Contribution à la gestion de l'évolution des processus métiers

Mohammed Oussama Kherbouche

Université du Littoral Côte d'Opale

THÈSE

Présentée en vue d'obtenir le grade de
DOCTEUR de l'Université du Littoral Côte d'Opale
Spécialité : Informatique

Présentée et soutenue par :

Mohammed Oussama KHERBOUCHE

Le 02 décembre 2013

Contributions à la gestion de l'évolution des processus métier

COMPOSITION DU JURY :

M. Henri Basson	Professeur à l'Université du Littoral Côte d'Opale	(Directeur)
Mme. Selmin Nurcan	Maître de Conférences à l'Université Paris 1, HDR	(Rapportrice)
M. François Charoy	Professeur à l'Université de Lorraine	(Rapporteur)
M. Mourad Oussalah	Professeur à l'Université de Nantes	(Examinateur)
M. Mourad Bouneffa	Maître de Conférences à l'Université du Littoral Côte d'Opale	(Co-Encadrent)
M. Christophe Renaud	Professeur à l'Université du Littoral Côte d'Opale	Examinateur (Président)

Thèse réalisée au sein du Laboratoire d'Informatique Signal et Image de la Côte d'Opale - EA 4491-
Équipe Model (multi-modélisation et évolution des logiciels)
Maison de la Recherche Blaise Pascal 50, rue Ferdinand Buisson BP 719 62228 Calais Cedex France

« …Le savoir scientifique n'est pas absolu, mais socialement, culturellement, technologiquement et historiquement marqué, donc provisoire… » ***Steven Rose***

Remerciements

Les travaux présentés dans ce manuscrit ont été effectués au sein du Laboratoire d'Informatique Signal et Image de la Côte d'Opale (LISIC) de l'Université du Littoral Côte d'Opale (ULCO). Je souhaite adresser mes remerciements à toutes les personnes qui ont contribué de près ou de loin à l'élaboration de mes travaux de recherche.

Je tiens tout d'abord à exprimer ma gratitude et mes remerciements particuliers à mon directeur de thèse le Professeur Henri Basson, pour m'avoir encouragé, soutenu et conseillé durant toutes ces années, ainsi que pour la confiance dont il a fait preuve à mon égard.

Je remercie aussi chaleureusement mes rapporteurs pour toute l'attention et l'intérêt particulier qu'ils ont apportés à ma thèse : madame Selmin Nurcan, et monsieur François Charoy. Associer à ces remerciements les membres du jury : monsieur Mourad Oussalah, monsieur Christophe Renaud et monsieur Mourad Bouneffa, qui m'ont honoré de leur présence aujourd'hui toute en acceptant d'être examinateurs de mon travail.

Mes remerciements seraient incomplets si je n'y associe pas toutes les personnes qui ont contribué à ce travail, en particulier, mes collègues, messieurs Adeel Ahmad et Mourad Bouneffa, le directeur du laboratoire monsieur Christophe Renaud et tout le personnel du laboratoire.

Au final, j'aimerais exprimer toute mon attention très affectueuse et ma gratitude à mes parents, pour leur indéfectible soutien, leur patience et leur conseil avisé dans tous mes choix professionnels et personnels, durant toute cette période. À mes frères et sœurs, à toute ma famille ainsi que mes amis.

Calais, le 12 Octobre 2013

Mohammed Oussama Kherbouche

Résumé

La gestion de l'évolution des processus métier exige une compréhension approfondie des causes des changements, de leurs niveaux d'application ainsi que de leurs impacts sur le reste du système. Dans cette thèse, nous proposons une approche de gestion et de contrôle de l'évolution des processus métier permettant d'analyser ces changements et de comprendre leurs impacts. Cela assistera les concepteurs et les chargés de l'évolution des processus métier à établir une évaluation *a priori* de l'impact pour réduire les risques et les coûts liés à ces changements et d'améliorer le service et la qualité des processus métier.

Ce travail consiste à proposer un ensemble de contributions permettant une vérification de la cohérence et de la conformité des modèles de processus métier après chaque changement, mais aussi d'établir une évaluation *a priori* de l'impact structurel et qualitatif des modifications. Les différentes approches proposées sont en cours d'expérimentation et de validation à travers le développement d'une plate-forme basée sur l'environnement Eclipse.

Mots-clés

Processus métier – modèles BPMN – Model-checking – Analyse d'impacts – relations de dépendance, BPM Quality.

Abstract

The evolution management of the business processes requires an exhaustive understanding of the change. An evolution engineer needs to understand reasons of a change, its application levels, and subsequently its impact on the whole system. In this thesis, we propose an approach for an *a priori* change impact analysis, to better control the business process evolution. This may help the business experts and the process designers to evaluate change impact in order to reduce the associated risks and estimate the related costs. It may also help to improve the service and quality of the business processes.

This work contributes an eventual improvement, in regard, to verify the coherence and the compliance of the business process models, after each change. It leads to evaluate an *a priori* change impact analysis in structural and qualitative aspects. The multiple-perspectives of the proposed approach have been reviewed experimentally. The validation of the approach is evaluated by extending the Eclipse Development Environment, with the help of a set of plug-ins, as a prototype plate-form.

Keywords

Business process, BPMN models, Model-checking, Impact Analysis, dependency relationships, BPM Quality.

Table des matières

Table des figures

Liste des tableaux

Listing

Introduction

Le concept de processus métier occupe aujourd'hui une place majeure dans le domaine des systèmes d'information. Toutefois, ces processus métier doivent être en constante évolution afin de permettre aux entreprises de répondre aux exigences du marché et aux inflexions des besoins de leurs différents interlocuteurs, et par conséquent, de construire ou préserver leur avantage concurrentiel.

À cet égard, une compréhension des connaissances descriptives des processus métier est une condition *sine qu'a none* de la réussite d'une telle évolution. La mise en œuvre de cette évolution revient donc à réaliser des changements plus ou moins importants de certains constituants du processus métier. Ces changements qui sont dus généralement à une correction d'une ou plusieurs erreurs, à une exception dans le processus métier, à la mise en conformité avec des nouvelles normes ou lois juridiques, à l'évolution du métier et des services de l'entreprise (innovation), à l'amélioration des performances et l'optimisation des processus existants (re-engineering) peuvent engendrer des erreurs (blocages, des exécutions infinies, des multiples terminaisons etc.), une non-conformité avec la réglementation, une dégradation du service (temps de réponse, sécurité, taille des messages, etc.), ou une mauvaise qualité de l'application tout entière. Cela conduit à la nécessité de mettre en place un mécanisme de gestion et de contrôle permettant de réaliser à la fois une vérification de la cohérence et de la conformité des modèles de processus métier modifié et à une analyse *a priori* de l'impact du changement de ces derniers et une analyse *a posteriori* de l'impact effectif de ce changement.

Dans le cadre du contrôle et de la gestion de l'évolution des processus métier, cette thèse propose une approche permettant entre autres, d'assister les modeleurs et les experts métier à établir une vérification formelle des processus métier modélisé en BPMN après chaque changement, et une évaluation *a priori* de l'impact structurel et qualitatif de ces changements.

L'approche est validée en réalisant une plate-forme spécifique permettant la gestion de l'évolution des processus métier. Cette plate-forme permet à la fois une vérification de la cohérence et de la conformité des modèles de processus métier par la tech-

nique du model-checking, une analyse *a priori* de l'impact du changement de ces derniers et une gestion qualitative de ces modèles. Le reste de la thèse est organisé comme suit :

- En préambule, le chapitre 1 présente brièvement chacune des définitions et notions employées dans la discipline BPM *"Business Process Management"*, et décrit en détail la problématique du domaine de recherche de la thèse qui a trait aux principes de gouvernance et l'étude de l'impact du changement des processus métier.
- Le chapitre 2 souligne tout d'abord l'importance de la cohérence des modèles de processus métier dans l'efficacité et la qualité de ces derniers, en particulier après chaque changement, ce chapitre comporte un état de l'art qui relate les avantages et les inconvénients de chacune des approches proposées dans la littérature pour gérer les différents aspects de l'évolution des processus métier.
- Le chapitre 3 détaille notre contribution dans le domaine de la vérification formelle des modèles de processus métier modélisés en BPMN qui est basée sur la technique du model-checking. Il aborde en particulier les détails de la vérification de la cohérence et la conformité de ces modèles.
- Le chapitre 4 décrit la mise en œuvre de notre contribution permettant l'analyse *a priori* de l'impact du changement. Il présente en particulier les différentes relations de dépendance impliquées dans le processus de propagation de l'impact du changement et de son analyse. Le chapitre aborde également la formulation d'une base de connaissances qui permet une compréhension, à la fois du processus métier et de son évolution et un ensemble de règles de faisabilité et d'analyse d'impact du changement, et discute l'application de ces derniers.
- Le chapitre 5 est consacré à la proposition d'un modèle qualitatif permettant d'évaluer la qualité des modèles de processus métier inspiré par la norme ISO / CEI 9126-1. Le chapitre aborde les questions liées à l'assurance qualité des processus métier suivis d'un tour d'horizon des approches proposées dans la littérature pour assurer la qualité des processus métier. La dernière partie du chapitre est consacrée à l'analyse qualitative de l'impact du changement des processus métier.

- Le chapitre 6 valide la mise en œuvre et les résultats de l'approche de la modélisation proposée. Il décrit le développement d'une plate-forme de prototype pour la gestion de l'évolution des processus métier.
- La conclusion générale de la contribution de cette thèse et les perspectives des travaux de recherche sont données au chapitre 7 de ce manuscrit.

CHAPITRE 1

Cadre théorique et problématique

1.1 Contexte et terminologie

Afin de demeurer compétitives et être en mesure de répondre aux différents besoins évolutifs du marché, et d'interagir de manière optimale avec l'ensemble des acteurs qui les constituent, les organisations modernes se doivent d'être agiles. Cela veut dire que les processus mis en œuvre dans les différents systèmes constituant une organisation doivent s'adapter rapidement à l'échelle des valeurs de cette dernière. Cela constitue concerne aussi bien les niveaux intra et inter-organisationnels. Autrement dit, cela concerne aussi bien la structuration interne de l'organisation que son interaction avec son environnement extérieur. L'adaptation à l'échelle de valeur est, en effet, une condition nécessaire à la disposition d'un système d'information (SI) efficace et efficient supportant à la fois les stratégies métiers et les processus qui y sont rattachés.

En effet, la notion de processus joue un rôle primordial dans la maîtrise de l'évolution des systèmes d'information et des systèmes informatiques associés. Ils sont souvent considérés comme un préalable indispensable à la conception de la dynamique d'un système d'information d'une organisation (Alter, 1999), (Butler, Esposito, & Hebron, 1999) et cela en fournissant des processus robustes, et adapté à leurs activités. Leur maîtrise constitue alors un enjeu important, pour tous les acteurs d'une organisation tels que les experts métiers, les équipes fonctionnelles et les équipes techniques en termes de réactivité des systèmes, et de compétitivité sur les marchés.

La définition et l'exécution de ces processus nécessitent un modèle et des outils permettant leur définition, déploiement, exécution, analyse et leur contrôle. Dans ce contexte, la gestion des processus métier appelée aussi BPM *"Business Process Management"* est une approche globale permettant de gérer de bout en bout les processus métier de l'entreprise en associant l'analyse, la supervision et l'amélioration de ces processus dans une optique de forte intégration avec le système d'information.

Dans ce chapitre, nous introduisons tout d'abord la terminologie utilisée dans la discipline dite BPM suivie de la problématique de notre thèse qui a trait aux principes de gouvernance et d'étude de l'impact du changement des processus métier, ainsi que les solutions apportées.

1.2 Entreprise, pilotage et système d'information

Une entreprise est un système socio-économique visant à créer de la valeur ajoutée[1] pour ses partenaires et ses interlocuteurs internes et externes et dans laquelle l'information est considérée comme une ressource vitale pour son fonctionnement. Cette valeur ajoutée est atteinte grâce à l'orchestration d'activités diverses réalisées à l'aide d'acteurs et des ressources de l'entreprise.

Dans ce contexte, il nous semble intéressant de rappeler l'évolution de la perception des systèmes d'information au sein des organisations. En théorie systémique, Le Moigne (Le Moigne, 1984) considère les fonctionnalités d'une entreprise ou d'une organisation

[1] La valeur ajoutée est un indicateur économique qui mesure la valeur ou la richesse créée par une entreprise

comme étant assurées par trois systèmes : le système opérant (SO), le système de pilotage (SP) et le système d'information (SI) comme le montre la figure 1.1.

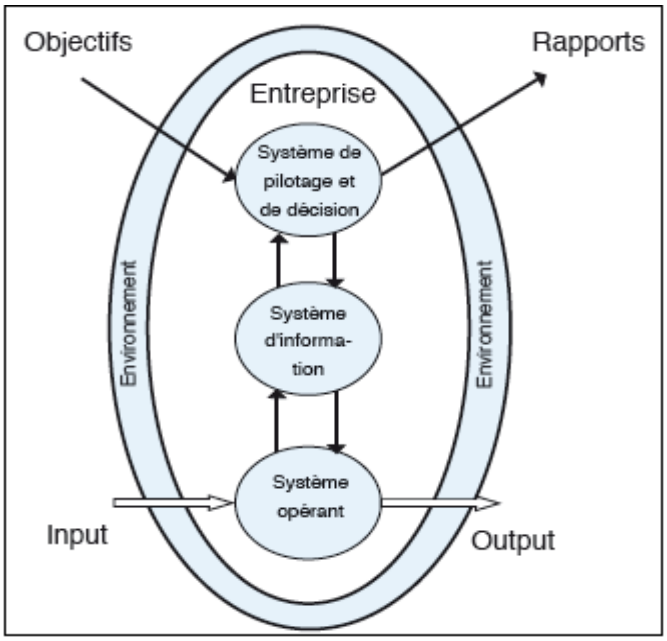


Figure 1.1 *La composition d'un système d'entreprise*

Le système opérant (SO) réagit aux flux des événements quotidiens, qui viennent de l'environnement d'une entreprise, selon des règles bien définies. Il représente des éléments matériels ou immatériels en interaction transformant par un processus des flux d'entrées tels que les flux financier, les flux de matière, les flux d'information en flux de sortie, et donne ainsi des informations sur l'état du système au SP via le SI.

Le système de pilotage (SP) permet d'engager le processus de décision tout en définissant au préalable les objectifs, les critères d'évaluation et les règles de gestion.

Enfin, le Système d'Information (SI) joue un rôle primordial permettant de relier les deux systèmes précédents en transmettant les directives décidées par le SP au SO. Le SI renvoie les informations issues du SO permettant ainsi au SP de prendre les bonnes décisions pour un fonctionnement optimal du système global.

1.3 Le processus dans la gouvernance du système d'information

Le Système d'Information est au cœur des enjeux métiers et stratégiques de toute organisation (Morley, Bia-Figueiredo, & Gillette, 2011). Son efficacité constitue un élément clé de la performance de cette dernière. Il est généralement considéré comme l'ensemble identifiable de processus, de ressources et d'acteurs, ayant pour rôle de traiter, gérer et diffuser de l'information au sein d'une organisation ainsi que d'interagir avec le milieu extérieur.

Cependant, la notion de système d'information n'a cessé d'évoluer au fil du temps. Certains acteurs invoquent un «*système de traitement de l'information*», d'autre de « *système d'information pour le management*», ou encore de «*système d'information organisationnel*». Cependant ces définitions qui servaient de référence pendant une certaine

période ont évoluée en fonction de la diffusion croissante de l'informatique dans les activités d'une organisation.

Ainsi, des auteurs comme (Alter, 1999) définissent un système d'information comme «*un système de travail dont les fonctions internes sont limitées à traiter l'information, en exécutant six types d'opérations : saisir, transmettre, stocker, retrouver, manipuler, afficher l'information. Un système d'information produit de l'information, assiste ou automatise le travail exécuté par d'autres systèmes de travail.*». Cette définition, est plutôt réductrice du fait qu'elle résume le système d'information comme étant quasiment un système informatique.

Les définitions vont progressivement s'élargir pour traduire le fait qu'un système d'information a dépassé le stade d'outil pour devenir l'élément structurant d'une organisation. Ainsi, (Reix, 2004) définit un système d'information comme *« un ensemble organisé de ressources matériel, logiciel, personnel, données, procédures permettant d'acquérir, traiter, stocker, communiquer des informations (sous forme de données, textes, images, sons, etc.) dans les organisations* ». Dans cette définition, on voit apparaître une notion essentielle qui est «la procédure». Celle-ci décrit comment, quand et où l'acteur est supposé utiliser des ressources : matériels, logiciels et données pour que l'organisation soit informée. Cette définition été amplifiée par Morley *et al.* (Morley, Hughes, Leblanc, & Hugues, 2007), qui ont clairement fait la distinction entre le système d'information et le système informatique comme le montre la figure 1.2. En considérant le système d'information sous la gouvernance du système de pilotage, supportant la gestion de l'entreprise est supporté par les ressources informatiques d'aide à la décision.

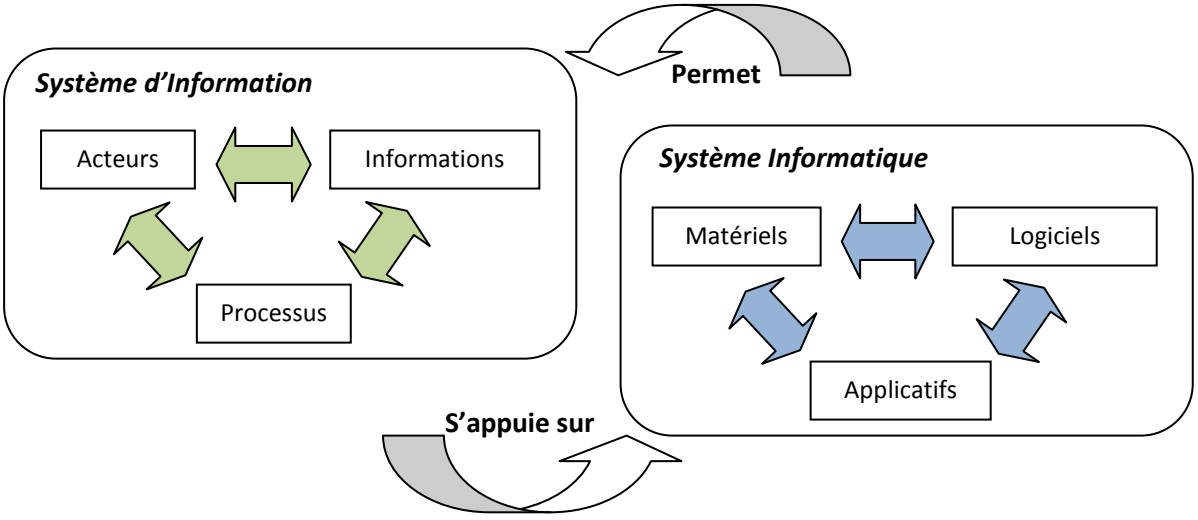


Figure 1.2 *Système d'information et système informatique*

La diffusion de l'approche processus a conduit récemment à intégrer cette notion dans la définition d'un système d'information. Ainsi (Alter, 1999) fut parmi les premiers auteurs à mettre l'accent sur les liens existant entre processus et SI. Il est parti de la notion de « processus métier » abordée en détail plus loin, comme étant un ensemble coordonné d'activités visant à produire un résultat pour des clients internes ou externes. S. Alter appelle « système de travail », l'ensemble des processus, des acteurs, et des ressources dont la finalité est de produire un résultat escompté. Dans certains cas, les activités des

processus sont uniquement du traitement de l'information : saisie, stockage, transmission, manipulation. Le système de travail est alors appelé système d'information.

De nos jours, la notion de processus joue un rôle majeur dans la définition et la gestion des systèmes d'information même si ses utilisations sont diverses. La plupart des méthodes d'analyse actuelles proposent des concepts pour modéliser un système d'information autour du processus et ses interactions, et non simplement autour d'applications informatiques.

La notion de processus est donc considérée comme étant un ensemble d'activités, exécutées dans un objectif bien déterminé par un acteur correspond à un rôle. Le déroulement du processus utilise des ressources et peut être conditionné par des évènements d'origine interne ou externe. L'agencement des activités correspond à la structure du processus.

La typologie de processus la plus communément reconnue distingue trois catégories selon (Debauche & Mégard, 2004):

- *Les processus de pilotage* ou de management qui ont pour but d'organiser les objectifs stratégiques de l'entreprise.
- *Les processus métier (ou processus opérationnel, processus de réalisation)* qui ont pour fonction d'accomplir une mission dans un domaine donné et utilisent plusieurs fonctions de l'entreprise.
- *Les processus de support* ou de soutien sont périphériques au métier de l'entreprise et ne participent qu'indirectement à l'accomplissement d'un objectif métier.

Selon (SPINOV, 2006), une quatrième catégorie de processus vient se greffer autour de ces catégories, appelée processus de mesure, ces processus permettent de fournir les métriques nécessaires à l'évaluation des processus et à leur amélioration continue. Cette typologie est représentée figure 1.3.

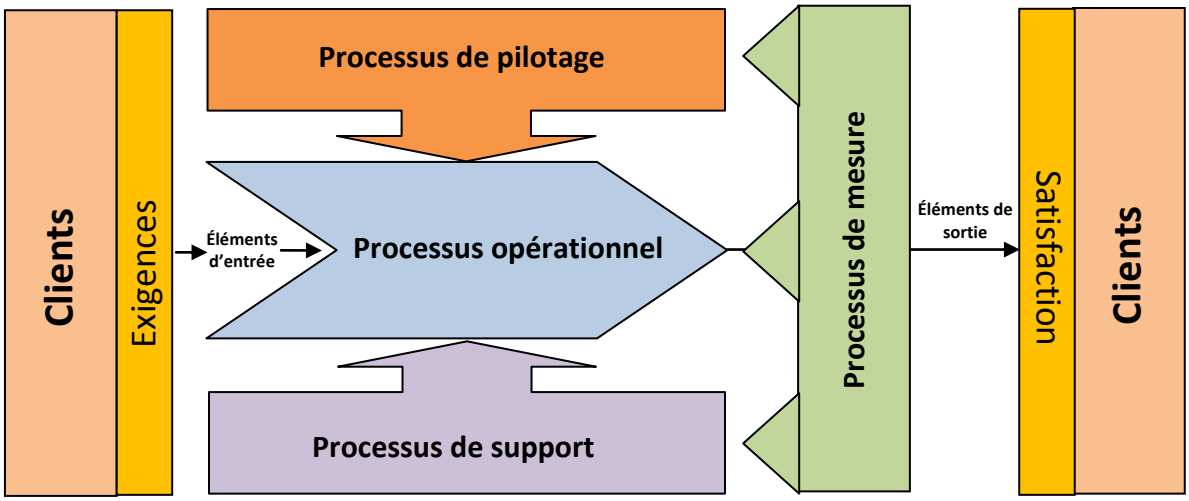


Figure 1.3 *Typologie des processus*

1.4 Processus, procédure et procédé

Processus, procédure ou procédé sont trois termes qui peuvent prêter à confusion dans le langage courant, ils sont dévirer du mot latin *« procedere »*, qui signifie « Aller en avant ». La procédure a des définitions diverses selon la discipline. Dans le domaine juridique par exemple, cela désigne « les formes et les règles selon lesquelles on doit se comporter en justice ». Dans le monde de l'entreprise, ce mot désigne « l'ensemble des étapes successives dans la conduite d'une opération complexe » par exemple « Quelles sont les démarches à suivre en cas d'incendie ?». Le procédé après avoir été évoqué de façon générale comme une « façon de faire » au XVII^{e} siècle, a évalué pour être défini comme « la manière méthodique employée pour parvenir à un résultat ». Le terme «procédé », est plutôt utilisé dans la fabrication de produits à partir de matières premières.

Un processus après avoir désigné en général « une suite d'événements naturels, se déroulant dans le même ordre », il se définit dans le domaine technique comme étant « une suite d'opérations aboutissant à un résultat » notamment dans l'expression *processus de fabrication*. Par cette définition, nous pouvons voir que le terme de processus et procédure, ont convergé, et même parfois été confondus dans le langage courant.

En effet, dans le domaine des SI, la procédure constitue l'aspect organisé du processus. La distinction entre les deux termes a été normalisée par la norme ISO 9000 qui définit la procédure comme « une manière spécifiée d'effectuer une activité ou un processus ». Cela signifie qu'un processus peut donner lieu à plusieurs procédures différentes bien définies sur le plan organisationnel. Autrement dit, les rôles sont bien attribués, les supports (documents, données, logiciels) bien identifiés, et les méthodes de travail bien arrêtées.

1.5 Processus métier

Un processus métier ou *"Business Process"* en anglais est défini par le Workflow Management Coalition (WfMC) comme : "*un ensemble de procédures ou d'activités liées les unes aux autres pour atteindre collectivement un objectif métier en définissant les rôles et les interactions fonctionnelles au sein d'une structure organisationnelle*" (Coalition, 1999). En effet, un processus métier met en exergue une collection d'activités et de tâches organisées dans le temps qui, une fois achevées, permettront d'atteindre un objectif organisationnel et un résultat précis et mesurable. Autrement dit, il est considéré comme un ensemble de relations logiques entre un groupe d'activités incluant des interactions entre différents partenaires et participants tels que des applications ou des services du SI, des acteurs humains ou d'autres processus métier sous la forme d'échange d'informations et de données pour fournir une valeur ajoutée tangible aux clients et aux différents interlocuteurs d'une organisation.

Le méta modèle proposé par (Morley, Hughes, Leblanc, & Hugues, 2007) permet de voir l'interaction entre ces différents concepts (*cf.* figure 1.4).

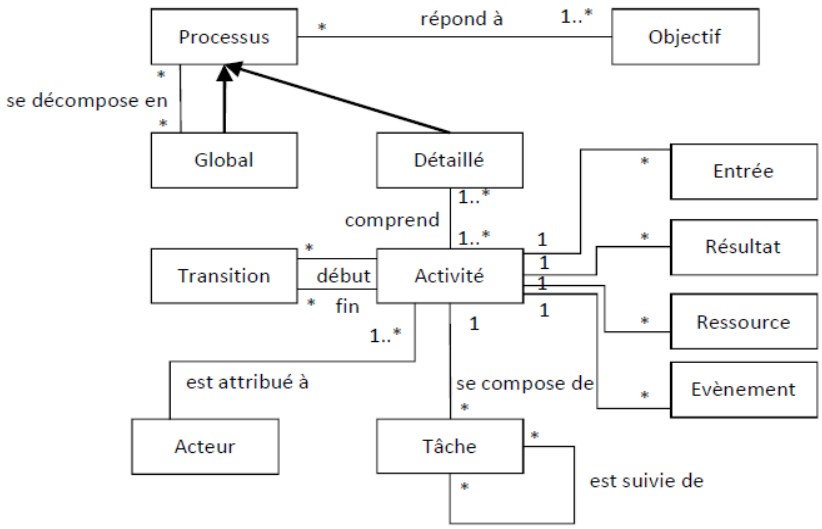


Figure 1.4 *Méta modèle du processus métier selon* **(Morley, Hughes, Leblanc, & Hugues, 2007)**

1.6 La gestion des processus métier (BPM)

Le "*Business Process Management*" (BPM), traduit littéralement en français par la gestion de processus métier, est défini par (Van der Aalst, Ter Hofstede, & Weske, 2003) comme : "*l'utilisation de méthodes, techniques et systèmes logiciels pour concevoir, exécuter, contrôler et analyser des processus opérationnels faisant intervenir des hommes, des applications, des documents et d'autres sources d'information*". Cela est donc considéré comme une approche managériale axée sur l'alignement de tous les aspects d'une organisation aux besoins des clients qui place les processus métier au centre d'une réflexion globale d'intégration *"Process-centric"*. Le but est, en effet, de favoriser l'efficacité opérationnelle tout en abordant la question de l'amélioration continue des processus priorisant le point de vue « métier » au point de vue « technique », cela rend donc les processus plus efficaces et capables de s'adapter à d'éventuels changements dans l'entreprise. En d'autres termes c'est une approche permettant à une organisation de s'assurer que ses processus sont mis en œuvre efficacement tout en répondant aux besoins de ses différents interlocuteurs avec un niveau, de performances optimales et une maîtrise des coûts.

1.7 Cycle de vie d'un processus métier

Le cycle de vie d'un processus métier selon une démarche BPM permet un accompagnement d'un processus métier depuis sa conception à sa gestion et son pilotage tout en évoluant en permanence selon les objectifs métiers qui le définissent. Cela se manifeste par une mise en œuvre du point de vue "théorique" jusqu'au point de vue "pratique" d'un processus métier à travers plusieurs définitions et perspectives.

Dans la littérature, il n'y a pas de vue uniforme sur le nombre de phases du cycle de vie BPM. Il varie en fonction de la granularité choisie pour l'identification des phases (Gillot, 2008), (Crusson, 2003), (Debauche & Mégard, 2004).

Il est composé principalement de quatre phases comme le montre la figure 1.5 :

1. La phase de modélisation "*Process Modeling*"
2. La phase d'implémentation "*Process Implementation*"
3. La phase d'exécution "*Process Execution*"
4. La phase de pilotage et d'optimisation "*Process Analysis*"

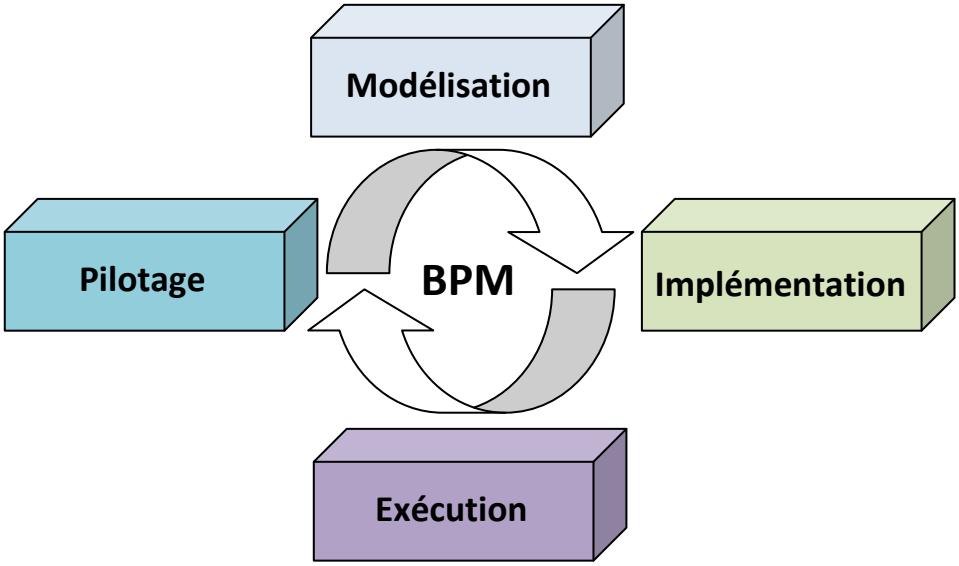


Figure 1.5 *Cycle de vie du BPM*

1.7.1 Phase de modélisation

La modélisation est la première phase dans le cycle de vie du BPM. Dans cette phase les experts métier définissent, d'une manière abstraite ou détaillée, les processus métier ou redéfinissent un processus existant dans le but de l'améliorer à l'aide d'un outil de modélisation qui permet de spécifier l'ordre des tâches dans le processus métier. L'outil de modélisation supporte une approche utilisant une notation de modélisation graphique basée généralement sur l'adoption du standard *"Business Process Modeling Notation"* (BPMN) (OMG, 2011). Les modèles de processus créés dans cette phase sont généralement d'un niveau d'abstraction élevé pour être directement exécutés par un moteur de processus en raison du manque d'informations techniques telles que les liaisons entre les différents services, les formats de données pour chaque tâche ...etc. Par conséquent, un modèle de processus métier ou *"Business Process Diagram"* doit être transformé en un modèle de processus exécutable, qui est l'objet de la phase suivante.

1.7.2 Phase d'implémentation

Dans la phase d'implémentation, le processus créé dans la phase de modélisation est transformé et enrichi par les ingénieurs IT dans le but d'être exécuté par un moteur de processus *"Process Engine"* (Leymann & Roller, 2000). A cet effet, le langage standard pour décrire les processus exécutables dans le contexte d'une architecture orientée ser-

vices (SOA) et les services web (Weerawarana, Curbera, Leymann, Storey, & Ferguson, 2005) est le *"Business Process Execution Language"* ou appelé aussi BPEL (OASIS Standard, 2007). En effet, BPEL exprime donc une séquence d'événements du Business Process contrairement aux services web qui fournissent les fonctionnalités du service. Le modèle de processus exécutable qui en résulte peut-être déployé dans un moteur de processus pour son exécution, et cela afin de réaliser l'interfaçage avec les différents systèmes nécessaires au fonctionnement du processus et pour mettre en œuvre les règles métier.

1.7.3 Phase d'exécution

La phase d'exécution est la phase opérationnelle où la solution de BPM est mise en œuvre. En effet, durant cette phase, le processus exécutable qui spécifie le déroulement de l'ensemble des activités d'un processus est interprété par un moteur d'exécution appelé BPE *"Business Process Engine"*. Le BPE est le responsable des interactions entre les acteurs du processus (les documents, les informations et les tâches). Il exécute des instances de processus tout en déléguant les tâches automatiques aux services web et les tâches manuelles aux acteurs. Si une exception se produit durant l'exécution du processus, le BPE a le rôle de lancer une action de compensation pour amener le processus à une exécution valide.

1.7.4 Phase d'analyse et d'optimisation

La phase de pilotage sert à superviser l'exécution opérationnelle des processus métier et mesurer les performances en se basant sur des fichiers logs. En effet, dans une organisation, le pilotage efficace d'une activité métier représente un point important pour la performance technique et économique de cette dernière.

Le BPM dans son objectif principal de management des processus métier doit fournir des outils de pilotage permettant ainsi une prise de décision concernant l'efficacité et l'amélioration des processus. Ces outils doivent permettre de mesurer et de présenter la performance de l'activité métier qu'elle gère. De manière générale, les solutions de BPM nomment cette fonctionnalité BAM pour *"Business Activity Monitoring"* ou Supervision de l'activité métier, en français.

Le BAM est une technologie de reporting adaptée à l'ensemble des acteurs métier de l'entreprise : les responsables, les analystes, les dirigeants ou les services informatiques, cette technologie constitue un élément-clé des solutions de BPM qui permet de surveiller les processus et s'assurer que les performances ne se dégradent pas au fil du temps, d'améliorer l'efficacité des processus, de donner la capacité d'acquérir la maîtrise et la vision d'ensemble du déroulement de l'activité métier ou encore de contrôler le bon déroulement de l'activité à travers des tableaux de bord et en utilisant des KPI *"Key Performance Indicators ou* indicateurs clés de performance.

Les KPI sont des données collectées lors de l'exécution des processus et cela dans un but de les améliorer et les optimiser. Les analystes métier ont besoin de ces indicateurs sur les différentes instances des processus. Les KPIs permettent de comparer et d'analyser le déroulement des activités basées sur les processus par rapport aux résultats attendus. L'analyse porte sur l'identification des différentes zones du processus qui sont peu ou pas performantes et qui sont susceptibles d'être améliorées.

1.8 Technologies de processus métier

Construire des processus métier requière un ensemble de technologies telles que des API *"Application Programming Interface"*, des moteurs de workflow. Dans cette section, nous décrivons quelques technologies qui participent à l'implémentation et la gestion des processus métier.

1.8.1 Processus et workflow

1.8.1.1 La notion de workflow

Un processus ne doit pas être confondu avec un Workflow. En effet, contrairement au processus métier qui représente un ensemble d'activités et de procédures qui permettent collectivement la réalisation d'un objectif métier, le Workflow ou « flux de travail » en français représente l'automatisation partielle ou entière d'un processus métier (zur Muehlen, 2004). Il sert à fournir à chaque acteur d'un processus, les informations nécessaires à l'exécution des activités qui le composent à savoir les documents, les informations et les tâches et cela d'après un ensemble de règles procédurales (Workflow Management Coalition, 1999). Autrement dit, c'est une représentation spécifique pour laquelle les mécanismes de coordination entre activités, applications ou participants peuvent être gérés par un système de gestion de workflow *"WfMS" (**W**ork**f**low **M**anagement **S**ystem)*.

La notion de workflow est apparue au début des années 90 dans le cadre des recherches sur les outils logiciels facilitant le travail collaboratif.

1.8.1.2 Les systèmes de gestion de workflow

Un système de gestion de workflow est un système permettant de définir des processus de workflow, de créer et de gérer l'exécution des instances de ces derniers. Cette exécution est pilotée par un moteur de workflow pouvant interpréter la définition du processus, interagir avec les différents participants et déclencher l'exécution d'un ou plusieurs programmes ou applications.

Un processus de workflow modélisé est composé (1) d'un ensemble d'étapes logiques, appelées activités, automatiques ou interactives, ainsi que d'un ensemble de relations (séquence, parallélisme, synchronisation, etc.) permettant de les relier, (2) d'un ensemble de critères pour déclencher ou interrompre un processus, (3) d'un ensemble d'informations, telle que les acteurs qui peuvent être une ressource soit humaine ou automatique et qui doivent exécuter une activité, les données, ou les applications informatiques associées.

1.8.2 Services web

Un service web est une application autonome ou un composant, c.à.d. une fonctionnalité sémantiquement bien définie, identifiée par un identificateur uniforme de ressource URI. Il représente une manière standardisée d'intégration des applications basées sur le Web en utilisant les standards XML[2], SOAP[3], WSDL[4], UDDI[5] et les protocoles de transport

[2] http://www.w3schools.com/xml/
[3] http://www.w3.org/TR/soap12-part1/

de l'Internet (*cf.* figure 1.6). XML est utilisé pour représenter les données, SOAP pour transporter les données, WSDL pour décrire les services disponibles, et UDDI pour répertorier les différents fournisseurs de services et les services disponibles.

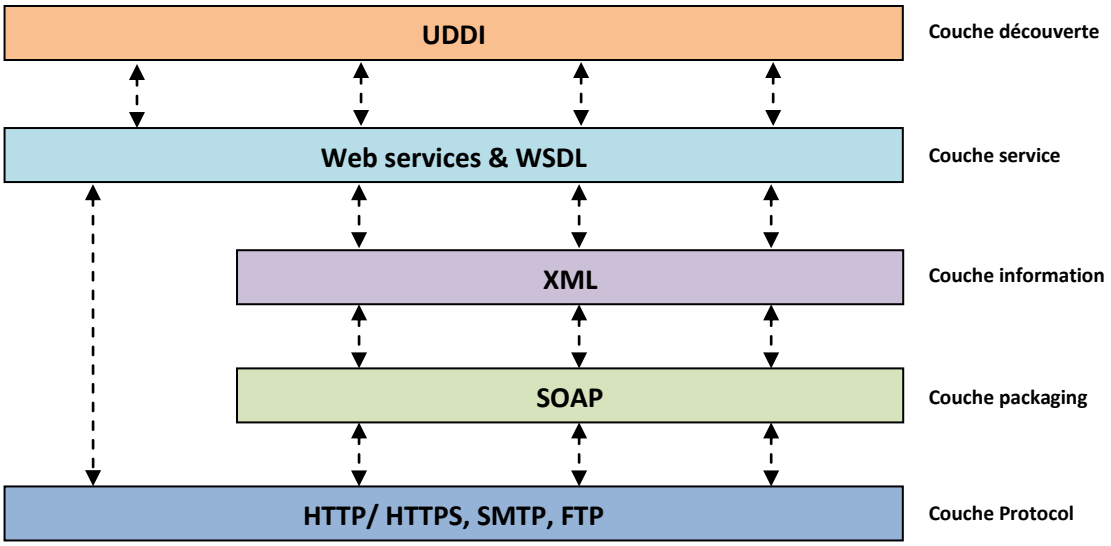


Figure 1.6 *Pile de protocoles de services web*

Les principales caractéristiques d'un service web sont :

- L'interopérabilité grâce à son indépendance vis-à-vis des systèmes d'exploitation et des langages de programmation.
- Publiable et accessible en utilisant le langage XML, les protocoles ouverts et les standards du web.
- Vise à exposer une ou plusieurs fonctionnalités.

1.8.3 Le moteur du BPM

Le moteur du BPM remplace les moteurs de workflow. Il est chargé de chorégraphier les services du SI. Aujourd'hui, les moteurs d'exécution des BPM respectent la norme BPEL *"Business Process Execution Language"*. L'utilisation générale de cette norme permet une interopérabilité entre les différents outils du marché.

1.9 La 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 approche dite impérative et une approche dite déclarative. La figure 1.7 représente un exemple de ces deux approches.

L'approche impérative se concentre sur la définition précise de la façon dont l'ensemble d'activités doit être achevé. 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 qui devrait être fait »

[4] http://www.w3.org/TR/wsdl/
[5] http://www.uddi.org/

au lieu de « quoi faire». Ce qui permet aux scénarios d'exécution de rester implicites dans la phase de modélisation du processus et cela en évitant d'énumérer explicitement tous les scénarii d'exécution possibles dans cette phase. Cette approche passe par l'utilisation d'un ensemble de contraintes et de règles pour restreindre les possibilités d'exécution des activités d'un processus. Autrement dit, tous les chemins d'exécution, qui ne violent pas les contraintes mise en place sont autorisés, ce qui offre beaucoup plus de chemins d'exécution possibles et une flexibilité des modèles obtenus. Ces contraintes remplacent, donc, les connecteurs explicites qui existent entre différentes activités dans l'approche impérative.

Dans les langages déclaratifs tels que ConDec (Pesic & Van der Aalst, 2006), PLM-flow (Zeng, Flaxer, Chang, & Jeng, 2002), PENELOPE (Goedertier & Vanthienen, 2006), un processus est vu comme un ensemble d'états et un ensemble de contraintes qui contrôlent les transitions d'un état vers un autre. Plusieurs paradigmes sont utilisés pour représenter les états et les contraintes tels que : la logique temporelle linéaire, la modélisation basée sur les règles, etc.

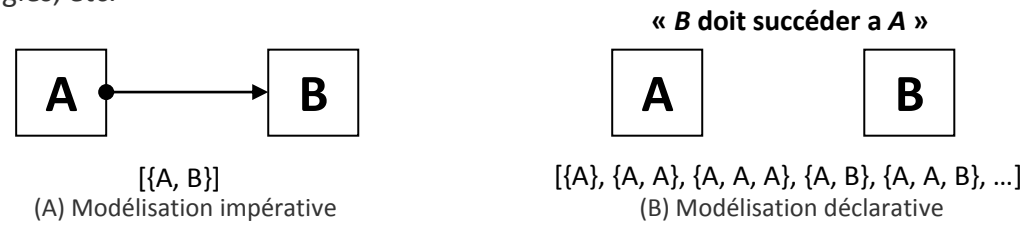


Figure 1.7 *Modélisation impérative et modélisation déclarative*

1.9.1 Les langages de modélisation des processus métier

Les langages de modélisation permettent de définir la manière dont les différentes activités du processus sont exécutées. Autrement dit, le scénario d'exécution qui représente une suite ordonnée d'activités. Plusieurs langages impératifs tels que BPMN (OMG, 2011), UML activity diagram (Group & OMG, 2007), EPC (ARIS, 2007) ainsi que des langages déclaratifs tels que ConDec (Pesic & Van der Aalst, 2006), PLM-flow (Zeng, Flaxer, Chang, & Jeng, 2002),PENELOPE (Goedertier & Vanthienen, 2006) ont été proposés pour la modélisation des processus métier.

1.9.1.1 Le standard BPMN

Le langage BPMN *"Business Process Modeling Notation"* est un standard pour la modélisation des processus métier d'une entreprise permettant de définir une notation graphique commune à tous les outils de modélisation. Il est proposé par le consortium l'OMG/BMPI *"Object Management Group* et le *Business Process Management Initiative"* depuis leur fusion en 2005.

Le principal objectif de BPMN est de fournir un cadre commun permettant de décrire un processus d'une manière compréhensible pour tous les acteurs d'une entreprise, depuis les analystes métier qui crée les ébauches initiales des procédures, jusqu'aux développeurs responsables de mettre en place la technologie qui va exécuter ces procédures, et ce, indépendamment de l'outil utilisé étant bien sûr censé supporter la norme. BPMN découple les informations métier des informations techniques et fournit une correspondance vers des langages d'exécution. Autrement dit, il permet de créer une passerelle

standardisée pour combler le vide existant entre la modélisation des processus métier et les langages d'exécution des processus métier.

Un diagramme 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 comme le montre la figure 1.8:

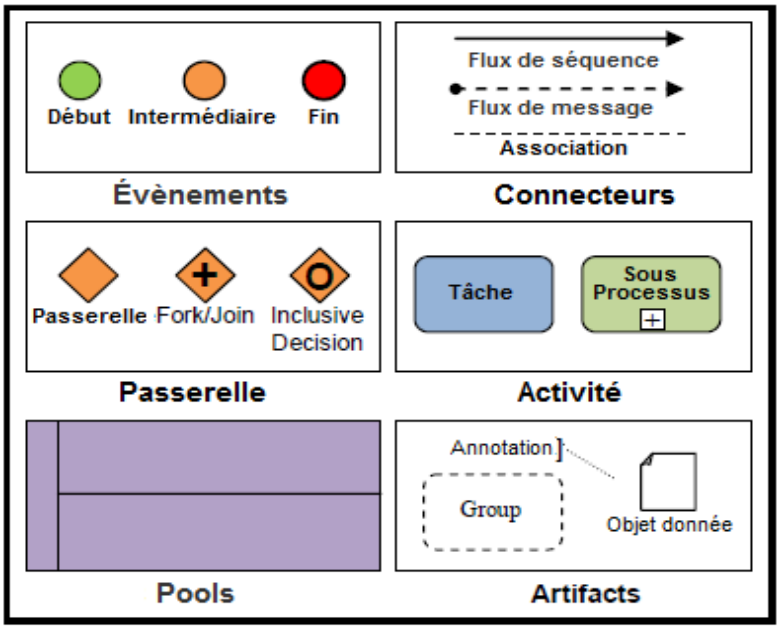


Figure 1.8 *Éléments de base d'un BPMN*

- Les objets de flux *"Flow Objects"*: ce sont les principaux éléments graphiques qui permettent de définir le comportement d'un processus métier. Il existe trois types d'objets de flux: *les activités "Activities"* qui correspondent à une action qui peut être réalisée par un humain ou une machine. Elles possèdent un début et une fin et ne peuvent commencer que si les activités qui les précédent sont terminées. Une activité peut être atomique ou composée. Les évènements *"Events"* qui correspondent à une action qui survient durant le processus. Ils ont en général une cause et une conséquence. Il est ainsi possible de modifier le déroulement d'un processus lorsqu'un événement particulier intervient au cours de l'exécution du processus. Il existe 3 catégories d'événements : Départ, Intermédiaires, Arrêt et enfin les passerelles ou les branchements *"Gateways"* qui correspondent à un branchement dans le processus qui permet de représenter une action dans l'avancement de ce dernier.
- Les objets de connexion *"Connecting Objects"*: ce sont les éléments graphiques qui permettent de relier les objets de flux les uns aux autres pour représenter le cheminement du processus. Il existe trois sortes de connecteur d'objets : les flux de séquence *"Sequence flow"* qui indiquent l'ordre d'exécution des actions. Les flux de message *"Message flow"* qui correspondent à un lien entre deux processus séparés et enfin, les associations *"Association"* qui permettent de lier des données ou des documents aux objets du processus.

- Les partitions *"Swimlanes"*: ce sont les éléments graphiques permettant de structurer et regrouper les différents éléments qui composent le diagramme BPMN. Ces éléments sont les partitions *"Pools"* et les sous-partitions *"Lanes"*.
- Les artefacts *"Artifacts"* : ils sont utilisés pour fournir des informations supplémentaires sur le processus. Il existe trois types d'artefacts : Les objets de données *"Data object"* qui montrent comment des données sont liées à une tâche (via une association). Les groupes *"Groups"* qui permettent de regrouper des taches d'une même catégorie afin de mieux les repérer visuellement sur le diagramme et enfin, les annotations *"Annotations"* qui permettent d'ajouter des commentaires pour faciliter la lecture du diagramme BPMN.

La figure 1.9 représente un exemple modélisant le fonctionnement d'un processus de vente aux enchères et utilisant la notation BPMN.

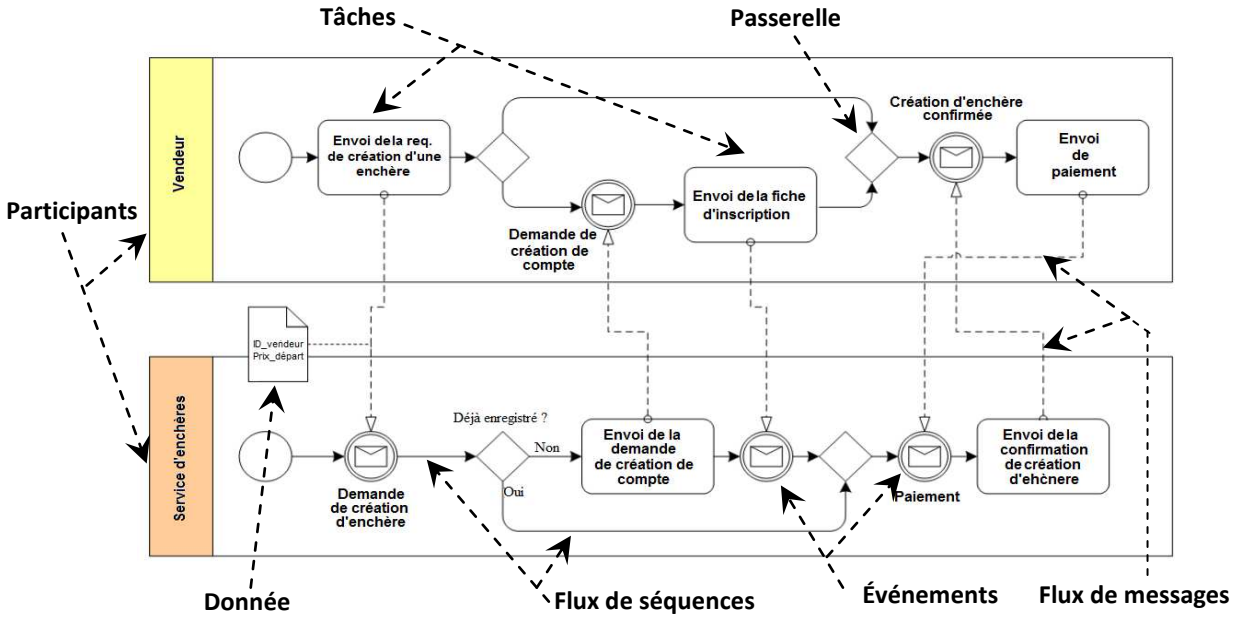


Figure 1.9 *Processus de vente aux enchères modélisé par la notation BPMN*

1.9.1.2 Le diagramme d'activités UML

Le langage UML *"Unified Modelling Language"* est un standard proposé par l'OMG *"Object Management Group"* permettant d'exprimer des besoins fonctionnels et techniques dans un environnement de développent orienté objet en utilisant plusieurs diagrammes :

Le diagramme de classes, le diagramme d'activités, …etc. et plusieurs concepts pour augmenter la sémantique de ces modèles : stéréotype, profil, … etc.

Le diagramme couramment utilisé en UML pour modéliser les processus est le diagramme d'activités. Il permet de décrire le comportement des processus sous la forme de flux ou d'enchaînement d'activités. La figure 1.10 montre les éléments graphiques des diagrammes d'activités.

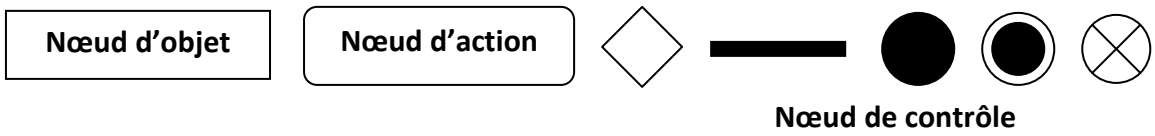


Figure 1.10 *Représentation graphique des principaux concepts dans un diagramme d'activités.*

La figure 1.11 représente un exemple de diagramme d'activité illustrant l'utilisation de nœuds de contrôle. Ce diagramme décrit la prise en compte d'une commande.

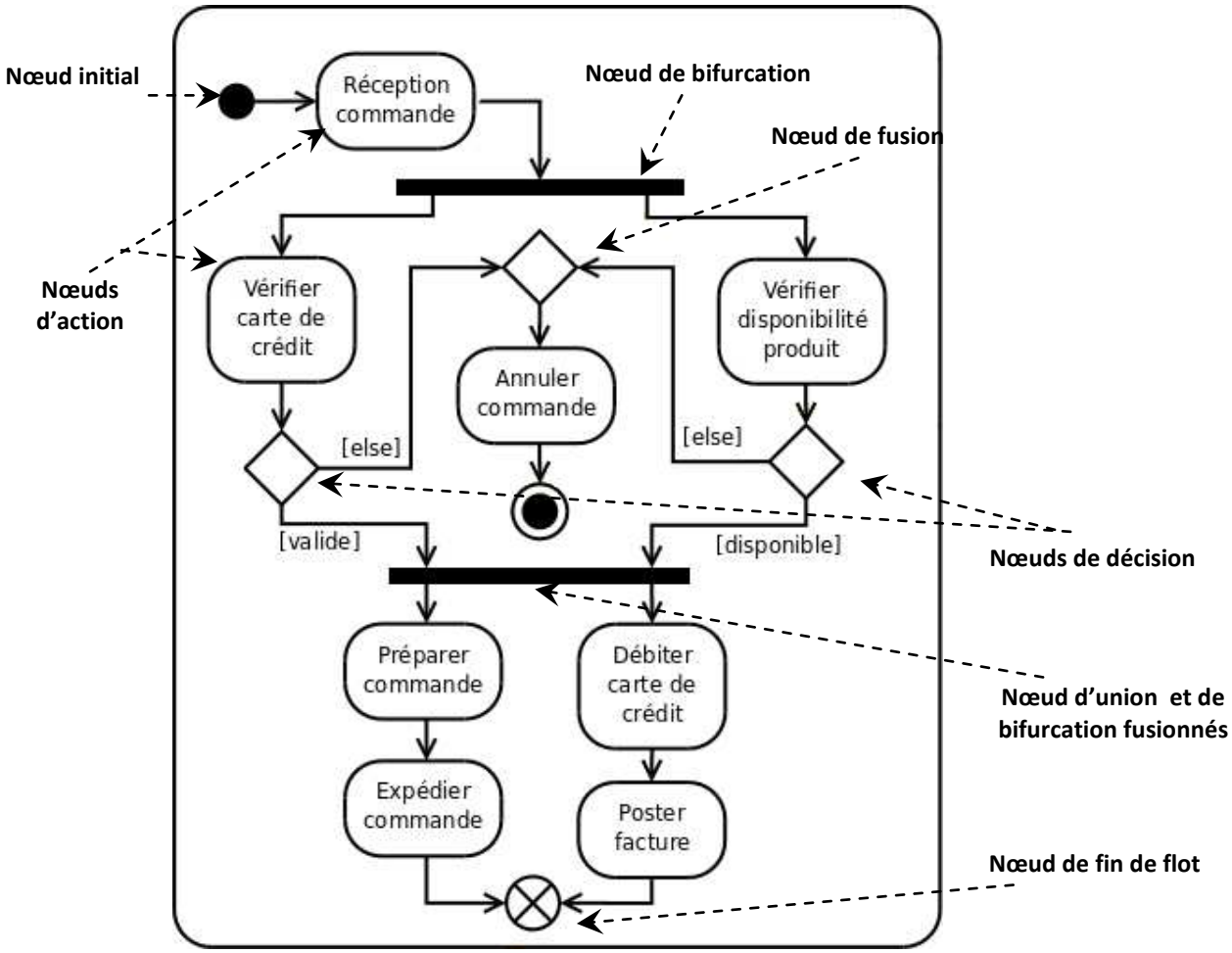


Figure 1.11 *Processus de commande modélisé par le diagramme d'activités*

1.9.1.3 Le paradigme de logique déontique

La logique déontique est utilisée dans (Goedertier & Vanthienen, 2006) à travers le langage PENELOPE pour modéliser d'une manière déclarative un processus métier, et cela, en se basant sur les axiomes temporels et déontiques pour modéliser un processus. Cette logique permet de formaliser les variantes possibles dont l'obligation, l'interdiction, la permission et le facultatif d'une exécution d'une activité métier. Cela permet de tenir compte des obligations et des autorisations dans les interactions métier.

1.9.1.4 Le paradigme de la logique temporelle linéaire (LTL)

Le langage ConDec proposé par (Pesic & Van der Aalst, 2006) permet de modéliser d'une manière déclarative un processus métier en utilisant la théorie des automates et la

logique temporelle linéaire (LTL) (Rozier, 2011), (Hornus & Schnoebelen, 2002) comme le montre la figure 1.12. Ces modèles peuvent être exécutés par un moteur spécifique.

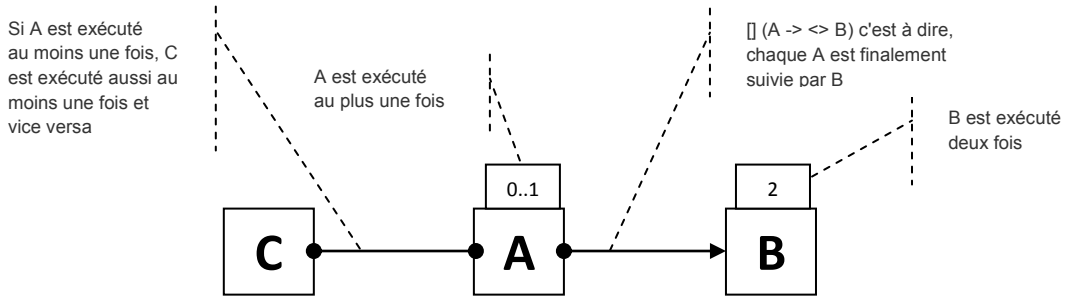


Figure 1.12 *Un exemple simple d'un modèle CONDEC.*

1.9.1.5 Le paradigme de modélisation basée sur les règles

Le paradigme de modélisation basée sur les règles ou *"Rule based modeling"* permet de modéliser la logique du processus par un ensemble de règles en utilisant des langages déclaratifs. L'exécution de ces langages est assurée par des moteurs d'inférence de règles qui déterminent l'ordre d'exécution des activités en fonction de l'évaluation des conditions. Dans ce paradigme, le formalisme *"Event-Condition-Action",* ou appelé aussi *"ECA"* a été adopté par de nombreux langages de modélisation des processus métier basés sur les règles comme par exemple les travaux de (Zeng, Ngu, Benatallah, & O'Dell, 2001) qui proposent par ailleurs de voir le processus comme un ensemble de tâches coordonnées entre elles par des règles ECA et d'utiliser les agents pour encapsuler les services qui exécutent les tâches du processus. Le langage BPTrigger, proposé par Chulsoon *et al.* dans (Chulsoon & Injun, 2004), pour modéliser et exécuter un processus métier complexe dans un environnement distribué et hétérogène en se basant sur le formalisme ECA.

1.9.1.6 Les diagrammes EPC (chaîne de processus événementielle)

"Event-Process Chains" (EPC) ou Chaînes de processus événementielles est un concept de modélisation développé dans le cadre de l'outil ARIS *"Architecture of integrated Information System"* d'IDS-Scheer (ARIS, 2007). Il permet une description des différentes dépendances temporelles et logiques dans un processus métier en sélectionnant ou en agrégeant un ensemble d'événements pour lancer l'exécution d'une activité spécifique. Autrement dit, EPC représente un processus métier comme une succession de fonctions et d'événements.

Un diagramme EPC est constitué d'éléments graphiques comme le montre l'exemple dans la figure 1.13 représentant les fonctions (activités), les unités organisationnelles (les participants), les événements, les opérateurs logiques et des objets divers comme les ressources utilisées. Il utilise des connecteurs logiques pour permettre de construire des événements complexes.

Pour élaborer un diagramme EPC, il faut prendre en considération cinq règles selon (Briol, 2008).

- Un processus doit commencer toujours avec un événement.
- Un processus doit se terminer toujours avec un événement.
- Les fonctions et les événements sont alternés.

- Les fonctions et les événements disposent au moins d'une entrée et d'une sortie.
- Un événement est un élément sans capacité de décision.

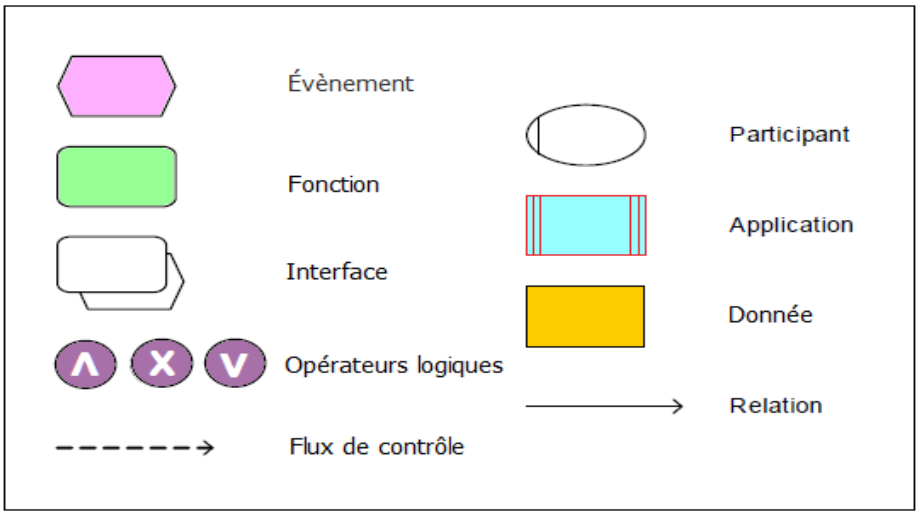


Figure 1.13 *Éléments de base d'un diagramme EPC.*

Le diagramme de la figure 1.14, représente un exemple de diagramme EPC illustrant l'utilisation des différents éléments détaillés ci-dessus. Ce diagramme décrit un processus de traitement des commandes.

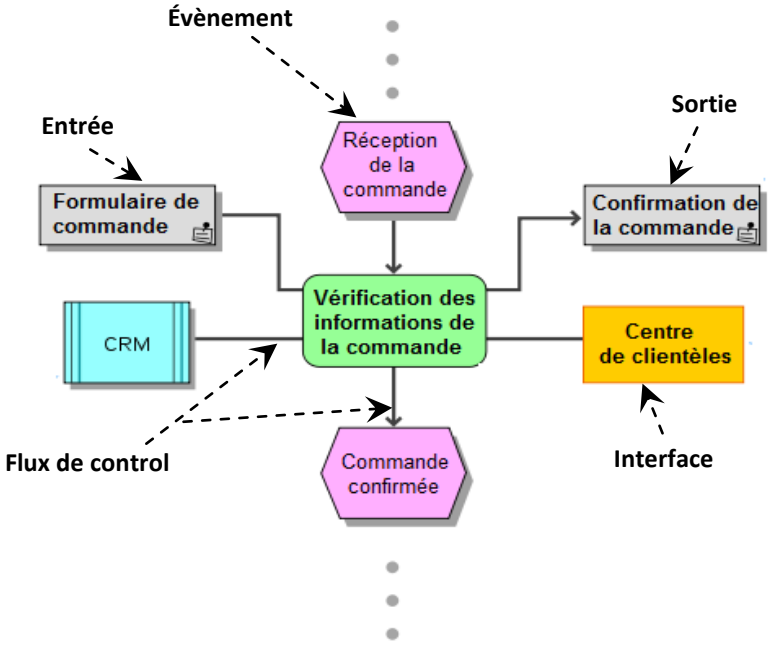


Figure 1.14 *Processus de traitement des commandes modélisé en EPC.*

1.9.2 Les langages d'exécution des processus métier

Bien que le "*Business Process Diagram*" permet de représenter d'une manière simple et compréhensible les différentes activités et leurs ordres d'exécution, Il est notamment impossible de l'exploiter directement par des moteurs d'exécution en raison de son niveau d'abstraction élevé. A cet égard, les langages d'exécution des processus métier tel

que BPEL (Matjaz, 2006), (Farahbod, Glässer, & Vajihollahi, 2005) ont pour but de spécifier le déroulement de l'ensemble des activités d'un processus. Ces langages qui sont interprétés par des moteurs d'exécution ou d'orchestration, utilisent une sérialisation XML pour décrire l'implémentation du processus.

Les langages tels que XPDL (The Workflow Management Coalition, 2008) et ebXML (OASIS, 2003) permettent de modéliser les flux d'information échangés entre les différents acteurs et les activités à accomplir par ces différents acteurs. Tandis que WSFL (Leymann, 2001), XLANG (Thatte, 2001), et BPEL4WS (OASIS, 2007) permettent d'orchestrer les services web mis en œuvre au sein d'un processus. Ces langages exploitent les capacités d'extension de WSDL.

Dans la figure 1.15, une chronologie des différents langages permettant d'exécuter les processus métier est donnée.

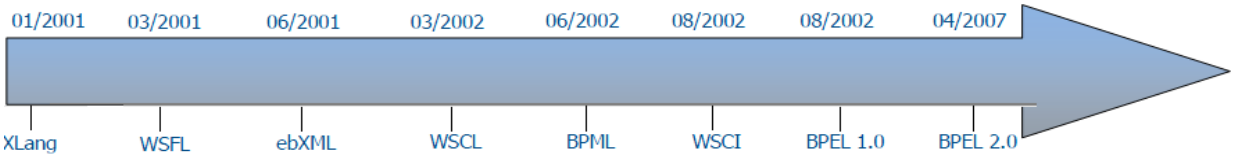


Figure 1.15 *Chronographe des différents langages d'exécution des processus métier*

1.9.2.1 Le langage XPDL

Le langage XPDL ou *"XML Process Definition Language"* est adopté par la plupart des moteurs de workflow. Il est proposé par Workflow Management Coalition et permet de définir un processus métier à l'aide du langage XML, en définissant les activités, les interrelations, les attributs qualifiant certains comportements de l'activité, les transitions, les partenaires et les interactions entre eux.

1.9.2.2 Le langage ebXML

Le langage ebXML *"Electronic Business using eXtensible Markup Language"* est né du besoin d'avoir une suite de spécifications pour le projet commun entre OASIS *"Organization for the Advancement of Structured Information Standards"* et UN/CEFACT *"United Nations Centre for Trade Facilitation and Electronic Business"*. Il permet d'assurer les échanges électroniques professionnels *"B2B"* de manière interopérable. Il propose des spécifications telles que ebMS pour décrire les transmissions de messages, CPPA pour décrire les protocoles de collaborations et ebBP pour décrire les processus métier.

1.9.2.3 Le langage WSFL

Le langage WSFL *"Web Services Flow Language"* est un langage basé sur XML, développé par IBM en 2001 afin de fournir une description de la composition de services web sur deux niveaux : le premier niveau appelé modèles de flux *"flow models"* permet de spécifier comment réaliser un but métier particulier dans ce cas, le résultat est une description d'un processus métier et le deuxième niveau appelé modèles globaux *"global models"* spécifie le modèle d'interaction d'une collection de services de Web, dans ce cas, le résultat est une description des interactions globales entre partenaires.

1.9.2.4 Le langage XLANG

Le langage XLANG *"eXtensible LANGuage"* est un langage de composition de services web développé et normalisé par Microsoft en 2001 pour sa plate-forme de gestion de processus BizTalk et cela afin d'apporter des solutions aux lacunes existantes dans le langage de description WSDL. Il est basé sur des normes internet tels que XML, XSD *"XML Schema Definition"* et WSDL. XLANG assure l'automatisation des processus métier et permet l'orchestration des services web par la spécification du comportement d'échange de messages. Ce langage utilise des structures simples tel que les Actions qui correspondent aux opérations WSDL, les éléments de programmation structurée tels que : *while, switch*, …etc, ou encore les éléments permettant le parallélisme et la synchronisation des éléments de base. Ce langage à disparu au profit de la norme connue sous le nom de BPEL.

1.9.2.5 BPEL

La spécification BPEL *"Business Process Execution Language"* est un langage destiné à l'exécution des processus métier orienté blocs ou *"bloc-oriented"*. Il permet de définir le processus, ou l'enchaînement et la logique des actions qui seront exécutées par le moteur d'orchestration. Il est issu des langages WSFL (Leymann, 2001) et XLANG (Thatte, 2001) , et est dérivé de XML. Il supporte les protocoles standards des services web tel que SOAP, WSDL, UDDI, WS-Reliable Messaging et les protocoles de transport d'Internet (Weerawarana, Curbera, Leymann, Storey, & Ferguson, 2005).

Le nom exact de cette spécification du consortium OASIS est BPEL4WS (OASIS, 2007) *"Business Process Execution Language for Web Services"*. Ce qui pourrait induire en erreur car un processus BPEL ne peut se résumer à l'orchestration des services web mais il peut également inclure des connecteurs transactionnels comme les EJB ou des procédures stockées en SQL.

Les principales caractéristiques du langage BPEL4WS sont les suivantes :

- Exprimer les processus métier par un langage standard.
- Décrire la logique des processus métier à travers la composition de services web.
- Invoquer les opérations du service web dans un ordre séquentiel ou parallèle.
- Garantir la synchronisation de l'exécution des processus parallèles.
- Permettre d'annuler le traitement fait par une activité en cas d'échec.
- Maintenir l'exécution des activités pour une longue période.

Un fichier BPEL permet de décrire de manière détaillée chaque élément d'un processus métier ainsi que de représenter le parallélisme et la synchronisation. Sa structure globale est décrite comme suit:

La force du langage BPEL se manifeste dans la structure de son processus qui est basée sur un fichier XML composé de trois blocs principaux comme présenté ci-dessus : les services web qui participent dans la composition du processus (les partenaires), les variables manipulées dans le processus et les activités de traitement qui composent le processus métier.

```
<process name="processName"
  xmlns="http://docs.oasis-open.org/wsbpel/2.0/process/executable"
  targetNamespace="http://example.com"
  xmlns:tns="http://example.com"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <variables>
       <variable name="x" type="xsd:int"/>
  </variables>
  <partnerLinks>
       <partnerLink name="ResultRequester"
            partnerLinkType="tns:requestResultPartnerLinkType"
            partnerRole="requestResultAdder"
            myRole="requestResultRequester"/>
  </partnerLinks>
  <faultHandlers> ...</faultHandlers>
  <compensationHandlers>...</compensationHandlers>
  [Activités]* (atomiques et composées)
</process>
```

La balise <process> est l'élément racine du fichier BPEL. Elle permet la description complète du processus. Grâce à l'attribut name, on peut attribuer un nom au processus. Tandis que la balise <variables> permet de représenter les différentes variables qui sont utilisées par les activités et qui sont utilisées aussi dans la description des comportements du processus. Autrement dit, elles sont utilisées pour détenir les données relatives à l'état interne du processus, pour stocker, reformater et transformer des messages avec les services web invoqués. Ces variables peuvent servir aussi à déterminer les types de messages échangés au sein du processus. La balise <Partnerlinks> permet de représenter les participants qui ont pour rôle d'exécuter les activités du processus et cela en définissant les liaisons entre les actions définies dans le fichier WSDL (via partnerLinkType) au processus BPEL. L'attribut myRole ou partnerRole définit si c'est une action qui appelle le processus ou si c'est une action appelée par le processus

Il existe deux types d'activités: les activités atomiques ou dites aussi de base et les activités composées ou dites aussi structurées. BPEL utilise un ensemble de balises pour définir des activités atomiques comme par exemple la balise <invoke> qui permet d'invoquer une opération d'un partenaire ou un service web partenaire. Cette invocation peut être réalisée de manière synchrone ou asynchrone. La balise <receive> permet de recevoir un message d'une source externe (partenaire) et la balise <reply> permet à un processus de répondre à un message qu'il aurait reçu par l'activité receive. La balise <assign> permet de fournir une méthode pour manipuler les données telle que la copie ou l'affectation du contenu entre les variables ou expressions. La balise <validate> permet de valider la valeur des variables dans leurs schémas de définition et enfin la balise <Empty> permet d'insérer une instruction non-opérationnelle qui ne fait rien dans un processus.

Les activités structurées sont des activités composées. Elles décrivent l'ordre des flux de contrôle et par conséquence décrivent les comportements du processus. Elles peuvent être formées d'autres activités (atomiques ou structurées) d'une manière récursive.

Trois catégories de flux de contrôle dans les activités structurées : traitements séquentiels, parallèles et sélectifs. Elles utilisent différentes balises tels que la balise <se-

quence> qui permet d'exécuter de façon séquentielle une ou plusieurs activités BPEL. La séquence se termine elle-même lorsque la dernière activité du processus est achevée. La balise <flow> permet d'exécuter en parallèle plusieurs processus. Ces différents processus peuvent être indépendants. Aussi, elle ne termine son exécution que si tous les processus sont achevés en assurant une synchronisation totale. La balise <If> permet d'ajouter une logique conditionnelle à la définition du processus BPEL. L'utilisation des éléments elseif ou else qui sont facultatifs permet la définition d'une ou plusieurs branches conditionnelles au sein de l'activité If. La balise <switch> décrit l'acheminement conditionnel.

1.10 La traduction des langages graphiques vers les langages d'exécution

La traduction automatique des langages graphiques vers les langages d'exécution a pour but d'offrir un modèle exploitable et compréhensible par les moteurs d'exécution. A cet effet, plusieurs auteurs ont proposé un ensemble de méthodes et d'algorithmes afin de traduire automatiquement une modélisation graphique vers un modèle BPEL comme l'algorithme décrit dans (Ouyang, Van der Aalst, Dumas, & ter Hofstede, 2006) qui permet de traduire un modèle BPMN en un modèle BPEL (*cf.* figure 1.16), ou encore (Gardner, 2009) qui propose un algorithme pour traduire un modèle exprimé en UML en un modèle BPEL.

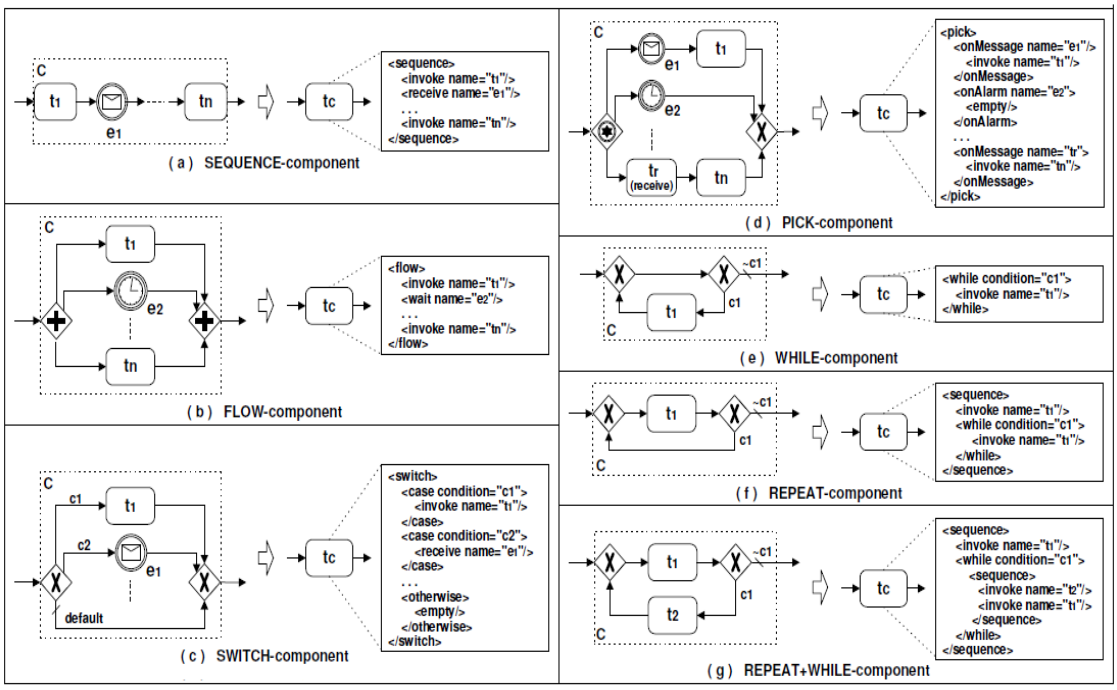


Figure 1.16 *Transformation de BPMN à BPEL*

Cela dit, cette traduction automatique demeure imparfaite puisque les aspects graphiques comme les partitions et les sous-partitions des modèles BPMN sont perdus. D'autre part, le modèle BPEL obtenu peut nécessiter des ajustements de code. Par exemple, les variables contenues au sein des activités peuvent nécessiter d'être déclarées au préalable comme assignées à des paramètres de services web.

1.11 Les règles métier

Une règle métier ou règle de gestion ou encore *"Business Rule"* en anglais est une directive qui permet de gouverner, d'influencer les prises de décision, et de contrôler les comportements métiers et l'information au sein d'une organisation. Selon le groupe BRG *"Business Rules Group"*, les règles métier sont « *des définitions de haut niveau structurées, qui permettent de contraindre, contrôler et influencer un aspect du métier* » (Group T. B., 2000). Ces règles sont donc utilisées pour implémenter les stratégies ou les politiques d'une organisation (zur Muehlen & Indulska, 2010) mais elles sont aussi utilisées pour modéliser un processus métier d'une manière déclarative (Knolmayer, Endl, & Pfahrer, 2000). Il existe cinq catégories de règles métier selon (Wagner, 2005) :

- ***Les règles d'intégrité*** *:* Il s'agit des contraintes ou assertions qui doivent être satisfaites. Exemple : Un projet doit avoir un et un seul chef de projet ;
- ***Les règles de dérivation*** *:* Il s'agit d'une ou plusieurs conditions et d'une ou plusieurs conclusions. Exemple: Le client *"platinum"* reçoit une remise de 5%. Cela se traduisant par le fait que si Dupond est un client *"platinum"*, alors Dupond recevra une remise de 5% ;
- ***Les règles de réaction*** *:* Il s'agit des règles qui se déclenchent par des occurrences d'événements (Event, Condition, Action, Alternative-action, Post-condition) et qui exigent une satisfaction de conditions pour exécuter des actions. Exemple : à la réception d'une demande de prêt, si le client n'est pas endetté alors le prêt est approuvé ;
- ***Les règles de production****:* Il s'agit d'une ou plusieurs conditions et d'une ou plusieurs actions (condition, action). Exemple: Si il n' ya pas de défaut dans la carrosserie de la voiture et que le contrôle technique se révèle normale, alors la voiture peut circuler normalement ;
- ***Les règles de transformation*** *:* Il s'agit des règles qui contrôlent le changement d'état du système. Exemple : la date du jour doit être changée de manière incrémentale;

1.12 Caractéristiques des modèles & langages de processus métier

Comme soulevé précédemment, la phase de modélisation constitue une étape primordiale dans le cycle de gestion d'un processus métier. Elle permet de mettre en exergue les spécifications et les connaissances métier d'une organisation à des fins d'exploitation et de réutilisation tout en séparant la logique « processus » de la logique « application », et cela afin d'offrir un moyen de dialogue entre les experts métier et les équipes fonctionnelles. Cette modélisation des processus métier doit prendre en compte plusieurs caractéristiques saillantes telles que définies dans (Lu & Sadiq, 2007) :

- ***L'expressivité*** *:* la puissance d'expressivité d'un langage de modélisation de processus métier représente sa capacité à fournir une description complète des différents éléments d'un processus (Sadiq & Orlowska, 1999), (Sadiq & Orlowska, 1997) ;
- ***La flexibilité*** *:* la capacité d'un processus métier à s'exécuter d'une manière stable sur la base d'une perte et/ou d'une spécification partielle du processus suite à un

changement (Schonenberg H. , Mans, Russell, Mulyar, & van der Aalst, 2008), (Burkhart & Loos, 2010);

- ***L'adaptabilité*** *:* la capacité d'un processus métier à réagir à des exceptions imprévisibles, qui peuvent survenir généralement durant l'exécution de ses instances (Weber, Wild, Lauer, & Rei, 2006)
- ***La complexité*** *:* la capacité à mesurer la difficulté à modéliser, vérifier et déployer un processus métier ainsi que de supporter les changements dynamiques d'un processus métier en constante évolution (Cardoso J. , 2006), (Latva-Koivisto, 2001);
- ***Le dynamisme*** *:* la capacité d'un modèle de processus métier à changer lorsque le processus évolue. Cette évolution peut être due à l'amélioration des processus, ou à l'innovation de procédés ou a la réingénierie des processus (Schonenberg H. , Mans, Russell, Mulyar, & van der Aalst, 2008);

1.13 Les dimensions de la modélisation d'un processus métier

Afin d'obtenir des modèles qui répondent aux différentes caractéristiques détaillées précédemment, il faut prendre en considération plusieurs éléments. Selon (Jablonski & Bussler, 1996), les éléments clés d'un processus métier peuvent être classés en cinq groupes. Chaque groupe représente une dimension qu'un processus métier doit posséder. Ces cinq dimensions sont : la dimension fonctionnelle, la dimension comportementale, la dimension informationnelle, la dimension organisationnelle et la dimension opérationnelle.

1.13.1 La dimension fonctionnelle

La dimension fonctionnelle concerne l'identification des différentes activités d'un processus métier que l'on souhaite modéliser avec une certaine hiérarchie. En effet, Il existe deux types d'activités (1) les activités atomiques dites aussi Tâches ou *"Tasks"* en anglais et (2) les activités composées appelées aussi sous-processus ou *"sub-process"* en anglais. L'ensemble de ces activités représente un ensemble d'actions qui s'exécutent d'une manière indivisible par un participant afin de réaliser un objectif global de ce processus comme définie par le WfMC « *une activité est une description d'une unité de travail qui forme une étape logique dans un processus* » (Coalition, 1999). L'exécution d'une activité peut être manuelle dans le cas où elle exige une intervention humaine, ou automatisée dans le cas où elle s'exécute par une machine. Le modèle fonctionnel doit aussi représenter le flux de données associées aux activités et les interdépendances de données entre les activités (data flow).

1.13.2 La dimension comportementale

La dimension comportementale, appelée aussi dimension de coordination représente un aspect primordial du processus métier puisqu'il correspond à la dynamique du processus. Ce comportement s'exprime par la modélisation d'un flux de contrôle entre les différents éléments à exécuter dans un processus et plus précisément les dépendances entre les diverses activités. Ce dernier permet d'indiquer la chronologie de l'exécution. Autrement dit, les relations logiques qui contrôlent l'acheminement et l'ordre d'exécution de

ces activités. Ces relations peuvent être définies d'une manière séquentielle ou parallèle avec des points de synchronisation et/ou de disjonction entre activités (Russell, Ter Hofstede, & Mulyar, 2006). De plus, la dimension comportementale doit représenter les événements qui permettent de déclencher ces activités.

1.13.3 La dimension informationnelle

La dimension informationnelle représente l'ensemble des informations et des données qui sont produites ou manipulées par un processus métier et qui sont associées aux activités. Il décrit en détail les relations qui existent entre les données, leur type et leur structure. Ces entités peuvent être des données simples (des chaînes de caractères, des dates, des entiers, etc.), des données complexes (des ensembles, tableaux) ou des documents (Russell, Ter Hofstede, Edmond, & van der Aalst, 2005).

1.13.4 La dimension organisationnelle

Comme son nom l'indique, la dimension organisationnelle permet d'établir des liens hiérarchiques entre les ressources qui ont la responsabilité d'exécuter les éléments d'un processus ainsi que des relations entre unités organisationnelles ou départements. Ces ressources qui sont une personne « acteur», une application « service-web » ou une entité sont appelées aussi participants parce qu'elles participent à la réalisation de l'objectif global du processus. Un ensemble de patrons pour décrire les différentes manières dont les ressources sont représentées et utilisées dans un processus est donné par (Russell, van der Aalst, Ter Hofstede, & Edmond, 2004).

1.13.5 La dimension opérationnelle

La dimension opérationnelle permet de détailler l'aspect opérationnel d'un processus métier. Ces détails représentent, en effet, les informations techniques utilisées lors de l'exécution du processus comme : le format des messages échangés, le mode d'invocation (synchrone ou asynchrone) et les protocoles de transport utilisés.

1.14 Réactivité, flexibilité et agilité d'une entreprise

La réactivité d'une entreprise représente en général sa capacité à réagir rapidement aux différents besoins de ses interlocuteurs par une synergie entre « l'optimisation de ses méthodes de travail » et « l'amélioration de ses ressources » et cela afin de garder un avantage concurrentiel sur ses concurrents (Ross, Rhodes, & Hastings, 2008). Cette réactivité passe par une série de changements qui peuvent être vus comme une réponse à un besoin d'amélioration et d'optimisation lié à l'atteinte d'un certain seuil d'insatisfaction dans le système d'information d'une organisation. Ainsi, Il est alors plus facile de constater que la réactivité est une conséquence de la construction ou la reconstruction d'un SI afin d'assurer une amélioration tangible de l'efficacité d'une organisation, et qui est accomplie généralement grâce aux technologies de l'information (IT). La réactivité doit passer d'abord par la flexibilité des ressources, aussi bien humaines que technologiques d'une organisation. Cette flexibilité de gestion permet d'obtenir l'agilité souhaitée de l'entreprise (Mooney, Beath, Fitzgerald, Ross, & Weill, 2003), (Boucher, 2007). Autrement

dit, ces ressources doivent être rapidement adaptables (Ross, Rhodes, & Hastings, 2008) d'une façon agile afin d'atteindre l'objectif de la réactivité d'une organisation.

Notons que les concepts d'agilité et de flexibilité peuvent prêter à confusion en théorie. Cependant, ces deux termes présentent des différences notables. La flexibilité est semblable à la notion de réactivité industrielle, à savoir la capacité d'une organisation à répondre rapidement aux différents besoins de ses clients par une attribution différente de ses ressources tandis que l'agilité représente la capacité d'une organisation à s'adapter rapidement à son environnement, permettant ainsi de procéder à des changements harmonieux et continus de l'entreprise.

1.15 La flexibilité de la modélisation des processus métier

La flexibilité ne cesse de susciter un intérêt majeur des organisations et cela afin d'améliorer leur productivité, et leur rapidité d'adaptation aux changements. Dans ce contexte et pour faire face à l'évolution des processus et des exceptions qui peuvent survenir durant l'exécution de ces derniers, la flexibilité des modéles de processus métier se définit comme une propriété des modèles à être modifiés sans provoquer l'instabilité du système où ils sont utilisés (Kumar & Narasipuram, 2006). Autrement dit, la flexibilité qui constitue une caractéristique élémentaire des processus métier représente la capacité de mettre en œuvre des changements au niveau du schéma *"Process Type Level"* ou au niveau des instances *"Process Instance Level"* d'un processus métier en modifiant uniquement les parties qui ont besoin d'être changées tout en conservant la stabilité des autres parties du processus (Soffer, 2005), (Regev & Wegmann, 2005) et cela en définissant des mécanismes permettant d'assurer la cohérence du modèle métier et du modèle exécutable. Il est aussi soutenu que la flexibilité ne doit pas se résumer qu'aux processus métier mais aussi aux règles métier qui permettent de capturer la sémantique de la politique et de la réglementation d'une organisation (Goedertier & Vanthienen, 2006).

1.16 Les taxonomies de la flexibilité des processus métier

Il existe des travaux dans la littérature qui ont tenté de proposer une taxonomie qui soit la plus générique possible de la flexibilité des processus métier. Parmi ces travaux on peut citer ceux de (Regev, Soffer, & Schmidt, 2006) , (Schonenberg M. , Mans, Russell, Mulyar, & van der Aalst, 2007) et (Nurcan, 2008).

La taxonomie proposée par Regev *et al* (Regev, Soffer, & Schmidt, 2006), s'intéresse aux changements qui peuvent survenir durant le cycle de vie d'un processus métier (*cf.* figure 1.17). Elle prend en considération trois dimensions de changement :

- Le niveau d'abstraction du changement ou le niveau d'application du changement qui peut être appliqué soit au niveau du schéma ou spécification du processus *"Process Type Level"*, soit au niveau de l'instance du processus *"Process Instance Level"*.
- L'objet du changement qui peut concerner les différents aspects ou dimensions du processus métier. En effet, le changement peut concerner les activités du processus (dimension fonctionnelle), les flots de contrôle (dimension comportementale),

les données du processus (dimension informationnelle) ou les différents protocoles utilisés dans le processus (dimension opérationnelle).

- Les propriétés du changement tel que l'ampleur du changement *"Extent of change"* qui peut être progressive pour changer une partie du processus ou révolutionnaire ou radical pour créer un nouveau processus. La durée du changement *"Duration of change"* qui peut être temporaire c.à.d. limité dans le temps ou permanent c.à.d. valable jusqu'au prochain changement. La rapidité du changement *"Swiftness of change"* qui concerne l'instantanéité du changement. Autrement dit, qui peut être soit pris immédiatement en considération ou en différé; et l'anticipation du changement " *Anticipation of change*" qui peut être soit ad hoc pour faire face à des situations exceptionnelles ou planifié et qui fait souvent partie d'une refonte du processus.

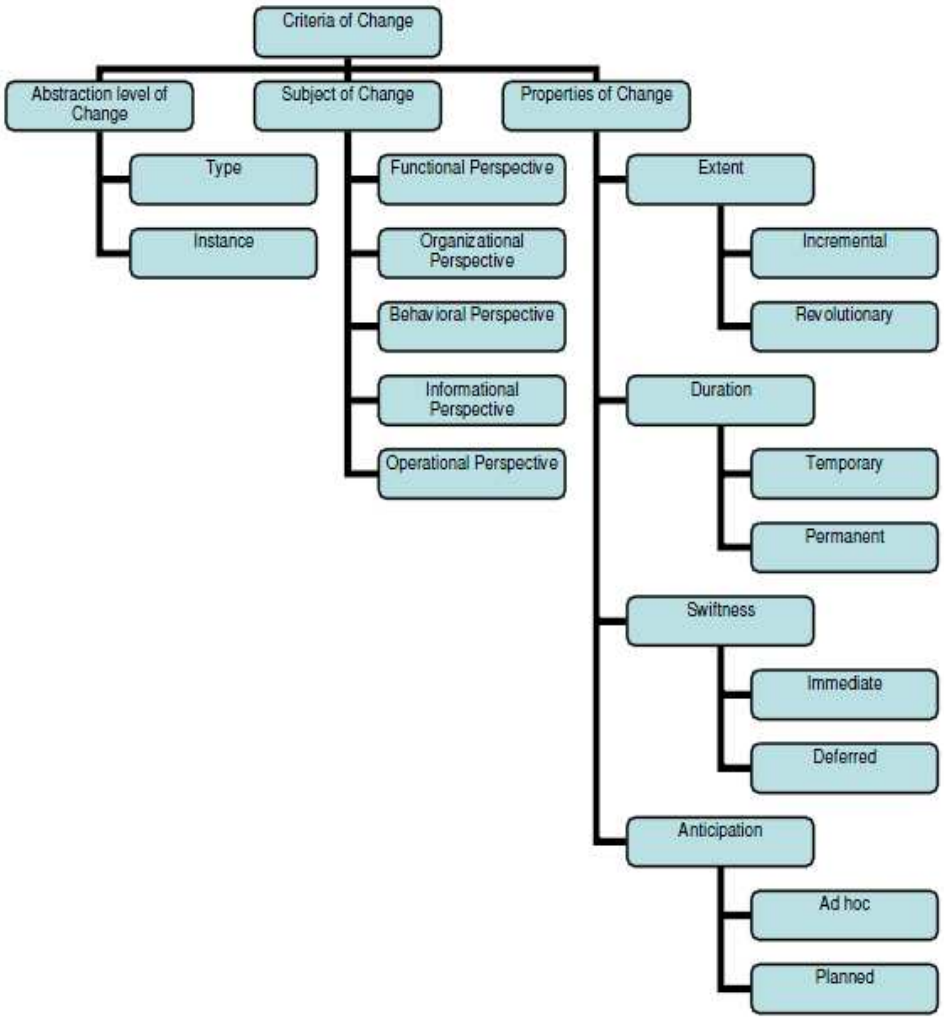


Figure 1. 17 *Les notions de changement selon la classification de Regev et al.*

Schonenberg *et al* (Schonenberg M. , Mans, Russell, Mulyar, & van der Aalst, 2007), proposent une taxonomie qui s'intéresse à la flexibilité du point de vue opérationnel. Ils proposent quatre types de flexibilités :

- La flexibilité par conception *"Flexibility by design"* qui est la capacité d'intégrer des chemins d'exécution alternatifs au moment de la conception de telle sorte que la

sélection se porte sur le chemin le plus approprié lors de l'exécution de chaque processus et cela en fournissant la possibilité d'exprimer le parallélisme entre les activités. L'instanciation multiple d'une activité dans la même exécution, l'itération de l'exécution d'une activité ou encore l'annulation de l'exécution d'une activité.

- La flexibilité par déviation *"Flexibility by deviation"* qui est la possibilité de dévier l'exécution d'une séquence d'une instance du processus de sa trajectoire initiale prescrite par le procédé original sans pour autant modifier la définition du processus et cela en permettant par exemple d'annuler, de défaire, de refaire ou de court-circuiter certaines activités.
- La flexibilité par spécification partielle *"Flexibility by Underspecification"* est la capacité d'exécuter un processus spécifié partiellement au moment de l'exécution, à savoir celle qui ne contient pas suffisamment d'informations pour lui permettre d'être achevé. Autrement dit, la possibilité de définir certaines structures de processus dans la phase d'exécution en permettant la sélection tardive d'un fragment de processus parmi plusieurs alternatives ou la modélisation tardive pour spécifier des fragments durant l'exécution du processus.
- La flexibilité par changement *"Flexibility by Change"* est la possibilité de modifier la définition du processus durant la phase d'exécution en permettant à l'une ou à l'ensemble des instances du processus en cours d'exécution de migrer vers la nouvelle définition du processus.

La taxonomie proposée par Nurcan (Nurcan, 2008) s'intéresse principalement à la nature de la flexibilité et comment cette dernière peut-elle être prise en compte, c.à.d. au moment de la conception *"Design-time"* ou l'exécution *"Runtime"* du processus métier. Pour cela elle propose deux types de flexibilités comme le montre la figure 1.18 :

- La flexibilité par adaptation (a posteriori) permet l'adaptation du modèle de processus métier ou ses instances durant leurs exécutions. C'est le cas le plus fréquent dans la littérature. Le principal défi consiste à savoir (1) Quand et selon quels critères cette adaptation devrait être faite ? (2) Sur quel niveau doit-elle être appliquée c.à.d. au niveau du schéma de processus ou au niveau de ses instances ? (3) Comment peut-on faire face aux instances qui sont en cours d'exécution? Les approches qui offrent ce genre de flexibilité sont basées sur des formalismes de modélisation qui considèrent que la définition de processus qui en résulte n'est pas vraiment flexible, mais plutôt adaptatif. Autrement dit, ces approches ne peuvent pas anticiper la capacité de changer lors de la phase exécution des modèles de processus. Elles sont trop rigides pour être capables de capturer la nature dynamique des processus métier.

- La flexibilité par sélection (*a priori*) est basée sur des formalismes de modélisation qui peuvent offrir une capacité à traiter avec le changement de l'environnement de processus sans aucune évolution dans la définition de ce dernier (Weske, 2001), (Rinderle, Reichert, & Dadam, 2003). Cela signifie que la capacité de flexibilité de ces processus devrait être incorporée dans leurs définitions pendant la phase de modélisation. de manière à ce que les instances de processus soient toujours conformes à la définition de ces processus.

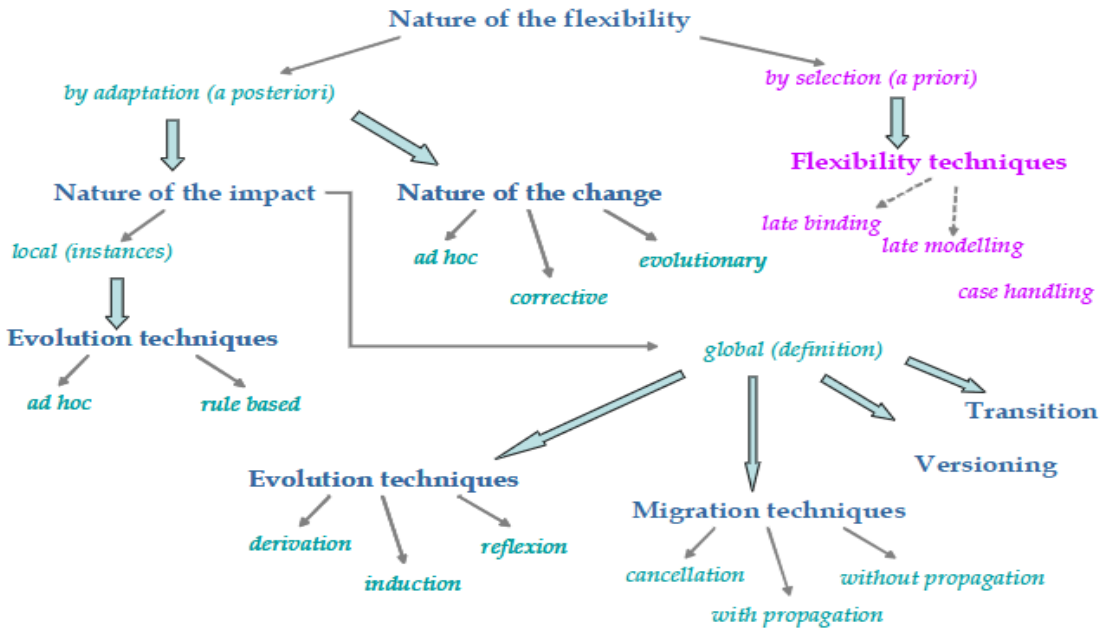


Figure 1.18 *Les notions de changement selon la classification de Nurcan*

1.17 Gouvernance du changement des processus métier

La gestion des processus métier est une approche managériale proposée pour concevoir, exécuter et piloter les processus métier d'une organisation. Elle permet d'avoir une meilleure vue sur la cartographie des procédures d'entreprise, du management des workflow et du pilotage des processus métier. Toutefois, ces processus métier suscitent un besoin permanent d'évolution et cela afin de réagir rapidement aux mouvements du marché, et aux transformations métiers des différents interlocuteurs d'une organisation.

Les processus doivent être flexibles afin de faire face aux exigences de changements. En d'autres termes, la gestion de l'évolution des processus métier est essentielle pour permettre aux organisations concernées d'être en phase avec les besoins et l'environnement de leurs interlocuteurs. Ainsi, la capacité de flexibilité et d'adaptabilité des processus métier à été identifiée comme étant l'un des facteurs critiques de succès pour tout modèle de processus métier appelé aussi *"Business Process Model"* (Heinl, Horn, Jablonski, Neeb, Stein, & Teschke, 1999), (Reijers, 2006). A contrario, l'incapacité d'appliquer avec succès les changements nécessaires peut souvent aboutir à un dysfonctionnement dans la gestion des processus métier et à des conséquences graves sur le plan économique.

Cette exigence de flexibilité à donner naissance à plusieurs champs de recherche comme la prise en compte du besoin de réingénierie de processus tel que la flexibilité, et l'adaptation au changement durant la phase de modélisation ou encore la vérification du bon fonctionnement du processus par des modèles formels ou par simulation. Toutefois, ces changements comme détaillés dans le chapitre suivant peuvent être source de dégradation sur le plan structurel, fonctionnel, comportemental ou qualitatif du processus modifié et nécessiter une modélisation permettant de réaliser une analyse *a priori* de l'impact prévisionnel du changement du processus métier et une analyse a posteriori de l'impact effectif de ce changement.

1.18 Nos contributions

Les recherches que nous avons entamées dans cette thèse visent à proposer une contribution à la gouvernance du changement et la vérification des processus métier. Notre objectif est de proposer un mécanisme qui permet d'assurer à la fois une analyse *a priori* de l'impact du changement des processus métier en analysant l'impact structurel et qualitatif, et cela afin de déterminer l'impact d'une évolution sur l'ensemble du système et de vérifier la cohérence du modèle métier et du modèle exécutable. À partir de ces constats, nous avons mis en exergue les problèmes suivants (Kherbouche & Basson, 2011), (Kherbouche, Bouneffa, Ahmad, & Basson, 2013):

1. Quel sont les incohérences et les conséquences qui émanent d'un changement des processus métier ?
2. Quel sont les types d'impact qui peuvent être générés suite à ce changement ?
3. Quelles sont les parties impactées directement ou indirectement par ce changement ?
4. Comment peut-on vérifier la cohérence du modèle métier et du modèle exécutable après chaque changement ?
5. Comment garantir le respect des règles métiers, règles organisationnelles ou encore règles de législation suite à un changement de processus métier ?
6. Comment peut-on gérer le changement d'une règle métier tout en gardant la cohérence des processus métier existants ?
7. Comment peut-on évaluer la qualité d'un processus métier durant son cycle de vie et plus précisément après chaque changement ?

Notre solution aux problèmes énoncés précédemment se décline en quatre contributions formant une démarche globale de gouvernance du changement des processus métier allant de la définition du processus jusqu'à sa vérification, à savoir :

1. Nous proposons, d'abord, une approche basée sur la technique du "*model-checking*" afin de modéliser la sémantique d'exécution d'un processus métier et cela dans un but de vérifier formellement son bon fonctionnement.
2. La même technique est utilisée pour vérifier la conformité de ces modèles avec les règles de conformité après chaque changement de l'un des deux.
3. Nous proposons aussi une analyse *a priori* de l'impact du changement des processus métier qui aiderait ainsi les concepteurs et les experts métier à se rendre compte *a priori* des conséquences potentielles de l'évolution et d'en estimer le

coût. Cette analyse de l'impact du changement des processus métier est réalisée à travers une approche basée sur une définition de la relation de dépendance tout en tenant compte des spécificités des modélisations de ces processus.

4. Pour évaluer la qualité des processus métier durant tout leur cycle de vie et plus particulièrement après chaque changement, nous proposons une adaptation de certaines caractéristiques de la norme qualitative ISO/CEI 9126 qui sert d'habitude à évaluer la qualité des produits logiciels pour les processus métier. Cette adaptation est justifiée par la similitude structurelle existante entre un processus métier et un produit logiciel.

1.19 Conclusion

Nous avons présenté dans ce chapitre le contexte et le cadre théorique de nos travaux, nous avons d'abord souligné le rôle important des processus dans le système d'information des organisations, l'approche BPM qui permet de gérer ces processus métier durant leur cycle de vie, et les langages qui permettent de les modéliser et de les exécuter ces derniers. Nous avons aussi mis en exergue l'importance de l'adaptabilité et la flexibilité des processus métier afin de pouvoir faire face à la nature changeante des différents besoins des organisations, toute en présentant les différentes taxonomies de la flexibilité qui existent dans la littérature. Enfin nous avons présenté la problématique et les solutions apportées par ces travaux. Dans le chapitre suivant, nous allons présenter un aperçu des travaux de recherche qui portent sur la gestion et la gouvernance du changement des processus métier.

CHAPITRE 2

Vérification et intégration des processus métier et État de l'art

2.1 Introduction

La cohérence et la fiabilité des processus métier est une question primordiale pour toute organisation dans l'accomplissement de ses fonctionnalités. Un processus peu fiable ou erroné peut souvent aboutir à un dysfonctionnement dans la gestion du système d'information et par conséquent, causer des pertes coûteuses aux entreprises pouvant conduire à leur affaiblissement ou, dans des cas extrêmes, à leur faillite. A cet égard, les organisations doivent accorder à la fois une attention plus grande à la problématique de vérification et de l'intégration du changement des processus métier.

Deux axes principaux de recherche ont vu le jour dans ce domaine durant cette dernière décennie, l'un porte sur la vérification des processus métier et l'autre est relatif à l'intégration du changement et la gestion d'évolution de ces processus.

En effet, la vérification des processus métier a pour but de s'assurer que les activités du processus s'exécutent en conformité avec ce qui été prévu initialement et qu'elles n'ont pas induit des erreurs dans le résultat escompté et cela, tout en identifiant l'ensemble des incohérences provoquées par l'évolution d'un processus métier, et de proposer ainsi des solutions pour les résorber. L'intégration et la gestion du changement des processus métier a pour objectif principal d'incorporer les modifications des processus sans perturber leur environnement d'exécution matérialisé par les instances en cours d'exécution.

La suite de ce chapitre est organisée comme suit : Nous invoquons dans la section 2.2 l'importance de la cohérence des modèles de processus métier dans l'efficacité et la qualité de ces derniers. Dans la section 2.3, nous faisons un tour d'horizon sur les différentes techniques qui existent dans la littérature et qui permettent de vérifier la cohérence des modèles de processus métier. Dans la section 2.4, nous commençons par présenter une typologie des changements des processus métier, une typologie des propriétés qu'un changement de processus doit avoir. Enfin nous élaborons une synthèse des travaux de recherche qui existent dans le domaine de l'intégration et la gestion du changement des processus métier.

2.2 La cohérence des modèles de processus métier

La cohérence des processus métier appelé aussi en anglais *"Soundness"* est un point essentiel. Éviter les interruptions d'exécution de ces processus, permettre leurs suivis et leurs traçabilités sont autant d'atouts au service de l'efficacité et de la qualité de ces derniers.

L'objectif du processus de gestion de la cohérence est d'assurer la robustesse et la cohérence des modèles de processus métier, de manière automatique, lors de leurs évolutions. Ceci nécessite la mise en place d'un mécanisme de détection des incohérences et proposer ainsi des actions réparatrices pour les résoudre.

On peut définir l'incohérence d'un processus métier par le fait qu'un ou plusieurs éléments de ce processus tels que des activités, des données ou des ressources y sont employées en transgressant les relations sémantiques qui les lient et l'ensemble des contraintes et des règles qui régissent leurs comportements. Cette définition fait apparaitre

le terme « un ou plusieurs » car il est possible d'avoir une incohérence avec un seul élément de processus métier. Par exemple, l'indisponibilité d'une ressource au moment où une ou plusieurs activités en cours d'exécution veulent y accéder.

L'ensemble de ces incohérences qui peuvent survenir suite à un changement de processus métier sont généralement le fruit d'une mauvaise conception, d'une imprécision de modélisation des processus métier ou encore d'un dysfonctionnement lors de l'exécution de ces processus.

Ces incohérences peuvent être classées en quatre catégories distinctes qui sont : les incohérences structurelles, les incohérences fonctionnelles, les incohérences comportementales et les incohérences techniques.

2.2.1 Les incohérences structurelles

Une incohérence structurelle représente un dysfonctionnement d'une ou de plusieurs propriétés structurelles observables sur un modèle de processus métier, cette incohérence peut être la conséquence d'erreurs syntaxiques ou encore une violation de contraintes et de règles de conformité. En effet, les erreurs syntaxiques peuvent survenir suite à une mauvaise utilisation des éléments de modélisation. Par exemple, une passerelle ET-jointure, OR / XOR-jointure ou un événement qui reçoit plus qu'un arc entrant, etc. Les combinaisons valides ou non valides de ces éléments sont habituellement prescrites par la norme correspondante et leurs corrections nécessitent un délai raisonnable en utilisant des outils de modélisation tels que BizAgi[6], Intalio[7] ou Bonita[8], etc.

Un autre cas de ces erreurs est celui de la violation des contraintes et des règles de conformité qui assurent l'intégrité et la cohérence métier des processus et qui peuvent être spécifiées sur les données, les ressources ou sur le comportement de ces processus. Par exemple, avant l'ouverture d'un compte bancaire, les informations à propos du client doivent être vérifiées et validées. Le non-respect de cette contrainte dans la succession et l'acheminement des activités du processus peut conduire à une situation erronée ou invalide. Cette situation peut se produire aussi en définissant des règles qui soit redondantes, jamais applicables ou qui violent le domaine de leur définition (Chniti, Albert, & Charlet, 2011).

2.2.2 Les incohérences fonctionnelles

L'origine des erreurs qui peuvent causer des incohérences fonctionnelles est due à une mauvaise conception des modèles de processus métier par les concepteurs et les experts métier comme par exemple, (i) une situation de blocage *"Deadlock"* qui exprime une situation où l'exécution du processus est dans une impasse ou une attente infinie (Onoda, Ikkai, Kobayashi, & Komoda, 1999) ; (ii) une situation de boucles infinies *"Livelock"* qui exprime une situation où certaines activités du processus s'exécutent indéfiniment (Tantitharanukul & Jumpamule, 2010) ; (iii) une situation de terminaisons multiples *"Multiple Termination"* (Tantitharanukul, 2010), ou encore (iv) une situation où des activités continuent d'être exécutées après la terminaison du processus.

[6] http://www.bizagi.com/
[7] http://www.intalio.com/
[8] http://fr.bonitasoft.com/

Des erreurs sémantiques peuvent être aussi la cause d'incohérences fonctionnelles et qui n'ont aucune incidence sur l'exécution elle-même du processus, mais plutôt sur le résultat obtenu qui n'est pas celui attendu. Cela représente un dysfonctionnement dans la sémantique du processus métier. Par exemple, définir une activité non atteignable *"Dead-activity"*, dans ce cas, l'activité définie ne sera jamais exécutée (Sun, Huang, & Meng, 2011).

2.2.3 Les incohérences comportementales

Une incohérence comportementale se manifeste par un dysfonctionnement lors de l'exécution d'un processus métier. En effet, une erreur d'exécution appelée également une exception se caractérise par toute situation imprévue qui peut surgir durant l'exécution d'une instance d'un modèle de processus métier (Adams, ter Hofstede, Edmond, & van der Aalst, 2005), (Lerner, Christov, Wise, & Osterweil, 2008). Un conflit sémantique entre les différents schémas de processus métier et ses instances peut aussi être une cause d'incohérence comportementale (Rinderle, Reichert, & Dadam, 2003).

2.2.4 Les incohérences techniques

Une incohérence technique est la conséquence d'événements aléatoires qui sont susceptibles d'entraîner des bugs ou des exceptions pendant l'exécution d'un processus métier. En effet, ces événements sont rares et imprévus comme une panne matérielle du système informatique (une panne du serveur) qui peut déstabiliser le fonctionnement du processus métier ou encore une indisponibilité d'une ou de plusieurs ressources au moment où une activité en cours d'exécution veut y accéder et qui peuvent générer des exceptions.

2.3 Les techniques de vérification des modèles de processus métier

La vérification de modèles peut s'appliquer à tout système ayant une sémantique comportementale et une multitude de scénarii d'exécution possibles. Le principe de cette vérification consiste à s'assurer que le système s'exécute conformément à ce qui a été prévu initialement durant la phase de sa conception. Cela passe par l'extraction automatique des comportements inattendus ou erronés dans des délais raisonnables tout en garantissant qu'une propriété est vérifiée par toutes les exécutions possibles du système.

Comme vu précédemment, l'ensemble de ces incohérences des processus métier est dû à différent types d'erreurs qui peuvent souvent être repérées et gérées dans la phase de modélisation, mais aussi dans la phase d'exécution et la phase de monitoring. Dans ce contexte, les travaux de recherche qui existent dans la littérature consistent à fournir un ensemble de techniques et d'outils afin d'assurer une absence totale de toute erreur qui peut induire des incohérences dans les processus métier modélisés par des diagrammes EPC, des diagrammes d'activité UML, des diagrammes BPMN ou encore par un modèle BPEL.

Sadiq *et al.* (Sadiq & Orlowska, 1996), sont parmi les pionniers dans l'identification des erreurs structurelles, comme les impasses et les boucles infinies, dans les workflow. Leur travail vise principalement à identifier les erreurs syntaxiques avec un manque relatif pour gérer les erreurs sémantiques des workflow. Actuellement, la majorité des travaux

qui sont menés dans ce registre se concentre sur l'aspect sémantique des processus métier, par exemple « le processus doit toujours finir par se terminer ?» et d'autres questions similaires. Dans les sous-sections suivantes, nous détaillons les travaux les plus connus dans ce domaine.

2.3.1 La vérification par modèles formels

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. A cet égard, 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. Cette technique a pour but de vérifier certaines propriétés telles que l'absence de blocage, la propriété d'atteignabilité dans un processus métier afin de détecter les erreurs et cela, tout en tenant compte des propriétés du modèle et par conséquent, de sa notation.

En effet, le point de vue adopté par ces techniques de vérification formelle vise à offrir un cadre mathématique permettant de fournir d'une part une description formelle du système $\mathcal{M}$ et d'autre part, un modèle $\wp$ des comportements souhaités (ou de propriétés) de ce système et de s'atteler à répondre à la question suivante :

$$\mathcal{M} \mid = \wp \, ?$$

C'est à dire que l'on cherche à prouver que le modèle du système satisfait bien le modèle des propriétés.

De nombreux formalismes de spécification dédiés aux systèmes concurrents ont été proposés dans la littérature et en particulier pour vérifier les modèles de processus métier. Ce sont, notamment les réseaux de Petri (Rdp), les systèmes de transition, les automates communicants ou encore l'algèbre de processus CCS, CSP, LOTOS, etc. Une description de ces différents formalismes est détaillée dans ce qui suit.

2.3.1.1 Les automates

Les automates sont des modèles mathématiques, qui permettent de modéliser les spécifications formelles d'un grand nombre de systèmes (Hopcroft, Motwani, & Ullman, 2006). Ils sont composés d'un ensemble d'états du système, reliés entre eux par des transitions qui sont marquées par des symboles comme le montre la figure 2.1. Étant donné un mot fourni en entrée, l'automate lit les symboles du mot un par un et va d'état en état selon les transitions. Le mot lu est soit accepté par l'automate, soit rejeté.

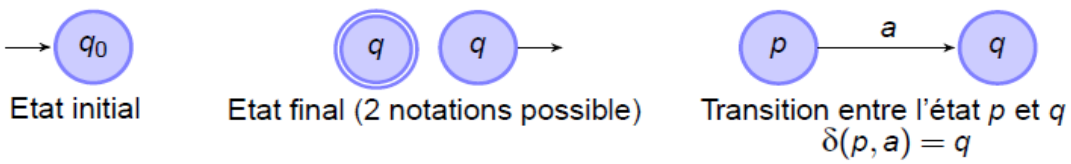


Figure 2.1 *Les éléments constituant un automate*

Formellement, un automate peut est défini par le tuple $A = (Q, \Sigma, \delta, q_0, F)$ où Σ représente l'alphabet ou l'ensemble de caractères (ou symboles), Q représente l'ensemble fini

d'états, $\delta : Q * \Sigma \rightarrow Q$ représente la fonction de transition, $q_0 \in Q$ représente l'état de départ et enfin $F \subseteq Q$ représente l'ensemble des états finaux (ou acceptant).

Fu *et al.* dans (Fu, Bultan, & Su, 2004), proposent une plate-forme pour analyser et vérifier les propriétés d'un modèle BPEL qui représente des interactions de services web composites communiquant avec des messages XML asynchrones. La plate-forme commence par convertir le processus en un type particulier d'automate dit gardé dont chaque transition est équipée d'un *"guard"* en format XPath[9] avec des files d'attente non bornées comme une représentation intermédiaire après quoi ces automates « gardés » sont convertis en Promela *"Protocol Meta Language*[10]*"* afin d'être vérifié par SPIN model-checker (Holzmann, 1997). SPIN consiste à vérifier si ces modèles satisfont ou pas des propriétés formulées en logique temporelle linéaire LTL *"Linear Temporal Logic"* (Rozier, 2011), (Hornus & Schnoebelen, 2002).

Wombacher *et al.* dans (Wombacher, Fankhauser, & Neuhold, 2004) présentent une traduction de la plupart des activités BPEL en automates à états finis déterministes. Les états de l'automate sont annotés avec des expressions booléennes. Ces expressions booléennes permettent de capturer comment un processus BPEL interagit avec son environnement.

Une vérification des services web exprimés en BPEL-WSCDL *"XML-based description languages"* par UPPAAL model-checker[11] est proposée dans (Diaz, Pardo, Cambronero, Valero, & Cuartero, 2005). Ces services web sont traduits automatiquement en une orchestration d'automates temporisés *"Timed Automata"* afin de simuler et analyser le comportement du système. Dans (Song Dong, Liu, Sun, & Zhang, 2006), les auteurs proposent une plate-forme pour vérifier automatiquement l'orchestration des services web capturés en Orc (Misra & Cook). Les auteurs utilisent un modèle d'automates temporisés pour les différentes expressions Orc afin de vérifier la sémantique opérationnelle de ces services web, cette vérification est faite par UPPAAL model-checker.

Koehler *et al.* (Koehler, Tirenni, & Kumaran, 2002), définissent une approche dirigée par les modèles pour la vérification de la propriété d'atteignabilité *"Reachability"*. Cette approche est fondée sur l'abstraction du processus métier en un automate non-déterministe comme un modèle intermédiaire. Ensuite, ils traduisent l'automate en code SMV, le langage d'entrée du model-checker NuSMV, afin de vérifier la propriété d'atteignabilité exprimée en CTL.

Tantitharanukul propose dans (Tantitharanukul, 2010) de vérifier l'absence d'impasses *"Deadlock"* et de terminaisons multiples *"Multiple Termination"* dans les modèles BPMN. L'approche consiste à transformer les modèles BPMN en automates à états finis et à vérifier la compatibilité des différentes transitions. Le modèle BPMN est considéré comme correcte s'il existe au moins une séquence valide qui est acceptée par les automates.

2.3.1.2 Les réseaux de Petri

Les réseaux de Petri (RdP) (Proth & Xie, 1995) sont apparus en 1962, dans la thèse de doctorat de Carl Adam Petri. Ils représentent un modèle mathématique qui se prête par-

[9] http://www.w3schools.com/xpath/
[10] http://spinroot.com/spin/Man/promela.html
[11] http://www.uppaal.com/

ticulièrement à la modélisation et l'analyse du comportement des systèmes dynamiques à événements discrets ainsi qu'au développement de systèmes distribués et concurrents, et cela d'une manière graphique. Mais avant toute chose ils permettent de simuler cette dynamique et surtout, par des techniques de vérification *"model-checking"*, de rechercher des propriétés telles que celles détaillées ci-dessous comme l'inter-blocage, la vivacité, etc.

Il existe une variété de réseaux de Petri tels que les réseaux de Petri hiérarchisés (Fehling, 1993), les réseaux de Petri colorés (Jensen, 1992), les réseaux de Petri temporels (Wang J. , 1998), les réseaux de Petri autonomes (David & Alla, 1993), les réseaux de Petri non autonomes (David & Alla, 2010), etc.

Un RdP est un graphe orienté biparti composé de places *P* qui représentent des conditions ou les états des ressources du système, de transitions *T* qui représentent des événements ou des actions qui se déroulent au sein du système et d'arcs qui représentent les conditions nécessaires pour déclencher une action. Autrement dit, les effets d'une action sur l'état du système sont modélisés par les arcs reliant les transitions aux places comme le montre la figure 2.2.

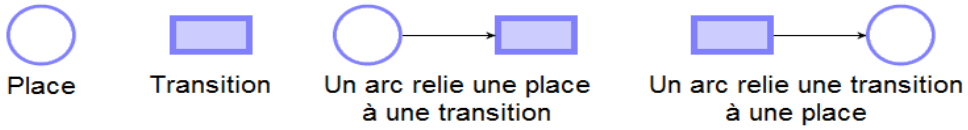


Figure 2.2 *Les éléments constituants un réseau de Petri*

Chaque place du réseau contient un nombre entier positif ou nul de « marques » ou appelé aussi « jetons » représentant le fait qu'une condition est vérifié ou une ressource est disponible. Le nombre de jetons disponibles dans chaque place du réseau qu'on appellera marquage *M* définit l'état du système décrit par le réseau à un instant *T*. Il se caractérise par un vecteur colonne de dimension égale au nombre de places dans le réseau. Le $i^{éme}$ élément du vecteur correspond au nombre de jetons contenus dans la place P_i. On dit qu'une transition *t* est franchissable lorsque toutes les places qui sont en amont contiennent au moins un jeton. Ce franchissement consiste à retirer un jeton de chacune des places en amont et à rajouter un jeton à chacune des places en aval de la transition *t*.

Un réseau de Petri est défini formellement par un triplé $R= (P, T, W)$ où $P = \{p_1, p_2,\ldots, p_n\}$ représente un ensemble fini non vide de places, $T = \{t_1, t_2,\ldots,t_n\}$ représente un ensemble fini non vide de transitions et $W : P \times T \cup P \times T \rightarrow N$ représente la fonction d'incidence correspondant aux arcs $W(p, t) / p \in P, t \in T$ qui contient la valeur entière associée à l'arc allant de *p* à *t* et $W(t, p) / p \in P, t \in T$ qui contient la valeur entière associée à l'arc allant de *t* à *p*.

Un RdP offre un large éventail de propriétés mathématiques qui permettent l'analyse du bon fonctionnement d'un système. Une classification de ces différentes propriétés concernant la vérification des systèmes est présentée ci-dessous :

2.3.1.2.1 Propriété d'absence de blocage "No deadlock"

L'absence de blocage ou *"deadlock"* est une propriété qui exprime le fait qu'un système ne se trouvera jamais dans une situation de blocage ou une attente infinie qui l'empêchera de progresser. En effet, la situation de blocage est une situation dans la-

quelle deux ou plusieurs processus sont incapables de procéder parce que chacun d'entre eux attend la progression de l'autre. Le problème du « dîner des philosophes » (Dijkstra, 1971) proposé par Edsger W. Dijkstra en 1965 est un cas d'école classique qui illustre bien la problématique d'inter blocage.

2.3.1.2.2 Propriété d'atteignabilité "Reachability"

Dans un système approprié, certains états indésirables ne doivent en aucun cas être atteints. Le résultat d'une telle vérification est soit des cas qui satisfont cette exigence ou non. La propriété d'atteignabilité qui joue un rôle crucial dans la vérification des propriétés d'accessibilité telle que l'absence d'inter blocage et l'exclusion mutuelle permet de dire qu'une certaine situation particulière ou un état peut être atteint ou non comme par exemple « On peut entrer dans une section critique ». Cette propriété peut être considérée comme la négation de la propriété de sûreté.

2.3.1.2.3 Propriété de sûreté "Safety"

Une propriété de sûreté permet de dire que, sous certaines conditions, quelque chose de non désirable ne se produit jamais. Autrement dit, cette propriété affirme qu'un système ne doit pas présenter de mauvais comportement, comme par exemple « deux systèmes ne seront jamais simultanément en section critique », « il n'est pas possible de parvenir à un état d'erreur » ou encore « un débordement de mémoire ne se produira jamais ».

2.3.1.2.4 Propriété d'équité "Fairness"

Une propriété d'équité exprime le fait que sous certaines conditions, quelque chose va se produire (ou pas se produire) infiniment souvent de fois comme par exemple « La porte sera ouverte infiniment souvent ».

2.3.1.2.5 Propriété de vivacité "Liveness"

Une propriété de vivacité correspond à vérifier qu'à tout instant le système conserve la possibilité de reproduire toutes les actions. Autrement dit, elle permet de dire que, sous certaines conditions, quelque chose de correct ou de bon finira par arriver, comme par exemple « toute demande sera satisfaite », « la lumière passe au vert » ou encore « après la pluie, le soleil ».

Les RdP sont largement utilisés pour décrire et vérifier les processus métier. En effet, plusieurs travaux de recherche ont exploité les points forts de ces réseaux afin de vérifier les erreurs que peut contenir un processus métier comme l'absence de blocage, la vivacité, etc. Le principe consiste à transformer les modèles de processus métier exprimés en BPMN ou BPEL en un réseau de Petri en utilisant des « traducteurs » puis analyser ce dernier par des outils appelés « analyseurs ». Cette démarche est résumée dans la figure 2.3.

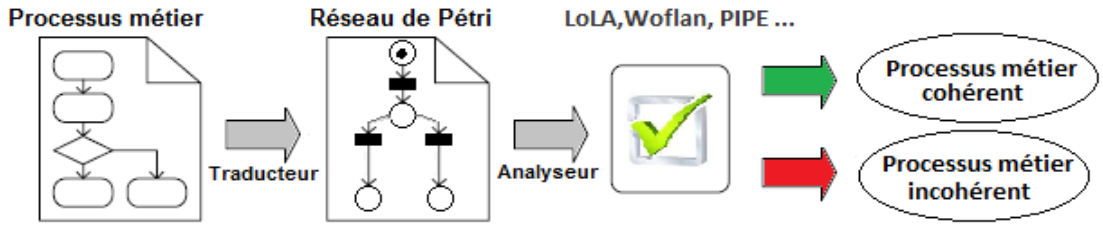


Figure 2.3 *La vérification d'un processus métier par les RdP*

En effet, ces travaux proposent une multitude d'approches et d'outils qui permettent de réécrire la spécification des processus métier en terme de RdP. Parmi ces outils nous pouvons citer BPEL2PNML (Hinz, Schmidt, & Stahl, 2005) ou encore BPEL2oWFN (Ouyang, Verbeek, van der Aalst, Breutel, Dumas, & ter Hofstede, 2005) qui permet de transformer automatiquement un modèle de processus métier exprimé par le langage BPEL en un réseau de Petri exprimé en PNML *"Petri Net Markup Language"*. L'outil BPEL2PNML génère un format qui est accepté par la majorité des analyseurs de RdP tels que LoLA (Schmidt, 2000), Woflan (Verbeek, Basten, & van der Aalst, 2001), PIPE (Bloom, Clark, Clifford, Duncan, Khan, & Papantoniou, 2003) tandis que le réseau de Petri généré par BPEL2oWFN est sous format oWFN, ce qui présente une limite car il n'y a que l'analyseur Fiona (Massuthe & Weinberg, 2008) qui accepte ce format. Dans (Yi & Kochut, 2004), les auteurs proposent un outil de design et de vérification de la composition des services web basé sur les réseaux de Petri colorés.

L'article (Hinz, Schmidt, & Stahl, 2005) propose une sémantique formelle pour BPEL basée sur les réseaux de Petri et cela afin d'assurer la gestion des exceptions et des compensations. En outre, les auteurs présentent un parseur qui convertit automatiquement les modèles BPEL en réseaux de Petri. Par conséquent, cette sémantique permet une analyse automatique des réseaux de Petri obtenus par de nombreux outils de vérification. Dans (Rosario, Benveniste, Haar, & Jard, 2006), les auteurs proposent une plate-forme qui permet de traduire l'orchestration des services web capturés en Orc (Misra & Cook) en réseaux de Petri colorés. Van der Aalst dans (van der Aalst W. , 1998), propose une plate-forme qui permet de vérifier les workflow en utilisant les réseaux de Petri.

Yang *et al.* (Yang, Tan, & Xiao, 2005) définissent une approche de vérification des propriétés de deadlock et livelock d'un service web composé. En effet, cette approche se base sur la transformation d'un service web composé spécifié avec BPEL en un réseau de Petri coloré CP-nets comme un modèle intermédiaire. Ensuite, ce dernier sera pris comme entrée par le model-checker CPN Tools pour vérifier les propriétés désirées. La propriété deadlock a été définie aussi dans le travail de Nakajima (Nakajima, 2006) dans lequel il propose une approche pour extraire la spécification comportementale d'un processus BPEL et de vérifier la propriété en utilisant SPIN model-checker. En effet, le passage du processus BPEL vers le code Promela est réalisé par un automate à états finis étendu comme une représentation intermédiaire. La propriété définie est formulée en logique LTL.

Takemura dans (Takemura, 2008), montre comment il est possible de vérifier les mécanismes de transaction et compensation des modèles BPMN en utilisant les réseaux de Petri et par conséquent, comment appliquer l'analyse de l'atteignabilité et la couvrabilité de ces réseaux à ces mécanismes de processus métier.

Des travaux comme ceux de Dijkman *et al.* (Dijkman, Dumas, & Ouyang, 2007), ou encore (De Backer & Snoeck, 2008), proposent de fournir une sémantique formelle aux modèles BPMN en transforment ces derniers en réseaux de Petri et cela tout en identifiant un certain nombre de composants de cette notation BPMN qui ne peuvent pas avoir une correspondance dans les réseaux de Petri comme les instances multiples, la gestion des exceptions pour les cas de sous-processus simultanés, etc. Dans la même optique, les auteurs dans (Ramadan, Elmongui, & Hassan, 2011) proposent d'utiliser les réseaux de Petri colorés pour fournir une sémantique formelle aux modèles BPMN au lieu des réseaux de Petri classiques pour plus d'expressivité.

YAWL ou *"Yet Another Workflow Language"* (van der Aalst & ter Hofstede, 2005) est un langage à la fois graphique et d'exécution destiné à représenter les processus métier en étendant la syntaxe des réseaux de Petri pour qu'elle supporte tous les patrons de flux de contrôle, tels que la modélisation de l'instance multiple, les tâches composites, le retrait des jetons et les transitions connectées directement (Decker, Dijkman, Dumas, & Luciano, 2008), (Ye, Sun, Wen, & Song, 2008). YAWL hérite des réseaux de Petri les mécanismes de vérification du bon fonctionnement des processus. Une nouvelle version de ce langage est récemment proposée dans (Russell & ter Hofstede, 2009) appelé newYAWL. Cette nouvelle version tente de couvrir, en plus des patrons de flux de contrôle, tous les patrons de données *"data patterns"* et les patrons de ressources *"ressource patterns"*.

D'autres travaux tels que ceux de (van der Aalst W. , 1999) et de (Langner, Schneider, & Wehler, 1998) , proposent une vérification de processus métier modélisés à l'aide du diagramme EPC. Cette approche consiste à transformer des diagrammes EPC en réseaux de Petri pour vérifier leur cohérence. L'idée de base de ces approches est de restreindre les classes du modèle EPC en sous-classes pour lesquelles il est plus facile de générer des réseaux de Petri.

L'importance de la vérification des flux de données dans les workflow a été abordée dans (Sadiq, Orlowska, & Sadiq, 2004). Les auteurs ont identifié plusieurs erreurs possibles dans le flux de données par exemple, l'erreur de redondance de données, la lecture d'un type de donnée non initialisé ou erroné, mais aucun moyen de vérifier ces erreurs n'a été fourni dans ce travail. Dans (Fan, Dou, & Chen, 2007), un modèle appelé réseau de Petri double *"DWF-nets"* est proposé, afin de vérifier à la fois le flux de données et le flux de contrôle.

En dépit des faiblesses soulevées précédemment, les RdP restent les modèles les plus utilisés pour décrire et vérifier les processus métier.

2.3.1.3 L'algèbre de processus

Les algèbres de processus sont un formalisme mathématique permettant de modéliser les systèmes concurrents ou distribués afin de vérifier automatiquement des propriétés associées à leur comportement. Cette approche se base principalement sur les techniques des algèbres universelles et les théories de la concurrence. A cet égard, de nombreux langages d'algèbres de processus ont vu le jour, les plus populaires entre eux sont :

- CSP *"Communicating sequential processes"* (Hoare, 1978) qui est une algèbre de processus permettant de modéliser l'interaction de systèmes. Il permet de fournir un mécanisme unifié pour la communication et la synchronisation des

systèmes, basé sur la notion de message et ne présupposant pas de l'existence d'une mémoire commune entre les processus.

- CCS *"Calculus of Communicating Systems*" (Milner, 1980), (Milner, 1989) qui est utile pour évaluer le bien-fondé qualitative des propriétés d'un système comme une impasse *"Deadlock"* ou boucles infinies *"Livelock"*.
- π-calcul (Milner, 1999) qui fournit une théorie solide pour la description des interactions entre plusieurs threads en concurrences.
- Lotos *"Language Of Temporal Ordering Specification"* le standard d''ISO (ISO, 1989) qui se base sur CCS et qui permet une spécification des systèmes parallèles.

Plusieurs travaux de recherche ont utilisé ces algèbres afin de modéliser et vérifier les processus métier. En effet, dans les travaux de (Salaün, Bordeaux, & Schaerf, 2004), les auteurs proposent d'utilisation l'algèbre de processus pour décrire, composer et vérifier les services web, avec un accent particulier sur leurs interactions. Ils montrent un exemple dans lequel ils utilisent CCS pour spécifier et composer un service-web. Ils utilisent également la simultanéité de Workbench[12] afin de valider les propriétés telles que la composition correcte des services web. Cette approche peut être utile, si le π- calcul est utilisé à la place du CSC afin de résoudre des problèmes tels que l'échange de messages au cours d'interactions des services web.

Dans les travaux de (Ferrara, 2004), une approche est proposée afin de mettre en correspondance BPEL/WSDL et le langage LOTOS. L'avantage de cette proposition est qu'elle prend en compte les compensations et le traitement des exceptions en vérifiant ainsi, des propriétés temporelles avec CADP model-checker (Fernandez, Garavel, Kerbrat, Mounier, Mateescu, & Sighireanu, 1996).

La principale contribution du travail de (Wong & Gibbons, 2008) est de fournir une sémantique aux différents éléments d'un modèle BPMN en termes d'algèbre de processus CSP, et cela en utilisant le langage ensembliste Z (Woodcock & Davies, 1996). Les auteurs Smith *et al.* (Smith & Fingar, 2003), proposent une plate-forme de modélisation graphique et de vérification d'un processus métier basée sur π-calcul tandis que (Puhlmann & Weske, 2005) utilisent le π-calcul afin de formaliser un ensemble de patterns de workflow. Dans la même optique, (Yang & Zhang, 2003) proposent une nouvelle approche pour la modélisation de workflow basée sur le π-calcul, qui permet de caractériser le comportement dynamique du workflow en termes de LTS *"Labeled Transition Semantics"*.

Les auteurs dans (Koshkina & Van Breugel, 2003) introduisent un langage, appelé BPEL-calcul inspiré du π-calcul, pour représenter les caractéristiques dynamiques du langage BPEL en utilisant une suite comportementale du système type. Ce système type a été codé en utilisant l'algèbre de processus sans tenir compte des données échangées dans les différents messages. Cette approche permet de vérifier non seulement la propriété de sûreté mais aussi la propriété de vivacité exprimée à l'aide de LTL.

Abouzaid et Mullins (Abouzaid & Mullins, 2009) proposent une approche de vérification des propriétés de services web composés plus exhaustive. L'approche proposée se base sur la transformation du processus BPEL en algèbre de processus BP-calcul pour véri-

[12] http://www.cs.sunysb.edu/~cwb/

fier par la suite les propriétés. En effet, ce travail permet de vérifier plusieurs types de propriétés spécifiées formellement avec la logique π-calcul en utilisant le model-checker HAL-Toolkit telles que la propriété de vivacité, équité, fiabilité, disponibilité, sûreté et la garantie de message *"Responsiveness"*.

Une traduction d'un modèle de processus métier exprimé en BPMN en COWS (Lapadula, Pugliese, & Tiezzi, 2007) est présentée dans (Prandi, Quaglia, & Zannone, 2008). Cet outil inspiré de π-calcul permet un raisonnement quantitatif sur le comportement des processus métier. Un tel raisonnement est montré à l'aide d'un cas d'étude qui utilise PRISM model checker (Hinton, Kwiatkowska, Norman, & Parker, 2006).

Toutefois, la vérification par π-calcul implique la vérification d'équivalence par bi-simulation. Ce qui implique une grande consommation de temps pour obtenir des résultats, et cela même pour prouver des exigences d'exactitude simples.

Ceci étant, d'autres approches, se basant sur des modèles différents, ont été proposées comme le travail de (Pu, Zhao, Wang, & Qiu, 2006) qui propose µ-BPEL, un nouveau langage basé sur BPEL et qui définit la sémantique des activités de base et structurées. µ-BPEL est traduit en automate temporisé TA *"Timed Automata"*, puis vérifié en utilisant l'outil UPPAAL (Behrmann, David, & Larsen, 2004) permettant ainsi la vérification des propriétés du réseau de TA.

La plate-forme VERBUS (Fisteus, Fernández, & Kloos, 2004) est une plate-forme modulaire extensible qui n'est pas liée à un outil de vérification ou à un langage de description de processus particulier. En effet, un modèle formel commun (CMF) est défini et chaque langage de spécification de processus peut être intégré en définissant sa sémantique en termes de CMF, et en implémentant un traducteur associé. Chaque outil de vérification peut être intégré en implémentant un traducteur du CMF en langage d'entrée de l'outil de vérification.

Malgré le temps que consomme l'approche de vérification par les modèles formels dont elle dépend de la complexité des algorithmes de traduction de la définition des processus métier en modèles formels et de la complexité des algorithmes de détection des propriétés à vérifier. Cette approche s'avère avantageuse en matière d'efficacité puisqu'elle augmente la certitude d'une absence de dysfonctionnements dans les processus métier.

2.3.2 La vérification par guide de style

Dans cette technique, des règles de style ou de bonne pratique sont définies afin de détecter les erreurs dans la phase de modélisation telle que « interdire l'association d'une ressource à deux activités métier en même temps ». Ces règles sont considérées comme des guides de bonne pratique de modélisation *"good modeling style"* qui sont utilisés afin de guider le concepteur à éviter les éléments qui peuvent conduire à des erreurs ou les éléments qui représentent une source potentielle d'erreurs.

Un exemple de ces règles est proposé dans le travail de Gruhn *et al.* (Gruhn & Laue, 2006), (Gruhn & Laue, 2007) pour la vérification des diagrammes EPC ou encore le travail de Van Dongen *et al.* (van Dongen & Jansen-Vullers, 2005) qui vérifie les modèles de références MySAP à travers l'AGL ARIS.

Toutefois, cette approche reste très limitée et permet de détecter peu d'erreurs, puisqu'elle tente de limiter les erreurs les plus connues, d'autant plus que cette technique ne détecte pas les erreurs produites lors de l'exécution du processus métier.

2.3.3 La vérification par conception

La vérification par conception ou " *Design approaches*" sont des méthodes de vérification basée sur des modèles de conception proposée dans un langage de requête graphique *ad hoc*. En effet, Awad *et al.* (Awad & Puhlmann, 2008), (Laue & Awad, 2011) présentent une approche pour détecter les inters blocages *"Deadlock"* en utilisant un langage graphique appelé BPMN-Q (Awad, 2007) et qui permet de formuler des requêtes sur le modèle de processus métier (sur sa structure graphique). Une autre approche (Di Francescomarino & Tonella, 2009) basée sur le langage de requêtes BPMN VQL (Di Francescomarino & Tonella, 2009) se focalise sur la structure graphique du modèle. Son objectif principal est de trouver les préoccupations qui transcendent BPM. Toutefois, les processus de modélisation utilisant cette notation nécessitent des compétences techniques avancées et le modèle qui en résulte est généralement complexe et loin d'être intuitive.

2.3.4 La vérification par simulation

La vérification par simulation consiste à comprendre, évaluer et comparer le processus modélisé en utilisant plusieurs scénarii possibles. La simulation fournit ainsi des estimations quantitatives sur l'impact qu'une conception de processus est susceptible d'avoir sur son exécution (Jansen-Vullers & Netjes, 2006). Plusieurs simulateurs de processus métier ont été proposés dans la littérature tels que ARIS (Scheer, 2010), (Scheer, 2000), Protos (Verbeek, van Hattem, Reijers, & de Munk, 2005), etc. Ces outils permettent aux entreprises d'analyser rapidement leurs processus métier.

Dans (Jansen-Vullers & Netjes, 2006), un certain nombre de ces outils sont comparés par rapport à leur applicabilité dans le domaine du BPM et aux modèles utilisés. Toutefois, la technique par simulation utilise des données théoriques et la qualité des estimations fournies dépend fortement de la batterie de tests utilisée.

2.3.5 La vérification par process-mining

Cette technique appelée vérification par *"Process mining"* consiste à explorer et analyser les traces d'exécution d'un processus métier ou appelé aussi journaux des événements *"Process Event Logs"* et cela, dans le but de mesurer les performances opérationnelles et métier de ce processus comme le montre le travail de (van der Aalst W. M., 2007). Le but principal est de reconstruire un modèle à partir des informations cumulées lors de la phase d'exécution *"Reverse Business Engineering"*. Le processus résultat sera comparé au processus modélisé pour savoir s'il s'agit du même déroulement de l'ensemble des activités. Autrement dit, permettre aux experts métier d'assurer que les processus exécutés correspondent aux exigences métier.

Toutefois, cette technique de vérification ne peut être que complémentaire aux autres techniques puisqu'elle ne peut être appliquée durant la phase de modélisation mais plutôt durant la phase d'exécution des processus métier.

2.4 Les techniques d'intégration & de gestion des changements des processus métier

Les processus métier sont en permanente évolution durant tout leur cycle de vie et cela afin de répondre à la nature changeante du métier de l'entreprise. Cette exigence de flexibilité des processus métier suscite depuis plus de vingt ans un intérêt croissant de l'industrie et de la communauté scientifique afin d'offrir un mécanisme permettant l'intégration et la gestion des changements des processus métier.

Le but principal étant de répondre aux questions qui peuvent subsister suite à un changement de processus métier comme par exemple : *(1) Que faire en cas de changement d'un schéma de processus métier ? (2) Comment passer d'une version à l'autre ? (3) Que faire des instances en cours d'exécution ? (4) Comment propager un changement du modèle à une instance ? (5) Comment garantir la cohérence structurelle et comportementale du processus métier ? (6) Que faire des instances modifiées individuellement "Concurrent changes" ?*

La conception d'un tel mécanisme de gestion des changements des processus métier requiert une compréhension détaillée des causes de changements appliqués sur les modèles de ces processus, de leurs types et de leurs niveaux d'application. Par conséquent, dans les sous-sections suivantes, nous commençons par détailler tout d'abord, la typologie des changements des processus métier, suivie de la typologie des opérations et des propriétés de ces changements et enfin les travaux de recherche qui existent dans le domaine de l'intégration et la gestion des changements des processus métier.

2.4.1 Typologie des changements des processus métier

Les changements affectant les processus métier peuvent être motivés par des raisons multiples. Ils peuvent, en effet, être évolutifs en réponse à l'évolution des besoins ou à l'introduction de nouvelles réglementations. Ils peuvent également être perfectifs et cela est particulièrement vrai dans le cadre de l'amélioration des critères de performance se manifestant par un besoin de gain de temps, d'optimisation des efforts, etc. En dernier, les changements peuvent être correctifs en réponse à la constatation d'une ou de plusieurs anomalies (erreurs ou exceptions) durant l'exécution d'un processus métier.

Ces changements peuvent s'appliquer sur deux niveaux :

- Le niveau du schéma du processus métier *"Process Type Level"*.
- Le niveau des instances du processus métier *"Process Instances Level"*.

Les changements appliqués au niveau des instances du processus métier sont également connus comme étant des changements spécifiques *"Instance-specific changes"*. Autrement dit, ils sont souvent appliqués de manière dite « *ad-hoc* » pour faire face à des situations imprévues ou à des exceptions résultant de l'exécution d'un processus métier (Klein & Dellarocas, 2000), (Nurcan, Etien, Kaabi, Zoukar, & Rolland, 2005), (Minor, Schmalen, Koldehoff, & Bergmann, 2007). Ils n'ont généralement pas d'incidence sur le reste des instances du processus en cours d'exécution et génèrent un impact local (Nurcan, 2008). Dans la figure 2.4.b, par exemple, l'instance I_4 peut être modifiée individuellement en supprimant l'activité *D* et en ajoutant l'activité *X*.

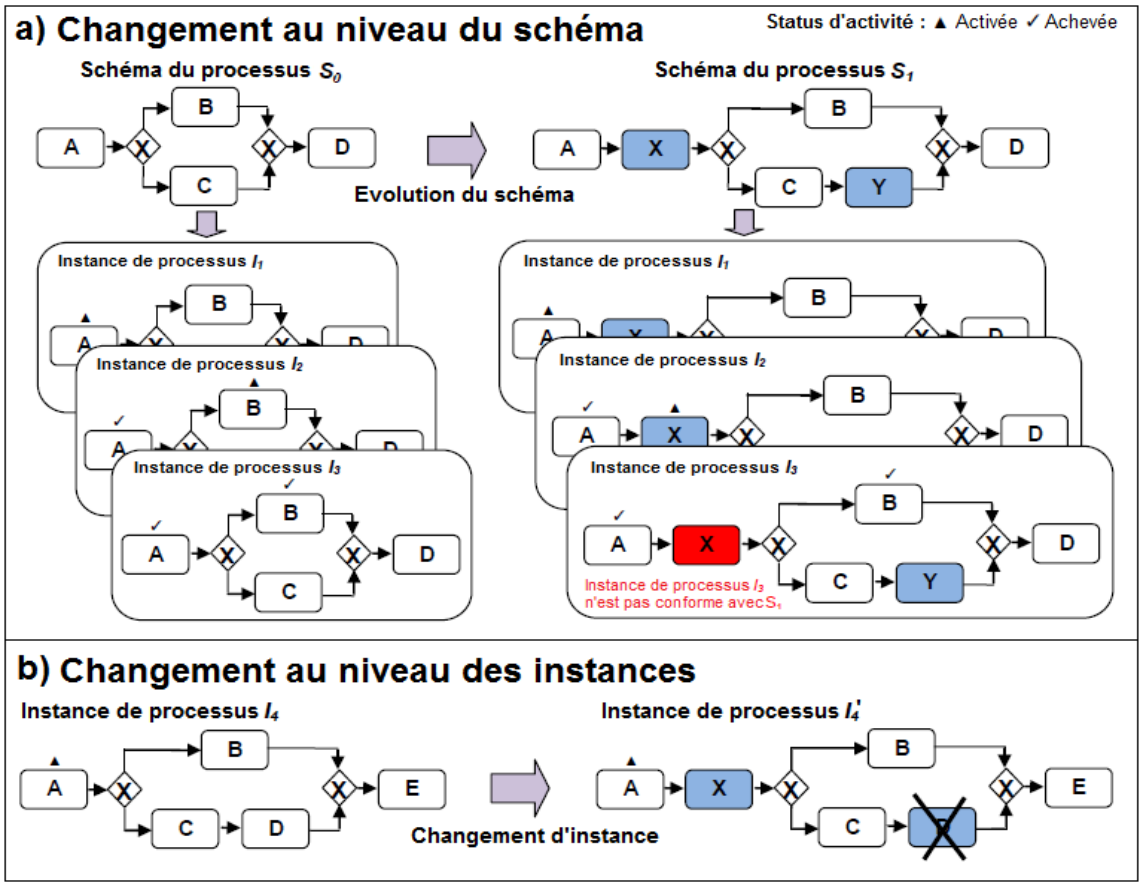


Figure 2.4 *Niveaux de changement de processus*

En général, changer une instance en cours d'exécution n'est pas suffisant pour résorber tous les problèmes rencontrés. C'est la structure du processus métier elle-même qui doit être remise en question (Kradolfer & Geppert, 1999), (van der Aalst W. , 2001), (Reichert, Dadam, & Bauer, 2003). À cet égard, les changements appliqués au niveau du schéma du processus métier sont nécessaires pour faire face à la nature changeante des rôles dans les processus métier induits généralement par l'adaptation aux nouvelles règlementations, etc.

L'évolution d'un schéma de processus métier nécessite souvent la propagation de ces changements sur le reste des éléments du schéma ainsi que sur ses instances en cours d'exécution. Autrement dit, ce type de changement produit un impact global (Nurcan, 2008). La figure 2.4.a, illustre un changement au niveau du schéma du processus métier. Le schéma S_1 est le fruit des changements apportés sur le schéma initial S_0 consistant à ajouter deux activités *X* et *Y*.

Ces changements ne peuvent être accomplis avec succès que si les instances en cours d'exécution sont conformes à la nouvelle version S_1 du schéma. Dans notre cas, les instances I_1 et I_2 (*cf.* figure 2.4.a) peuvent migrer vers le nouveau schéma S_1 contrairement à l'instance I_3 qui a déjà largement progressé dans son exécution. Cette instance doit donc être réalisée sur la base de la version initiale S_0 du schéma.

2.4.2 Typologie des opérations de changement

Nous pouvons distinguer deux types d'opérations de changement pour accomplir ce dernier au niveau du schéma du processus métier ou de ses instances :

- Les opérations dites basiques *"Basic operations"*.
- Les opérations dites complexes *"Complexe operations"*.

Les opérations basiques permettent d'insérer, de supprimer ou de modifier une activité, changer un flux de contrôle ou un flux de données ou encore changer l'affectation d'un rôle. Quelques opérations basiques sont représentées dans la Table 2.1.

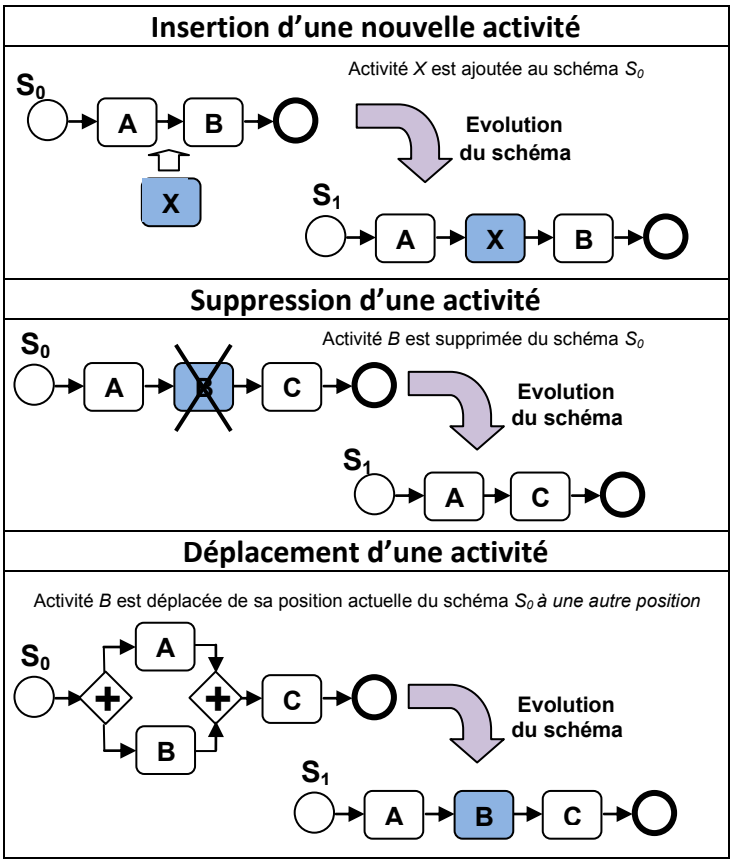


Table 2.1 *Opérations basiques de changement.*

Les opérations complexes permettent le remplacement d'un fragment de processus par un autre, le déplacement d'un fragment de processus d'une position *A* à une position *B*, la duplication d'un fragment de processus, l'échange d'un fragment de processus par un autre, la parallélisassions d'un fragment de processus ou encore une autre action complexe. Ces opérations peuvent être une combinaison des opérations basiques de changement comme l'ajout, la suppression ou la modification d'activités. Quelques exemples d'opérations complexes sont représentés dans la Table 2.2.

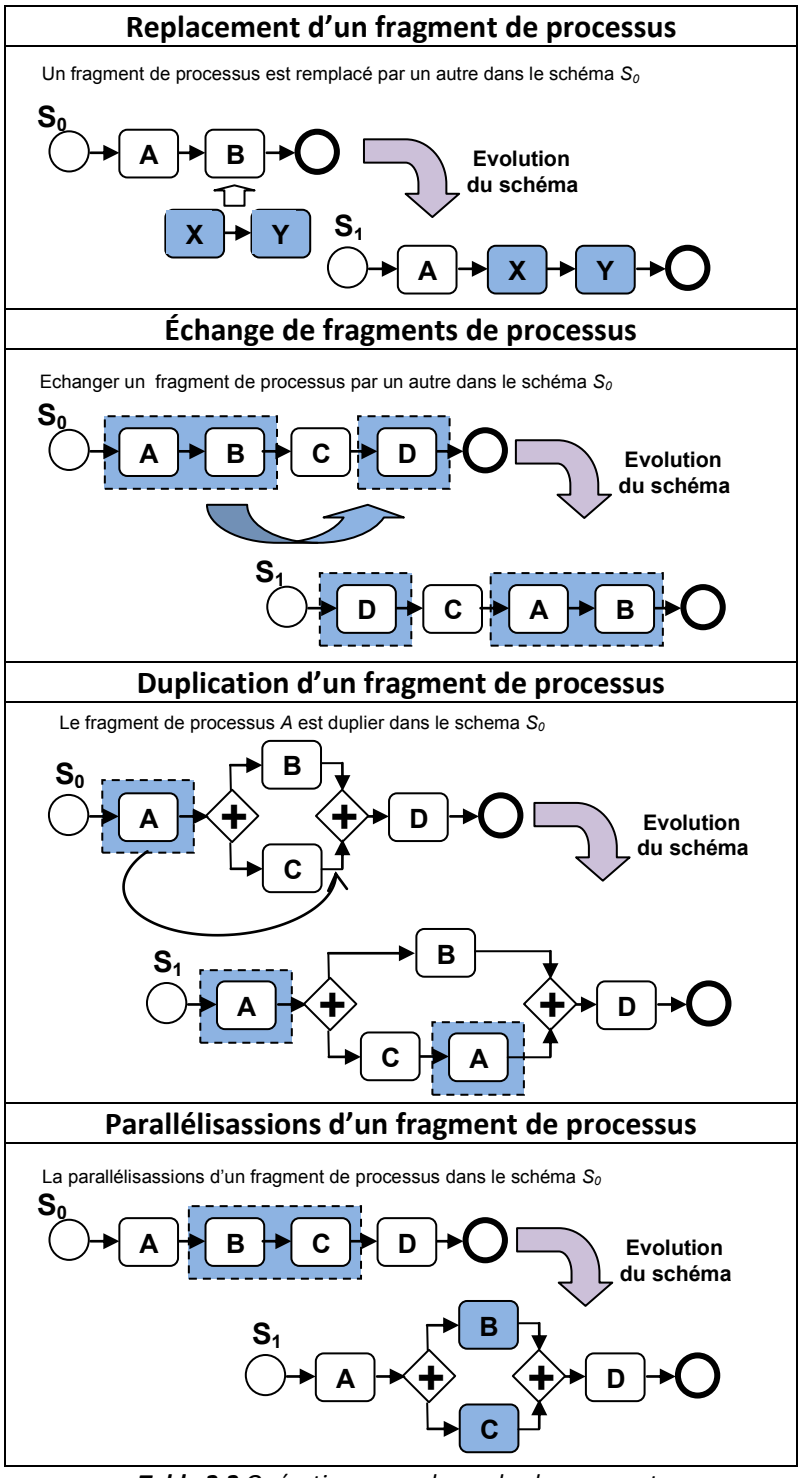


Table 2.2 *Opérations complexes de changement.*

D'autres patrons de changement sont détaillés dans le travail de (Weber, Reichert, & Rinderle-Ma, 2008).

2.4.3 Les propriétés de changement

Un changement de processus métier a, selon (Regev, Soffer, & Schmidt, 2006), quatre propriétés:

2.4.3.1 L'ampleur du changement *"Extent of change"*

Un changement peut être incrémental ou révolutionnaire. Un changement incrémental représente des modifications qui s'appliquent sur un schéma de processus métier existant. Tandis qu'un changement révolutionnaire ne prend pas en considération l'existant mais crée un nouveau schéma de processus. Ce qui est le cas en général, puisque les experts métier sont tenus de faire souvent des changements révolutionnaires.

2.4.3.2 La durée du changement *"Duration of change"*

Le changement peut être temporaire ou permanent. Un changement temporaire n'est valable que pour une période limitée, ce qui implique un retour au schéma de départ après la fin d'une échéance. Un changement permanent est valable jusqu'au prochain changement prévu.

2.4.3.3 La rapidité du changement *"Swiftness of change"*

Un changement doit être immédiatement pris en compte ou en différé, un changement immédiat répond généralement à une situation d'urgence, il est appliqué à toutes les instances d'un processus, et même à celles en cours d'exécution. Un changement différé n'est appliqué qu'au niveau des nouvelles instances du processus. Les instances en cours d'exécution restent inchangées, ce qui implique une coexistence de différentes versions d'un même schéma de processus.

2.4.3.4 L'anticipation du changement *"Anticipation of change"*

L'anticipation du changement représente le fait de savoir si un changement est planifié ou *ad-hoc*. Un changement *ad-hoc* est souvent appliqué pour faire face à des situations exceptionnelles. Tandis qu'un changement planifié ou prévu fait partie souvent d'une refonte totale ou partielle d'un processus. Ces changements peuvent être liés à différents niveaux d'abstraction et peuvent être soit temporaires ou permanentes.

Il existe une variété d'approches et de techniques permettant d'intégrer et de gérer des changements des processus métier et cela même durant l'exécution de leurs instances, dans le reste de cette section, nous faisons un tour d'horizon des travaux existant dans la littérature.

2.4.4 Techniques d'évolution

Ces techniques permettent de définir la façon dont la définition du processus métier ou ses instances vont évoluer. Il existe plusieurs approches permettant cette évolution dont on distinguera l'approche *ad-hoc*, l'approche par dérivation, l'approche par héritage, l'approche par induction, l'approche par réflexion et l'approche à base de règles.

2.4.4.1 L'approche ad-hoc

Cette approche concerne les changements non prévus pouvant découler de la nécessité de modifier le comportement d'un processus métier durant son exécution (Agostini & De Michelis, 2000), (Rinderle, Reichert, & Dadam, 2003) , (Dellen, Maurer, & Pews, 1997). Cela sous-entend donc que ces changements affectent les instances d'un processus métier.

Cake2 (Minor, Schmalen, Koldehoff, & Bergmann, 2007), WASA2 (Weske, 2000), ADEPT2 (Reichert, Rinderle, Kreher, & Dadam, 2005) sont des outils permettant de prendre en charge ce type de changement *ad-hoc* comme par exemple l'ajout, la suppression ou encore le déplacement d'une activité ou d'un fragment d'activités au niveau des instances d'un processus métier.

2.4.4.2 L'approche par dérivation

Dans cette approche, le schéma de processus cible du changement est connu, mais le problème qui se pose est de migrer les instances de l'ancien schéma vers le nouveau schéma. Cela nécessite souvent un modèle de transition appelé également modèle *"bridge".*

Des travaux comme ceux de (Kradolfer & Geppert, 1999), (Sadiq & Orlowska, 1999), (Weske, 2001) offrent quelques solutions à cette problématique en se basant sur l'approche dite de dérivation.

2.4.4.3 L'approche par héritage

Cette approche exploite le concept d'héritage connu dans la modélisation orientée objet afin de l'étendre à la définition des modèles de processus métier génériques.

Chiu *et al*. (Chiu, Li, & Karlapalem, 2001), proposent une méta-modélisation orientée objet étendue afin de prendre en compte la flexibilité et l'adaptabilité des workflow ainsi que la gestion des exceptions. Les auteurs (Dąbrowski, Drabik, Trzaska, & Subieta, 2011) présentent l'idée d'un système orienté objet de gestion de workflow déclaratifs. Un prototype est mis en œuvre sur la base de ODRA[13], qui est un SGBD distribué orienté objet, et SBQL[14] qui est un langage de requêtes conçu et mis en œuvre pour ODRA.

Rajabi *et al.* (Rajabi & Lee, 2010), proposent d'utiliser un réseau de Petri colorié orienté objet *"OOCPN"* pour la modélisation et l'analyse du changement des processus métier. Ils visent à intégrer des processus métier modélisés en utilisant des diagrammes d'activités UML et les réseaux de Petri coloriés et cela afin de prendre en considération les changements dynamiques du processus pendant leurs exécutions.

Des travaux similaires qui utilisent la même approche orientée objet sont proposés par (van der Aalst W. , 2001), (van der Aalst, Basten, Verbeek, Verkoulen, & Voorhoeve, 1999).

2.4.4.4 L'approche par induction

Le principe de cette approche consiste à extraire le schéma du processus métier par le *"process mining"* à partir des traces d'exécutions *"traces logs"* de ses instances (Herbst, 1999). Le but étant d'adapter ce dernier à l'exécution précédente de ses instances, et cela à l'aide d'un mécanisme d'apprentissage inductif. Autrement dit, il s'agit d'intégrer les adaptations qui ont été apportées aux instances, suite par exemple à l'occurrence de cas exceptionnels, comme des modifications permanentes au niveau du schéma du processus.

[13] http://code.google.com/p/odra3/
[14] http://www.sbql.pl/

Des outils tels que ProM (van Dongen, de Medeiros, Verbeek, Weijters, & van der Aalst, 2005), EMiT (van der Aalst & Van Dongen, 2002), Little Thumb (Weijters & van der Aalst, 2003), et MinSon (van der Aalst & Song, 2004) permettent une extraction des connaissances à partir des journaux d'exécutions des instances d'un processus métier *"Process Event Logs"*.

2.4.4.5 L'approche par réflexion

Cette approche propose de concevoir des schémas de processus qui ont la capacité de contrôler et de modifier leur propre comportement, automatiquement ou par l'intervention de l'utilisateur. Cela est réalisé sur la base des *"feed-backs"* sur les instances comme décrit dans les travaux de (Borghoff, Bottoni, Mussio, & Pareschi, 1997).

2.4.4.6 L'approche à base de règles

Cette approche propose de modéliser la logique du processus par un ensemble de règles en utilisant des langages déclaratifs et cela afin d'offrir des processus métier flexibles. L'exécution de ces langages est souvent assurée par des moteurs d'inférence de règles qui déterminent l'ordre d'exécution des activités en fonction de l'évaluation des conditions. Cela permet une évolution des instances du processus métier basée sur des événements survenant au cours de leurs exécutions (Klein & Dellarocas, 2000), (Sadiq & Orlowska, 1999).

Selon plusieurs travaux, (Knolmayer, Endl, & Pfahrer, 2000) et (Bry, Eckert, Pătrânjan, & Romanenko, 2006), les règles de réaction ECA *"Event-Condition-Action"* sont les mieux adaptées pour décrire la logique du processus par un ensemble de règles. Giurca *et al.* dans (Giurca, Lukichev, & Wagner, 2006), justifient cela par le fait que le formalisme ECA, sur lequel se basent les règles de réaction, permet de spécifier le flux de contrôle d'un processus d'une manière flexible en utilisant les événements.

Une plate-forme appelée AgentWork est proposée par Müller dans (Müller, Greiner, & Rahm, 2004) où un ensemble de règles ECA qui régissent le fonctionnement des différents agents permet la gestion temporelle de Workflow. Le travail de Zeng *et al.* dans (Zeng, Ngu, Benatallah, & O'Dell, 2001) proposent par ailleurs de voir le processus comme un ensemble de tâches coordonnées entre elles par des règles ECA et d'utiliser les agents pour encapsuler les services qui exécutent les tâches du processus. D'autres travaux utilisent d'autres formalises de règles basés sur des contraintes comme dans (Rinderle, Reichert, & Dadam, 2004) où les auteurs proposent une approche formelle basée sur la notion de contraintes de processus appelée *"Constraint-Based Flexible business process management"* permettant de démontrer comment la spécification de sélection et l'ordonnancement des contraintes peut conduire à une plus grande flexibilité dans l'exécution des processus, et cela tout en maintenant un niveau élevé de maîtrise.

DECLARE (Pesic, Schonenberg, & van der Aalst, 2007) est un prototype de WFMS qui utilise un langage déclaratif de modélisation des processus à base de contraintes formulées en logique temporelle pour le développement et l'exécution des modèles de processus métier.

PLM_{flow} (Zeng, Flaxer, Chang, & Jeng, 2002) est un framework qui fournit un ensemble de règles d'inférence conçu pour générer et exécuter dynamiquement des workflow. La définition du processus est spécifiée dans la règle de gestion, qui comprend deux types de

règles, les règles en chaînage arrière *"backward-chain rules"* et les règles en chaînage avant *"forward-chain rules"*. L'exécution des instances de processus est guidée par un moteur de règles non déterministe qui utilise une combinaison de ces règles.

2.4.5 Techniques de migration

La technique de migration permet de définir ce qui se passe avec les instances en cours d'exécution et dont la définition du processus à été modifiée. Il existe plusieurs approches permettant cette migration parmi elles on distinguera l'approche par annulation, l'approche avec propagation et l'approche sans propagation.

2.4.5.1 L'approche par annulation

Les instances en cours d'exécution affectées par le changement sont annulées et de nouvelles instances sont créées en fonction de la nouvelle définition de processus. Cette technique est la moins recommandée en raison de la perte d'information et de temps (Kradolfer & Geppert, 1999).

2.4.5.2 L'approche sans propagation

Les instances en cours continuent leurs exécutions conformément à l'ancienne définition de processus métier tandis que les nouvelles instances sont exécutées selon la nouvelle définition ou version du processus métier. Ceci implique une coexistence de différentes versions d'un même schéma de processus métier.

En effet, la propagation de la modification des instances peut être différée. Autrement dit, les instances sont exécutées conformément à l'ancienne définition de processus jusqu'à ce qu'ils atteignent un état sûr.

Dans (Weske, 1998), l'auteur propose une modélisation et une exécution flexible des différentes activités du workflow basée sur un méta-modèle métier. Cette approche prend en charge les changements dynamiques telles que l'ajout ou la suppression d'activités, mais exige que l'activité ne soit pas en état de fonctionnement durant l'intégration du changement.

Cette stratégie est également utilisée aussi dans l'outil Milano proposé dans le travail de (Agostini & De Michelis, 2000) ou encore l'outil Wasa2 proposé dans le travail de (Weske, 2001).

2.4.5.3 L'approche avec propagation

Le changement au niveau du schéma de processus métier est propagé aux instances en cours d'exécution. Il s'agit de la technique la plus complexe car elle nécessite un modèle de transition permettant la conformité des instances actuelles avec la définition du processus cible (Rinderle, Reichert, & Dadam, 2003), (Kradolfer & Geppert, 1999).

Casati *et al.* (Casati, Ceri, Pernici, & Pozzi, 1996), présentent un langage de gestion du changement des workflow appelé *"WFML"* pour répondre à la question de la migration des instances en cours d'exécution quand la définition du processus métier évolue. Ils présentent un ensemble de critères formels permettant de déterminer l'ensemble des instances qui peuvent migrer en toute sécurité vers la nouvelle version du schéma de processus métier. De la même manière, (Zhao & Liu, 2013) proposent un mécanisme de

gestion de versions pour les différents schémas du processus métier en représentant les différents schémas de processus métier et les dépendances.

$ADEPT_{flex}$ (Reichert & Dadam, 1998), (Rinderle, Reichert, & Dadam, 2004) est un modèle de workflow graphique connu pour son intégration des changements d'une façon dynamique et fluide, et cela même durant l'exécution de ses instances sans perdre le contrôle et la cohérence structurelle de ces instances. Il propose une politique migratoire qui est liée à la gestion des conflits potentiels qui peuvent être causés par ces changements.

2.4.6 Techniques de flexibilité

Cette technique est appliquée au cours de la modélisation des processus métier afin de mettre en œuvre des définitions de processus métier flexibles qui pourraient être affinées lors de la phase d'exécution. Il existe plusieurs approches permettant cette flexibilité. Parmi elles, on notera l'approche par sélection tardive et l'approche par modélisation tardive.

2.4.6.1 L'approche par sélection tardive *"Late binding"*

Dans cette approche, les différents éléments du processus métier sont considérés comme des objets dont le comportement est défini lors de la phase d'exécution, ce qui implique qu'aucun changement n'est nécessaire sur le schéma du processus métier, mais un large choix d'alternatives d'exécution est disponible durant cette phase (Dellen, Maurer, & Pews, 1997). Ceci inclut la capacité de sélection de la meilleure ressource ou de l'acteur le plus approprié pour réaliser une activité qui doit satisfaire un critère donné.

Un exemple de cette approche est présenté dans le travail de (Belhajjame, Vargas-Solar, & Collet, 2001) qui consiste à fournir un outil appelé "*AFLOWS*" qui offre la possibilité d'associer la même définition de processus à un comportement différent des processus en fonction du contexte d'exécution.

2.4.6.2 L'approche par modélisation tardive *"Late modeling"*

Dans cette approche, certaines parties du processus métier sont laissées ouvertes à l'innovation et à la créativité des utilisateurs, en particulier lorsque certaines spécifications ne peuvent pas être identifiées par avance dans la phase de modélisation. Autrement dit, certaines activités peuvent être déclarées comme critiques et obligatoires, tandis que d'autres pourraient être optionnelles et omissent que durent la phase d'exécution.

Dans (Klingemann, 2000) plusieurs comportements peuvent être associés à des activités où l'utilisateur peut sélectionner le comportement le plus approprié d'une manière dynamique. Dans la même optique, les travaux de (Kradolfer & Geppert, 1999), offrent à l'utilisateur la possibilité de sélectionner les performances appropriées pour une activité qui a été initialement définie comme un objectif à atteindre. Si aucune méthode existante n'est appropriée, il est également possible d'en créer une nouvelle dynamiquement.

2.5 Conclusion

Dans ce chapitre, nous avons vu que la vérification des processus métier a pour but de montrer que les activités du processus s'exécutent en conformité à son plan de réalisation et qu'elles n'ont pas introduit d'erreurs. Pour cela, plusieurs techniques sont proposées dans la littérature. Généralement, la technique de vérification par réseaux de Petri est la plus utilisée et cela est dû à la nature de ce formalisme qui est à la fois, formel et graphique permettant ainsi une représentation graphique avec l'aspect sémantique attribué au comportement du processus modélisé.

Nous avons aussi mis en exergue l'importance de l'intégration et la gestion des changements des processus métier en présentant les différentes techniques et approches qui existent dans la littérature pour faire face à cette problématique. Au préalable de ce tour d'horizon, nous avons d'abord présenté une typologie des changements des processus métier, suivie de la typologie des opérations et des propriétés de ces changements.

Dans le chapitre suivant, nous allons présenter le premier axe de notre contribution qui consiste à vérifier la cohérence des processus métier en utilisant le Model-checking. Cette même technique est utilisée pour vérifier les règles dites de conformités.

CHAPITRE 3

La Vérification des modèles de processus métier par le Model Checking

3.1 Introduction

Les réseaux de Petri demeurent parmi les formalismes les plus utilisés pour la vérification formelle des processus métier. Ce formalisme occupe, en effet, une place prépondérante en dépit de ses limites (*cf.* chapitre 2), et se trouve être l'outil le plus largement étudié si l'on considère la diversité des techniques automatiques de vérification qui lui sont associées. Nous pouvons distinguer trois techniques principales permettent la preuve formelle des bonnes propriétés d'un modèle de système. Ce sont la vérification basée sur les preuves de théorèmes, la vérification basée sur les équivalences, la vérification basée sur le model-checking.

La première approche, basée sur les preuves de théorèmes, a pour principe de poser un ensemble d'axiomes qui servent à prouver un ensemble d'assertions ou de propriétés déterminant ainsi la conformité du système. Cette technique n'est que peu automatisée et nécessite souvent une intervention humaine. Cependant, des outils d'aide à la preuve existent mais demandent une forte expertise humaine et une disponibilité des ressources machine puisque ces algorithmes induisent une forte consommation de ressources de calcul. Toutefois, ce type de vérification est rarement employé, il est utilisé essentiellement dans le domaine des spécifications algébriques et logiques (Pnueli, 1979).

La deuxième approche consiste à trouver une équivalence entre le modèle de description d'une implémentation d'un système et une spécification de ce système. Il s'agit donc d'identifier une relation de bi-similarité entre ces deux modélisations qui sont décrites par un même formalisme (Milner, 1999). L'équivalence des deux modèles prouve que l'implémentation est conforme à la spécification. Cette technique a un défaut, qui peut se résumer dans la consommation conséquente de temps pour obtenir des résultats, et cela, même pour prouver des exigences d'exactitude simples.

La troisième approche qui représente une des plus puissantes méthodes formelles en termes d'efficacité et d'automaticité est le "model-checking" appelée aussi la vérification sur modèle. Cette méthode représente une technique de vérification automatique des systèmes dynamiques, permettant de déterminer ainsi si un modèle donné d'un système satisfait une spécification. Les modèles de système utilisés par les model-checkers sont généralement des modèles de type états/transitions, tandis que les modèles des propriétés sont généralement des formules de logique temporelle.

En effet, le principe de cette vérification sur modèle consiste à générer tous les états possibles d'un système dans lesquels la propriété à vérifier est testée de façon exhaustive sur l'ensemble des exécutions possibles du système. Par exemple, pour démontrer l'absence d'erreurs à l'exécution, on pourra tester l'absence d'états d'erreur dans l'ensemble des états accessibles du système. Il s'agit alors d'une forme de test exhaustif, mais mené à l'aide d'algorithmes astucieux permettant d'énumérer tous les états du système sous une forme symbolique. En revanche, l'inconvénient majeur de ce type de vérification est la taille souvent excessive de l'espace d'états (le système de transition) énumérés pour vérifier la validité d'une propriété. En effet, elle peut être exponentielle par rapport à la taille de la description du système ce qui conduit rapidement à un dépassement des capacités de l'ordinateur en matière de mémoire. Une des causes principales de cette explosion combinatoire du nombre d'états est le fait que l'exploration est réalisée en prenant en compte tous les entrelacements possibles d'évènements concurrents.

Par conséquent, et en dépit de ces inconvénients qui sont en partie résolus par les travaux de (Holzmann, 1997), (Burch, Clarke, Mcmillan, Dill, & Hwang, 1990) la vérification formelle basée sur le model-checking reste le meilleur choix à faire puisque la vérification est complètement automatique et un contre-exemple est fourni au cas où la propriété serait violée.

Fort de ce constat, un ensemble de travaux de recherche ont essayé de tirer profit de l'utilisation du model-checking afin de vérifier la validation des processus BPEL, des processus workflow et des processus métier. En effet, dans (Fisteus, Fernández, & Kloos, 2004), (Nakajima, 2006), les auteurs proposent une approche permettant la vérification et la validation d'une orchestration de services web spécifiée avec BPEL pour s'assurer de sa consistance et d'éviter les changements inattendus. Cette approche permet de vérifier un ensemble de propriétés génériques, telles que la vivacité et la sûreté, et spécifiques en utilisant le model-checker SPIN avec son langage d'entrée PROMELA.

Dans (Feja, Speck, & Pulvermüller, 2009), (Speck, Feja, Witt, & Pulvermüller, 2011) les auteurs proposent l'utilisation du model-checking pour vérifier un ensemble de règles dans les processus métier modélisés par les diagrammes EPC. En effet, dans cette vérification l'ensemble des règles est exprimé en CTL (Wolfgang, 1989) à travers une notation graphique appelée G-CTL *"Graphical Computationnal Temporal Logic"*, tandis que, les diagrammes EPC à vérifier sont traduits en structure de kripke afin que les deux soient utilisées comme entrée pour le model-checker.

Dans un autre registre, Vaz et Ferreira (Vaz & Ferreira, 2007), (Vaz & Ferreira, 2007) proposent de vérifier l'exactitude des processus workflow et le comportement des processus métier basés sur les services web par le model-checker SPIN. Dans ces travaux, les auteurs proposent de traduire une collection de patterns de workflow en code Promela. Cette traduction est illustrée par deux processus métier basés sur les services web.

Dans la continuité de ces travaux, l'approche proposée dans ce chapitre consiste à utiliser la technique du model-checking pour vérifier les propriétés de cohérence des processus métier modélisés en BPMN. En effet, notre approche permet tout d'abord de traduire un modèle BPMN en modèle Promela. Cette traduction se fait en deux phases. La première phase, consiste à transformer le modèle BPMN vers une structure de kripke comme représentation intermédiaire et cela, afin de donner une sémantique formelle à ces modèles, et par conséquent l'ensemble des exécutions de ce système. La deuxième phase consiste à traduire cette structure de kripke vers un code Promela.

En second lieu, nous exprimons en Logique Temporelle Linéaire (LTL) les différentes propriétés de cohérence des modèles de processus métier et nous utilisons le model-checker SPIN pour tester si chaque propriété de cohérence est satisfaite par le modèle Promela du modèle BPMN en question.

Dans le cas d'une réponse négative, SPIN fournit un contre-exemple montrant une exécution qui ne satisfait pas ces propriétés. De la même manière, cette approche est utilisée afin de vérifier la conformité des processus métier avec l'ensemble des règles de conformité suite à un changement affectant l'un des deux.

La suite de ce chapitre est organisée comme suit : nous commençons par présenter dans la section 3.2 les notions fondamentales du model-checking ainsi que le principe de son fonctionnement. Dans la section 3.3, nous présentons le model-checker SPIN qui est

l'outil utilisé pour implémenter et valider notre approche. Les erreurs structurelles et le détail de notre approche sont relatés dans la section 3.4 et 3.5. Ensuite, nous détaillons dans la section 3.6 la description des processus métier à vérifier en langage Promela. Enfin, dans la section 3.7, nous étendons cette approche à la vérification des règles de conformité mises en place suite à un changement du modèle de processus métier.

3.2 Le model-checking

3.2.1 Principe du model-checking

Le model-checking (Clarke Jr., Grumberg, & Peled, 1999), (Biere, 2008) ou la vérification de modèle est reconnu comme étant l'une des méthodes formelles les plus puissantes et la plus utilisée. En effet, le model-checking est une méthode de vérification exhaustive basée sur des modèles mathématiques permettant l'analyse des systèmes réactifs qui présentent un comportement aléatoire ou probabiliste. Cette vérification utilise un algorithme qui explore exhaustivement l'ensemble des exécutions possibles du système. Le nombre des états générés par ce modèle peut être exponentiel, les outils de model-checking actuels peuvent énumérer explicitement des espaces d'états contenant 10^{20} états (Burch, Clarke, Mcmillan, Dill, & Hwang, 1990).

Le principe du model-checking consiste à générer toutes les exécutions possibles d'un système et de vérifier ses caractéristiques comportementales, en s'assurant que les propriétés spécifiées telles que l'absence de blocage "*deadlock*" sont vérifiées dans chaque exécution et donc dans chaque état du modèle.

Pour ce faire, le model-checker prend en entrée une abstraction du comportement du système réactif ou système de transition et une formule d'une logique temporelle et vérifie si l'abstraction satisfait la formule (*cf.* figure 3.1). L'abstraction du comportement du système peut être exprimée en utilisant des structures de Kripke (Browne, Clarke, & Grümberg, 1988) ou d'autres modèles tels que les réseaux de Petri, les automates finis, les automates temporisés, etc. Les formules de logique temporelle peuvent être exprimées en utilisant LTL (Rozier, 2011) , en PLTL (Laroussinie & Schnoebelen, 1995), en CTL (Wolfgang, 1989), en CTL* (Schneider & Prof.Dr Schmid, 1997), ou TCTL (Bel Mokadem, Bérard, Bouyer, & Laroussinie, 2006), etc. Le système de transition est dit modèle de la formule, d'où le terme *vérification de modèle*.

A l'issue de cette vérification, si une propriété n'est pas satisfaite, un contre-exemple est fourni permettant ainsi d'identifier un cas d'exécution non conforme et ainsi localiser et corriger la source d'erreur. Ce contre-exemple peut être obtenu dès l'instant où la propriété à vérifier n'est pas satisfaite durant l'exécution, ce qui permet une vérification plus rapide qu'avec les techniques traditionnelles et cela en plus du fait que celle-ci soit automatique.

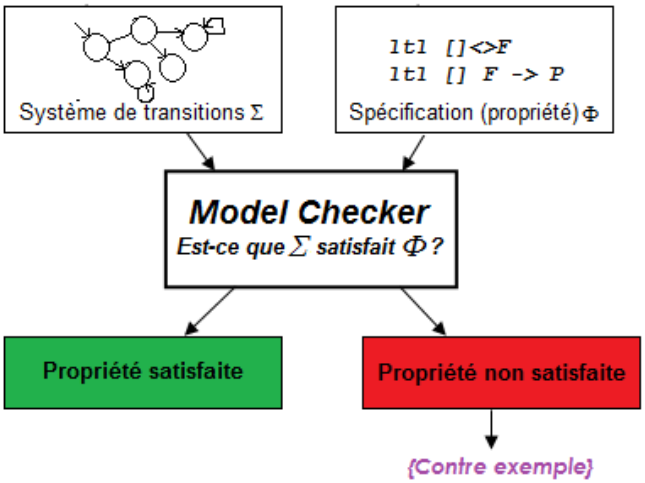


Figure 3.1 *Principe du model-checking.*

Il existe plusieurs catégories de propriétés qu'un model-checking permet de vérifier. Ce sont notamment:

- Les propriétés d'accessibilité comme par exemple « il existe un état accessible où x vaut 0 ».
- Les propriétés d'invariance ou de sûreté comme par exemple « durant toute l'exécution, x est différent de 0 ».
- Les propriétés de vivacité comme par exemple « l'application se terminera finalement ».

Ces propriétés sont détaillées ci-après. Enfin, il existe également plusieurs model-checkers puissants tels que SPIN (Holzmann, 1997), (Holzmann, 2003), NuSMV (Cimatti A. , Clarke, Giunchiglia, & Roveri, 2000), (Cimatti A. , et al., 2002) ou BLAST (Henzinger, Jhala, Majumdar, & Sutre, 2003).

3.2.2 Système de transitions

Le model-checking étudie des systèmes réactifs abstraits exprimés syntaxiquement sous la forme de machines à états. La sémantique d'une machine à états est donnée par un système de transitions. Ces machines à états peuvent être plus ou moins complexes, allant des machines à états finis (automates finis) à de vrais programmes (machines de Turing). Cependant, plus le formalisme d'entrée est puissant, moins on peut décider de leurs propriétés de façon automatique.

En effet, pour vérifier toutes les exécutions possibles d'un système lambda, on doit représenter son modèle sous la forme d'un système dit « système de transitions à états », où chaque état correspond à un état du système étudié, à chaque pas d'exécution et à chaque transition correspondant à une évolution possible d'un état donné vers un autre, selon l'action exécutée par le système. Ce système de transition ou automate au sens large, est un modèle de machine abstraite qui est utilisée en théorie de la calculabilité et dans l'étude des langages formels. Il peut être représenté par un graphe orienté, une machine de Turing ou encore un réseau de Petri comme souligné précédemment.

Un système de transitions *S* est un quadruplet *S = (Q, I, T, →)* tel que :

- *Q est l'ensemble des états ou des configurations.*
- *I est l'ensemble des états initiaux tel que I ⊆ Q.*
- *T est l'ensemble des transitions-actions*
- *→ ⊆ Q X T X Q est la relation de transition.*

Dans ce qui suit, on utilisera la notation $q \xrightarrow{t} q'$ pour désigner le fait que *q* est en relation avec *q'* au lieu de la notation *(q, t, q') ∈ →*.

Une action-transition *t ∈ T* est dite exécutable si et seulement si, il existe deux états *q, q' ∈ Q* tels que $q \xrightarrow{t} q'$.

Une exécution de S est une séquence infinie $\sigma = q_0, q1, ..., q_n$ d'états $q_i \in Q$ tels que $q_0 \in I$ et pour tout $i \in \mathbb{N}$, il existe $(q_i, T_i, q_{i+1}) \in \rightarrow$ pour un certain $T_i \in T$. Un état $q \in Q$ est dit accessible si $q_i = q$ pour une certaine exécution $\sigma = q_0, q1, ..., q_n$ de S.

Pour un model-checking on va considérer des structures de kripke$\mathcal{M}$, plutôt que des systèmes de transitions *S*. En effet, un système très similaire mais pas identique à la notion de système de transition, appelé structure de kripke, est utilisé par le model-checking pour représenter le comportement d'un système. La différence entre les deux concepts qui sont similaires réside dans le fait qu'un système de transition est pratique pour modéliser des systèmes, tandis que la structure de kripke est plus commode pour définir la logique temporelle dans le model-checking. De plus, l'ensemble des transitions-actions *T* qui existe dans le système de transitions est supprimé comme expliqué ci-après car on ne s'intéresse qu'à des propriétés sur les suites d'états visitées plus que les transitions-actions qui se produisent dans ce système, puisqu'on attribue des propriétés atomiques aux états du système.

3.2.2.1 Structure de kripke

Une structure de kripke (Kripke, 1963) est une variante des automates non-déterministes utilisée dans le model-checking pour représenter le comportement d'un système. Elle est considérée comme une structure, dont les nœuds représentent les états atteignables du système et les arcs les transitions d'état.

La fonction de marquage *"Labeling function"* marque chaque nœud par un ensemble de propriétés atomiques qui doivent être valides dans l'état correspondant comme le montre la figure 3.2.

La sémantique de cette structure est basée sur des logiques temporelles pour la plupart des langages de spécification les plus largement utilisés dans le cadre de l'étude des systèmes réactifs.

Une structure de kripke sur un ensemble de variables propositionnelles *AP* est un quadruplet $\mathcal{M} = (Q, I, R, \mathcal{L})$ tel que :

- *Q est l'ensemble fini des états.*
- *I est l'ensemble des états initiaux tel que I ⊆ Q.*
- *R ⊆ Q x Q est une relation de transition qui vérifie: ∀q ∈ Q, ∃q' ∈ Q tel que (q, q') ∈ R.*

- *$\mathcal{L} : Q \rightarrow 2^{AP}$ est une fonction d'étiquetage ou de labellisation des états qui définit pour chaque état $q \in Q$. l'ensemble $\mathcal{L}(q)$ de toutes les propositions atomiques qui sont vraies dans q.*

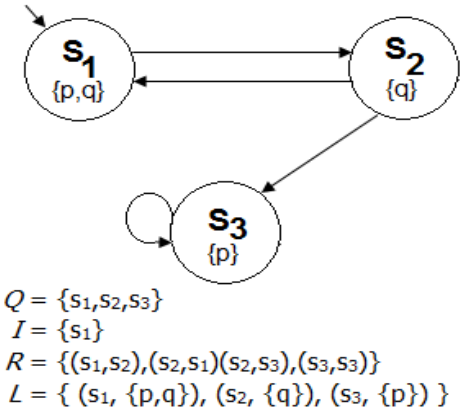


Figure 3.2 *Un exemple de structure de kripke.*

Pour vérifier tous les chemins d'exécutions possibles du système, il est utile de déplier cette structure afin d'obtenir un arbre infini dont la racine est l'état initial de la structure, et chaque nœud de l'arbre a pour successeurs ceux obtenus par la relation de transition *R*.

3.2.3 Les propriétés vérifiées par le model checking

Les propriétés les plus simples et les plus importantes qu'on puisse vérifier par le model-checking sont celles d'accessibilité, d'invariance ou de sûreté telle que « *durant toute l'exécution, x est différent de 0* » ou encore de vivacité telle que « *l'application finit par se terminer* ».

3.2.3.1 Propriété d'atteignabilité *" Reachability"*

L'atteignabilité ou l'accessibilité représente le fait qu'une certaine situation peut être atteinte. On dira que dans une propriété d'atteignabilité *"Reachability property"* on teste si un ensemble de mauvais états *A* peut être atteint comme le fait « qu'*il existe un état accessible où x vaut 0* ». Ce type de propriété doit donc être vérifié et satisfait à chaque étape de l'exécution du système.

3.2.3.2 Propriété de sûreté *"Safety"*

On définit informellement la propriété de sûreté *"Safety property"* comme celle stipulant que « quelque chose de mauvais n'arrive jamais ». Dans le cas où la propriété de sûreté est violée, on obtient un contre-exemple fini, c.à.d. une exécution finie qui viole la propriété. Ainsi, l'invariance est un cas particulier de sûreté, Par exemple, « une variable n'est jamais égale à 0», « le système n'a pas de deadlock ». Ce type de propriété doit donc être vérifié et satisfait à chaque étape de l'exécution du système. Pour la vérification de la propriété de sûreté ou d'invariance, il est possible d'utiliser la logique temporelle mais également de simples assertions.

3.2.3.3 Propriété d'équité *"Fairness"*

Une propriété d'équité *"Fairness property"* permet d'énoncer que sous certaines conditions, quelque chose aura lieu (ou pas) un nombre infini de fois comme par exemple « la porte sera ouverte infiniment souvent ». On parle aussi de vivacité répétée ou d'atteignabilité répétée. Ces différentes terminologies se justifient par des utilisations différentes d'un point de vue méthodologique. Ce sont des propriétés courantes des systèmes dès lors que ceux-ci sont obtenus en réalisant le produit de plusieurs automates.

3.2.3.4 Propriété de vivacité *"Liveness"*

Une propriété de vivacité *"Liveness property"* exprime le fait que « quelque chose de bien arrivera finalement ». Il s'agit d'une propriété plus forte que l'atteignabilité puisque cette dernière indique qu'il est possible que quelque chose se passe alors que la propriété de vivacité indique que quelque chose doit se passer. Cela garantit donc qu'une propriété sera vraie à partir d'un certain état de l'exécution du système.

Ce type de propriété va donc être évalué sur un ensemble d'états représentant un chemin d'exécution du système. Par exemple, « l'application se terminera finalement » ou « chaque processus sera finalement en section critique », etc. Dans ce cas, nous travaillons donc sur un temps infini contrairement à une propriété d'accessibilité ou de sûreté.

3.2.3.5 Propriété d'absence de blocage *"Deadlock-freeness"*

L'absence de blocage "*Deadlock-freeness property*" exprime le fait que le système ne peut pas se trouver dans une situation où il est impossible d'évoluer, si cela n'a pas été explicitement spécifié. Cette propriété est importante dans le sens où elle constitue une propriété de correction pour les systèmes supposés ne jamais se terminer, et constitue une propriété de correction pour tous les systèmes dès lors que l'on a spécifié les états finaux d'un système.

Pour formuler l'ensemble de ces propriétés, il est nécessaire d'utiliser un langage plus riche permettant d'exprimer le temps infini, comme la logique temporelle. En effet, les logiques sont des langages de spécification formels qui ont une définition mathématique précise, ce qui permet d'exprimer les propriétés sans ambiguïté, mais cela permet surtout la simulation et la vérification automatique du système.

Pour exprimer par exemple les propriétés de vivacité, on utilise des logiques temporelles telles que LTL *"Linear Temporal Logic"* qui permet de considérer une propriété le long d'un chemin d'exécution grâce aux connecteurs et non plus sur un seul état. On peut également utiliser la logique CTL *"Computation Tree Logic"* dite branchante, qui permet de prendre en compte les entrelacements des futurs états possibles à un point donné de l'exécution contrairement à LTL qui est linéaire.

3.2.4 La logique Temporelles Linéaire (LTL)

La Logique Temporelle Linéaire (LTL) (Rozier, 2011) est une des variantes de la logique formelle permettant d'exprimer les formules à vérifier par le model-checking. La LTL est basée sur le calcul propositionnel. Les formules du calcul propositionnel sont composées de propositions atomiques, des expressions booléennes, des opérateurs logiques et des quantificateurs comme le montre la figure 3.3.

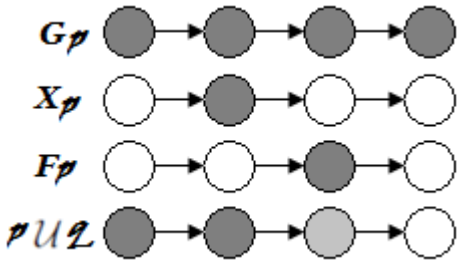


Figure 3.3 *Les connecteurs temporels LTL.*

La Logique Temporelle Linéaire (LTL) est définie par :

- Un ensemble de propositions atomiques *AP* : *p, q, r, etc*.
- Un ensemble de connecteurs logiques portant sur des variables : $\top$ (Vrai), $\bot$ (Faux), ¬ (Négation), ∨ (OU logique), ∧ (ET logique), → (Implication), ↔ (Équivalence).
- Un ensemble de quantificateurs ou connecteurs temporels: $\mathcal{G}$, $\mathcal{X}$;$\mathcal{F}$, $\mathcal{U}$.

Les quatre connecteurs temporels $\mathcal{G}$, $\mathcal{X}$;$\mathcal{F}$ et $\mathcal{U}$ peuvent être expliqués brièvement, comme suit :

- $\mathcal{G}$(toujours où *"always"*): $\mathcal{G}$p signifie que *p* est vrai dans toute l'exécution. Le connecteur $\mathcal{G}$ est représenté graphiquement par : □
- $\mathcal{X}$(prochainement où *"next time"*): $\mathcal{X}$p signifie que *p* est vrai dans l'état suivant le long de l'exécution. Le connecteur $\mathcal{X}$ est représenté graphiquement par : ○
- $\mathcal{F}$(éventuellement où *"eventually"*): $\mathcal{F}$p signifie que *p* est vrai plus tard au moins dans un état de l'exécution. Le connecteur $\mathcal{F}$ est représenté graphiquement par : ◊
- $\mathcal{U}$(jusqu'où *"until"*): *p* $\mathcal{U}$ *q* signifie que *p* est toujours vrai jusqu'à un état où *q* est vrai. Le connecteur $\mathcal{U}$ est représenté graphiquement par : u

Les opérateurs ○, ◊ et □ sont unaires alors que l'opérateur u est binaire. Les opérateurs temporels et propositionnels peuvent être combinés dans la même formule LTL. Ainsi les formules LTL peuvent être sous la forme suivante:

- Toute proposition atomique à savoir p, q est une formule.
- Si ϕ et ψ sont des formules, alors $\neg \phi$, $\phi \vee \psi$, $\phi \wedge \psi$, $\phi \rightarrow \psi$ *et* $\phi \leftrightarrow \psi$ sont aussi des formules.
- Si ϕ et ψ sont des formules, alors $\mathcal{G}\varphi$, $\mathcal{X}\varphi$, $\mathcal{F}\varphi$ et φ $\mathcal{U}$ ψ sont aussi des formules.

La sémantique d'une formule LTL syntaxiquement correcte est définie en lui attribuant une interprétation (une assignation des valeurs de vérité, $\top$ (vrai), $\bot$ (faux)) à ses propositions atomiques et l'extension de l'assignation à une interprétation de la formule entière en tenant compte des règles associées aux opérateurs. Pour le calcul propositionnel, ces règles sont définies par les tables de vérité usuelles. Pour la logique temporelle, la sémantique d'une formule est donnée en termes d'exécutions (calculs) et les états d'une exécution. Les propositions atomiques de la logique temporelle sont des expressions booléennes qui peuvent être évaluées dans un seul état indépendamment d'une exécution.

3.3 SPIN Model Checker

SPIN model-checker (Mordechai, 2008) a été choisi à l'origine comme un acronyme pour Simple Promela INterpreter. C'est un outil open-source largement utilisé pour vérifier et valider les propriétés d'exactitude des systèmes logiciels de grande envergure. Ce dernier a été développé par Gerard J. Holzmann (Holzmann, 1997), (Holzmann, 2003) pour vérifier les protocoles de communication. Il a été très utilisé dans les industries qui réalisent des systèmes critiques.

Les modèles qui doivent être vérifiés par SPIN sont écrits en Promela (PROcess MEta LAnguage)[15], Promela est un langage de spécification de systèmes asynchrones tels que les systèmes concurrents ou les protocoles de communication qui décrivent le comportement du système. La syntaxe de Promela est proche du langage C enrichi avec des primitives de communications basées sur des machines à états finis. Ce langage est conçu pour faciliter la construction de modèles de systèmes distribués en soutenant la spécification des structures de contrôle non-déterministe. Il autorise la création dynamique de processus. La communication entre ces processus peut être effectuée par le partage de variables globales ou en utilisant des canaux de communication.

Promela permet ainsi de détecter notamment la présence de deadlock, race condition ou des suppositions injustifiées sur les vitesses relatives de processus. Les propriétés à vérifier doivent être exprimées en Logique Temporelle Linéaire (LTL).

Un modèle Promela peut être analysé par SPIN, soit par simulation où le modèle est exécuté étape par étape, ce qui rend plus facile de se familiariser avec son comportement, soit par vérification où les états du modèle sont explorés de façon exhaustive afin de vérifier que le modèle satisfait les propriétés spécifiées dans LTL.

En effet, étant données la spécification formelle du comportement du système et la spécification d'une propriété en logique temporelle, SPIN model-checker vérifie si cette propriété est satisfaite. Son mode de vérification détermine si le modèle proposé en Promela satisfait ou non une propriété LTL. Ceci est effectué en menant, si nécessaire, une vérification exhaustive qui permet de parcourir l'ensemble des exécutions possibles du système.

Le principe de vérification de SPIN consiste à transformer les processus PROMELA en structure de kripke à travers une cartographie (en vrac) des processus, des états et des canaux de communication de Promela vers les états-transitions de la structure de kripke (Walton, 2005) et les formules LTL en automates de Büchi (Büchi, 1962).

En effet, SPIN transforme, à partir des propriétés spécifiées, leur négation en automates de Büchi (Büchi, 1962), puis calcule le produit synchronisé de cet automate avec celui du système c.à.d. la structure de kripke. Le résultat est encore un automate de Büchi dont on vérifie si le langage est vide ou non. S'il est vide, alors il n'existe aucune exécution violant la formule de départ, sinon il accepte au moins une exécution.

L'ensemble des transitions pour lesquelles la propriété n'est pas vérifiée forment les contre-exemples qui peuvent être générés par SPIN en appliquant un algorithme de recherche en profondeur imbriquée (Nested Depth-first Search) sur cet espace d'états. Ces

[15] http://spinroot.com/spin/Man/Quick.html

contre-exemples sont censés permettre à l'utilisateur de modifier par la suite son modèle ou la propriété si elle a été mal exprimée, afin de corriger les erreurs éventuelles.

Avant de détailler notre approche qui consiste à vérifier un des aspects de la cohérence des processus métier qui se résume en la détection de l'absence des erreurs structurelles dans des modèles de processus métier à l'aide de SPIN, nous allons commencer par présenter en détail ces erreurs qui peuvent causer des incohérences fonctionnelles comme expliqué brièvement dans le chapitre 2.

3.4 Les erreurs structurelles

Un processus est dit sain ou cohérent s'il ne contient pas d'activité inutile et où chaque cas se termine complètement sans laisser de référence à lui-même. Un des aspects de cohérence des processus métier le plus vérifié par les travaux existants dans la littérature se manifeste par la vérification de l'absence des erreurs structurelles.

Dans cette section, nous allons présenter quelques erreurs structurelles qui peuvent survenir lors de l'exécution des modèles de processus métier. Le terme d'erreurs structurelles fait référence aux erreurs qui peuvent survenir durant l'exécution de ces modèles, et qui peuvent causer des incohérences fonctionnelles telles que des impasses, des boucles infinies ou des terminaisons multiples. Ces erreurs sont utilisées pour illustrer notre approche proposée (Kherbouche, Ahmad, & Basson, 2012), (Kherbouche, Ahmad, & Basson, 2013) pour garantir la cohérence *"soundness"* des modèles de processus métier.

3.4.1 L'impasse *"Deadlock patterns"*

En règle générale, une impasse ou un blocage est défini comme étant un système qui a atteint un état non-final de non-progression dans son exécution. Pour les modèles de processus métier, cela correspond au cas où certaines activités ne peuvent pas continuer à progresser, alors qu'elles n'ont pas atteint la fin du processus. Par conséquent, une impasse est une condition utilisée pour décrire une situation où un processus métier ne peut être terminé.

Selon Onada *et al.* (Onoda, Ikkai, Kobayashi, & Komoda, 1999), il existe deux concepts complémentaires qui permettent de détecter une impasse. Le premier est l'atteignabilité *"Reachability"* qui est symbolisé par l'existence d'au moins un chemin d'un nœud *A* vers un nœud *B* dans un processus et le second concept est la transférabilité absolue ou *"absolute transferability"*, qui est un concept plus fort que le premier, et qui permet d'affirmer qu'un jeton peut toujours être transféré à partir du nœud *A* à tous les points d'entrée du nœud *B*. Le blocage se produit, chaque fois qu'il y a accessibilité sans transférabilité absolue.

Dans le même travail, les auteurs ont également identifié plusieurs causes potentielles de blocages, comme suit:

- Impasse ou blocage infinis *"Loop deadlock"*: comme indiqué dans la figure.3.4.a, se produit quand il ya un chemin d'exécution dans lequel la sortie d'une passerelle *AND-Join* retourne à nouveau à ses points d'entrée. Dans ce cas, si le chemin ne contient pas de passerelles *XOR-Split*, l'apparition de blo-

cage est certaine, sinon il peut se produire si la branche menant à la boucle est choisie.

- Sources multiples *"Multiple sources"*: comme indiqué dans la figure.3.4.b, les sources multiples se produisent lorsque deux sources différentes mènent aux points d'entrée d'une passerelle *AND-Join*. En supposant qu'aucun des nœuds sources n'est une passerelle *AND-Join* en elle-même, on peut observer que les sources multiples peuvent se produire lorsque l'une des structures de processus est la suivante:
  - L'une des deux sources est une passerelle *XOR-split*.
  - Un processus avec multiples événements de début qui seront synchronisés plus tard par une passerelle *AND-Join.* D'où on peut en déduire qu'il existe une accessibilité entre deux ou plusieurs sources pour le nœud *AND-join*.
  - Une structure impropre comme le montre la figure 3.4.d, représente le fait d'une passerelle *AND-Join* reçoit comme entrée un chemin qui passe par une passerelle XOR-Split

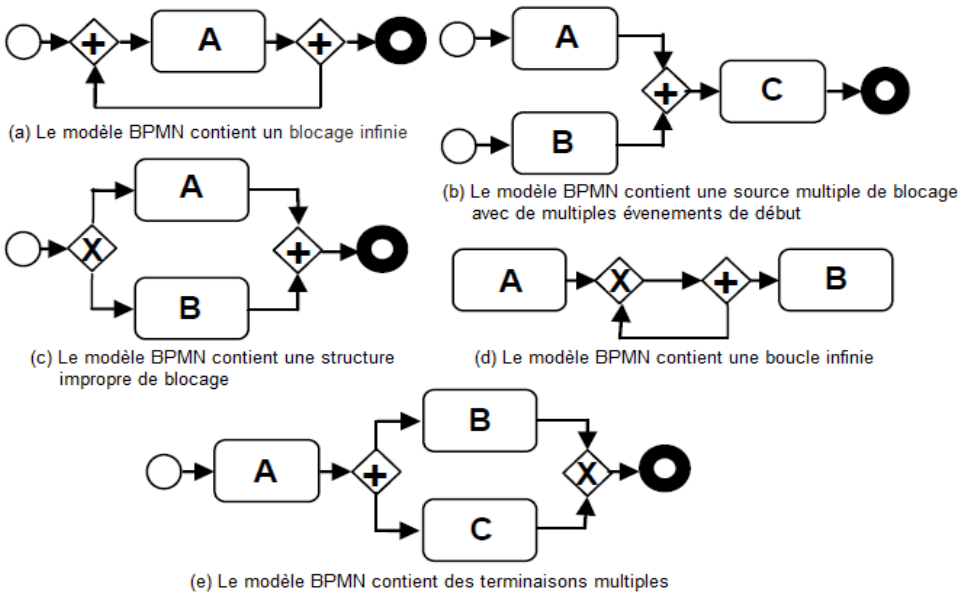


Figure 3.4 *Les erreurs structurelles dans les modèles BPMN.*

3.4.2 Boucles infinies *"Livelocks patterns"*

Une boucle infinie ou *"Livelock"* peut être définie comme étant un état à partir duquel il est possible de procéder, mais impossible d'atteindre l'état final désiré (le système est verrouillé dans un petit sous-ensemble d'états et ne progresse pas). Comme le montre la figure.3.4.d, une boucle peut entraîner une exécution infinie du processus. Dans ce cas, certains processus peuvent s'exécuter avec succès, tandis que d'autres peuvent être dans une boucle sans fin d'exécution. Un exemple de cette situation peut se produire quand une passerelle *AND-Split* est utilisée à la place d'une passerelle *XOR-Split* pour modéliser une boucle existante.

3.4.3 Terminaison multiples *"Multiples terminations patterns"*

Les multiples terminaisons correspondent à des situations où il existe une passerelle *AND-Split* avant une passerelle *XOR-Join*, comme représenté sur la figure.3.4.e. Dans ce cas, une seule séquence est traversée lorsque la passerelle *XOR*-Join est exécutée, ce qui conduit à la violation du critère de cohérence. Ainsi, le modèle de processus BPMN ne terminera pas l'exécution du processus, correctement et dans la façon attendue.

3.5 La vérification des modèles de processus métier basée sur le model-checking

Afin de vérifier notre modèle de processus métier, nous avons à décrire en LTL l'ensemble des propriétés qui doivent être utilisées par le model-checker pour détecter d'éventuelles erreurs structurelles et d'assurer ainsi de la cohérence de ce dernier. Mais avant une telle description, il faut proposer une structure de kripke représentant le comportement de ce modèle qui est nécessaire pour la vérification de ces propriétés comme soulevé précédemment.

Notre approche est décrite dans la figure. 3.5. L'idée principale est de cartographier les différents éléments qui constituent un modèle de processus métier exprimé en BPMN en structure de kripke dans un but de spécifier le comportement de ce dernier, et de lui attribuer ainsi une sémantique formelle.

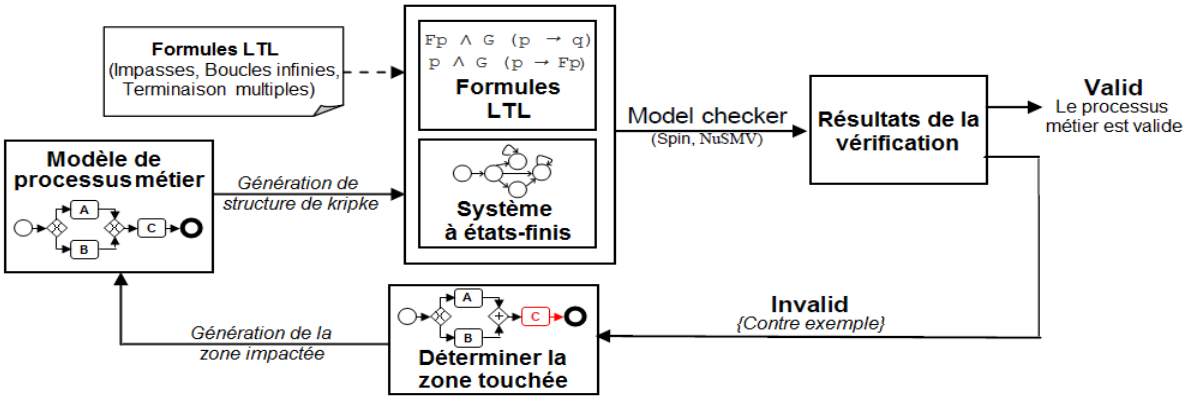


Figure 3.5 *Vérification des modèles de processus métier basée sur le model-checking.*

Plusieurs propriétés LTL peuvent être définies simultanément pour une structure de kripke. Les étapes de vérification sont détaillées comme suit :

3.5.1 Génération de l'automate à états finis

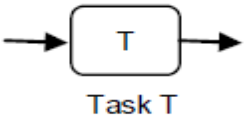
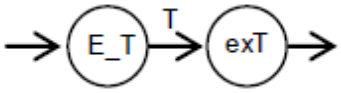
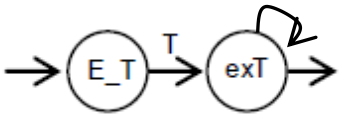
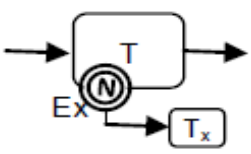
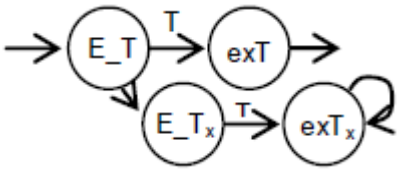
La première étape consiste à cartographier les modèles de processus BPMN en structure de kripke pour exprimer le comportement de ces derniers. Cette traduction permet une meilleure vérification des propriétés temporelles souhaitées telles que: $\mathcal{M} \mid = \phi$ ($\mathcal{M}$ satisfait ϕ) si est seulement si, $\mathcal{M}$, $\pi \mid = \phi$ est vérifiée pour tous les chemins d'exécutions π dans une structure de kripke $\mathcal{M}$.

L'ensemble non vide de l'automate à états finis S de la structure de kripke représente l'ensemble des nœuds N du modèle de processus (N représente l'ensemble des objets de flux du processus BPMN qui peut être partitionné en événements E, activités A et passerelles G). Les relations de transition R représentent l'ensemble des flux de contrôle reliant les différents objets de flux du processus BPMN tel que $T \subseteq F \times F$.

Pour obtenir une structure de kripke, nous définissons *AP* comme étant l'ensemble des propositions atomiques associées à chaque état s ∈ S tels que $\mathcal{L}(s)$ est valide *"holds"* en *s* ($\mathcal{L}$ est la fonction d'étiquetage de la structure de kripke $\mathcal{M}$). Elle exprime toutes les propriétés d'un état s ∈ S.

L'état initial I ⊆ S est le point de départ E^6 (événement de début) du modèle de processus métier. Chaque état *s* est étiqueté avec des transitions : "activée" ou "exécutée". *E_A* signifie que la transition *A* est activée et *ex_A* signifie que la transition *A* est exécutée (achevée).

Une brève description de la cartographie des principaux éléments du modèle de processus BPMN en structure de Kripke est donnée dans la Table.3.1.

Les éléments BPMN	**Structure de kripke**
	
	
	
	
	
	

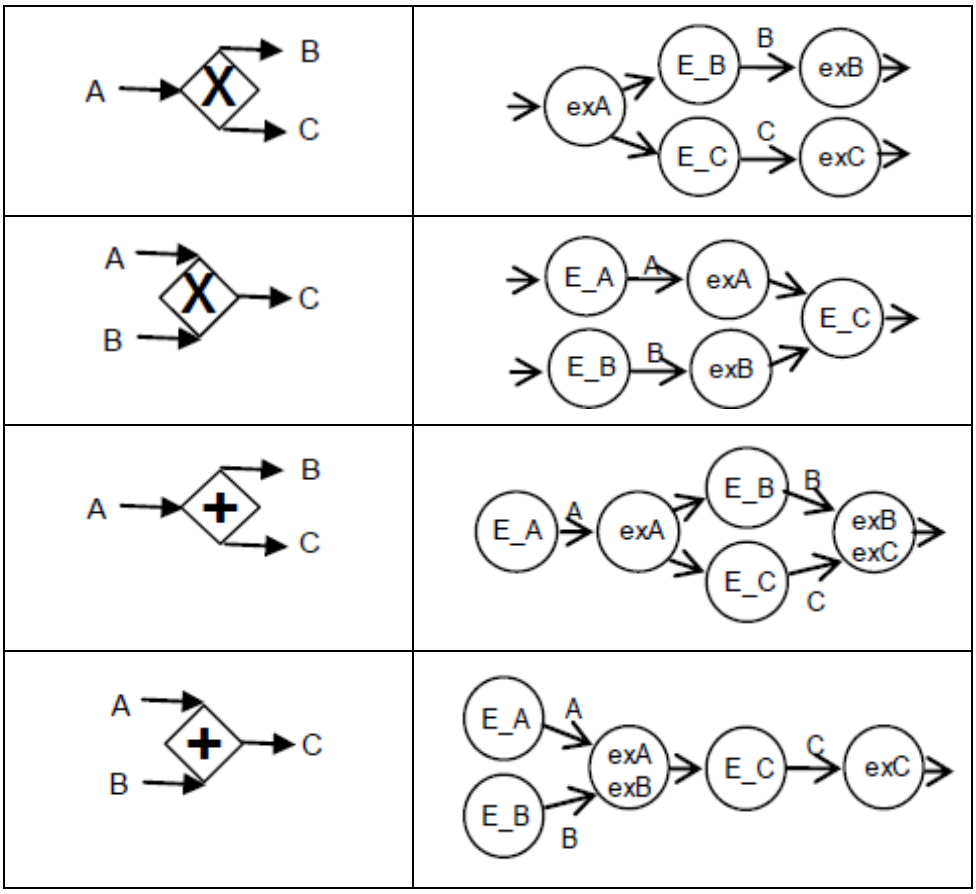


Table 3.1 *La cartographie des éléments de BPMN en structure de kripke.*

3.5.2 Formules LTL de la cohérence d'un modèle de processus métier

La cohérence d'un modèle de processus métier et par conséquent l'absence d'erreurs structurelles peut être assurée en vérifiant les formules LTL suivantes :

3.5.2.1 Détection de l'absence de blocages

Une structure de kripke est dite être sans blocage *"Deadlock-free",* si elle ne contient aucun chemin qui peut conduire à une impasse. Le non-blocage est considéré comme une propriété de sûreté « quelque chose de mauvais n'arrive jamais ».

Supposons que la formule temporelle `Final`, représente l'ensemble des états finaux dans une structure de kripke comme le montre la deuxième ligne, deuxième colonne dans la Table 3.1. Dans un tel cas, nous pouvons exprimer la propriété de non-blocage par la formule LTL suivante:

$$\Box (\circ \bot \rightarrow \text{Final})$$

Cette formule doit être considérée comme valide dans chaque chemin. La formule $\circ \bot$ (signifie qu '«il n'y a pas d'état prochain») est facile à déduire, c.à.d. aucune transition n'est possible. Donc, nous pouvons exprimer la propriété d'accessibilité d'un état de blocage comme étant l'existence d'un état avec la double propriété.

$$\Diamond (\circ \bot \rightarrow \neg \text{Final})$$

3.5.2.2 Détection des boucles infinies

Comme décrit précédemment, la boucle infinie est un état où il est impossible de progresser jusqu'à l'état `Final` recherché. La propriété qui exprime la non-existence de boucle infinie est une propriété de vivacité « quelque chose de bien arrive finalement ». Cette formule peut être exprimée en LTL comme suivant:

$$\Diamond\Box\phi \rightarrow \Box\Diamond\psi$$

Si une activité essaie de s'exécuter à l'infini, alors elle sera toujours dans un état d'exécution. Cela peut être simplifié par la formule ¬ ◊ □ ϕ (c'est-à-dire qu'elle ne réussira jamais à s'exécuter pour toujours). Le contre-exemple de ces propriétés est une exécution infinie selon laquelle le comportement attendu ne se produit pas (c'est-à-dire le processus ne se termine pas).

3.5.2.3 Détection des terminaisons multiples

La terminaison multiple est une situation où il existe un *AND-Split* avant une passerelle *XOR-Join* comme vu précédemment. La détection de ce cas est basée sur la vérification de la propriété de sûreté « quelque chose de mauvais n'arrive jamais ». Il peut être vérifié par la formule LTL suivante:

$$\Box \neg (\Diamond (\phi \wedge \circ \psi) \rightarrow \text{Final})$$

Un contre-exemple de ces propriétés est une exécution finie qui conduit à un comportement inattendu.

3.5.3 Model Checking

La structure de kripke et les formules de logique temporelle sont présentées comme entrée pour le model-checker. Ce dernier vérifie si la formule temporelle ϭ est satisfaite par la structure de kripke $\mathcal{M}$ ou pas. En conséquence, il confirme le bien-fondé du modèle de processus. Sinon, il renvoie un contre-exemple en cas d'erreurs structurelles.

3.5.4 Déterminer la zone impactée

Le model-checker a la capacité de fournir des contre-exemples dans le cas où les propriétés temporelles à vérifier ne sont pas satisfaites par le modèle de processus (Liu, Müller, & Xu, 2007), (Foerster, Engels, & Schattkowsky, 2005).

Généralement, ces contre-exemples sont donnés en termes de transitions d'états internes plutôt qu'en termes de modèles de processus, ce qui les rend difficiles à comprendre par un utilisateur non-initié.

Pour bénéficier de ces contre-exemples, cette sortie du model-checker devrait être traduite dans une notation visuelle, ce qui est plus facile pour l'utilisateur à comprendre. La transformation de ce contre-exemple vers le modèle de processus BPMN source permet de mettre en évidence visuellement les chemins dans un modèle de processus métier dont l'exécution entraîne des violations.

À cet effet, nous utilisons *"Model checker dependency"*, qui contient une chaîne d'outils permettant de traduire la sortie du model-checker vers les modèle de processus métier. Cela peut permettre de faire correspondre chaque état de la structure de kripke vers

l'élément du modèle de processus BPMN correspondant, en mettant en évidence la zone impactée (par une couleur ou un étiquetage particulier), et cela afin de mieux la visualiser. Cette correspondance peut aider à trouver les causes réelles de ces erreurs dans le modèle de processus métier et de les corriger ainsi.

3.6 La vérification des modèles de processus métier à l'aide de SPIN

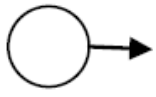
Nous présentons dans cette section une méthode, basée sur le model-checker SPIN, pour implémenter notre approche qui permet la vérification de certaines propriétés de cohérence d'un modèle de processus métier. Avant d'interroger SPIN, nous proposons une translation adaptée des travaux de (Vaz & Ferreira, 2007), (Vaz & Ferreira, 2007) d'une spécification du modèle de processus métier modélisé en BPMN vers le langage Promela et l'illustrons par un exemple.

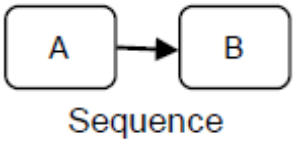
Les processus métier sont traduits en langage Promela en mappant les différents processus, les sous-processus, et les activités de ces derniers en processus Promela *"PROMELA processes"*, et les flux de séquence en canaux de communication Promela *"PROMELA channels"*. Les flux de message entre les différents processus sont représentés, sans perte de généralité en utilisant des entiers dans Promela.

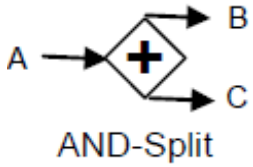
Dans cette translation nous utilisons quelques variables telles que `c` pour représenter un canal de communication, `m` pour représenter un message (envoyé ou reçu) dans un canal `c` qui peut être un entier, un byte, etc. `cS` désigne un tableau de canaux de communication et `mS` désigne un tableau de messages à envoyer ou à recevoir dans chaque canal de `cS`.

Dans la Table 3.2, nous représentons la traduction des principaux éléments de la notation BPMN qui sont *la séquence*, la passerelle *AND-Split*, la passerelle *AND-Join*, la passerelle *XOR-Split*, la passerelle *XOR-Join*, la passerelle *OR-Split*, la passerelle *OR-Join* en langage PROMELA.

Les éléments BPMN	Langage Promela

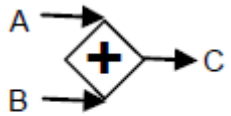
	`chan c = [1] of {int};`
	`active proctype start(){`
	`  /* To start process */`
	`  c!1;`
	`}`

	`proctype A(){`
	`  /* Details of activity */`
	`  c!1; /* Activate process B */`
	`}`
	`proctype B(){`
	`  int x;`
	`  c?x; /* Waiting token to run */`
	`  /* Details of activity */`
	`}`



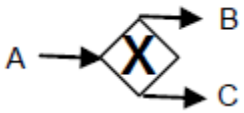
AND-Split

```
inline ANDSplit(cS, mS){
  int ind, length;
  ind = 0; length = len(cS);
  atomic {
    do
      ::ind <length->cS[ind]!mS[ind]; ind++;
      ::ind >=length->break;
    od;
  }
}
```



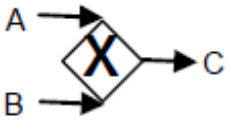
AND-Join

```
inline ANDJoin(cS, mS){
int ind, length;
ind = 0; length = len(cS);
skip;
 S:
  if
    ::full(cS) ->
    do
      ::ind <length && nempty(cS[ind])->
cS[ind]?mS[ind];ind++;
      ::ind >=length-> break; goto E
    od;
    :: nfull(cS) -> ind=0; timeout; goto S
  fi;
 E: skip;
}
```



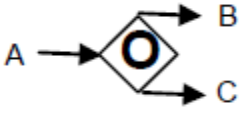
XOR-Split

```
inline XORSplit(cS,choice,m) {
 int length;
 length = len(cS);
 if
   ::(choice<length && choice>=0) -> cS[choice]!m;
   :: else -> skip;
 fi;
}
```



XOR-Join

```
inline XORJoin(cS, m){
 int ind,length;
 ind=0;
 length=len(cS);
 skip;
 B:
   if
    ::nempty(cS[ind])-> cS[ind]?m; goto E
    ::empty(cS[ind])-> ind++; goto I
   fi;
 I:
   if
    ::ind==length -> ind=0; timeout; goto B
    ::ind<length -> goto B
   fi;
 E: skip;
}
```



OR-Split

```
inline ORSplit(cS, mS, aCs, choices){
 int ind, length;
 ind=0; length=len(cS);
 skip;
 S:
   if
    ::choices[ind]==1 -> aCs[ind]!1;cS[ind]?mS[ind];
    ::choices[ind]==0 -> aCs[ind]!0;
   fi;
   ind++;
   if
    ::ind < length -> goto S
    ::ind >= length -> skip
   fi;
}
```

	`inline ORJoin(cS, aCs, mS){`
	` int ind, length, x;`
	` ind =0;length = len(aCs);`
	` do`
	`  ::ind <length -> timeout; ind++;`
	`  ::ind ==length -> ind=0; break;`
	` od;`
	` skip;`
	` B:`
	`  if`
	`  ::length > 0 && ind <length && aCs[ind]?x == 1->`
	`  cS[ind]? mS[ind]; ind++;`
	`  ::ind >= length -> goto E`
	`  fi;`
	` E: skip;`
	`}`

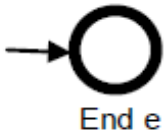
	`proctype End(cS, mS){`
	` int ind, length;`
	` ind = 0; length = len(cS);`
	` end:`
	` do`
	`  ::ind <length->cS[ind]?mS[ind];ind++;`
	`  ::ind >=length->break;`
	` od;`
	`}`

Table 3.2 *La cartographie des éléments de BPMN en Promela model.*

3.6.1 Processus de vente de voitures

L'exemple d'illustration de cette translation est représenté par un simple processus de vente de voitures *"Car Salesman process"*, comme le montre la figure 3.6.

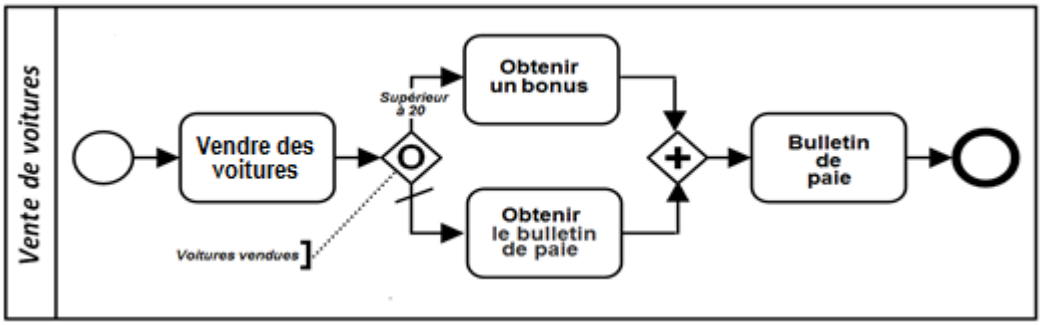


Figure 3.6 *Processus de vente de voitures.*

Dans ce processus, le vendeur reçoit son bulletin de paie à la fin de chaque mois. Il obtient un bonus à chaque fois qu'il réussit à vendre plus de vingt voitures durant le mois (en plus de son salaire régulier).

Le blocage ou l'impasse dans ce processus peut se produire éventuellement lorsque le vendeur vend moins de vingt voitures par mois c.à.d. le vendeur n'obtient pas de bonus. Dans ce cas, la passerelle *AND-Join* reçoit une seule séquence qui est traversée par une des deux activités c.à.d. l'activité « obtenir le bulletin de paie » et pas l'activité « obtenir un bonus » ce qui provoque une situation de blocage comme décrit dans la section 4.

L'absence de blocages et de boucles infinies dans SPIN est détectée par la vérification de l'absence d'états invalides *"invalid endstates"* et l'absence de cycle de non-progression *"non-progress cycle"*, respectivement.

Le modèle PROMELA correspondant au processus de vendeur de voiture est décrite comme suit:

Nous définissons un tableau de six canaux de communication représenté par la variable `cS` et cela afin d'établir des communications entre les différents processus Promela. Nous définissons aussi `CS1` et `CS2` comme étant des tableaux auxiliaires de canaux de communication pour simplifier les échanges de message au sein du même processus. Le modèle PROMELA des différentes activités est détaillé ci-dessous:

L'activité Vendre des voitures : cette activité se lance dès qu'elle reçoit un signal de l'événement de début, elle choisit de faire appel aux activités qui la succèdent d'une manière non-déterministe sans perte de généralité après qu'elle soit achevée. Elle peut être traduite en processus PROMELA comme souligné dans le Listing 3.1 :

Listing 3.1 *Traduction de l'activité Vendeur de voiture en processus Promela.*

```
proctype sellCars(){
  chan cS1[2]=[1] of {int};
  int x, choice, msgs[2];
  cS1[0]= cS[1]; /* send to Get Bonus */
  cS1[1]= cS[2]; /* send to Get Paycheck */
  R: cS[0]?x; /* receive from start event */
  /* Choice of a non-deterministic manner */
     if
          ::choice=0  /* Less that 20 cars */
          ::choice=1  /* More that 20 cars */
     fi;
     msgs[0]=1;
     msgs[1]= choice;
     ORSplit(cS1,msgs,cS1, msgs);
     goto R
}
```

L'activité Obtenir un bonus : comme mentionné ci-dessus, cette activité est choisie de manière non-déterministe sans perte de généralité. Elle peut être traduite en processus PROMELA comme souligné dans le Listing 3.2 :

Listing 3.2 *Traduction de l'activité Obtenir un bonus en processus Promela.*

```
proctype GetBonus(){
  int x;
  cS[1]?x; /* To receive from Sell Cars */
  cS[3]!1; /* To send to Pay Bills */
}
```

L'activité Obtenir le bulletin de paie : de la même manière, cette activité est choisie de manière non-déterministe sans perte de généralité. Elle peut être traduite en processus PROMELA comme le montre le Listing 3.3 :

Listing 3.3 *Traduction de l'activité Obtenir le bulletin de paie en processus Promela.*

```
proctype GetPayCheck(){
  int x;
  cS[2]?x; /* To receive from Sell Cars */
  cS[4]!1; /* To send to Pay Bills */
}
```

L'activité Bulletin de paie : comme vu précédemment, quand le vendeur vend plus de vingt voitures, un bonus (en plus de son salaire régulier) est ajouté à son bulletin de paie.

Cette activité peut être traduite par le processus PROMELA comme le montre le Listing 3.4 :

Listing 3.4 *Traduction de l'activité bulletin de paie en processus Promela.*

```
proctype PayBills(){
  chan cS2[2]=[1] of {int};
  int msgs[2];
  cS2[0]= cS[3]; /* receive from GetBonus */
  cS2[1]= cS[4]; /* receive from GetPaycheck */
  T: ANDJoin(cS2, msgs); /* To receive from Get Bonus and Get PayCheck */
     cS[5]!1; /* send to end event */
     goto T;
}
```

Le fichier « Car_Salesman_process.pml» représente le modèle Promela complet qui contient la description détaillée du processus de vente de voitures fourni à SPIN model-checker comme entrée, sa description est détaillée dans le Listing 3.5 :

Listing 3.5 *Le fichier Car_Salesman_process.pml*

```
chan cS[6] = [1] of {int};
proctype Start(){....}
proctype sellCars(){....}
proctype getBonus(){....}
proctype getPayCheck(){....}
proctype PayBills(){....}
proctype End(){....}
init {
  atomic{
         run start(); run sellCars();
         run getBonus();run getPayCheck();
         run PayBills();run End();
  }
}
```

La capture d'écran imprimée dans la figure 3.7 présente la spécification Promela du processus de vendeur de voitures ainsi que le résultat de la vérification de la cohérence par jSpin[16]. La figure 3.8 présente la structure de kripke générée par cet outil correspondant à ce processus.

3.7 La vérification des règles de conformité

Les processus métier doivent se conformer à un ensemble de règles, appelées règles de conformité ou *"Compliance rules"* décrivant les règlements, les politiques et les contraintes qu'une organisation doit respecter et dont le non-respect peut conduire à des sanctions financières ou à un déficit en matière de bonne réputation.

Cependant, l'évolution rapide de ces règles exige une vérification des modèle de processus métier à chaque fois qu'une règle est ajoutée ou modifiée (El Kharbili, de Medeiros, Stein, & van der Aalst, 2008). Selon Goedertier dans (Goedertier & Vanthienen, 2006), l'origine de cette dynamicité vient essentiellement du changement fréquent dans les réglementations extérieures tel que l'évaluation des risques dans le secteur bancaire (Bâle **II**[17]) que doit respecter l'entreprise et aussi le changement de politiques intérieures

[16] http://code.google.com/p/jspin/
[17] http://www.banque-credit.org/pages/pilier-de-bale2.html

définies par l'entreprise comme par exemple, certains ordres d'exécution entre les activités.

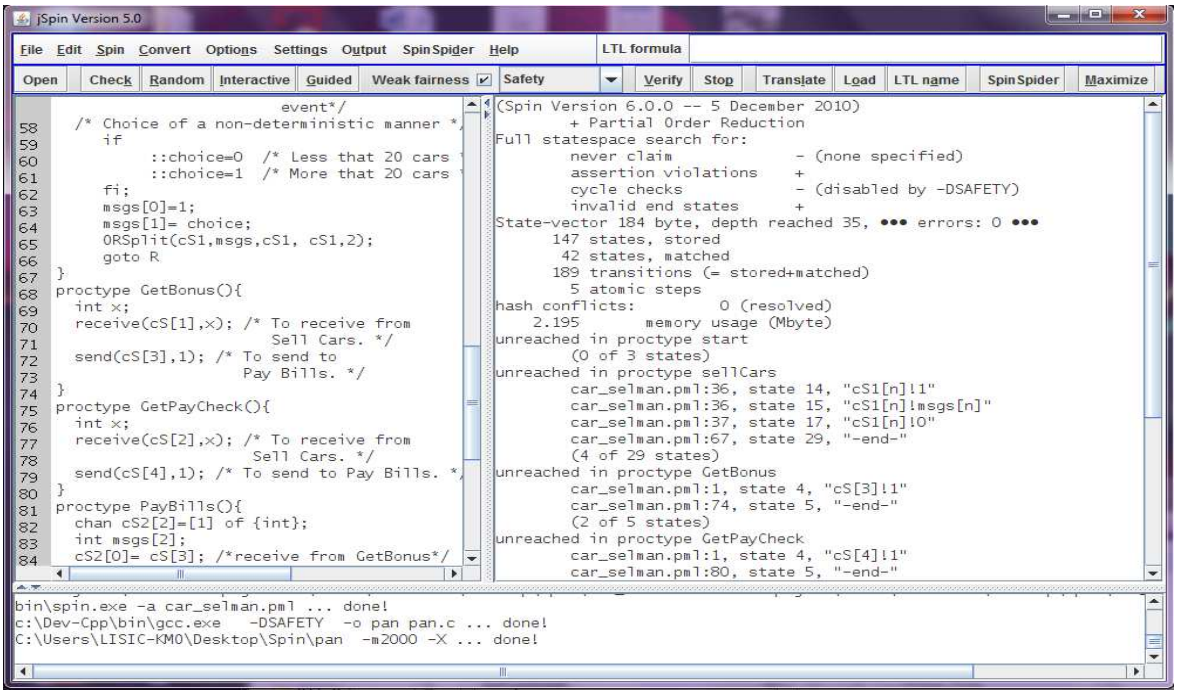


Figure 3.7 *Vérification de la cohérence du processus de vente de voitures en jSpin*

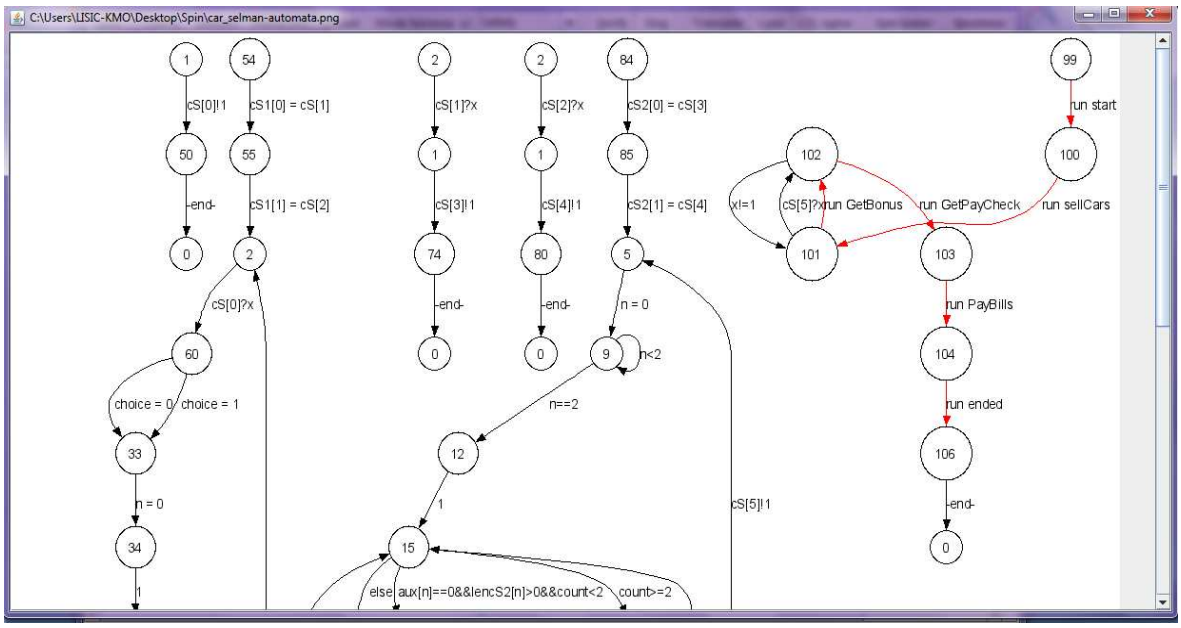


Figure 3.8 *La structure de kripke générée par jSpin*

Étant donné le grand nombre de règles et leurs fréquences de changement, une vérification manuelle de ces règles représentera une tâche assez onéreuse en matière de temps et d'effort. Il convient donc de mettre en place des mécanismes de vérification automatiques de la conformité par les entreprises afin de gagner en gain de productivité.

Différentes approches ont été proposées pour permettre une telle vérification, comme dans (Goedertier & Vanthienen, 2006), où les auteurs proposent une approche pour garantir la conformité des processus métier par le design, en introduisant PENELOPE comme

étant un langage déclaratif qui permet de capturer les obligations et les autorisations imposées par les politiques commerciales (séquençage et contraintes entre activités) et plus tard, de générer automatiquement des modèles de processus métier qui sont par leur conception , conforme avec ces politiques.

Dans (Ghose & Koliadis, 2007), une approche pour vérifier la conformité des modèles de processus métier et la résolution des violations a été introduite. Le document définit un réseau sémantique de processus (SPN), où chaque activité est annotée avec des prédicats permettant ainsi une vérification de l'ensemble des règles. Dans (Liu, Müller, & Xu, 2007), une approche basée sur le model-checking a été proposée pour vérifier la conformité des modèles de processus métier défini en BPEL avec un ensemble de contraintes définies dans un langage de spécification de propriétés (BPSL) qui est converti par la suite en LTL.

Une autre approche (Awad, Decker, & Weske, 2008) est capable de vérifier les règles de conformité exprimées en PLTL *"Past linear time temporal logic"* plutôt que LTL pour plus expressivité à l'aide d'un langage de requête appelé BPMN-Q (Awad, 2007).

Dans notre approche (Kherbouche, Ahmad, & Basson, 2013), la même démarche proposée dans ce chapitre permettant la vérification de la cohérence des processus métier exprimés en BPMN est adoptée pour vérifier la conformité de ces modèles. À cet effet, et en premier lieu, les modèles de processus BPMN sont mappés en structures de kripke pour exprimer leur comportement comme détaillé précédemment. Les règles de conformité sont exprimées en formules LTL à l'aide d'une notation graphique, permettant à la fois d'obtenir le même niveau d'abstraction que les modèles de processus métier mais aussi d'avoir une interface intermédiaire entre les règles de conformité et les formules LTL exprimées dans une notation particulière pour un outil donné comme jSpin, EpiSpin, etc (*cf.* Sous-section 7.2).

Cette technique permet aux experts de vérifier facilement et rapidement les règles de conformité après chaque changement. Notre approche n'est pas limitée aux processus définis en BPMN mais elle peut être appliquée à tout langage graphique permettant la définition des processus métier comme le diagramme d'activités UML, le diagramme EPC, etc.

3.7.1 Représentation déclarative des règles de conformité

Les règles de conformité sont généralement difficiles à comprendre sous un format textuel pour le modèle de processus métier. À cet égard, nous avons proposé un ensemble de notations graphiques appelées G-LTL, qui permet d'exprimer des formules LTL sous forme graphique pour les utilisateurs non-initiés aux différents outils de model-checker.

L'objectif de cette définition est de faciliter l'expression des différentes règles de conformité et d'obtenir le même niveau d'abstraction que les modèles de processus métier. La Table 3.3 résume les différents éléments utilisés dans cette notation graphique G-LTL.

G-LTL	Formule LTL	G-LTL	Formule LTL
Prop.	G Prop.	Prop. 1 ∧ Prop. 2	Prop. 1 ∧ Prop. 2
Prop.	X Prop.	Prop. 1 ∨ Prop. 2	Prop. 1 ∨ Prop. 2
Prop.	F Prop.	Prop. 1 X Prop. 2	Prop. 1 XOR Prop. 2
Prop. 1 Prop. 2	Prop. 1 $\mathcal{U}$ Prop. 2	Prop. 1 → Prop. 2	Prop. 1 → Prop. 2
! Prop.	¬ Prop.	Prop. 1 ↔ Prop. 2	Prop. 1 ↔ Prop. 2
Prop. 1 ∧ Prop. 2	(Prop. 1 ∧ Prop. 2)		

Table 3.3 *La notation graphique G-LTL.*

Le générateur transforme automatiquement cette notation graphique en formules LTL correspondantes. Toutes les formules générées sont cantonnées par l'opérateur $\mathcal{G}$ pour exprimer le fait que la règle à vérifier doit être satisfaite dans tous les scénarios d'exécution possibles. Un exemple d'utilisation de cette notation ainsi que les formules générées sont donnés dans la sous-section suivante.

3.7.2 Processus de remboursement des frais de fonctionnement

A titre d'illustration, nous proposons un exemple (*cf.* figure 3.9) représentant un processus de remboursement des frais de fonctionnement des employés d'une entreprise.

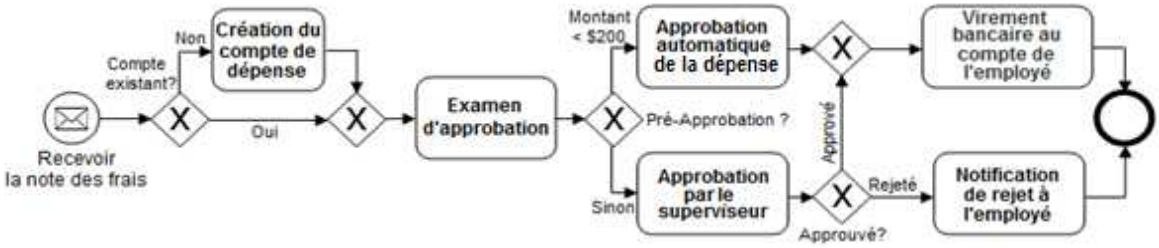


Figure 3.9 *Processus de remboursement des frais de fonctionnement*

Dans ce processus, les employés d'une entreprise réclament un remboursement de leurs dépenses, par exemple, l'achat de fournitures de bureau ou des logiciels. Après que la note de frais soit reçue, un nouveau compte de dépense est créé si l'employé n'en possède pas un.

Une approbation automatique du remboursement est effectuée si le montant de la dépense est inférieur à 200$. Sinon, si le montant est supérieur ou égal à 200$, cela né-

cessite une approbation par le superviseur. En cas de refus, l'employé reçoit un avis de rejet par e-mail, sinon, en cas d'acceptation, un virement bancaire qui permet de rembourser le dû de l'employé est effectué.

Ce processus de remboursement des frais de fonctionnement doit se conformer à plusieurs règles expliquées comme suit :

- *R1: si l'employé ne possède pas de compte, un nouveau compte sera créé.*
- *R2: si le montant est inférieur à 200$, alors la dépense doit être approuvée automatiquement.*
- *R3: après que la demande de remboursement soit approuvée, l'argent est transféré au compte bancaire de l'employé; autrement, une notification de rejet est envoyée par e-mail à l'employé.*

Le fichier *« Expense_Reimbursement_Process.pml »* comme détaillé dans le Listing 3.6 représente le modèle Promela complet qui contient la description détaillée du processus de remboursement des frais de fonctionnement.

Listing 3.6 *Le fichier Expense_Reimbursement_Process.pml*

```
chan cS[9] = [1] of {int};
/* RECEIVE EXPENSE REPORT PROMELA PROCESS */
proctype ReceiveExpenseReport() {
  int exist;
  chan cS1[2] = [1] of {int} ;
  cS1[0]= cS[0];/* Send to create expense Account */
  cS1[1]= cS[1];/* Send to review pre-Approval */
  R:
   if
    :: exist=0 /* Account doesn't exist */
    :: exist=1 /* Account exist */
   fi;
  XORSplit(cS1,exist,1); goto R
}
/* CREATE EXPENSE ACCOUNT PROMELA PROCESS */
proctype createExpenseAccount() {
  int x;
  cS[0]?x;/* Receive from Expense Report */
  cS[2]!1;/* Send to Review for Preapproval */
}
/* REVIEW FOR PRE-APPROVAL PROMELA PROCESS */
proctype PreApproval() {
  mtype Amount;
  int x;
  chan cS2[2]=[1] of {int};
  chan cS3[2]=[1] of {int};
  cS2[0]= cS[1];
  cS2[1]= cS[2];
  cS3[0]= cS[3]; /* Send to Auto-approve Expense */
  cS3[1]= cS[4]; /* Send to Approval review by supervisor */
  P: XORJoin(cS2,x);/* Receive from Expense Report or Create Expense Account */
   if
    ::(Amount<200)-> x=0
    ::(Amount>=200)-> x=1
   fi;
  XORSplit(cS3,x,1); goto P
}
/* AUTO-APPROVE EXPENSE ACCOUNT PROMELA PROCESS */
proctype AutoApp() {
  int x;
  A:
   cS[3]?x;/* Receive from review for pre-Approval */
```

```
    cS[5]!1;/* Send to Transfer Money to Employee's Bank */
   goto A:
}
/* APPROVAL REVIEW BY SUPERVISOR PROMELA PROCESS */
proctype AppBSupervisor() {
   int approved, x;
   chan cS3[2]=[1] of {int}
   cS3[0]= cS[6];/* Send to Notify Employee of Rejection */
   cS3[1]= cS[7];/* Send to Transfer Money to Employees Bank */
   B:
    cS[4]?x;/* Receive from Review for pre-Approval */
    if
     ::(approved=1) /* Approved */
     ::(approved=0) /* Rejected */
    fi;
   XORSplit(cS3,approved,1);goto B
}
/* TRANSFER MONEY TO EMPLOYEE'S BANK PROMELA PROCESS */
proctype TransferMoney() {
   int x;
   T:
    XORJoin(cS,x);
    cS[8]!1;/* Send to end process */
   goto T;
}

/* NOTIFY EMPLOYEE OF REJECTION PROMELA PROCESS */
proctype Notification() {
   int x;
   N:
     cS[6]?x; /* Receive from Approval Review by Supervisor * /
     cS[9]!1;/* Send to end process */
   goto T;
}
/* Run processes */
init {
   atomic{
    run ReceiveExpenseReport();run createExpenseAccount();run PreApproval();
    run AutoApp(); run AppBSupervisor(); run TransferMoney();
    run Notification();run end();
  }
}
```

Les figures 3.10, 3.11 et figure 3.12, représentent respectivement la formalisation des règles de conformité R1, R2, R3 exprimées en G-LTL ainsi que les formules LTL correspondantes générées et enregistrées dans le fichier « *Expense_Reimbursement_CRules.prp* ».

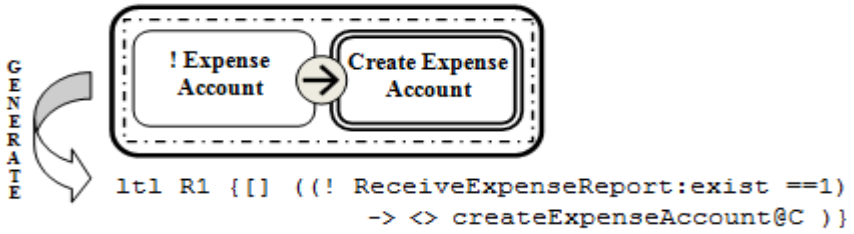


Figure 3.10 *La représentation graphique de la règle «R1»*

Figure 3.11 *La représentation graphique de la règle «R2»*

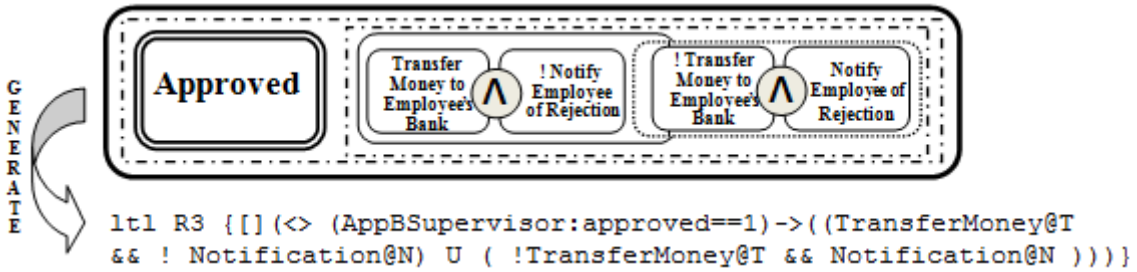


Figure 3.12 *La représentation graphique de la règle «R3»*

Les deux fichiers cités ci-dessus sont fournis comme entrée pour SPIN model-checker, la figure 3.13 représente les résultats de la vérification de ce processus par jSpin.

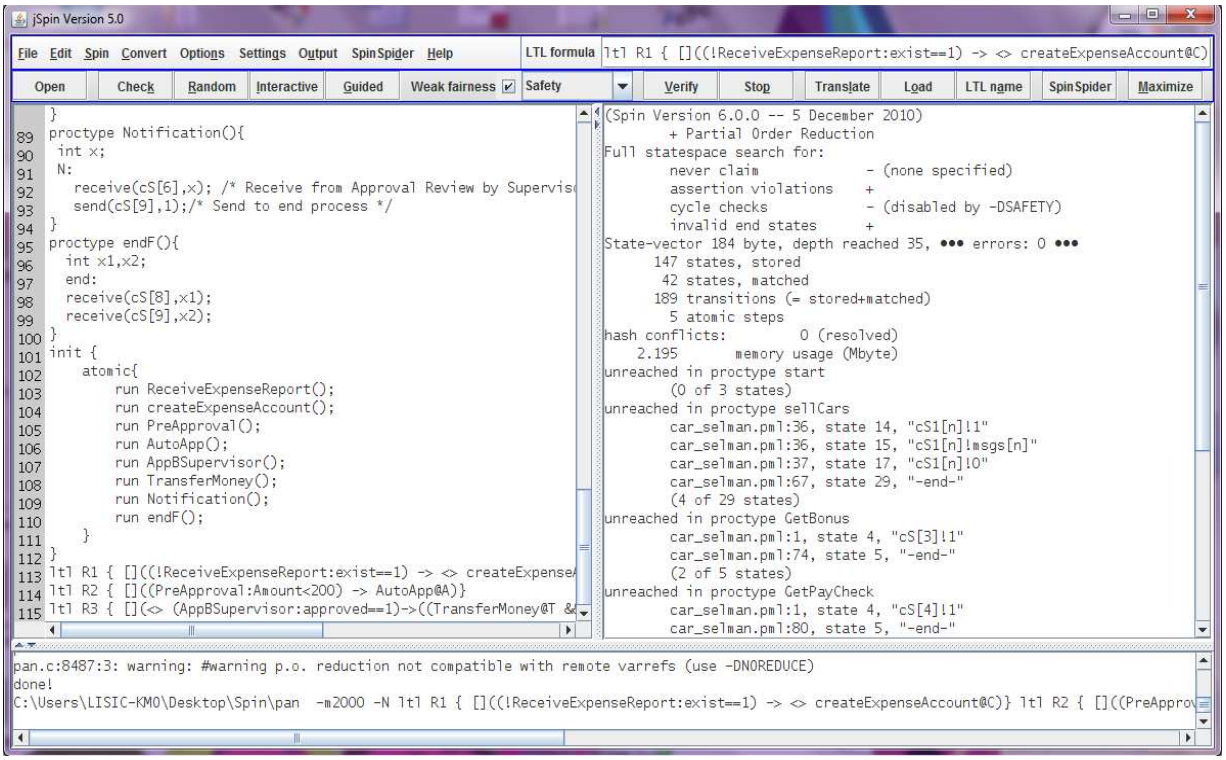


Figure 3.13 *Résultats de la vérification de conformités du processus en jSpin*

3.8 Conclusion

Les modèles de processus métier sont généralement constitués d'un certain nombre d'activités interdépendantes qui sont déployées pour atteindre les objectifs métiers d'une entreprise. L'exactitude de ces modèles de processus métier est essentielle pour le bon

fonctionnement de l'entreprise. Pour cette raison, les éventuelles erreurs telles que les erreurs structurelles dans les spécifications doivent être détectées et corrigées le plus tôt possible, et cela même lors de la phase de conception.

Dans ce chapitre, nous avons présenté une méthode de validation des modèles de processus métier basée sur la technique de model-checking. En effet, nous avons proposé une approche pour la vérification de certaines propriétés de cohérence des modèles de processus métier à l'aide du model-checker SPIN.

Tout d'abord, nous avons décrit comment traduire n'importe quel modèle de processus métier vers une structure de kripke pour exprimer son comportement, nous avons ensuite, étudié certaines propriétés de cohérence des modèles de processus métier et présenté la traduction de ces propriétés en formules LTL et cela afin de permettre au model-checking de vérifier si l'abstraction donnée par cette structure de kripke satisfait ou non les différentes formules et par conséquent de détecter d'éventuelles erreurs structurelles.

Cette approche a été implémentée et validé par un modèle vérifiable par SPIN. Ce modèle, devant être décrit en langage Promela. Les simulations faites sur SPIN en testant les propriétés évoquées sur différents modèles de processus métier ont montré l'efficacité de cette méthode de vérification des modèles. En particulier, la notion du contre-exemple généré par SPIN est d'une grande importance vue qu'elle nous aide à trouver rapidement la cause de l'erreur et par la suite la corriger.

Dans la même démarche, cette approche à été utilisée pour vérifier les règles de conformité qui sont essentielles pour gouverner et contrôler les comportements métiers et l'information au sein d'une organisation. Les modèles de processus métier sont mappés en structures de kripke pour exprimer leur comportement tandis que les règles de conformité sont exprimées à l'aide d'un outil qu'on a proposé appelé G-LTL. Cet outil a la vertu d'offrir un moyen simple pour formuler les règles à vérifier, sans avoir connaissance des différentes notations LTL pour chaque outil de model-checker. Il permet également d'obtenir des règles dans le même niveau d'abstraction que les modèles de processus métier.

Dans le chapitre suivant, nous allons présenter le deuxième axe de notre contribution qui consiste à analyser l'impact du changement sur les modèles de processus métier.

CHAPITRE 4

Analyse *a priori* de l'Impact du changement des modèles de processus métier

4.1 Introduction

Pour mener à bien leurs missions dans un contexte fortement concurrentiel et afin de faire face aux évolutions du marché, les entreprises ont, de plus en plus, besoin de gérer des processus complexes dont elles doivent assurer la fiabilité et de s'adapter aux changements.

L'origine de cette dynamique vient essentiellement des changements fréquents des réglementations extérieures que l'organisation doit respecter et de ceux de la politique intérieure de cette organisation. Cela rend les éléments de processus susceptibles d'évoluer (Goedertier & Vanthienen, 2006). Des changements doivent donc pouvoir affecter un ou plusieurs éléments d'un processus sans altérer la stabilité des autres éléments.

Cependant, du fait des dépendances fortes des diverses entités qui constituent un processus métier, tout changement affectant une entité de ce processus a de fortes chances de provoquer des effets de bord concernant les autres entités. Il est donc nécessaire que tout changement d'un processus métier soit accompagné d'une analyse de son impact, ce qui permettra de mieux mesurer ses effets et surtout ses effets indésirables généralement appelés effets de bord ou *"Side effects"*.

L'un des aspects principaux de l'analyse de l'impact du changement est l'aspect structurel. En d'autres termes, cet impact concerne les effets de vague ou *"Ripple effect"* associé à la modification d'un processus. En effet, toute opération de changement d'un processus qu'elle soit un simple renommage d'une activité ou le remplacement d'un fragment du modèle de processus par un autre peut engendrer des effets qui affectent une grande partie voir la totalité de la structure du processus.

Il est donc primordial de mettre en œuvre des mécanismes d'analyse *a priori* permettant de mieux comprendre et cerner la propagation de l'impact du changement des processus métier afin d'éviter des situations incontrôlées survenant généralement après le changement et de conserver ainsi la cohérence du processus.

La compréhension du processus métier est donc d'une nécessité cruciale pour toute mise en œuvre de changement. La difficulté de compréhension est liée au nombre très important d'interconnexions verticales (inter-couches) et horizontales (intra-couches) entre les différentes entités du processus métier. Le terme couche désigne un niveau d'abstraction donné. Par exemple, une modélisation d'un processus par la notation BPMN ou EPC appartient à une couche plus abstraite que celle associée au déploiement de ce processus en utilisant le langage BPEL.

L'analyse du changement et de la propagation de son impact peut donc nécessiter une description exhaustive des constituants du processus métier et de leurs interdépendances. Elle nécessite également l'existence d'un référentiel exploitable de façon automatique (une base de connaissances) et servant au stockage et à la manipulation des données et des éléments constituant le processus.

Dans ce chapitre, nous présentons avec plus de détails les résultats de nos travaux concernant l'analyse de l'impact du changement des processus métier avec un intérêt particulier pour les relations de dépendance inter-couches et intra-couches. Cela nous a

conduits à la construction d'un ensemble de règles et d'un système à base de connaissances intégrant les concepts de base nécessaires pour l'analyse et la propagation du changement de la structure du processus.

Le chapitre est organisé comme suit : la section 4.2 présente la gestion du changement des processus métier d'une façon générale. La section 4.3 établit une synthèse de ce qui existe dans la littérature à propos de l'analyse de l'impact du changement. La section 4.4 explique le principe de cette analyse et la section 4.5 explique plus particulièrement celui d'analyse de l'impact du changement des processus métier. La section 4.6 montre comment il est possible d'analyser l'impact du changement à travers les différentes relations de dépendance. La propagation de l'impact du changement et sa modélisation sont détaillés dans la section 4.7 et la section 4.8 respectivement. Finalement la section 4.9 conclut le chapitre.

4.2 Gestion du changement des processus métier

La capacité d'adaptation d'un processus métier est une exigence essentielle pour les entreprises afin de faire face à la nature dynamique de leurs environnements. Toutefois, sans contrôle adéquat, les changements des processus métier peuvent engendrer des effets de bord *"Side effects"* qui peuvent conduire à des blocages, des exécutions infinies, des terminaisons multiples (Kherbouche, Ahmad, & Basson, 2012), (Kherbouche, Ahmad, & Basson, 2013) , des conflits sémantiques (Rinderle, Reichert, & Dadam, 2003), une non-conformité avec la réglementation (Awad, Decker, & Weske, 2008), une dégradation de la qualité du service (temps de réponse , la sécurité , la taille du message, etc.) , ou encore une détérioration de la qualité globale du processus métier (Sánchez-González, Ruiz, García, & Piattini, 2011).

À cet égard, les travaux de recherche portant sur la gestion des changements des processus métier continuent à susciter un intérêt croissant de l'industrie et de la communauté scientifique, et cela depuis plus de vingt ans (Rajabi & Lee, 2010), (Reijers, 2006). Cependant, la majorité de ces approches et paradigmes (Weske, 1998), (Casati, Ceri, Pernici, & Pozzi, 1996), (Zhao & Liu, 2013), (Reichert & Dadam, 1998), (Reichert, Rinderle, Kreher, & Dadam, 2005), (Weber, Reichert, & Rinderle-Ma, 2008), (van der Aalst & ter Hofstede, 2005) focalisent sur la gestion des changements dynamiques des processus métier. Autrement dit, la façon dont on peut intégrer les changements avec cohérence sans affecter les instances en cours d'exécution plutôt que de proposer des solutions visant à accroître la prise de conscience des modeleurs et experts métiers suite à un changement dans les processus métier.

En effet, en dépit des travaux novateurs proposés dans la littérature par la communauté *"Business Process"* (BP), et permettant la gestion du changement dynamique des processus métier et hormis quelques tentatives (Chountas, Petrounias, & Kodogiannis, 2003), (Chun, Atluri, & Adam, 2002), (Kim, 2003) qui offrent la possibilité d'analyser les relations de dépendance qui existent au sein d'un workflow à des fins de modélisation, peu d'efforts ont été faits pour fournir des mécanismes permettant de comprendre a priori[18] comment un changement de processus métier pourra avoir un impact sur le reste du sys-

[18] Durant la phase de conception et avant la phase d'exécution des processus métier

tème (Dai & Covvey, 2005). Le processus qui a pour but d'accroître la prise de conscience des modeleurs afin d'éviter les effets de bord (Arnold & Bohner, 1993) et/ou d'estimer les effets de vague (Bohner S. A., 1996) du changement concerné est appelé analyse de l'impact du changement.

4.3 Analyse de l'impact du changement dans la littérature

Le mot « impact du changement » fait référence aux répercussions et aux effets qu'un changement d'une entité du système peut avoir sur les autres entités de ce dernier. L'analyse d'impact du changement est souvent utilisée pour évaluer ces effets après que le changement soit appliqué.

Il existe différentes définitions de l'analyse de l'impact du changement dans la littérature. Parmi ces définitions on peut citer celle du travail de Haney (Haney, 1972) qui porte sur une technique d'analyse de la connexion des différents modules d'un système et qui est souvent désigné comme étant le premier papier qui porte sur l'analyse d'impact. La technique tient à l'idée que pour chaque paire de modules dans un système, il existe une probabilité qu'un changement dans un module nécessite un autre changement conséquent dans l'autre module. La technique a ensuite été utilisée pour la propagation du changement de modèle entre tous les composants du système. Pfleeger et Bohner (Pfleeger & Bohner, 1990), définissent l'analyse de l'impact du changement comme : « *l'évaluation des nombreux risques associés au changement, y compris les estimations des effets sur les ressources, sur les efforts et sur la planification* ». Turver et Munro (Turver & Munro, 1994) définissent l'analyse de l'impact du changement comme « *l'évaluation d'un changement du code-source d'un module, sur les autres modules du système. Il détermine ainsi la portée du changement et fournit une mesure de sa complexité* ». Arnold et Bohner (Arnold & Bohner, 1993) définissent l'analyse de l'impact du changement comme « *un mécanisme qui permet d'identifier les conséquences potentielles d'un changement, ou l'estimation de ce qui doit être modifié pour réaliser un changement* ». Ils mettent l'accent sur l'estimation de l'impact. Dans une définition étendue, Pfleeger prolonge sa définition initiale à l'évaluation des impacts en analysant l'effet de vague d'une modification du code-source d'un système, il définit cela comme étant « *les effets indirects sur les autres parties du système résultant de ce changement* ». Ces effets peuvent être classés en plusieurs catégories telles que les effets sur la logique du système, les effets sur la performance du système mais aussi les effets sur la compréhension du système.

Toutefois, une approche plus récente et proactive utilise l'analyse d'impact afin de prédire les effets du changement en amont c.à.d. avant qu'il ne soit appliqué (Bohner S. A., 2002) pour estimer les coûts et les ressources nécessaires pour appliquer ce dernier.

4.4 Principe de l'analyse de l'impact du changement

Du fait des dépendances fortes des divers éléments d'un processus métier, les changements concernant un de ces éléments ont de fortes chances d'affecter les autres. Il est donc nécessaire de mener une analyse de l'impact de tout changement en identifiant de façon précise les différents éléments qu'il affecte. Une des difficultés majeures pour mener ce genre d'analyse consiste en l'existence d'un phénomène appelé effet de vague ou

"Ripple effect" se manifestant par une propagation des effets de tout changement aussi minime soit-il à un très grand nombre d'entités. Afin d'analyser ces effets, l'analyse de l'impact du changement utilise généralement deux techniques qui sont l'analyse des dépendances et d'analyse de la traçabilité.

L'analyse de dépendance fournit un ensemble d'outils permettant de détecter et d'analyser les différentes relations de dépendance dans un système. Cela peut comprendre les dépendances de données qui représentent les relations entre les instructions de programme qui définissent ou utilisent des données. Autrement dit, une dépendance de données existe quand une déclaration fournit une valeur utilisée par une autre déclaration dans un programme. Les dépendances de contrôle sont les relations entre les états du programme qui contrôlent son exécution, et enfin les dépendances des composants qui se réfèrent aux relations générales entre les composants du code-source, par exemple, packages, modules, fichiers, etc.

L'analyse de traçabilité est généralement un travail qui permet d'établir un lien entre l'environnement du système et sa documentation qui contiennent des niveaux variés d'information sur ce système. Il met l'accent sur les interconnexions entre le code source, les tests, les documents de conception et les exigences, etc.

Cependant, bien que l'analyse de dépendance (Ajila, 1995) , (Deruelle, Bouneffa, Melab, Basson, & Goncalves, 2000), (Fisler, Krishnamurthi, Meyerovich, & Tschantz, 2005), (Ryder & Tip, 2001) et l'analyse de traçabilité (Pfleeger & Bohner, 1990), (Arango, Schoen, & Pettengill, 1993), (Baniassad & Clarke, 2004), (Knethen, 2001), (Marcus & Maletic, 2003) puissent être utilisées séparément, elles peuvent également être utilisées conjointement comme proposé dans (Lindvall & Sandahl, 1998).

4.5 Analyse de l'impact du changement des processus métier

Selon la taxonomie de changement de Regev *et al.* (Regev, Soffer, & Schmidt, 2006), tous les éléments du processus sont susceptibles d'être modifiés. En effet, l'évolution du processus métier peut être vue comme un ou plusieurs changements (par exemple, ajout d'une activité, suppression d'une activité, ou modification d'une contrainte) opérés sur un ou plusieurs éléments du processus et peut concerner tous les aspects du processus.

Les causes de ces changements sont multiples, elles émanent pour la plupart d'une correction d'une ou plusieurs erreurs, d'une exception dans le processus métier, d'une mise en conformité avec de nouvelles normes ou réglementations, d'une évolution du métier et des services de l'entreprise *"innovation"*, ou d'une amélioration des performances et une optimisation des processus existants *"reengineering"*.

Un changement de processus métier peut être décrit formellement comme une différence (notée Δ) entre le schéma du processus métier initial *S* et le schéma du processus métier résultant *S'*, il peut être quantifié comme suit:

$$S' = S + \Delta$$
$$\Delta = | S' - S |$$

La variation (Δ) peut générer différents types d'impact. Ces impacts peuvent être structurels, fonctionnels, comportementaux, logiques et/ou qualitatifs sur une partie ou

sur l'ensemble du modèle de processus métier et ses instances et peuvent être propagés d'une manière horizontale (intra-couche) et verticale (inter-couches). Par conséquent, une analyse *a priori* de cette variante est donc primordiale pour maintenir la cohérence des modèles de processus métier obtenus après ce changement.

Cette analyse de l'impact du changement des processus métier vise à déterminer les éléments du modèle qui sont susceptibles d'être impactés directement ou indirectement par ce changement. Elle peut être résumée dans le méta modèle de la figure 4.1 sous la forme d'un diagramme de classes.

En effet, un changement est considéré comme une entité qui s'applique au niveau du schéma du processus métier, ou au niveau des instances. Un changement cause différents types d'impact et nécessite une analyse qui consiste à les décrire ou à les définir.

Comme tout système, un processus métier est composé de différents types d'éléments ou entités en interaction les uns avec les autres. Autrement dit, les entités sont interdépendantes à travers des relations directes ou indirectes. L'une des relations les plus couramment identifiées comme étant un fil conducteur dans le processus d'analyse de l'impact est la relation générique dite de dépendance. En effet, une telle relation permet de dire : *si une entité B dépend d'une entité A, le changement de A aura un impact sur B*.

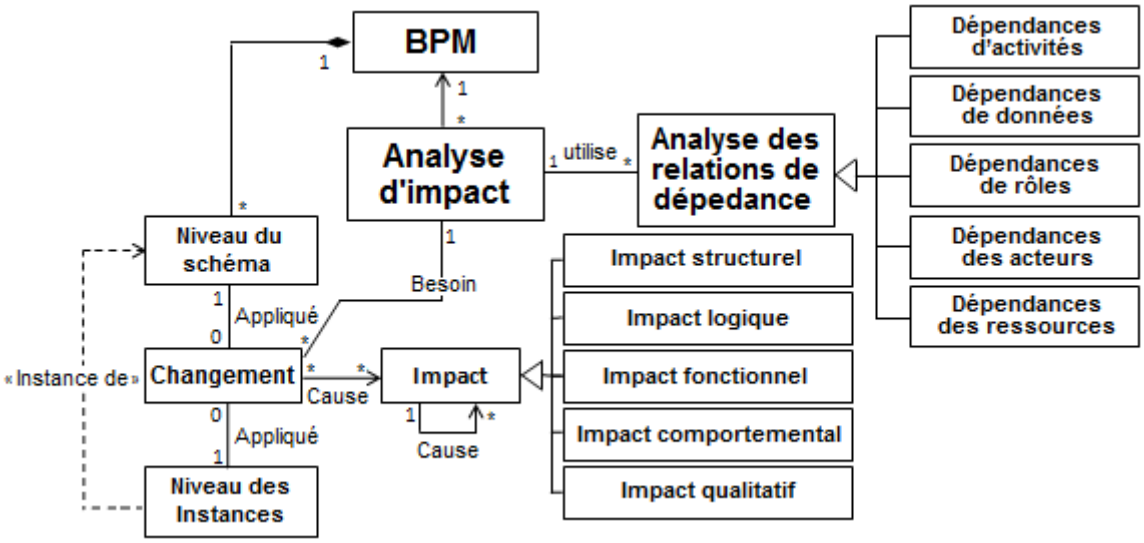


Figure 4.1 *Analyse de l'impact du changement des processus métier*

Notre travail (Kherbouche, Bouneffa, Ahmad, & Basson, 2013), (Kherbouche O. M., Ahmad, Bouneffa, & Basson, 2013) consiste à traiter l'analyse de l'impact du changement des processus métier à travers une approche basée sur une définition de la relation de dépendance tenant compte des spécificités de modélisation de ces processus.

L'approche proposée permet non seulement d'analyser les relations de dépendance qui existent entre la partie changée et les autres parties potentiellement affectées au sein du modèle de processus métier (propagation d'impact horizontal), mais aussi d'analyser les relations entre ces modèles de processus métier et les services associés (propagation d'impact vertical).

4.6 Analyse des relations de dépendance

Outre l'analyse de la traçabilité, l'analyse de dépendance joue un rôle important dans l'analyse d'impact du changement des logiciels comme vu précédemment. Son objectif principal est de capturer les relations de dépendance existantes dans un système à travers différentes techniques tels que des graphes appelés « Graphe des dépendances » *"Dependency graph"* (Rajlich, 1997) ou des « Matrices de Dépendance Structurelle » *"Dependency Structure Matrix (DSM)"* (Yu, Goldberg, Sastry, Lima, & Pelikan, 2009), (Li, 2003) afin d'identifier les entités qui peuvent être potentiellement touchés par un changement.

Dans la même optique, et en se basant sur ce principe, notre approche consiste à traiter l'analyse de l'impact du changement des modèles de processus métier à travers une définition de la relation de dépendance qui est considérée comme le vecteur de la conduction et donc de la propagation de l'impact.

En effet, un processus métier est composé de différents types d'entités qui jouent des rôles différents et qui interagissent les uns avec les autres dans de multiples aspects, que ce soit d'une façon directe ou indirecte. Ces interactions font référence à des relations de dépendance.

Avant d'introduire la notion de dépendance dans les processus métier, nous proposons dans ce qui suit, un scénario représentant un processus de demande de crédits (*cf.* figure 4.2) et cela dans un but d'illustration de nos propos par un exemple. L'exemple est composé de deux processus à savoir, la banque et le client.

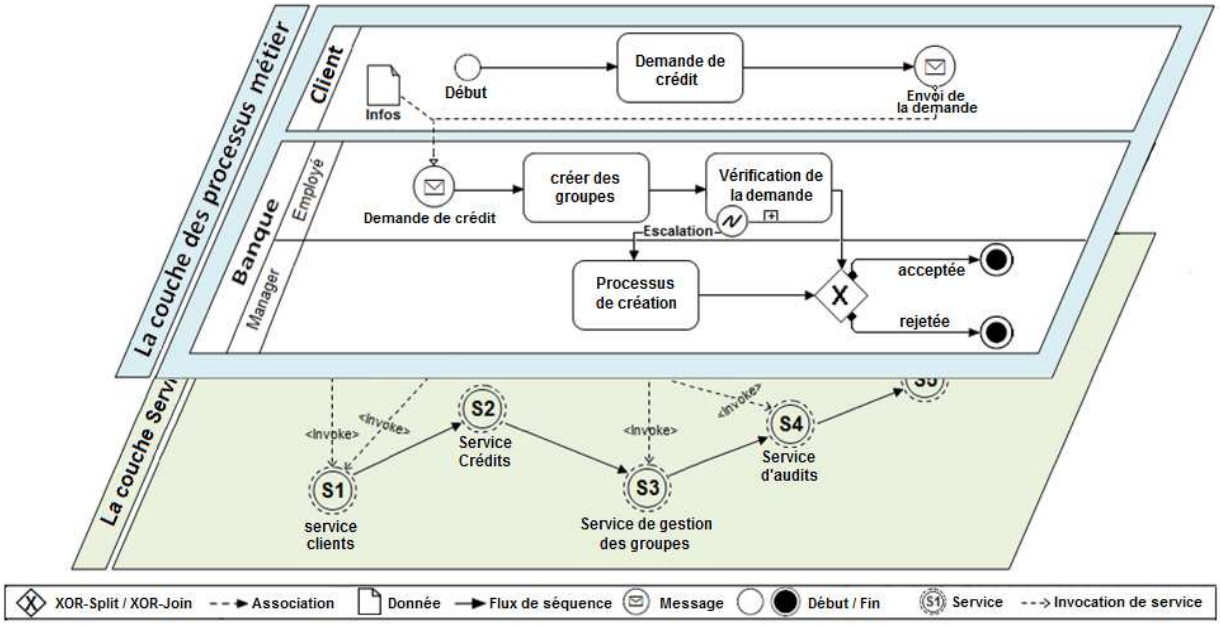


Figure 4.2 *Dépendances multicouches dans les processus métier*

Dans ce scénario, nous pouvons distinguer deux sortes de relations de dépendance dans la couche des processus métier. La dépendance au sein d'un même processus, par exemple le processus client, que nous appellerons intra-processus, et la dépendance entre processus, par exemple, le processus client et le processus banque, que nous appellerons inter-processus.

La relation intra-processus se réfère aux relations existant entre des activités appartenant à un même processus. Par exemple, dans le processus client l'événement « Envoi de la demande de crédits » dépend de l'achèvement de l'activité « Demande de crédits ».

La relation inter-processus est la relation de dépendance entre des processus différents. En d'autres termes, elle désigne une relation de dépendance entre des activités appartenant à des processus différents. Dans notre scénario, cela se manifeste par les échanges de messages entre les deux processus (*cf.* figure 4.2).

Les relations de dépendance que nous avons mises en évidence ci-dessus sont appelées dépendances d'activités ou dépendances de routage. Elles traduisent l'ordre d'exécution des différentes activités d'un processus métier. Ces ordres d'exécution sont définis par les concepteurs et les experts métiers. Ils sont basés sur les exigences techniques, les réglementations juridiques, ou les politiques de gestion. Par exemple, si deux activités sont exécutées de façon séquentielle, cela signifie que l'achèvement de l'exécution de la première activité est une pré-condition pour l'exécution de la seconde.

Il existe également un autre type de relations de dépendance appelée dépendance de données. C'est le cas, par exemple, de la relation existant entre l'activité « Demande de crédits » et l'activité « Créer des groupes ». En effet, cette activité dépend de la seconde du fait qu'elle lui fournit des données concernant le client qui effectue la demande du crédit. Il est aussi intéressant de constater dans certains cas que les dépendances de données ne correspondent pas toujours aux dépendances d'activités. Autrement dit, le flux de données peut ne pas correspondre à l'ordre d'exécution des différentes activités du processus. Il est donc nécessaire d'analyser la dépendance de données séparément de la dépendance d'activités.

Le scénario de demande de crédits (*cf.* figure 4.2) ne montre pas la totalité des dépendances qui peuvent exister entre les différents éléments qui constituent un modèle de processus métier. En effet, il faut également prendre en compte d'autres types de dépendance qui peuvent exister entre les différentes entités formant un modèle de processus métier tels que les rôles, les acteurs, les ressources et les règles, mais aussi les dépendances qui peuvent exister entre les processus et les services associés.

4.6.1 Taxonomie de la dépendance dans les modèles de processus métier

Nous pouvons résumer les quatre types de dépendance qui peuvent jouer un rôle important dans l'analyse de l'impact du changement des processus métier comme suit :

1) La dépendance structurelle: il s'agit de la dépendance syntaxique qui existe entre deux entités du processus métier. Par exemple, un changement appliqué sur une activité dans le modèle de processus métier peut avoir un impact structurel sur les activités adjacentes.
2) La dépendance sémantique: il s'agit de la dépendance sémantique qui existe entre deux entités du processus métier. Par exemple, un changement dans une entité tel qu'une passerelle peut provoquer un changement dans le sens sémantique ou l'interprétation des entités dépendantes.
3) La dépendance directe: elle met en évidence un flux de contrôle F (séquence ou message) tel que $F \subseteq N \times N$ entre deux entités voisines tel que : $\{(n_1, n_2) \in F\}$. Ainsi

un arc (n_1, n_2) représente le fait que n_2 est le successeur direct du nœud n_1 et par conséquent n_1 est le prédécesseur de n_2.

4) La dépendance indirecte: il s'agit d'une dépendance d'une entité envers une autre par transitivité. Il met en évidence un ensemble de flux de contrôle intermédiaire qui peut exister entre deux entités. C-à-d s'il existe un ensemble de relations de dépendance directes $\mathcal{R}$ entre les activités $\{a_1, a_2,\ldots, a_n\} \in A$ tel que : (a_1 $\mathcal{R}$ $a_2,\ldots, a_{n-1}$ $\mathcal{R}$ a_n alors a_1 $\mathcal{R}$ a_n).

4.6.2 Modèle de dépendance pour l'étude d'impact

Nous proposons dans notre travail, un modèle multidimensionnel de dépendance qui comprend des relations de dépendance intra-couches qui sont : la dépendance d'activités, la dépendance de données et la dépendance de rôles, mais aussi les relations de dépendance inter-couches c.à.d. entre la couche des processus métier et de la couche des services (*cf.* figure 4.2).

Notre objectif est de construire le modèle le plus exhaustif possible pour l'analyse *a priori* de l'impact du changement des processus métier et d'établir ainsi une base extensible par la prise en compte progressive d'autres dimensions telles que les acteurs, les règles et les ressources, etc.

4.6.2.1 Relations de dépendance intra-couches

Les relations de dépendance intra-couches comprennent les points suivants:

4.6.2.1.1 Dépendances d'activités (routage)

Comme évoqué précédemment, il existe deux types de dépendances d'activités ou de routage et qui sont : l'inter-processus et l'intra-processus. Ce type de dépendance définit l'ordre d'exécution des activités dans un processus et cela par le biais des flux de contrôle (flux de séquences *"Sequence flows"* et flux de messages *"Message flows"*). Ces dépendances définissent non seulement l'ordre d'exécution mais également une certaine sémantique associée à cet ordre. Ainsi, pour une passerelle *AND-Join* reliant deux activités *A* et *B* à une activité *C*, l'activité *C* ne peut être exécutée que si *A* et *B* ont achevé leurs exécutions qui se déroulent en parallèle.

Formalisation de la dépendance d'activités :

Une dépendance d'activités peut être définie comme suit :

$$\mathcal{D}_A = (\mathcal{D}_P, \Omega)$$

Sur un ensemble d'activités A = $\{a_1, \ldots, a_n\}$ et un ensemble de flux de contrôle T= $\{t_1,.., t_n\}$.

$$\mathcal{D}_P(a) = \mathcal{D}_P i\ (a) \cup \mathcal{D}_P o\ (a) \text{ avec } a \in A$$

$\mathcal{D}_P$o (a) représente l'ensemble des activités $a_i \in$ A qui succèdent à l'activité a (on notera cela par $a_i \rightarrow a$) et dont l'exécution dépend *a fortiori* de *a*. C'est une relation *"one-to-many"*, où plusieurs activités dépendent d'une seule autre activité. De la même manière, $\mathcal{D}_P$i (a) représente l'ensemble des activités $a_i \in$ A qui précèdent l'activité a (on notera cela par $a \rightarrow a_i$). En d'autres termes l'exécution de *a* dépend de celles des activités apparte-

nant à $\mathcal{D}_{pi}(a)$. C'est une relation *"many-to-one"*, où une activité dépend de plusieurs autres activités.

$$\Omega = \Omega i \cup \Omega o$$

Ωi est l'ensemble des flux de contrôle $t_i \in T$ reliant chaque activité de $\mathcal{D}_{pi}(a)$ à a. Autrement dit, l'ensemble des arcs entrants $(\mathcal{D}_{pi}(a), a)$ de a. Ωo est l'ensemble des flux de contrôle $t_i \in T$ reliant l'activité a à toute activité appartenant à $\mathcal{D}_{po}(a)$. Ce sont donc les arcs sortants $(\mathcal{D}_{po}(a), a)$ de a.

4.6.2.1.2 Dépendances de données

La dépendance de données met en avant les ressources communes ou les données liées à plusieurs activités. Il existe trois grands types de dépendance de données (Malone, et al., 1999): la dépendance par flux ou *"Flow dependency"*, la dépendance par partage ou *"Sharing dependency"* et la dépendance par mutualisation ou *"Fit dependency"*.

Une dépendance par flux existe si une activité produit une ressource ou donnée, utilisée par une autre activité. Une dépendance par partage représente le fait que plusieurs activités utilisent ou partagent la même ressource ou donnée, et enfin une dépendance par mutualisation existe si plusieurs activités produisent collectivement une ressource ou donnée.

Formalisation de la dépendance de données :

On définira l'ensemble des données D échangées entre les activités (Reichert & Dadam, 1998) par :

$$D = \{d_1, d_2, \ldots, d_n\}$$

Chaque activité $a_i \in A$ possède des paramètres d'entrée et des paramètres de sortie, notés respectivement $InPARs(a)$ et $OutPARs(a)$. On définira une connexion de donnée notée dc par:

$$dc = \{d, a, pars, mode\}$$

Où, $d \in D$, $a_i \in A$, $pars \in InPARS(a) \cup OutPARs(a)$, et $mode \in \{lecture, écriture\}$.

Cela signifie que l'activité $a_i \in A$ utilise (en lecture ou en écriture) une donnée $d \in D$ fournie comme paramètre d'entrée ou/et de sortie.

Une activité $a_i \in A$ dépend d'une autre activité $a_j \in A$ en termes de données, noté par la relation $a_j \xrightarrow{(D)} a_i$ *ssi*

$$\exists\, dc_x, dc_y \in DC$$

Où, DC= $\{dc_1, dc_2, \ldots, dc_n\}$ est l'ensemble de toutes les connexions de données tel que :

$$dc_x = \{d, a_j, pars, écriture\}$$

$$dc_y = \{d, a_i, part, lecture\}$$

Où, $d \in D$, $part \in InPARs\ (a_i)$, $pars \in OutPARs\ (a_j)$ et a_j précède a_i dans le schéma du processus métier.

4.6.2.1.3 Dépendances de rôles

Dans (Kim, 2003), l'auteur définit des relations de dépendance entre les rôles en remplaçant l'activité par le rôle associé à cette dernière. Autrement dit, le diagramme d'activités devient un diagramme de rôles. Cependant, ce diagramme de rôles est basé sur la structure hiérarchique de l'organisation (Dai & Covvey, 2005).

La relation de dépendance entre les rôles est définie de la manière suivante :

Si un rôle *A* est affecté à la même activité a exécutée par un autre rôle *B*, alors nous pouvons conclure que le rôle *B* à une relation de dépendance avec le rôle *A*. Par exemple, l'activité *« Prise de radio »* dans un processus de l'examen d'imagerie médicale peut être effectuée par un « technicien » ou un « radiologue ». C'est-à-dire, le *rôle (Technicien, prend une radio, l'examen d'imagerie médicale)* et le *rôle (Radiologue, prend une radio, l'examen d'imagerie médicale)* sont dépendants l'un de l'autre.

Formalisation de la dépendance de rôles :

Soit $R= \{r_1, \ldots, r_n\}$ un ensemble de rôles et $A= \{a_1, \ldots, a_n\}$ un ensemble d'activités. Une dépendance de rôles peut être définie comme suit :

$$\mathfrak{D}_R = (\sigma, \Psi) \text{ Où: } \sigma(r)= \sigma i(r) \cup \sigma o(r) \text{ avec } r \in R.$$

$\sigma o(r)$ représente l'ensemble des rôles qui sont des successeurs immédiats à *r*, c'est-à-dire affectés à des activités $\alpha i \in A$ qui succèdent à l'activité *a* à laquelle *r* est associé. $\sigma i(r)$ représente l'ensemble des rôles prédécesseurs immédiats de *r*. En d'autres termes les rôles affectés aux activités $\alpha i \in A$ qui précèdent l'activité *a* et a laquelle *r* est associé.

$$\Psi = \Psi_i \cup \Psi_o$$

Ψ_i est l'ensemble des flux de contrôle $t_i \in T$ ou arcs reliant chaque rôle de $\sigma i(r)$ à *r*. Autrement dit, l'ensemble des arcs entrants de *r*. Ψ_o est l'ensemble des flux de contrôle $t_i \in T$, reliant le rôle *r* à tout rôle appartenant à Ψ_o. Ce sont donc les arcs sortant de *r*.

4.6.2.2 Relations de dépendance inter-couches

Comme la couche des processus métier, la couche des services se compose d'un ensemble de services qui interagissent les uns avec les autres par des relations directes ou indirectes et qui sont également en constante évolution (Wang & Capretz, 2009). Cependant, il existe à côté de cela, une relation de couplage entre les services et les processus métier, comme indiqué dans la figure 4.2, que nous avons appelé une relation de dépendance inter-couche.

En effet, la corrélation, les entrées et les sorties des services web sont liées par l'orchestration de ces processus à travers des normes tels que WS-BPEL. Par conséquent, un processus métier peut faire appel à plusieurs services. Cela signifie que chaque fois qu'un changement se produit dans le processus métier, ce dernier peut affecter les services qui lui sont associés et *vice-versa*. Ceci peut être justifié par le fait qu'une activité dans le modèle de processus métier peut interagir avec un service d'une façon unidirectionnelle ou bidirectionnelle *via* l'invocation de l'un de ses opérations.

Un service est généralement défini comme un ensemble d'opérations $O = \{o_1, \ldots, o_n\}$ où chaque opération $o_i \in O$ est associée à un ensemble de messages. $T \subseteq S \times S$ représente

l'ensemble des relations entre ces opérations. Chaque transition $t = (o_i, o_j) \in T$ $(o_i, o_j \in O)$ désigne l'invocation de l'opération o_j par l'opération o_i.

Nous pouvons définir formellement la relation de dépendance entre la couche des processus métier et la couche de service par : $d_{ic} = (A_s, r)$ sur un ensemble d'activités $A = \{a_1 \ldots a_n\}$ et un ensemble de services web $S = \{s_1 \ldots s_n\}$ où:

A_s représente l'ensemble d'activités qui fait appel au même service $s_i \in S$ (noté comme suit: $a_i \rightsquigarrow s_i$) et $r \subseteq A \times S$ est l'ensemble des arcs reliant les nœuds qui représentent la relation de dépendance entre chaque activité $a_i \in A_S$ et le service invoqué $s_i \in S$.

4.7 Propagation de l'impact du changement

La propagation de l'impact du changement connu aussi dans la littérature sous le nom de l'effet de vague consiste à exécuter ou simuler (dans le cas de l'analyse *a priori*) le changement initial puis identifier les incohérences ou les inconsistances que ce changement initial peut induire et définir en conséquence les éléments à corriger par de nouveaux changements correcteurs dérivés ou conséquents. L'analyse de cette propagation de l'impact du changement utilise un ensemble de méthodes algorithmiques, permettant de mesurer l'impact de ce changement et son étendue.

Dans notre cas, il s'agit principalement de déterminer, *a priori*, les effets directs et indirects d'une opération de changement sur l'ensemble des éléments d'un processus métier et cela, avec une certaine nuance sur le degré d'impact de chaque élément susceptible d'être touché par ce changement.

En effet, l'analyse de la propagation de l'impact du changement ne doit pas se limiter à savoir combien et quelles autres parties du processus peuvent être affectées par ce changement de processus métier, mais à connaître où l'impact du changement doit commencer et où il doit se terminer. Autrement dit, calculer la profondeur de la propagation de l'impact tout en faisant la distinction entre les entités impactées en termes de degré ou de puissance d'impact.

Pour faire face à ce problème, nous avons proposé dans (Kherbouche O. M., Ahmad, Bouneffa, & Basson, 2013) d'assigner des poids numériques (IWF) aux différents types de relations de dépendances selon leur importance dans la propagation de l'impact du changement. Ainsi, on pourra spécifier qu'un type de relation est plus critique et donc plus important à analyser lors du processus de propagation de l'impact.

Le poids IWF que nous avons appelé facteur de pondération ou *"Impact Weight Factor"* en anglais représente une valeur numérique qui permet d'aidé à mesurer la profondeur de l'impact du changement. Elle est calculée sur la base d'un ensemble de métriques, comme suit:

$$IWF\ (a_i) = \Sigma\ (P\ (a_i) + E\ (a_i) + F\ (a_i) + ND\ (a_i))\ /TDR$$

TDR ou *"Total Dependency Relationships"* représente le nombre total des relations de dépendance au sein du processus métier et aussi entre ce dernier et les services associés.

4.7.1 La métrique P

La métrique *P*, représente la priorité d'une activité $a_i \in A$ (*High*, *Medium* ou *Low*) dans le flux d'exécution d'un processus métier, qui est définie généralement par les modeleurs durant la phase de conception. Elle peut être calculée en se basant sur les valeurs qui correspondent aux priorités assignées comme par exemple : si une activité *A* doit s'exécuter avec une priorité *Haute* alors *P (A) = 0,75 (75/100)*.

4.7.2 La métrique E

La métrique *E*, représente la fréquence d'exécution d'une activité $a_i \in A$ dans chaque chemin d'exécution π d'une instance I_x avec x ∈ {1 ... n} du processus métier *p*. Elle peut être calculée mathématiquement à partir des journaux d'exécution *"Process events logs"* comme suivant :

$$E(a_i) = \sum_{I=0}^{n} \left(\sum_{\pi=0}^{n} ex(a_i)\right)$$

En effet, les journaux d'exécution d'un processus métier peuvent être exploités pour découvrir le nombre de fois où chaque activité a été exécutée $ex(a_i)$ dans chaque chemin d'exécution *π* d'une instance *I*. Par exemple, si nous observons que le chemin d'exécution qui mène à l'activité *A* et *B* a été exécuté plutôt que celui menant à l'activité *C* dans les différentes instances, cela implique que la valeur de *E(A)* et *E(B)* doit être supérieur à celle de *E(C)*.

4.7.3 La métrique F

Cette métrique se réfère aux fréquences d'invocation notée *F* d'une activité $a_i \in A$ dans un modèle de processus métier *p*. Par exemple, si l'activité *B* est appelée dans chaque chemin d'exécution *π* possible de l'activité *A*, alors la probabilité que *A* soit impactés par un changement dans *B*, devient élevée. Toutefois, si *B* est appelée dans un chemin d'exécution parmi d'autres possibles de l'activité *A*, alors la probabilité de *A* étant impactée par un changement dans *B*, devient beaucoup plus faible.

4.7.4 La métrique ND

La profondeur imbriquée *ND* est une métrique qui représente la position de chaque activité dans le chemin d'exécution d'un processus métier. En d'autres termes, l'activité qui s'exécute en amont dans le chemin d'exécution du processus métier reçoit un coefficient plus élevé, tandis que, l'activité qui se produit plus tard dans le chemin de processus métier reçoit un coefficient moins élevé.

Le degré de l'impact de chaque élément affecté par un changement peut être calculé alors comme la somme de tous les IWF menant de l'entité changée à l'entité susceptible d'être touchée. Par conséquent, nous disons qu'un chemin peut être marqué en rouge lorsque le degré de l'impact est élevé, par exemple entre 15 et 20. On dit qu'un chemin peut être marqué comme orange lorsque le degré de l'impact est moyen, par exemple entre 10 et 15 et que le chemin peut être marqué comme vert lorsque le degré de l'impact est faible, par exemple entre 1 et 10. Ces valeurs de seuils peuvent être définies par les modeleurs et les experts métiers on se basant sur des études empiriques.

4.8 Modélisation de la propagation de l'impact du changement

Les résultats majeurs de recherche de notre approche se manifestent par deux principaux constituants qui sont (i) une base de connaissances obtenue à partir de la définition d'une ontologie[19] BPMN étendue décrivant la sémantique de l'ensemble des éléments d'un processus métier et leurs interdépendances et (ii) une base de règles incarnant des règles génériques dédiées à l'étude a priori de la faisabilité d'un changement à un instant T, mais aussi à la compréhension et l'analyse du changement du processus métier et de son impact.

4.8.1 La base de connaissances

La mise en œuvre effective de l'analyse de l'impact du changement nécessite une description exhaustive des différents éléments d'un processus métier et leurs interdépendances. Ainsi, l'utilisation des bases de connaissances *"Knowledge-Based"* nous paraît être le meilleur moyen qui peut aider dans la compréhension, à la fois du processus métier et de son évolution. De telles bases sont utilisées depuis plusieurs années dans le cadre de la mise en œuvre automatiquement assistée de différentes activités du génie logiciel (Abd-El-Hafiz, 1996).

Pour faire face sémantiquement à l'analyse de l'impact du changement et par conséquent, à la compréhension des différentes relations de dépendance au sein d'un processus métier, nous avons proposé dans (Kherbouche M. O., Ahmad, Bouneffa, & Basson, 2013) une approche consistant à mettre en œuvre un système à base de connaissances obtenu à partir de la définition d'une ontologie étendue appelé *"Extended BPMN 2.0 Ontology"* pour le stockage et la manipulation des méta-informations concernant les processus métier.

En effet, notre définition se base sur une ontologie proposée par (Ghidini, Rospocher, & Serafini, 2008) qui est utilisée comme une base de connaissances permettant de représenter formellement les attributs et les conditions décrivant la manière dont les éléments d'un processus métier peuvent être combinés pour obtenir des modèles BPMN conformes à la spécification.

L'approche proposée établit des liens entre les différents éléments du modèle de processus métier et les concepts de l'ontologie basée sur les spécifications de BPMN 1.1[20]. Elle exprime chaque élément du modèle BPMN en tant qu'une classe dans l'ontologie et ses attributs en tant que des attributs de cette dernière. En effet, une ontologie est composée de classes et de relations. Une classe représente un concept identifiable. Une classe peut avoir des liens structurels et/ou sémantiques avec d'autres classes.

L'ontologie proposée se compose actuellement de 95 classes, 108 objets de propriétés, 70 données de propriétés et 439 axiomes de classe. Les éléments sont divisés en deux catégories représentant des éléments de support *"Supporting Elements"* et des éléments graphiques *"Graphical Elements"* qui sont des sous-classes de la classe principale *"BPMN_ELEMENT"* comme le montre la figure 4.3.

[19] Une ontologie, en informatique, est un ensemble structuré de savoirs dans un domaine de connaissance particulier
[20] www.omg.org/spec/BPMN/1.1/PDF.

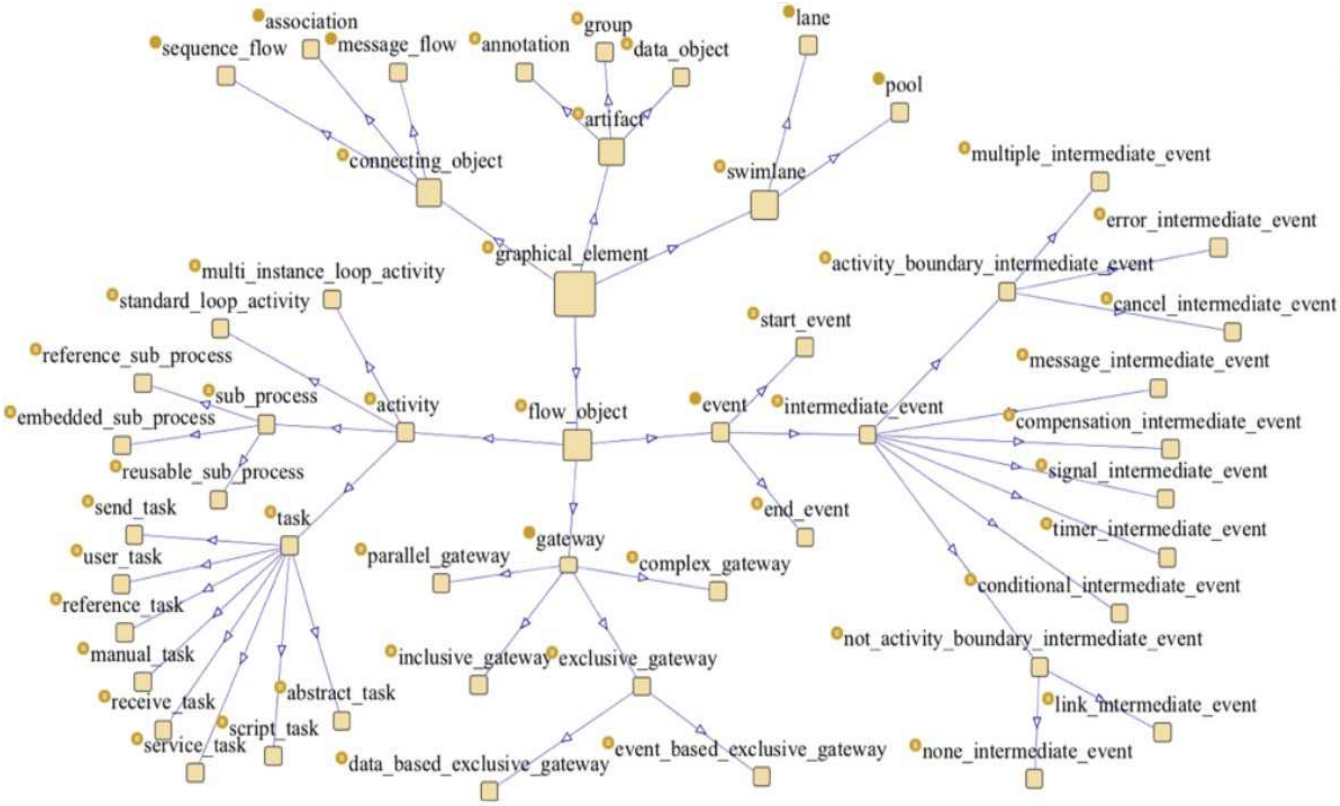


Figure 4.3 *Hiérarchie des classes de BPMN 2 Ontology*

Dans notre ontologie étendue, nous avons redéfini la classe *"BPMN_ELEMENT"* en ajoutant une troisième classe appelée analyse d'impact *"Impact_Analysis"* au même niveau que la classe des éléments de support et la classe des éléments graphiques. Cette classe est affinée par la suite en plusieurs relations de dépendance comme détaillée ci-dessous (dépendances d'activités *"ACTIVITY_DEPENDENCY"*, dépendances de données *"DATA_DEPENDENCY"*).

Afin d'illustrer les différents axiomes de notre ontologie étendue dans leur forme logique, nous avons utilisé la logique de description (DL)[21] sur laquelle se base le langage d'ontologie Web (OWL[22]) qui est un dialecte XML qui étend RDFS et qui est utilisé pour modéliser des ontologies. La redéfinition de la classe *"BPMN_ELEMENT"* est détaillée dans le Listing 4.1.

La définition de la classe *"Impact_Analysis"*, la classe "DEPENDENCY_RELATIONSHIPS", la classe "ACTIVITY_DEPENDENCY" et la classe "DATA_DEPENDENCY" sont détaillées respectivement dans les Listing 4.2, Listing 4.3, Listing 4.4 et Listing 4.5.

Listing 4.1 *Classe BPMN_element.*

```
Class: BPMN_ELEMENT
Label: BPMN element
Description: Base element
BPMN_ELEMENT ≡ GRAPHICAL_ELEMENT ⊔ SUPPORTING_ELEMENT ⊔ IMPACT_ANALYSIS
```

[21] http://www.cs.ox.ac.uk/ian.horrocks/Seminars/download/Horrocks_Ian_pt1.pdf
[22] http://www.w3.org/standards/techs/owl#w3c_all

```
GRAPHICAL_ELEMENT ⊑ ¬SUPPORTING_ELEMENT
GRAPHICAL_ELEMENT ⊑ ¬IMPACT_ANALYSIS
SUPPORTING_ELEMENT ⊑ ¬IMPACT_ANALYSIS
```

Listing 4.2 *Classe IMPACT_ANALYSIS.*

```
Class: IMPACT_ANALYSIS
Label: Impact analysis
Description: Impact analysis
IMPACT_ANALYSIS ≡ DEPENDENCY_RELATIONSHIPS
```

Listing 4.3 *Classe DEPENDENCY_RELATIONSHIPS.*

```
Class: DEPENDENCY_RELATIONSHIPS
Label: Dependency analysis
Description: Dependency relationship defines the existing dependencies concerning
             activities and data
DEPENDENCY_RELATIONSHIPS ≡ ACTIVITY_DEPENDENCY ⊔ DATA_DEPENDENCY
```

Listing 4.4 *Classe ACTIVITY_DEPENDENCY.*

```
Class: ACTIVITY_DEPENDENCY
Label: Impact analysis
Description: Activity Dependency
ACTIVITY_DEPENDENCY ⊑ CONNECTING_OBJECT
ACTIVITY_DEPENDENCY ⊑ (≥ 1) has_connecting_activity_name
Property: has_connecting_activity_name
Label: Name
Description:Name is an attribute that is text description of the concerned acivi-
ty.
has_connecting_activity_name has domain ACTIVITY
has_connecting_activity_name has range xsd:string
ACTIVITY_DEPENDENCY ⊑ (≥ 1)has_connecting_activity_source_ref.ACTIVITY ⊔ (≥ 1)
has_connecting_activity_target_ref.ACTIVITY
Property: has_connecting_activity_source_ref
Label: aSourceRef
Description: aSourceRef is an attribute that identifies which Activity the
             Connecting Object is connected from.
has_connecting_activity_source_ref has domain CONNECTING_OBJECT
has_connecting_activity_source_ref has range ACTIVITY
Property: has_connecting_activity_target_ref
Label: aTargetRef
Description: aTargetRef is an attribute that identifies which Activity the
             Connecting Object is connected to.
has_connecting_activity_target_ref has domain CONNECTING_OBJECT
has_connecting_activity_target_ref has range ACTIVITY
```

Listing 4.5 *Classe DATA_DEPENDENCY.*

```
Class: DATA_DEPENDENCY
Label: Data Dependency
Description: Data Dependency
DATA_DEPENDENCY ⊑ CONNECTING_OBJECT
DATA_DEPENDENCY ⊑ (≥ 1)has_activity_input_sets.ACTIVITY ⊔ (≥ 1)
has_activity_output_sets.ACTIVITY
has_activity_input_sets.ACTIVITY ≡ (DATA_OBJECT)
```

Pour valider cette partie du notre travail, nous avons utilisé une des plate-formes les plus populaires appelé Protégé-OWL basé sur le langage d'ontologie Web (OWL). Cette plate-forme est à la fois un éditeur d'ontologies, et un framework de base de connaissances très convivial, basé sur Java. La figure 4.4 montre la hiérarchie des classes (telles qu'il apparaît dans l'outil Protégé) pour notre ontologie BPMN étendue, et la figure 4.5 représente le fichier OWL associé *"ExtendedOntoBPMN.owl"*.

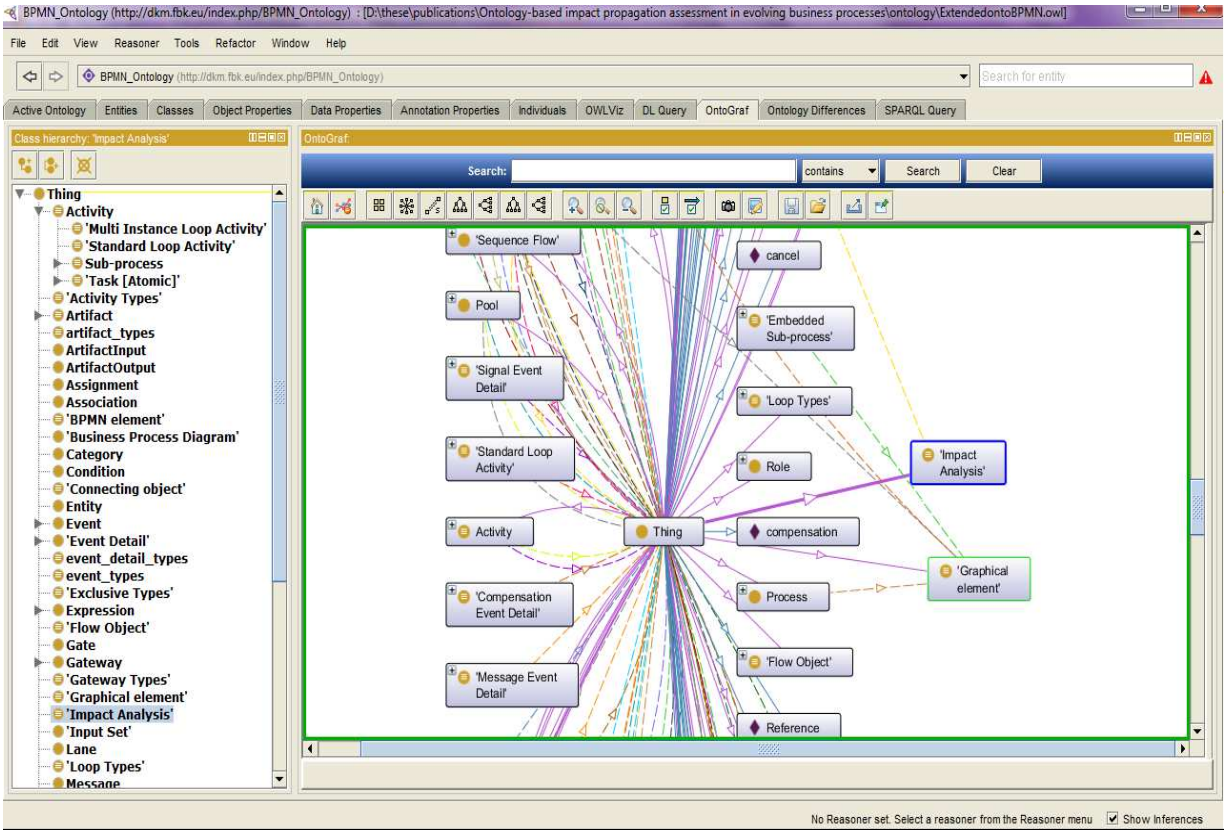


Figure 4.4 *Hiérarchie des classes de extended BPMN 2 Ontology*

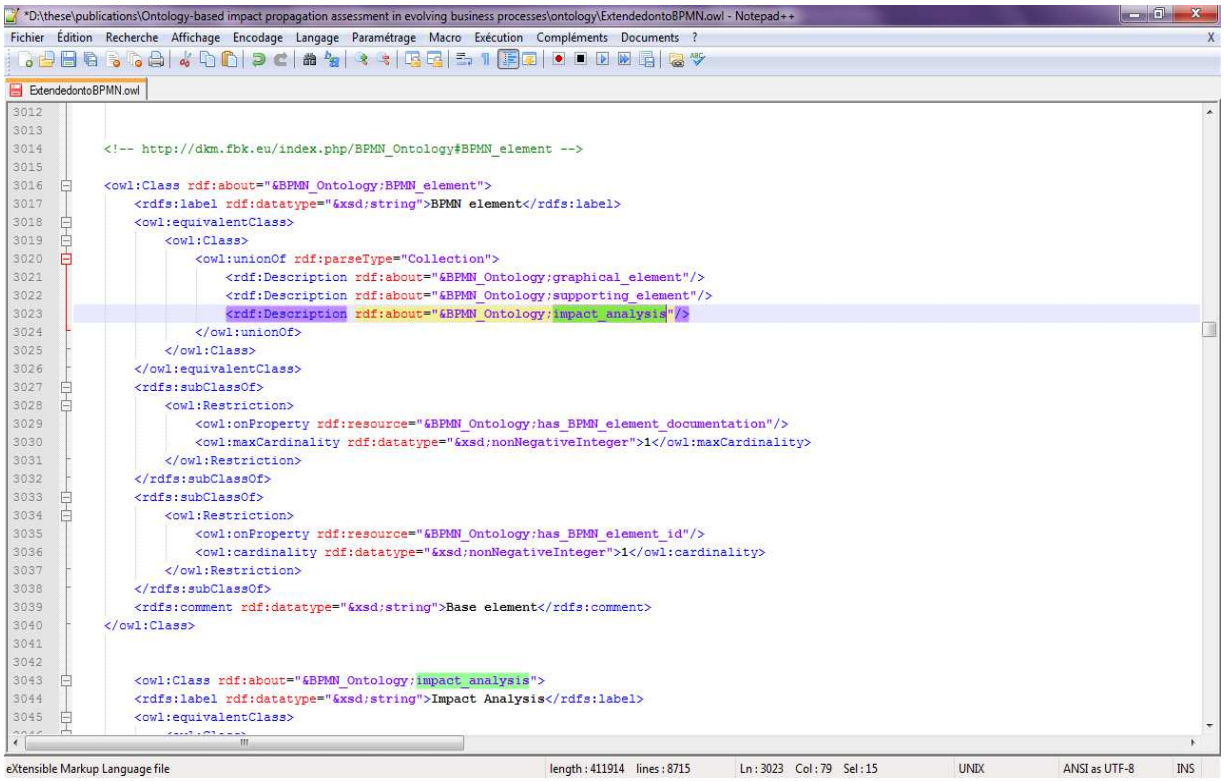


Figure 4.5 *Le fichier ExtendedOntoBPMN.owl*

L'objectif principal étant de construire une base de connaissances à partir de cette ontologie pour permettre au chargé de l'évolution des processus métier, qui est généralement le modeleur et l'expert métier, de formuler ses requêtes de changement.

Pour ce faire, le moteur qui répond à la demande du changement utilise le langage de requête d'ontologie SPARQL[23] pour analyser les relations de dépendance en déclenchant toutes les règles d'analyse de dépendance concernées expliquées dans la section suivante. Ces relations de dépendance sont ensuite utilisées comme un moyen pour suivre la propagation de l'impact du changement.

4.8.2 Le système à base de règles

Le système à base de règles qu'on à mis en place assure deux principales fonctionnalités. La première fonctionnalité est matérialisée par toutes les règles permettant d'analyser la faisabilité *a priori* d'un changement de processus métier à un instant T. L'autre fonctionnalité est encapsulée par les règles traduisant la propagation de l'impact par l'identification des différentes entités et relations directement ou indirectement affectées par une opération de changement de processus métier.

Notre approche de mise en œuvre de ces règles est basée sur le formalisme Evénement-Condition-Action ou ECA. Partant de cette approche, nous avons défini un ensemble de règles pour étudier en premier lieu, la faisabilité d'un changement à un instant T, et en second lieu, un autre ensemble de règles permettant l'analyse *a priori* de l'impact du changement du processus métier.

Afin de fournir une approche générique, nous avons proposé un raisonnement de haut niveau à l'aide d'une notation abstraite. Autrement dit, nous ne tenons pas compte d'une notation particulière pour la modélisation des processus métier, mais nous prenons plutôt en compte les éléments de base d'un processus.

4.8.2.1 Définition 1 (processus métier)

Un modèle de processus métier est représenté sous la forme d'un graphe orienté, où les flux d'objet représentent l'ensemble des nœuds et les flux de contrôle représentent l'ensemble des arcs. Il est représenté par le tuple *Gp = (N, Type, F, S, D, Status)* où:

- N = {n_1, n_2, … n_m} représente l'ensemble fini des nœuds. Les nœuds représentent l'ensemble des objets de flux ou *"Flow objects"* qui peuvent être partitionnés en trois catégories : les événements *"Events"* (E), les activités *"Activities"* (A) et les passerelles *"Gateways"* (G) ;
- Type: N → {Activity, Event, AND-Join/Split, XOR-Join/Split, OR-Join/Split} est une fonction qui assigne à chaque nœud un type;
- F ⊆ N × N (arcs du graphe) représente les flux de contrôle où *"Connecting Objects"* entre les différents nœuds tel que : {(n, n') ∈ F}. Ainsi un arc (n, n') représente le fait que le nœud n' est le successeur direct du nœud n et par conséquent n est le prédécesseur de n';
- S est l'ensemble des services invoqués par le processus ;
- D représente l'ensemble des données ;

[23] http://jena.apache.org/tutorials/sparql.html

- Status → {init, waiting, added, deleted, modified} représente le statut courant de chaque nœud $n \in N$ dans le graphe *Gp* (traces de changement).

4.8.2.2 Définition 2 (instance d'un processus métier)

L'état d'une instance de processus métier est l'un des principaux critères pour déterminer si un changement structurel spécifique peut être appliqué à un instant T sans perturber le fonctionnement des instances encours d'exécutions. Par conséquent, nous pouvons définir une instance de processus métier *I* par le tuple *I = (S, NS, Inst, Stat, Status)* où:

- S = (N, Type, F, Status) est le schéma de processus métier correspondant ;
- NS →{NOT_ACTIVATED, ACTIVATED, RUNNING, COMPLETED, SKIPPED} est l'ensemble des états d'une instance d'un nœud $n \in N$;
- Inst: N → I est une fonction qui associe chaque nœud $n \in N$ avec son instance ;
- Stat: I → S est une fonction qui définit la relation entre une instance d'un nœud $n \in N$ et son état ;
- Status → {init, waiting, added, deleted, modified} représente le statut courant de chaque instance d'un nœud $n \in N$ dans le graphe *Gp* (traces de changement).

4.8.2.3 Définition 3 (activité)

Une activité $a_i \in A$ est définie par un le triplet *A= (Input, Output, s)* où $Input \subset D$ représente l'ensemble des entrées de a_i, $Output \subset D$ représente l'ensemble des sorties de a_i et $s \in S$ représente le service invoqué par a_i.

4.8.3 Définition des règles de "faisabilité du changement"

Pour garantir la cohérence et la conformité de l'ensemble des instances de processus métier avec le schéma de processus modifié, comme expliqué dans le chapitre 2, nous avons défini un ensemble de règles de faisabilité (FR) pour évaluer *a priori* la faisabilité d'un changement de processus métier à un instant T. L'ensemble de ces règles devrait respecter quelques rudiments décrit ci-dessous :

- Afin d'éviter l'insertion d'une nouvelle tâche *A* comme un prédécesseur d'une autre tâche *B* en cours d'exécution *"RUNNING"* ou achevé *"COMPLETED"*, nous devons nous assurer que l'ensemble des éléments qui succèdent à *A* soit en état non-activé *"NOT_ACTIVATED"* ou activé *"ACTIVATED"*. En revanche, les tâches qui précèdent cette dernière peuvent être dans un état arbitraire.
- La suppression d'une tâche *A* d'une instance d'un processus métier en cours d'exécution n'est possible que si *A* est en état non-activé *"NOT_ACTIVATED"* ou activée *"ACTIVATED"*, dans ce cas, les éléments associés à cette tâche sont supprimés à partir du modèle de processus métier correspondant. Tandis que, les tâches en cours d'exécution *"RUNNING"*, achevée *"COMPLETED"*, ou annulée *"SKIPPED"* ne peuvent pas être supprimés (il n'est pas autorisé à supprimer une tâche ou à modifier ses attributs si elle est déjà achevée).

4.8.3.1 Insertion d'un fragment de processus au niveau du schéma

La première règle tel que décrit dans l'algorithme 1 (*cf.* Listing 4.6) se déclenche quand un modeleur tente d'ajouter un objet de flux (FO_x) au niveau du schéma de processus métier.

La règle commence par vérifier tous les états des successeurs de FO_x au niveau de l'instance du processus métier concernée par le changement. Si ces instances sont tournées vers un état non-activé *"NOT_ACTIVATED"* ou activée *"ACTIVATED"* alors le changement peut être appliqué et le processus de la propagation de l'impact du changement peut commencer. Dans ce cas, l'objet de flux FO_x et ses connecteurs sont marqués en couleur verte pour notifier la faisabilité du changement et son statut est changé en *"added"*. Dans le cas contraire, FO_x et ses connecteurs sont marqués en couleur rouge pour notifier le fait que le changement peut être appliqué au niveau du schéma avec un risque d'incohérence dans la migration de ses instances vers ce nouveau schéma. Le statut de FO_x dans ce cas est changé en *"waiting"*.

Listing 4.6 *Insertion d'un fragment de processus au niveau du schéma.*

```
Input: N // set of nodes
Init: succs ← ∅
Begin
 if FOx ∉ N then
   /* Return all successors instances of FOx corresponding instance */
   for Ii ∈ Inst(FOx)(i = 1,…,n)do
     succs ← succs ∪ {Stat(Ii)};
   end for
   /* Verify the states of all successors instances of FOx corresponding
      instance*/
   for si ∈ succs(i = 1,…,n)do
    if si ∈ {Not_Activated, Activated} then
      /* Add FOx to a set of nodes N */
      N ← N ∪{FOx};
      /* The change can be applied and the corresponding instances can be
         Migrated correctly */
      status(FOx,"added");
      /* FOx and the control flows (incoming and outgoing sequence flows) are
         marked in green */
      mark(FOx,GREEN);
      mark (FC ∈ {Fsucc(FOx) ∪ Fpred(FOx)},GREEN);
      /* Call impact analysis rules */
      Call(Rule 06 ∧ Rule 07) ;
    else
     /*The change cannot be propagated immediately to the corresponding
       instances*/
      print("the change cannot be applied to the"+ Inst(FOx) +"instance
             immediately");
      status(FOx,"waiting");
      mark(FOx, RED) ;
      mark (FC ∈ {Fsucc(FOx) ∪ Fpred(FOx)},RED);
    end if
    end for
 end if
End
```

4.8.3.2 Suppression d'un fragment de processus au niveau du schéma

La deuxième règle telle que décrit dans l'algorithme 2 (*cf.* Listing 4.7) se déclenche quand un modeleur tente de supprimer un objet de flux (FO_x) au niveau du schéma de

processus métier. Dans ce cas, FO_x est supprimé et son statut est changé en *"deleted"*, si les instances en cours d'exécution sont tournées vers un état non-activé *"NOT_ACTIVATED"* ou activée *"ACTIVATED"*.

FO_x est marqué aussi en couleur verte pour notifier la faisabilité du changement ainsi que ses connecteurs. Dans le cas contraire, FO_x et ses connecteurs sont marqués en couleur rouge pour notifier le fait que la suppression de FO_x peut être appliquée au niveau du schéma avec un risque d'incohérence dans la migration de ses instances vers ce nouveau schéma. Le statut de FO_x dans ce cas est changé en *"waiting"*.

Listing 4.7 *Suppression d'un fragment de processus au niveau du schéma.*

```
Input: N // set of nodes
Init: S ← ∅
Begin
 if FOx ∈ N then
   S ← Stat(Inst(FOx));
   /* Verification of corresponding instances states of FOx */
    if S ∈ {Not_Activated, Activated} then
      /* The change can be applied and the corresponding instances can be
         migrated correctly */
      status(FOx,"deleted");
      /* FOx and the control flows (incoming and outgoing sequence flows) are
         marked in green */
      mark(FOx,GREEN);
      mark (FC ∈ {Fsucc(FOx) ∪ Fpred(FOx)},GREEN);
      /* Call impact analysis rules */
      Call(Rule 06 ∧ Rule 07) ;
      /* Remove FOx from a set of nodes N */
      N ← N-{FOx};
    else
     /*The change cannot be propagated immediately to the corresponding
       instances*/
      print("the change cannot be applied to the"+ Inst(FOx) +"instance
             immediately");
      status(FOx,"waiting");
      mark(FOx, RED) ;
      mark (FC ∈ {Fsucc(FOx) ∪ Fpred(FOx)},RED);
    end if
 end if
End
```

4.8.3.3 Insertion et suppression d'un fragment de processus au niveau de l'instance

Afin d'analyser la faisabilité d'un changement au niveau des instances I_x d'un processus métier tel que l'insertion ou la suppression d'un fragment de processus, les règles décrites dans l'algorithme 3 (*cf.* Listing 4.8) et l'algorithme 4 (*cf.* Listing 4.9) sont déclenchées respectivement.

Listing 4.8 *Insertion d'un fragment de processus au niveau des instances*

```
Input: N // set of nodes
Init: succs ← ∅
Begin
/* Ix is concerned instance */
if Ix ∉ S then
   for Ii ∈ Inst(FOx)(i = 1,…,n)do
     succs ← succs ∪ {Ii};
   end for
   /* Verification of successors of corresponding instances */
   for si ∈ succs(i = 1,…,n)do
```

```
      if s_i ∈ {Not_Activated, Activated} then
       /* The change can be applied */
       status(I_x,"added");
       mark(I_x,GREEN);
      else
       /* The change cannot be applied */
       print("the change cannot be applied to the"+ Ix +"instance immediately");
     end if
  end for
End
```

Listing 4.9 *Suppression d'un fragment de processus au niveau des instances.*

```
Input: N // set of nodes
Init: I ← ∅
Begin
if I_x ∈ S then
     I ← Inst(I_i);
   /* Verification of corresponding instances stats */
   if I ∈ {Not_Activated, Activated} then
      /* The change can be applied */
      status(I_x,"deleted");
      mark(I_x,GREEN);
   else
      /* The change cannot be applied */
      print("the change cannot be applied to the"+ Ix +"instance immediately");
   end if
 end if
End
```

L'utilisation de l'ensemble des règles de faisabilité du changement du processus métier est illustrée par l'exemple de la figure 4.6. Nous voulons dans cet exemple, ajouter l'activité *X* et supprimer l'activité *B* au niveau du schéma du processus. Dans ce cas, l'ajout de l'activité X ne peut pas être propagé immédiatement aux instances correspondantes, puisque les instances qui succèdent à cette dernière sont déjà achevées. Tandis que la suppression de l'activité *B* est possible, parce que le statut de son instance est tourné vers un état non activé.

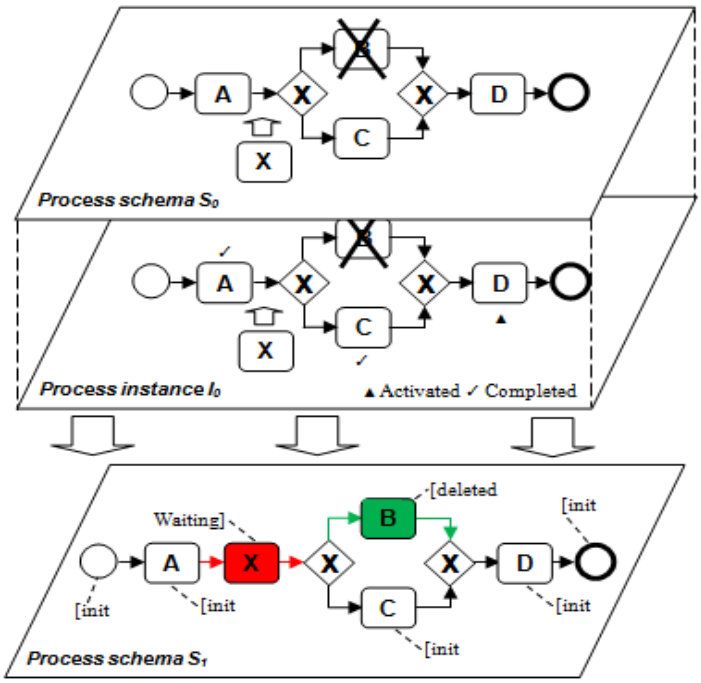


Figure 4.6 *L'ajout de l'activité «X» et la suppression de l'activité «B»*

4.8.4 Définition des règles de propagation de l'impact du changement

Les règles de propagation d'impact du changement, que nous avons appelées aussi règles d'implémentation de la stratégie de propagation, utilisent en grande partie une connaissance formalisé par une ontologie concernant les relations intra-couches et inter-couches qui existent au sein d'un processus métier ainsi que la qualification de ces relations par des renseignements sur leur conductivité de l'impact.

Partant de ce principe, nous avons défini un ensemble cohérent de règles qui permet d'analyser la propagation de l'impact du changement à travers l'analyse des relations de dépendance, mais aussi un processus générique de marquage des éléments affectés par le changement qui se repose principalement sur l'application de ces règles. Ces règles seront définies de façon algorithmique.

4.8.4.1 Règle d'analyse des dépendances intra-couche (analyse des dépendances d'activités)

L'algorithme 5 (*cf.* Listing 4.10), présente la première règle qui permet de propager l'impact d'un changement à travers l'analyse des dépendances d'activité. Cette règle est déclenchée lors d'un ajout, modification ou suppression d'un flux d'objets FO_x à partir d'un schéma de processus métier. La règle consiste à marquer le nœud correspondant à FO_x dans le graphe et ses connecteurs. Les ensembles $\mathcal{D}_{po}(FO_x)$ et $\mathcal{D}_{pi}(FO_x)$ ainsi que les flux de contrôle associés sont à leurs tours marqués pour exprimer la profondeur de l'impact du changement qui est calculée sur la base du facteur de pondération (IWF).

Listing 4.10 *Règle d'analyse de l'impact intra-couche (analyse des dépendances d'activités).*

```
Input: N // set of nodes
Init: Dpo ← ∅, Dpi ← ∅
Begin
 if Status (FOx) == "added" || "deleted" || "modified" then
  mark(FOx);
  mark (F ∈ {Ωi(FOx) ∪ Ωo(FOx)});
  /* Dpo gets successively each succeeding activities depends on FOx and the
     corresponding routing relationships in N */
  for all ai ∈ N (i = 1,…,n) do
    /* ai depend on FOx */
    if ai → FOx then
       Dpo ← Dpo U {ai}
        /* Compute the chance of impact for ai ∈ Dpo */
       IWF ← CalcIWF(ai);
       pathImp ← calcPathImpact(execPath(ai),IWF);
       putInMap(ai, pathImp) ;
    end if
  end for
  /* Dpi gets successively each preceding activities which FOx depends on and the
     corresponding routing relationships in N */
  for all ai ∈ N (i = 1,…, n) do
    /* FOx depend on ai */
    if FOx → ai then
       Dpi ← Dpi U {ai};
        /* Compute the chance of impact for ai ∈ Dpi */
       IWF← CalcIWF(ai);
       pathImp ← calcPathImpact(execPath(ai),IWF);
       putInMap(ai, pathImp) ;
    end if
  end for
  for all ai ∈ Dpo U Dpi (i = 1,…,n) do
```

```
      ImpactPow ← getFromMap(a_i) ;
      clr ← getColorPathImpact(ImpactPow);
      mark(a_i, clr) ;
      mark(F ∈ {Ωi(a_i) ∪ Ωo(a_i)}, clr)
   end for
  end if
End
```

4.8.4.2 Règle d'analyse des dépendances intra-couche (analyse des dépendances de données)

L'algorithme 6 (*cf.* Listing 4.11), présente la deuxième règle qui permet de propager l'impact d'un changement à travers l'analyse des dépendances de données et qui est complémentaire à la première règle. Dans cet algorithme, *OutputDs* représente l'ensemble des données *D* produites par FO_x, *AllOutDep* représente l'ensemble des activités qui dépendent de *D*. Tandis que *InputDs* est l'ensemble des données *D* utilisées par FO_x et *AllInDep* est l'ensemble d'activités dont les données produites sont utilisées par FO_x.

Listing 4.11 *Règle d'analyse de l'impact intra-couche (analyse des dépendances de données).*

```
Input: N // set of nodes
Init : OutputDs ← ∅, AllOutDep ← ∅, InputDs ← ∅,  AllInDep ← ∅
Begin
  if Status (FO_x) ==”added” ||”deleted” ||”modified” then
    mark(FO_x);
    mark (F ∈ {Ωi(FO_x) ∪ Ωo(FO_x)});
    for all D_i∈ OutPARs(FO_x) (i = 1,…, n) do
       OutputDs ← OutputDs U {D_i}
    end for
    for all D_i ∈ OutputDs(i = 1,…, n) do
      mark(D_i);
      mark(F ∈ {Ωi(D_i) ∪ Ωo(D_i)});
      /* AllOutDep gets successively each activities which depend on output data
         of the given activity FOx in N */
       for all a_j ∈ N( j = 1,…,n) do
        /* a_j depend on D produced by FO_x */
        If a_j --(D)--> FO_x then
          AllOutDep ← AllOutDep ∪ {a_i}
         /* Compute the chance of impact for a_i ∈ AllOutDep */
          IWF ← CalcIWF(a_i);
          pathImp ← calcPathImpact(execPath(a_i),IWF);
          putInMap(a_i, pathImp) ;
        end if
       end for
    end for
    for all D_i∈ InPARs(FO_x)(i = 1,…, n) do
       InputDs ← InputDs ∪ {D_i}
    end for
    for all D_i ∈ InputDs(i = 1,…,n) do
       mark(D_i);
       mark(F ∈ {Ωi(D_i) ∪ Ωo(D_i)});
       /* AllInDep gets successively each Activities where the output data is the
          input data of given activity FO_x in N */
       for all aj ∈ N( j = 1,…,n) do
          /* FO_x depend on D produced by a_j */
          if FO_x --(D)--> a_j then
            AllInDep ← AllInDep ∪ {a_j}
            /* Compute the chance of impact for a_j ∈ AllInDep */
            IWF ← CalcIWF(a_j);
            pathImp ← calcPathImpact(execPath(a_j),IWF);
            putInMap(a_j, pathImp);
```

```
          end if
        end for
        for all a_k ∈ ALLInDep U ALLOutDep(k = 1,…, n) do
            ImpactPow ← getFromMap(a_k) ;
            clr ← getColorPathImpact (ImpactPow);
            mark(a_k, clr) ;
            mark(F ∈ {Ωi(a_k) U Ωo(a_k)} , clr);
        end for
      end for
    end if
End
```

4.8.4.3 Règle d'analyse des dépendances inter-couche

La troisième règle telle que décrit dans l'algorithme 7 (*cf.* Listing 4.12) analyse la propagation de l'impact entre la couche métier et la couche services. $\mathcal{D}ps(FO_x)$ représente l'ensemble de tous les services invoqués par FO_x. Tous les services invoqués et les connecteurs correspondants (invocation de service) sont marqués pour exprimer la profondeur de l'impact des changements.

Listing 4.12 *Règle d'analyse de l'impact inter-couche.*

```
Input: N // set of nodes
Init: Dps ← Ø
Begin
   if Status (FO_x) ==”added” ||”deleted” ||”modified” then
      mark(FO_x);
      mark (F ∈ {Ωi(FO_x) U Ωo(FO_x)});
      /* Dps gets each services invoked by FOx and the corresponding routing
        relationships */
      for all s_i ∈ S (i = 1,…, n) do
        /* s_i is invoked by FO_x */
        if FO_x ⇝ s_i then
          Dp_s ← Dp_s U {s_i}
           /* Compute the chance of impact for s_i ∈ Dp_s */
          IWF ← CalcIWF(s_i);
          pathImp ← calcPathImpact(invokepath(s_i),IWF);
          putInMap(s_i, pathImp) ;
        end if
      end for
      for all s_i → Dps (i = 1,…,n) do
          ImpactPow ← getFromMap(s_i) ;
          clr ← getColorPathImpact(ImpactPow);
          mark(s_i,clr) ;
          mark(F ∈ {r(FOx,s_i)},clr)
      end for
   end if
End
```

4.9 Conclusion

L'analyse de l'impact du changement est une étape très importante dans la gestion de l'évolution des processus métier. Elle permet d'accroître la prise de conscience des différents acteurs afin d'éviter les effets secondaires et d'estimer les effets de vagues du changement concerné.

Dans ce chapitre, nous avons présenté une méthode permettant l'analyse *a priori* de l'impact du changement des processus métier à travers une approche basée sur une défi-

nition de la relation de dépendance tenant compte des spécificités de modélisation de ces processus. En effet, nous avons proposé une classification exhaustive des différentes relations de dépendance qui peuvent exister au sein d'un processus métier, l'objectif étant d'offrir une meilleure identification de leur rôle dans la propagation de l'impact du changement.

Le chapitre a décrit ensuite en détail, le processus de propagation de l'impact du changement à travers ces relations de dépendance, pour cela un ensemble de métriques est proposé afin de calculer la profondeur de la propagation de l'impact tout en faisant la distinction entre les entités impactées en termes de puissance d'impact.

Nous avons ensuite proposé une modélisation de cette propagation de l'impact à travers deux constituants qui sont une base de connaissances obtenue à partir de la définition d'une ontologie étendue décrivant une sémantique de l'ensemble des éléments d'un processus métier et leurs interdépendances et une base de règles incarnant un ensemble de règles dédiées à l'étude *a priori* de la faisabilité d'un changement à un instant T, mais aussi à la compréhension et à l'analyse du changement du processus métier et de son impact.

L'objectif principal étant de déterminer *a priori* la conduction de l'impact du changement depuis une entité modifiée vers d'autres entités susceptibles d'être affectés par la modification. La plate-forme développée selon la modélisation proposée est conçue pour permettre aux modeleurs et aux experts métiers d'évaluer le risque associé à de tels changements et d'estimer ainsi l'effort requis pour leurs réalisations.

Dans le chapitre suivant, nous allons présenter le troisième axe de notre contribution qui est la gestion qualitative des processus métier durant leur cycle de vie et plus particulièrement, après chaque changement.

CHAPITRE 5

La gestion qualitative des processus métier

5.1 Introduction

La modélisation et l'amélioration de la qualité des processus métier ont connu un intérêt croissant durant ces dernières décennies. En effet, les entreprises ont pris conscience de l'impact indéniable que peuvent procurer une meilleure compréhension et une meilleure gestion de la qualité des processus métier en matière d'efficacité, de cohérence et de transparence de leurs activités. Cet intérêt est appuyé par la norme ISO9000 qui recommande d'évaluer régulièrement la qualité des processus métier de l'entreprise afin de pouvoir les surveiller et les améliorer (Morley, Hugues, Leblanc, & Hugues, 2004).

À cet égard, l'évaluation permanente de la qualité des processus métier et en particulier celle des modèles de processus métier est l'une des exigences de toute mise en œuvre réussie d'un ensemble de processus au sein d'une entreprise. En effet, obtenir des modèles compréhensibles, réutilisables et facilement maintenables permet de minimiser les défauts de conception et d'éviter des dysfonctionnements susceptibles de se produire une fois les processus déployés.

Dans l'optique d'une telle évaluation, une variété de caractéristiques (Gucegliogl u & Demirors, 2005), (Heravizadeh, Mendling, & Rosemann, 2009) et de métriques (Aguilar, Cardoso, García, Ruiz, & Piattini, 2009), (Aguilar, Ruiz, García, & Piattini, 2006), (Vanderfeesten, Cardoso, & Reijers, 2007) ont récemment été proposés afin de gérer, mesurer et / ou d'améliorer la qualité des modèles de processus métier. La majorité de ces travaux focalisent sur l'aspect structurel de ces modèles, consistant ainsi à évaluer certaines caractéristiques de la qualité telle que la compréhensibilité *"understandability"* et la facilité de modification *"modifiability"* qui appartiennent respectivement aux concepts plus généraux de la facilité d'utilisation *"usability"* et de la maintenabilité *"maintainability"*, respectivement.

Dans ce chapitre, nous proposons une approche plus exhaustive inspirée par la norme ISO / CEI 9126-1 utilisant un ensemble pertinent de facteurs, de critères et de métriques, permettant ainsi une meilleure évaluation de la qualité globale d'un processus métier, et cela durant tout son cycle de vie et plus particulièrement après chaque changement de ce dernier.

En effet, nous nous intéressons tout particulièrement dans ce chapitre à l'impact du changement de la structure des processus métier sur la qualité, ce que nous appelons l'impact qualitatif du changement. Le nombre de modifications d'un processus peut être un indicateur important du taux de défaillance de ce processus, de sa stabilité et de sa complexité. Nous discuterons l'analyse de la propagation des impacts structurels et qualitatifs des changements. Notre objectif est de proposer une approche systématique de mise en œuvre de changements « réussis », en d'autres termes de minimiser les risques de dégradations de la qualité engendrés par les changements.

Le reste du chapitre est organisé comme suit : la section 5.2 met en exergue les enjeux de l'évaluation de la qualité des processus métier. La section 5.3 explore la notion d'évaluation de la qualité des processus métier avec une synthèse de l'état de l'art des différentes approches et métriques existantes permettant de mener une telle évaluation. La section 5.4 explicite en détail l'élaboration de l'approche que nous proposons à l'aide d'un exemple. La section 5.5 explore la notion d'évaluation de la qualité avec l'aide du

modèle qualitatif de l'évolution du processus métier. Dans la section 5.6 nous concluons le chapitre en soulignant les perspectives de notre travail.

5.2 Gestion qualitative des processus métier

La norme (ISO 9000:2000, 2005) définit la qualité comme « l'aptitude d'un ensemble de caractéristiques intrinsèques à satisfaire des exigences». Partant de ce constat, la gestion de la qualité des logiciels a fait l'objet de nombreux travaux de recherche. Le principal objectif de ces recherches a consisté à proposer un ensemble de méthodes (Boehm, Brown, & Lipow, 1976), (Cavano & McCall, 1978), (Ishikawa, 1985), (Wong & Jeffery, 2001), de standards (Wong & Jeffery, 1995) (ISO/IEC 9126-1, 2001) et d'outils permettant de gérer, mesurer, surveiller et améliorer la qualité des logiciels.

Plus récemment, la gestion de la qualité des processus métier et plus précisément celle des modèles de processus métier a constitué un sujet de recherche très actif. Le but est de mettre en place un mécanisme efficace qui permet d'améliorer la compréhension, la fiabilité et la réutilisation de ses modèles de processus métier durant tout leur cycle de vie, et plus particulièrement, après chaque changement.

En effet, l'obtention d'un modèle de processus métier de haute qualité réduit considérablement les défauts de conception et les dysfonctionnements qui peuvent se produire une fois les processus déployés. Dans ce contexte, l'évolution d'un modèle de processus métier sans dégradation de la qualité dépend en grande partie de la compréhension des relations qui existent entre la propagation de l'impact du changement structurel et qualitatif. Ainsi, les changements de la structure d'un modèle de processus métier peuvent avoir des effets sur la qualité originelle de ce dernier. On observe en effet, une tendance à une dégradation de la qualité du modèle de processus métier proportionnelle à l'accroissement de sa taille et de sa complexité qui est une conséquence des additions de nouvelles activités et de nouvelles connexions entre ces activités se traduisant, généralement, par une modification de sa structure. La structure du modèle de processus métier et par conséquent sa qualité change donc avec le développement de ces activités. Il est donc impératif dans ce contexte d'évaluer l'amélioration ou la dégradation de la qualité des processus métier après chaque changement.

5.3 Évaluation de la qualité des modèles de processus métier

La réévaluation régulière de la qualité du processus dans l'objectif d'en suivre l'évolution est l'une des exigences de toute mise en œuvre réussie d'un ensemble de processus au sein d'une entreprise. Bien qu'il n'existe pas de définition standard de la qualité d'un processus métier, diverses approches pour l'évaluation de cette qualité ont été proposées dans la littérature. Notre étude nous a permis de les classer en trois catégories : les approches centrées sur les méthodes, les approches centrées sur l'évaluation de la qualité des processus métier et les approches centrées sur l'évaluation de la qualité des modèles de processus métier.

5.3.1 Tour d'horizon des approches existantes

Les approches centrées sur les méthodes sont des recherches qui proposent des guides méthodologiques et des bonnes pratiques pour assurer la qualité des modèles de processus métier. Dans (Becker, Rosemann, & von Uthmann, 2000) les auteurs proposent un ensemble de guides pour améliorer certaines caractéristiques de qualité telle que l'exactitude, la compréhension des modèles de processus métier. D'autres auteurs proposent des patrons de conception réutilisables qui aident à produire des modèles de processus métier (van der Aalst W. , ter Hofstede, Kiepuszewski, & Barros, 2003).

La deuxième catégorie centrée sur l'évaluation de la qualité des processus métier s'intéresse à ces derniers au niveau de leur exécution et de leur contrôle. L'objectif principal étant de proposer des techniques pour la vérification, la validation et l'amélioration des performances des processus métier. Dans (Heravizadeh, Mendling, & Rosemann, 2009) les auteurs ont présenté une approche appelée QoBP permettant d'évaluer la qualité des processus métier. L'approche représente un moyen pour capturer certaines dimensions de la qualité d'un processus. Elle définit quatre catégories de la qualité des processus métier qui sont la qualité des fonctions, la qualité des entrées et des sorties d'objets et la qualité des ressources humaines et non-humaines. Gucegliogl et Demirors dans (Gucegliogl & Demirors, 2005) utilisent un ensemble de caractéristiques de qualité adopté de la norme ISO/IEC 9126 pour mesurer les effets d'un système d'information sur les processus métier et par conséquent la qualité des processus métier. Le travail de (Heinrich & Paech, 2010), qui est similaire à celui de Gucegliogl et Demirors dans la démarche, propose un autre ensemble de caractéristiques supplémentaires toujours inspiré par la norme ISO/IEC 9126 qui couvrent d'autres aspects du processus métier tels que les acteurs et les ressources en plus des activités et qui participent à l'évaluation de la qualité globale d'un processus métier.

La troisième catégorie centrée sur l'évaluation de la qualité des modèles de processus métier passe souvent par la mesure de la qualité à travers la définition d'un ensemble de métriques de qualité inspiré du domaine des produits logiciels. Les approches qui s'inscrivent dans ce courant justifient cette utilisation de métriques par les similitudes structurelles existantes entre un processus métier et un produit logiciel comme le montre la Table 5.1.

En effet, le concept de métriques de processus a d'abord été introduit dans (Cardoso J. , 2005) pour fournir une base quantitative pour la conception, le développement, la validation et l'analyse des modèles de processus métier. Plus tard, ce concept a été connu sous le nom de *"Business Process Quality Metrics"* (BPQM).

Dans (Gruhn & Laue, 2006) les auteurs ont présenté un ensemble d'enquêtes où ils ont étudié l'importance de l'exactitude des modèles de processus métier d'un point de vue empirique. Ils ont défini deux métriques qui quantifient la bonne organisation structurelle *"structuredness"*. Van Belle dans (Van Belle, 2004) décrit un cadre permettant l'évaluation et la comparaison des modèles de processus métier en se basant sur une analyse syntaxique, sémantique et pragmatique. Ils mentionnent cinq des mesures qu'ils considèrent importantes : couplage, cohésion, complexité, modularité, et taille.

Les produits logiciels	Les processus métier
Module/Class	Processus/sous processus
Méthodes/Fonctions	Activité/Tâche
Variables / constantes	Données
Appel de méthode	Réception d'un flux de séquence ou d'un flux de messages par une des tâches ou un des événements
Commentaires	Annotations
Interfaces	L'ensemble des tâches d'un processus ou d'un sous-processus qui peuvent envoyer ou recevoir un flux de messages.

Table 5.1 *Les similitudes entre les produits logiciels et les processus métier.*

Dans la même optique, d'autres travaux de recherche comme ceux détaillés ci-dessous ont été proposés dans la littérature pour évaluer la complexité, le couplage, la cohérence, la modularité, et la taille d'un modèle de processus métier.

5.3.1.1 Mesurer la complexité d'un modèle de processus métier

La complexité mesure la simplicité *"simpleness"* et la compréhensibilité *"understandability"* d'un modèle de processus métier. Cardoso *et al.* (Cardoso, Mendling, Neumann, & Reijers, 2006), ont identifié trois types de complexité d'un processus métier: (i) la complexité des calculs *" computational complexity"*, (ii) la complexité psychologique, *"psychological complexity"* et (iii) la complexité de représentation "representational complexity".

La première métrique présentée dans la littérature, qui est adaptée du nombre cyclomatic de McCabe (McCabe, 1976), permet de mesurer la complexité d'un flux de contrôle au sein d'un modèle de processus métier. Cette métrique est appelée *"Control Flow Complexity"* (CFC) (Cardoso, Mendling, Neumann, & Reijers, 2006).

L'idée principale derrière cette métrique est d'évaluer la complexité introduite dans un processus par la présence de passerelles tel que XOR-Split, OR-Split et AND-Split.

Ainsi, pour une passerelle XOR-Split, le CFC d'un flux de contrôle est tout simplement le nombre de points de sortie "*fan-out*" de la scission. Il est noté comme suit:

$$CFC_{XOR\text{-}split}\,(a) = fan\text{-}out\,(a)$$

Pour une passerelle OR-split, le CFC d'un flux de contrôle est de 2n-1, où n représente les points de sortie *"fan-out"* de la scission. Il est noté par:

$$CFC_{OR\text{-}split}(a) = 2\,fan\text{-}out(a) - 1$$

Pour une passerelle AND-split, le CFC d'un flux de contrôle est de 1 :

$$CFC_{AND\text{-}split}\,(a) = 1$$

Mathématiquement, la métrique CFC est additionnelle. Ainsi, il est très facile de calculer la complexité d'un modèle de processus métier, par l'addition des différents CFC de toutes les scissions dans le modèle de processus métier.

$$CFC\,(P) = \Sigma_{l=0}^{n}\, CFC_{XOR\text{-}split}\,(i) + \Sigma_{j=0}^{n}\, CFC_{OR\text{-}split}\,(j) + \Sigma_{k=0}^{n}\, CFC_{AND\text{-}split}\,(k)$$

Pour tester la validité de cette métrique, une expérience basée sur les propriétés de Weyuker a été réalisée à travers une validation empirique (Cardoso J. , 2005). Il était constaté que la métrique CFC est fortement corrélée avec la complexité du flux de contrôle d'un processus.

La métrique de cyclicité (CYC) proposée dans (Heravizadeh, Mendling, & Rosemann, 2009) compte le ratio entre le nombre de nœuds (activités et passerelles) dans un cycle (NNC) et le nombre total de nœuds (TNN) dans un modèle de processus métier comme suit :

$$CYC = NNC / TNN$$

Mendling propose dans (Mendling J. , 2006) une métrique dite de densité "Density metric" inspirée par l'analyse des réseaux sociaux afin de quantifier la complexité d'un modèle de processus métier exprimé à l'aide du diagramme EPC. Dans (Cardoso J. , 2005) l'auteur présente une métrique permettant de mesurer la complexité du flux de données des modèles de processus métier. Gruhn et Laue (Gruhn & Laue, 2006) utilisent la notion de poids cognitifs *"cognitive weights"* comme une structure basique de contrôle pour mesurer la difficulté de la compréhension d'une structure de contrôle d'un workflow.

5.3.1.2 Mesurer le couplage d'un modèle de processus métier

La notion de couplage dans les modèles de processus métier se réfère à la façon dont les activités d'un processus métier sont liées ou connectées. Une activité est dite connectée à une autre activité, si et seulement si elles partagent un ou plusieurs éléments d'information telle que des données. La métrique de couplage d'une activité détermine le nombre total d'activités qui lui sont liées (Vanderfeesten, Cardoso, & Reijers, 2007).

Pour un modèle de processus métier lambda, la métrique de couplage est égal au nombre d'interconnexions entre toutes ses activités, en d'autres thèmes, cette métrique compte toutes les paires d'activités dans le modèle de processus métier qui sont liées l'une à autre. En outre, le degré de couplage dépend de la complexité et du type de connexions entre les activités (AND, OR, XOR).

La métrique de couplage d'une activité reflète la façon dont une activité critique ou importante se situe dans un modèle de processus métier. En fait, une activité avec une valeur de couplage élevée détermine le fonctionnement d'un grand nombre d'autres activités dans le processus métier. Ainsi, son dysfonctionnement peut causer le dysfonctionnement de plusieurs autres activités. Ceci peut compromettre le fonctionnement global du processus. Ce genre d'activités devrait être soit évitée dans un modèle de processus métier, ou traitée avec un soin particulier, par exemple, en ayant une action de contrôle pour cela.

D'autre part, un modèle de processus métier avec une valeur de couplage élevé indique un niveau élevé de dépendances informationnelles entre ses activités ce qui rend le processus vulnérable et sa maintenabilité difficile.

Une limite de la métrique de couplage est qu'elle ne donne pas une indication sur la réutilisabilité d'un BPM. Cette caractéristique de qualité est importante pour la conception d'un modèle réutilisable. Une deuxième limite est axée sur les données échangées. En effet, les informations sur la dépendance de l'activité en termes d'utilisation de don-

nées ne sont pas fournies. Cette métrique de couplage orientée donnée est complétée par la métrique de cohésion décrite dans la section suivante.

5.3.1.3 Mesurer la cohésion d'un modèle de processus métier

Vanderfeesten *et al.* (Vanderfeesten, Reijers, & van der Alst, 2008), définissent la métrique de cohésion comme le produit à la fois de la cohésion de la relation et de l'information d'une activité. Cela signifie que, pour chaque activité dans le modèle de processus métier, la cohésion totale est calculée en multipliant la cohésion de l'information et la cohésion de la relation de l'activité.

La relation de cohésion quantifie combien d'opérations différentes sont liées au sein d'une activité. Cela détermine pour chaque opération d'une activité, combien d'autres opérations partagent une entrée ou de sortie avec cette dernière. D'autre part, la cohésion de l'information se concentre sur l'ensemble des éléments d'information qui sont utilisés comme entrée ou de sortie par une opération à l'intérieur de cette activité.

La cohésion d'un modèle de processus métier représente le ratio entre la moyenne de toutes les valeurs de cohésion d'activité (additionnant toutes les valeurs de cohésion) par le nombre d'activités. La valeur de cette métrique de cohésion est toujours comprise entre 0 et 1.

Cette proposition est appuyée par le travail de Reijers et Vanderfeesten (Reijers & Vanderfeesten, 2004) qui proposent une métrique de cohésion qui calcule le degré de couplage et de cohésion dans les modèles BOM (Bill Of Materials) en analysant les éléments de données.

5.3.1.4 Mesurer la modularité d'un modèle de processus métier

La métrique de modularité *M* (Gruhn & Laue, 2006) comme son nom l'indique est utilisée pour évaluer la modularité du modèle de processus métier. Elle est calculée comme suit :

$$M = (fan\text{-}in * fan\text{-}out)^2$$

Où fan-in représente tous les modules (ex : sous-processus) qui font appel à un autre module *m* dans le modèle de processus métier et fan-out représente tous les modules qui sont appelés à partir d'un module *m*.

5.3.1.5 Mesurer la taille d'un modèle de processus métier

Les auteurs dans (Cardoso, Mendling, Neumann, & Reijers, 2006) proposent d'adapter les métriques de Halstead pour mesurer la taille d'un modèle de processus métier. Cela passe par le calcul de la longueur *N*, le volume *V* et la difficulté *D* d'un modèle de processus métier. Elles sont appelées *"Halstead-based Process Complexity"* (HPC) et calculées comme suit:

$$N = n_1 * \log_2 (N_1) + N_2 * \log_2 (n_2)$$

$$V = (N_1 + N_2) * \log_2 (n_1 + n_2)$$

$$D = (n_1 / 2) * (N_2 / N_2)$$

où:

- n_1 représente le nombre distincts d'activités, de passerelles, et de flux de contrôle d'un modèle de processus métier.
- n_2 représente le nombre distincts de données manipulées par le processus et ses activités.
- N_1 et N_2 représentent respectivement le nombre total des éléments et des données d'un modèle de processus métier respectivement.

Dans le même travail, le nombre d'activités (tâches et sous-processus) (NOA) et le nombre d'activités et des flux de contrôle (NOAC) dans un modèle de processus métier sont deux métriques de complexité supplémentaires adaptées de la métrique LOC *"Line Of Code"* (Fenton & Pfleeger, 1998) qui permet de mesurer le nombre de lignes de code d'un produit logiciel. La première métrique est une pure adaptation qui compte le nombre d'activités dans un modèle de processus métier sans tenir compte de la fonctionnalité et de la complexité tandis que la seconde tient compte de ces deux caractéristiques.

En conclusion de ce tour d'horizon, nous avons constaté que l'ensemble des métriques proposées dans la littérature se focalisent sur l'aspect structurel des modèles de processus métier plutôt que l'aspect sémantique et/ou comportemental de ces derniers.

Dans la section suivante, nous allons proposer une approche plus exhaustive inspirée par la norme ISO / CEI 9126-1 qui utilisent un ensemble pertinent de dimensions, de critères et de métriques permettant ainsi de mesurer la qualité des processus métier allant au-delà des considérations purement syntaxiques mais intégrant aussi l'aspect sémantique et comportemental des processus métier.

5.4 L'approche proposée

La qualité globale du logiciel est généralement évaluée comme le résultat d'une évaluation sur trois niveaux (Facteurs, Critères, et Métriques). Les facteurs représentent généralement un des besoins non-fonctionnels exprimés par l'utilisateur tels que la facilité d'usage *"Usability"* ou la fiabilité *"Reliability"*, etc. Ils représentent des attributs de la qualité du point de vue externe. En d'autres termes, ce sont des attributs extérieurement appréciables alors que les critères représentent des considérations plus détaillées ou techniques. Autrement dit, des attributs internes. Ils ne sont généralement pas appréciables ou non visibles par l'utilisateur du logiciel. Ils expriment des perspectives de la qualité concernant le développeur et l'ingénieur en assurance qualité. Parmi ces critères, nous pouvons citer la modularité *"Modularity"* comme exemple. Si l'on considère un modèle de qualité comme un arbre où les nœuds sont les facteurs (F) raffinés par des critères (C) mesurés par des métriques (M) (Mccall, Cavano, & Walters, 1977), ces dernières représenteraient donc les feuilles de cet arbre. Elles servent à mesurer et/ou estimer les différents critères.

Soit donc $G(\mathcal{X}, \mathcal{U})$ le graphe FCM (Facteurs, Critères, Métriques) de la qualité établi pour une application lambda. L'ensemble $\mathcal{X}$ des nœuds du graphe $G(X, U)$ de la qualité peut donc être défini par :

$$\mathcal{X} = \mathcal{X}_0 \cup \Sigma F \cup \Sigma C \cup \Sigma M$$

- $\mathcal{X}_0$ est la racine du graphe. Il représente l'élément le plus abstrait de la qualité qu'on appellera la qualité globale du processus métier. Il correspond au niveau 0 du raffinement.
- $\Sigma F = \{f_1, f_2, \ldots, f_n\}$ est l'ensemble des facteurs constituant le premier raffinement de la qualité générale. C'est le niveau 1 du raffinement.
- $\Sigma C = \{c_1, c_2, \ldots, c_n\}$ représente l'ensemble des critères raffinant les facteurs. C'est le niveau 2 du raffinement.
- $\Sigma M = \{m_1, m2, \ldots, m_n\}$ représente l'ensemble des métriques permettant une évaluation quantitative et/ou qualitative des critères.

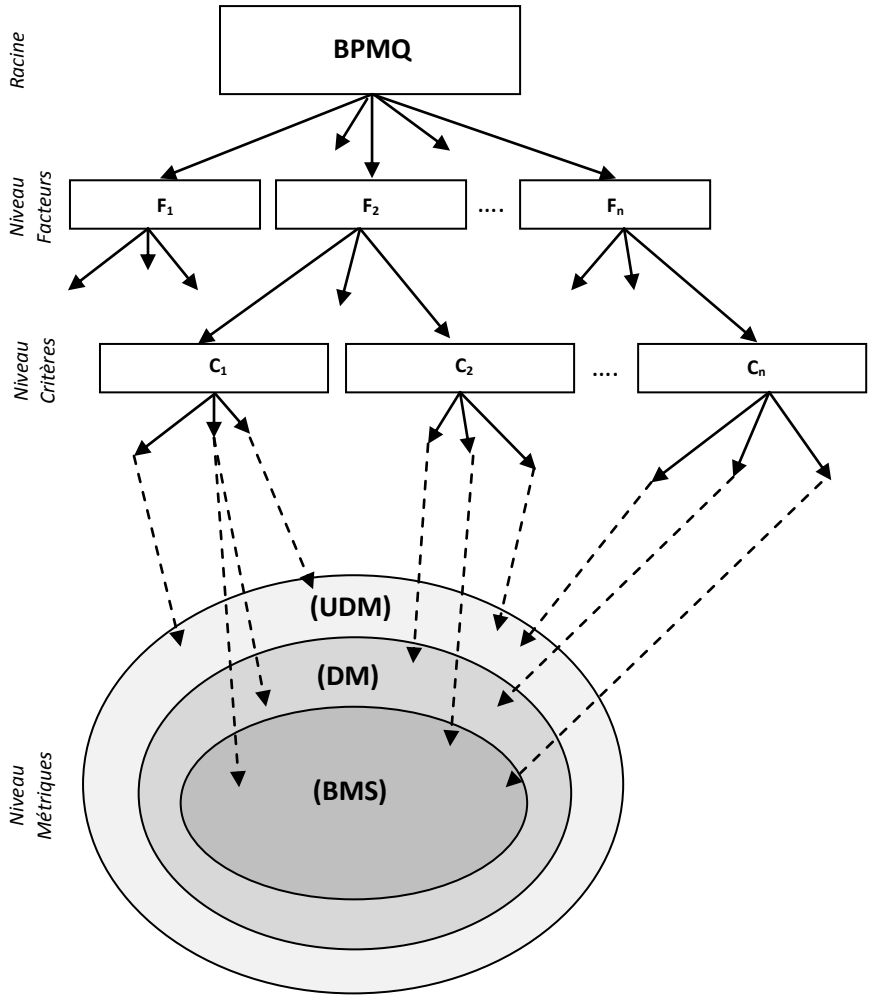
Dans cette section, nous allons proposer un cadre exhaustif inspiré par la norme ISO/CEI 9126-1 représentant à la fois une contribution à la maturité des travaux existants à l'égard de l'évaluation et de l'amélioration de la qualité des processus métier et un moyen de traçabilité de l'impact qualitatif du changement d'un processus métier durant son cycle de vie.

En effet, pour analyser les variations de la qualité dans le cadre d'une analyse qualitative des changements des processus métier, nous tenons compte des dépendances entre Facteurs et Critères. Cela peut aider les modeleurs et les experts métiers à tracer l'impact sur la qualité et cela dans le sens d'une amélioration ou détérioration d'un facteur de cette qualité.

Dans notre approche, nous avons identifié quatre niveaux hiérarchiques comme le montre la figure 5.1. Le premier niveau représente la qualité globale d'un processus métier (la racine) nommé BPMQ *"Business Process Models Quality"* qui dépend de l'évaluation d'un ensemble de facteurs ou de dimensions qui sont la performance, l'efficacité, la fiabilité, les ressources et les acteurs et qui se trouvent dans un niveau inférieur. Les critères de qualité sont regroupés en dimensions de qualité, chaque dimension représentant une facette de la qualité des processus métier. Par conséquent, un critère de qualité représente un aspect particulier d'une dimension donnée. Un critère de qualité est associé à un ensemble de métriques classées en couches permettant de l'évaluer de manière efficace.

Notre approche offre aussi la possibilité d'évaluer la qualité de service QoS fournie par le processus métier à travers des facteurs tels que la performance et tient aussi compte de certains facteurs de qualité qui sont souvent négligés dans l'évaluation qualitative globale même si elles ont une importance significative, par exemple les ressources et des acteurs.

L'approche proposée peut être étendue à d'autres dimensions de la qualité existant dans la littérature telle que la facilité d'utilisation *"usability"* et de maintenabilité *"maintainability"* qui peuvent s'ajouter à l'ensemble proposés. Dans ce qui suit, nous allons décrire chacun des facteurs, des critères, et des mesures sélectionnées.



(BMS) : Ensemble des métriques de base
(DM) : Ensemble des métriques dérivées
(UDM) : Ensemble des métriques définies par l'utilisateur

Figure 5.1 *Représentation arborescente des facteurs, critères et métriques.*

5.4.1 Efficacité *"Efficiency"*

L'efficacité est considérée comme l'un des éléments fondamentaux d'un bien fondé de tout processus métier. Elle représente la capacité d'un processus métier à fournir une performance appropriée, par rapport à la quantité de ressources utilisées ainsi que la durée d'utilisation par les différents acteurs sous des conditions spécifiques.

L'efficacité est raffinée en plusieurs critères qui sont : l'efficacité du temps de réalisation, l'efficacité des ressources employées, et l'efficacité des coûts.

5.4.1.1 L'efficacité du temps de réalisation *"Time Behaviour"*

La capacité d'un processus métier à fournir une réponse appropriée dans un délai raisonnable lors de l'exécution de ses activités sous des conditions spécifiques.

5.4.1.2 L'efficacité des ressources employées *"Resource efficiency"*

L'utilisation des ressources est considérée comme l'un des facteurs de qualité associé à des ressources. Il traduit la capacité d'un processus métier à utiliser un type et une quantité de ressources appropriés lorsque ses activités sont exécutées sous des conditions spécifiques.

5.4.1.3 L'efficacité des coûts *"Cost efficiency"*

L'efficacité des coûts est caractérisée par le coût associé à l'exécution des activités d'un processus métier. En effet, le coût de l'exécution d'une activité représente le coût de l'utilisation de l'équipement, le coût de l'intervention humaine, et toutes les fournitures nécessaires à la réalisation de cette activité. Ces fonctions de coût sont utilisées pour calculer le coût total associé à l'exécution d'une activité.

5.4.2 Fiabilité *"Reliability"*

La capacité d'un processus métier à maintenir un certain niveau de performance lorsqu'il est utilisé sous certaines conditions spécifiées. Ce facteur de qualité peut être raffiné en plusieurs critères qui sont : la maturité, la tolérance aux pannes et la capacité de récupération.

5.4.2.1 La maturité *"Maturity"*

La maturité d'un processus métier représente sa capacité à éviter un échec en raison d'un défaut relevé dans ses activités.

5.4.2.2 La tolérance aux pannes *"Fault Tolerance"*

La tolérance aux pannes d'un processus métier représente l'aptitude d'un processus métier à maintenir un certain niveau de performance en cas de défauts ou de violation de ses spécifications.

5.4.2.3 La capacité de récupération *"Recoverability"*

La capacité d'un processus métier à rétablir un niveau spécifique de performances et de restaurer les données et les ressources directement affectées par une défaillance.

5.4.3 Performance *"Performance"*

La performance est un facteur de qualité qui fait référence à l'aspect comportemental du processus métier. Il permet de mesurer la qualité du service QoS fourni par ces processus en évaluant la relation entre les services fournis et l'utilisation des ressources. Ce facteur de qualité peut être raffiné en plusieurs critères qui sont : le débit de traitement, le temps de réponse, le temps d'exécution et le coût d'exécution.

5.4.3.1 Le débit de traitement *"Throughput rates"*

Ce critère permet de recenser le nombre d'événements traités par une activité, soit le nombre d'entrées et/ou de sorties traitées par une activité durant un intervalle d'observation.

5.4.3.2 Le temps d'exécution *"Execution time"*

Le temps est l'un des critères les plus utilisés pour évaluer la performance d'un système. Dans notre cas, le temps d'exécution d'une activité représente le temps achèvement d'une instance de cette activité (Cardoso, Miller, & Arnold, 2002).

5.4.3.3 Le temps de réponse *"Timeliness"*

Le temps de réponse d'un système correspond à l'intervalle entre la demande d'un utilisateur et la réponse du système (Laird & Brennan, 2006). Dans notre cas, le temps de réponse représente le temps qu'un ensemble d'entrées et/ou de sorties doit être traité par une activité afin de répondre à une demande particulière.

5.4.3.4 Le coût d'exécution *"Execution cost"*

Le coût d'exécution d'une activité représente le coût associé à la réalisation de cette dernière dans un processus métier. Le coût est un facteur important, car les organisations doivent fonctionner conformément à leur plan financier (Cardoso, Miller, & Arnold, 2002).

5.4.4 Ressource *"Resource"*

Les activités d'un processus métier ont généralement besoin de ressources qui peuvent être par exemple une machine, un appareil ou un système informatique pour leur exécution. La qualité de ces ressources peut affecter la qualité des activités. Le contexte de l'utilisation d'une ressource est l'activité qui utilise la ressource. Par conséquent, la satisfaction de contexte caractéristique est omise pour les ressources. Le facteur ressource est raffiné en plusieurs critères qui sont : l'interopérabilité, la sécurité et le déploiement.

5.4.4.1 Interopérabilité *"Interoperability"*

L'interopérabilité est la capacité d'une ressource à interagir avec une ou plusieurs autres ressources appropriées.

5.4.4.2 Sécurité *"Security"*

La capacité d'un processus métier à protéger les données, les ressources et tout autre objet physique afin que les acteurs ou autres ressources non autorisées ne puissent pas y accéder et de permettre un accès libre et continu à ceux qui en ont droit.

5.4.4.3 Déploiement *"Deployment"*

C'est la capacité de la ressource à être déployée d'une manière efficace.

5.4.5 Acteur *"Actor"*

Un acteur peut accomplir une ou plusieurs activités au sein d'un processus métier. La qualité du processus métier dépend donc fortement de la disponibilité et des compétences de ceux qui l'utilisent, en l'occurrence les différents acteurs. La disponibilité et l'aptitude sont les principaux critères de ce facteur de qualité.

5.4.5.1 Disponibilité *"Availability"*

C'est la capacité d'un acteur à être en mesure d'exécuter une activité pendant un interval d'observation.

5.4.5.2 Aptitude *"Suitability"*

C'est la capacité d'un acteur à mener à bien l'exécution d'une activité.

Cependant, les facteurs et les critères de qualité que nous avons définis précédemment sont assez abstraits et ne peuvent en aucun cas évaluer directement la qualité globale des processus métier. Par conséquent, il est important de leurs associer un ensemble de métriques afin de les évaluer.

5.4.6 Mesurer la qualité des processus métier

Dans le cadre de notre modèle qualitatif BPMQ, nous avons proposé un ensemble de métriques pertinentes qui s'ajoutent à ceux qui existent dans la littérature (Rolón Aguilar, Ruiz, García, & Piattini, 2006), (Gruhn & Laue, 2006), (Reijers & Vanderfeesten, 2004), (Cardoso, Mendling, Neumann, & Reijers, 2006), (Cardoso, Miller, & Arnold, 2002), (Vanderfeesten, Cardoso, & Reijers, 2007), (Rolón Aguilar, García, & Ruiz, 2006), (Mendling J. , 2008) pour mesurer les différents critères et par conséquent, les différentes dimensions de qualité détaillées précédemment. En effet, l'un de nos objectifs est d'affiner les critères de qualité à des mesures spécifiques afin de lever l'abstraction qui demeure sur ces derniers.

A cet égard, nous avons proposé une classification de l'ensemble des métriques en trois couches (*cf.* figure 5.1): la première couche est constituée d'un ensemble de métriques dites de base *BMS "Basic Metric Set"*, la deuxième couche est constituée d'un ensemble de métriques dites dérivées "*Derived Metrics*" et enfin, la troisième couche est composée d'un ensemble de métriques définies par l'utilisateur *"User-Defined Metrics"*.

L'ensemble des métriques est composé ainsi: M = BMS ∪ DM ∪ UDM

Les métriques dites de base ou BMS *"Basic Metric Set"* sont des métriques qu'on qualifiera d'atomiques dans le sens où leurs valeurs ne peuvent pas être obtenues par combinaison de celles d'autres métriques. Le travail de (Rolón Aguilar, Ruiz, García, & Piattini, 2006) propose un ensemble d'une soixantaine de métriques dites de base ; ces métriques dénombrent les éléments de base qui constituent un modèle de processus métier, modélisé en BPMN, tels que le nombre d'activités (*NA),* le nombre de tâches (NT), le nombre de flux de séquences entre activités (*NSFA*), etc.

Les métriques de base sont insuffisantes pour répondre aux besoins des diverses évaluations de la qualité. Nous avons donc défini deux autres couches de métriques qui sont :

- DM (Derived Metric Set) dites aussi métriques composées représentent des métriques formulées par des fonctions ayant comme paramètres des métriques de base comme cela est le cas des métriques proposées dans (Gruhn & Laue, 2006), (Reijers & Vanderfeesten, 2004), (Cardoso, Mendling, Neumann, & Reijers, 2006), (Cardoso, Miller, & Arnold, 2002), (Vanderfeesten, Cardoso, & Reijers, 2007) ou encore dans la (section 4.6.1 - section 4.6.4).

- UDM (User-Defined Metrics) sont des métriques définies par les experts de la qualité. L'utilisation de ces métriques se justifie par l'expérience de l'expert. Cette expérience peut reposer sur des études empiriques qui justifient leur utilisation dans un contexte bien précis. Cela peut être pertinent pour la mesure d'un aspect spécifique d'un processus métier. UDM répond donc à des besoins spécifiques et exprime le point de vue de l'expert par rapport à une situation donnée. La définition de ces métriques est généralement basée sur une ou plusieurs métriques appartenant à BMS ∪ DM.

Dans ce qui suit, nous passons en revue, les différentes métriques que nous avons définies pour évaluer les critères de qualité définies ci-dessus.

5.4.6.1 Mesurer l'interopérabilité

L'interopérabilité peut être évaluée en utilisant deux métriques qui sont : l'échange de données "Data exchangeability" (Dex) et la cohérence de l'interface "Interface Consistency" (InC). Elles peuvent être calculées comme suit :

La métrique *Dex* permet de calculer le ratio entre le nombre des objets de données qui sont autorisés à être échangés avec succès avec d'autres processus ou activités au cours des essais notés par NDA et le nombre total d'objets de données à échanger notée TND.

$$Dex = NDA / TND$$

Où: TND = Σ Data Object-In du processus + Σ Data Object-Out du processus.

La métrique *InC* permet de calculer le ratio entre le nombre de messages échangés entre participants noté NoM et le nombre total des activités noté TNA.

$$INC = \Sigma\ NoM / \Sigma\ TNA$$

5.4.6.2 Mesurer la maturité

Pour mesurer le critère de maturité, nous avons adapté deux métriques du domaine des produits logiciels qui sont la densité des défauts *"Fault density"* (*FD*), et le mécanisme de rappel *"Callback"* (*NCB*).

La métrique *FD* dénombre le nombre de défauts détectés lors de l'inspection ou du déploiement du modèle de processus métier. Elle calcule le ratio entre le nombre de défauts détectés *NF* et la taille du modèle de processus métier *N*.

$$FD= NF / N$$

Un mécanisme de rappel ou de *"callback"* en anglais répond à la nécessité de retransmettre directement un message suite à l'invocation simulant une relation synchrone. La métrique callback dénombre le nombre de callbacks dans un modèle de processus métier noté *NCB*.

$$NCB= \Sigma\ callback$$

5.4.6.3 Mesurer la tolérance aux pannes

Pour évaluer ce critère de qualité nous avons défini une métrique appelée gestion des exceptions *"Exception Handled"*. Cette métrique notée *EH* compte le nombre d'exceptions qui existent au sein d'un modèle de processus métier.

$$EH = \Sigma \text{ handled exceptions}$$

5.4.6.4 Mesurer le temps de réponse

Pour mesurer le temps de réponse d'un processus métier, nous avons proposé deux métriques qui sont temps de réponse *"Response time"* notée *RTM* et le débit *"Throughput"* notée *THG*.

La première métrique *RTM* mesure le temps écoulé entre la demande de l'acteur et la réponse fournie par le processus métier. Elle représente la somme des temps d'exécution pour les différentes activités notées RToA:

$$RTM = \Sigma \, RToA$$

La deuxième métrique Thg calcule le nombre d'activités qui peuvent être réalisées avec succès noté *NCA* sur une période de temps donnée (période d'observation) notée *OTP*.

$$THG = NCA \, / \, OTP$$

5.4.7 Analyse et interprétation des mesures obtenues

Après avoir obtenu les différentes valeurs de ces mesures (métriques), ces dernières doivent être interprétées et analysées. Par conséquent, il est intéressant de savoir quelles sont les mesures par rapport auxquelles on peut évoquer le cas d'une bonne (ou d'une mauvaise) qualité du processus métier.

Les valeurs de seuil pourraient être utilisées comme un indicateur fiable de détection de bonne ou de mauvaise qualité dans les modèles de processus métier. En effet, dans (Henderson-Sellers, 1995), l'auteur affirme qu' «une alarme peut se produire chaque fois que la valeur d'une mesure interne spécifique dépasse une certaine valeur prédéfinie».

En se basant sur ce principe, les modeleurs et les experts de qualité doivent présenter pour chaque métrique une valeur de seuil $m_{threshold} \in M_{thresholds}$ qui reflète sa valeur optimale. Cette valeur peut être fournie à la suite d'études empiriques.

La comparaison entre $m_i \in M$ et $m_{threshold} \in M_{thresholds}$, fournit donc une évaluation de la dimension de la qualité.

A partir de ce principe, nous pouvons définir des niveaux de qualité de processus métier sur la base de ces valeurs de seuils:

- Niveau 1: il ya 10% de probabilité pour que la qualité d'un modèle de processus métier soit considérée comme acceptable;
- Niveau 2: il ya 30% de probabilité pour que la qualité d'un modèle de processus métier soit considérée comme acceptable;
- Niveau 3: il ya 50% de probabilité pour que la qualité d'un modèle de processus métier soit considérée comme acceptable;

- Niveau 4: il ya 70% de probabilité pour que la qualité d'un modèle de processus métier soit considérée comme acceptable.

Par conséquent, un Niveau 2 par exemple peut être interprété comme suit :

Si le nombre de nœuds d'un modèle est compris entre 8 et 20, le diamètre de ce modèle est compris entre 5 et 12, la profondeur est comprise entre 1 et 5, le coefficient de connectivité est compris entre 0,4 et 2,80 et le caractère cyclique est compris entre 0,10 et 0,89 alors la probabilité pour que la qualité d'un modèle de processus métier soit considérée comme acceptable est d'environ 30%.

5.5 Analyse de l'impact qualitatif du changement des processus métier

La qualité d'un modèle de processus métier doit être évaluée durant tout son cycle de vie et plus particulièrement après chaque changement de processus métier. Dans cette section, nous nous intéressons tout particulièrement à l'impact du changement structurel d'un modèle de processus métier sur la qualité ou ce que nous appelons l'impact qualitatif du changement.

En effet, outre l'impact comportemental (Sun & Jiang, 2009), un impact qualitatif peut en résulter d'un changement de processus métier. Généralement, il est observé que la qualité du modèle de processus métier tend à s'éroder avec l'augmentation de la taille et de la complexité due à l'ajout de nouvelles activités. Autrement dit, le nombre croissant d'activités augmente le nombre de connexions ce qui peut conduire à des difficultés de compréhensibilité et de maintenabilité de ce modèle.

Dans le chapitre 4, nous avons quantifié un changement de processus métier (notée Δ) comme la différence entre le modèle de processus métier initiale *S* et le modèle de processus métier résultant *S'* noter comme suit:

$$S' = S + \Delta$$
$$\Delta = | S - S' |$$

Par conséquent, tout changement structurel *(ΔS)* opéré sur un modèle de processus métier peut causer un changement qualitatif *(ΔQ)* sur les critères qualitatifs correspondants. Ainsi, la notation $\Delta S \Longrightarrow \Delta Q$ signifie que le changement structurel *ΔS* implique un changement qualitatif *ΔQ*. Cette affirmation vient du fait qu'un changement de processus métier peut affecter les métriques de qualité concernant ce dernier. L'interdépendance des attributs de la qualité peut alors conduire à une mauvaise situation dans laquelle la détérioration d'une métrique de qualité peut avoir un impact défavorable sur d'autres métriques.

Une connaissance exhaustive de la qualité du modèle de processus métier peut donc aider à mieux tracer la propagation de l'impact du changement de façon progressive entre métriques, critères et facteurs de la qualité du processus métier. Elle permet de déterminer le changement qualitatif ΔQ résultant d'un changement structurel de processus métier, par la propagation progressive de ce changement aux critères prédécesseurs dans le graphe de la qualité, aux facteurs correspondants et ainsi de suite jusqu'à la qualité globale ou générale du processus métier comme le montre la figure 5.2.

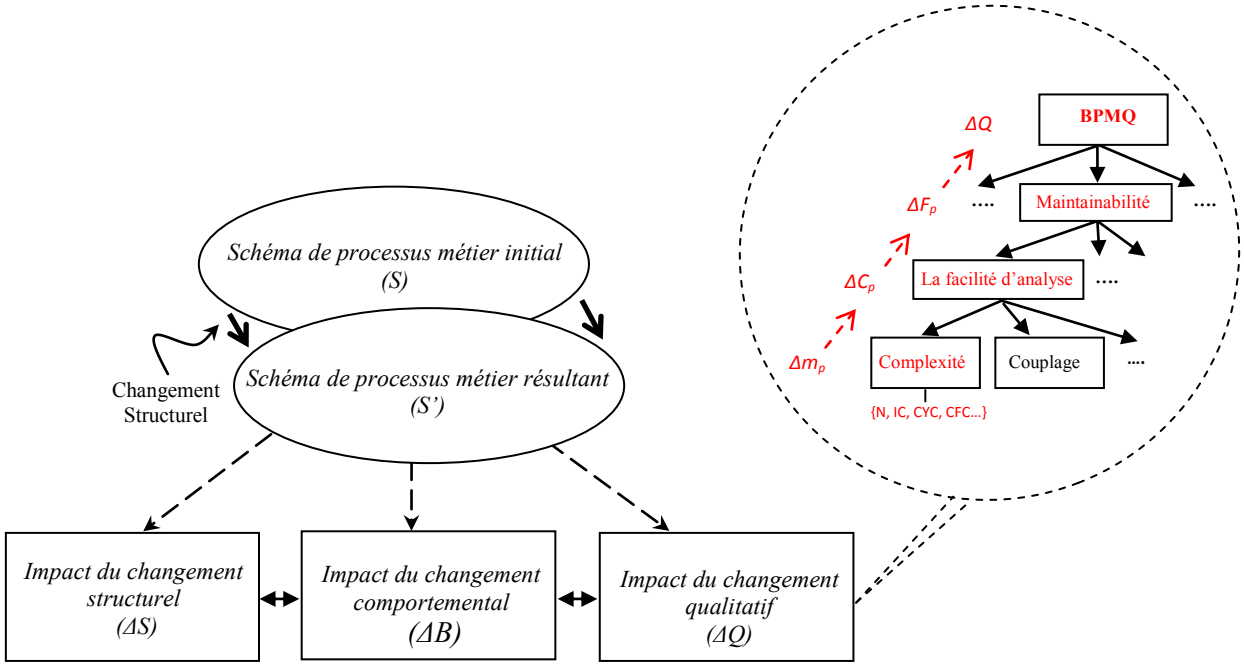


Figure 5.2 *Relations entre l'impact structurel, comportemental et qualitatif du changement.*

Considérons un ensemble de métriques $M = \{m_1, m_2, m_3,\ldots, m_n\}$ associées à un modèle de processus métier *S*. Les valeurs associées à ces métriques peuvent être représentées par l'ensemble $V = \{v_1, v_2, v_3, \ldots v_n\}$. Un changement structurel ΔS appliqué sur un modèle de processus métier *S* peut résulter en de nouvelles valeurs de ses métriques associées que l'on notera $V' = \{v'_1, v'_2, v'_3, \ldots, v'_n\}$. Ceci est formulé par :

$$m_1, m_2, m_3,\ldots, m_n\ (S) = (v_1, v_2, v_3, \ldots v_n)$$

$$\Longrightarrow M(S) = V$$

Après le changement, ΔQ(S') = ΔQ(S) + ΔQ(change), alors nous obtenons la valeur de V' telle que :

$$m_1, m_2, m_3,\ldots, m_n\ (S') = (v'_1, v'_2, v'_3, \ldots v'_n)$$

$$\Longrightarrow M(S') = V'$$

La différence des valeurs de métriques ($\Delta M = V' - V$) dénote une variation des critères qualitatifs correspondants. La différence qui correspond aux valeurs de métriques Δm_i, Δm_{i+1}, Δm_n représente donc le changement du critère de qualité qu'on notera ΔC.

Le changement d'un critère peut affecter la valeur d'autres critères associés (ΔC_j, ΔC_{j+1}, ΔC_n) du même niveau (*cf.* section 5.1). Cette différence identifiée au niveau des attributs qualitatifs matérialise l'impact qualitatif.

5.5.1 Dépendances entre les critères de qualité

Hormis les relations de dépendance verticales entre les différents facteurs, critères et métriques expliquées dans la section 4, il peut exister des dépendances horizontales entre les facteurs, critères et métriques de la qualité d'un même niveau de raffinement ou d'abstraction comme le montre la figure 5.3.

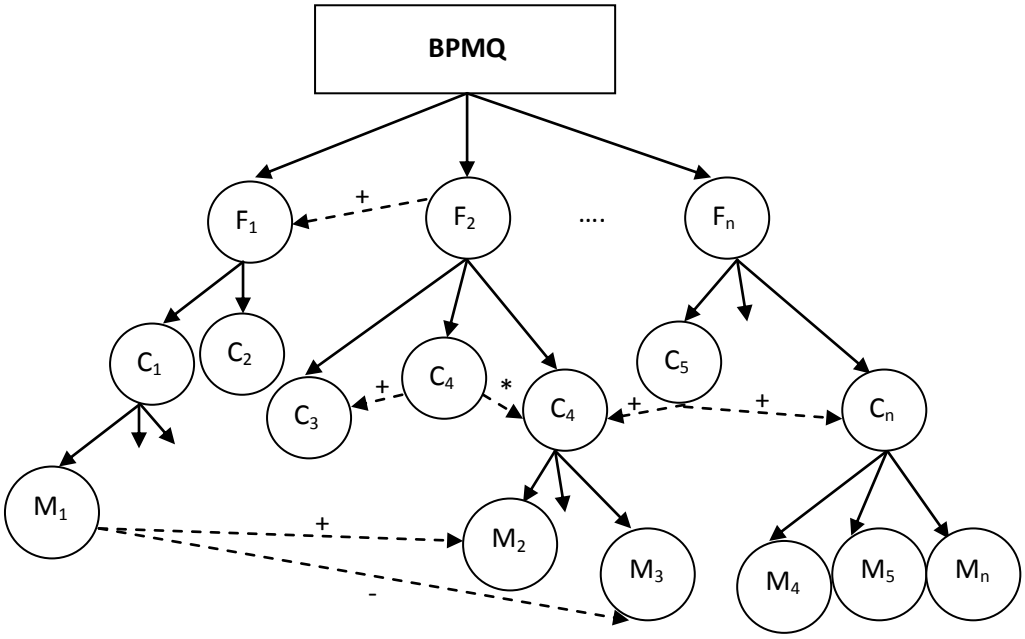


Figure 5.3 *Les dépendances horizontales dans l'évaluation de la qualité des processus métier.*

Dans cette section, nous définissons les relations horizontales responsables de la propagation de l'impact du changement entre les facteurs et critères de la qualité d'un même niveau d'abstraction. La présentation de ces relations constitue un moyen d'analyse des répercussions causales de modification d'un attribut de la qualité sur les autres attributs de la qualité. En effet, ces dernières modélisent la complémentarité ou la contradiction de deux attributs dans l'évaluation de la qualité. Il est ainsi possible de spécifier des relations de types : l'amélioration de la qualité d'un attribut A_1 participe à améliorer (respectivement détériorer) celle d'un autre attribut A_2 et *vice-versa*.

Durant l'évolution d'un facteur de la qualité, les autres facteurs qui lui sont reliés peuvent également évoluer. Cela correspond à l'existence ou l'absence de la propagation de l'impact entre les nœuds du graphe de la qualité. Par exemple, il est démontré par les travaux de Laura Sánchez-González *et al.* (Sánchez-González, García, Mendling, Ruiz, & Piattini, 2010) , (Sánchez-González, García, Mendling, & Ruiz, 2010), (Sánchez-González, Ruiz, García, & Piattini, 2011) que les métriques qui évaluent les deux critères de qualité qui sont la compréhensibilité *"understandability"* et la facilité de modification *"modifiability"* évoluent de manière totalement corrélées.

Partant de ce constat, nous pouvons définir l'ensemble U des arcs du graphe de la qualité par deux sous-ensembles disjoints : l'ensemble V représentant les relations verticales entre facteurs, critères et métriques, et l'ensemble H des arcs reliant des nœuds d'un même niveau de raffinement. L'orientation de l'arc indique la nature de la propagation de l'impact qualitatif qui peut être libellé par les symboles +, −, ou * indiquant respectivement un impact positif, négatif ou un changeant.

L'ensemble des arcs est alors défini par U = V ∪ H, où H est l'ensemble des arcs horizontaux. H représente les relations de dépendance entre des nœuds d'un même niveau, telle que :

$$H = H^{+} \cup H^{-} \cup H^{*}$$

Où, H^+, H^-, et H^* désignent respectivement les arcs libellés par le symbole « + », « − », et « * ».

Donc, $U = V \cup H^+ \cup H^- \cup H^*$ telle que :

$(f_i, f_j) \in F$:

- $(Qt(f_i), Qt(f_j) \in H^+) \Leftrightarrow Qt(f_i)$ présente une évolution favorable pour $Qt(f_j)$;
- $(Qt(f_i), Qt(f_j) \in H^-) \Leftrightarrow Qt(f_i)$ présente une évolution défavorable pour $Qt(f_j)$;
- $(Qt(f_i), Qt(f_j) \in H^*) \Leftrightarrow Qt(f_i)$ présente une évolution changeante de $Qt(f_j)$;

Où Qt(fi) désigne l'estimation du degré avec lequel le facteur Fi est assuré (une valeur quantifiant le facteur Fi).

$\forall\ Qt(c_i), Qt(c_j) \in Qt(C)$:

- $(Qt(c_i), Qt(c_j) \in H^+) \Leftrightarrow Qt(c_i)$ présente une évolution favorable pour $Qt(c_j)$;
- $(Qt(c_i), Qt(c_j) \in H^-) \Leftrightarrow Qt(c_i)$ présente une évolution défavorable pour $Qt(c_j)$;
- $(Qt(c_i), Qt(c_j) \in H^*) \Leftrightarrow Qt(c_i)$ présente une évolution changeante de $Qt(c_j)$;

Où Qt(ci) désigne l'estimation du degré avec lequel le critère Ci est assuré (une valeur quantifiant le critère Ci).

5.6 Conclusion

La gestion de la qualité des processus métier et plus précisément celle des modèles de processus métier a fait l'objet de multiples travaux de recherche récemment. Le but étant de mettre en place un mécanisme efficace qui permet d'améliorer la compréhension, la fiabilité et la réutilisation de ces modèles durant leur cycle de vie, et plus particulièrement, après chaque changement de processus métier.

En effet, l'amélioration du contrôle de l'évolution des processus métier en évitant la dégradation de la qualité dépend fortement de la compréhension des dépendances des différents éléments d'un processus métier incluant ceux relatifs à la qualité.

Dans ce chapitre, nous avons proposé un cadre exhaustif inspirée par la norme ISO/ CEI 9126-1 représentent à la fois une contribution à la maturité des travaux existants à l'égard de l'évaluation et de l'amélioration de la qualité des processus métier et un moyen permettant de traiter la problématique de l'analyse de l'impact du changement selon les points de vue structurel et qualitatif. L'approche proposée permet une meilleure traçabilité de la propagation de l'impact du changement.

Nous avons également représenté des relations horizontales entre attributs de qualité en plus des relations verticales permettant de définir ainsi des liens de contradiction ou plus généralement de causalité en matière de contribution à la qualité générale du processus métier.

CHAPITRE 6

Prototype de validation

6.1 Introduction

Afin de valider l'ensemble des contributions de nos travaux proposés tout au long de ce manuscrit, nous avons mis en œuvre une plate-forme intégrant différents modules ou composants implémentant chacun des aspects évoqués dans cette thèse. Cette plate-forme qui est intégrée à un environnement de modélisation BPMN permet entre autres de réaliser les fonctionnalités suivantes : (1) vérifier la cohérence des processus métier après chaque changement, (2) de vérifier la conformité de ces modèles avec l'ensemble des règles de conformité, (3) d'évaluer la qualité des modèles de processus métier tout au long de leur cycle de vie et plus particulièrement, après chaque changement, et enfin (4) de propager l'impact du changement des processus métier.

Notre plate-forme ainsi que tous les modules qui la constituent sont intégrés au sein de l'IDE *"Integrated Development Environment"* Eclipse[24] sous forme de plug-ins. Le choix d'Eclipse a été dicté par le fait que c'est un IDE open source largement utilisée pour construire des plates-formes de développement ouvertes et extensibles constituées d'outils et de runtimes destinés à la construction, au déploiement et à la gestion de logiciels mais aussi par l'environnement de modélisation BPMN utilisé comme noyau pour notre plate-forme appelé BPMN2 Modeler[25] qui est implémenté sous forme d'un plug-in au sein de ce dernier.

Dans ce chapitre, nous présentons la plate-forme appelée BPMN-CM *"Business Process Modeling Notation Change Management"* en détaillant les différents modules qui constituent son architecture. Le premier module de base est le plug-in BPMN2 Modeler qui permet de modéliser un processus métier moyennant la notation BPMN. Ce module est le composant central de notre prototype. Le second module qui est composé de deux sous-modules permet à la fois de vérifier la cohérence et la conformité des modèles de processus métier après chaque changement et cela, à travers l'utilisation du plug-in EpiSpin[26] comme model chercker. En effet, après avoir transformé automatiquement le modèle de processus métier à vérifier en Promela, EpiSpin vérifie l'absence de toute erreur structurelle susceptible de causer des incohérences dans ce modèle. La même démarche est utilisée pour vérifier la constitution de ce modèle comme étant satisfaisante au regard des règles de conformité.

Un autre modules du prototype a été conçu et implémenté pour gérer la qualité des processus métier tout au long de leur cycle de vie et plus particulièrement, l'impact qualitatif qui peut être le résultat d'un changement de processus métier. Enfin un dernier module qui permet une analyse *a priori* de l'impact de ce changement. Le résultat de cette analyse est mis en évidence par une visualisation de l'impact sur les différents éléments du modèle de processus métier.

Ce chapitre est organisé comme suit : la section 6.2 résume l'architecture et la conception générale du prototype de validation ainsi que les détails de chacun des modules qui composent notre plate-forme. La section 6.3 présente les résultats de notre approche en considérant un scénario illustratif de processus de remboursement des frais de fonction-

[24] http://www.eclipse.org/pde
[25] http://eclipse.org/bpmn2-modeler/
[26] http://epispin.ewi.tudelft.nl/index.html

nement. La section 6.4 conclut le chapitre et discute brièvement les perspectives du développement de notre prototype.

6.2 Architecture Globale du prototype de validation

Notre plate-forme appelée BPMN-CM *"Business Process Modeling Notation Change Management"* permet à la fois de modéliser un processus métier à travers l'utilisation du plug-in BPMN2 Modeler, de vérifier la cohérence et la conformité d'un modèle de processus métier et enfin d'analyser les différents impacts structurels et qualitatifs résultant d'un changement de processus métier.

BPMN-CM est construit à l'aide d'Eclipse Plug-in Development Environment (PDE) qui représente un outil open source incontournable de développement de logiciel, car il offre une plate-forme extensible qui permet l'ajout de nouvelles fonctionnalités grâce à des plug-ins. En plus, cette plate-forme est vue comme une collection de projets open sources développés et maintenus par une large communauté de développeurs. Chaque projet a comme but de proposer un ensemble spécifique de plug-ins et d'outils de développement. Parmi ces projets, on peut citer Business Intelligence and Reporting Tools[27] qui permet de générer des rapports, SOA platform project[28] qui vise à proposer un ensemble de plug-ins et d'outils de développement orientés web services (des éditeurs BPMN, BPEL, un moteur BPEL ...etc.) ou encore Eclipse Modeling Project[29] qui vise à proposer un ensemble de plug-ins et d'outils de développement orientés modèle.

Dans notre travail, nous nous sommes intéressés au projet SOA platform project et plus particulièrement à un de ces plug-ins appelé BPMN2 Modeler. Ce dernier représente le noyau de notre prototype BPMN-CM, il permet de modéliser un processus métier à l'aide de la notation BPMN (*cf.* figure 6.1).

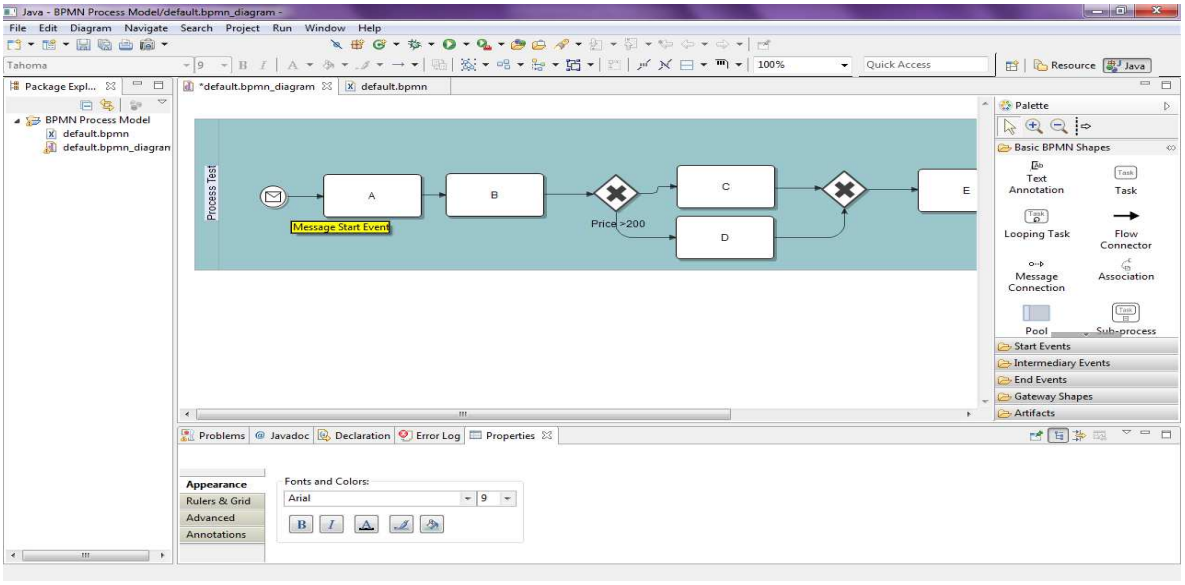


Figure 6.1 *Le plug-in BPMN2 Modeler.*

[27] http://projects.eclipse.org/projects/birt
[28] http://projects.eclipse.org/projects/soa
[29] http://projects.eclipse.org/projects/modeling

Notre plate-forme solution BPMN-CM est donc composée de quatre plug-ins :

- Un plug-in de modélisation appelé BPMN2 Modeler permettant à un analyste métier de modéliser un processus métier, ce plug-in représente le noyau de notre prototype ;
- Un plug-in de vérification appelé BPMNVT "*Business Process Modeling Notation Verification Tool* " composé de deux modules qui sont BPMN2PROMELA *"BPMN to Promela"*, CR2LTL *"Compliance Rules to LTL"* permettant respectivement de transformer un modèle BPMN en modèle Promela et d'exprimer l'ensemble des règles de conformité en LTL et cela, afin de vérifier à la fois la cohérence d'un modèle de processus métier et la conformité de ce dernier avec l'ensemble de ces règles ;
- Un plug-in de gestion de qualité appelé BPMNQ *"BPMN Quality"* permettant de gérer la qualité d'un processus métier durant tout son cycle de vie et plus particulièrement après chaque changement ;
- Un plug-in d'analyse d'impact du changement de processus métier BPMN-CPA *"BPMN Change Propagation Analyser"* permettant de propager l'impact du changement des processus métier.

L'architecture interne montrant l'interaction de ces plug-ins est schématisée par la figure 6.2. Nous discuterons par la suite les détails de l'implémentation de ces quatre plug-ins.

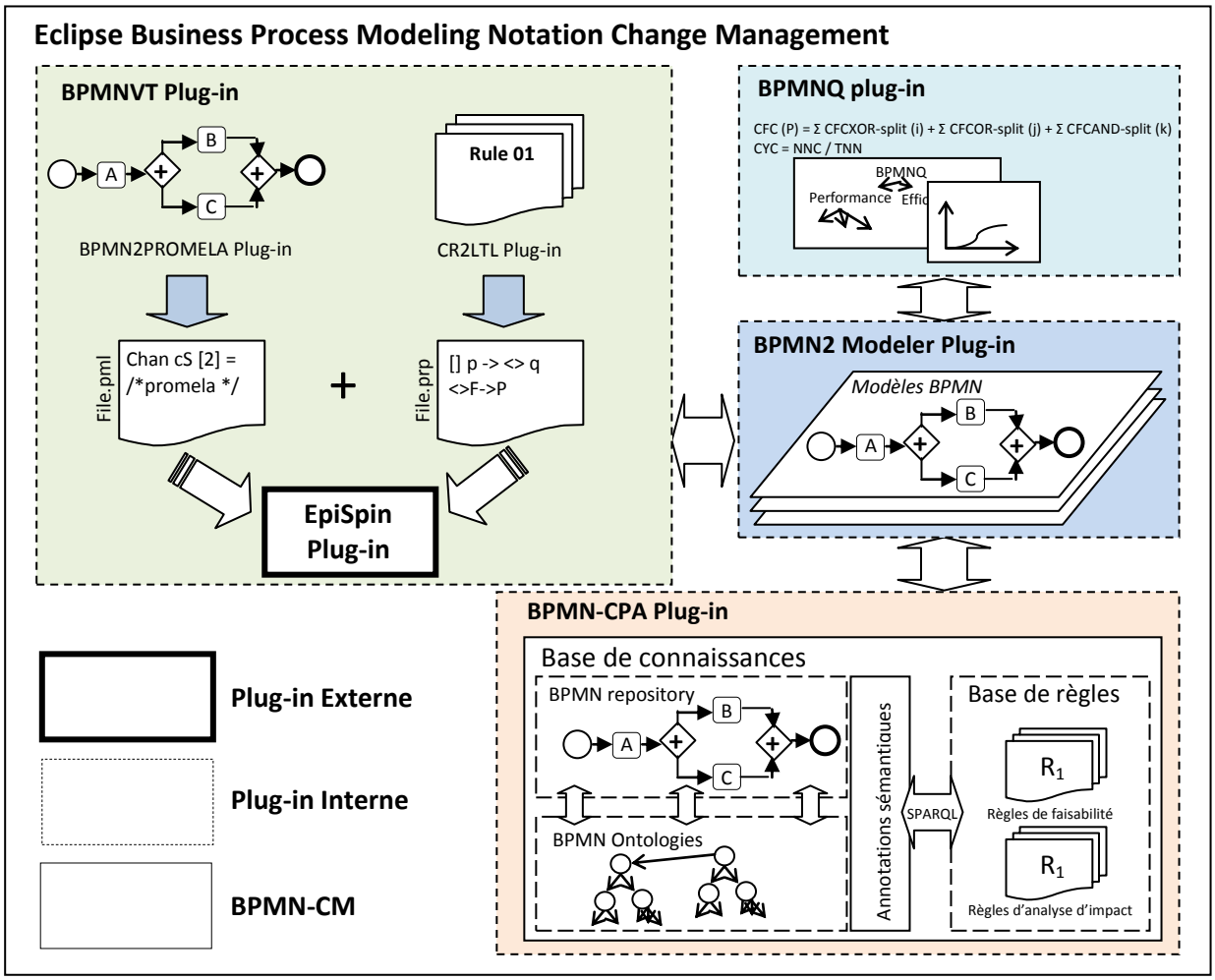


Figure 6.2 *L'architecture générale de la plate-forme BPMN-CM.*

6.2.1 Le plug-in BPMN2 Modeler

Le plugin BPMN2 Modeler représente le noyau de notre plate-forme BPMN-CM. Ce plug-in permet de construire des modèles de processus en utilisant une palette de symboles BPMN (*cf.* figure 6.1). Cet outil respecte globalement les spécifications de la notation BPMN 2.0.

BPMN2 Modeler génère deux fichiers en sortie, le premier fichier, au format XML, détaille la logique du processus (*cf.* figure 6.3), et le second comporte les éléments relatifs à l'aspect graphique du processus (*cf.* figure 6.4). Ces deux fichiers sont le point de départ de notre plate-forme.

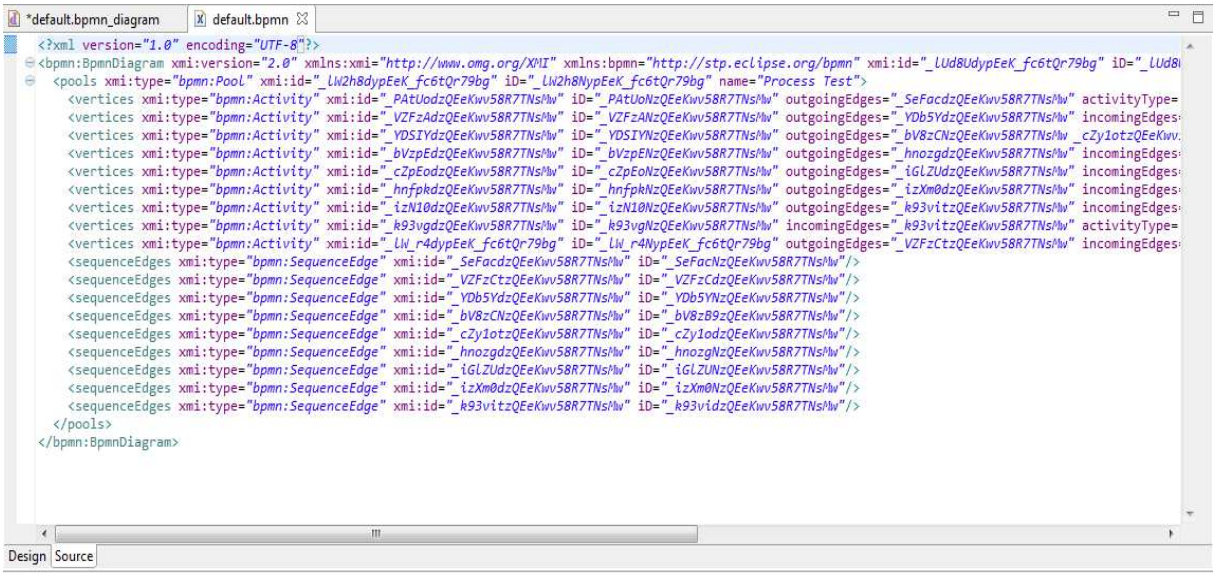


Figure 6.3 *Extrait du fichier d'entrée « Description logique ».*

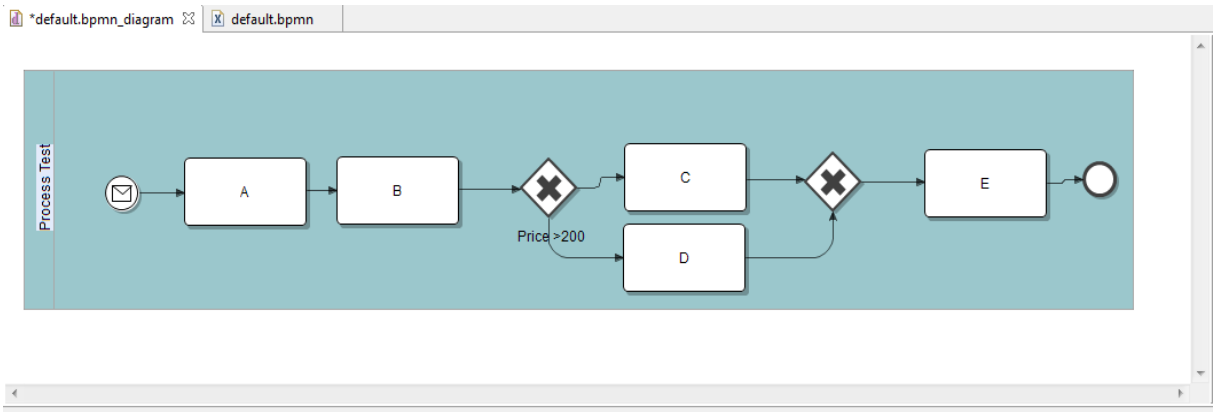


Figure 6.4 *Extrait du fichier d'entrée « Description graphique».*

6.2.2 Le plug-in BPMNVT

Afin de mettre en œuvre notre approche de vérification formelle dans un contexte pratique, nous avons mis en place un plugin de vérification appelé BPMNVT *"BPMN Verification Tool"* composé de deux modules qui sont BPMN2PROMELA *"BPMN to Promela"* et CR2LTL *"Compliance Rules to LTL"*. L'architecture de BPMNVT est présentée dans la figure 6.5.

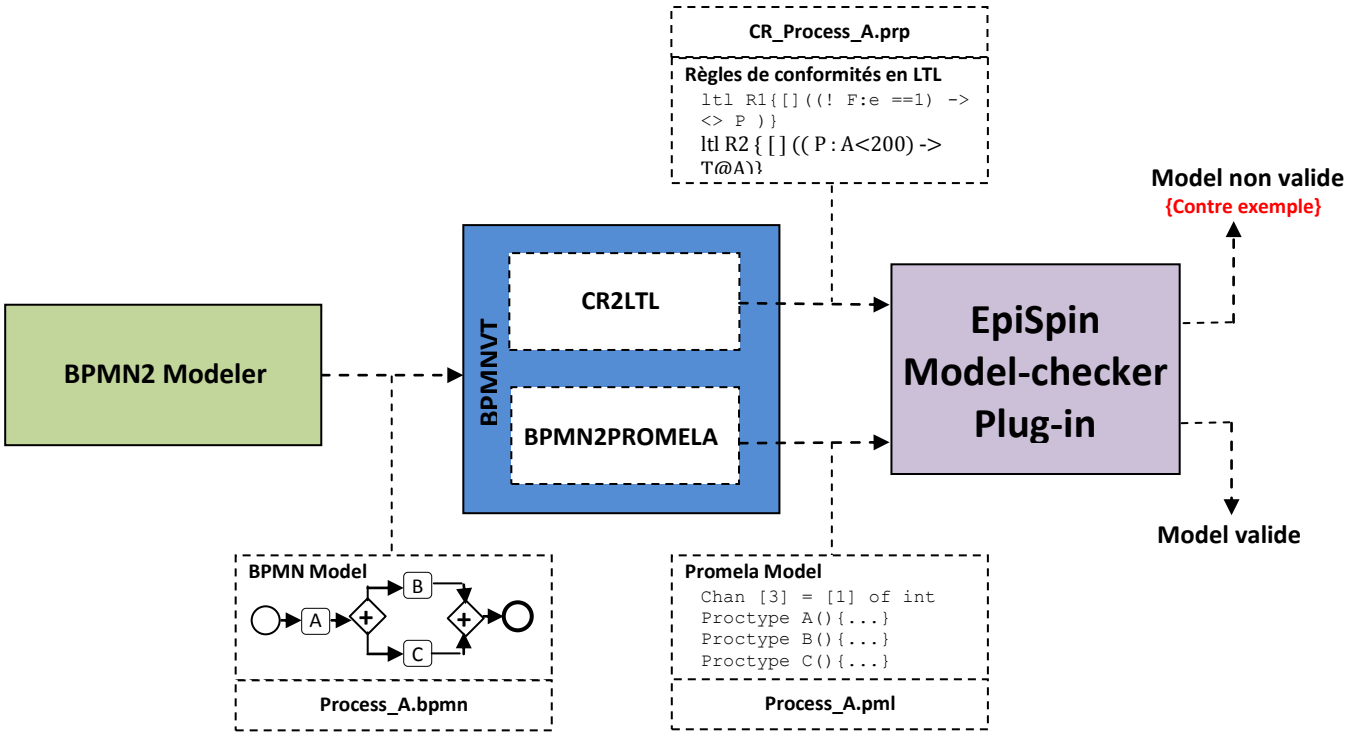


Figure 6.5 *L'architecture du plug-in BPMNVT.*

6.2.2.1 BPMN2PROMELA

Le plugin BPMN2PROMELA est un traducteur qui permet de transformer un modèle BPMN en un modèle Promela. Le fichier **.pml* est le résultat de cette traduction. Ce fichier contient le code Promela correspondant au modèle BPMN source qu'on veut vérifier et dont le contenu est fourni comme entrée pour le plugin le model-checker EpiSpin qui servira comme outil de vérification formelle (*cf.* figure 6.5).

Le parsing ou l'analyse des modèles BPMN par le traducteur BPMN2PROMELA est basée sur l'association d'un ensemble de routines sémantiques (des règles de réécriture) pour la construction d'arbres syntaxiques abstraits ou AST *"Abstract Syntax Tree"* dont le parcours et l'exploitation serviront à la production du code Promela. Précisément, la syntaxe Promela sera organisée en objets BPMN correspondant (la Table 3.2 du chapitre 3 contient la correspondance entre les différents éléments BPMN et la syntaxe Promela). Ainsi les instructions Promela telles que proctype, bloc, if-fi bloc, etc. représentent les branches de l'arbre syntaxique et les instructions Promela telles que les variables, les canaux, envoi/ réception de messages représentent les feuilles de cet arbre.

La figure. 6.6 (a) nous donne un aperçu sur la façon dont le modèle BPMN est parsé en arbre syntaxique, puis mappé en fragments de code PROMELA. La figure 6.6 (b) montre l'arbre syntaxique étiqueté de (1 à 10) et la figure 6.6 (c) représente le code PROMELA correspondant.

Dans cette traduction chaque activité est transformée en processus Promela par exemple : `proctype A(){…}`, les flux de séquences en canaux de communication Promela par exemple : `chan ch[2]= [1] of {int}`. Les flux de messages entre les différents processus sont représentés en utilisant des entiers dans Promela.

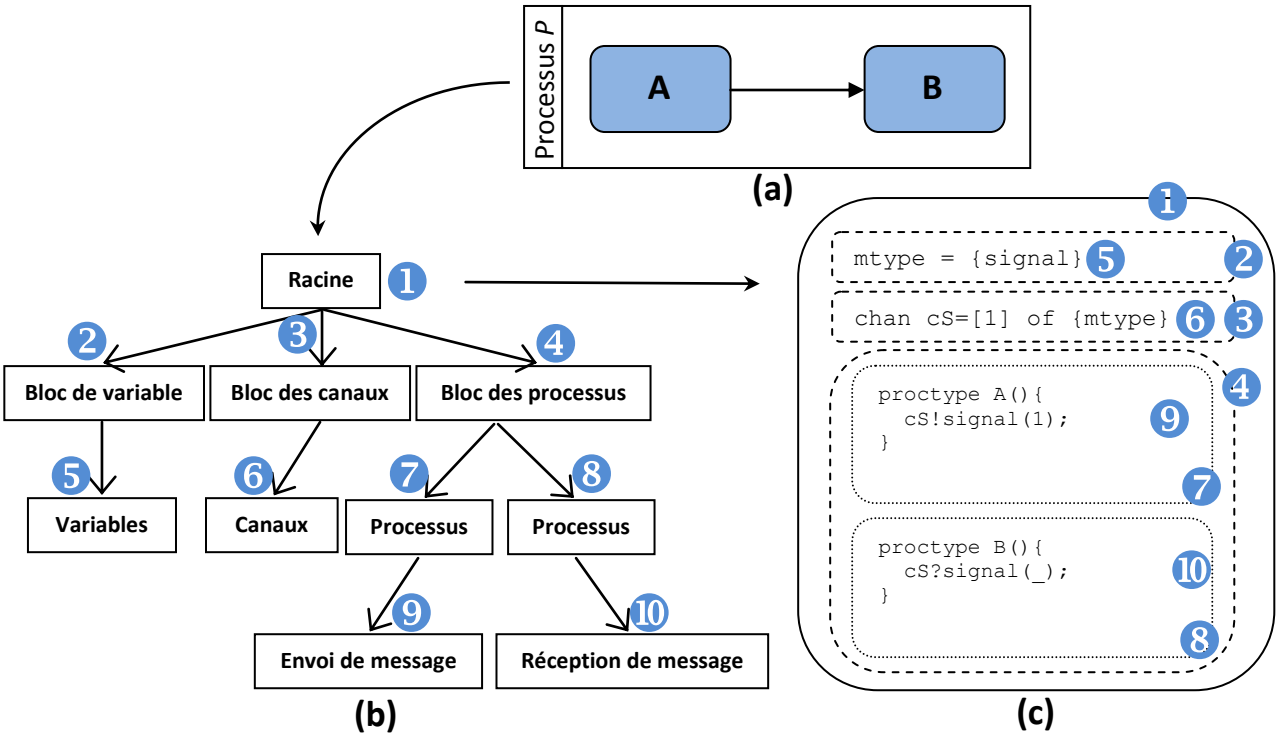


Figure 6.6 *Exemple de l'arbre syntaxique.*

Le fichier `config.properties` sert à configurer le traducteur. Il permet aux utilisateurs à titre d'exemple, de spécifier la taille du buffer des canaux de communication `buffer.size`, le chemin de stockage des fichiers Promela générés `files.locator`, les types de messages `message.type` échangés dans les canaux (`bits`, `bool`, `byte`, `short`, `int`) ou encore le nom du modèle BPMN à traduire `BPMNmodel.name`. Cette configuration est prise en considération par le traducteur à chaque exécution (*cf.* figure 6.7).

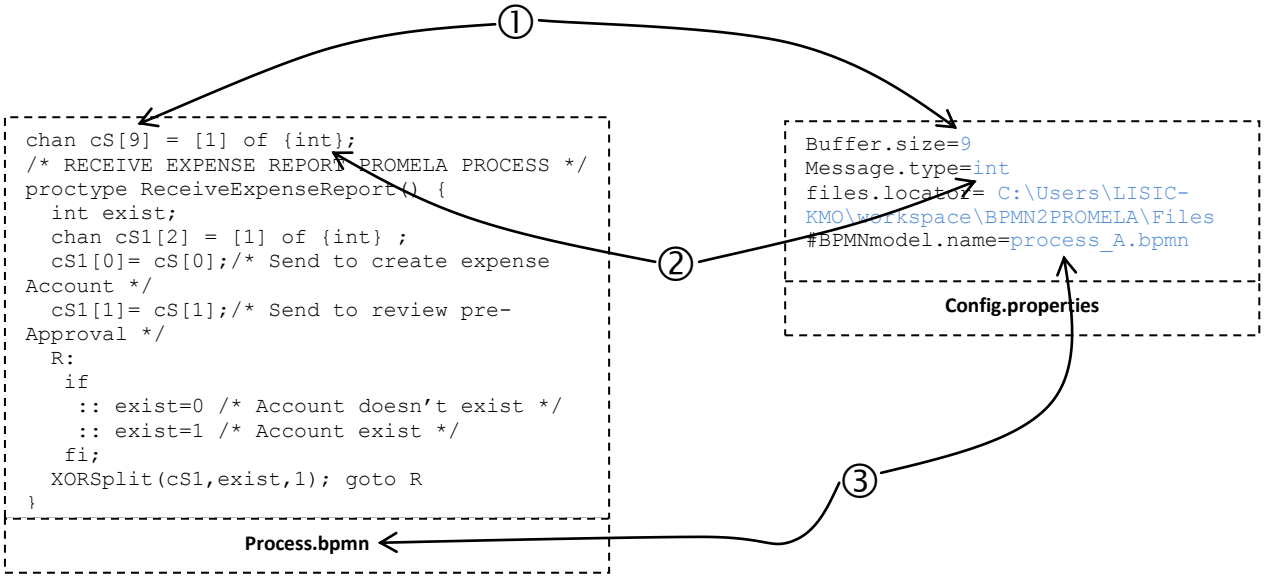


Figure 6.7 *Fichier de configuration de BPMNVT.*

6.2.2.2 CR2LTL

CR2LTL est un module qui propose une palette composée d'un ensemble de connecteurs temporels et booléens permettant d'exprimer ainsi des formules LTL graphiquement (*cf.* figure 6.8). L'objectif de ce module étant de fournir un fichier **.prp* a partir de la représentation graphique de l'ensemble des règles de conformité exprimées en LTL et cela afin de fournir ce dernier comme entrée pour EpiSpin en plus du fichier **.pml* contenant le code Promela correspondant au modèle BPMN dont on veut vérifier la conformité comme le montre l'architecture de BPMNVT dans figure 6.5.

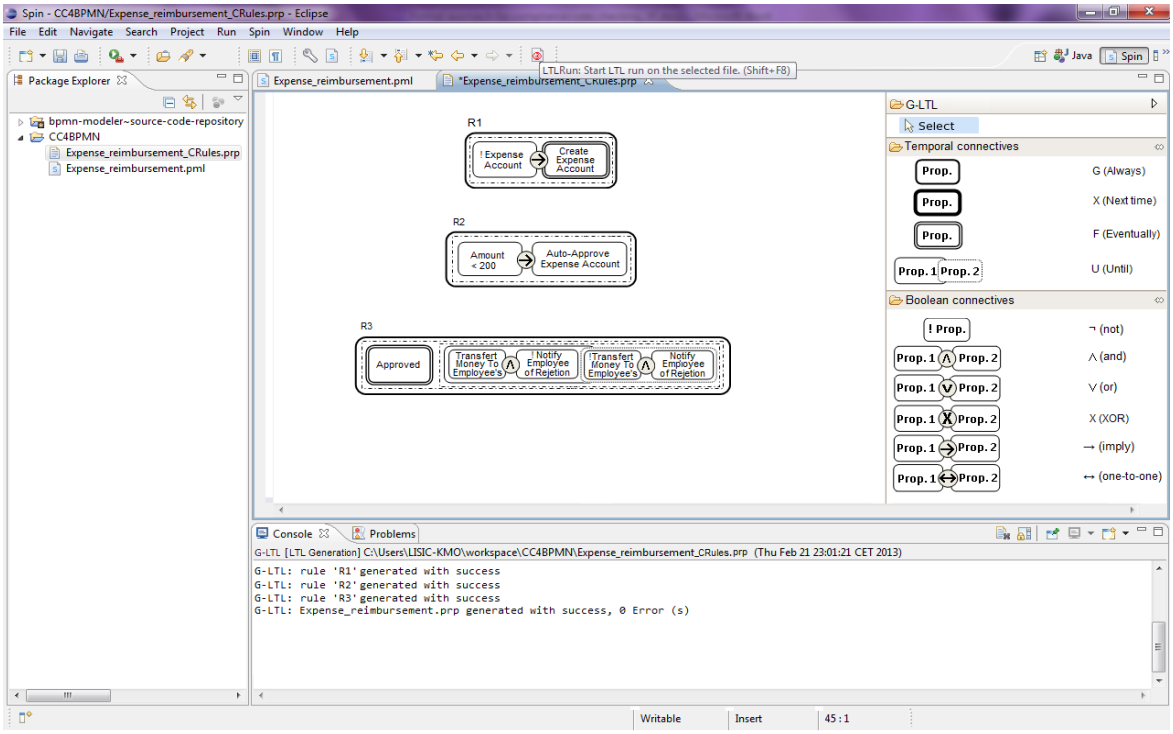


Figure 6.8 *Le module CR2LTL.*

6.2.3 Le plug-in BPMNQ

Le plug-in BPMNQ ou *"BPMN Quality"* permet de gérer la qualité des modèles de processus métier durant tous leurs cycles de vie, et plus particulièrement après chaque changement. Il est composé de trois modules : (1) un module de calcul, (2) un module d'évaluation et d'interprétation et (3) un module de représentation (*cf.* figure 6.9).

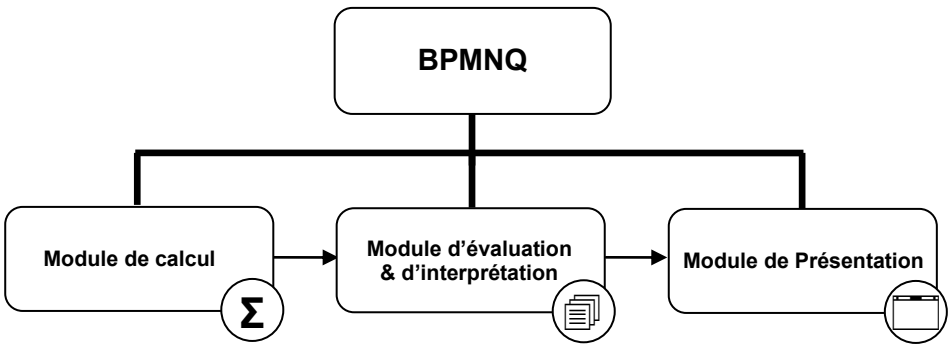


Figure 6.9 *Architecture du plug-in BPMNQ.*

Le premier module sert à calculer l'ensemble des métriques mises en place dans BPMNQ. Cet ensemble est classé en trois couches: (1) les métriques dites de base permettent de dénombrer les différents éléments qui composent le modèle BPMN tel que le nombre d'activités (*NA),* le nombre de tâches (NT), le nombre de flux de séquence entre activités (*NSFA*), etc, (2) Les métriques dites dérivées qui représentent des métriques formulées par des fonctions ayant comme paramètres des métriques de base tel que la complexité d'un flux de contrôle au sein d'un modèle BPMN (CFC), la longueur du modèle BPMN (N) ou encore l'échange de données au sein d'un modèle BPMN (Dex).

BPMNQ donne aussi la possibilité aux modeleurs et aux experts de qualité d'ajouter (3) des métriques *ad-hoc* qui se basent essentiellement sur l'ensemble des métriques de base et dérivées et d'associer chacune entre eux aux critères de qualité définis dans le chapitre 5 (*cf.* figure 6.10).

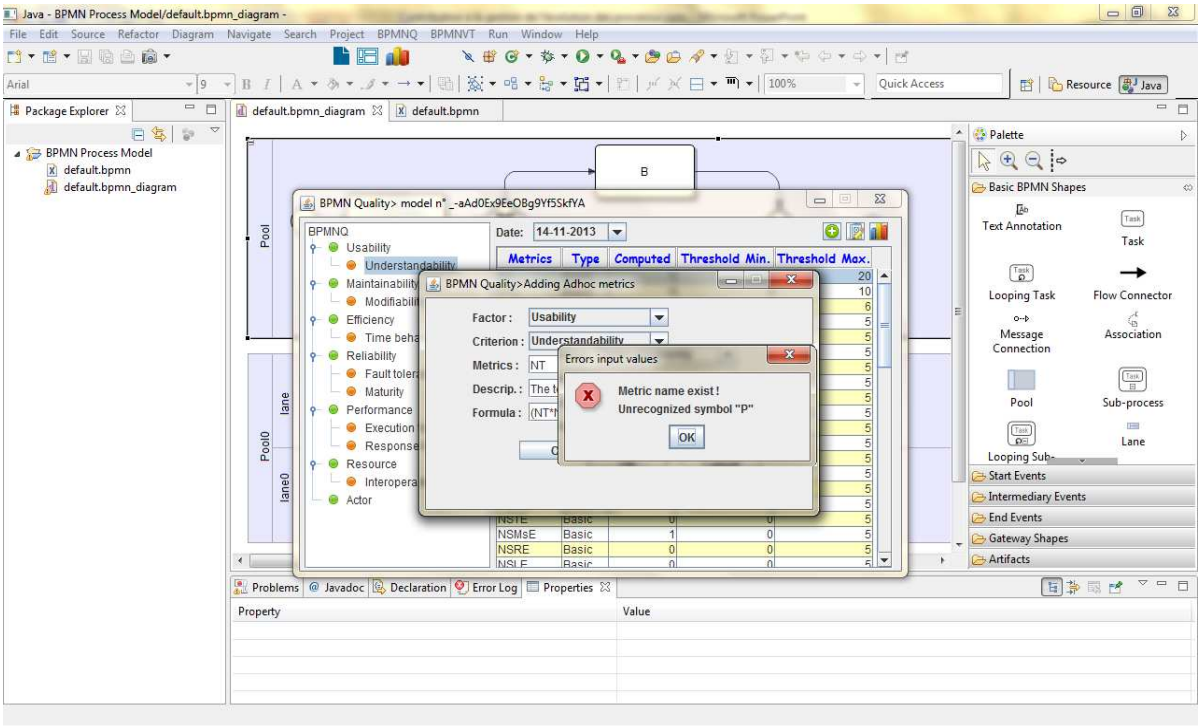


Figure 6.10 *Ajout de métriques Ad-hoc dans le plug-in BPMNQ.*

Le module d'évaluation et d'interprétation permet de comparer les différentes valeurs des métriques obtenues avec les valeurs de seuil introduites ultérieurement par les modeleurs et les experts de qualité. En effet, grâce à l'interface présentée dans la figure 6.11, l'interprète affiche pour chaque dimension et critère de notre modèle qualitatif les valeurs des métriques qui lui sont associées. Ce module permet aussi aux modeleurs et aux experts de qualité de fournir aussi pour chacune des métriques une valeur de seuil reflétant sa valeur optimale, cette valeur peut être fournie à la suite d'études empiriques.

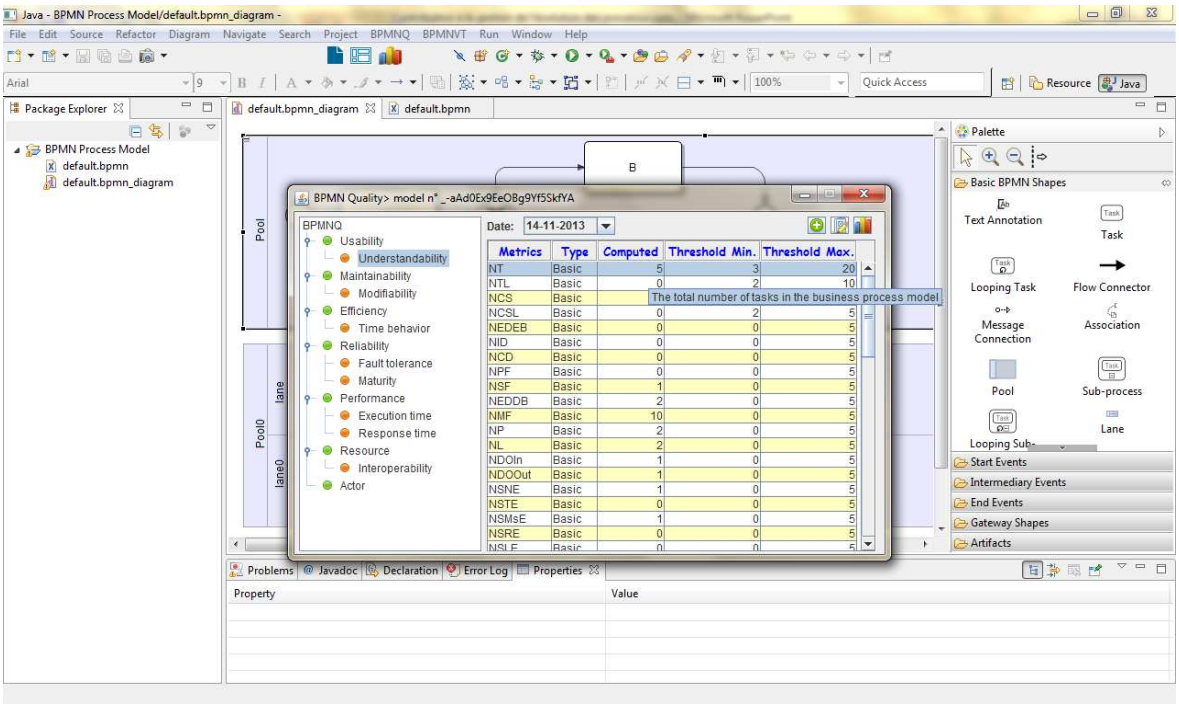


Figure 6.11 *Évaluation des métriques d'un modèle BPMN par plug-in BPMNQ.*

L'évaluation de chaque critère de la qualité Q_c se base sur la comparaison entre les différentes mesures obtenues σ_i (i=0,…,n) et les différentes valeurs de seuils σ'_i (i=0,…,n).

Pour une meilleure visibilité des différentes mesures obtenues, le module de présentation offre la possibilité de représenter les différents résultats sous forme de graphe d'évaluation qui peut être visualisé comme le montre la figure 6.12. L'intervalle de seuil est représenté en rouge, tandis que la variation de la valeur de la métrique dans le temps est représentée en bleu.

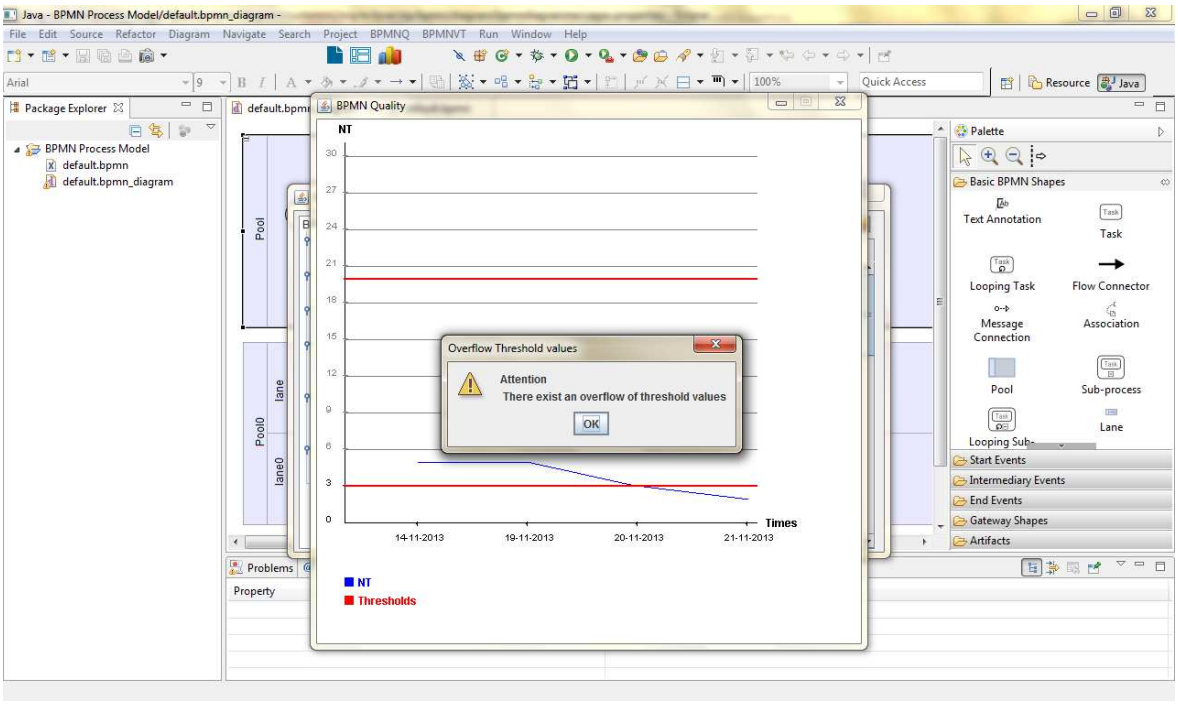


Figure 6.12 *Graphe d'évaluation de la métrique N.*

6.2.4 Le plug-in BPMN-CPA

Le plug-in BPMN-CPA ou *"BPMN Change Propagation Analyser"* qui est en cours de développement permet de propager l'impact du changement des processus métier.

Ce plugin est composé de deux modules : une base de connaissances (*cf.* chapitre 4) qui a pour principale fonction la gestion des méta-informations sur les processus métier et une base de règles qui permettra la mise en place du processus de propagation de l'impact du changement.

L'implémentation de la base de règles est basée sur le moteur de règles orienté objet Drools[30]. Ce moteur permet la gestion de l'ensemble des règles de propagation de l'impact du changement des processus métier proposées dans le chapitre 4. En effet, Drools est une plate-forme de gestion de règles métier fournissant un atelier intégré pour la définition et l'exécution de règles. Il permet également la définition et l'exécution de workflow ainsi que la gestion des événements. L'ensemble des règles de propagation de l'impact sont invoquées de façon interactive au cours de la manipulation des éléments du modèle BPMN (ajout, suppression ou modification).

6.2.4.1 Implémentation du système à base de règles

Ces règles gèrent la propagation de l'impact des changements. Ce sont des règles qui calculent l'effet de vague ou *"Ripple effect"* provoqué par un changement d'un processus métier. La propagation de l'impact dépend de l'opération de changement, des types de relation reliant les éléments du processus métier et de la conductivité de l'impact à travers ces relations de dépendance entre les éléments directement ou indirectement concernés par le changement.

Le calcul ou l'identification de la propagation de l'impact est obtenu par la règle *"ChangeImpactPropagation"* (*cf.* Listing 6.1).

Listing 6. 1 *La règle de propagation d'impact du changement*

```
rule "ChangeImpactPropagation"
  when
       FOxNode: BPMNNode(typeNode==”Activity”,stateNode==state.STATE_INIT) then
       FOxEdge: BPMNNode(nodeSrc=activityNode,nodeDest=activityNode)
  then
       mark(FOxNode);
       mark(FOxEdge);
       if (FOxNode.getTypeChange() == “ADDED”)
          FOxNode.setstateNode(states.STATE_ADDED);
       if (FOxNode.getTypeChange() == “MODIFIED”)
          FOxNode.setstateNode(states.STATE_ MODIFIED);
       if (FOxNode.getTypeChange() == “DELETED”)
          FOxNode.setstateNode(states.STATE_DELETED);
       Calculate_Depth_of_Impac(FOxNode);
       Propagate_Impact(FOxNode);
  endif
end
```

En effet, la règle *"ChangeImpactPropagation"* joue un rôle important dans la simulation de la propagation de l'impact. Elle marque les éléments du processus métier affectés

[30] http://www.jboss.org/drools/

par l'opération de changement. Cela provoque le marquage de tous les éléments affectés par la propagation de l'impact. Le marquage d'un nœud change l'état de ce nœud.

La propriété stateNode décrit l'état d'un élément du modèle BPMN. La valeur par défaut de cette propriété est `STATE_INIT`. L'ensemble des états qu'un nœud peut avoir sont définis dans la classe `BPMN_CPA_StateNode` appartenant au plug-in BPMN-CPA. Les valeurs notables de cette propriété sont : `STATE_ADDED`, `STATE_MODIFIED` et `STATE_DELETED`.

6.3 Expérimentation

À titre d'illustration de fonctionnement de l'outil mis en place, nous proposons un scénario représentant un processus de remboursement des frais de fonctionnement des employées d'une entreprise (*cf.* figure 6.13).

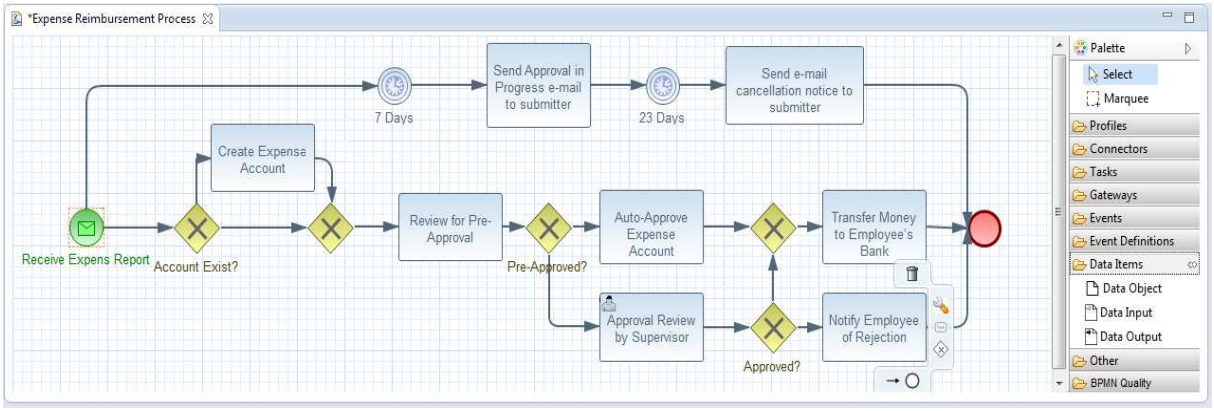


Figure 6.13 *Processus de remboursement des frais de fonctionnement.*

Dans ce processus, les employés d'une entreprise réclament un remboursement de leurs dépenses, par exemple, l'achat de fournitures de bureau, frais de déplacement ou montant d'achat de logiciels.

6.3.1 Schéma initial S du processus

Après que la note de frais soit reçue « Recevoir la note des frais », un nouveau compte est créé si l'employé ne possède pas un compte de dépense « Création du compte de dépense ».

La note de frais est approuvée soit automatiquement ou manuellement. En effet, une approbation automatique du remboursement est effectuée « Approbation automatique du compte » si le montant est inférieur à 200$. Sinon, si le montant est supérieur ou égal à 200$, cela nécessite une approbation par le superviseur « Approbation par le superviseur ». En cas de refus, l'employé reçoit un avis de rejet par e-mail « Notification de rejet à employer », sinon, en cas d'acceptation, un virement bancaire qui permet de rembourser le dû de l'employé est effectué « Virement bancaire au compte de l'employer ».

6.3.2 Schéma résultant S' du processus

Après le déploiement de ce processus métier, de nombreux employés se sont plaint du retard et de l'absence de nouvelles suite à leurs demandes. Par conséquent, l'entreprise a

décidé d'apporter des changements sur le modèle de processus métier initial *S* en ajoutant un service qui permet de répondre à leurs besoins. Les modifications suivantes sont apportées, pour satisfaire cette demande:

- Si aucune action n'est entamée dans les 7 jours qui suivent la demande, l'employé doit recevoir une approbation par e-mail l'informant de la progression de sa demande.
- Si la demande n'est pas traitée dans les 30 jours ouvrables, le processus est arrêté et l'employé reçoit un avis d'irrecevabilité par e-mail.

Ce processus de remboursement des frais de fonctionnement doit aussi se conformer à un ensemble de règles détaillées comme suivant:

- *R1 : si l'employé ne reçoit pas de réponse après 30 Jours de sa demande, un e-mail d'irrecevabilité lui est adressé.*
- *R2: si l'employé ne possède pas de compte, un nouveau compte sera créé.*
- *R3: si le montant est inférieur à 200$, alors le compte de dépenses doit être approuvé automatiquement.*
- *R4: Après que la demande de remboursement est approuvée, l'argent est transféré au compte bancaire de l'employé; autrement, une notification de rejet est envoyé par e-mail à l'employé.*

6.3.3 Vérification de la cohérence et de la conformité du processus

Afin de vérifier la cohérence et la conformité du processus de remboursement des frais de fonctionnement, nous avons utilisé le module BPMNVT de notre plate-forme . Les captures d'écrans de la figure 6.14 et 6.15 montrent respectivement le modèle Promela de ce processus générer par le module BPNM2PROMELA ainsi que le résultat de la vérification de l'absence de blocage " *invalid endstates*".

Pour plus de lisibilités et une meilleure compréhension des résultats, il est possible de générer une structure de kripke à partir du modèle Promela comme le montre la figure 6.16. En effet, EpiSpin propose de générer un code DOT (**.dot*) à partir du fichier (**.pml*), et dont la compilation en image ou en fichier PDF et la visualisation demande l'utilisation des outils tel que Graphiz[31] comme le montre la figure 6.17

6.3.4 Évaluation de la qualité du processus métier

Un changement (Δ) sur un processus métier peut provoquer la propagation d'un impact structurel, fonctionnel, logique, comportemental et/ou qualitatif aux éléments qui lui sont reliés. Pour l'aspect qualitatif il est important de déterminer le taux avec lequel la qualité d'un processus est affectée suite à un changement opéré sur un de ses éléments.

La création de règles de propagation d'impact du changement au niveau de l'aspect qualitatif est basée sur une connaissance des relations incluant notamment la conductivité de l'impact et les variations des valeurs de métriques, etc.

[31] http://www.graphviz.org/

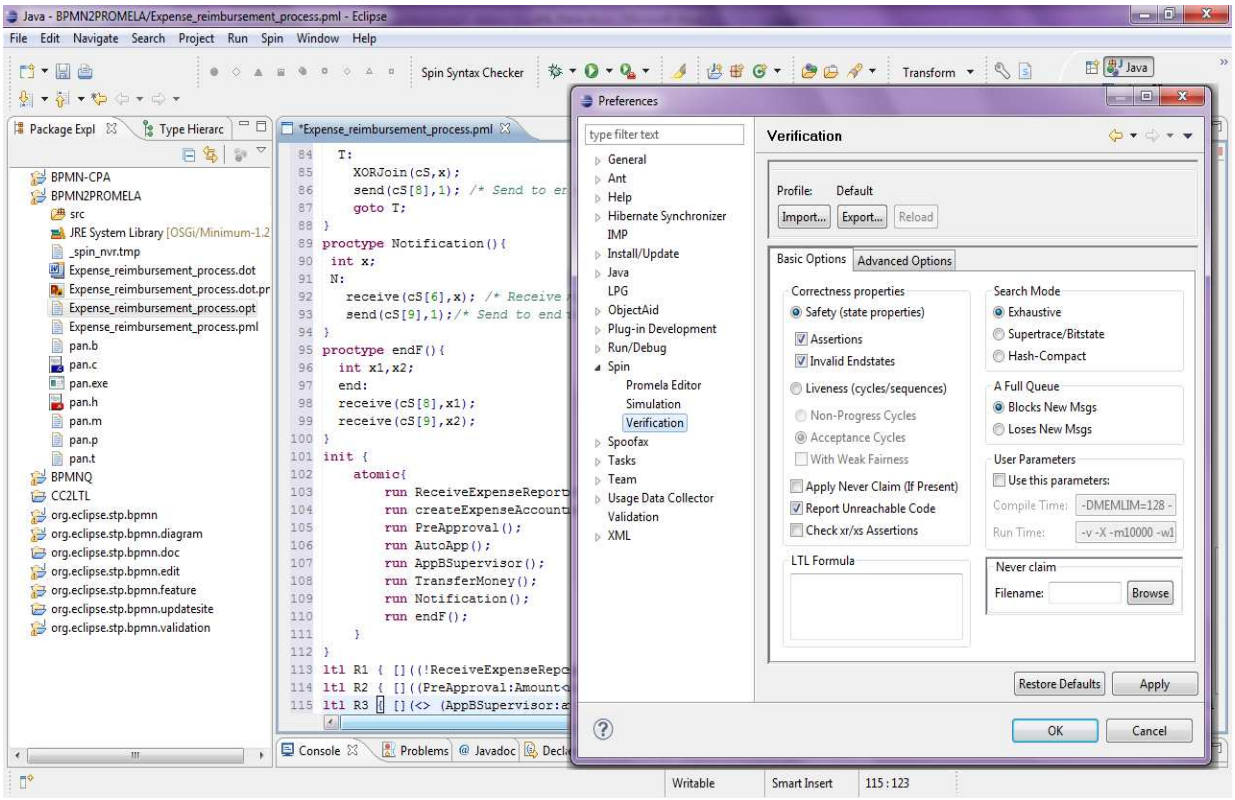


Figure 6.14 *Modèle Promela généré par BPMN2PROMELA.*

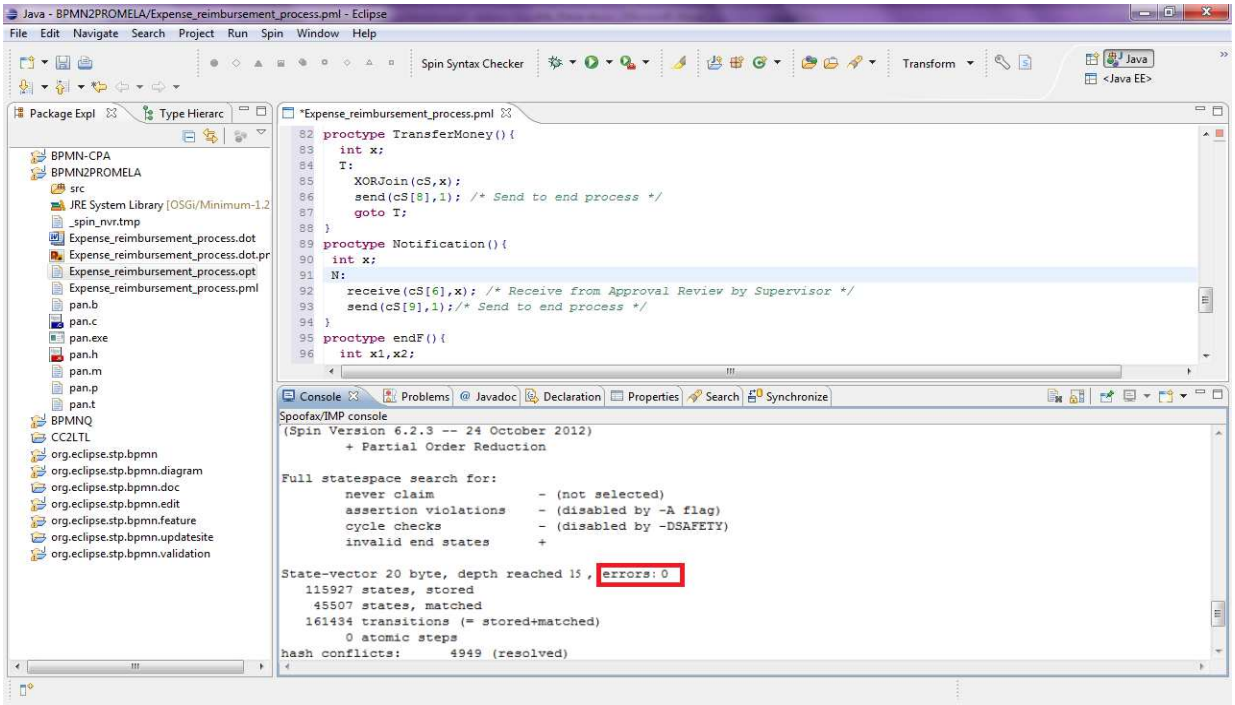


Figure 6.15 *Vérification de la cohérence du processus de remboursement des frais par EpiSpin.*

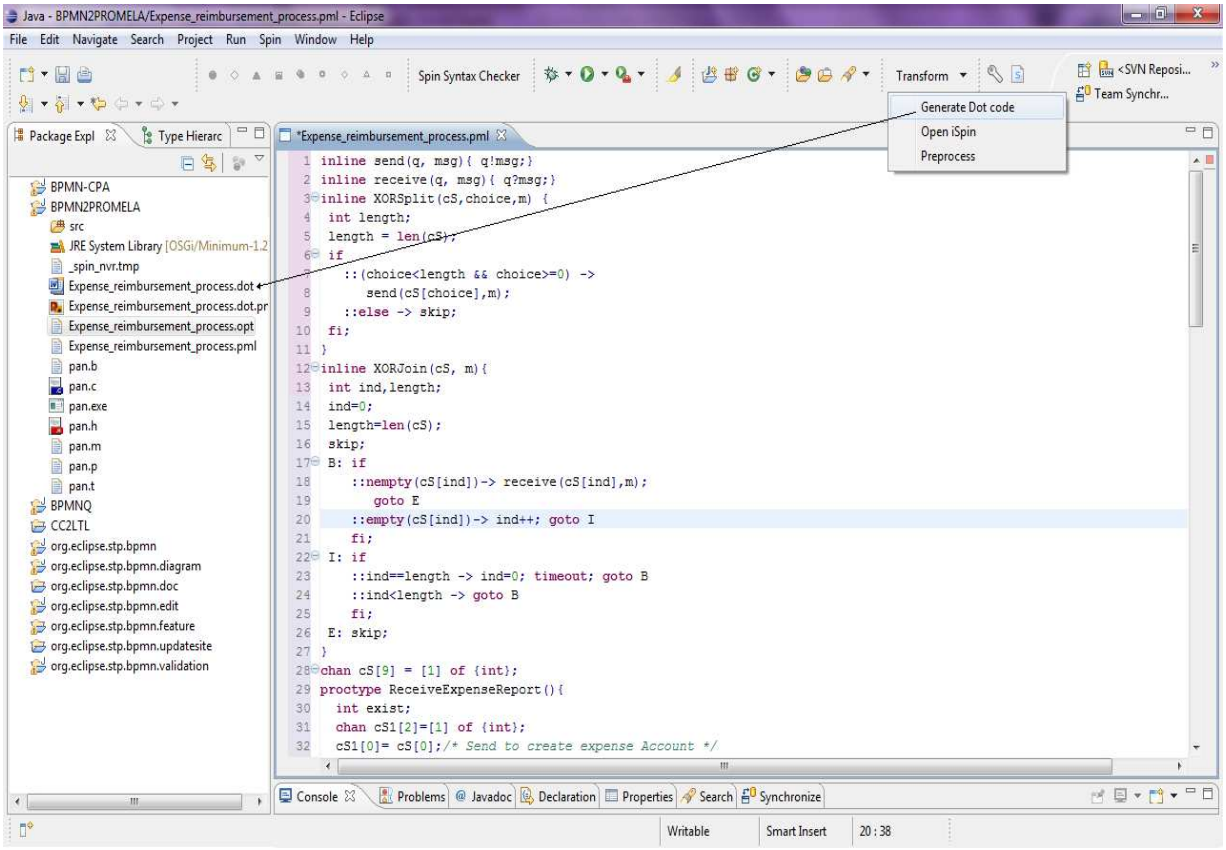


Figure 6.16 *Génération du fichier Expense_reimbursement_process.dot.*

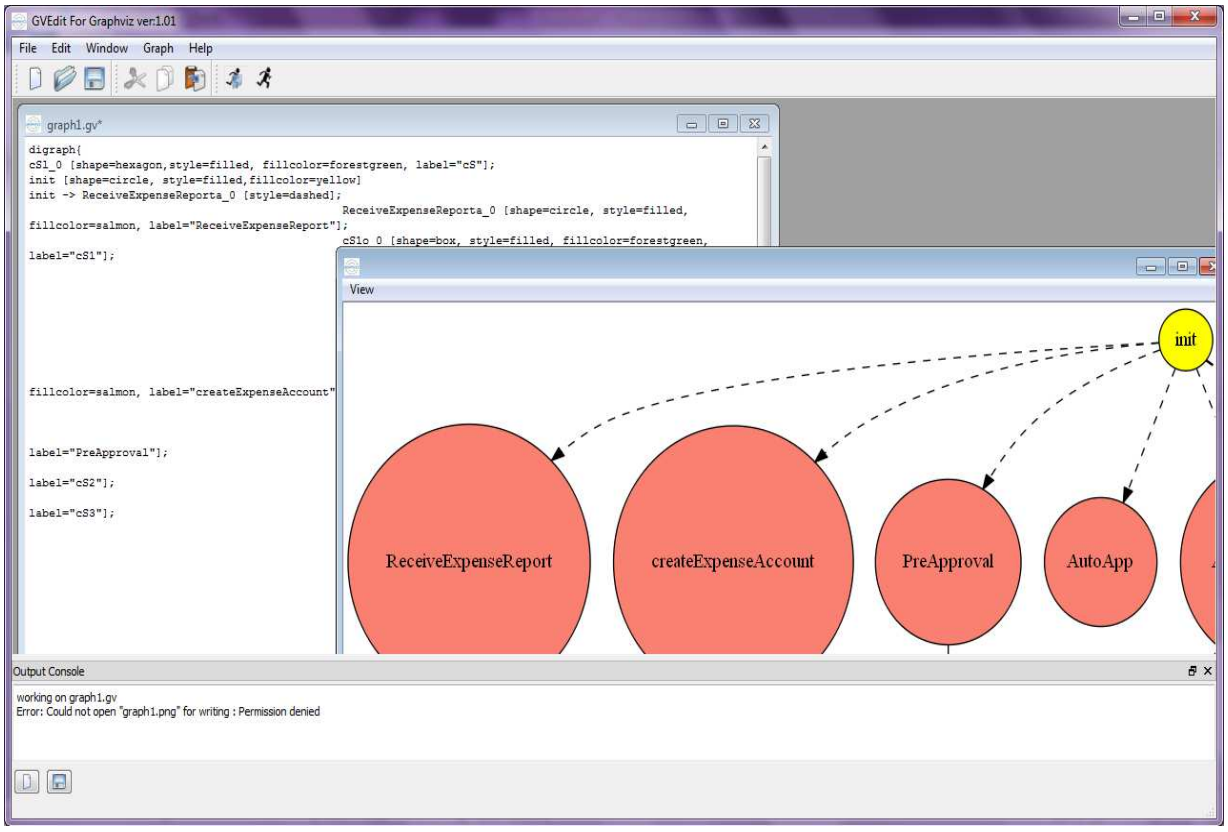


Figure 6.17 *Visualisation du fichier Expense_reimbursement_process.dot avec Graphiz.*

Afin d'illustrer l'impact d'un changement structurel sur le modèle de la qualité, nous avons utilisé notre plug-in BPMNQ. En effet, pour expliquer la propagation de l'impact qualitatif, nous mesurons et observons la qualité de ce processus avant le changement (schéma *S*) et après le changement (schéma S').

La figure 6.12 a et la figure 6.12 b, montrent respectivement les différentes mesures obtenues avant et après ce changement. Comme expliqué dans le chapitre 6, tout changement structurel (ΔS) invoqué sur un processus métier peut causer un impact relatif (ΔQ) sur un critère de la qualité. L'impact de ΔQ peut être propagé progressivement aux critères précédents (dans le sens arbre de la qualité) et aux facteurs correspondants, etc.

Cette évaluation que nous venons de décrire illustre l'impact qualitatif comme étant matérialisé par la variation des différentes valeurs de métriques et par conséquent les critères et les facteurs associés suite à un changement structurel.

6.4 Conclusion

Dans ce chapitre nous avons décrit, la conception et les fonctionnalités de la plate-forme que nous avons développée et déployée pour la validation de notre contribution pour la gestion de l'évolution des processus métier. Cette plate-forme a été développée sous forme d'un ensemble de plug-ins Eclipse pour une meilleure utilisation auprès de la communauté BP et un meilleur retour d'expérience.

Dans ce chapitre nous avons d'abord décrit l'architecture globale de la plate-forme qui est constituée de quatre principaux plug-ins Eclipse qui sont BPMN2 modeler, BPMN verification tool, BPMN Quality, et BPMN change propagation analyser. Nous avons donc détaillé les fonctionnalités de chacun de ces plug-ins.

Finalement nous avons montré, à travers un scénario représentant un processus de remboursement de frais de fonctionnement, l'utilisation de notre plate-forme pour la gestion de l'évolution des processus métier. Les expérimentations que nous avons effectuées à l'aide de la plate-forme BPMN-CM ont concerné particulièrement des processus de différents secteurs (bancaires, gestion administrative, etc.) et avec différents degrés de complexité.

Actuellement, notre travail de développement porte à la fois sur l'accomplissement et l'achèvement du plug-in BPMN-CPA, et cela en intégrant une base de connaissance à l'aide de la plate-forme Jena[32] mais aussi étendre cette analyse *a priori* de l'impact du changement sur autres couches que celle des processus métier.

Nos travaux d'implémentation concernent aussi le développement d'un outil qui permet d'interpréter le résultat d'un contre-exemple fourni par EpiSpin sous forme graphique *"COUNTEREXAMPLE2BPMN"* et non plus textuel comme il est le cas actuellement.

[32] http://jena.sourceforge.net

CHAPITRE 7

Conclusion et perspectives

7.1 Résumé des contributions

Dans ce dernier chapitre, nous allons présenter le bilan de nos trois contributions majeures que nous replaçons dans le contexte général de nos travaux de recherche. Après avoir porté un regard critique sur les sujets que nous avons évoqués dans ce manuscrit, nous présenterons les perspectives de nos travaux.

7.1.1 Bilan des contributions

Le travail présenté dans cette thèse traite de la gestion de l'évolution des processus métier. L'objectif principal étant, de permettre, d'une part, la vérification du bon déroulement du processus métier après chaque changement et d'autre part, d'offrir un moyen qui permet d'évaluer la propagation de l'impact d'un changement de processus métier. Afin de s'atteler à cette tâche, nous avons proposé de découper la problématique globale en trois axes principaux de recherche : (1) la vérification formelle des processus métier, (2) l'analyse *a priori* de l'impact du changement des processus métier et enfin (3) la gestion qualitative des processus métier.

Après avoir dressé un état de l'art des différentes approches et techniques les plus largement connues dans le domaine de la vérification des processus métier et de l'intégration des changements dans ces derniers, nous avons proposé dans un premier volet de nos travaux, une approche basée sur la technique du model-checking permettant la vérification de la cohérence des modèles de processus métier telle que l'absence d'inter-blocage, de boucles infinies, etc. Cette technique permet également de vérifier la conformité des processus métier avec l'ensemble des règles en place. Dans un deuxième volet de nos travaux de recherche, et afin d'évaluer les effets de vague et d'estimer le coût d'un changement de processus métier, nous avons proposé un mécanisme qui permet de simuler et d'évaluer *a priori* la propagation de l'impact des changements dans les processus métier. Cette démarche consiste à définir les relations qui existent entre les différents éléments qui composent le modèle de processus métier. L'ensemble de ces relations appelées relations de dépendance permet de déterminer les éléments qui sont suceptibles d'être affectés par un changement de modèle de processus métier. Dans le troisième et dernier volet de nos travaux, nous avons proposé un modèle qualitatif inspiré par la norme ISO / CEI 9126-1 permettant une meilleure évaluation de la qualité globale d'un processus métier, et cela durant tout son cycle de vie et plus particulièrement après chaque changement de ce dernier.

7.1.1.1 Contribution à la vérification formelle des processus métier

La première contribution de cette thèse a porté sur la vérification formelle des modèles de processus métier par la méthode du model-checking. En effet, nous avons proposé une approche qui se base sur l'utilisation du model-checker SPIN pour vérifier les propriétés de cohérence des modèles de processus métier.

Notre démarche, que nous avons d'ailleurs formalisée, débute par une transformation automatisée d'un modèle de processus métier vers un modèle Promela. Le passage du modèle de processus métier vers le code Promela est réalisé par un automate à états finis étendu (structure de kripke) comme une représentation intermédiaire. L'objectif étant de

fournir une sémantique formelle à ces modèles et de générer ainsi tous les états possibles d'un système dans lesquels la propriété à vérifier est testée exhaustivement sur l'ensemble des exécutions possibles. En second lieu, nous avons exprimé en Logique Temporelle Linéaire les différentes propriétés de cohérence des processus métier.

Le modèle résultant, ainsi que les différentes formules LTL sont fournis comme entrée pour le model-checker SPIN qui a la charge de vérifier si ces propriétés sont satisfaites. Dans le cas d'une réponse négative, SPIN fournit un contre-exemple montrant une exécution qui ne satisfait pas ces propriétés. De la même manière, cette approche a été adoptée pour vérifier la conformité des modèles de processus métier avec l'ensemble des règles de conformité exprimées en LTL par l'intermédiaire de G-LTL suite à un changement affectant l'un des deux.

Cependant, outre le fait que le model-checking ne peut adresser que des problèmes à nombre d'états finis, une limite connue à l'utilisation de cette technique est l'explosion combinatoire du nombre d'états. Cela dit, et en dépit de ces inconvénients qui sont résorbés en partie par certains travaux de recherche, la vérification formelle basée sur le model-checking reste le meilleur choix à faire puisque la vérification est complètement automatique et un contre-exemple est fourni dans le cas où la propriété serait violée.

7.1.1.2 Contribution à l'analyse a priori de l'impact du changement des processus métier

Lors de l'évolution des processus métier, il s'avère, parfois, que les modifications apportées à une partie du modèle de processus métier ont un impact sur les autres éléments du modèle. En effet, ce phénomène connu sous le nom d'effet de vague se manifeste par une propagation des effets de tout changement aussi minime soit-il sur un très grand nombre d'entités du processus. Il est donc primordial de mettre en œuvre des mécanismes d'analyse *a priori* permettant de mieux comprendre et cerner la propagation de l'impact du changement des processus métier afin d'éviter des situations incontrôlées survenant généralement après le changement et de conserver ainsi une certaine cohérence du processus.

Dans notre deuxième contribution, nous avons proposé un mécanisme qui permet une analyse *a priori* de l'impact du changement des processus métier à travers une classification exhaustive des différentes relations de dépendance qui servent à véhiculer la propagation de l'impact du changement. Ce mécanisme permet non seulement d'analyser les différentes relations de dépendance qui existent entre la partie changée et les autres parties potentiellement affectées au sein du modèle de processus métier, mais aussi d'analyser les relation entre ces modèles de processus métier et les services associés.

Enfin, et pour mettre en place notre approche, nous avons proposé l'élaboration d'un système à base de connaissances et un ensemble de règles formulées pour les différents types de relation permettant ainsi une analyse *a priori* de la propagation d'impact du changement entre les différents éléments qui contituent et/ou qui communiquent avec ce modèle de processus métier.

Cependant, une des limites actuelles de notre approche d'analyse d'impact du changement sur un modèle de processus métier consiste à ne pas prendre en

considération certains types de relation de dépendance telles que les acteurs et les ressources, et qu'il faudrait intégrer par la suite dans notre démarche d'analyse. D'ailleurs, ceci est l'une de nos perspectives, que nous présentons dans la section suivante.

7.1.1.3 Contribution à la gestion qualitative des processus métier

Même s'il n'existe pas actuellement un consensus sur une définition standard de la qualité des processus métier, diverses approches pour l'évaluation de la qualité des processus métier ont été proposées dans la littérature. Dans une troisième contribution, et après avoir fait un tour d'horizon des différents travaux de recherche dans ce domaine, nous avons tenté de proposer un cadre exhaustif inspiré de la norme ISO/ CEI 9126-1 représentant à la fois une contribution à la maturité des travaux existants à l'égard de l'évaluation et de l'amélioration de la qualité des processus métier et un moyen permettant de traiter la problématique de l'analyse de l'impact qualitatif du changement.

Enfin, les trois phases de notre approche représentent les fonctionnalités d'un outil que nous avons développé et appelé BPMN-CM. Cet outil offre un cadre qui accompagne les modeleurs et les experts métier dans la modélisation BPMN et la gestion des changements de ces modèles. Toutefois, nous pensons qu'un prototype en dépit de son coût de réalisation restera toujours un sujet d'amélioration. Dans la section suivante, nous présentons les perspectives de notre thèse.

7.1.2 Perspectives des travaux

Comme une première perspective de nos travaux de recherche concernant la vérification formelle des processus métier, nous envisageons de prendre en compte le flux de données en plus du flux de contrôle dans la vérification de la cohérence et de la conformité des processus métier. Nous envisageons également de valider notre approche sur des modèles de processus métier plus volumineux et plus complexes avec des notations, et surtout d'automatiser l'interprétation des contre-exemples en cas d'une incohérence dans ces modèles.

Dans un futur proche, et afin de compléter notre approche d'analyse d'impact du changement dans les processus métier, nous envisageons tout d'abord d'inclure d'autres types de relations de dépendance telles que les acteurs, les ressources en plus de celles concernant les activités, données et rôles déjà utilisés dans ce manuscrit.

Dans le chapitre 4, nous avons également souligné l'importance de considérer l'impact du changement des processus métier de différents points de vue conceptuels d'où une perspective principale de rendre l'analyse actuelle plus exhaustive en tenant compte des points de vue fonctionnel, logique et comportemental en plus des deux points de vue structurel et qualitatif déjà abordés dans notre thèse.

Afin de rendre cela possible, on prévoit d'utiliser la technique de l'analyse de la traçabilité en complément de celle de l'analyse des relations de dépendance. Cette analyse de traçabilité portera sur la comparaison des différentes traces d'exécution "Process mining" d'un processus métier avant et après changement afin d'extraire un comportement exploitable par cette analyse.

De la même manière, et dans le domaine de la gestion qualitative des processus métier, nous estimons qu'il est important de considérer la sémantique des modèles de processus métier aussi bien lors de l'évaluation que lors de l'amélioration de leur qualité. Pour cela, nous proposons d'assister cette modélisation par une démarche centrée sur la qualité qui vise à exploiter des connaissances de domaine exprimées sous la forme d'une ontologie. Cette démarche devrait s'appuyer sur les techniques de méta modélisation pour rapprocher les connaissances du domaine de la connaissance exprimée par les modèles de processus métier, le but étant d'améliorer l'expressivité de ces modèles.

Bibliographie

Abd-El-Hafiz, S. K. (1996). Evaluation of a knowledge-based approach to program understanding. *The 3rd Working Conference on Reverse Engineering (WCRE '96)* (pp. 275-284). Washington, DC, USA: IEEE Computer Society.

Abouzaid, F., & Mullins, J. (2009). Model-checking Web Services Orchestrations using BP-calculus. *Electronic Notes in Theoretical Computer Science (ENTCS) Vol. 255*, 3-21.

Adams, M., ter Hofstede, A. H., Edmond, D., & van der Aalst, W. M. (2005). facilitating flexibility and dynamic exception handling in workflows through worklets. *In 17th International Conference on Advanced Information Systems Engineering (CAiSE'05)*, (pp. 45-50).

Agostini, R., & De Michelis, G. (2000). A light workflow management system using simple process models. *Computer Supported Cooperative Work*, 335-363.

Aguilar, E. R., Cardoso, J., García, F., Ruiz, F., & Piattini, M. (2009). Analysis and Validation of Control-Flow Complexity Measures with BPMN Process Models. *The 10th International Workshop on Enterprise, Business-Process and Information Systems Modeling (BPMDS'09), and 14th International Conference, EMMSAD 2009, held at CAiSE 2009* (pp. 58-70). Amsterdam, The Netherlands: Springer Berlin Heidelberg.

Aguilar, E. R., Ruiz, F., García, F., & Piattini, M. (2006). Applying Software Metrics to evaluate Business Process Models. *CLEI Electron. Journal, Vol. 9, No. 1*.

Ajila, S. (1995). Software Maintenance: An Approach to Impact Analysis of Objects Change. *Journal of Software - Practice and Experience Vol.25*, 1155-1181.

Alter, S. (1999). A general, yet useful theory of information systems. *Association for Information Systems, Vol.1*, 1-70.

Alter, S. (1999). *Information Systems: A Management Perspective.* Edition 3e éd., Addison-Wesley.

Arango, G., Schoen, E., & Pettengill, R. (1993). A process for consolidating and reusing design knowledge. *The 15th international conference on Software Engineering (ICSE '93)* (pp. 231-242). Baltimore, MD, USA: IEEE Computer Society.

ARIS. (2007). The Event-driven Process Chain. ARIS Design Platform,.

Arnold, R. S., & Bohner, S. A. (1993). Impact Analysis - Towards a Framework for Comparison. *The International Conference on Software Maintenance (ICSM '93)* (pp. 292-301). Montréal, Quebec, Canada: IEEE Computer Society.

Awad, A. (2007). BPMN-Q: A Language to Query Business Processes. *The 2nd International Workshop on Enterprise Modelling and Information Systems Architectures (EMISA'07)*, (pp. 115-128). St. Goar, Germany.

Awad, A., & Puhlmann, F. (2008). Structural detection of deadlocks in business process models. *The 11th International Conference On Business Information Systems (BIS'08)* (pp. 239-250). Innsbruck, Austria: Springer Berlin Heidelberg.

Awad, A., Decker, G., & Weske, M. (2008). Efficient Compliance Checking Using BPMN-Q. *The 6th International Conference on Business Process Management (BPM'08)* (pp. 326-341). Milan, Italy: Springer Berlin Heidelberg.

Baniassad, E., & Clarke, S. (2004). Theme: An Approach for Aspect-Oriented Analysis and Design. *The 26th International Conference on Software Engineering (ICSE '04)* (pp. 158-167). IEEE Computer Society.

Becker, J., Rosemann, M., & von Uthmann, C. (2000). Guidelines of Business Process Modeling. Dans *Business Process Management* (pp. 30-49). Springer Berlin Heidelberg.

Behrmann, G., David, R., & Larsen, K. G. (2004). A tutorial on UPPAAL. *The International School on Formal Methods for the Design of Computer, Communication, and Software Systems* (pp. 200-236). Bertinora, Italy: Springer Berlin Heidelberg.

Bel Mokadem, H., Bérard, B., Bouyer, P., & Laroussinie, F. (2006). Timed Temporal Logics for Abstracting Transient States. *The 4th International Symposium on Automated Technology for Verification and Analysis (ATVA'06)* (pp. 337-351). Beijing, China: Springer Berlin Heidelberg.

Belhajjame, K., Vargas-Solar, G., & Collet, C. (2001). Towards an Adaptable Workflow Management System. *17èmes Journées BDA.* Agadir, Maroc.

Biere, A. (2008). Tutorial on Model Checking, Modelling and Verification in Computer Science. *The 3th International Conference on Algebraic Biology (AB'08)* (pp. 16-21). Castle of Hagenberg, Austria: Springer Berlin Heidelberg.

Bloom, J., Clark, C., Clifford, C., Duncan, A., Khan, H., & Papantoniou, M. (2003). *Platform Independent Petri-net Editor.* Dans Rapport final du projet PIPE. Department of Computing, Imperial College London.

Boehm, B. W., Brown, J. R., & Lipow, M. (1976). Quantitative evaluation of software quality. *The 2nd international conference on Software engineering (ICSE '76)* (pp. 592-605). IEEE Computer Society.

Bohner, S. A. (1996). *A Graph Traceability Approach for Software Change Impact Analysis.* Doctoral Dissertation - George Mason University.

Bohner, S. A. (2002). Software Change Impacts an Evolving Perspective. *The International conference on Software Maintenance (ICSM'02)* (pp. 263-272). IEEE Computer Society.

Borghoff, U. M., Bottoni, P., Mussio, P., & Pareschi, R. (1997). Reflective Agents for Adaptive Workflows. *The 2d International Conference and Exhibition on Practical Application of Intelligent Agents and Multi-Agents (PAAM'97)* (pp. 405-420). London, UK: Society Press.

Boucher, X. (2007). *Vers un pilotage agile de l'évolution des systèmes de production.*

Briol, P. (2008). *ingénierie des processus métiers de l'élaboration à l'exploitation.* Edition Lulu.com.

Browne, M. C., Clarke, E. M., & Grümberg, O. (1988). Characterizing finite Kripke structures in propositional temporal logic. *Theoretical Computer Science - International Joint Conference on Theory and Practice of Software Development Volume 59 Issue 1-2*, 115-131.

Bry, F., Eckert, M., Pătrânjan, P.-L., & Romanenko, I. (2006). Realizing business processes with ECA rules: benefits, challenges, limits. *The 4th international conference on Principles and Practice of Semantic Web Reasoning (PPSWR'06)* (pp. 48-62). Budva, Montenegro: Springer-Verlag Berlin.

Büchi, J. (1962). On a decision method in restricted second order arithmetic. *The International Congress on Logic, Method, and Philosophy of Science*, (pp. 1-12).

Burch, J. R., Clarke, E. M., Mcmillan, K. L., Dill, D. L., & Hwang, L. J. (1990). Symbolic model checking: 10^20 states and beyond. *The 5th Annual IEEE Symposium on Logic in Computer Science (LICS '90)* (pp. 428-439). Philadelphia, Pennsylvania, USA: IEEE Computer Society.

Burkhart, T., & Loos, P. (2010). Flexible Business Processes - Evaluation of Current Approaches. *Multikonferenz Wirtschaftsinformatik.* Göttingen, Germany.

Butler, K. A., Esposito, C., & Hebron, R. (1999). Connecting the design of software to the design. *Communications of the ACM. 42(1)*, 38-46.

Cardoso, J. (2005). About the Data-Flow Complexity of Web Processes. *The 6th International Workshop on Business Process Modeling, Development, and Support: Business Processes and Support Systems: Design for Flexibility.* Porto, Portugal.

Cardoso, J. (2005). Control-flow Complexity Measurement of Processes and Weyuker's Properties. *The 6th International Enformatika Conference (IEC'05)*, (pp. 213-218).

Cardoso, J. (2005). *Evaluating Workflows and Web Process Complexity.* FL, USA: in Workflow Handbook 2005, L. Fischer, Editor. 2005, Future Strategies Inc.: Lighthouse Point, p. 284-290.

Cardoso, J. (2006). Process control-flow complexity metric: An empirical validation. *IEEE International Conference on Services Computing SCC '06*, (pp. 167-173).

Cardoso, J., Mendling, J., Neumann, G., & Reijers, H. A. (2006). A discourse on complexity of process models. *International Workshops on Business Process Management (BPM'06)* (pp. 117-128). Vienna, Austria: Springer Berlin Heidelberg.

Cardoso, J., Miller, J., & Arnold, J. (2002). *Modeling Quality of Service for Workflows and Web Service Processes.* LSDIS Lab, Computer Science, University of Georgia. #02-200.

Casati, F., Ceri, S., Pernici, B., & Pozzi, G. (1996). Workflow Evolution. *Data and Knowledge Engineering*, 438-455.

Cavano, J. P., & McCall, J. A. (1978). A framework for the measurement of software quality. *SIGSOFT Softw. Eng. Notes, 3(5)*, (pp. 133–139).

Chiu, D., Li, Q., & Karlapalem, K. (2001). Facilitating workflow evolution in an advanced object environment. *The 7th International Conference on Database Systems forAdvanced Applications*, (pp. 148-149). HongKong, China.

Chniti, A., Albert, P., & Charlet, J. (2011). Gestion de la cohérence des règles métier éditées à partir d'ontologies OWL*. *22èmes Journées francophones d'Ingénierie des Connaissances (IC 2011).* Chambéry, France.

Chountas, P., Petrounias, I., & Kodogiannis, V. (2003). Temporal Modelling in Flexible Workflows. *The 18th International Symposium on Computer and Information Sciences (ISCIS' 03)* (pp. 123-130). Antalya, Turkey: Springer Berlin Heidelberg.

Chulsoon, P., & Injun, C. (2004). Management of business process constraints using BPTrigger. *Journal of Computers in Industry Volume 55, Issue 1*, 29–51.

Chun, S. A., Atluri, V., & Adam, N. R. (2002). Domain Knowledge-Based Automatic Workflow Generation. *The 13th International Conference on Database and Expert Systems Applications (DEXA'02)* (pp. 81-93). Aix-en-Provence, France: Springer Berlin Heidelberg.

Cimatti, A., Clarke, E., Giunchiglia, E., Giunchigli, F., Pistore, M., Roveri, M., et al. (2002). NuSMV 2: An OpenSource Tool for Symbolic Model Checking. *The 14th International Conference on Computer Aided Verification (CAV'02)* (pp. 359-364). Copenhagen, Denmark: Springer Berlin Heidelberg.

Cimatti, A., Clarke, E., Giunchiglia, F., & Roveri, M. (2000). NUSMV: a new symbolic model checker. *International Journal on Software Tools for Technology Transfer*.

Clarke Jr., E. M., Grumberg, O., & Peled, D. A. (1999). *Model Checking.* Cambridge, MA, USA: The MIT Press; n edition (January 7, 1999).

Coalition, T. W. (1999). *Workflow Management Coalition Terminology & Glossary.* Rapport numéro WFMC-TC-1011 Issue 3.0.

Crusson, T. (2003). *Business Process Management – De la modélisation à l'exécution, Positionnement par rapport aux Architectures Orientées Services.* Intalio white paper.

Dąbrowski, M., Drabik, M., Trzaska, M., & Subieta, K. (2011). Prototype of Object-Oriented Declarative Workflows. *The 3th International Conference on Intelligent Information and Database Systems (ACIIDS'11)* (pp. 47-56). Daegu, Korea: Springer Berlin Heidelberg.

Dai, W., & Covvey, H. D. (2005). *Query-Based Approach to Workflow Process Dependency Analysis.* Waterloo, Ontario,Canada: Technical Report 01, School of Computer Science and the Waterloo Institute for Health Informatics Research .

David, R., & Alla, H. (1993). *Autonomous and timed continuous Petri nets.* Springer Berlin Heidelberg.

David, R., & Alla, H. (2010). *Non-Autonomous Petri Nets.* Springer Berlin Heidelberg.

De Backer, M., & Snoeck, M. (2008). *Business Process Verification: a Petri Net Approach.* Belgium: Technical report, Catholic University of Leuven.

Debauche, B., & Mégard, P. (2004). *BPM Business Process Management : Pilotage métier de l'entreprise.* Edition Hermes Science Publications.

Decker, G., Dijkman, R., Dumas, M., & Luciano, G.-B. (2008). Transforming BPMN Diagrams into YAWL Nets. *The 6th International Conference on Business Process Management (BPM'08)* (pp. 386-389). Milan, Italy: Springer Berlin Heidelberg.

Dellen, B., Maurer, F., & Pews, G. (1997). Knowledge Based Techniques to Increase the Flexibility of Workflow Management. *Data and Knowledge Engineering, North-Holland*, 269-295.

Deruelle, L., Bouneffa, M., Melab, N., Basson, H., & Goncalves, G. (2000). A Change Impact Analysis Approach For CORBA-Based Federated Databases. *The 11th International Conference on Database and Expert Systems Applications (DEXA'00)* (pp. 949-958). London, UK: Springer Berlin Heidelberg.

Di Francescomarino, C., & Tonella, P. (2009). Crosscutting Concern Documentation by Visual Query of Business Processes. *The International Workshops On Business Process Management (BPM'08)* (pp. 18-31). Milano, Italy: Springer Berlin Heidelberg.

Diaz, G., Pardo, J.-J., Cambronero, M.-E., Valero, V., & Cuartero, F. (2005). Automatic Translation of WS-CDLChoreographies to Timed Automata. *The international conference on European Performance Engineering, and Web Services and Formal Methods, international conference on Formal Techniques for Computer Systems and Business Processes (EPEW'05/WS-FM'05)* (pp. 230-242). Versailles, France: Springer-Verlag Heidelberg.

Dijkman, R. M., Dumas, M., & Ouyang, C. (2007). *Formal Semantics and Analysis of BPMN Process Models using Petri Nets.*

Dijkstra, E. (1971). Hierarchical Ordering of Sequential Processes. *Acts Informatica I*, 115–138.

El Kharbili, M., de Medeiros, A. K., Stein, S., & van der Aalst, W. P. (2008). Business Process Compliance Checking Current State and Future Challenges. *The Modellierung betrieblicher Informations syteme (MobIS'08)*, (pp. 107-113).

Fan, S., Dou, W., & Chen, J. (2007). Dual Workflow Nets: Mixed Control/Data-Flow Representation for Workflow Modeling and Verification. *The International Workshops: DBMAN 2007, WebETrends 2007, PAIS 2007 and ASWAN 2007* (pp. 433-444). Huang Shan, China: Springer-Verlag Berlin Heidelberg.

Farahbod, R., Glässer, U., & Vajihollahi, M. (2005). A Formal Semantics for the Business Process Execution Language for Web Services. *Web Services and Model-Driven Enterprise Information Services* (pp. 144-155). INSTICC Press.

Fehling, R. (1993). A concept of hierarchical Petri nets with building blocks. *The 12th International Conference on Application and Theory of Petri Nets* (pp. 148-168). Springer Berlin Heidelberg.

Feja, S., Speck, A., & Pulvermüller, E. (2009). Business Process Verification. *GI-Jahrestagung*, (pp. 4037-4051). Lübeck, Germany.

Fenton, N. E., & Pfleeger, S. L. (1998). *Software Metrics: A Rigorous and Practical Approach.* Boston, MA, USA: PWS Publishing Co, 2ème edition.

Fernandez, J.-C., Garavel, H., Kerbrat, A., Mounier, L., Mateescu, R., & Sighireanu, M. (1996). CADP - A Protocol Validation and Verification Toolbox. *The 8th International Conference of Computer Aided Verification (CAV '96)* (pp. 437-440). Brunswick, NJ, USA: Springer Berlin Heidelberg.

Ferrara, A. (2004). Web services: a process algebra approach. *The 2nd international conference on Service oriented computing (ICSOC'04)*, (pp. 242-251).

Fisler, K., Krishnamurthi, S., Meyerovich, L. A., & Tschantz, M. C. (2005). Verification and change-impact analysis of access-control policies. *The 27th international conference on Software engineering (ICSE'05)* (pp. 196-205). St. Louis, MO, USA: ACM New York.

Fisteus, J. A., Fernández, L. S., & Kloos, C. D. (2004). Formal Verification of BPEL4WS Business Collaborations. *The 5th International Conference on E-Commerce and Web Technologies (EC-Web'04)* (pp. 76-85). Zaragoza, Spain: Springer Berlin Heidelberg.

Fisteus, J. A., Fernández, L. S., & Kloos, C. D. (2004). Formal Verificationof BPEL4WS Business Collaborations. *Lecture notes in computer science* , 76-85.

Foerster, A., Engels, G., & Schattkowsky, T. (2005). Activity Diagram Patterns for Modeling Quality Constraints in Business Processes. *The 8th International Conference on Model Driven Engineering Languages and Systems (MoDELS'05)* (pp. 2-16). Montego Bay, Jamaica: Springer Berlin Heidelberg.

Fu, X., Bultan, T., & Su, J. (2004). Analysis of Interacting BPEL Web Services. *13th international conference on World Wide Web (WWW '04)* (pp. 621-630). ACM New York.

Gardner, T. (2009). *UML Modelling of Automated Business. Processes with a Mapping to BPEL4WS.* IBM UK Laboratories.

Ghidini, C., Rospocher, M., & Serafini, L. (2008). *A formalisation of BPMN in Description Logics.* Technical report, FBK.

Ghose, A. K., & Koliadis, G. (2007). Auditing business process compliance. *The 5th International Conference on Service-Oriented Computing (ICSOC'07)* (pp. 169-180). Vienna, Austria: Springer Berlin Heidelberg.

Gillot, J.-N. (2008). *The Complete Guide to Business Process Management: Business Process Transformation or a Way of Aligning the Strategic Objectives of the Company and the Information System Through the Processes.* Edition Lulu.com.

Giurca, A., Lukichev, S., & Wagner, G. (2006). Modeling Web Services with URML. *The Semantics for Business Process Management Workshop (SBPM'06).*

Glinz, M. (2007). On Non-Functional Requirements. *The 5th IEEE International Requirement Engineering Conference (RE '07)* (pp. 21–26). Delhi, India: IEEE Computer Society.

Goedertier, S., & Vanthienen, J. (2006). Compliant and Flexible Business Processes with Business Rules. *The CAISE*06 Workshop on Business Process Modelling, Development, and Support (BPMDS 2006).* Luxemburg.

Goedertier, S., & Vanthienen, J. (2006). Designing Compliant Business Processes with Obligations and Permissions. *Business Process Management Workshops, BPM'06* (pp. 5-14). Vienna , AUTRICHE: Springer-Verlag Berlin, Heidelberg.

Grim-Yefsah, M., Rosenthal-Sabroux, C., & Thion-Goasdoué, V. (2011). BoQal : une méthode pour l'évaluation de la qualité d'un processus métier. *29ème congrès INFORSID.* Lille, France.

Group, O. M., & OMG, O. M. (2007). *Unified Modeling Language.* le rapport de spécification version 2.1.1, numéro : formal/2007-02-03.

Group, T. B. (2000, July). *Defining Business Rules, What are they really?* Récupéré sur www.businessrulesgroup.org

Gruhn, V., & Laue, R. (2006). Complexity Metrics for Business Process Models. *The 9th International Conference on Business Information Systems (BIS'06)*, (pp. 1-12). Klagenfurt, Austria.

Gruhn, V., & Laue, R. (2006). How Style Checking Can Improve Business Process Models. *8th International Conference on Enterprise Information Systems (ICEIS'06).* Paphos, Cyprus.

Gruhn, V., & Laue, R. (2007). What business process modelers can learn from programmers. *Science of Computer Programming*, 4-13.

Guceglioglu, A. S., & Demirors, O. (2005). Using Software Quality Characteristics to Measure Business Process Quality. *The 3rd International Conference on Business Process Management (BPM'05)* (pp. 374-379). Nancy, France: Springer Berlin Heidelberg.

Haney, F. M. (1972). Module connection analysis: a tool for scheduling software debugging activities. *The AFIPS Joint Computer Conference* (pp. 173-179). AFIPS / ACM / Thomson Book Company.

Heinl, P., Horn, S., Jablonski, S., Neeb, J., Stein, K., & Teschke, M. (1999). A Comprehensive Approach to Flexibility in Workflow Management Systems. *international joint conference on Work activities coordination and collaboration (WACC '99)* (pp. 79-88). ACM SIGSOFT Software Engineering.

Heinrich, R., & Paech, B. (2010). Defining the Quality of Business Processes. *The Modellierung 2010. LNI, vol. P-161*, (pp. 133–148). Bonner Köllen, Germany.

Henderson-Sellers, B. (1995). *Object-Oriented Metrics: Measures of Complexity.* Prentice Hall.

Henzinger, T. A., Jhala, R., Majumdar, R., & Sutre, G. (2003). Software Verification with BLAST. *The 10th International SPIN Workshop on Model Checking Software (SPIN)* (pp. 235-239). Portland, OR, USA: Springer Berlin Heidelberg.

Heravizadeh, M., Mendling, J., & Rosemann, M. (2009). Dimensions of Business Processes Quality (QoBP). *The International Workshops on Business Process Management (BPM'08)* (pp. 80-91). Milano, Italy: Springer Berlin Heidelberg.

Herbst, J. (1999). An Inductive Approach to the Acquisition and Adaptation of Workflow Models. *The IJCAI'99 Workshop on Intelligent Workflow and Process Management: The New Frontier for AI in Business*, (pp. 52-57).

Hillah, L. M., Kindler, E., Kordon, F., Petrucci, L., & Treves, N. (2009). A primer on the Petri Net Markup Language and ISO/IEC 15909-2. *Petri Net Newsletter, Vol. 76 (2009)*, (pp. pp. 9-28).

Hinton, A., Kwiatkowska, M., Norman, G., & Parker, D. (2006). PRISM: A Tool for Automatic Verification of Probabilistic Systems. *The 12th International Conference On Tools and Algorithms for the Construction and Analysis of Systems (TACAS'06)* (pp. 441-444). Vienna, Austria: Springer Berlin Heidelberg.

Hinz, S., Schmidt, K., & Stahl, C. (2005). Transforming BPEL to Petri Nets. *The International Conference on Business Process Management (BPM'05)* (pp. 220-235). Springer-Verlag.

Hoare, C. A. (1978). Communicating Sequential Processes. *Communications of the ACM Volume 21 Issue 8*, 666-677.

Holzmann, G. J. (1997). State compression in spin. *The 3rd SPIN Workshop.*

Holzmann, G. J. (1997). The Model Checker SPIN. *IEEE Transactions on Software Engineering*, 279-295.

Holzmann, G. J. (2003). *The SPIN Model Checker.* Primer and Reference Manual.

Hopcroft, J. E., Motwani, R., & Ullman, J. D. (2006). *Introduction to Automata Theory, Languages, and Computation.* 3rd edition. Addison-Wesley, Reading.

Hornus, S., & Schnoebelen, P. (2002). On solving temporal logic queries. *9th International Conference on Algebraic Methodology and Software Technology (AMAST'02)* (pp. 163-177). Saint-Gilles-les-Bains, Reunion Island, France: Springer Berlin Heidelberg.

Ishikawa, K. (1985). *What is Total Quality Control? the Japanese Way.* Prentice Hall.

ISO. (1989). *LOTOS — A Formal Description Technique Based on the Temporal Ordering of Observational Behaviour.* International Organization for Standardization — InformationProcessing Systems— Open Systems Interconnection.

ISO 9000:2000. (2005). *ISO 9000:2000, Quality Management Systems - Fundamentals and Vocabulary.* International Organization for Standardization (ISO).

ISO/IEC 9126-1. (2001). *ISO/IEC 9126-1.: Software engineering — Product quality — Part 1: Quality model, First edition.*

Jablonski, S., & Bussler, C. (1996). *Workflow Management: Modeling Concepts, Architecture and Implementation.* International Thomson Publishing.

Jansen-Vullers, M., & Netjes, M. (2006). Business process simulation – a tool survey. *The 7eme Workshop and Tutorial on Practical Use ofColoured Petri Nets and CPN Tools.*

Janssen, W., Mateescu, R., Mauw, S., & Springintveld, J. (1998). Verifying Business Processes using SPIN. *The 4th International SPIN Workshop*, (pp. 21-36).

Jensen, K. (1992). Coloured Petri Nets. *Modelling and Validation of Concurrent Systems*.

Kherbouche, M. O., & Basson, H. (2011). Modélisation et analyse pour la gestion de l'évolution des BPM. *29ème congrès INFORSID (Forum des jeunes chercheurs)* (pp. 441-442). Lille, France: INFORSID.

Kherbouche, M. O., Ahmad, A., & Basson, H. (2012). Detecting structural errors in BPMN process models. *The 15th IEEE International Multitopic Conference (INMIC'12)* (pp. 425-431). Islamabad, Punjab, Pakistan: IEEE Computer Society.

Kherbouche, M. O., Ahmad, A., & Basson, H. (2013). Formal approach for compliance rules checking in Business Process Models. *2013 IEEE International Conference on Emerging Technologies (ICET'13).* Islamabad, Punjab, Pakistan: IEEE Computer Society.

Kherbouche, M. O., Ahmad, A., & Basson, H. (2013). Using model checking to control the structural errors in BPMN models. *The 7th IEEE International Conference on Research Challenges in Information Science (RCIS'13)* (pp. 1-12). Paris, France: IEEE Computer Society.

Kherbouche, M. O., Ahmad, A., Bouneffa, M., & Basson, H. (2013). Ontology-based change impact assessment in dynamic business processes. *The 11th International Conference on Frontiers of Information Technology (FIT'13).* Islamabad, Pakistan: IEEE Computer Society .

Kherbouche, O. M., Ahmad, A., Bouneffa, M., & Basson, H. (2013). Analyzing the ripple effects of change in business process models. *16th IEEE International Multitopic Conference (INMIC'13).* Lahore, Pakistan: IEEE Computer Society.

Kherbouche, O. M., Bouneffa, M., Ahmad, A., & Basson, H. (2013). Analyse a priori de l'impact du changement des processus métiers. *31ème congrès INFORSID* (pp. 257-265). Paris, France: INFORSID.

Kim, K.-H. (2003). Workflow Dependency Analysis and Its Implications on Distributed Workflow Systems. *The 17th International Conference on Advanced Information Networking and Applications (AINA'03)* (pp. 677-682). IEEE Computer Society.

Klein, M., & Dellarocas, C. (2000). A Knowledge-based Approach to Handling Exceptions inWorkflow Systems. *Journal Computer Supported Cooperative Work Special Issue on Adaptive Workflow System, 9(3/4).*, 399-412.

Klingemann, J. (2000). Controlled flexibility in workflow management. *The 12th International Conference on Advanced Information Systems Engineering (CAiSE' 00)* (pp. 126-141). Stockholm, Sweden: Springer Berlin Heidelberg.

Knethen, A. v. (2001). A trace model for system requirements changes on embedded systems. *The 4th International Workshop on Principles of Software Evolution (IWPSE '01)* (pp. 17-26). ACM New York.

Knolmayer, G., Endl, R., & Pfahrer, M. (2000). Modeling Processes and Workflows by Business Rules. *Business Process Management, Models, Techniques, and Empirical Studies* (pp. 16-29). Springer-Verlag.

Koehler, J., Tirenni, G., & Kumaran, S. (2002). From Business Process Model to Consistent Implementation: A Case for Formal Verification Methods. *the 6th International Entreprise Distributed Object Computing Conference (EDOC'02)* (pp. 96-106). Lausanne, Switzerland: IEEE Computer Society.

Koshkina, M., & Van Breugel, F. (2003). *Verification of business processes for Web services.* Report CS-2003-11, York University.

Kradolfer, M., & Geppert, A. (1999). Dynamic Workflow Schema Evolution based on Workflow Type Versioning and Workflow Migration. *The International Conference on Cooperative Information Systems (CoopIS '99)*, (pp. 104-114).

Kripke, S. (1963). Semantical Considerations on Modal Logic. *Acta Philosophica Fennica (16)*, 83-94.

Kumar, K., & Narasipuram, M. M. (2006). Defining Requirements for Business Process Flexibility. *BPMDS 2006.*

Laird, L. M., & Brennan, M. C. (2006). *Software Measurement and Estimation: a Practical Approach (Quantitative Software Engineering Series).* Wiley-IEEE Computer Society Pr; 1 edition .

Langner, P., Schneider, C., & Wehler, J. (1998). Petri Net Based Certification of Event-Driven Process Chains. *The 19th International Conference on Application and Theory of Petri Nets* (pp. 286-305). Springer-Verlag London.

Lapadula, A., Pugliese, R., & Tiezzi, F. (2007). A Calculus for Orchestration of Web Services. *The 16th European Symposium on Programming (ESOP'07)* (pp. 33-47). Braga, Portugal: Springer Berlin Heidelberg.

Laroussinie, F., & Schnoebelen, P. (1995). A hierarchy of temporal logics with past. *Theoretical Computer Science 148(2)*, 303–324.

Latva-Koivisto, A. M. (2001). *Finding a complexity measure for business process models.*

Laue, R., & Awad, A. (2011). Visual suggestions for improvements in business process diagrams. *Journal of Visual Languages & Computing*, 385–399.

Le Moigne, J.-L. (1984). *La théorie du système général:théorie de la modélisation.* Presses Universitaires de France.

Lerner, B. S., Christov, S., Wise, A., & Osterweil, L. J. (2008). Exception Handling Patterns for Process Modeling. *4th international workshop on Exception handling (WEH '08)* (pp. 55-61). ACM New York.

Leymann, F. (2001). *Web Services Flow Language (WSFL 1.0).* Récupéré sur http://cin.ufpe.br/~redis/intranet/bibliography/standards/leymann-wsfl01.pdf

Leymann, F., & Roller, D. (2000). *Production Workflow: Concepts and Techniques.* Edition PTR Prentice Hall.

Li, B. (2003). Managing Dependencies in Component-Based Systems Based on Matrix Model. Dans *Proc. Of Net.Object.Days 2003* (pp. 22-25).

Lindvall, M., & Sandahl, K. (1998). Traceability aspects of impact analysis in object-oriented systems. *Journal of Software Maintenance: Research and Practice Vol. 10 Issue 1*, 37-57.

Liu, Y., Müller, S., & Xu, K. (2007). A static compliance-checking framework for business process models. *IBM Systems Journal Volume 46 Issue 2*, 335-361.

Lu, R., & Sadiq, S. (2007). A Survey of Comparative Business Process Modeling Approaches. *10th International Conference on Business Information Systems (BIS), number 4439 in LNCS* (pp. 82-94). Springer.

Malone, T. W., Crowston, K., Lee, J., Pentland, B., Dellarocas, C., Wyner, G., et al. (1999). Tools for Inventing Organizations: Toward a Handbook of Organizational Processes. *Journal of Management Science Vol.45 Issue 3*, 425-443.

Marcus, A., & Maletic, J. I. (2003). Recovering documentation-to-source-code traceability links using latent semantic indexing. *The 25th International Conference on Software Engineering (ICSE '03)* (pp. 125-135). IEEE Computer Society.

Massuthe, P., & Weinberg, D. (2008). Fiona: A tool to analyze interacting open nets. *The 15th German Workshop on Algorithms and Tools for Petri Nets (AWPN'08)* (pp. 99-104). CEUR Workshop Proceedings Vol. 380.

Matjaz, B. J. (2006). *Business Process Execution Language for Web Services BPEL and BPEL4WS 2nd Edition.*

McCabe, T. (1976). A Complexity Measure. *IEEE Transactions on Software Engineering Vol. 2 issus 4*, 308-320.

Mccall, J. A., Cavano, J. P., & Walters, G. (1977). *Factors in Software Quality 1, 2, and 3.*

Mendling, J. (2006). *Testing Density as a Complexity Metric for EPCs.* Austria: Technical Report JM-2006-11-15. Vienna University of Economics and Business Administration.

Mendling, J. (2008). *Metrics for Process Models : Metrics for Process Models Empirical Foundations of Verification, Error Prediction, and Guidelines for Correctness.* Springer Publishing Company.

Milner, R. (1980). *A Calculus of Communicating Systems .* Springer-Verlag .

Milner, R. (1989). *Communication and Concurrency.* Prentice-Hall International Series in Computer Science.

Milner, R. (1999). *Communicating and Mobile Systems: The Pi Calculus.* Cambridge University Press; 1st edition.

Minor, M., Schmalen, D., Koldehoff, A., & Bergmann, R. (2007). Structural adaptation of workflows supported by a suspension mechanism and by case-based reasoning. *16th IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE'07)*, (pp. 370-375). Evry, France.

Misra, J., & Cook, W. R. (s.d.). Récupéré sur orc.csres.utexas.edu/papers/OrcJSSM.pdf

Mooney, J., Beath, C., Fitzgerald, G., Ross, J., & Weill, P. (2003). Managing Information Technology for Strategic Flexibility and Agility: Rethinking Conceptual Models, Architecture, Development, and Governance. *24th International Conference on Information Systems (ICIS 2003).* Seattle, WA, USA.

Mordechai, B.-A. (2008). *Principles of the Spin Model Checker.* Springer Berlin Heidelberg.

Morley, C., Bia-Figueiredo, M., & Gillette, Y. (2011). *Processus métiers et S.I. - Gouvernance, management, modélisation.* DUNOD.

Morley, C., Hughes, J., Leblanc, B., & Hugues, O. (2007). *Processus métiers et S.I. : évaluation, modélisation, mise en oeuvre.* Edition DUNOD.

Morley, C., Hugues, J., Leblanc, B., & Hugues, O. (2004). *Processus Métiers et systèmes d'information : Evaluation, modélisation, mise en oeuvre.* Dunod.

Müller, R., Greiner, U., & Rahm, E. (2004). AGENT WORK: a workflow system supporting rule-based workflow adaptation. *Data & Knowledge Engineering Vol. 51(2)*, 223-256.

Nakajima, S. (2006). Model-Checking Behavioral Specification of BPEL Applications. *Electronic Notes in Theoretical Computer Science (ENTCS) Vol.151 Issue 2*, 89-105.

Nurcan, S. (2008). A Survey on the Flexibility Requirements Related to Business Processes and Modeling Artifacts. *The 41st Annual Hawaii International Conference on System Sciences (HICSS '08)* (pp. 378-388). Big Island, Hawaii, USA: IEEE Computer Society Washington.

Nurcan, S., Etien, A., Kaabi, R. S., Zoukar, I., & Rolland, C. (2005). A Strategy Driven Business Process Modelling Approach. *Special issue of the Business Process Management Journal on "Goal-oriented business process modeling", Emerald, 11:6.*

OASIS. (2003). *Collaboration-Protocol Profile and Agreement Specification,ebXML Trading-Partners Team Version 1.0.* Dans ebXML specification.

OASIS. (2007). *Business Process Execution Language for Web Services (BPE4WS 2.0).* Récupéré sur https://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wsbpel

OASIS Standard. (2007). *OASIS Web Services Business Process Execution Language (WSBPEL) TC.* Récupéré sur https://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wsbpel

OMG, O. M. (2011, January). *Business Process Model and Notation (BPMN) Version 2.0.* Récupéré sur http://www.omg.org/spec/BPMN/2.0/PDF

Onoda, S., Ikkai, Y., Kobayashi, T., & Komoda, N. (1999). Definition of Deadlock Patterns for Business Processes Work ow Models. *Thirty-second Annual Hawaii International Conference on System Sciences (HICSS '99)* (pp. 50-65). IEEE Computer Society.

Ouyang, C., Dumas, M., ter Hofstede, A. H., & Van der Aalst, W. M. (2006). From BPMN Process Models to BPEL Web Services. *IEEE International Conference on Web Services (ICWS'06)* (pp. pp. 285-292). IEEE Computer Society Washington, DC, USA.

Ouyang, C., Van der Aalst, W. M., Dumas, M., & ter Hofstede, A. H. (2006). *Translating BPMN to BPEL*.* Dans le rapport BPM-06.

Ouyang, C., Verbeek, E., van der Aalst, W. M., Breutel, S., Dumas, M., & ter Hofstede, A. H. (2005). WofBPEL: A Tool for Automated Analysis of BPEL Processes. *The Third International Conference On Service-Oriented Computing - (ICSOC'05)* (pp. 484-489). Amsterdam, Netherlands: Springer Berlin Heidelberg.

Pesic, M., & Van der Aalst, W. M. (2006). A declarative approach for flexible business processes management. *International conference on Business Process Management Workshops,BPM'06* (pp. 169-180). Springer-Verlag Berlin, Heidelberg.

Pesic, M., Schonenberg, H., & van der Aalst, W. M. (2007). DECLARE: Full Support for Loosely-Structured Processes. *The 11th IEEE International Enterprise Distributed Object Computing Conference (EDOC '07)* (pp. 287-300). Annapolis, Maryland, USA: IEEE Computer Society.

Pfleeger, S. L., & Bohner, S. A. (1990). A Framework for Software Maintenance Metrics. *The IEEE Transactions on Software Engineering* (pp. 320-327). San Diego, CA, USA: IEEE Computer Society.

Pistore, M., Traverso, P., Bertoli, P. G., & Marconi, A. (2005). Automated Synthesis of Composite BPEL4WS Web Services. *The IEEE International Conference on Web Services (ICWS '05)* (pp. pp. 293-301). IEEE Computer Society .

Pnueli, A. (1979). The temporal semantics of concurrent programs. *The International Symposium on Semantics of Concurrent Computation* (pp. 1-20). Evian, France: Springer Berlin Heidelberg.

Prandi, D., Quaglia, P., & Zannone, N. (2008). Formal analysis of BPMN via a translation into COWS. *The 10th international conference on Coordination models and languages* (pp. 249-263). Oslo, Norway: Springer Berlin Heidelberg.

Proth, J.-M., & Xie, X. (1995). *Les réseaux de Petri pour la conception et la gestion des systèmes de production.* Masson.

Pu, G., Zhao, X., Wang, S., & Qiu, Z. (2006). Towards the Semantics and Verification of BPEL4WS. *Electronic Notes in Theoretical Computer Science*, 33–52.

Puhlmann, F., & Weske, M. (2005). Using the pi-calculus for Formalizing Workflow Patterns. *The 3rd international conference on Business Process Management (BPM'05)* (pp. 153-168). Springer-Verlag Berlin, Heidelberg.

Rajabi, B. A., & Lee, S. P. (2010). Modeling and analysis of change management in dynamic business process. *International Journal of Computer and Electrical Engineering, 2 (1)*, 181-189.

Rajlich, V. (1997). A Model for Change Propagation Based on Graph Rewriting. *The International Conference on Software Maintenance (ICSM '97)* (pp. 84-91). Washington, DC, USA: IEEE Computer Society.

Ramadan, M., Elmongui, H. G., & Hassan, R. (2011). BPMN Formalisation using Coloured Petri Nets. *The 2nd GSTF Annual International Conference on Software Engineering & Applications (SEA'11).* Singapore, Singapore.

Regev, G., & Wegmann, A. (2005). A regulation-based view on business process and supporting system flexibility. *CAiSE'05 Workshops*, (pp. 91-98).

Regev, G., Soffer, P., & Schmidt, R. (2006). Taxonomy of Flexibility in Business Processes. *7th Workshop on Business Process Modeling, Development, and Support (BPMDS'06)*, (pp. 90-93). Luxemburg.

Reichert, M., & Dadam, P. (1998). ADEPTflex-supporting dynamic changes of workflows without losing control. *Journal of Intelligent Information Systems*, 93-129.

Reichert, M., Dadam, P., & Bauer, T. (2003). Dealing with forward and backward jumps in workflow management systems. *Software and Systems Modeling Volume 2, Issue 1*, 37-58.

Reichert, M., Rinderle, S., Kreher, U., & Dadam, P. (2005). Adaptive Process Management with ADEPT2. *The 21st International Conference on Data Engineering (ICDE '05)* (pp. 1113-1114). Tokyo, Japan: IEEE Computer Society.

Reijers, H. A. (2006). Workflow Flexibility: The Forlorn Promise. *15th IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE '06)* (pp. 271-272). IEEE Computer Society.

Reijers, H. A., & Vanderfeesten, I. T. (2004). Cohesion and Coupling Metrics for Workflow Process Design. *The 2nd International Conference on Business Process Management (BPM'04)* (pp. 290-305). Potsdam, Germany: Springer Berlin Heidelberg.

Reix, R. (2004). *Systèmes d'Information et Management des Organisations.* Edition VUIBERT.

Rinderle, S., Reichert, M., & Dadam, P. (2003). Evaluation of Correctness Criteria for Dynamic Workflow Changes. *The International Conference On Business Process Management (BPM'03)* (pp. 41-57). Eindhoven, The Netherlands: Springer Berlin Heidelberg.

Rinderle, S., Reichert, M., & Dadam, P. (2003). *On Dealing With Semantically ConflictingBusiness Process Changes.* Technical Report. University of Ulm, Ulm.

Rinderle, S., Reichert, M., & Dadam, P. (2004). Correctness criteria for dynamic changes in workflow systems: a survey. *Data & Knowledge Engineering*, 9-34.

Rinderle, S., Reichert, M., & Dadam, P. (2004). Disjoint and Overlapping Process Changes: Challenges, Solutions, Applications. *The 11th International Conference on Cooperative Information Systems (CoopIS'04)* (pp. 101-120). Agia Napa, Cyprus: Springer Berlin Heidelberg.

Rolón Aguilar, E., García, F., & Ruiz, F. (2006). Evaluation Measures for Business Process Models. *The 2006 ACM Symposium on Applied Computing (SAC'06)* (pp. 1567-1568). Dijon, France: ACM New York.

Rolón Aguilar, E., Ruiz, F., García, F., & Piattini, M. (2006). Applying Software Metrics to evaluate Business Process Models. *CLEI electronic journal, Vol.9 Number 1*.

Rosario, S., Benveniste, A., Haar, S., & Jard, C. (2006). *Net systems semantics of Web Services Orchestrations modeled in Orc.* Research Report IRISA, No. 1780 Istitut de Recherche en Informatique et Systèmes Aléatoires.

Ross, A. M., Rhodes, D. H., & Hastings, D. E. (2008). Defining Changeability: Reconciling Flexibility, Adaptability, Scalability, Modifiability, and Robustness for Maintaining System Lifecycle Value. *Systems Engineering*, 246-262.

Rozier, K. Y. (2011). Survey: Linear Temporal Logic Symbolic Model Checking. *Computer Science Review Volume 5 Issue 2*, 163-203.

Russell, N., & ter Hofstede, A. H. (2009). newYAWL: Towards Workflow 2.0. Dans T. o. II, *Special Issue on Concurrency in Process-Aware Information Systems* (pp. 79-97). Springer Berlin Heidelberg.

Russell, N., Ter Hofstede, A. H., & Mulyar, N. (2006). *Workflow ControlFlow Patterns: A Revised View.* BPM Center Report BPM-06-22.

Russell, N., Ter Hofstede, A. H., Edmond, D., & van der Aalst, W. M. (2005). Workflow Data Patterns: Identification, Representation and Tool Support. *International Conference on Conceptual Modeling (ER)*, (pp. 353-368). Klagenfurt, Austria.

Russell, N., van der Aalst, W. M., Ter Hofstede, A. H., & Edmond, D. (2004). Workflow Resource Patterns: Identification, Representation and Tool Support. *17th International Conference, CAiSE 2005* (pp. 216-232). Porto, Portugal: Advanced Information Systems Engineering.

Ryder, B. G., & Tip, F. (2001). Change impact analysis for object-oriented programs. *The 2001 ACM SIGPLAN-SIGSOFT workshop on Program analysis for software tools and engineering (PASTE '01)* (pp. 46-53). ACM New York.

Sadiq, S., & Orlowska, M. (1999). Architectural Considerations in Systems Supporting Dynamic Workflow Modification. *The Workshop on Software Architectures for Business Process Management at the 11 th Conf. on Advanced Information Systems Engineering (CaiSE'99).* Heidelberg, Germany.

Sadiq, S., Orlowska, M., & Sadiq, W. (2004). Data flow and validation in workflow modelling. *The the 15th Australasian database conference (ADC '04)* (pp. 207-214). Australian Computer Society.

Sadiq, W., & Orlowska, M. E. (1996). *Modeling and Verification of Workflow Graphs.* Australia: Technical Report No. 386 Department of Computer Science, The University of Queensland, Australia.

Sadiq, W., & Orlowska, M. E. (1997). On Correctness Issues in Conceptual Modeling of Workflows. *European Conference on Information Systems (ECIS '97).* Cork, Ireland.

Sadiq, W., & Orlowska, M. E. (1999). On Capturing Process Requirements ofWorkflow BasedBusiness Information Systems. *3rd International Conference on Business Information Systems BIS'99* (pp. 281-294). Poznan, Poland: Springer Verlag.

Salaün, G., Bordeaux, L., & Schaerf, M. (2004). Describing and reasoning on Web services using process algebra. *The IEEE International Conference on Web Services*, (pp. 43-50).

Salaün, G., Ferrara, A., & Chirichiello, A. (2004). Negotiation Among Web Services Using LOTOS/CADP. *The European Conference Web Services (ECOWS'04)* (pp. pp. 198-212). Erfurt, Germany: Springer-Verlag Berlin Heidelberg.

Sánchez-González, L., García, F., Mendling, J., & Ruiz, F. (2010). Quality Assessment of Business Process Models Based on Thresholds. *The Confederated International Conferences On the Move to Meaningful Internet Systems: OTM 2010* (pp. 78-95). Hersonissos, Crete, Greece: Springer Berlin Heidelberg.

Sánchez-González, L., García, F., Mendling, J., Ruiz, F., & Piattini, M. (2010). Prediction of Business Process Model Quality Based on Structural Metrics. *The 29th International Conference on Conceptual Modeling (ER'10)* (pp. 458-463). Vancouver, BC, Canada: Springer Berlin Heidelberg.

Sánchez-González, L., Ruiz, F., García, F., & Piattini, M. (2011). Improving Quality of Business Process Models. *The 6th International Conference on Evaluation of Novel Approaches to Software Engineering (ENASE'11)* (pp. 130-144). Beijing, China: Springer Berlin Heidelberg.

Scheer, A.-W. (2000). *ARIS: Business Process Modelling.* Springer-Verlag Berlin and Heidelberg .

Scheer, A.-W. (2010). *ARIS - Business Process Frameworks.* Springer; 3rd edition (June 29, 2010) .

Schmidt, K. (2000). LoLA: a low level analyser. *The 21st International Conference Application and Theory of Petri Nets (ICATPN'00)* (pp. 465-474). Aarhus, Denmark: Springer Berlin Heidelberg.

Schneider, K., & Prof.Dr Schmid, D. (1997). CTL and Equivalent Sublanguages of CTL. *The 5th International Conference on Computer Hardware Description Languages and their Applications* (pp. 40-59). Toledo, Spain: Springer US.

Schonenberg, H., Mans, R., Russell, N., Mulyar, N., & van der Aalst, W. (2008). *Process Flexibility: A Survey of Contemporary Approaches.* Advances in Enterprise Engineering.

Schonenberg, M., Mans, R., Russell, N., Mulyar, N., & van der Aalst, W. (2007). *Towards a Taxonomy of Process Flexibility (Extended Version).* Dans le rapport interne BPM Center Report BPMcenter.org, 2007.

Smith, H., & Fingar, P. (2003). *Workflow is just a Pi process.* BPTrends.

Soffer, P. (2005). On the Notion of Flexibility in Business Processes. *CAiSE'05 Workshops*, (pp. 35-42).

Song Dong, J., Liu, Y., Sun, J., & Zhang, X. (2006). Verification of Computation Orchestration via Timed Automata. *The 8th International Conference on Formal Engineering Methods* (pp. 226-245). Springer Verlag.

Speck, A., Feja, S., Witt, S., & Pulvermüller, E. (2011). Formalizing Business Process Specifications. *Computer Science and Information Systems 2011 Vol. 8, Issue 2*, 427-446.

SPINOV. (2006). *Modélisation des Processus Métiers: Etat de l'Art et Conseils Pratiques.* Rapport CITI.

Sun, P., & Jiang, C. (2009). Analysis of workflow dynamic changes based on Petri net. *Information and Software Technology Vol. 51, Issue 2*, 284–292.

Sun, Y., Huang, J. Z., & Meng, X. (2011). Integrating constraints to support legally flexible business processes. *Information Systems Frontiers archive Volume 13 Issue 2*, 171-189.

Takemura, T. (2008). Formal Semantics and Verification of BPMN Transaction and Compensation. *The IEEE Asia-Pacific Services Computing Conference (APSCC '08)*, (pp. 284-290). Los Alamitos.

Tantitharanukul, N. (2010). Detecting deadlock and multiple termination in BPMN model using process automata. *International Conference on Electrical Engineering/Electronics Computer Telecommunications and Information Technology (ECTI-CON 2010)*, (pp. 478-482). Chaing Mai.

Tantitharanukul, N., & Jumpamule, W. (2010). Detection of LiveLock in BPMN Using Process Expression. *The 4th International Conference on Advances in Information Technology (IAIT'2010)* (pp. 164-174). Bangkok, Thailand: Advances in Information TechnologyCommunications in Computer and Information Science.

Thatte, S. (2001). *XLANG - Web Services For Business Process Design.* Récupéré sur http://www.gotdotnet.com/team/xml_wsspecs/xlang-c/default.htm

The Workflow Management Coalition. (2008). *Final XPDL 2.1 Specification.* dans le Rapport de spécification numéro WFMC-TC-10.

Turver, R. J., & Munro, M. (1994). An early impact analysis technique for software maintenance. *Journal of Software Maintenance:Research and Practice, Volume 6, No. 1*, 35-52.

Van Belle, J.-P. (2004). A proposed framework for the analysis and evaluation of business models. *The 2004 annual research conference of the South African institute of computer scientists and information technologists on IT research in developing countries (SAICSIT '04)*, (pp. 210-215). South African Institute for Computer Scientists and Information Technologists , Republic of South Africa.

van der Aalst, W. (1998). The Application of Petri Nets to Workflow Management. *The Journal of Circuits, Systems and Computers, Vol. 8, No. 1*, 21-66.

van der Aalst, W. (1999). Formalization and verification of event-driven process chains. *Information and Software Technology, 41(10)*, 639-650.

van der Aalst, W. (2001). How to handle dynamic change and capture management information? An approach based on generic workflow models. *Journal of Computer Systems Science Engineering 15(5)*.

van der Aalst, W. M. (2007). Challenges in Business Process Analysis. *The 9th International Conference On Enterprise Information Systems (ICEIS'07)* (pp. 27-42). Funchal, Madeira: Springer Berlin Heidelberg.

van der Aalst, W. M., & Song, M. (2004). Mining Social Networks: Uncovering Interaction Patterns in Business Processes. *The 2nd International Conference on Business Process Management (BPM'04)* (pp. 244-260). Potsdam, Germany: Springer Berlin Heidelberg.

Van der Aalst, W. M., Ter Hofstede, A. H., & Weske, M. (2003). Business Process Management: A Survey. *The international conference on Business process management, BPM 2003* (pp. 1–12). LNCS 2678, 2003. Springer-Verlag Berlin Heidelberg.

van der Aalst, W., & ter Hofstede, A. (2005). YAWL: yet another workflow language. *Information Systems Volume 30, Issue 4*, 245-275.

van der Aalst, W., & Van Dongen, B. F. (2002). Discovering Workflow PerformanceModels from Timed Logs. *The International Conference on Engineering and Deployment of Cooperative Information Systems (EDCIS' 02)* (pp. pp. 45-63). Springer Berlin Heidelberg.

van der Aalst, W., Basten, T., Verbeek, H. M., Verkoulen, P. A., & Voorhoeve, M. (1999). Adaptive workflow - On the interplay between flexibility and support. *Filipe, J.; Cordeiro, J.: Proceedings of the first International Conference on Enterprise Information Systems, Vol. 2*, (pp. 353-360). Setúbal, Portugal.

van der Aalst, W., ter Hofstede, A., Kiepuszewski, B., & Barros, A. (2003). Workflow Patterns. *Distributed and Parallel Databases, 14(3):5-51*, 5-51.

van Dongen, B. F., & Jansen-Vullers, M. H. (2005). Verification of SAP Reference Models. *The 3rd International Conference On Business Process Management (BPM'05)* (pp. 464-469). Nancy, France: Springer Berlin Heidelberg.

van Dongen, B. F., de Medeiros, A. K., Verbeek, H., Weijters, A. J., & van der Aalst, W. M. (2005). The ProM Framework: A New Era in Process Mining Tool Support. *The 26th International Conference on Applications and Theory of Petri Nets (ICATPN' 05)* (pp. 444-454). Miami, USA: Springer Berlin Heidelberg.

Vanderfeesten, I. T., Cardoso, J., & Reijers, H. A. (2007). A weighted coupling metric for business process models. *The Workshop of 19th International Conference on Advanced Information Systems Engineering (CAiSE'07)*, (pp. 41-44). Trondheim, Norway.

Vanderfeesten, I. T., Reijers, H. A., & van der Alst, W. (2008). Evaluating workflow process designs using cohesion and coupling metrics. *Computers in industry*, 420-437.

Vaz, C., & Ferreira, C. (2007). *Formal verification of workflow patterns with spin.* Technical Report, April 2007. INESC-ID Tec. Rep. 12.

Vaz, C., & Ferreira, C. (2007). Towards Automated Verification of Web Services. *The IADIS International Conference on WWW/Internet 2007.*

Verbeek, E., van Hattem, M., Reijers, H., & de Munk, W. (2005). Protos 7.0: Simulation Made Accessible. *The 26th International Conference On Applications and Theory of Petri Nets (ICATPN'05)* (pp. 465-474). Miami, USA: Springer Berlin Heidelberg.

Verbeek, H., Basten, T., & van der Aalst, W. (2001). Diagnosing Workflow Processes Using Woflan. *THE COMPUTER JOURNAL*.

Wagner, G. (2005). Rule Modeling and Markup. *Reasoning Web* (pp. 251-274). Springer.

Wagner, G., Giurca, A., & Lukichev, S. (2006). A Usable Interchange Format for Rich Syntax Rules Integrating OCL, RuleML and SWRL (2006). *Reasoning on the Web (RoW2006).*

Walton, C. D. (2005). Model Checking Agent Dialogues. *The 2nd International Workshop on Declarative Agent Languages and Technologies II (DALT'04)* (pp. 132-147). New York, NY, USA: Springer Berlin Heidelberg.

Wang, J. (1998). *Timed Petri Nets: Theory and Application.* Springer; 1998 edition .

Wang, S., & Capretz, M. A. (2009). A Dependency Impact Analysis Model for Web Services Evolution. *The IEEE International Conference on Web Services (ICWS'09)* (pp. 359-365). Los Angeles, CA, USA: IEEE Computer Society.

Weber, B., Reichert, M., & Rinderle-Ma, S. (2008). Change patterns and change support features – Enhancing flexibility in process-aware information systems. *Data & Knowledge Engineering Volume 66 Issue 3*, 438-466.

Weber, B., Wild, W., Lauer, M., & Rei, M. (2006). Improving Exception Handling by Discovering Change Dependencies in Adaptive Process Management Systems. *BPM 2006 International Workshops, BPD,*, (pp. 93-104). Vienna, Austria.

Weerawarana, S., Curbera, F., Leymann, F., Storey, T., & Ferguson, D. F. (2005). *Web Services Platform Architecture: SOAP, WSDL, WS-Policy, WS-Addressing, WS-Bpel, WS-Reliable Messaging and More.* Edition Prentice Hall Computer.

Weijters, A. J., & van der Aalst, W. M. (2003). Rediscovering workflow models from event-based data using little thumb. *Journal of Integrated Computer-Aided Engineering*, 151-162.

Weske, M. (1998). Flexible modeling and execution of workflow activities. *The 31th Annual Hawaii International Conference on System Sciences (HICSS'98)* (pp. 713-722). IEEE Computer Society.

Weske, M. (2000). *Workflow Management Systems: Formal Foundation, Conceptual Design, Implementation Aspects.* University of M□unster, Habil Thesis.

Weske, M. (2001). Formal Foundation and Conceptual Design of Dynamic Adaptations in a Workflow Management System. *The 34th Annual Hawaii International Conference on System Sciences (HICSS '01).* IEEE Computer Society .

Wolfgang, T. (1989). Computation tree logic and regular ω-languages. *The School/Workshop of Linear Time, Branching Time and Partial Order in Logics and Models for Concurrency* (pp. 690-713). Noordwijkerhout, Netherlands: Springer Berlin Heidelberg.

Wombacher, A., Fankhauser, P., & Neuhold, E. (2004). Transforming BPEL into Annotated Deterministic Finite State Automata for Service Discovery. *The IEEE International Conference on Web Services (ICWS'04)* (pp. 316-323). EE Computer Society.

Wong, B., & Jeffery, R. (1995). *Quality Metrics : ISO9126 and Stakeholder Perceptions.*

Wong, B., & Jeffery, R. (2001). Cognitive Structures of Software Evaluation: A Means-End Chain Analysis of Quality. *The3rd International Conference on Product Focused Software Process Improvement* (pp. 6-26). Kaiserslautern, Germany: Springer Berlin Heidelberg.

Wong, P. Y., & Gibbons, J. (2008). A Process Semantics for BPMN. *The 10th International Conference on Formal Methods and Software Engineering (ICFEM'08)* (pp. 355-374). Kitakyushu-City, Japan: Springer-Verlag Berlin, Heidelberg.

Woodcock, J., & Davies, J. (1996). *Using Z: specification, refinement, and proof.* Prentice Hall.

Workflow Management Coalition. (1999). *Workflow Management Coalition Terminology & Glossary (Document No. WFMC-TC-1011).* Brussels.

Yang, D., & Zhang, S.-s. (2003). Approach for workflow modeling using pi-calculus. *Journal of Zhejiang University Science*, 643-650.

Yang, Y., Tan, Q., & Xiao, Y. (2005). Verifying web services composition based on hierarchical colored petri nets. *The 1st international workshop on Interoperability of heterogeneous information systems (IHIS '05)* (pp. 47-54). ACM Press.

Ye, J., Sun, S., Wen, L., & Song, W. (2008). Transformation of BPMN to YAWL. *The International Conference on Computer Science and Software Engineering (CSSE'08)*, (pp. 354-359). Wuhan, China.

Yi, X., & Kochut, K. J. (2004). A CP-nets-based Design and Verification Framework for Web Services Composition. *The IEEE International Conference on Web Services (ICWS '04)* (pp. 756-760). IEEE Computer Society.

Yu, T.-L., Goldberg, D. E., Sastry, K., Lima, C. F., & Pelikan, M. (2009). Dependency structure matrix, genetic algorithms, and effective recombination. *Journal of Evolutionary Computation Vo. 17 Issue 4*, 595-626.

Zeng, L., Flaxer, D., Chang, H., & Jeng, J.-J. (2002). PLM flow-Dynamic Business Process Composition and Execution by Rule Inference. *The Third International Workshop on Technologies for E-Services TES '02* (pp. 141-150). Springer-Verlag London.

Zeng, L., Ngu, A., Benatallah, B., & O'Dell, M. (2001). An agent-based approach for supporting cross-enterprise workflows. *The 12th Australasian database conference (ADC'01)* (pp. 123-130). IEEE Computer Society.

Zhao, X., & Liu, C. (2013). Version management for business process schema evolution. *Information Systems Volume 38, Issue 8*, 1046–1069.

zur Muehlen, M. (2004). *Workflow-based process controlling: foundation, design, and application of workflow-driven process information systems.* Process information system.

zur Muehlen, M., & Indulska, M. (2010). Modeling languages for business processes and business rules: A representational analysis. *Information Systems Volume 35, Issue 4*, 379-390.