPMN avec son interprétation en Promela. Enfin on a présenté un exemple en BPMN et le code Promela correspondant.

Chapitre 4

Étude de cas

4.1 Demande passeport biométrique

Dans cette section, on va présenter les détails de spécification BPMN pour le processus métier "Demande passeport biométrique", ce travail à été réalisé dans le cadre d'un travail de master[H2020]. Le processus commence par la réservation d'un rendez-vous auprès de n'importe quelle commune, en suite la saisie de la demande, l'enrolement des données biométrique et enfin la délivrance d'un récépissé. Après, le citoyen reçoit un SMS pour l'inviter à se présenter au lieu du dépôt du dossier pour recevoir leur passeport biométrique. Le diagramme de collaboration BPMN proposer contient deux piscines "Citoyen" et "Passeport biométrique". Le détail de ses processus est donné par la figure 4.1 :

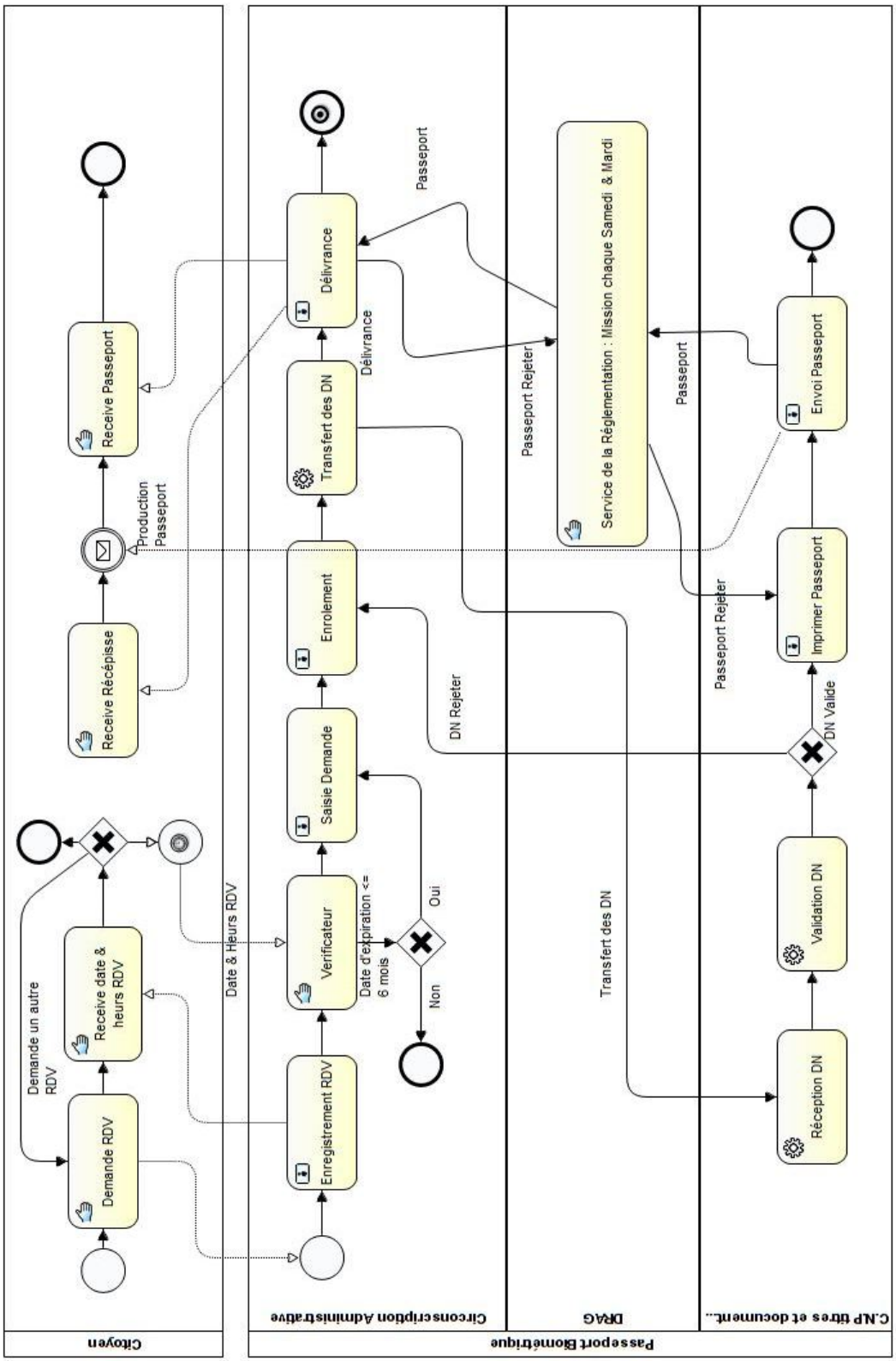


FIGURE 4.1 – Processus de demande passeport biométrique

4.1.1 Cycle enrôlement des données

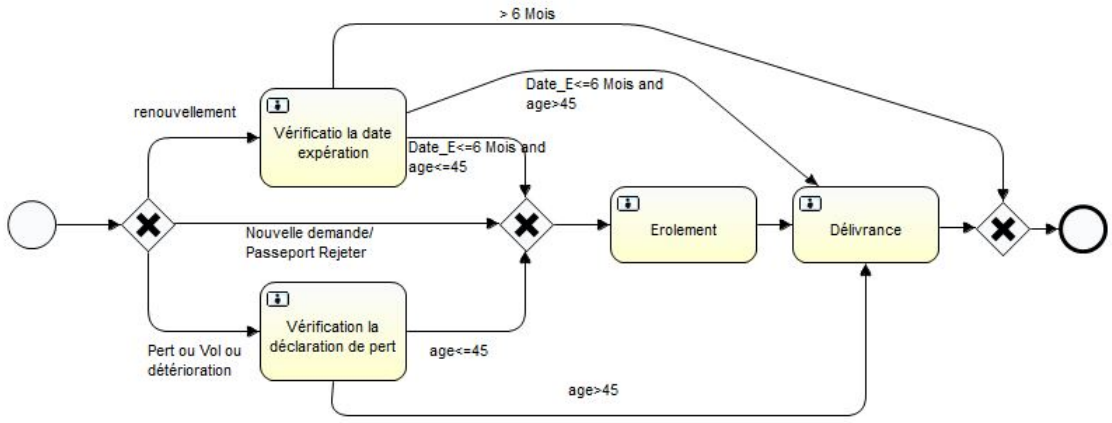


FIGURE 4.2 – Sous-processus cycle enrôlement

4.1.2 Cycle délivrance des passeports

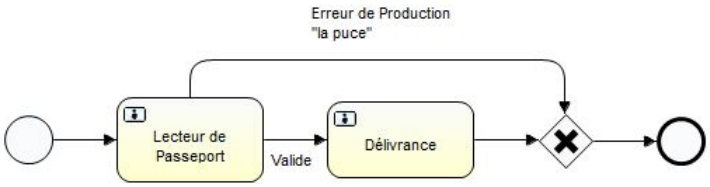


FIGURE 4.3 – Sous-processus cycle délivrance

4.2 Le code PROMELA

La spécification Promela générée par l'approche proposée est la suivante :

```
proctype Manuelle(){
 Demande RDV-> Receive date et heure RDV;
   if
   :: date et heure RDV->Verificateur;
```

```
   else->skip;
   fi
   }
proctype Utilisateur(){
 Enregistrement RDV->Verificateur;
   if
   :: (date d'expration<=6 mois)->Saisie demande;
   :: (date d'expration>6 mois)->end;
   else->skip;
   fi
   }
proctype Utilisateur_2(){
 Enrolement-> Transfert des DN;
   }
proctype Service(){
Transfert des DN;

 Transfert des DN->Délivrance;
 Transfert des DN->Réception DN;
   }
proctype Utilisateur_3(){
Délivrance;

 Production Passport->Receive Réception;
 Passport rejeter->Service de la Réglementation;
   }
proctype Service_2(){
 Réception DN-> Validation DN;
   }
proctype Service_3(){
 Validation DN;
  if
   :: DN Valide ->Imprimer Passport;
   :: DN Rejeter->Enrolement;
   else->skip;
   fi
   }
proctype Utilisateur_4(){
```

```
 Imprimer Passport-> Envoi Passport;
   }
proctype Utilisateur_5(){
 Envoi Passport;
  Passport-> Service de la Réglementation;
  Envoi Passport! Production Passport;
   }
proctype Manuelle_2(){
 Service de la Réglementation;
   if
   :: Passport-> Délivrance;
   :: Passport Rejeter-> Imprimer Passport;
   else->skip;
   fi
   }
init{
   run Manuelle();
   run Utilisateur();
   run Utilisateur_2();
   run Service();
   run Utilisateur_3();
   run Service_2();
   run Service_3();
   run Utilisateur_4();
   run Utilisateur_5();
   run Manuelle_2();

}
```

4.3 Conclusion

Dans ce chapitre nous avons présenté une étude de cas à savoir "Demande passeport biométrique", et on a proposé le code Promela correspondant.

Conclusion et perspectives

Dans le cadre de ce travail, nous nous sommes intéressés à la transformation des spécifications BPMN vers le langage PROMELA.

Nous avons présenté une introduction des deux langages suivie d'une représentation des approches de transformation de graphes permettant la génération d'un code PROMELA à partir dun diagramme BPMN.

L'approche proposée est capable de générer un code PROMELA à partir de diagramme de collaboration BPMN.

Comme perspective pour ce travail, c'est l'utilisation du code Promela généré pour une éventuelle vérification formelle en utilisant le modèle checker SPIN.

Bibliographie

[AGG] AGG Home page : http ://tfs.cs.tu-berlin.de/agg/

[JC2014] Julien Da Costa. BPMN 2.0 pour la modélisation et l'implémentation de dispositifs pédagogiques orientés processus.

[H2020] Nadia Hoggas. Vérification des processus métiers BPMN : approche logique.

[BPMN2] l'essentiel BPMN2, version révisée et mise à jour à 2016.

[Pierre18] Pierre Veyrat. Comprendre l'usage du BPMN et la signification de ses symboles, 25 octobre 2018.

[BPMN-Example] BPMN 2.0 by example, OMG, *http ://www.omg.org/spec/BPMN/2.0/examples/PDF*

[Larissa20] Larissa Lewis. Comprendre les diagrammes et les symboles du BPMN ,5 août 2020 .

[OMG] Business Process Model and Notation (BPMN) Version 2.0,janvier 2011.

[KATIRI10] Mohamed Yassine KATIRI. LA MODÉLISATION DES PROCESSUS D'AFFAIRES EN UTILISANT BPMN AINSI QUE QUALIGRAMME,29 JUILLET 2010.

[Silver09] Silver B. BPMN Method and Style : A levels-based methodology for BPM process modeling and improvement using BPMN 2.0,2009.

[Promela] Promela : https ://dept-info.labri.fr/ magoni/teleinfo/PolyPromela.pdf

[SVL] SVL — Introduction au Model-Checker Spin,20 avril 2008.

[FB2013] Francesco Belardinelli .Cours 6 : SPIN and Promela II ,30 octobre 2013.