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.

Vérification des processus métiers BPMN : approche logique

Présenté par :

HOGGAS Nadia

Dirigé par :

Dr MAAROUK Toufik Messaoud

Promotion : 2019/2020

Remerciement

Je tiens à remercier monsieur : MAAROUK Toufik Messaoud, Docteur à l'Université Abbes Laghrour de Khenchela d'avoir assuré l'encadrement de ce mémoire ainsi que pour tous ses conseils, sa confiance, ses idées, ses encouragements, ses corrections et ses remarques qui m'ont permis d'améliorer ce travail.

Je dédie ce mémoire :
À mes chers Parents.
À mon mari Nabil pour sa compréhension et son soutien.
À mes sœurs et mes enfants.
À tous mes amis qui m'ont soutenu.

Résumé

BPMN est la norme mondiale pour la modélisation des processus et l'une des composantes les plus importantes d'un alignement IT(information technology)-Business réussi. L'objectif principal de ce travail est de proposer une approche de vérification formelle par le modèle checking des diagrammes de collaboration. Dans ce contexte nous nous proposons une transformation des diagrammes de collaboration BPMN en un automate temporisé, ensuite le modèle checker UPPAAL est utilisé pour vérifier quelques propriétés comme l'absence de l'interblocage, la sûreté et la vivacité. L'idée est de prendre un diagramme de collaboration et de le transformer en spécification DD-LOTOS, cette spécification est interprétée par un C-DATA : une version étendue des automates temporisées et d'appliquer des méthodes de modèle checking.
Mots clé : BPMN, DD-LOTOS, UPPAAL, C-DATA.

Abstract

BPMN is the global standard for process modeling and one of the most important components of successful Business IT(information technology) Alignment. The main objective of this work is to propose a formal verification approach of collaboration diagrams by the model checking . In this context we propose a transformation of BPMN collaboration diagrams into a timed automaton, then the UPPAAL model checking is used to check some properties like the absence of deadlock, safety and liveness. The idea is to take a collaboration diagram and turn it into a specification DD-LOTOS, this specification is interpreted by a C-DATA : an extended version timed automaton and apply model checking methods.
Keywords :BPMN , DD-LOTOS, UPPAAL, C-DATA.

Table des matières

Table des figures

Liste des tableaux

Introduction générale

Lors de la réalisation d'un logiciel, nous sommes toujours appelés à assurer le bon fonctionnement de celui-ci afin d'éviter les catastrophes en termes humain et économique. C'est pourquoi plusieurs chercheurs se sont penchés davantage aux domaines liés à la vérification du logiciel dans le but de fournir des produits logiciels de meilleure qualité.

La bonne gestion d'une entreprise passe par la connaissance, la compréhension et le meilleur alignement possible de ses processus métier sur les objectifs de l'entreprise. Le concept de processus occupe aujourd'hui une place majeure dans le domaine des systèmes d'informations. Cependant, alors qu'il y a un quasi-consensus sur l'analyse et la représentation des informations, on observe une grande diversité dans l'utilisation de la notion de processus. Les niveaux de granularité sont variés, les méthodes et langages sont multiples et les outils incluent parfois leur propre approche.

L'objectif principal de BPMN(Business Process Model and Notation)[BPMN][OMG] est de fournir une notation qui est compréhensible par chaque intervenant du projet informatique : de l'analyste métier qui rédige la version initiale du processus, au client qui validera le processus, au développeur technique qui est responsable de l'implémentation technologique du logiciel qui exécutera ces processus puis finalement aux manageurs qui géreront et contrôleront l'efficacité du processus.

Dans ce travail nous utilisons le modèle formel C-DATA [MT2012] pour la modélisation des systèmes. Ce modèle suscitent pour l'utilisation dans la validation des systèmes temps réel, concurrents et distribués. Et notre contributions est basées sur l'interprétation des modèles BPMN par le modèle C-DATA .

L'objectif de ce travail est de vérifier quelques propriétés sur les modèles BPMN en utilisant les méthodes de modèle cheking , les travaux présentés portent sur les points suivants :

Premier approche :

— Interprétation des diagrammes BPMN en DD-LOTOS,
— Génération du modèle C-DATA à partir de spécification DD-LOTOS,
— Vérification a l'aide de model-checking UPPAAL.

Deuxième approche :

— Interprétation des diagrammes BPMN en automate temporisé.

Ce mémoire est structuré en deux parties :

Partie :État de l'art

▷ Le premier chapitre concernera les modèles formels, il présentera les langages de spécification LOTOS, DD-LOTOS et enfin le modèle C-DATA.
▷ Le deuxième chapitre comporte une description générale sur les processus métiers BPMN, leurs notations graphiques et diagrammes.

Partie :Contributions

▷ Le troisième chapitre Porterai sur l'interprétation des modèles BPMN en C-DATA.
▷ Le quatrième chapitre considère notre approche de la vérification des modèles BPMN.
Une étude de cas : demande un passeport biométrique est présentée.
Et nous nous concluons ce mémoire par des perspectives qui permettront de donner une suite à cette étude.

Première partie

État de l'art

Modèles formels de spécification

1.1 Introduction

Un modèle peut être qualifié de « formel » s'il est fondé sur une syntaxe et une sémantique précise, construite sur des bases théoriques pour démontrer des propriétés d'une spécification d'un système donné. La spécification doit permettre de décrire le comportement d'un système en décrivant ses propriétés importantes de façon abstraite, sans détails inutiles sans dire comment il sera obtenu (non-algorithmique)[KITOUNI]. Pour la spécification des propriétés, des exigences et le respect des contraintes, des langages de spécification basés sur les algèbres de processus sont traités tels que : ACP de Bergstra et al. [BK1984], CSP de Tony Hoare[Hoa1985], CCS de Robin Milner[Mil1980][Mil1989], MEIJE de R.de Simone [RS1985] et LOTOS [EB1988]. Ces langages formels ont une syntaxe et une sémantique bien définies ; les diagrammes syntaxiques peuvent être une syntaxe possible. La sémantique est décrite en algèbres, automates et systèmes de transitions, fonctions, relations et prédicats, ...

1.2 LOTOS (Language Of Temporal Ordering Specification)

LOTOS est une norme internationale[FDT87]. Conçu pour la spécification, la conception, et l'analyse fonctionnelle des systèmes concurrents. La spécification LOTOS possède deux parties clairement séparées. La première offre un modèle comportemental dérivé des algèbres de processus, principalement de CCS (Calculus of Communicating Systems), mais aussi de CSP (Communicating Sequential Processus). La deuxième partie du langage permet de décrire des types de données abstraites et des valeurs, et est basée sur le langage des types de données abstraits ACT-ONE [EM1985]. LOTOS utilise les concepts de processus, événements et expressions comportementales comme concepts de base pour la modélisation. Basic LOTOS le sous-ensemble de LOTOS où les processus interagissent entre eux par synchronisation pure, sans échange de va-

leurs. En Basic LOTOS, les actions correspondent aux portes de synchronisation des processus.

1.2.1 Syntaxe du langage LOTOS

LOTOS se présente comme suit[MT2012][KITOUNI][BN2005] : Soit PN l'ensemble des variables de processus parcouru par X et soit G l'ensemble des noms de portes définies par l'utilisateur (ensemble des actions observables) parcouru par g. Une porte observable particulière $\delta \in G$ est utilisée pour notifier la terminaison avec succès des processus. L dénote tout sous-ensemble de G, l'action interne est désignée par i. B parcouru par $E, F, \ldots$ dénote l'ensemble des expressions de comportement dont la syntaxe est :

$$\begin{array}{c} E ::= stop \quad | \quad exit \quad | \quad X\,[L] \quad | \quad g; E \\ | \quad i; E \quad | \quad E\,[]\,E \quad | \quad E|\,[L]\,|E \\ | \quad hide\ L\ in\ E \quad | \quad E \gg E \quad | \quad E\,[>\,E \end{array}$$

FIGURE 1.1 – Syntaxe du langage LOTOS

1.2.2 Présentation informelle de Basic-LOTOS

Basic-LOTOS est un sous ensemble de LOTOS où les processus interagissent entre eux par synchronisation pure, sans échange de valeurs. En Basic-LOTOS les actions sont identiques aux portes de synchronisation des processus. Les principaux opérateurs de Basic-LOTOS sont listés dans le tableau [MT2012] [RG2017] suivant :

Opération	Notation	Description informelle
Inaction	*stop*	Processus de base n'interagissant pas avec son environnement
Terminaison avec succés	*exit*	Processus qui se termine (action δ) et se transforme en *stop*
Préfixage par une action		
Non observable	$i; P$	Processus qui réalise l'action i ou g,puis se transforme en P
Observable	$g; P$	
Choix non-déterministe	$P_1 \,[]\, P_2$	Processus qui se transforme en P_1 ou en P_2 suivant l'environnement
Composition parallèle		
Cas général	$P_1 \vert [g_1, ..., g_n] \vert P_2$	$P1$ et $P2$ s'exécutent en parallèle et se synchronisent sur les portes $g_1, ..., g_n$ et δ
Asynchrone	$P_1 \vert\vert\vert P_2$	P_1 et P_2 s'exécutent en parallèle sans se synchroniser (sauf sur δ)
Synchrone	$P_1 \Vert P_2$	P_1 et P_2 s'exécutent en parallèle et se synchronisent sur chaque porte visible
Intériorisation	*hide* $g_1, ..., g_n$ *in* P	Les actions $g_1, ..., g_n$ sont cachées à l'environnement de P et deviennent des actions internes
Composition séquentielle	$P_1 \gg P_2$	P_2 est activé dès que P_1 se termine.
Préemption (interruption)	$P_1 [> P_2$	P_2 peut interrompre P_1 tant que P_1 ne s'est pas terminé

Table 1.1 – Principaux opérateurs de Basic-LOTOS

1.2.3 Sémantique formelle de Basic-LOTOS

— Soit P l'ensemble des identifiants de processus.
— Soit $X \in P$
— Soit G l'ensemble des portes définissables (les actions observables en Basic-LOTOS).
— Soient g, $g_1, ..., g_n \in G$.

— Soit L un sous-ensemble (pouvant être vide) quelconque de G noté $L = g_1...g_n$.
— Soit i l'action interne.

La syntaxe formelle du Basic-LOTOS [MT2012] est donnée par la figure 1.2 :

$$\textbf{Processus}\,[g_1, ..., g_n] := P\ \textbf{endproc}$$
$$\begin{aligned} \mathrm{P} := \mathrm{stop} \quad &| \quad exit \quad | \quad X\,[L] \quad | \quad g; P \\ &| \quad i; P \quad | \quad P\,[]\,P \quad | \quad P|\,[L]\,|P \\ &| \quad hide\ L\ in\ P \quad | \quad P \gg P \quad | \quad P\,[>\,P \end{aligned}$$

FIGURE 1.2 – Syntaxe du Basic-LOTOS

Notons que ni l'opérateur $|||$ ni l'opérateur $\|$ n'apparaissent explicitement dans la syntaxe. Ceux-ci correspondent à une notation simplifiée de l'opérateur de composition parallèle ($||| = |\,[\varnothing]\,|$ *et* $\| = |\,[G]\,|$,G désignant l'ensemble des portes visibles des deux processus composés).

1.3 Présentation de DD-LOTOS

La motivation majeure de DD-LOTOS [MT2012] est de fournir un modèle abstrait pour la programmation distribuée. Les auteurs s'intéressent aux modèles de processus qui représentent explicitement l'existence d'objets, nommés localités, domaine ou site, qui abstraient les lieux physiques, comme les ordinateurs, ou logiques, comme les agents. Une localité est un environnement d'activité de calcul ; les processus exécutent des actions et sont les responsables de l'évolution globale du système. Intuitivement, une localité héberge les ressources locales que les agents mobiles utilisent pour interagir les uns avec les autres. Pour leur permettre de remplir tous ces rôles, les localités sont nommées et structurées selon une structure plate, c'est une manière de programmer les localités explicitement. Ainsi une localité typique peut être décrite dans la figure 1.3 :

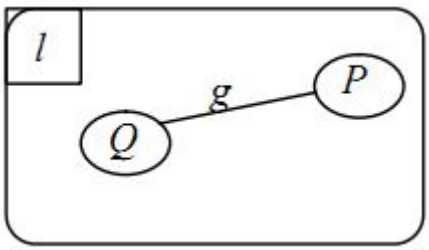


FIGURE 1.3 – Localité

P, Q représentent deux processus qui résident dans la localité l, et peuvent communiquer via la porte g. Un système distribué peut être vu comme une collection de localités indépendantes.
En effet, si nous analysons le comportement d'un émetteur, on constate que, après avoir transmis un message, il est prêt à attendre un accusé de réception pendant un certain temps (notion de durée d'une offre) ou de retransmettre le message lorsque ce temps s'écoule (notion de retard). Cette description peut être traduite par l'expression de comportement suivante :

$$transmit!m\{d\}; (ack\{wt\}; P \,[]\, \Delta^{wt} Q)$$

L'expression de comportement $transmit!m\{d\}$ tel que $d \in D$,(les éléments de D désignent des valeurs temporelles). signifie que l'action d'envoi $transmit!m$ ne sera pas offerte après d unités de temps, en d'autres termes cette action ne peut se produire que dans l'intervalle de temps [0, d].

1.3.1 Syntaxe

Pour prendre en considération les localités et la communication à distance, la syntaxe D-LOTOS est étendue comme suit :

E : :=	$\cdots$	**Comportements**
	$\|a!v\{d\}; E$	**Envoyer un message**
	$\|a?v; E$	**Recevoir un message**
S : :=		**Système**
	$\varnothing$	**Vide**
	$\|S\|S$	**Composition**
	$\|l(E)$	**Activité E dans la localité l**

TABLE 1.2 – Syntaxe du langage DD-LOTOS

Définition 1 (Action)

Les actions dans le système global [MT2012] sont :
— L'ensemble des actions de communication entre les localités : L'envoi et la réception de messages via les canaux de communication $Act_{com} ::= a!m|a?x|\tau$ (action d'envoi ou de production, action de réception et l'action silencieuse).
— L'ensemble $Act = \mathcal{G} \cup \{i, \delta\}$

Définition 2 (Localité et canaux de communication)

L'ensemble $\mathcal{L}$ parcouru par l, désigne l'ensemble des localités. ϑ un ensemble infini de canaux définis par les utilisateurs parcouru par $a, b, \cdots$ Les canaux sont utilisés pour l'échange de messages entre les localités. La syntaxe du calcul est donnée par le tableau 1.2 . Deux catégories d'expressions, la syntaxe des systèmes S et l'expression de comportement E. La sémantique informelle des expressions syntaxiques est la suivante :

- L'expression de comportement $a!v\,\{d\}\,;E$, spécifie l'envoi du message v via le canal de communication a. Cette opération d'envoi doit se produire dans l'intervalle temporel $[0.d]$. De l'autre côté, l'expression de comportement a ?x ;E, spécifie la réception d'un message sur le canal a, qui sera substitué à la variable x. Cette variable est utilisée dans l'expression de comportements E.
- Un système est défini par :
  - Vide, exprimé par $\varnothing$
  - La composition de sous-systèmes $S|S$, ou
  - Une expression de comportement E dans une localité l exprimée par $l(E)$

1.4 Durational Action Timed Automata(DATA)

Les modèles non temporisé supposent la non-atomicité des actions (temporelle et structurelle) que se soit ceux basés sur l'algèbre de processus ou issue des systèmes de transitions étiquetées, la sémantique sous-jacente est celle de l'entrelacement. Intuitivement elle suppose l'atomicité des actions et réduit l'exécution des actions parallèles à leur exécution entrelacer dans le temps. Cette hypothèse peut rendre invalides certaines spécifications et même le test exécuter sur la base de cette modélisation, de même les modèles temporisé qui formellement traitent trois types d'actions[BH2006][BB2005][BA1998] :

- Les actions observables
- Les actions internes
- Le passage de temps matérialisé par une durée

Le modèle des DATAs considère que chaque action dure dans le temps, cette quantité est capturée par la condition de durée sur un état, destination d'une transition étiquetée par cette action, toutes les transitions offertes sont possibles alors que toutes celles qui dépendent de la terminaison de l'action en cours sont inhibées jusqu'à ce que la condition d'exécution attribuée à ces transitions soit vérifiée. Les actions qui doivent s'exécuter en parallèle sont au contraire lancer sans être concerné par les conditions de durées. Dans ce cas la condition de durée offre explicitement l'information sur l'évolution du système. La condition de durée d'une action a décore chaque état atteint

par l'exécution de l'action a. Alors que toute transition est gardée par une condition d'exécution, c'est une conjonction sur des conditions de durée, elles concernent une transition dont le passage est conditionné par la terminaison d'un ensemble d'actions (dépendance causale dans le sens d'événements maximaux). Ce modèle est aux frontières des modèles temporisés est ceux non temporisés car les contraintes temporelles qu'il manipule expriment la réalité physique de tout système sans pour autant traiter les contraintes temps réel de celui ci.

Exemple

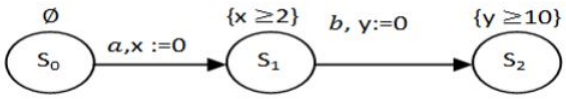


FIGURE 1.4 – Exemple d'un DATA

Ce DATA est composé de trois états et deux action a de durée 2 unités de temps et b de 10 unité de temps.A partir de l'état initial S_0 ,l'exécution de l'action a entraine une initialisation de l'horloge x qui lui est associée. $x >= 2$ est une condition de durée concernant a , elle est enregistrée dans l'état S_1 et qui signifie que a est en cours d'exécution,de même pour la condition $y >= 10$ relative à l'action b. La transition S_0 S_1 est franchissable ssi la condition sur la transition est vérifiée,celle-ci est dite condition d'exécution et elle signifie que b ne peut étre exécuter que si l'action a est achevée [HH2013][KITOUNI].

Définition 1

Un DATA -Durational Action Timed Automaton- A est un tuple (S, L_S, s_0, H, T) tel que :

— S est un ensemble fini d'états,

— $L_S : S \to 2^{\Phi_t(H)}_{fn}$ est une fonction qui correspond à chaque état s l'ensemble F des conditions de durées des actions potentiellement en exécution dans l'état s .

— $s_0 \in S$ est l'état initial,

— H est un ensemble fini d'horloges.

— $T \subseteq S \times \bigwedge 2^{\Phi_t(H)}_{fn} \times Act \times H \times S$ est l'ensemble des transitions. Une transition (s, γ, a, x, s') représente le passage de l'état s à l'état s', en lançant l'exécution d'une action a et en initialisant x à zero. γ est la contrainte temporelle qui conditionne le

passage de la transition. (s, γ, a, x, s') est aussi notée $s \xrightarrow{\gamma,a,x} s'$. L'ensemble de transition T parcouru par $\tau_0, \tau_1, ...$ étant donnée une transition $\tau_i = (s, \gamma, a, x, s'), \alpha(\tau_i)$ (resp.$\beta(\tau_i)$) représente l'état initial de τ_i (resp.l'état final de τ_i).(i.e. $\alpha(\tau_i) = s$ et $\beta(\tau_i) = s'$). L'étiquette de la transition τ_i est $\lambda(\tau_i) = a$. L'horloge associée à la transition τ_i est donnée par la fonction $\xi(\tau_i) = x$

1.5 Communicating Durational Action Timed Automata (C-DATA)

Dans le modèle C-DATA [MT2012] chaque localité est représenter par un DATA, le système globale est représenté par l'ensemble des DATAs locaux, qui communiquent entre-eux par échange de messages via des canaux de communications.

1.5.1 Définitions préliminaires

Définition 1

(actions) : Les actions dans le système global sont :

— L'ensemble des actions de communication entre les localités : L'émission et la réception des messages via les canaux de communication $Act_{com} ::= a!m|a?x|\tau$ (action d'émission ou de production, action de réception et l'action silencieuse).

— L'ensemble $Act = \mathcal{G} \cup \{i, \delta\}$.

Intuitivement, le terme a !m dénote l'émission du message m sur le canal a, le terme a ?x dénote la réception sur le canal a d'une valeur qui est affectée à la variable x, τ désigne l'action interne qui n'engendre aucune communication.

Définition 2

(Localités et canaux de communication) : L'ensemble $\mathcal{L}$ parcouru par l, désigne l'ensemble des localités. ϑ un ensemble infini de canaux définis par les utilisateurs. Cet ensemble est parcouru par a, b, etc. Les canaux sont utilisés pour l'échange de messages entre les localités.

Définition 3

Un Communicating DATA (C-DATA) est un septuple $A(S, L_s, s_0, \vartheta, H, \Pi, T_D)$ tel que :

— S est un ensemble fini d'états,

— $L_S : S \to 2^{\Phi_t(H)}_{fn}$ est une fonction qui fait correspondre à chaque état s l'ensemble F des conditions de terminaison des actions potentiellement en exécution dans s,
— $s_0 \in S$ est l'état initial,
— ϑ l'ensemble de l'alphabet des canaux de communication sur lesquels les messages transitent entre les sous-systèmes (DATA),
— H est un ensemble fini d'horloges,
— $\Pi = Act_{com} \cup Act$, est l'ensemble des actions locales et de communications,
— $T_D \subseteq S \times 2^{\Phi_t(H)}_{fn} \times 2^{\Phi_t(H)}_{fn} \times \Pi \times H \times S$ est l'ensemble des transitions.

Dans ce modèle une transition $(s, G, D, \alpha/(a(!/?)v)/\tau, z, s')$ représente le passage de l'état s à l'état s', en lançant l'exécution d'une action $\alpha \in Act$ ou action de communication (émission ou réception) ou synchronisation dans une communication (action silencieuse) et en réinitialisant l'horloge z. G est la contrainte correspondante, qui doit être satisfaite pour tirer cette transition. D est l'échéance correspondante qui exige, au moment de son expiration, que l'action α doit être tirée. $(s, G, D, \alpha/(a(!/?)v)/\tau, z, s')$ peut être écrite $s \xrightarrow{G,D,\alpha/(a(!/?)v)/\tau,z} s'$.

Définition 4

(Système) Un système $S = (A_1, ..., A_n)$, est un C-DATA, tel que chaque $A_i = (S_i, L_{S_i}, s_{0_i}, \vartheta, H_i, \Pi_i, T_{iD})$ $(1 \leq i \leq n)$ est un DATA.

Définition 5

On se donne les notation suivantes :

Etats : $GS(S) = (s_1, v_1) \times ... \times (s_n, v_n) \times (\vartheta^*)^p$ est l'ensemble des états.

Etat initial : L'état initial de S est : $q_0 = ((s_{01}, 0), ..., (s_{0n}, 0) : \epsilon_1, ..., \epsilon_p)$ tel que ϵ est le mot vide sur l'alphabet des canaux ϑ.

Etats de système : Soit $S = (A_1, ..., A_n)$ un système de n DATA ; tel que chaque $A_i = (S_i, L_{S_i}, s_{0_i}, \vartheta, H_i, \Pi_i, T_{iD})$ $(1 \leq i \leq n)$: L'état global de S est défini par l'état de chaque sous-système A_i et l'état de chaque canal, un état de S est un élément de $(s_1, v_1) \times ... \times (s_n, v_n) \times (\vartheta^*)^p$ tel que $v_i(h)$ sont les valuations sur H.

1.6 Conclusion

Nous avons présenté dans ce chapitre d'une manière général les modèles LOTOS et DD-LOTOS ses expressions de comportements, leurs syntaxes formelles. Ensuite on a présenté le modèle sémantique C-DATA. Dans le chapitre suivant, nous présentons le standard de modélisation des processus métier BPMN.

Notation BPMN

2.1 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. 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. Ces systèmes ont a leur tour créé une industrie pour le logiciel BPM (Business process management) qui a pour but d'automatiser la gestion des processus en reliant les divers acteurs des processus par le biais de la technologie.

2.2 Business Process Model and Notation

BPMN (Business Process Model and Notation) [BPMN]est une notation pour la représentation graphique des processus métier 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]. L'OMG est une association américaine à but non lucratif créée en 1989, dont l'objectif est de standardiser et promouvoir le modèle objet sous toutes ses formes. L'OMG [OMG] est un organisme influant dans l'industrie de conception logiciel et notamment à la base des standards

UML (Unified Modelling Language), MOF (Meta-Object Facility) ou CORBA (Common Object Request Broker Architecture). Elle est aussi à l'origine de la recommandation MDA (Model Driven Architecture) développant les principes de l'ingénierie dirigée par les modèles[ELM5432]. 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 .

2.3 Modélisation des processus métier

La modélisation des processus métier permet de représenter le fonctionnement d'un processus en définissant l'ensemble des activités à exécuter ainsi que leurs ordres d'exécution. Pour cela, il existe dans la littérature deux approches permettant de définir le comportement d'un processus. Une dite impérative et une dite déclarative. L'approche impérative se concentre sur la définition précise de la façon dont l'ensemble d'activités doivent être achevés, Pour cela, l'ordre d'exécution entre les différentes activités est décrit d'une façon explicite en utilisant un ensemble de liens ou de connecteurs, tandis que, l'approche déclarative se concentre plus sur "ce que devrait être fait" au lieu de "quoi faire", ce qui permet aux scénarios d'exécution de rester implicite dans la phase de modélisation du processus et cela en évitant d'énumérer explicitement tous les scénarios d'exécution possibles dans cette phase.

2.4 Représentation graphique des processus métiers

Il n'existe pas des règles absolues de représentation et de modélisation des processus métiers. Cependant, Il est important, dès le début de la modélisation, d'organiser les modèles en plusieurs niveaux ; d'un niveau élevé global et simple, vers des niveaux plus détaillés. Un diagaramme BPMN est composé d'un ensemble d'éléments graphiques qui permettent de modéliser les activités, les flux, les relations, les données ainsi que leurs interactions. Il 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 2.1 – Tâches d'un BPMN

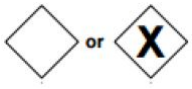
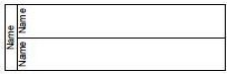
Les passerelles		
Passerelle exclusive	Un branchement exclusif évalue l'état du processus métier et, selon la condition définie, fait passer le flux sur un chemin en particulier, à l'exclusion du ou des autres.	
Passerelle parallèle	Les branchements parallèles sont utilisés pour représenter deux tâches 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 2.2 – 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 2.3 – Événements d'un BPMN

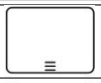
Activité en boucle		
Activité en boucle	Toutes les instances doivent se terminer , pour que l'activité soit terminée	
Sous processus en boucle	Un sous processus qui se répète tant qu'une condition n'est pas remplie.	
Activité multi-instance	**En parallèle**	**En séquence**
Activité MI		
Sous processus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

TABLE 2.4 – Tâches en boucle et MI d'un BPMN

2.4.1 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.

2.5 Diagramme BPMN

Le diagramme BPMN [BPMN][OMG]utilise les notations graphiques pour la conception d'un processus d'affaires, de sorte que tout le monde est capable de comprendre et de communiquer les procédures de manière standard, au sein de la notation BPMN2.0, on retrouve quatre catégories de diagrammes.

2.5.1 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.

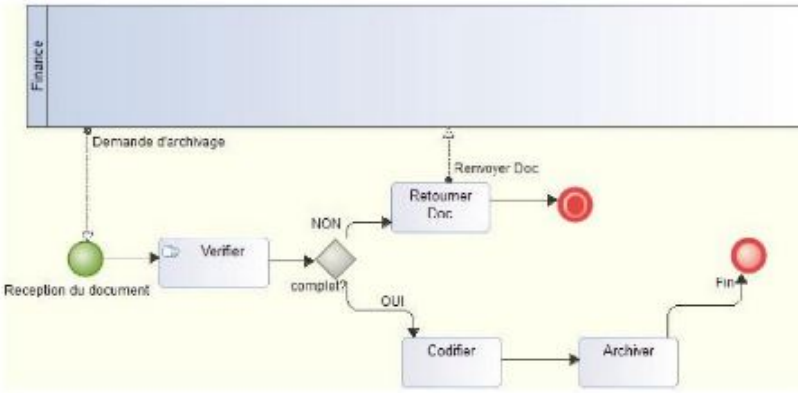


FIGURE 2.1 – Diagramme d'orchestration

2.5.2 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.

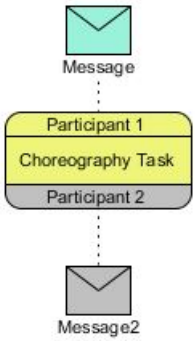


FIGURE 2.2 – Diagramme de chorégraphie

2.5.3 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 2.3 – Diagramme de conversation

2.5.4 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).

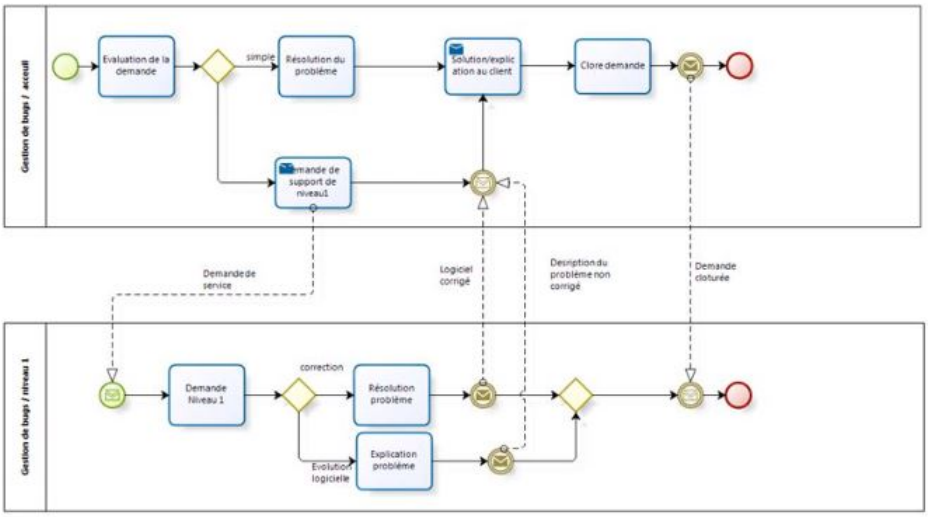


FIGURE 2.4 – Diagramme de Collaboration

2.6 Conclusion

BPMN est au coeur d'une approche de conception orientée services. Les systèmes de gestion des processus métier (BPM systems ou BPMS) offrent une plateforme de développement apte au développement d'applications flexibles, exécutables sur différentes plateformes et interconnectable avec tous services externes. L'objectif de BPMN est de supporter la gestion des processus métier pour les utilisateurs techniques et métier, tout en étant capable de représenter des sémantiques complexes de processus.

Deuxième partie

Contributions

Interprétation des modèles BPMN en C-DATA

3.1 Introduction

Les méthodes formels reposent sur l'utilisation de langages formels. Un langage formel est un langage doté d'une sémantique mathématique rigoureuse. Pour démontrer l'applicabilité de l'approche proposée, nous montrons maintenant quelle sont les étapes a suivre pour interpréter un système modéliser en BPMN en modèle C-DATA

3.2 Interprétation des diagrammes BPMN en DD-LOTOS

Dans[RG2017] à été proposé une sémantique des éléments BPMN en terme du langage formel DD-LOTOS, le tableau suivant récapitule la correspondance entre BPMN et DD-LOTOS

Elément BPMN		Interprétation formelle
Les objets de flux		
Activité		Selon son type
Tache "tache"		Tcahe
Type	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
	tache d'appel	Tache
	Script"script"	Script
Sous processus "proc"		processus proc(porte) := endproc ;
Type	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 ;
Evènement		Selon son type
	Début	Selon son type
	Fin	Selon son type
	Intérmédiaire	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	$([p(X)] -> chemin1\, en\, DD - Lotos)$ [] $[not(p(X))] -> chemin2\, en\, DD - Lotos$ [] chemin de sortie obligatoire
	Passerelle parallèle pour la synchronisation de plusieurs chemins parallèles	chemin1 en DD-Lotos||| chemin2 en DD-Lotos $\|\|\| \cdots \|\|\|$ cheminN en DD-Lotos

	Passerelle inclusive	$([p(X)] -> chemin1\, en\, DD - Lotos)\ []$ $([G(X)] -> chemin2\, en\, DD - Lotos)$
	Passerelle évènementielle exclusive	$([Ev1] -> chemin1)\ []\ ([Ev2] ->$ $-> chemin2)\ [] \cdots []\ ([EvN] -> cheminN)$
	Passerelle Complexe	Selon la spécification
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	$(Ev1\ []\ Ev2\ [] \cdots []\ Ev3) ->$
	Conditionnel	$[prédicat] ->$
	parallèle	$(Ev1 \vert\vert\vert Ev2 \vert\vert\vert Ev3) ->$
Evènement de fin		
	Simple	exit
	Envoi d'un message	a !message ;exit
	Envoi d'un signal	a !signal ;exit
	Erreur	Stop
	Cancel	$[> exit$
	Multiple	$(Ev1\ []\ Ev2\ [] \cdots []\ Ev3) -> exit$
	Parallèle	$(Ev1 \vert\vert\vert Ev2 \vert\vert\vert Ev3) -> exit$

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	$(Ev1 \,[]\, Ev2 \,[]\, \cdots \,[]\, Ev3)$
	Parallèle	$(Ev1 \| \| \| Ev2 \| \| \| Ev3)$

3.3 Génération du modèle C-DATA à partir de spécification DD-LOTOS

3.3.1 Construction opérationnelle des C-DATA's

Définition 6 [MT2012]

Soit $S = (A_1, ..., A_n)$ un système de n C-DATA ;
tel que chaque $A_i = (S_i, L_{S_i}, s_{0_i}, \vartheta, H_i, \Pi_i, T_{iD})$ $(1 \leq i \leq n)$. La transition T entre létat $s = ((q_1, v_1), ..., (q_n, v_n) : x_1, ..., x_p)$ et l'état $s' = ((q'_1, v'_1), ..., (q'_n, v'_n) : x'_1, ..., x'_p)$ peut être soit une émission (RE), une réception (RR), une exécution d'une action locale (RL), un passage de temps (RP), une synchronisation par rendez-vous (RS). La sémantique du système S est définie par la plus petite relation de transition satisfaisant les règles suivantes :

1. (RE règle d'émission)
$$\frac{((q_i,v_i),a!v,(q'_i,v'_i))\in T_D \qquad x_j \textit{ mot du canal } a}{(...,(q_i,v_i),...:...,x_j,...) \xrightarrow{a!v} (...,(q'_i,v'_i),...:...,x_j.v,...)}$$

2. (RR règle de réception)
$$\frac{((q_i,v_i),a?x,(q'_i,v'_i))\in T_D \qquad x_j \textit{ mot du canal } a}{(...,(q_i,v_i),...:...,x_j.v,...) \xrightarrow{a?x} (...,(q'_i,v'_i),...:...,x_j,...)}$$

3. (RL exécution d'une action locale, c'est à dire n'affecte pas le reste du système)
$$\frac{((q_i,v_i),\alpha,G,D,z,(q'_i,v'_i))\in T_D \qquad v\models G}{(...,(q_i,v_i),...:...,x_i,...) \xrightarrow{\alpha} (...,(q'_i,v'_i),...:...,x_i,...)}$$
où G est une contrainte temporal ou garde , D est l'échéance correspondante qui exige, au moment de sa satisfaction, que l'action α doit être tirée, z l'horloge qui doit être réinitialisée.

4. (RP passage de temps)
$$\frac{d \in R^{+} \qquad \forall d' \leq d \qquad (v_i + d) \not\models D}{(\ldots,(q_i,v_i),\ldots:x_1,\ldots,x_p) \xrightarrow{d} (\ldots,(q_i,v_i+d),\ldots:x_1,\ldots,x_p)}$$

5. (RS pour cette règle les canaux x_i et x_j ne contiennent aucun messages)
$$\frac{((q_i,v_i),a?x,(q'_i,v'_i)) \in T_D \qquad ((q_j,v_j),a!v,(q'_j,v'_j)) \in T_D \qquad i \neq j}{(\ldots,(q_i,v_i),\ldots,(q_j,v_j),\ldots:x_1,\ldots,x_p) \xrightarrow{\tau} (\ldots,(q'_i,v'_i),\ldots,(q'_j,v'_j),\ldots:x_1,\ldots,x_p)}$$

3.3.2 Approche de génération du modèle C-DATA

Pour générer le modèle C-DATA à partir d'une spécification DD-LOTOS on doit avoir :

— La spécification complète du système en DD-LOTOS.
— Les sous-systèmes qui composent le système complète.
— Les localités du système.
— Les types d'actions dans chaque localité.
— Sélectionner Les règles à appliquer de modèle C-DATA
— Application des règles de modèle C-DATA sur la spécification initiale jusqu'à ce que plus aucune règle ne soit applicable.
— Le résultat c'est un modèle C-DATA (ensemble de configurations et conditions)

3.3.3 Présentation des correspondances entre une spécification en DD-LOTOS et le modèle C-DATA

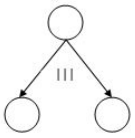
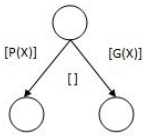
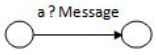
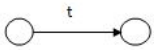
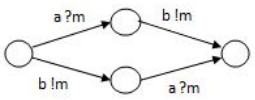
Spécification DD-LOTOS	C-DATA
$L(P)$	Sous graphe
Specification :=	Nœud initial
$Exit$	Nœud final
$Stop$	Nœud final
$Arcs$	Flot de contrôle
$\|\|\|$	
$([p(X)] -> chemin1\, en\, DD - Lotos)$ [] $([G(X)] -> chemin2\, en\, DD - Lotos)$	
a ?Message	
a !Message	
$\triangle^t$	
$a?m\|b!m$	

TABLE 3.1 – Correspondances entre DD-LOTOS et C-DATA

3.3.4 Algorithme de transformation spécification en DD-LOTOS vers C-DATA

Algorithme : Transformation Spécification en DD-LOTOS vers graphe(C-DATA)
Entrée : Spécification en DD-LOTOS
Sortie : Graphe(C-DATA)
Début
Pour chaque ligne de la spécification **faire**

- ◇ Appliquer des règles de modèle C-DATA sur la spécification initiale jusqu'à ce que plus aucune règle ne soit applicable.
- ◇ Extraire les configurations obtenues.
- ◇ Aller à la ligne suivante de la spécification.

FinPour
Fin

Exemple : Échange de message[MT2012]

Soit le processus E qui reçoit l'information($RqData$) sur le canal a, puis il attend pour une période de t unités de temps. Le processus offre le message m sur le canal b pour une période de d unités de temps, et termine son exécution en exécutant le processus $exit$:
L'expression de comportement pourrait être spécifiée comme suit

$$E ::= a?RqData(\triangle^t(b!m\,\{d\}\,;exit))$$

C-DATA pour cette exemple :

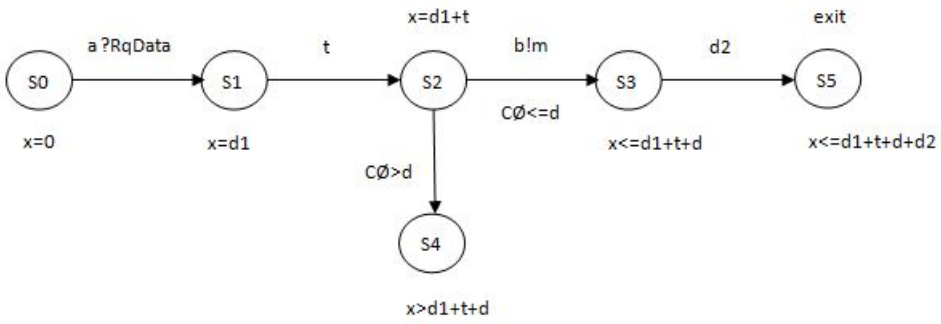


FIGURE 3.1 – C-DATA : Échange de message

3.4 Deuxième approche : interprétation des diagrammes BPMN en automate temporisé

3.4.1 Automate Temporisé

Les automates temporisés ont été introduits par R. Alur et D. Dill[AD1990][AD1994] dans les années 1990. Il s'agit d'automates classiques munis d'horloges qui évoluent de manière continue et synchrone avec le temps. Chaque transition contient une garde (sur la valeur des horloges) décrivant quand la transition peut être exécutée et un ensemble d'horloges qui doivent être remises à zéro lors du franchissement de la transition. Chaque état de contrôle contient un invariant (une contrainte sur les horloges) qui peut restreindre le temps d'attente dans l'état et donc forcer l'exécution d'une transition d'action.

Définition 1

Un automate temporisé A est un sixtuplet $(\sum, S, s_0, X, I, T)$ tel que :

— $\sum$ un ensemble fini d'étiquettes de transition.
— S un ensemble fini d'états.
— $s_0 \in S$ un état initial.
— X un ensemble fini de variable à valeur dans $\mathbb{R}_+$ (horloges).
— $I : S \to C(X)$ est une fonction qui fait correspondre à un état s l'ensemble des conditions sur l'état appelées Invariant et
— $T \subseteq S \times C(X) \times \sum \times 2^X \times S$ est l'ensemble de transitions. Une transition $e =< s, \varphi, a, \lambda, s' >\in T$(notée $s \xrightarrow{\varphi,a,\lambda} s'$) représente le passage de l'état s à l'état s', gardée par la contrainte φ , étiquetée par a, et qui réinitialise les variables $\lambda \subset X$

3.4.2 Approche de transformation

Nous proposons une approche de transformation qui permet d'interpréter un diagramme BPMN en un automate temporisé TA. Essentiellement :

processus métier : c'est un ensemble de tâches.

Tâche : est une unité atomique réalisées par une ressource.

Flux d'exception : représente un occurrence d'erreur dans l'état.

Flux de message : représente la communication entre les objets de flux.

Etats : peuvent être une tâche, un événement ou un passerelle.

Etat initial : un événement de départ.

Transitions : flux de séquence d'exception ou flux de message.

Garde : flux de séquence peuvent être soit entrant ou sortant d'un état.

Pour transformer un processus métier décrit par BPMN en réseau TA, il faut qu'il soit modélisé en diagramme de collaboration, le comportement de chaque participant (représenté par un pool dans un processus métier) est modélisé en tant que système de transition d'états ou TA.

Exemple : Commande pizza

Le processus métier " Commander une pizza dans un restaurant " peut être modélisée en BPMN dans la figure suivante :

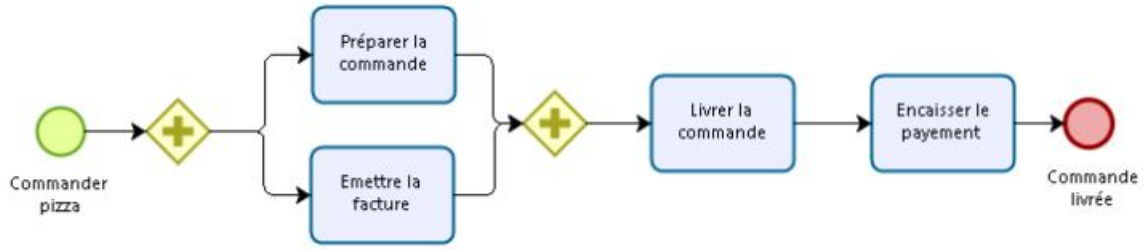


FIGURE 3.2 – Commande pizza

Commande pizza en DD-LOTOS

L'expression de comportement du processus : commande pizza **E** est la suivante :

$$E ::= E1$$
$$E1 ::= CommanderPizza; PréparerCommande \| EmettreFacture;$$
$$LivrerCommande; EncaisserPayement; exit$$

Et la spécification de notre exemple en DD-LOTOS :

Specification *CommandePizza*
l(E)
Where
Process E : := E1
Where
E1 : := CommanderPizza ;PréparerCommande $\|EmettreFacture$;
LivrerCommande ;EncaisserPayement ;exit
Endproc
Endspec

Commande pizza en automate temporisé

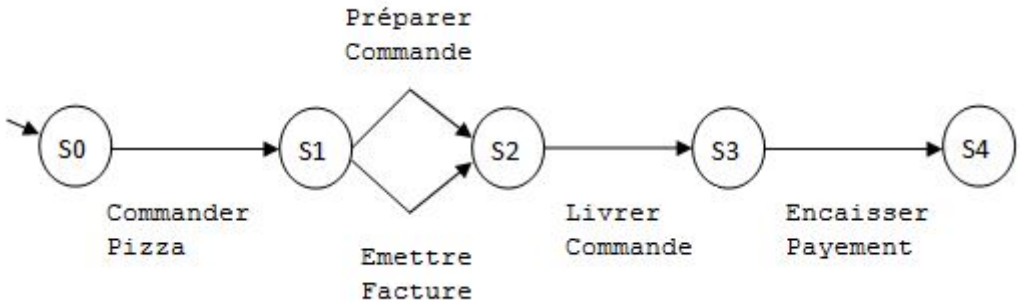


FIGURE 3.3 – TA : Commande pizza

3.5 Conclusion

La cohérence et la fiabilité des processus métier est une question primordiale pour toute organisation dans accomplissement de ses fonctionnalités. Un processus peu fiable ou erroné peut souvent aboutir à un dysfonctionnement dans la gestion du système d'information. Dans le chapitre suivant, nous tenons la vérification par model-checking UPPAL

Vérification des modèles BPMN par modèle checking

4.1 Introduction

La technique de vérification par modèles formels consiste à fournir en plus de la représentation graphique, une sémantique attribuée au comportement du processus modélisé qui peut manquer dans des standards de notation tels que BPMN et cela afin de vérifier le bon fonctionnement de ce dernier.

4.2 UPPAAL

UPPAAL [BA2010] est un ensemble d'outils pour la vérification automatique des propriétés de sûreté et de vivacité bornée, des systèmes temps réel. La machine (ou noyau) UPPAAL est le serveur, elle est développée en C++. L'interface graphique (GUI) est le client, développé en JAVA. La communication s'effectue via des protocoles internes. Ainsi, la conception d'UPPAAL offre la possibilité d'exécuter le serveur et l'interface GUI sur deux machines différentes.
La base du modèle d'UPPAAL est la notion des automates temporisés, développée par R. Alur et D. Dill[AD1990][AD1994], comme extension des automates à états-finis classiques avec les variables d'horloges et de données. Pour avoir une modélisation plus expressive et pour la rendre plus facile, les automates temporisés sont étendus avec des types plus généraux des variables de données tel que les variables booléennes et entières. Le but est le développement d'un langage de modélisation le plus proche que possible d'un langage de programmation des systèmes temps réel de haut niveau.

4.2.1 Description UPPAAL

Se compose de trois parties principales :

1. Un éditeur

2. Un simulateur,
3. Un vérificateur de modèle

Editeur : permet la modélisation du système qui est défini comme un réseau d'automates temporisés, appelés processus, dans l'outil, mis en parallèle. Un processus est instancié d'une classe paramétrée (Template). L'éditeur est divisé en deux parties : un volet pour accéder aux différents Templates et déclarations et un canevas pour dessiner l'automate.

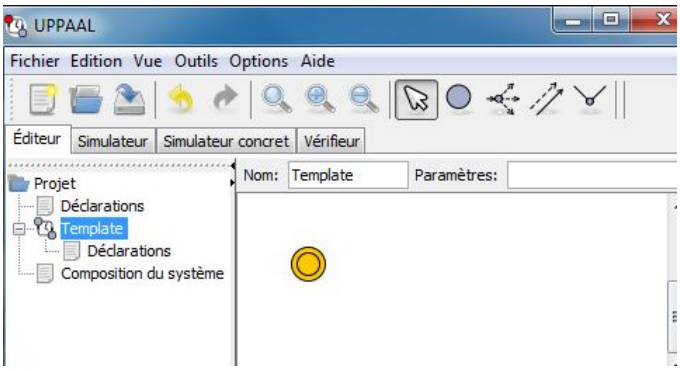


FIGURE 4.1 – Editeur UPPAAL

Le simulateur : est un outil de validation qui permet l'examen dynamique des exécutions possibles d'un système au début de la conception (ou de modélisation). Il offre ainsi un moyen peu coûteux de détection des défauts avant la vérification exhaustive qui couvre le comportement dynamique complet du système.

Vérificateur : permet l'introduction d'une propriété CTL et la vérifie. Quand l'option "génération de trace" est activée, elle sera importée au simulateur sous la demande de l'utilisateur. Les propriétés satisfaites sont marquées en vert et celles violées en rouge.

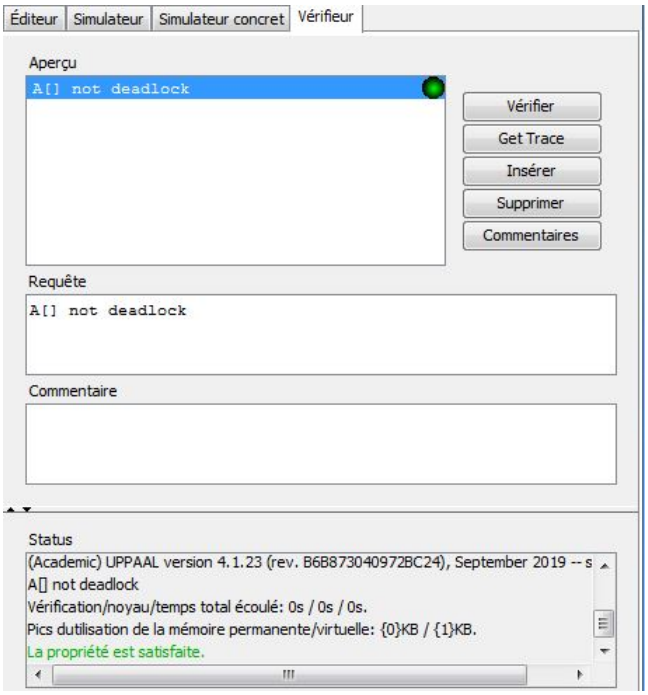


FIGURE 4.2 – Vérification d'UPPAAL

4.2.2 Vérification avec UPPAAL

UPPAAL est un outil pour la modélisation, la validation et la vérification des systèmes temps réel. Le but principal d'un modèle checker est de vérifier sur le modèle sémantique une propriété. Comme le modèle, la spécification de la propriété doit être exprimée formellement dans un logique temporelle . Plusieurs logiques existent dans la littérature ; UPPAAL utilise un sous ensemble de la logique TCTL.

4.2.3 La logique TCTL

La logique TCTL (Timed CTL) est une extension proposée dans [ACD93] de la logique CTL, afin d'énoncer des propriétés temporisées, i.e qui font intervenir des informations quantitatives sur le temps, par exemple :
"La barrière s'ouvre en moins de 5 secondes à l'approche d'une voiture"

La syntaxe TCTL

La syntaxe de la logique TCTL est donnée par :
— Formules **TCTL** sur les états : $\Phi ::= true|a|g|\Phi_1 \wedge \Phi_2|\neg\Phi_1|\exists\varphi|\forall\varphi$
— Formules **TCTL** sur les états :$\varphi ::= \Phi_1 \cup^J \Phi_2|\ \mathrm{F}^J\Phi|\ \mathrm{G}^J\Phi$

Propriétés à tester UPPAAL

Uppaal capable de vérifier les propriétés d'atteignabilité, de sûreté, de vivacité et de non-blocage.

Atteignabilité : Il s'agit de vérifier si une formule d'état donnée φ peut être satisfaite par un état accessible quelconque. Dans UPPAAL, nous écrivons cette propriété en utilisant la syntaxe : $E <> \varphi$.

Sûreté : exprimant le fait que le systèm ne sera jamais dans un état indésirable "quelque chose de mauvais n'arrivera jamais". Par exemple, dans un modèle d'une centrale électrique nucléaire, une propriété de sûreté peut éxiger que la température du fonctionnement est toujours sous un certain seuil (invariant). Dans UPPAAL, nous écrivons cette propriété en utilisant la syntaxe : $E [] \varphi$, $A [] \varphi$.

Vivacité : exprimant le fait que le système atteindra un certain état désirable peuvent être décrites comme des assertions de potentialité "quelque chose arrivera finalement", par exemple, si nous appuyons sur le bouton de la télécommande, alors la télévision finira par s'allume. Dans UPPAAL, nous écrivons cette propriété en utilisant la syntaxe : $A <> \varphi$

Interblocage : Un état est un état d'interblocage s'il n'y a ni de transitions d'action sortantes ni de successeurs de délai.Dans UPPAAL, nous écrivons cette propriété en utilisant la syntaxe : $A [] notdeadlock$

4.3 Étude de cas : Demande passeport biométrique

Dans cette section, on va présenter les détails de spécification BPMN pour le processus métier 'Demande passeport biométrique'. Le processus commence par la réservation d'un rendez-vous auprès de n'importe quelle commune, circonscription administrative ou auprès du service consulaire du lieu de résidence pour les algériens résidents à l'étranger 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 4.3 :

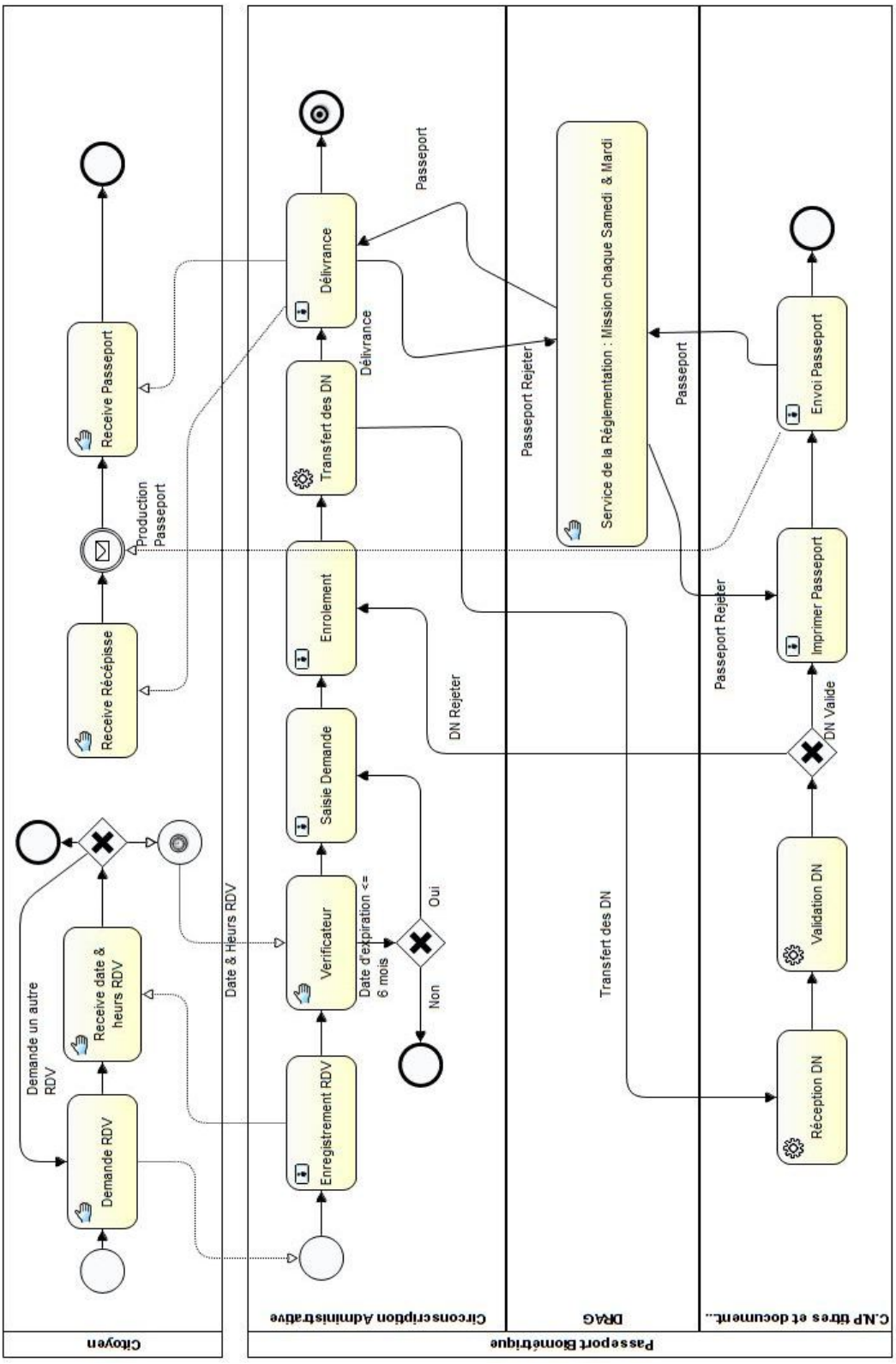


FIGURE 4.3 – Processus de demande passeport biométrique

4.3.1 Cycle enrôlement des données

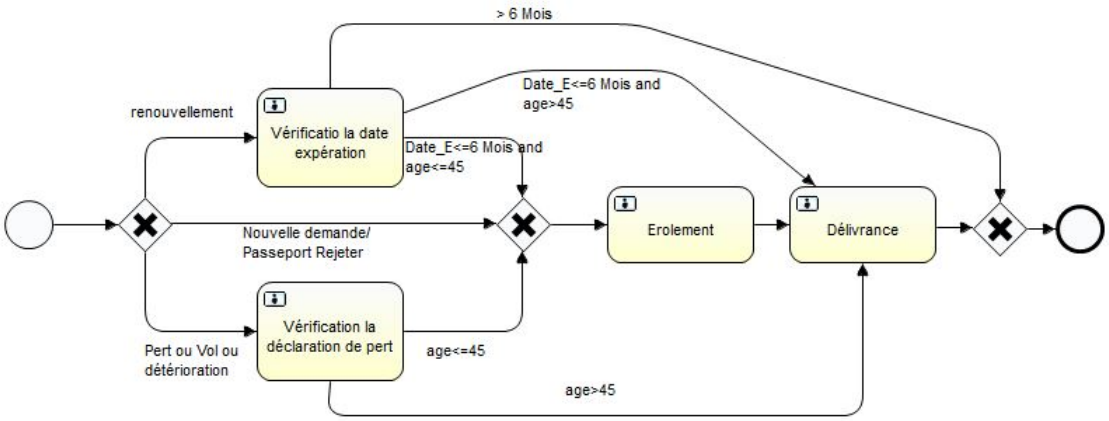


FIGURE 4.4 – Sous-processus cycle enrôlement

4.3.2 Cycle délivrance des passeports

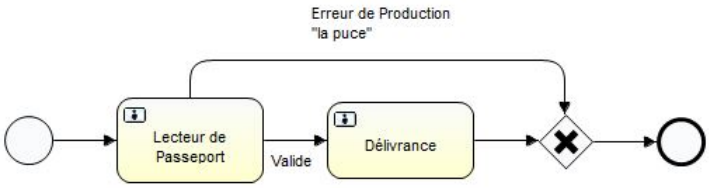


FIGURE 4.5 – Sous-processus cycle délivrance

4.3.3 Demande passeport biométrique en spécification DD-LOTOS :

Specification DemandePasseportBiometrique ::=
L(Enrolement des données) ||| L(Delivrance Passeport)
process Enrolement des données (RDV)::=
verification la liste des RDV
$[RDV =' Non'] - >$ exit
$[RDV =' Oui'] - >$ SaisieDemande ;
$[renouvellement =' Oui'] - >$
verification la date expération du passeport
$[date\ d'expiration\ plus\ de\ 6\ mois =' Non'] - >$ exit
$[date\ d'expiration\ moins\ de\ 6\ mois\ et\ l'age\ plus\ de\ 45\ ans =' Oui'] - >$
Delivrance ;exit
$[date\ d'expiration\ moins\ de\ 6\ mois\ et\ l'age\ inférieur\ ou\ égal\ 45\ ans =' Oui'] - >$
Enrolemet ;Delivrance ;exit
[ ]
$[Pert - Vol =' Oui'] - >$
$[l'age\ plus\ de\ 45\ ans =' Oui'] - >$ Delivrance ;exit
$[l'age\ inférieur\ ou\ égal\ 45\ ans =' Oui'] - >$ Enrolemet ;Delivrance ;exit
[ ]
$[Nouvelle\ Demande =' Oui'] - >$ Enrolemet ;Delivrance ;exit
[ ]
$[Dossier\ Rejeter =' Oui'] - >$ Enrolemet ;exit
EndProc
process Delivrance Passeport(SMS)::=
Lecture de passeport par le lecteur MRZ
$[passeport\ valide =' Non'] - >$ exit
$[passeport\ valide =' Oui'] - >$ DelPass ;exit
EndProc
EndSpac

4.3.4 Demande passeport biométrique en C-DATA

Notre Système : Demande passeport biométrique **E** est composée de deux sous activités E1 et E2, qui sont spécifiées par les expressions comportementales suivante :

$$E ::= E1|||E2$$

E1 : := Saisiedemande ; Enrolement ;Delivrance ;exit
E2 : := LecturePass ;DelPass ;exit

Et la spécification de notre système en DD-LOTOS :

Specification*DemandePasseportBiometrique*
l(E)
Where
Process E : := E1 $|||E2$
Where
E1 : := Saisiedemande ; Enrolement ;Delivrance ;exit
E2 : := LecturePass ;DelPass ;exit
Endproc
Endspec

Comme toutes les tâches de notre système sont des actions observables et l'expression de comportement représente une composition parallèle de deux expressions, elle se traduit par :

$$_{\phi}\,[E] ::=_{\phi} [E1] \,|||_{\phi}\, [E2]$$

On appliquons la règle (RL : exécution d'une action locale) du modèle C-DATA

$$\underbrace{_{\phi}\,[E1] \,|||_{\phi}\, [E2]}_{config0} \xrightarrow{RDV} \underbrace{(Saisiedemande; E1)|||_{\phi}\, [E2]}_{config1}$$

Dans le cas d'un dossier numérique rejeter on aura la configuration :$config2$

$$config0 \xrightarrow{DN-rejeter} \underbrace{(Enrolement; E1)|||_{\phi}\, [E2]}_{config2}$$

Dans le cas d'une nouvelle demande on aura la configuration :$config2$

$$config1 \xrightarrow{Nouvelle\ Demande} config2$$

$$config2 \xrightarrow{Citoyen\ Enroler} \underbrace{(Delivrance; E1)|||_{\phi}\, [E2]}_{config4}$$

$$config4 \xrightarrow{Recepisse\ Délivrer} config5$$

Dans le cas de renouvellement on aura une action implicite de renouvellement :

$$config1 \xrightarrow{DateE < 6\ mois} \underbrace{(renouvellement; E1)|||_{\phi}\,[E2]}_{config3}$$

$$config1 \xrightarrow{Pert\ ou\ Vol} config3$$

$$config3 \xrightarrow{age \leq 45\ ans} config2$$

$$config3 \xrightarrow{age > 45\ ans} config4$$

Dans le cas où le système démarré l'expression de comportement $E2$ on aura les configurations :

$$\underbrace{\phi\,[E1]\,|||_{\phi}\,[E2]}_{config0} \xrightarrow{Recepisse} \underbrace{\phi\,[E1]\,|||(LecturePass; E2)}_{config6}$$

$$config6 \xrightarrow{Pass-Valide} \underbrace{\phi\,[E1]\,|||(DelPass; E2)}_{config7}$$

$$config7 \xrightarrow{Pass-Délivrer} \underbrace{\phi\,[E1]\,|||(exit; E2)}_{config8}$$

C-DATA

Le résultat de la génération de modèle C-DATA a partir du système "Demande passeport biométrique" est un graphe ou les sommets sont les configurations et les arcs étiqueté par les conditions ou bien les gardes :

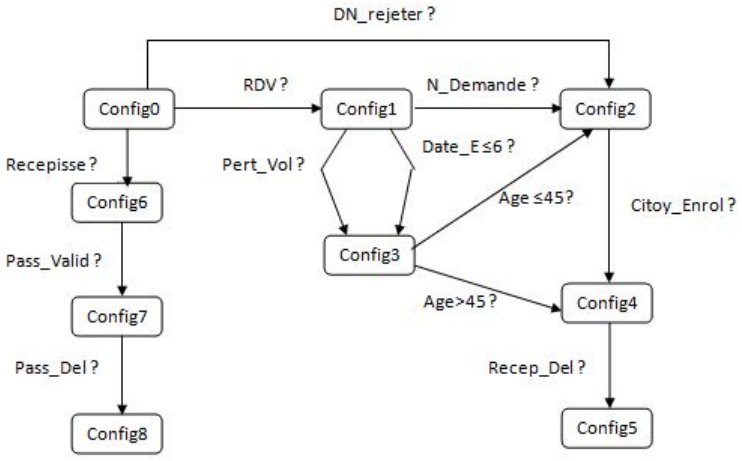


FIGURE 4.6 – C-DATA :Demande passeport biométrique

4.3.5 Demande passeport biométrique en automate temporisé

Notre processus métier "Demande passeport biométrique" représente un ensemble d'automates temporisés combiné en un seul système par la synchronisation de trois TAs : citoyen, passeport et délivrance

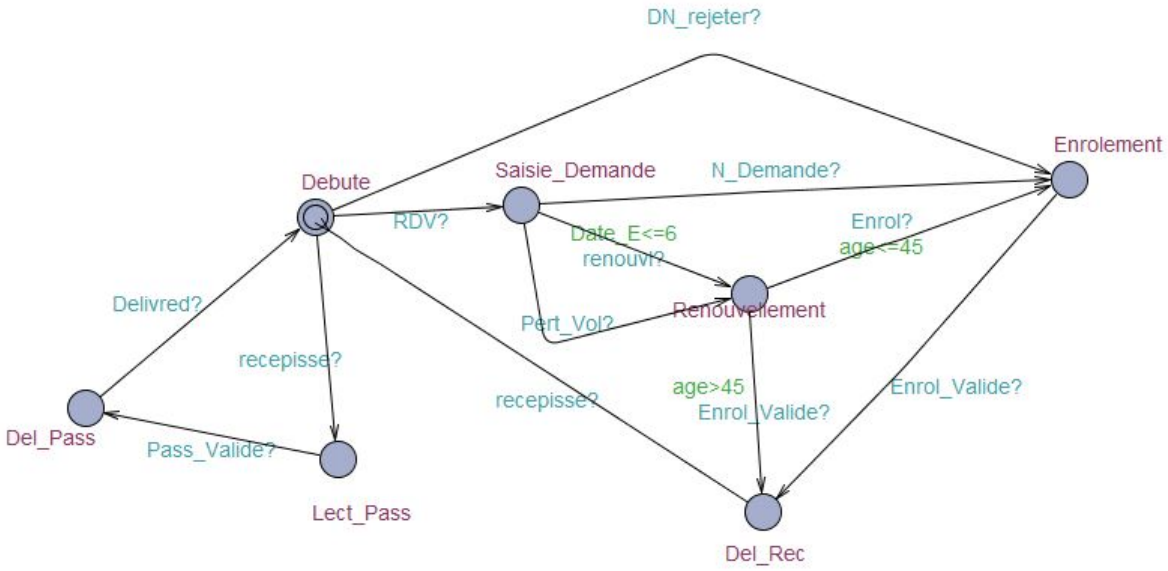


FIGURE 4.7 – TA :passeport

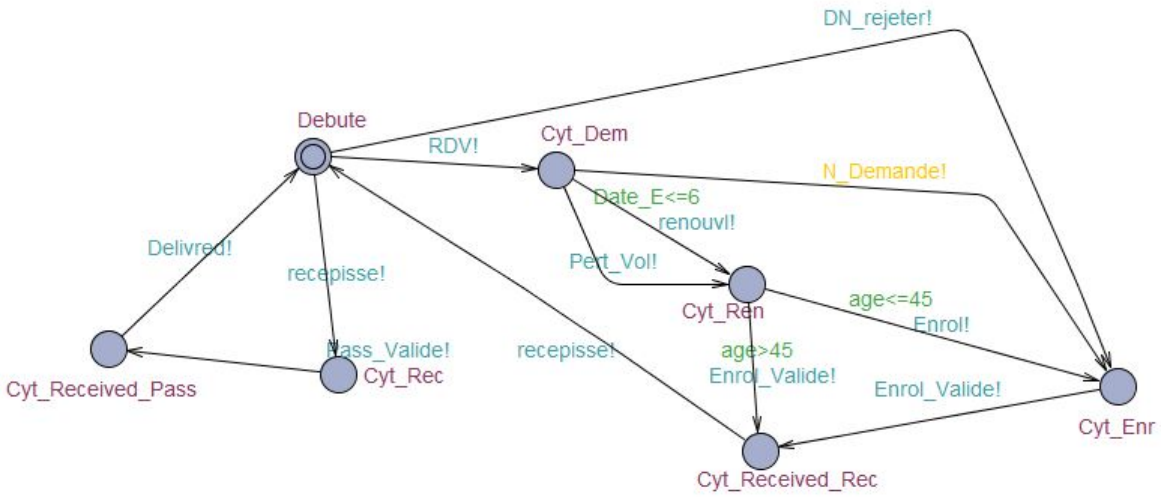


FIGURE 4.8 – TA :citoyen

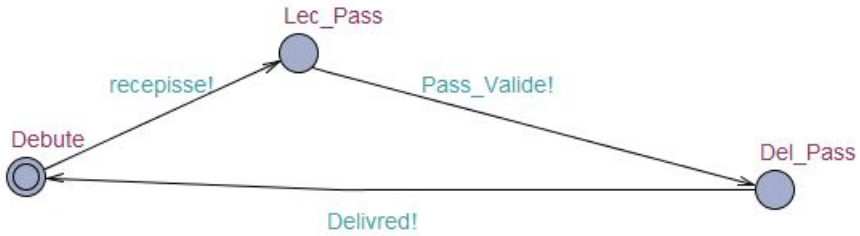


FIGURE 4.9 – TA :délivrance

4.3.6 Vérification formelle

Propriétés à vérifier

1. L'absence d'interblocage **A [] notdeadlock** . Cela signifie que pour tous les chemins du système et pour tout instant la propriété "not deadlock" est vérifiée. dans notre système cette propriété est satisfaite.
2. "Chaque citoyen fait une demande pour obtenir leur passeport il doit avoir un récépissé" la modélisation de cette propriété peut être exprimer comme suite :
   E <> Citoyen.Cyt − Dem imply Citoyen.Cyt − Received − Rec
   dans notre système cette propriété est satisfaite.
3. "Chaque citoyen a un récépissé il doit avoir son passeport" la modélisation de cette propriété peut être exprimer comme suite :
   E[]Citoyen.Cyt − Received − Rec imply Passeport.Del − Pas
   dans notre système cette propriété est satisfaite.
4. "La délivrance d'un passeport est toujours après la délivrance d'un récépissé " la modélisation de cette propriété peut être exprimer comme suite :
   A <> Passeport.DelRec imply Passeport.DelPass
   dans notre système cette propriété est satisfaite.

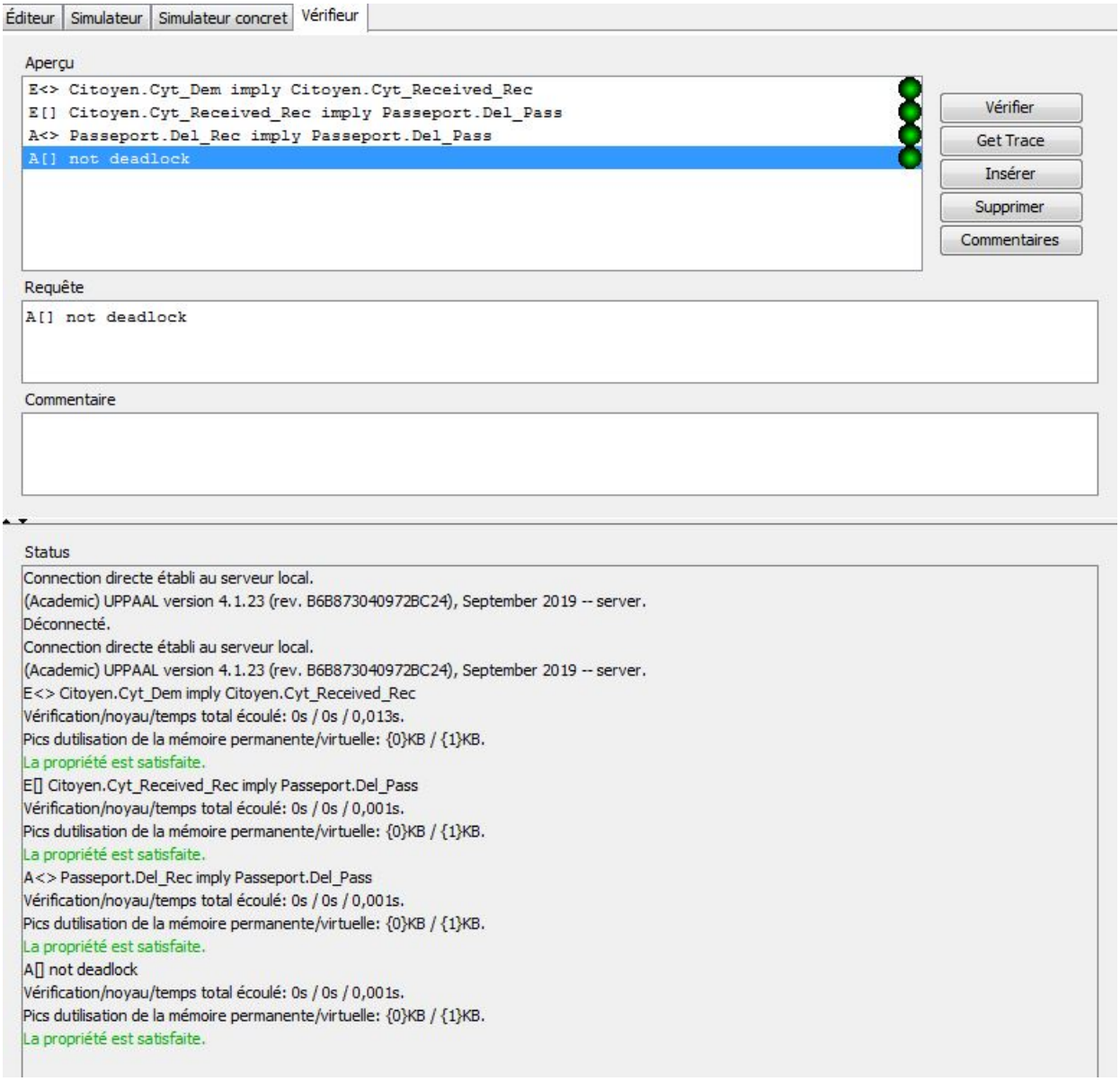


FIGURE 4.10 – Vérification avec UPPAAL

4.4 Conclusion

Les méthodes de vérification par model-checking se fondent sur une modélisation des systèmes à vérifier en automates, dans ce chapitre nous avons présenté notre modélisation du système demande un passeport biométrique en BPMN, ensuite l'interpréter en DD-LOTOS puis la génération de modèles C-Data, d'autres partent la transformation de modèle BPMN en automate temporisé pour arriver à vérifier quelques propriétés à l'aide du vérificateur d'UPPAAL.

Conclusion et perspectives

Nous avons présenté dans ce mémoire une approche pour la transformation et la vérification des modèles BPMN. Nous nous sommes focalisés sur les modèles de spécification LOTOS et DD-LOTOS. D'autre part, le modèle sémantique C-DATA et les automates temporisés.

Dans notre contribution, nous avons proposé premièrement une interprétation des modèles BPMN en spécification DD-LOTOS, ensuite la génération de modèles C-DATA à partir de cette spécification, deuxièmement une approche de transformation d'un diagramme BPMN en automate temporisé pour arriver à appliquer les méthodes de model-checking UPPAAL.

Une perspective intéressante de ce travail consisterait à réaliser une implémentation de l'approche d'interprétation des modèles BPMN en C-DATA et l'approche de transformation un diagramme BPMN en automate temporisé ont utilisons les outils de transformation de graphes comme ATOM3, FUJABA et VIATRA. De plus, il serait fort intéressant d'implémenter un model-checking pour la vérification des modèles C-DATA.

Bibliographie

[AD1990] R. Alur and D.L. Dill. Automata for Modeling Real-Time Systems.

[AD1994] R. Alur and D.L. Dill. A Theory of Timed Automata.

[ACD93] Rajeev Alur, Costas Courcoubetis, and David Dill. Model-Checking in Dense Real- Time.

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

[AS2013] Amen Souissi. Modélisation centrée sur les processus métier pour la génération complète de portails collaboratifs

[BA1998] Bérard, B., Diekert, V., Gastin, P. et Petit, A. Characterization of the expressive power of silent transitions in timed automata.

[BA2010] Boumaza Amel. Algorithmes de model checker opérant sur les DATA* (Durational Action Timed Automata).

[BB2005] Brinksma, E. et Briones, L. B. Test generation framework for quiescent real-time systems.

[BH2006] Belmokadem, H. Vérication Des Propriétés Temporisées Des Automates Programmables Industriels.

[BM2018] BENDIAF Mesaoud Spécification et vérification des systèmes embarqués temps réel en utilisant la logique de réécriture.

[BN2005] BELALA Nabil Formalisation des systèmes temps-réel avec durées d'actions.

[BK1984] J.-A. Bergstra and J.-W. Klop. Process algebra for synchronous communication. Information and Computation, 60 :109-137, 1984.

[BPMN] BPMN http ://www.bpmn.org/

[CT2003] C. Atkinson and T. Kuhne. Model-driven development : a metamodeling foundation Software.

[EB1988] E. Brinksma. A theory for the derivation of test. Protocol Specification, Testing and Verification- cation, S.Aggarwal and Sabnani,eds, VIII :6374, 1988.

[EM1985] H. Ehrig and B. Mahr. Fondamentals of Algebraic Specication,Equations and Initial Semantics. Springer-Verlag, 1985.

[HH2013] Hiba HACHICHI. Test formel des systèmes temps réel : Approche de transformation de graphes.

[HM2018] HAMDANE Mohamed ElKamel. Une approche basée sur l'ingénierie dirigée par les modèles pour la vérification des descriptions AADL

[Hoa1985] C. A. R. Hoare. Communicating Sequential Processes. Prentice-Hall, 1985.

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

[KITOUNI] Ilhem KITOUNI. Déterminisation des automates temporisés avec durées d'actions pour le test formel.

[LM2014] Luis Mendouz. Business Process Veri cation : The Application of Model Checking and Timed Automata.

[MT2012] Maarouk Toufik Messaoud. Modèles formels pour la conception des systèmes temps réel.

[Mil1980] R. Milner. Communication and Concurrency, volume 92 of LNCS. Springer-Verlag, 1980.

[Mil1989] R. Milner. Communication and Concurrency. Prentice Hall, 1989.

[OMG] OMG. Business process definition metamodel (bpdm), version 1.0,URL http ://www.omg.org/spec/BPDM/1.0/.

[RG2017] Rahabi,Ghaoui. Une sémantique formelle pour le processus métiers BMPN en utilisant les Algèbres de processus.

[RM5432] Raida ElMansouri. Modélisation et Vérification des processus métiers dans les entreprises virtuelles : Une approche basée sur la transformation de graphes.

[RS1985] R. de Simone. Higher-level synchronising devices in MEIJE-SCCS. Theoretical Computer Science„ 37 :245267, 1985.

[WT2018] Wafa Triaa. Gestion agile de processus métier : proposition d'une approche tirée par les compétences.