Proposition d'une méthode de spécification d'une architecture orientée services dirigée par le métier dans le cadre d'une collaboration inter-organisationnelle

Youness Lemrabet

N° d'ordre : 188

ECOLE CENTRALE DE LILLE

THESE

présentée en vue
d'obtenir le grade de

DOCTEUR

en

Génie industriel

par

Youness LEMRABET

DOCTORAT DELIVRE PAR L'ECOLE CENTRALE DE LILLE

Titre de la thèse :
Proposition d'une méthode de spécification d'une architecture orientée services dirigée par le métier dans le cadre d'une collaboration inter-organisationnelle

Soutenue le 7 Juin 2012 devant le jury d'examen :

Président	*Daniel Jolly, Professeur, Université d'Artois*
Rapporteur	*Yves Ducq, Professeur, Université de Bordeaux 1*
Rapporteur	*Slimane Hammoudi, Enseignant-Chercheur HDR, ESEO Université d'Angers*
Examinateur	*Raul Poler, Professor, Universidad Politécnica de Valencia*
Examinateur	*Daniel Jolly, Professeur, Université d'Artois*
Examinateur	*Michel Bigand, Maître de Conférences HDR, EC-Lille*
Directeur de thèse	*Jean-Pierre Bourey, Professeur, EC-Lille*

Thèse préparée dans le Laboratoire de Modélisation et de Management des Organisations EA 4344

École Doctorale SPI 287 (EC Paris, EC Lille, INT Évry)

PRES Université Lille Nord-de-France

Stephen Hawking et *Leonard Mlodinow* « La question de la réalité d'un modèle ne se pose pas, seul compte son accord avec la réalité », Y-a-il un grand architecte dans l'Univers ?

A mon père

Remerciements

J'ai choisi de faire une thèse CIFRE dans une PME pour acquérir une vision transversale de l'entreprise. Dans SRD et pendant une période de 3 ans, j'ai eu la chance d'approfondir et d'ajuster mon travail de thèse par le quotidien de l'entreprise et de ces problèmes. Je tiens tout d'abord à remercier Jean-Philippe Santi et Philippe Dubois d'avoir accepté d'entreprendre cette thèse CIFRE et de m'avoir accueillie au sein de l'entreprise SRD. Je tiens aussi à remercier Thomas Graf et Corinne Laborde pour avoir appuyé mon travail de recherche après le rachat de l'entreprise.

Mes remerciements vont particulièrement à Yves Ducq et Slimane Hammoudi qui ont accepté de rapporter sur mon travail de thèse, et qui ont ainsi contribué chacun à apporter une valeur ajoutée à mes travaux. Je remercie aussi Raul Poler et Daniel Jolly qui ont accepté d'examiner les travaux présentés dans ce manuscrit de thèse.

Je tiens à remercier tout particulièrement mon directeur de thèse, Jean-Pierre Bourey, pour la confiance qu'il a placée en moi. Avoir été sous sa direction a été une expérience extrêmement riche. Je vous suis reconnaissant de m'avoir fait bénéficier de votre grande compétence dans le domaine de la modélisation d'entreprise et je suis fier de votre amitié.

Je remercie tout aussi vivement Michel Bigand, mon co-directeur de thèse pour ces précieux conseils et sa capacité à prendre du recul qui ont été des atouts importants pour l'aboutissement de cette thèse.

Je tiens à remercier très sincèrement l'ensemble des membres du laboratoire LM²O pour leur amitié et leurs encouragements. Je suis très reconnaissant à Nordine Benkeltoum qui a eu le courage et la patience pour relire mon mémoire de thèse. Ma reconnaissance va bien sûr à mon collègue Hui Liu qui m'a accompagné, lors de nos innombrables discussions, et argumentations contradictoires, dans l'évolution et la maturation de ma thématique de recherche.

Je souhaite spécialement remercier Ahmed Mekki et tous mes amis et mes cousins pour leurs amitiés et leurs encouragements.

Je n'oublierai pas de remercier les visages inconnus qui ont défilé devant moi et partagé ma table à la médiathèque Jean-Lévy et la bibliothèque de la fac de médecine. J'espère qu'un jour les bibliothèques resteront ouvertes le soir.

Enfin, je tiens à remercier ma famille et mes parents pour leur soutien sans faille, ma reconnaissance pour vous et éternelle.

Sommaire

Liste des illustrations

Introduction

Chapitre I

Chapitre II

Chapitre III

Chapitre IV

Chapitre V

Conclusions et perspectives

Introduction

1. INTRODUCTION GÉNÉRALE

L'accélération des échanges de biens et services à l'échelle mondiale oblige les organisations à adopter une vision décloisonnée et élargie de leurs frontières aussi bien au niveau métier que technologique. Les théoriciens de la contingence ont longtemps mis en exergue le fait qu'il existait une structuration réciproque entre l'organisation et son environnement (Burns et al., 1971; Lawrence et al., 1989). Des travaux plus récents ont montré que les organisations contemporaines collaboraient de plus en plus avec leurs partenaires (Segrestin, 2006). Divers auteurs se sont penchés sur les raisons justifiant la coopération interentreprises. Pour les partisans de la théorie des jeux, le comportement coopératif est la stratégie qui est la plus efficace à long terme (Axelrod, 1992). D'autres recherches ont souligné que les coopérations étaient principalement motivées par l'acquisition de savoirs (Conner et al., 1996) ou de ressources (Quelin, 1994). Plus récemment, des travaux ont même érigé l'ouverture au statut de nécessité pour la croissance et la performance (Chesbrough, 2003). En effet, la dimension ouverte permet aux organisations d'être plus réactives face aux changements que leur impose leur environnement ainsi que dans leur capacité à identifier les besoins et les attentes de leurs partenaires. Dans le nouvel environnement économique mondial, l'interopérabilité est l'un des principaux résultats que les entreprises collaboratives doivent atteindre (Bourey et al., 2007).

Dans nos travaux, on met exergue deux dimensions nécessaires pour la croissance et la performance des entreprises: l'interopérabilité et l'agilité (alignement métier-IT). Pour faire face à ces défis, les entreprises ont investi massivement dans les méthodes, les cadres d'architecture et les infrastructures technologiques afin de devenir plus agiles et interopérables.

L'émergence de la Gestion des Processus Métiers (*Business Process Management - BPM*), de l'Architecture Orientée Services (*Service Oriented Architecture - SOA*), de l'Architecture Dirigée par les Modèles (*Model Driven Architecture - MDA*) et de l'Interopérabilité Dirigée par les Modèles (*Model Driven Interoperability - MDI*) a été accompagnée par la promesse de la mise en place d'une entreprise efficace, agile et interopérable (Liu, 2011). En raison de la difficulté à mettre en place ces approches dans les organisations et des débats qui divisent les chercheurs et les praticiens sur le comment faire, il est impératif de fixer un cadre et des fondements, au sens d'une méthodologie et des règles. Il s'agit de proposer une méthodologie déterminant les typologies et les différents niveaux d'expression sémantique de ces approches.

L'intérêt de ce travail est de proposer une méthode qui assure l'efficacité et l'efficience d'une collaboration d'entreprises, en abordant essentiellement les problématiques d'alignement métier-IT et d'interopérabilité. Ce travail s'inscrit dans le cadre de la question de recherche suivante : Comment spécifier une architecture orientée services dirigée par le métier dans le cadre d'une

collaboration inter-organisationnelle ? Pour répondre à cette question on commence par expliquer les facteurs techniques et non techniques qu'il faut prendre en considération afin de construire un système interopérable et flexible capable d'assurer la collaboration entre plusieurs partenaires.

La contribution de cette thèse consiste à trouver un processus pour combiner intelligemment les capacités des trois approches BPM, SOA et MDA/MDI. Dans un premier temps, pour aider à une meilleure compréhension des problématiques d'interopérabilité et d'alignement métier-IT. Ensuite, pour proposer une méthode de conception outillée encadrant les efforts des entreprises, notamment les PME voulant maîtriser leurs chaînes de valeurs, depuis la définition des exigences métier jusqu'à l'implémentation de la solution orientée service.

2. PÉRIMÈTRE DE LA MÉTHODE

La méthode proposée est basée sur une architecture orientée service dirigée par le métier qui respecte des principes de SOA et BPM. On utilise les éléments du modèle conceptuel de la Figure 1 pour décrire l'ensemble des concepts du périmètre de la méthode. Ce modèle est inspiré du standard (ISO/IEC/IEEE-42010, 2011) et du cadre d'architecture d'entreprise TOGAF (*The Open Group Architecture Framework*) version 9 (TOGAF, 2009).

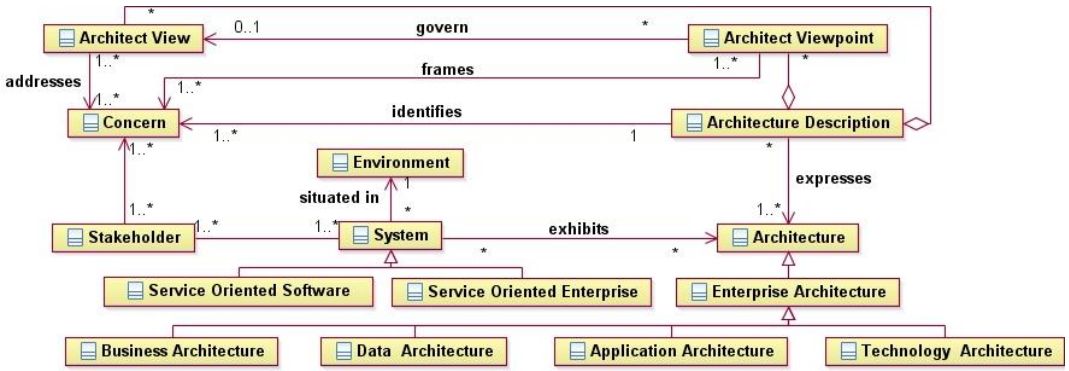


Figure 1. Modèle conceptuel de la description d'une architecture adapté à partir de (TOGAF, 2009; ISO/IEC/IEEE-42010, 2011)

La Figure 1 pose le contexte nécessaire à la compréhension de la méthode proposée, c'est une description de l'architecture dans laquelle la méthode peut être utilisée. Chaque système a une architecture qui est associée à une description qui peut être vue simultanément sous différentes vues. Ces vues représentent les différentes préoccupations des différentes parties prenantes. La méthode considère deux problématiques : l'interopérabilité et l'alignement entre le métier et l'IT (*Information Technology*) du système cible. Ces deux considérations peuvent être traitées dans chacun des quatre types d'architecture de TOGAF (TOGAF, 2009) : (1) métier, (2) d'applications, (3) des données et (4) technique.

La méthode proposée peut être utilisée dans deux types d'environnements: soit dans un environnement collaboratif entre plusieurs participants appartenant à différentes organisations voulant mettre en place une solution orientée service. Soit dans un environnement de collaboration entre plusieurs entités internes d'une organisation, dans ce cas la méthode peut s'inscrire dans la mise en place d'un cadre d'architecture d'entreprise (ex : TOGAF) pour porter l'utilisation de l'architecture orientée service à l'ensemble de l'entreprise (i.e., créer une entreprise orientée services). Cependant, la méthode traite seulement certains aspects des architectures métier et d'applications de TOGAF. Elle prend en compte deux vues[1] sur le système étudié: une vue métier et une autre IT et adopte deux points de vue ou perspectives[2] : service et processus.

La méthode montre comment utiliser les notions de processus et de service proposées respectivement par BPM et SOA dans le cadre d'une démarche MDI pour dépasser certaines barrières conceptuelle et technologique de l'interopérabilité. Elle explique comment identifier, spécifier et réaliser les processus et les services de collaboration entre plusieurs participants. Pour cela, on propose d'adopter une vision transversale de l'entreprise centrée sur les processus métiers. Cette vision permet d'améliorer les processus métiers du point de vue de l'utilisateur final. Ensuite l'approche MDA est utilisée comme un fil conducteur pour synchroniser les modèles des processus métiers découverts à l'aide de l'approche BPM avec ceux des services identifiés dans la démarche SOA. Dans ce schéma, les processus métiers assurent l'interopérabilité au niveau métier tandis que l'utilisation des services réutilisables, des standards et des architectures préconisés par SOA soutiennent l'interopérabilité au niveau IT.

Afin de positionner notre méthode par rapport aux différents types de méthodes SOA existantes. On adopte la classification établie par (Cheliah et al., 2012) qui ont identifié trois catégories de méthodes :

- **Les modèles de maturité de SOA :** ils prennent en compte plusieurs dimensions pour évaluer la maturité de l'organisation. Chaque dimension contient des objectifs à remplir et des étapes à compléter. Ces dimensions couvrent plusieurs domaines comme : le métier, l'organisation, le management, la gouvernance, l'architecture, la technologie, les opérations.
- **Les modèles de gouvernances SOA :** ils définissent comment les services sont délivrés ? et comment les organisations sont influencées par la mise en place de SOA ?

[1] Une vue d'architecture représente un ou plusieurs modèles d'architecture qui exprime l'architecture du système à partir de la perspective (point de vue) d'une partie prenante (ISO/IEC/IEEE-42010, 2011)

[2] Un point de vue ou une perspective est un ensemble de conventions pour construire, interpréter, utiliser et analyser les types de vue d'architecture (ISO/IEC/IEEE-42010, 2011).

- **Les modèles de livraison de services :** ils définissent comment les services sont identifiés et délivrés dans le cadre d'approches descendantes ou ascendantes.

La méthode qu'on propose appartient aux modèles de livraison de services, elle est basée sur une approche descendante dirigée par le métier et consiste à décomposer les processus métiers de la collaboration en patterns. Ces patterns sont représentés avec des services métiers publics partagés et réutilisables qui sont utilisés dans la construction de la solution SOA.

3. HYPOTHÈSES DE TRAVAIL

On propose une méthode précise, mise en œuvre dans un contexte théorique précis, pour répondre à des problèmes précis. On postule que les hypothèses suivantes sont vérifiées :

- Les partenaires de la collaboration n'ont jamais collaboré auparavant.
- Les participants de la collaboration ne disposent pas d'un inventaire de services.
- L'ensemble des partenaires utilise (*Business Process Modeling and Notation – BPMN*) (OMG01, 2011) pour modéliser les processus collaboratifs et (*Service Oriented Architecture Modeling Language – SoaML)* (OMG02, 2009) pour décrire l'architecture des services publics de collaboration.
- Une plateforme basée sur les technologies Web-services est utilisée pour illustrer les exemples fournis.

De plus, on se limite à l'identification et à la conception de processus et de services utilisés dans le cadre d'une collaboration.

4. CAS D'ÉTUDE

Cette thèse prend appui sur une recherche intervention de trois ans dans l'entreprise SRD[3] éditeur de progiciels de gestion de l'activité VAD (Vente À Distance). Dans ce cadre, on a été amené à travailler sur des projets desquels la méthode proposée a été dérivée. Comme la nature de notre question de recherche justifie l'utilisation d'une approche inductive. On adopte une méthodologie de recherche basée sur deux études de cas. En effet, l'étude de cas comme méthode de recherche est largement acceptée pour décrire, expliqué, prédire et contrôler les processus inhérents à des phénomènes individuels ou collectifs (Gagnon, 2012)

Plus précisément, les cas d'études proposés illustrent comment concevoir une solution orientée services et dirigée par le métier en se basant sur le contexte du projet ASICOM (Architecture de Système d'information Interopérable pour les industries du COMmerce)[4]. Ce

[3] http://www.srd.fr/

[4] Projet doublement labellisé par les pôles de compétitivité PICOM et Nov@log

projet avait pour ambition de produire une solution interopérable permettant des échanges simplifiés entre les entreprises commerciales, notamment les entreprises de VAD et de logistique pour leur permettre de bénéficier du régime Entrepôt Sous Douane (ESD). Les entreprises bénéficiaires de ce régime s'acquittent des frais de douane[5] sur les produits importés uniquement au moment du déstockage de la marchandise et non dès son arrivée sur le territoire national (ASICOM, 2007). Ainsi, la marchandise stockée dans un ESD en attente de mise en consommation future n'est pas assujettie aux droits de douane et taxes immédiatement. Ceci permet à l'importateur (i.e., l'entreprise commerciale) de faire des gains de trésorerie et d'optimiser l'importation de ses produits.

Ci-après une brève introduction des participants dans les collaborations étudiées :

Donneur d'ordre (importateur) : pour pouvoir intervenir auprès de la douane en cas de problème, l'importateur supervise les déclarations douanières en archivant les documents nécessaires pour faire la déclaration (factures, commandes fournisseur, certificat d'origine, etc.). Puis, il transmet une copie des documents à un prestataire qui se charge de saisir les déclarations dans l'application de son commissionnaire en douane. Actuellement, l'importateur utilise le courrier électronique et les fichiers *Excel* pour suivre l'état de ses déclarations.

À terme, l'importateur envisage de se dispenser de son prestataire. Il espère saisir les informations nécessaires pour établir les déclarations en interne en complétant la saisie de la déclaration qui sera initialement saisie en amont par son fournisseur. L'importateur compte profiter du passage à un ESD pour mettre en place cette nouvelle méthode de travail.

Le gestionnaire d'entrepôt (stockiste) : Il souhaite proposer à ses clients de stocker leurs marchandises dans un ESD. Pour pourvoir gérer un ESD, il doit satisfaire plusieurs exigences pour obtenir l'agrément de la douane. Par exemple, mettre en place un système de gestion d'entrepôt capable de distinguer les produits sous régime ESD du reste de la marchandise. De plus, il souhaite faire évoluer son système afin d'automatiser les échanges nécessaires à la demande de l'autorisation de mise en ESD et a la notification du donneur d'ordre de la disponibilité de sa marchandise.

[5] Les frais de douane sont d'une façon générale constitués par l'ensemble des impositions applicables aux marchandises importées. Plus précisément, ces impositions sont constituées par des droits de douane et par la TVA due en raison de l'importation. Les marchandises placées en entrepôt douanier étant juridiquement considérées comme situées à l'extérieur de l'Union douanière, les différentes impositions normalement exigibles du fait de l'importation ne sont pas applicables. Le régime de l'entrepôt n'est pas seulement suspensif de la perception des droits et taxes. Il permet également de ne pas rendre obligatoire la production des licences d'importation lors du placement des marchandises en entrepôt. Les impositions ne sont donc acquittées qu'au moment de l'importation définitive de la marchandise.

Actuellement, le stockiste utilise un fichier Excel pour gérer son quotidien (ex : planning d'arrivée des camions) et un ERP (Enterprise Resource Planning) hébergé chez un prestataire pour gérer ses entrepôts et le stock de produits.

Commissionnaire (transport et/ou douane) : actuellement le commissionnaire effectue les formalités douanières et est gestionnaire des entrepôts de ses clients. Le commissionnaire veut développer un nouveau partenariat avec des stockistes. Dans cette nouvelle configuration, le commissionnaire sera l'intermédiaire pour déposer les déclarations de mise en ESD auprès de la douane et c'est au stockiste qu'incombe la gestion de l'ESD. En plus, le commissionnaire apporte la caution nécessaire pour faire une déclaration auprès de la douane. Pour la mise en place de cette nouvelle activité le commissionnaire envisage d'améliorer son ERP pour intégrer les échanges d'informations avec les autres partenaires.

Fournisseur : c'est une filiale d'un groupe de textile français au Bangladesh qui confectionne des vêtements pour enfants pour des donneurs d'ordre en Europe.

Douane : afin de répondre aux besoins exprimés par les entreprises qui souhaitent plus de rapidité dans les démarches de dédouanement et une réduction du coût du dédouanement. L'administration de douane a décidé d'informatiser son système de dédouanement afin de permettre aux entreprises de compléter des procédures douanières soit en DTI[6] (*Direct Trader Interface*) soit en EDI (*Electronic Data Interchange*) (échange de données informatisées). En fait, la douane référence peu d'offreurs de solutions EDI. Elle souhaite que les entreprises, notamment les PME (Petites et Moyennes Entreprises) bénéficient d'un large éventail de solutions EDI.

5. STRUCTURE DE LA THESE

On a décidé d'organiser ce manuscrit de façon non conventionnelle en proposant une étude bibliographique diffuse au sein des différents chapitres. Le manuscrit de thèse est composé de cinq chapitres :

1) Dans le **premier chapitre,** on explique les domaines d'application de SOA et on introduit notre point de vue sur la notion de service. De plus, on se focalise sur la perspective conception d'architecture de SOA. Pour cela, on met en évidence les propriétés fondamentales d'une architecture orientée services à travers ses principes et certains de ses patterns de conception.
2) Dans le **deuxième chapitre,** on adopte la perspective métier de SOA pour étudier la contribution d'une architecture orientée services dans l'alignement métier-IT. Dans ce but on met en évidence les synergies entre SOA et BPM en proposant un modèle

[6] Échange de données via un formulaire en ligne

ad hoc nommé modèle EMT (Économie, Méthodologie et Technologie). Ce modèle prend en considération les aspects économiques, méthodologiques et technologiques justifiant la mise en place des approches BPM et SOA pour améliorer l'agilité des organisations. Ensuite, on propose un cadre de référence contenant les différents éléments à prendre en compte lors de l'implémentation des approches BPM et SOA.

3) Le **troisième chapitre** permet de faire la transition entre notre vision de SOA sur les deux perspectives : conception d'architecture (cf. Chapitre I) et métier (cf. Chapitre II) et la méthode de conception de services orientée métier qu'on propose pour modéliser les processus collaboratifs (cf. Chapitre IV) et les services publics de la collaboration (cf. Chapitre V). Dans ce court chapitre on positionne notre travail dans le cadre d'interopérabilité d'entreprise (*Enterprise Interoperability Framework - EIF*) (Chen et al., 2006). Puis, on construit une grille de lecture à deux dimensions à partir du cadre d'interopérabilité d'ATHENA (ATHENA01, 2007) et de l'approche MDI (Bourey et al., 2007). Cette grille montre comment la combinaison de SOA et MDI aide à dépasser les barrières conceptuelles et technologiques de l'interopérabilité. La Figure 2 représente le positionnement de la méthode de conception qu'on propose sur les différents niveaux de MDI et sur les aspects d'interopérabilité processus et service du cadre d'interopérabilité EIF.

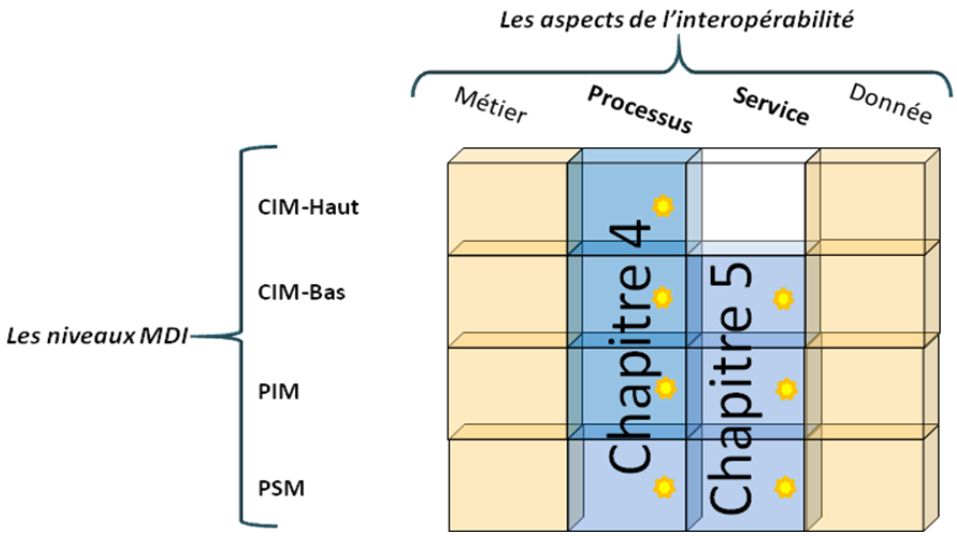


Figure 2. Positionnement de la méthode de conception orientée services et dirigée par le métier

4) À travers un cas d'étude, on propose dans le **quatrième chapitre** une méthode descendante (top-down) pour spécifier les processus métiers collaboratifs avec le formalisme BPMN 2.0. Cette méthode s'articule autour de sept étapes et s'inscrit dans le cadre de l'approche MDI avec plusieurs modèles hiérarchiques. Dans ce

chapitre, on démontre l'influence du choix de l'approche d'interopérabilité (intégrée, unifiée, fédérée) et du type de la collaboration (coordination, vue centralisée de la coopération, vue distribuée de la coopération) sur la conception des processus collaboratifs au niveau PIM de la démarche MDI.

5) Le **cinquième chapitre** est la suite de la méthode de conception qui se focalise sur l'aspect Service de la grille d'interopérabilité. Dans ce chapitre, on commence par réaliser une transformation de modèle entre les formalismes BPMN et SoaML afin d'apporter une nouvelle preuve de la complémentarité entre les deux approches BPM et SOA. Partant de ce fait, on propose un cas d'étude pour expliquer comment identifier les services publics de la collaboration à partir des processus collaboratifs décrits dans le chapitre précédent. Ensuite, on se sert de cet exemple pour illustrer comment utiliser les concepts SoaML pour spécifier et réaliser des services et pour montre en quoi la méthode de conception qu'on propose permet de respecter les principes SOA. Enfin, on a choisit d'utiliser les web-services comme plateforme cible pour générer le squelette de la solution SOA.

CHAPITRE I : Perspective conception d'architecture de SOA

1. INTRODUCTION DU CHAPITRE

La notion d'architecture orientée services (*Service Oriented Architecture - SOA*) a été proposée par Yefim V. Natis analyste du Gartner Group dès 1996 (Natis, 2003). Le Hype Cycle[7] de la Figure I-1 montre l'historique de l'évolution de SOA à travers les 5 phases du Hype : le déclic technologique, le pic des espérances exagérées, le creux de la désillusion, la pente de l'éclaircissement et le plateau de productivité (Banerjee, 2010; Erl et al., 2011).

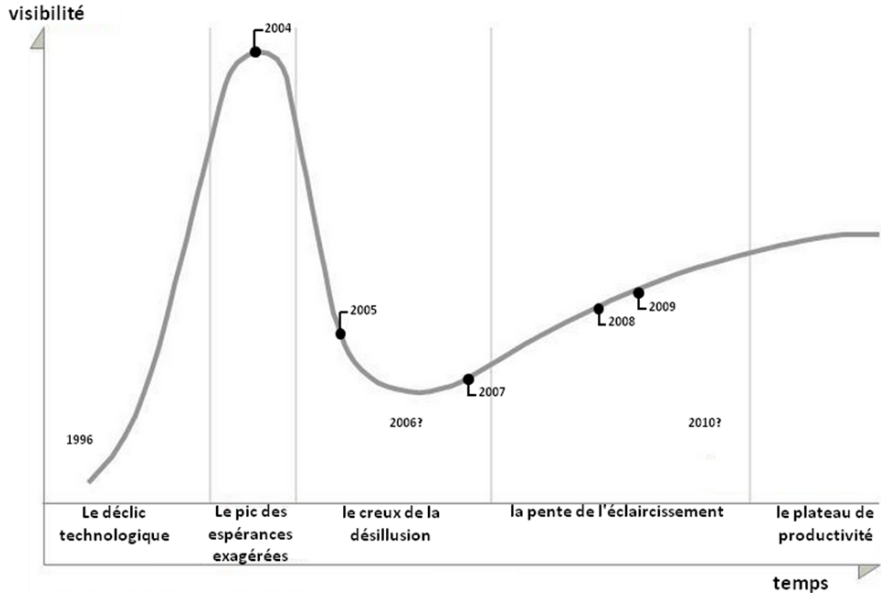


Figure I-1. Évolution historique de l'architecture orientée service (SOA) sur le Hype Cycle du Gartner[8].

Actuellement SOA passe de la pente de l'éclaircissement vers le plateau de la productivité. Elle est aujourd'hui présentée comme l'approche la plus efficiente pour le développement d'applications d'entreprise (Marks et al., 2006). De nombreuses définitions ont été proposées pour SOA. On estime comme (Gulledge et al., 2009) que SOA signifie différentes choses selon le public visé. Zimmermann (2009) propose trois perspectives pour définir SOA : (1) la perspective métier, (2) la perspective conception d'architecture et (3) la perspective de développement. Dans nos travaux, on se focalise uniquement sur les perspectives : métier et architecture. D'abord, (a) du point de vue métier une architecture orientée service doit fournir un ensemble de services qu'une unité d'entreprise expose à ses clients, partenaires ou à d'autres unités. L'alignement métier-IT est le principal objectif de cette perspective. Tandis que, (b) du point de vue architecture de

[7] Un diagramme mis en place par le cabinet d'analystes américain Gartner au milieu des années 90. Il positionne les nouvelles technologies de l'information et de la communication en fonction de leurs niveaux de maturité, de leur visibilité dans les médias et de leurs délais d'adoption sur le marché.

[8] Curieusement, le Gartner n'a pas fait apparaître SOA sur son Hype-cycle en 2006 et 2010

conception SOA est un ensemble de décisions qui respectent les principes du paradigme orienté services afin de permettre une activité de conception reproductible. SOA est un style d'architecture pour le développement et l'intégration d'applications d'entreprises.

La motivation fondamentale pour l'utilisation de SOA vient du constat suivant : le cloisonnement en silos applicatifs indépendants (blocs monolithiques) est une des sources majeures des difficultés rencontrées pour le traitement des évolutions et de la maintenance des systèmes. SOA respecte le paradigme orienté services pour positionner le service découvert au niveau de l'architecture logique comme le principal moyen pour construire des solutions logiques composées de services. Ces solutions forment une nouvelle génération de plateformes informatiques distribuées appelées (*Service-Oriented Computing - SOC*). C'est pourquoi l'objectif principal d'une entreprise s'engageant dans une démarche SOA est la rationalisation du métier (Bonnet, 2005). Enfin, Marks, Bell (2006) précisent que SOA est d'abord une démarche architecturale et organisationnelle : le choix des technologies et des outils reste secondaire. C'est un style d'architecture qui permet à l'entreprise de mieux utiliser le levier stratégique que constitue le système d'information devenant alors un véritable partenaire des directions fonctionnelles.

À travers ce chapitre, on introduit SOA, ses domaines d'application et son niveau de déploiement dans une organisation. Ensuite, on propose une typologie de services et on explique ce qu'est un service au sens SOA. Quelle est sa granularité ? Et à quel niveau est-il découvert ? Puis, on s'intéresse aux principes qu'un service doit respecter. Enfin, on présente quelques-uns des patrons de conception qui permettent la mise en place de ce style d'architecture pour atteindre les objectifs du SOC (Papazoglou et al., 2007).

2. DOMAINES D'APPLICATION DE SOA

Pour comprendre l'impact de SOA dans l'entreprise, il est nécessaire de clarifier le terme *architecture*. Ainsi, le standard (ISO/IEC/IEEE-42010, 2011) décrit l'architecture **(Définition 1 - Architecture)**[9] comme un ensemble de concepts fondamentaux ou de propriétés d'un système dans son environnement exprimé dans les relations de ses éléments et dans les principes de sa conception et de son évolution. L'architecture doit préciser le système auquel elle s'applique. Ainsi, dans une architecture orientée service, ce système est le système d'information selon la vision de (Reix, 2004). L'application de SOA à l'échelle de l'entreprise impacte l'entreprise en permettant l'émergence d'une nouvelle architecture qui favorise le rapprochement entre les deux logiques : métier et applicative.

[9] Fundamental concepts or properties of a system in its environment embodied in its elements, relationships, and in the principles of its design and evolution

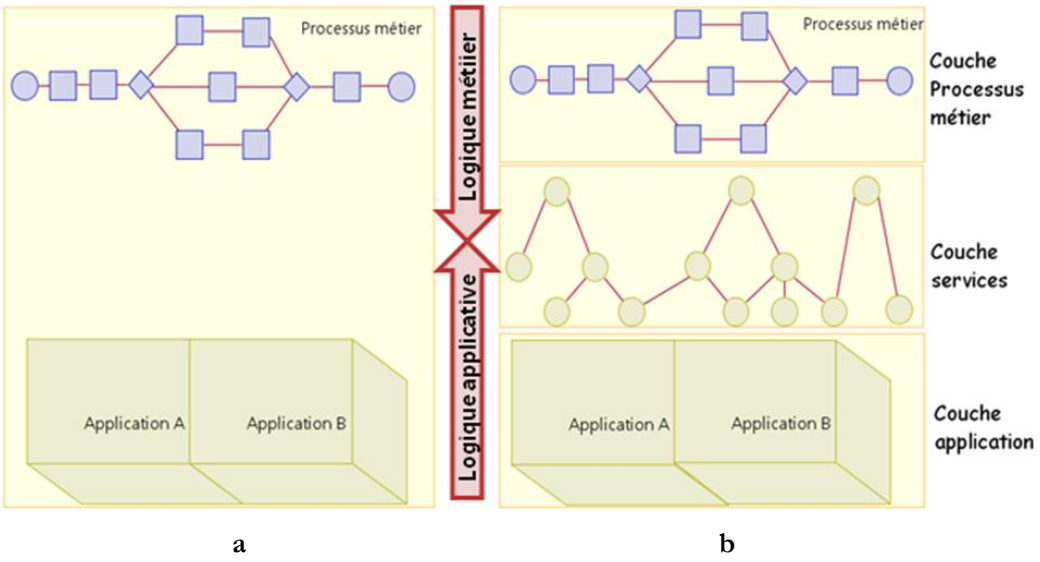


Figure I-2. Vues logiques du SI (Système d'Information) inspiré de (Erl01, 2010)

La Figure I-2 représente la vue logique traditionnelle du SI (a) et la vue logique d'un SI orientée services (b). La vu logique du SI est divisée en deux logiques : métier et applicatif. Chacune de ces deux logiques représente une partie de la structure de l'organisation. Dans la logique métier du SI orientée services, les exigences sont souvent exprimées avec des processus. Alors que la logique applicative est une automatisation de l'implémentation de la logique métier avec l'aide des technologies en utilisant les capacités du système opérant (infrastructure, contraintes sécurités, etc.).

On a constaté que SOA couvre deux domaines : l'architecture d'entreprise et l'architecture d'application.

2.1. L'architecture d'entreprise

(Définition 2 - l'Architecture d'Entreprise). L'Architecture d'Entreprise (AE) est définie comme l'ensemble des primitives et des artefacts descriptifs qui constituent une base de connaissances de l'entreprise (Zachman, 2000). Les artefacts doivent être présentés graphiquement. Quant aux primitives, elles représentent des conditions préalables d'ingénierie. Citons à titre d'exemple l'interopérabilité, la réutilisabilité, l'agilité, la réduction du délai de commercialisation, etc. On estime que l'architecture d'entreprise s'intéresse d'abord à la relation entre le métier et l'IT dans l'entreprise. Elle fournit une vision à long terme des processus, des systèmes et des technologies de l'entreprise afin que des projets individuels puissent construire des capacités et non pas simplement répondre à des besoins immédiats (Ross et al., 2006). D'ailleurs, une enquête réalisée par (Infosys, 2009) montre que l'objectif clé de l'AE est l'alignement métier-IT. Les

entreprises ont recours à l'AE pour conduire la mise en œuvre du changement dans l'organisation et pour supporter la prise de décision afin d'obtenir une certaine visibilité sur le métier et l'IT.

Une question essentielle est de savoir si l'architecture applicative doit répondre uniquement aux spécifications de l'implémentation ou si elle doit prendre aussi en compte les primitives définies dans le cadre de l'AE. Comme le souligne Zachman (2000) il est peu probable qu'un composant conçu dans un contexte donné (i.e., une application ou processus, etc.) soit réutilisable dans d'autres contextes (i.e., d'autres applications ou processus, etc.) par exemple, les données ou les règles de gestion associées au contexte d'un processus ne sont pas forcément réutilisables par d'autres processus. Par conséquent, il faut aborder les problématiques d'interopérabilité et de la réutilisation au niveau de l'AE et non seulement lors de la mise en place d'une application (i.e., au niveau de l'architecture de l'application). Les primitives doivent être définies au niveau de l'architecture d'entreprise et enrichies au niveau de l'architecture applicative pour développer l'application. Par exemple, les données (ou modèle de données) utilisées dans le cadre d'une application doivent être conformes aux modèles de données définis au niveau de l'entreprise. La difficulté consiste à savoir distinguer entre les modèles d'architecture d'entreprise et ceux de l'architecture applicative (Zachman, 2000).

2.2. L'architecture d'application

L'architecture d'application ou applicative intervient dans le processus de développement et d'intégration des applications. Elle permet de construire une application conformément à un nombre important de besoins différents. Chaque architecture d'application doit refléter les besoins immédiats de la solution, ainsi que les objectifs de la stratégie IT à long terme. Pour cette raison lorsqu'une organisation possède plusieurs architectures d'applications, ces dernières doivent être alignées avec une architecture d'entreprise.

2.2.1. L'intégration d'application

Une organisation peut avoir plusieurs architectures d'applications. Si elle est en présence de plusieurs environnements techniques (ex : Java Enterprise Edition – JEE et .NET) ou si elle contient des applications qui n'ont pas été conçues pour communiquer au-delà de leurs propres frontières. Dans ce cas, les approches d'intégration traditionnelles impliquent la connexion entre ces applications comme le montre la Figure I-3.

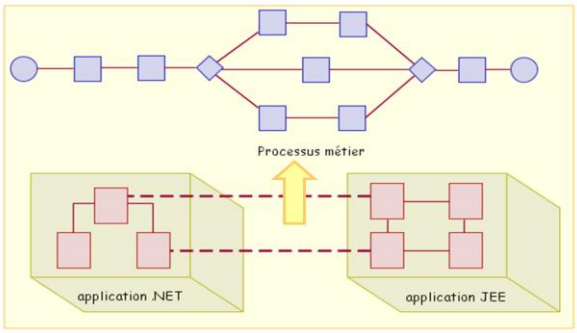


Figure I-3. Intégration traditionnelle entre deux applications pour automatiser un processus métier (vue logique de l'IT) inspirée de (Erl01, 2010)

En revanche, à la différence des approches traditionnelles, une orientation service introduit une couche de services dans la vue logique de l'IT (Figure I-4). Les services forment une couche intermédiaire entre les processus métiers et les applications qui supportent leur exécution. Les services encapsulent certaines capacités des applications ou du processus métiers. Dans ce style d'architecture l'intégration peut se faire au niveau des activités des processus et non pas au niveau des applications métiers. De plus, cette intégration se fait via des interfaces services bien définies et accessibles par plusieurs utilisateurs.

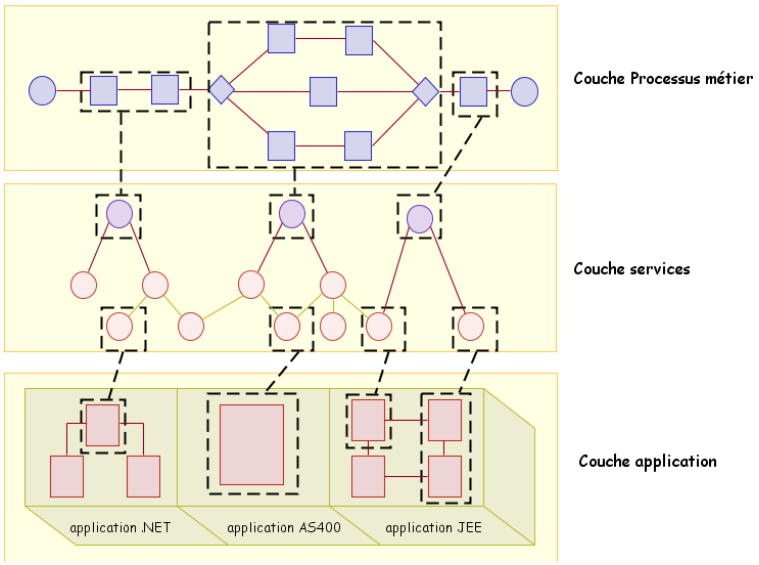


Figure I-4. Intégration dans une architecture orientée service (vue logique de l'IT) (Erl01, 2010)

2.2.2. Le développement d'application

Une architecture orientée services permet de construire des applications métiers orientées service ou (*Service Oriented Business Application* -SOBA) (Bloomberg, 2009) basées sur la composition de services à partir d'un inventaire de services (Figure I-5). L'application devient une composition de services pouvant être utilisés dans d'autres applications : une partie de la logique applicative

n'est pas exclusive à l'application. L'utilisation d'un inventaire de services rend la perspective de réutilisation globale au niveau de l'entreprise prédominante.

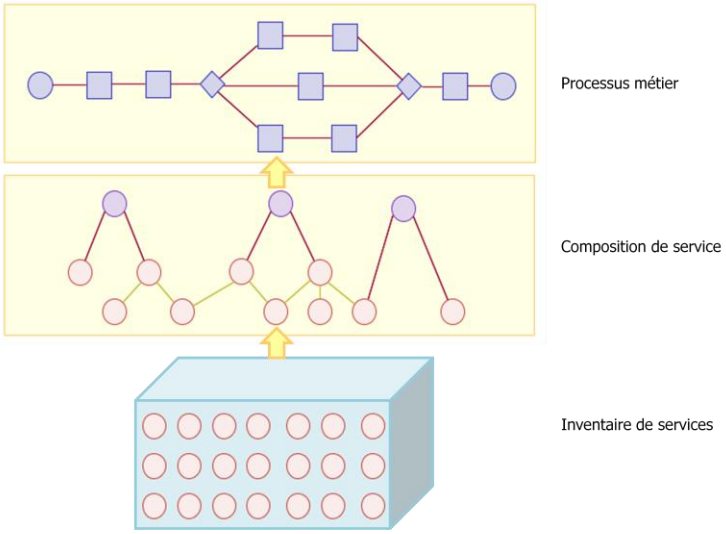


Figure I-5. Processus métiers basé sur les services de l'inventaire de services

3. NOTION DE SERVICE

3.1. Le service au sens SOA

Dans SOA le service est l'unité élémentaire de construction du système (PraxemeInstitute01, 2006). Il est le composant de plus bas niveau de l'architecture logique. C'est l'atome, le grain élémentaire, dans la construction logique du système. Dans une architecture orientée service pour obtenir une information ou pour accéder à la donnée, il faut passer par un service. Cette caractéristique change radicalement la façon de concevoir le logiciel (PraxemeInstitute02, 2007). Dans la méthode Praxeme (2007), le service SOA est appelé *service logique* pour bien signifier que l'effort de conception et de structuration se joue sur l'aspect logique.

Une typologie de services permet de structurer la démarche de conception dans une architecture orientée service. On propose de distinguer trois types de services : service métier, service entité et service utilitaire (Figure I-6).

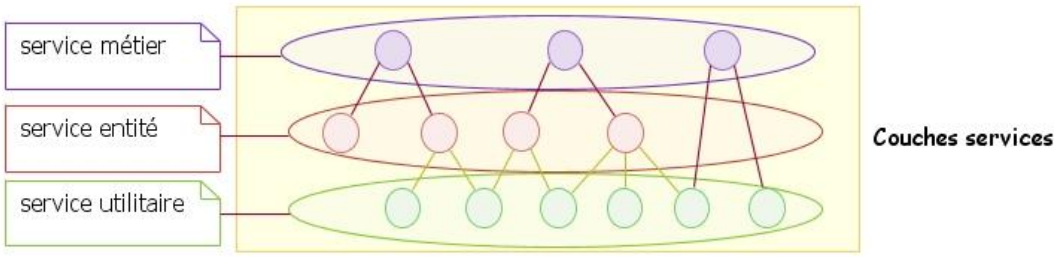


Figure I-6. Classification des services SOA

Il faut distinguer le concept de service métier spécifié au niveau du cahier des charges (fonctionnel ou métier) et le concept de service (entité ou utilitaire) issu d'un travail d'architecture applicative qui prépare l'implémentation des services métiers. Les services SOA les plus réutilisables (service entités et service utilitaires) sont découverts au niveau de l'architecture applicative.

3.1.1. Le service métier

Marlon et Thomas considèrent que le terme service métier représente le résultat d'une opération dans une organisation. Du fait que les opérations d'une organisation peuvent être à différents niveaux de granularité, le service métier peut aussi être représenté à différents niveaux (Marlon et al., 2010). D'autres auteurs considèrent qu'un service métier peut être compris comme l'agrégation de services techniques (Gulledge, 2010), ou découvert directement au niveau de la modélisation des processus (Bonnet, 2005).

Le traitement offert par un service métier est un service de type particulier. En effet, le rôle du service métier est d'offrir un ensemble cohérent de traitements métiers. Il correspond à un périmètre fonctionnel exposé à des consommateurs indépendamment des choix d'architecture applicative (Bonnet, 2005). On estime que le service métier doit être défini du point de vue de l'utilisateur final, qui perçoit le résultat de SOA à travers le service métier qu'il consomme. Pour l'utilisateur, un service métier est une opération offrant une valeur ajoutée pour l'activité, favorisant son efficacité et améliorant sa performance. La Figure I-7 représente un exemple d'un service métier *CommandeBusiness*. Après le choix des articles, des adresses de facturation (et/ou de livraison) et du mode de payement le client soumet sa commande ce qui déclenche un processus métier.

Figure I-7. Un exemple de la capacité d'un service métier

Enfin, il faut distinguer le service métier qui est le résultat d'une orchestration de plusieurs services et qui représente par exemple un processus métier du service métier qui est associé à une seule activité. Le premier a une forte dépendance au contexte du processus métier qu'il représente, alors que le second peut être utilisé indépendamment du processus métier.

3.1.2. Le service entité

Les services entités (de domaines) sont découverts à partir des objets sémantiques principaux qui expriment les fondamentaux du métier. Ces services permettent de dégager un noyau stable

composé des services hautement réutilisables. Ils représentent les domaines d'objets (objet métier) construits autour des objets centrals de l'organisation. Par exemple, pour une entreprise de VAD : facture, commande, adresse et article représentent des objets de domaines (Figure I-8).

Figure I-8. Un exemple des capacités offertes par un service entité

Un point critique consiste à distinguer et à définir la relation entre les services métiers et les services entités. La distinction entre ces deux types de services est tributaire des capacités offertes par le service. La capacité *getSupport* du service *CommandeEntity* est associée à la notion de commande. Pour calculer le montant de la commande, il faut identifier son support de vente (ex : Internet, catalogue, etc.) et son mode d'expédition. Le périmètre fonctionnel de cette capacité est clairement associé au périmètre fonctionnel du service entité *CommandeEntity*. En revanche la capacité *soumettre* du service *CommandeBusiness* déclenche un processus qui fait intervenir plusieurs objets de domaines qui n'appartiennent pas à un périmètre fonctionnel clair : la soumission de la commande par le client déclenche des traitements de validation de la commande (ex : adresses de facturation et de livraison, produit en stock, etc.), du payement et de facturation. Pour réaliser la capacité du service *CommandeBusiness,* il faut orchestrer plusieurs services entités.

3.1.3. Le service utilitaire

Ces services peuvent être utilisés pour enrober les ressources existantes de l'entreprise issues de systèmes patrimoniaux (*legacy systems*). Contrairement aux deux autres types de services, les services utilitaires ne dépendent pas forcément du métier de l'organisation. Ils peuvent assurer des fonctionnalités utiles aux autres services, comme la journalisation, la sécurité, la gestion d'exception ou la transformation des données.

Figure I-9. Un exemple des capacités d'un service utilitaire

Ces services doivent être implémentés une seule fois ensuite ils doivent être considérés comme faisant partie de l'infrastructure SOA (Figure I-9). Par exemple, certains *Enterprise Service Bus* ESB comme (WSO2, 2012) proposent nativement des services (ex : services de monitoring. etc.).

3.2. Le service technique

Le service au sens SOA n'est pas une solution technique. Les services techniques sont utilisés pour implémenter les services logiques découverts au niveau de l'architecture logique. Il est possible de concevoir un système informatique à l'aide de l'approche SOA, tout en utilisant des technologies classiques (ex : JEE, .NET). D'autres technologies comme les web-services (Alonso et al., 2004) et (*Service Component Architecture* - SCA) (OSOA, 2007) sont plus associés à SOA.

Actuellement le web services est la technologie la plus associée à SOA. L'arrivée de cette technologie en 2001 a contribué à la diffusion de SOA et à son acceptation (Cheliah et al., 2012). On considère dans ce travail les web-services comme l'implémentation de référence des services découverts avec SOA. Si un web-service est utilisé pour implémenter un service logique découvert au niveau de l'architecture logique, alors un service logique deviendra l'élément *interface*[10] dans le (*Web Services Description Language - WSDL*) version 2.0 (W3C08, 2007). Dans cette thèse, on n'adopte pas la terminologie définie dans la spécification WSDL version 2.0 qui utilise le terme service pour désigner un ensemble d'endpoints. Le terme **(Définition 3 – Endpoint)** endpoint désigne une adresse physique sur laquelle un service peut être invoqué avec un protocole spécifique comme (*Simple Object Access Protocol - SOAP*) (W3C00, 2007).

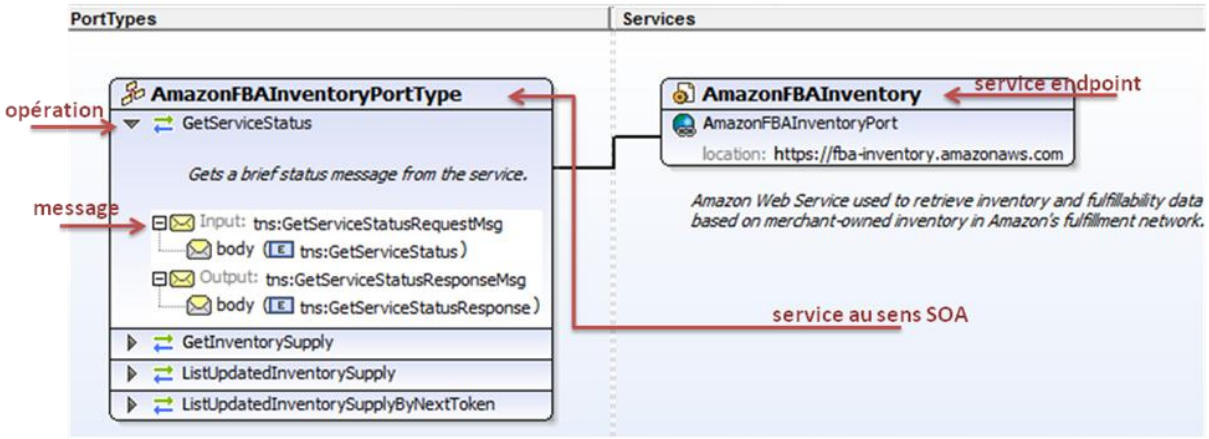


Figure I-10. Exemple du WSDL proposé par Amazon *Fulfillment* (Amazon, 2011)

[10] L'élément *portType* dans la version 1.1 du WSDL a été renommé *interface* dans le WSDL version 2.0

La Figure I-10 montre que le service Amazon *AmazonFBAInventoryPortType* contient quatre opérations : *GetServiceStatus*, *GetInventorySupply*, *ListUpdatedInventorySupply*, *ListUpdatedInventorySupplyByNextToken*.

4. PARADIGME DE CONCEPTION ORIENTÉE SERVICES

Le paradigme est une notion introduite par Thomas Kuhn. Le paradigme **(Définition 4 - Paradigme)** désigne une conception théorique dominante ayant cours à une certaine époque dans une communauté scientifique donnée, qui fonde les types d'explication envisageables, et les types de faits à découvrir dans une science donnée » (atilf et al., 2012). Plus précisément, un paradigme de conception se compose d'un ensemble de règles ou de principes complémentaires qui définissent collectivement l'approche globale représentée par ce dernier (Erl02, 2010). La conception orientée service représente un nouveau paradigme qui est le résultat d'un ensemble de caractéristiques basées sur l'application de plusieurs principes. L'application dans le monde réel de ces principes est assurée par des patrons de conception (i.e., *design patterns*) (Figure I-11). Comprendre SOA revient à comprendre les principes qu'un service au sens SOA doit respecter et les patrons pour implémenter ces derniers.

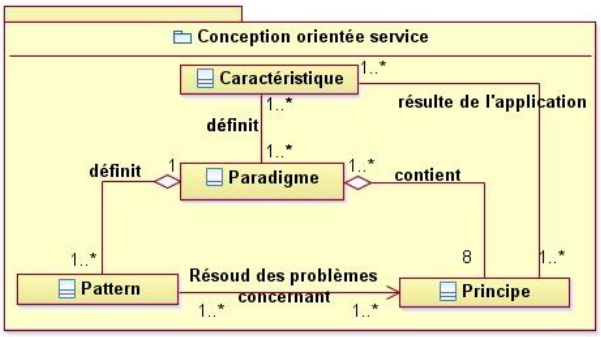


Figure I-11. La conception orientée services

4.1. Les principes de conception dans SOA

On a expliqué qu'il n'y a pas de définition unique pour SOA (cf. I.). Il en est de même pour les principes de SOA. Cependant, il y a des principes communs qu'une orientation services doit respecter.

4.1.1. Contrat de services

Le contrat de services décrit les modalités d'accès au service entre l'utilisateur et le fournisseur du service. Il peut être constitué de description technique ou non. Un contrat de services peut décrire (1) des besoins fonctionnels, (2) non fonctionnels ou (3) sémantiques. Le contrat de service peut être spécifié sans détails techniques avec un langage comme (*Service Oriented*

Architecture Modeling Language - SoaML) (OMG02, 2009) pour décrire les aspects fonctionnels du service et un langage comme (*Model-Driven Development for Service-Oriented Architecture - MDD4SOA*) (Mayer, 2010) ou UML (diagrammes de comportement) pour décrire les aspects non fonctionnels du service. Un document appelé Service Level Agreement[11] (SLA) peut-être aussi utilisé pour décrire la qualité des services.

Si le contrat de service est formalisé avec la technologie des web-services, alors le WSDL (W3C08, 2007) et le XML Schema (W3C09, 2004) peuvent être utilisés pour définir les exigences et les contraintes fonctionnelles, WS-Policy (W3C04, 2007) pour décrire les exigences et les contraintes non fonctionnelles et (*Semantic Annotations for WSDL and XML Schema - SAWSDL)* (W3C10, 2007) pour décrire la sémantique des services.

L'utilisation du contrat de service permet de standardiser la représentation des données utilisées par les services afin de réduit le recours aux transformations. Dans le chapitre V, on explique comment concevoir la structure du contrat de service avec SoaML.

4.1.2. Couplage faible entre les services

(Définition 5 - Couplage). Le couplage signifie que deux choses ont une relation de dépendance, le niveau de couplage dépend du niveau de dépendance. En génie logiciel ce terme désigne la manière et le degré d'interdépendance entre différents modules logiciels (IEEE01, 1990). Un système d'information basé sur SOA peut être faiblement couplé sur un aspect et fortement couplé sur un autre aspect (Schmelzer, 2007).

La direction du couplage dépend des choses couplées. Par exemple, deux applications intégrées point à point forment un couplage bidirectionnel. Chacune d'entre elles est dépendante de l'autre application pour fonctionner normalement. En revanche, une base de données ne dépend pas de l'application qui l'utilise, la relation de couplage entre une application et une base de données est de type couplage unidirectionnel.

On distingue deux catégories de couplage interne et externe (Schmelzer, 2007; Erl02, 2010) :

Le couplage interne concerne le contrat de service : dans ce type de couplage, il faut distinguer deux sous-types de couplage. D'une part, le couplage de la logique métier du service vers son contrat de service, d'autres services (composition de service), des technologies ou des ressources qu'il utilise (Tableau I-1). D'autre part, le couplage du contrat de service vers sa logique métier, le processus qui l'utilise ou des technologies (Tableau I-2). Les deux tableaux présentent pour chaque type de couplage une description succincte, des remarques et une recommandation concernant le type du couplage : faible ou fort.

[11] Contrat de niveau de service

Couplage	Description	Remarques	Type
Contrat de service	Construire le contrat de service avant la logique de la solution induit un couplage fort unidirectionnel de la logique de service vers le contrat de service	Ce couplage permet de mettre en place un contrat de service indépendant et standard. La logique du service est développée conformément au contrat de services « *Approches Contract First* » (Zhong et al., 2009).	fort
D'autres services	Un service composite est couplé à la logique des services qui le compose.	La composition de service composable sans réutilisation nécessite des transformations.	faible
Technologies	Plusieurs technologies d'implémentation existent : java, .Net, C++, etc.	Éviter l'utilisation des technologies propriétaires non standardisés.	faible
Ressources	Les services déployés ont besoin d'accéder à des ressources externes à la logique des services : bases des données ou des infrastructures middleware, etc.	L'implémentation des services doit être neutre et indépendante de l'infrastructure. Il faut limiter l'accès à des ressources externes à la logique des services.	faible

Tableau I-1. Les types de couplage internes à la logique du service

Couplage	Description	Remarques	Type
Logique du service	Ce couplage concerne la génération du contrat de service à partir de solutions existantes. La conception du contrat de service se trouve dépende de la logique de la solution.	Ce couplage augmente la dépendance du contrat de service vers la technologie de l'implémentation des services	faible

Technologies	Le développement du contrat de services avec des technologies propriétaires limite son utilisation, car il peut conduire à l'ajout de caractéristiques technologiques spécifiques dans ce dernier.	L'utilisation des technologies XML ou web-services permet au fournisseur de services de changer l'implémentation des services sans affecter l'utilisateur. D'ailleurs, le succès des web-services est en partie dû à la possibilité de construire un contrat de service sans détails techniques liés à la solution.	faible
Processus	Si le service est conçu spécifiquement pour un processus donné son contrat et sa logique dépendront fortement de la logique et du contexte du processus.	Les services métiers sont intentionnellement couplés aux processus métiers. Pour réduire ce couplage, il faut appliquer les principes de réutilisation, notamment avec l'utilisation des services d'entité dont le contexte est indépendant du contexte des processus métiers.	faible

Tableau I-2. Les types de couplage internes du contrat de service

Le couplage externe concerne le consommateur du service : on considère que l'utilisateur accède aux services uniquement via leur contrat de service. Donc, on s'intéresse au couplage entre l'utilisateur et le contrat de services. Ce couplage utilisateur-contrat de service peut conduire (1) soit à un couplage direct, si le contrat de service dépend d'une technologie quelconque que le client doit utiliser (l'utilisateur est conscient de ce couplage). (2) Soit à un couplage indirect, si le service décrit dans le contrat de service est fortement lié à la logique du processus métier du fournisseur ou à ses ressources (l'utilisateur est inconscient de ce couplage).

En conclusion, l'utilisateur du service peut hériter les caractéristiques du couplage du contrat de service (couplages internes). C'est pour cette raison qu'il est fortement recommandé de concevoir un contrat de service de qualité en se basant sur le principe d'abstraction de services (voir le paragraphe suivant). Si le contrat de service est bien conçu son couplage fort avec le consommateur assure l'indépendance de ce dernier par rapport au fournisseur de services. De plus, ayant expliqué que la logique du service influence la conception du contrat de service, il faut porter une attention particulière à la conception de services. Autant que possible, les services métiers

doivent être couplés uniquement à la logique des processus qui les utilise, alors que la conception de services entités réutilisables doit dépendre de la logique métier. Enfin, pour les services utilitaires, il faut minimiser le couplage aux technologies utilisées dans les systèmes existants.

4.1.3. Abstraction de services

En génie logiciel **(Définition 6 - Abstraction)** l'abstraction est une vue qui se concentre sur les informations pertinentes d'un objet dans un but particulier (IEEE01, 1990). L'abstraction d'un service détermine combien d'informations sont disponibles sur ce dernier. Cette abstraction vise à cacher le détail des informations inutiles du contrat de service. D'abord, pour permettre au propriétaire de service de faire évoluer le service facilement. Puis, pour éviter d'influencer l'implémentation du contrat de service par son utilisateur. Ainsi, plus il y a de détails dans le contrat de service plus le couplage entre l'utilisateur et le contrat de service est fort. L'abstraction concerne quatre types d'information (Erl02, 2010) :

1. **Technologique :** concerne le contrat de service et son implémentation. Il faut éviter d'avoir une description détaillée de l'implémentation technique. Par exemple, les web-services fournissent des informations concernant le Binding (ex : le protocole de communication SOAP) sans aucune information sur les technologies utilisées pour implémenter le service (i.e. langage de programmation, base de données, etc.).
2. **Fonctionnelle :** cette abstraction est limitée uniquement au contrat de service qui publie certaines capacités du service. Les capacités publiées dans le contrat de services délimitent le périmètre fonctionnel du service connu par le client.
3. **Programmatique :** concerne les détails de conception liés à la logique du programme utilisé pour implémenter les services (journalisation, exceptions, authentification, etc.).
4. **Qualité de service :** concerne la logique et le contrat du service. Ce terme englobe des méta-informations qui permettent de déterminer le comportement du service à l'aide d'un ensemble de règles et de contraintes. La qualité de service peut être décrite techniquement avec WS-Policy ou exprimée dans un document SLA.

L'abstraction de service peut être vue comme un principe régulateur entre deux groupes de principes SOA : le premier groupe comprend les principes d'autonomie et de couplage faible qui préconisent de mettre le moins possible d'informations dans le contrat de service. Le second groupe regroupe les principes de réutilisabilité, de composition et de découverte qui encouragent à mettre plus d'informations sur les services.

4.1.4. Réutilisabilité de services

Tous les autres principes de SOA contribuent à la réalisation du caractère de réutilisabilité du service qui représente le but principal d'une architecture orientée service. Ce principe intervient très en amont (dès la phase d'analyse) lorsqu'il faut choisir les services candidats à retenir pour la phase de conception.

Pour créer un service réutilisable, il faut une bonne compréhension de ses futurs utilisateurs potentiels. Cette compréhension vient d'une bonne connaissance du métier et de l'environnement technologique de l'entreprise. Ceci renforce l'idée que le périmètre de SOA s'étend sur le métier et l'IT. En fait, bien que l'émergence des plateformes comme les web-services ait largement contribué à l'effort de réutilisation, la technologie à elle seule ne permet pas de surmonter tous les obstacles qui se dressent devant l'effort de réutilisation. On estime que la réutilisation est liée à l'organisation, c'est un problème organisationnel et culturel qui dépasse même le cadre de l'architecture SOA. En fait, le risque de la non-utilisation d'un service peut être dû à la non-connaissance de son existence ou au refus de l'utiliser s'il est jugé non adapté ou contraignant. Pour la première possibilité, le principe de découverte de service contribue à identifier les services recherchés. Alors que l'application d'une méthode de conception de services peut contribuer à résoudre le deuxième problème.

Le but premier de la création des services réutilisables est d'augmenter l'agilité métier dans l'organisation pour supporter l'automatisation des processus en se basant sur la composition. Un service réutilisable doit avoir (1) une logique de service générique pour pouvoir prendre en compte plusieurs scénarios utilisateurs et (2) un contrat de service extensible et générique capable de traiter des messages variés. Ces deux caractéristiques rendent le service réutilisable et permettant son utilisation dans l'automatisation de plusieurs processus. Cependant, il faut préciser que les différents types de services ont des capacités de réutilisation différentes. Ainsi, contrairement aux services métiers qui dépendent des processus métiers et qui ont une capacité de réutilisation limitée, les services entités qui encapsulent la logique métier et les services utilitaires sont plus réutilisables.

Enfin, la réutilisation pose un vrai défi, car en insistant sur l'incorporation des développements existants des autres équipes, elle augmente la complexité et l'effort pour construire le logiciel.

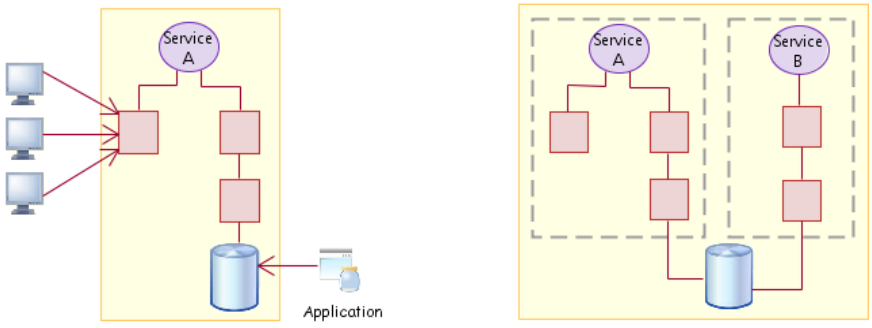
4.1.5. Autonomie de services

(Définition 7 - Autonomie) L'autonomie en ingénierie logicielle représente l'indépendance avec laquelle un logiciel accomplit sa logique (Erl02, 2010). De même, exercer un certain contrôle

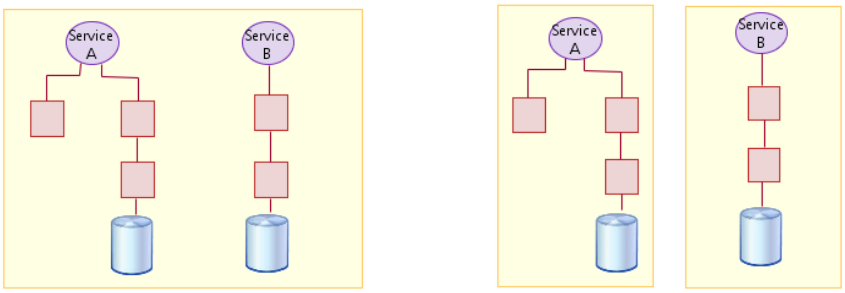
sur ses ressources rend le service autonome. Ce principe permet de trouver un équilibre avec le principe de composition de services. On distingue essentiellement deux types d'autonomies : (1) l'autonomie de conception qui a pour but de définir un périmètre fonctionnel distinct pour chaque service. L'application du principe de couplage faible garantit ce type d'autonomie. (2) L'autonomie d'exécution qui vise à accroitre le contrôle sur l'environnement d'exécution des services. L'environnement doit permettre de déplacer, d'isoler ou de composer les services tout en assurant un bon degré de performance.

Pour ce qui est de l'implémentation des services, Erl03 (2010) distingue quatre niveaux d'autonomie. Cette distinction est basée sur le partage de la logique, des ressources et de l'environnement d'exécution entre les services :

1. **L'autonomie partagée** dans laquelle le service partage sa logique et ses ressources avec d'autres services. Par exemple, le service A dans la Figure I-12-a partage sa logique avec des canaux d'utilisateur et sa ressource (base de données) avec une application existante. Le service ne respecte ni l'autonomie de conception ni l'autonomie d'exécution.
2. **L'autonomie de la logique** des services qui est une autonomie partielle puisque les services partagent les ressources de l'entreprise comme les bases de données ou les répertoires (Figure I-12-b). Ce niveau d'autonomie respecte l'autonomie de conception.
3. **L'autonomie par isolation fonctionnelle** dans laquelle le service respecte l'autonomie de conception mais pas celle de l'exécution. Elle est le résultat d'une méthode de conception et d'architecture descendante qui a comme résultat des services qui contrôlent leur propre logique et ressources. Dans ce type d'autonomie les services sont exécutés sur le même serveur (Figure I-12-c).
4. Dans **l'autonomie absolue,** le contrôle est total sur les services. En plus, des propriétés de l'autonomie fonctionnelle les services sont isolés dans des environnements différents (Figure I-12-d).



a **b**



c **d**

Figure I-12. Les niveaux d'autonomie des services selon (Erl02, 2010)

a. L'autonomie partagée | b. L'autonomie de la logique des services | c. L'autonomie fonctionnelle | d. L'autonomie absolue

Par nature un service composite ne garantit pas le principe d'autonomie, car il dépend de la logique de ses services composants. Le service composite est au plus aussi autonome que les services qui le composent. Le niveau d'autonomie d'un service métier est corrélé négativement avec sa hiérarchie dans la composition des services. De plus, la qualité d'autonomie dépend aussi de la qualité du processus métier auquel le service est associé. À l'inverse, le service entité a un niveau d'autonomie plus élevé et bénéficie du fait que sa construction est intentionnellement indifférente au contexte fonctionnel. Ceci permet aux services entités de jouer un rôle important dans l'architecture orientée services. Enfin, les services utilitaires sont généralement moins autonomes que les services entités parce qu'ils dépendent en partie des capacités du système qu'ils encapsulent.

4.1.6. Services sans état

Le terme sans état (*stateless*) est utilisé pour qualifier une condition d'exécution du service. Un service sans état ne traite pas l'état des données associé à une activité. Autrement dit, il ne garde pas la mémoire de son exécution précédente. Aucun état n'est préservé entre deux invocations du service. Par conséquent, toutes les invocations du même service sont indépendantes. En revanche, si le service traite et conserve l'état des données alors c'est un service avec état (*stateful*). En effet, l'implémentation de ce principe concerne plutôt les capacités du service et non le service lui-même. Erl (2010) distingue trois types d'état de données, chacune d'elles est associée à l'exécution du service :

1. **Données de la session :** représentent des informations pour garder une connexion entre le programme et ses clients. Par exemple, ces données peuvent être utilisées lors d'un appel asynchrone à un web-service avec le mécanisme de corrélation (Decker et al., 2008).

2. **Données du contexte :** ce sont les données spécifiques à l'exécution des activités (ex : les données échangées entre elles). Souvent, ces données sont étroitement liées aux règles associées au (*Moteur d'Exécution de Processus - MEP*).
3. **Données métiers :** ils représentent les données pertinentes pour l'activité métier, comme les données persistantes.

Un service métier d'orchestration est par nature *stateful*, en plus des données de session, il gère principalement l'état des données du contexte durant l'exécution du processus métier. Par contre, si le service métier représente une activité, alors il maintient d'abord les données de la session durant l'exécution de l'activité en plus des données du contexte. En revanche, le service entité doit être *stateless*, sauf s'il représente une composition complexe qui nécessite de gérer l'état des données. Enfin, un service utilité peut être intentionnellement *stateful* s'il a la responsabilité de gérer l'état des données pour le compte des autres services.

4.1.7. Découverte de services

La découverte est l'action de découvrir ce qui était ignoré, inconnu ou caché (Rey-Debove et al., 1993). Historiquement, le principe de découverte de service était la principale distinction entre SOA et les autres types d'architecture distribués (Figure I-13). La représentation originale de l'architecture orientée service était caractérisée par l'existence de trois parties : (1) le fournisseur de services qui publie les services et leur description dans le registre de services, (2) l'utilisateur du service qui découvre et utilise les services publiés dans le registre de service et (3) le composant de découverte (registre de services) qui fournit les interfaces de publication et de découverte des services.

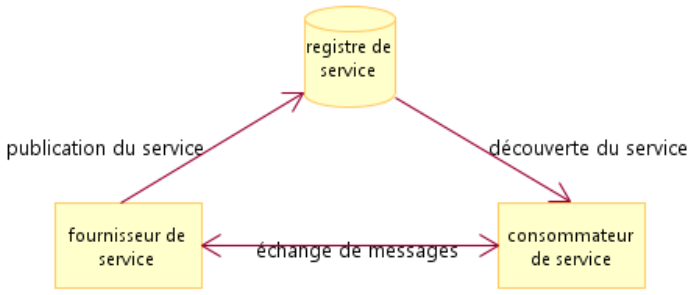


Figure I-13. La représentation originale de l'architecture orientée service SOA

Ce principe permet de répondre à la question suivante : « *Est-ce que le service dont on a besoin existe déjà ou faut-il créer un nouveau service ?* » dans le but d'éviter la redondance et de favoriser la réutilisation des services disponibles. Il est le seul à n'avoir aucun effet direct sur la granularité des services.

La découverte de services se focalise essentiellement sur l'ajout de méta-données au contrat de services. Elle demande une bonne expertise métier et technique pour avoir une bonne compréhension de la nature du service nécessaire à la réalisation d'un service découvrable. D'une part, le contrat de service doit contenir des méta-données qui communiquent clairement les objectifs et les capacités des services. D'autre part, l'implémentation du contrat de services nécessite l'utilisation des standards afin de rendre le contrat de service découvrable. En effet, pour que le service soit découvrable, il doit contenir des méta-données qui lui permettront d'être identifié. L'application de ce principe implique des décisions concernant : (1) la qualité de définition des méta-informations, (2) la manière dont les méta-informations sont définies et (3) la façon dont la découverte de services est implémentée.

En cas d'utilisation des web-services, des méta-informations peuvent être ajoutées informellement aux services sous forme de commentaires dans le WSDL ou dans le schéma XML. Elles peuvent aussi être rajoutées d'une manière standardisée avec WS-Policy (W3C04, 2007). La découverte des services peut être automatisée en utilisant (*Universal Description Discovery and Integration* – UDDI) (OASIS01, 2002) pour obtenir des informations sur la description des services stockés dans le registre de services. En plus, WS-MetadataExchange (Ballinge et al., 2006) peut être utilisé pour récupérer des méta-données des web-services. Par exemple, le client peut utiliser UDDI pour trouver les adresses endpoints des services candidats disponibles dans un registre de service. Ensuite, WS-MetadataExchange peut être utilisé pour récupérer les méta-données de chaque service candidat. Ceci permet au client d'avoir des informations précises sur le fournisseur de services et des détails (WSDL, XML schema) pour s'interagir avec le service choisi (Erl, 2009).

4.1.8. Composition de services

(Définition 8 - Composition) La composition est l'action ou la manière de former un tout en assemblant plusieurs parties ou plusieurs éléments (Rey-Debove et al., 1993). En ingénierie logicielle, le concept de composant logiciel a été introduit dans la première conférence d'ingénierie logicielle par McIlroy (1968). Plusieurs définitions existent, on retient la définition de Szyperski (2002)**(Définition 9 – Composant logiciel) :**[12] un composant logiciel est une unité de composition avec des interfaces contractuellement spécifiées et des dépendances uniquement explicites sur son contexte. Un composant logiciel peut être déployé indépendamment et est sujet à composition par des tierces entités. L'objectif et la forme des composants logiciels ont changé en

[12] A software component is a unit of composition with contractually specified interfaces and explicit context dependencies only. A software component can be deployed independently and is subject to composition by third parties.

fonction du paradigme[13] dans lequel ils ont été utilisés : depuis des éléments de code source comme les routines, les procédures, les modules ou les objets vers des unités architecturales enfichables (*pluggable*) dynamiquement sur des systèmes en exécution (Crnkovic et al., 2011). Historiquement et conceptuellement les systèmes à base de composants (*Component-Based Systems - CBS*) sont considérés comme le prédécesseur de l'informatique à base de services (*Service-Oriented Computing - SOC*). Néanmoins, il y a des différences entre la notion de composant dans le CBS et la notion de service dans le SOC. Pour (Elfatatry, 2007; Papazoglou et al., 2011) CBS et SOC sont différents en termes de philosophie, de Binding, de couplage, de granularité, de mécanisme de communication et d'architecture.

La composition de service est le résultat d'une logique de service sophistiquée, elle émerge de l'existence d'un inventaire de services matures et bien établis qui favorise l'émergence de services réutilisables dans plusieurs compositions. La composition est comme une forme de réutilisation complexe. D'une part, le service peut être réutilisable, mais non composable, s'il est utilisé par plusieurs utilisateurs, mais participe uniquement à des activités de type point à point qui font intervenir uniquement le service et son déclencheur. D'autre part, il peut aussi être composable mais non réutilisable, s'il est délivré pour une composition fortement couplée à un consommateur. En effet, la composition peut être fortement liée à un processus métier ou à une activité.

Chaque service joue le rôle soit d'un composable soit d'un composé soit des deux en même temps. Ainsi, il est considéré comme un composant lorsque sa logique applicative invoque les capacités des autres services et comme un composable lorsqu'il est appelé par d'autres services. En réalité, un service peut être temporairement composable ou composé, car cela dépend de sa capacité. C'est la capacité du service qui le rend composé quand elle fait appel aux capacités des autres services et composables quand c'est la capacité qui est appelée par d'autres services. Le déclencheur de la composition est celui qui initialise la composition sans pour autant en faire partie. En invoquant une composition le consommateur initialise la composition qui doit faire intervenir au moins deux services. En fait, la simple interaction entre le service et son consommateur n'est pas considérée comme une composition.

Enfin, la composition concerne d'abord les services métiers, puis les services entités qui peuvent contenir des capacités hautement réutilisables capables de répondre à une exigence métier. Par contre, il est rare d'utiliser la composition pour les services utilitaires qui sont d'abord destinés à encapsuler des capacités existantes de l'entreprise.

[13] Exemple : orienté objet, à base de composants ou orientée services, etc.

4.2. Les patterns de conception dans SOA

Erl (2009) propose 85 patterns de conception pour SOA. Chacun de ces patterns capte de bonnes pratiques apportées par le style d'architecture orienté services. On introduit ici uniquement les patterns Enterprise Service Bus – ESB, inventaire de services et les patterns qui les composent.

4.2.1. L'Enterprise Service Bus (ESB)

Le concept de Bus de Service (*Enterprise Service Bus* - ESB) a été introduit par Schulte (2002) qui le définit comme **(Définition 10 – Enterprise Service Bus) :** « *une nouvelle architecture qui exploite les web-services, le routage intelligent, la transformation et un middleware de messagerie* ». Pour Keen et al. (2004) l'ESB doit aussi supporter plusieurs styles d'intégration : orientée service, orientée message et orientée événement. Dans le contexte d'une architecture orientée service, une définition restrictive considère l'ESB comme un garant du principe de couplage faible à travers la virtualisation des services : la transparence de la localité, du format de données et du protocole des services (Wylie et al., 2009) (Figure I-14).

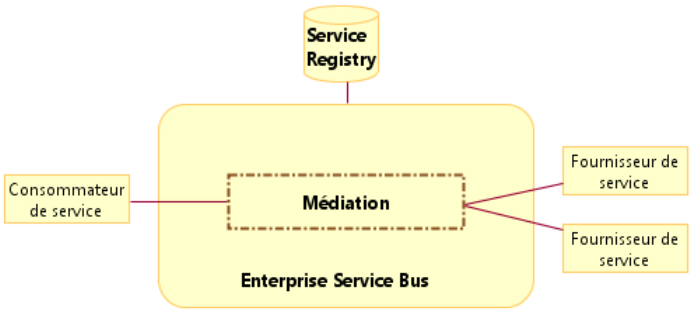


Figure I-14. La virtualisation de service avec un ESB

L'ESB est un pattern de conception orientée service

L'ESB est aussi un pattern d'intégration qui hérite les propriétés du pattern *Bus de message* et qui est composé du pattern *Broker message* (Figure I-15). En fait, plusieurs propositions des patterns composant l'ESB existent (Erl, 2009; Wylie et al., 2009). On retient ici la proposition de Erl (2009) qui considère que l'ESB dans le contexte d'une architecture orientée service doit être au minimum constitué des patterns : *Routage*, *Message Broker* et *File d'attente asynchrone*. La Figure I-15 définit le groupe de patterns qui composent le pattern l'ESB et met en évidence la relation entre le pattern ESB et le pattern Broker Message communément appelé à tort Hub-and-spoke.

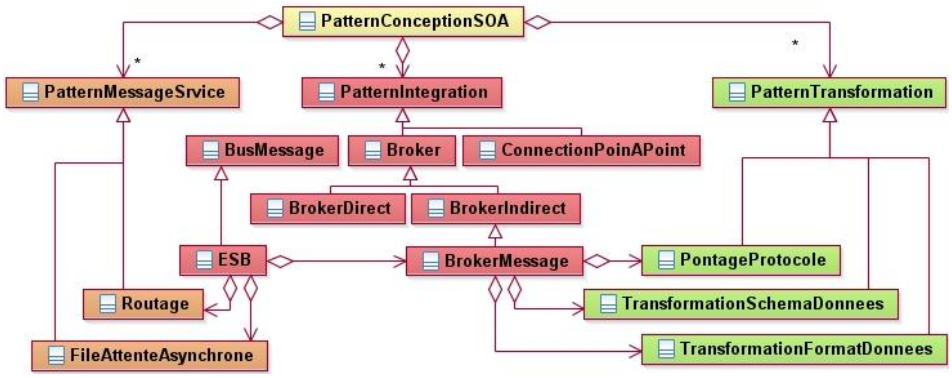


Figure I-15. Quelques patterns de conception SOA qui forment un ESB

Les patterns qui composent l'ESB peuvent être classés en trois groupes distincts : patterns de message de service (couleur ocre rouge), pattern d'intégration (couleur rouge framboise) et patterns de transformation (couleur vert anis).

Le premier groupe contient des patterns de type message de services :

- **File d'attente asynchrone (*Asynchronous Queuing*) :** mécanisme de file d'attente des messages qui permet des échanges de messages asynchrones afin d'augmenter la fiabilité de leur transmission. La file d'attente des messages agit comme un buffer intermédiaire qui reçoit les messages pour ensuite les rediriger aux services concernés. Ce pattern est primordial dans une architecture orientée services.
- **Routage (*Intermediate Routing*) :** c'est le mécanisme par lequel les chemins des messages sont sélectionnés dynamiquement en se basant sur leur contenu ou sur des données d'exécution. Plusieurs types de routages existent : basé sur le contenu, basé sur la répartition de charge (*load balancing*), etc. Le routage peut aussi nécessiter l'utilisation d'un moteur de règles pour déterminer les destinations ou des expressions comme XPath (W3C07, 1999) et XQuery (W3C06, 2007).

Trois principaux patterns d'intégration existent : le *Bus de message*, le *Broker* et la *Connexion point à point*, la portée de ces patterns dépasse largement le cadre dans lequel ils sont utilisés dans SOA (Trowbridge et al., 2004). Le pattern *connexion point à point* représente la façon la plus simple pour connecter deux systèmes. Il permet d'assurer que le message soit reçu par le système récepteur, pour cela le système qui envoie le message doit connaître la location du récepteur. Le but du pattern *Bus de message* est d'offrir un mécanisme de communication commun entre différents systèmes. Quant au pattern *Broker*, il permet de découpler les systèmes source des systèmes cible (Trowbridge et al., 2004) et de séparer la logique applicative de la logique de distribution. Le but de ce pattern est de réduire la prolifération des connexions point à point entre les systèmes (Endrei et al., 2004). Enfin, trois types de Broker existent (Hohpe et al., 2003) :

1. **Broker Direct :** initialise la communication entre les endpoint. Ensuite, les deux endpoints communiquent directement via des *Proxies* (Gamma et al., 1995). UDDI est un exemple d'application de ce type de pattern.
2. **Broker Indirect :** toutes les communications entre les endpoints passent par le broker qui joue le rôle du *Mediator* (Gamma et al., 1995).
3. **Broker Message :** c'est une spécialisation du Broker Indirect. Les communications entre les endpoints sont basées exclusivement sur des messages. Le Broker message est communément appelé Hub-and-spoke (Keen et al., 2004; Trowbridge et al., 2004).

Enfin, le troisième groupe contient des patterns de transformation qui traitent les aspects liés aux échanges des données. Ils sont composés de trois de patterns (Erl, 2009) :

- **Pontage de protocole (*Protocol Bridging*) :** Permet la communication entre des services qui utilisent différents protocoles de communication (JMS[14], HTTP[15], SOAP) ou différentes versions du même protocole (SOAP 1.1 et SOAP 1.2).
- **Transformation du format des données :** Même si deux services communiquent via le même protocole, ils peuvent échanger des données qui ont des formats différents : jeu de caractères (*charsets*), formats de fichiers ou de données (CSV[16] ou XML).
- **Transformation du modèle de données :** Même si le protocole de communication et le format de données sont les mêmes, la définition du modèle sur lequel la structure de données est basée peut-être différente. La transformation du modèle de données est nécessaire si le client et le serveur utilisent un schéma XML différent pour représenter la même donnée.

L'ESB est une technologie d'intégration orientée service

En plus d'être un pattern, l'ESB est aussi une technologie d'intégration. Dans ce paragraphe, on met en évidence les différences entre l'ESB par rapport à la technologie d'intégration appelée Hub-and-spoke. Puis, on positionne l'ESB par rapport aux topologies d'intégration et physique.

(Définition 11 - Enterprise Application Integration - EAI) l'intégration des applications d'entreprise met à disposition des applications d'entreprise des messages et d'autres technologies de communication distribuées pour leur permettre soit d'échanger des informations sur les ressources

[14] Java Message Service

[15] HyperText Transfer Protocol

[16] Comma-Separated Values

de l'entreprise, soit d'invoquer les fonctions des unes et des autres (Hohpe et al., 2003; Zimmermann, 2009). Plusieurs technologies d'intégrations existent : Batch, ETL[17], Hub-and-spoke, ESB.

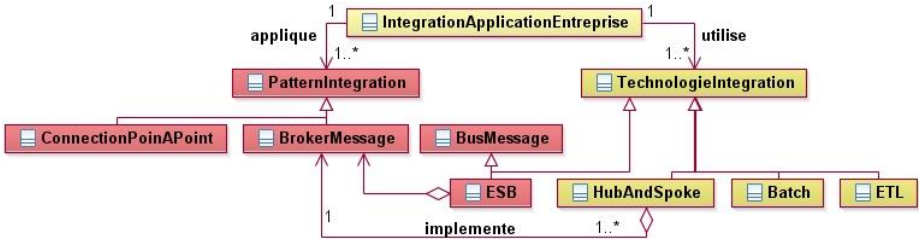


Figure I-16. Comparaison entre les technologies ESB et Hub-and-spoke

La Figure I-16 montre que la technologie Hub-and-spoke est une implémentation du pattern Broker Message. Cette technologie est apparue dans les années 90. Elle était généralement utilisée pour résoudre des problèmes d'intégration entre plusieurs applications existantes en centralisant le contrôle de la configuration : information de routage, nommage de service, etc. Les solutions Hub-and-spoke étaient largement propriétaires et n'utilisaient pas des standards. Elles n'avaient pas pour vocation de permettre au système d'information d'être ouverts et interopérables (Chappell, 2004).

La distinction entre un ESB et une solution de type Hub-and-spoke peut se faire sur la centralisation du contrôle et la distribution de l'infrastructure. Un ESB permet un contrôle centralisé et une infrastructure (implémentation) distribuée, alors que le contrôle et l'implémentation sont centralisés dans une solution traditionnelle de type Hub-and-spoke. Il est à noter que l'implémentation d'un ESB est susceptible d'être centralisée, s'il est installé sur un seul cluster. Cependant, l'implémentation peut facilement évoluer vers une infrastructure physiquement distribuée comme un Bus de messages. La technologie ESB permet un déploiement et une implémentation incrémentales et rend possible l'ajout des capacités de traitement supplémentaires sans affecter l'infrastructure existante. La Figure I-17 résume les relations entre les patterns d'intégration et la topologie physique (Trowbridge et al., 2004).

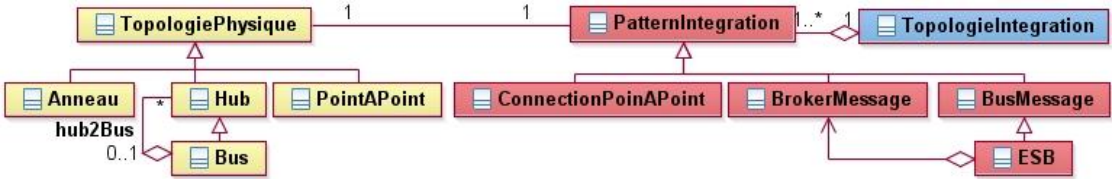


Figure I-17. La relation entre les concepts EAI, Hub, Bus, Hub-and-spoke et ESB

La compréhension de cette relation est nécessaire pour ne pas confondre les différents patterns d'intégration connexion Point-à-Point, Bus de message, Broker et ESB avec les topologies

[17] Extract Transform Load

d'intégration ou physique (Figure I-17). Pour illustrer nos propos, on fournit quelques exemples des différents types de déploiement possibles d'un ESB. Selon son besoin une organisation peut être amenée à déployer un seul ou plusieurs ESB (Roshen, 2009).

ESB connectées point à point

Dans ce type de déploiement les ESB sont connectés directement les uns aux autres et ils partagent un registre de services commun. Les services sont visibles dans toute l'entreprise (Figure I-18).

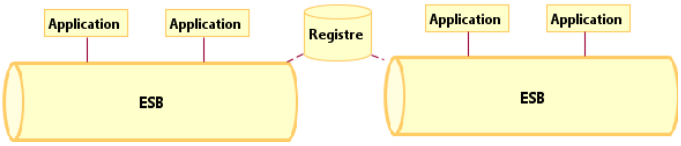


Figure I-18. ESB connectés point à point (Roshen, 2009)

ESB fédérés (Bus de message)

Dans ce type de déploiement les ESB sont aussi connectés les uns aux autres, mais l'un d'eux a une relation de type master/slave avec les autres ESB. Il exerce un contrôle sur ce qui est accessible aux participants. Ce type de déploiement peut être utilisé par des départements relativement autonomes d'une entreprise qui souhaitent partager certains services entre eux (Figure I-19).

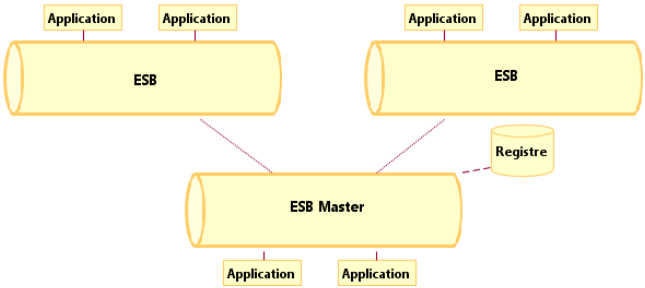


Figure I-19. ESB fédérés (Bus de message) (Roshen, 2009)

ESB avec une topologie d'intégration du type Broker

Dans ce type de déploiement chacun des ESB a son propre registre de services. Ce pattern permet à des entreprises autonomes d'exposer une sélection de services aux entreprises partenaires. Cette configuration est représentée dans la Figure I-20.

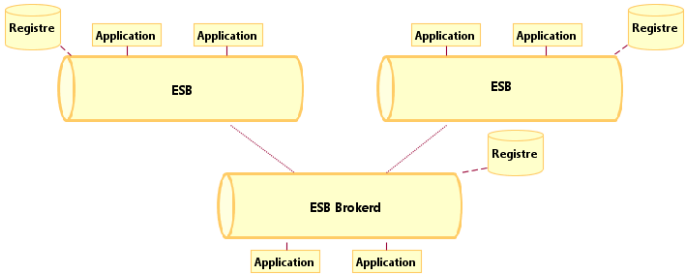


Figure I-20. ESB brokereds (Roshen, 2009)

On estime nécessaire de clarifier la relation entre les trois notions : Hub, Broker et Bus. En effet, d'une part, il est très commun de confondre Hub et Broker. D'autre part, il est intéressant de déterminer la relation entre un Bus et un Hub.

On insiste sur le fait que le terme Hub n'est pas un pattern d'intégration. C'est un concept utilisé au niveau de la topologie physique **(Définition 12 - Topologie physique)** qui décrit comment chaque système est connecté physiquement au réseau. Cette topologie est déterminée par les nœuds hardware et les câbles réseau qui les connectent. La topologie physique contient 4 types de connexion : pont-à-point, bus, hub, ring (Figure I-21).

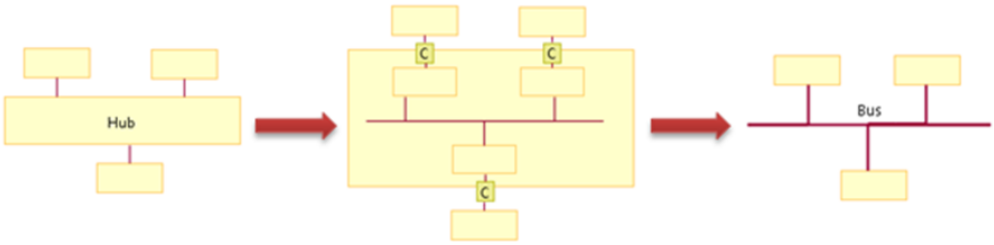


Figure I-21. Le Hub peut être physiquement distribué comme un Bus (Keen et al., 2004)

Un Bus **(Définition 13 - Bus)** fournit un mécanisme de communication commun entre des systèmes disparates. Il peut être considéré comme une variante d'un Hub physiquement distribué comme un ensemble de Hub fédérés (Keen et al., 2004; Trowbridge et al., 2004)

Il est aussi nécessaire d'introduire un deuxième type de topologie : **(Définition 14 - Topologie d'intégration)** la topologie d'intégration décrit les mécanismes utilisés par différents systèmes pour communiquer. Cette typologie est constituée de la combinaison des trois patterns : point-a-point, Broker et Bus (Trowbridge et al., 2004).

C'est la combinaison des deux topologies (d'intégration et physique) qui permet de comprendre, par exemple, que des ESB fédérés qui forment un Bus de message au niveau de la topologie de l'intégration peuvent être implémentés au niveau physique avec une topologie de type Bus ou Hub (Figure I-22).

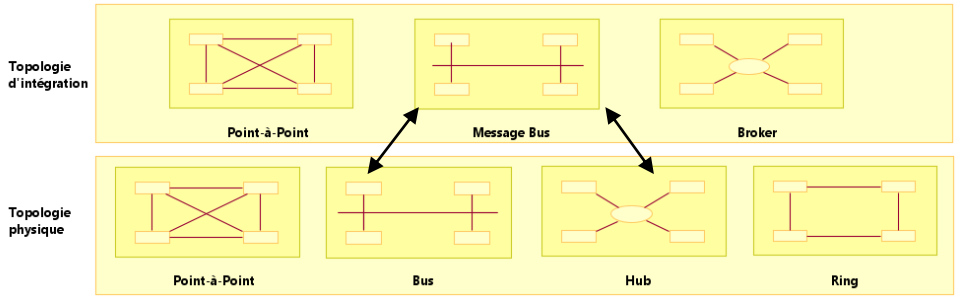


Figure I-22. Les topologies d'intégration et physique (adapté de (Trowbridge et al., 2004))

5. CONCLUSION DU CHAPITRE

L'application du style d'architecture orientée service est motivée par l'idéal de créer une entreprise orientée services (*Service Oriented Enterprise - SOE*) dans laquelle chaque processus métier est exposé à travers un service métier qui est lui-même composé de plusieurs services. Le déploiement de l'architecture orientée service à l'échelle de l'entreprise nécessite d'augmenter la portée des principes de l'orienté service non seulement aux applications de l'entreprise mais aussi à l'Architecture d'Entreprise. L'architecture d'une entreprise orientée service doit prendre en compte des considérations métiers et technologiques. On estime que les organisations qui ont un cadre d'architecture bénéficient d'un levier important pour porter le style d'architecture orientée service à l'ensemble de l'entreprise. Le cadre d'architecture permet d'amplifier les bénéfices attendus de SOA, notamment la réutilisation, l'agilité et l'interopérabilité. Cependant, les organisations ont des difficultés à soulever ce défi, parce qu'elles ont des difficultés à justifier la mise en place des cadres d'architecture (Infosys, 2009) et maîtrisent mal son déploiement. À titre d'exemple, les expériences de mise en place d'un cadre d'architecture dans le FBI (GAO00, 2005) et dans le département de défense américain (GAO01, 2005).

Comme la conception d'un service qui respecte les principes de SOA dépend fortement de la conception de son contrat de service, toute méthode de conception de services doit porter une attention particulière à la conception du contrat de services. La méthode de conception de services proposée dans ce manuscrit explique comment concevoir une architecture de services qui respecte les principes de SOA introduits dans ce chapitre. Une profonde compréhension des principes et des patterns de SOA est indispensable pour expliquer comment SOA aide les organisations à gagner en agilité (cf. Chapitre II) et comment elle supporte interopérabilité (cf. Chapitre III).

CHAPITRE II : Perspective métier de SOA

1. INTRODUCTION DU CHAPITRE

Les mutations contemporaines sont tellement rapides que la réactivité devient d'ores et déjà un facteur clé de succès dans le contexte économique actuel et plus particulièrement dans les industries de l'immatériel (Kellogg et al., 2006). Ces nouvelles exigences ont suscité un processus continu de synchronisation des efforts pour bien assimiler les transformations et les variations métiers. Traditionnellement, les technologies de l'information ont été considérées comme un centre de coût dont la finalité est de réaliser des affaires (Korhonen et al., 2010). Or, la démocratisation de l'utilisation des infrastructures IT au niveau opérationnel (Carr, 2003) a conduit au transfert de l'intérêt des entreprises vers une utilisation plus stratégique de l'IT (Bigand et al., 2004).

La problématique de l'alignement entre le métier et l'IT n'est pas nouvelle (Luftman et al., 1993) mais elle n'en reste pas moins d'actualité (Luftman et al., 2009). Toutefois, la mise en place d'un tel dispositif présente des difficultés (Avison et al., 2004). Dès lors, diverses recherches ont proposé différentes méthodes pour rendre opérationnel cet alignement. Ainsi, des travaux ont suggéré d'étendre le modèle d'alignement stratégique (*Strategic Alignment Model* -SAM) proposé par (Venkatraman et al., 1993) ; d'autres ont proposé de nouvelles démarches comme la correction de l'alignement stratégique et une évolution conjointe SI-processus (Etien, 2006) ou encore la structuration d'une intention partagée entre les niveaux opérationnels et stratégiques (Thevenet, 2009).

L'organisation traditionnelle de l'entreprise en logique sectorielle, en silos, ne permet ni l'alignement entre le métier et l'IT, ni l'amélioration de l'agilité des entreprises. Dans ce chapitre, on s'intéresse à la perspective métier de SOA et on étudie la combinaison de SOA et BPM pour mettre en évidence les facteurs clés à prendre en considération afin d'aligner le métier et l'IT. En effet, SOA a commencé comme un style d'architecture utilisé au niveau IT afin de permettre une activité de conception reproductible (cf. Chapitre I). Ensuite, il a évolué vers une initiative métier pour créer des services supportant le métier. Le but de ce chapitre est d'étudier la perspective métier de SOA.

Ce chapitre est structuré de la manière suivante : Dans la prochaine section, on présente un état de l'art actualisé de BPM. Puis, on met en évidence les synergies entre SOA et BPM à travers la relation entre les notions de processus métier et de service métier et on propose ensuite un modèle *ad hoc* nommé modèle EMT (*Économie, Méthodologie et Technologie*). Ce modèle prend en considération les aspects économiques, méthodologiques et technologiques justifiant la mise en place des approches BPM et SOA pour améliorer l'agilité des organisations. Enfin, on présente un cadre de

référence contenant les différents éléments à prendre en compte lors de l'implémentation des approches BPM et SOA pour l'alignement métier-IT.

2. MANAGEMENT DES PROCESSUS MÉTIERS

L'avènement du terme BPM (*Business Process Management*) introduit par l'analyste du Gartner David W. McCoy en 2000 a ajouté une nouvelle dimension à SOA (McCoy, 2009). Au début, terme était utilisé pour désigner une technologie déjà existante (workflow-brokers). Cependant, actuellement ce terme désigne aussi bien une discipline managériale, une plateforme technologique ou un style d'implémentation.

2.1. Une vision transversale de BPM

BPM propose une approche de management centrée sur les processus métiers dans le but d'améliorer l'agilité et les performances opérationnelles (Melenovsky et al., 2005). Il permet d'améliorer le processus du point de vue de l'utilisateur final : par exemple, l'amélioration du temps d'exécution du processus. BPM est basé sur un ensemble d'outils, méthodes et bonnes pratiques dont l'objectif est la gestion et l'optimisation continue des activités des organisations et de leurs processus. Ainsi, la mise en œuvre de BPM peut ne pas avoir de répercussions sur le système d'information (Zairi, 1997).

L'évolution de BPM a connu quatre phases (Smith et al., 2003; Zhao, 2007). La première a débuté à partir des années 1920 avec l'introduction de processus non automatisés dans les méthodes de travail. Ensuite, la deuxième période s'étala entre les années 60 et 90 avec l'émergence des SI, les processus métiers étaient mis en place avec des systèmes de *batchs*. À cette époque BPM n'était pas réellement supporté par les systèmes d'information. Puis, dans les années 90 l'émergence des approches et méthodologies d'intégration des entreprises a favorisé l'apparition des processus métiers intra-organisationnels. Les *workflows* (*Workflow Management System -WMS*) étaient alors utilisés pour intégrer les processus métiers internes. Enfin, actuellement la mondialisation et l'évolution rapide des nouvelles opportunités, ont drainé le quatrième type de BPM. Cette étape, caractérisée par une collaboration inter-organisationnelle, est soutenue par l'extension des processus métiers afin de prendre en compte tous les partenaires de l'organisation (fournisseurs, partenaires commerciaux, clients, administrations, etc.) ainsi que l'hétérogénéité des systèmes et des ressources. Le BPM est constitué de trois briques distinctes mais interdépendantes : une discipline de management, une plateforme technologique et un nouveau style pour l'implémentation de solutions basées sur des processus automatisés (Silver, 2010).

BPM comme discipline managériale : dans les années 80, Rummler présentait le BPM comme une nouvelle discipline de management en conseillant aux entreprises d'adopter une structuration par processus axée sur le client qu'il formalisa ensuite dans un ouvrage (Rummler et al., 1990). En documentant l'ensemble des phases du processus et en dépassant les limites fonctionnelles et organisationnelles, la modélisation par processus est un élément fondamental du BPM (Figure II-1).

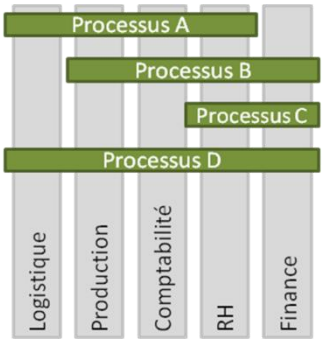


Figure II-1. Perspective par processus (Silver, 2010)

Cette structuration en processus est un des nombreux modèles utilisés pour l'amélioration continue de la performance organisationnelle comme le management par objectifs (Drucker, 1954) et ses applications (Humble, 1971), les tableaux de bord de tradition française (Bessire et al., 2005) ou prospectifs d'inspiration Anglo-Saxonne (Kaplan et al., 1998; Kaplan et al., 2001).

BPM comme plateforme technologique : il existe plusieurs aspects dans l'amélioration des processus métiers. Par exemple, la suppression des tâches sans valeur ajoutée. Cependant, le bénéfice de l'utilisation des processus métiers peut être plus important avec l'utilisation des technologies pour l'exécution des processus. BPM Suite (*Business Process Management Suite - BPMS*) est un ensemble d'outils et de composants permettant une automatisation et une amélioration continue des processus métiers.

BPM comme style d'implémentation : augmenter l'agilité métier exige que le métier joue un rôle plus important et direct dans les projets d'améliorations des processus. BPM prône et permet un nouveau style agile et itératif dans lequel le métier et l'IT collaborent dans des cycles courts. L'architecture des BPMS est adaptée à un changement continu des processus. Les BPMS proposent une plateforme de collaboration et d'exécution des processus partagée entre les métiers et l'IT.

2.2. La distinction entre les BPMS et BPA

Dans leur ouvrage *Business Process Management : The Third Wave* Smith Howard et Fingar Peter (2003) ont correctement prédit que l'appropriation par les utilisateurs professionnels de leurs

propres processus est critique pour l'évolution de BPM. Les outils BPM doivent être adaptés aux besoins des utilisateurs et des analystes métier. Aujourd'hui, on distingue deux types d'outils BPM : les (*Business Process Analysis* – BPA) et les BPMS.

Les BPA regroupent des outils et des technologies offrant une vue multidimensionnelle de l'entreprise à travers plusieurs types des modèles. Ils lient les modèles de processus aux modèles des règles métiers, aux rôles organisationnels, et aux buts stratégiques. Les BPA permettent une mise en relation des modèles stratégiques et par processus mais aussi de comprendre les dépendances et également l'importance stratégique des processus. L'un des objectifs des BPA est d'assurer l'adhésion entre les processus exécutés et les règles de gouvernance pour s'assurer que les obligations contractuelles et la qualité des contrats de services sont respectées. Ces outils visent à réduire le temps de réaction des décideurs aux événements qui peuvent affecter la performance des processus, en permettant une évaluation rapide de l'impact des décisions du management sur les indicateurs de performance des processus (Zur. Mühlen et al., 2010). Alors que, les BPM Suite se focalisent sur certains modèles de processus pour les rendre exécutables dans le but d'améliorer leurs performances tout en assurant une meilleure qualité. Ils permettent aux utilisateurs métier de passer rapidement de la modélisation d'un processus à son administration ou à son optimisation. Ils offrent également une représentation des processus en exécution partagés entre le métier et l'IT leur permettant ainsi de collaborer. Cette collaboration se traduit par le partage d'un modèle de processus où le processus en exécution correspond parfaitement à celui qui a été défini par le métier. De plus, les équipes métier recueillent les informations nécessaires pour l'amélioration continue du processus (Gilbert, 2007).

Le marché des suites BPM est complexe, il présente une multitude d'acteurs opérant sur différentes niches comme l'informatique décisionnelle, l'intégration d'applications d'entreprise ou la modélisation. Ces éditeurs offraient à la base des produits auxquels ils ont ajouté d'autres briques logicielles. Il y a donc une forte incertitude quant à l'évolution des BPMS (Harmon, 2010).

3. PROPOSITION DU MODÈLE EMT POUR RATIONALISER LA SYNERGIE BPM-SOA

3.1. BPM et SOA: concurrence ou synergie ?

Plusieurs travaux ont montré des synergies entre BPM et SOA (Kamoun, 2007). L'utilisation de SOA permet de découvrir des services réutilisables pouvant être orchestrés pour mettre en place des processus métiers dynamiques. Cette combinaison assure la conception itérative des processus basés sur des services qui pourront être changés rapidement. La combinaison de BPM et SOA crée

des opportunités en rendant visibles les processus de l'entreprise et en l'aidant à développer une infrastructure IT flexible. Cependant, l'alignement de ces deux concepts exige une profonde transformation des organisations (Kamoun, 2007; Hiemstra et al., 2009).

D'une part, la démarche BPM ne fournit pas d'unité élémentaire de granularité suffisamment fine pour construire un système, alors que la notion de service offre un cadre d'architecture pour la conception de processus métiers en assurant la répartition des capacités de l'entreprise. D'autre part, SOA peut produire des services réutilisables, mais sans garantie d'agilité métier (Bajwa et al., 2009).

Le but de SOA est de découvrir des services stables et réutilisables alors que les processus découverts avec BPM sont centrés sur les besoins du client (amenés à changer pour garantir l'agilité de l'entreprise). SOA promeut la consolidation des opérations redondantes et améliore la capacité d'adaptation aux changements métiers (c.f., la Figure II-2 inspirée de (Cummins, 2009)).

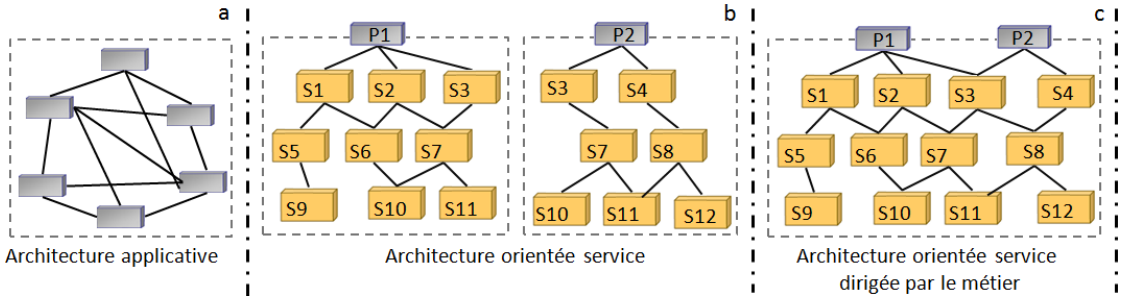


Figure II-2. Du plat de spaghettis à SOA-BPM

Lorsque SOA est utilisée, il y a une combinaison des capacités entre les différents métiers de l'entreprise. Cette consolidation des capacités permet à l'entreprise de faire des économies d'échelle et lui offre la possibilité de réduire les coûts ainsi que d'améliorer la qualité (Cummins, 2009). BPM aide l'organisation à supprimer ses silos fonctionnels et à unifier ses informations critiques comme les besoins du consommateur. L'entreprise devient un ensemble composé de capacités réutilisables dans des contextes métiers hétérogènes. La réalisation de cette synergie n'est pas simple (Kamoun, 2007; Hiemstra et al., 2009), une juxtaposition des implémentations technologiques n'est pas suffisante. Il faut un alignement complet des deux approches pour établir une architecture orientée service dirigée par le métier.

Pour comprendre la synergie SOA-BPM, il faut répondre à une question essentielle à savoir : *que représente un processus métier dans SOA ?* Deux visions s'opposent. La première, souvent utilisée dans des approches techniques de SOA, positionne les processus métiers au-dessus des services métiers. La deuxième explique qu'un processus métier qui invoque un service fait lui-même partie d'un autre service métier (Cummins, 2010). On adhère à la deuxième vision de la relation entre le processus métier et le service métier illustré dans la Figure II-3 dans laquelle l'architecture de

services *A* accepte deux requêtes. Chacune d'elles invoque un processus métier différent : les processus métiers *X* et *Y* offrent des capacités différentes. Le processus *X* délègue une partie de ses responsabilités à un autre service métier *T*. Les deux processus proposent des capacités différentes, mais peuvent partager la même capacité (ici le sous-processus *Z*). Les processus peuvent faire appel à différentes capacités : tâche humaine, application ou un autre service métier (Figure II-3).

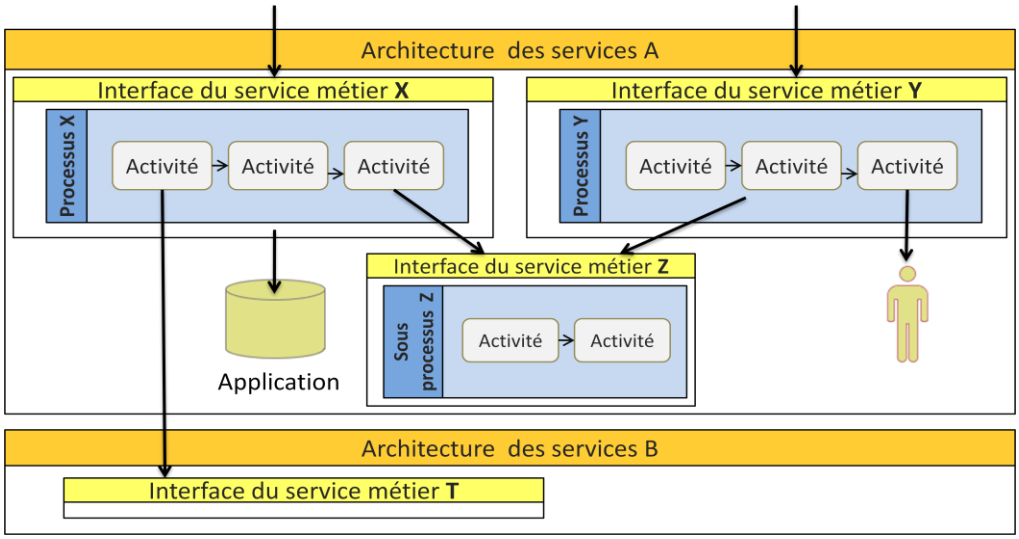


Figure II-3. Processus métier et service métier

3.2. Le modèle EMT

Pour démontrer la synergie des approches BPM et SOA on propose le modèle EMT avec ces trois dimensions : économique, méthodologique et technologique (Figure II-4).

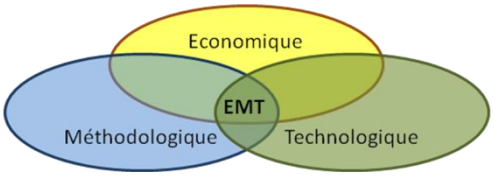


Figure II-4. Le modèle EMT

3.2.1. Le niveau économique

Une récente étude du Gartner (2010) confirme l'intérêt que portent les entreprises au BPM. En 2010, l'amélioration des processus métiers était la priorité des *Directeurs des Systèmes d'Information* (DSI).

L'ensemble SOA et BPM offrent une parfaite complémentarité pour l'informatisation de l'entreprise. Frye (2006) affirme que SOA et BPM *« sont les deux faces d'une même médaille »*. BPM est un bon argument pour déployer SOA en entreprise. Sans BPM, SOA permet uniquement d'augmenter le *Retour Sur Investissement* (RSI) de l'IT. L'utilisation de BPM justifie quant à elle la

mise en place de SOA en lui attribuant une partie du RSI résultant de l'amélioration des processus métiers.

3.2.2. Le niveau méthodologique

Dans BPM, il est recommandé d'utiliser une approche descendante pour découvrir les processus métiers à partir des objectifs et des exigences métiers. Ensuite, l'IT utilisera ses ressources systèmes et logicielles pour offrir une implémentation aux processus identifiés. Il y a plus de garantie que les exigences métiers soient réalisées si elles sont définies du point de vue du métier (Gulledge, 2010). En revanche, dans SOA les deux méthodes ascendante et descendante peuvent être combinées pour offrir une unité de travail conceptuelle unifiée : le service métier (Catts et al., 2009). Le service métier permet à l'IT et au métier de collaborer, en offrant à chacun une visibilité sur l'autre domaine. D'un point de vue métier, les experts métiers de BPM utilisent les techniques et les outils BPM pour découvrir et créer les processus métiers. Ensuite, ils les affinent jusqu'à l'obtention d'un ensemble de services métier élémentaires. D'un point de vue IT, les développeurs se concentrent en premier lieu sur les actifs existants et leur capacité en les transformant en services de base. Puis, ils conçoivent des services de plus en plus sophistiqués jusqu'à l'obtention des services métiers requis (Oracle, 2009).

3.2.3. Le niveau technologique

Pour diverses raisons, les projets d'améliorations des processus métiers ne débouchent pas toujours sur les résultats escomptés. Plusieurs facteurs de succès ont été proposés (Bashein et al., 1994). Comme Gulledge (2008; 2010), on estime que la réorganisation des processus n'est efficace que si et seulement si les flux d'informations s'appuient sur des systèmes alignés avec ces nouveaux processus.

L'émergence de SOA a été accompagnée par la promesse d'automatiser les processus métiers basés sur des services métiers à travers internet et en utilisant des standards. L'échange est alors dirigé par les processus métiers internes de chacun des partenaires. Le couplage faible entre le fournisseur et le consommateur leur permet de s'échanger des messages tout en cachant l'implémentation des services pour le client en lui offrant la possibilité d'utiliser des technologies différentes. Toutefois, pour automatiser les processus métiers, il faut une infrastructure adaptée à l'intégration des services métiers. Ceci peut être fait par les web-services par exemple.

Dans un SI traditionnel, il est difficile de faire évoluer les processus métiers, car ces derniers sont imbriqués dans les applications informatiques de l'entreprise. SOA apporte une réponse à ce problème en séparant la logique et les exigences métiers. Cette dissociation entre les processus métiers d'une part et l'infrastructure technologique utilisée d'autre part permet de composer les

services métiers dans des applications métiers orientées service ou *Service Oriented Business Application* (SOBA) (Bloomberg, 2009). En revanche, à la différence des approches d'intégration classiques, l'intégration se fait au niveau des activités des processus et non pas au niveau des applications métiers. De plus, cette intégration se fait via des interfaces services bien définies et accessibles par plusieurs utilisateurs (cf. Chapitre I - 2).

Les exigences métiers doivent justifier le choix de l'architecture qui conduit à un arbitrage technologique. Les organisations doivent tirer profit de leurs systèmes existants par le biais d'une infrastructure adéquate. Ils peuvent implémenter les BPMS pour automatiser leurs processus métiers et les ESB afin d'assurer l'acheminement des messages et pour faire abstraction des technologies utilisées pour implémenter les services.

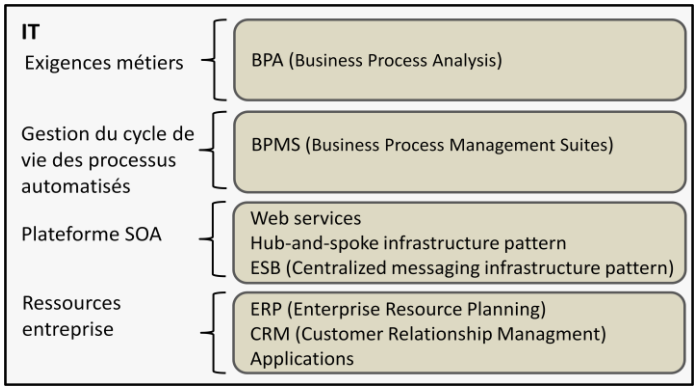


Figure II-5. Les briques logicielles pour l'implémentation BPM-SOA.

On distingue quatre couches dans l'IT de l'entreprise. La Figure II-5 décrit globalement la relation entre les différentes briques logicielles. Chaque entreprise doit choisir la configuration correspondant le mieux à ses besoins. Le but est de réduire l'écart entre les exigences métiers qui peuvent être décrites avec les outils BPA (couche 1) ou BPMS (couche 2) et les ressources de l'entreprise (couche 4). Dans cette architecture, les outils BPMS peuvent être utilisés pour faire le lien entre les processus automatisés et les ressources de l'entreprise organisées à l'aide de la plateforme SOA.

Bien que la Figure II-5 soit générale, la vision qu'elle propose reste pertinente pour construire une solution métier orientée services. D'ailleurs, plusieurs éditeurs de plateformes IT tels que Oracle[18] ou IBM[19] proposent des briques logicielles sur les différentes couches. D'autres entreprises comme MEGA[20] (spécialiste BPA) et Appian[21] (spécialiste BPMS) forment des alliances

[18] http://www.oracle.com/index.html

[19] http://www.ibm.com

[20] http://www.mega.com/fr

[21] http://www.appian.com/

pour proposer des solutions complètes. Il est aussi important de noter qu'à ce jour aucune solution Open Source ne se positionne sur l'ensemble des quatre briques logicielles.

3.3. Cadre de référence pour l'alignement avec BPM et SOA

À partir de la littérature et de notre expérience dans le projet ASICOM on a identifié six éléments pour assurer l'alignement métier-IT. Chacun de ces éléments représente un facteur de succès critique qu'il faut prendre en compte dans l'implémentation des approches BPM et SOA (Figure II-6). Ces éléments définis à un niveau générique permettent une compréhension globale de BPM et SOA et donnent une vue holistique de ces deux approches tout en structurant leur utilisation.

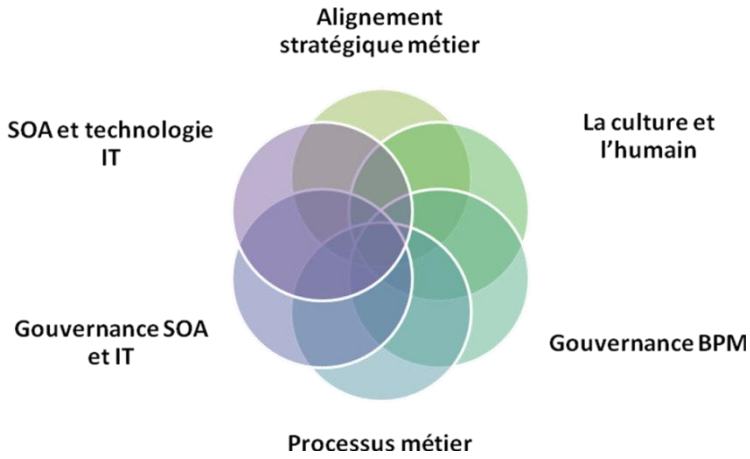


Figure II-6. Cadre de référence pour l'alignement métier-IT en utilisant BPM et SOA

Alignement stratégique métier

Pour que l'entreprise soit performante, ses systèmes et ses processus doivent être en permanence en cohérence avec sa stratégie (Thevenet, 2009). Les approches BPM et SOA ont besoin d'être alignées avec la stratégie de l'organisation. La stratégie d'alignement permet d'identifier les futures capacités métiers en combinant des éléments stratégiques et opérationnels : la connaissance, l'organisation, les processus et la technologie (Catts et al., 2009). Ces capacités métiers sont ensuite utilisées pour identifier les futures capacités des processus à implémenter (Figure II-7).

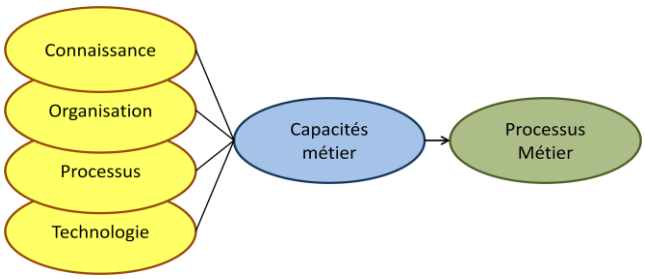


Figure II-7. Les capacités métiers (Catts et al., 2009).

Regev et Wegmann (2004) différencient l'alignement externe et interne. L'alignement métier concerne les composants internes de l'organisation comme les processus métiers et les systèmes IT dédiés à leur exécution. Alors que l'alignement externe correspond à l'alignement entre l'entreprise et son environnement. On estime que l'agilité métier-IT doit être l'objectif de chaque stratégie d'alignement interne et l'interopérabilité au sens de (Chen et al., 2010) l'objectif de la stratégie d'alignement externe.

La culture et l'humain

Le succès des initiatives BPM et SOA dépend beaucoup du facteur humain. Deux dimensions sont essentielles dans ce domaine : (a) les compétences et les expériences requises pour implémenter ces démarches dans une organisation (l'humain) et (b) le leadership et l'acceptation des approches BPM et SOA (la culture).

Gouvernance BPM

Dans les initiatives liées au BPM, l'un des défis en matière de gestion est de définir clairement les rôles, les responsabilités, et les processus de décision. La gouvernance BPM se consacre à des questions de type : *qui est responsable de quel processus ? Quels sont les droits d'un responsable de processus ? Comment mettre en place des incitations pour permettre une exécution plus efficace des processus ?*

La gouvernance BPM peut être divisée en deux dimensions : (a) la gouvernance des processus (Markus et al., 2010) (b) la gouvernance de la gestion des processus (Spanyi, 2010). La première dimension met l'accent sur la mise en place d'une structure de gouvernance rentable, alors que la deuxième dimensions s'intéresse aux pratiques de management de la gouvernance BPM.

Processus métiers

Cet élément concerne l'identification des futures capacités des processus métiers et leur implémentation, ainsi que la définition des mesures et des métriques de performance des processus (*Key Performance Indicators* - KPI).

Gouvernance SOA et IT

Dans cet élément, il faut définir les exigences pour les futures gouvernances IT et SOA qui sont requises pour une mise en œuvre réussie de l'environnement IT et SOA avec les futures capacités des processus métiers. Malinverno (2006) vice-président du Garnter affirme que l'absence de gouvernance SOA affecte lourdement le déroulement des projets SOA « *SOA governance isn't optional - it's imperative. Without it, return on investment will be low and every SOA project out of pilot phase will be at risk* ».

SOA et technologie IT

Cet élément concerne l'évaluation, la modélisation et le déploiement des services, architectures, infrastructures et applications SOA ou IT utilisées pour la mise en place des processus métiers définis dans l'élément précédent.

4. CONCLUSION

Ce chapitre s'intéresse aux disciplines BPM et SOA en proposant une vue d'ensemble de la relation entre ces deux approches, notamment à travers les deux concepts service métier et processus métier et une vision de leurs relations.

Le modèle EMT (économique, méthodologique et technologique) est proposé pour démontrer que SOA et BPM sont des disciplines complémentaires qui offrent un avantage concurrentiel aux organisations en leur permettant d'améliorer leur agilité. Ces deux approches jouent un rôle important dans la découverte et l'optimisation continue des processus métiers, elles permettent une mise en œuvre complète de l'alignement entre le métier et l'IT. D'une part, la discipline BPM offre la possibilité de mettre en place des processus métiers centrés sur les clients et alignés sur les exigences métiers. D'autre part, SOA assure une infrastructure efficace et capable de répondre rapidement aux évolutions des processus métiers. Enfin, on a proposé un cadre de référence qui précise les éléments à prendre en compte toute méthodologie d'alignement métier-IT utilisant les approches BPM et SOA. Dans la suite de ce manuscrit on s'intéresse particulièrement à deux éléments du cadre d'architecture : processus métiers (cf. Chapitre IV) et SOA et technologies IT (cf. Chapitre V).

CHAPITRE III : Une architecture orientée services et dirigée par le métier pour rationaliser l'interopérabilité

1. INTRODUCTION DU CHAPITRE

L'interopérabilité des organisations (et plus particulièrement l'interopérabilité des systèmes d'information des organisations) est l'un des problèmes complexes auxquels l'entreprise est confrontée. Ce problème est crucial, car les systèmes d'information ont de plus en plus besoin de travailler ensemble. Dans le nouvel environnement économique mondial, l'interopérabilité est l'un des principaux résultats que les entreprises collaboratives doivent atteindre (Bourey et al., 2007). De même Ducq (2007) considère l'interopérabilité des systèmes comme une exigence de performance particulière de l'entreprise.

Le but de ce chapitre est de faire le lien d'une part, entre notre vision de SOA sur les perspectives conception d'architecture (cf. Chapitre I), et métier (cf. Chapitre II) et d'autre part la méthode de conception de services orientée métier qu'on propose pour modéliser les processus collaboratifs (cf. Chapitre IV) et les services publics de collaboration (cf. Chapitre V). Pour cela, on présente les différents types de collaboration entre plusieurs organisations. Puis, on utilise le cadre d'interopérabilité d'entreprise EIF (*Enterprise Interoperability Framework*) afin de positionner nos travaux et pour préciser les aspects d'interopérabilité affectés par les deux perspectives de SOA. Enfin, on explique comment combiner les aspects (métier, processus, service et donnée) du cadre *ATHENA Interoperability Framework (AIF)* avec les différents niveaux de la démarche MDI (CIM-Haut, CIM-Bas, PIM, PSM et Code) pour mettre en place une solution interopérable basée sur une architecture orientée services.

2. ÉTAT DE L'ART

2.1. Collaboration

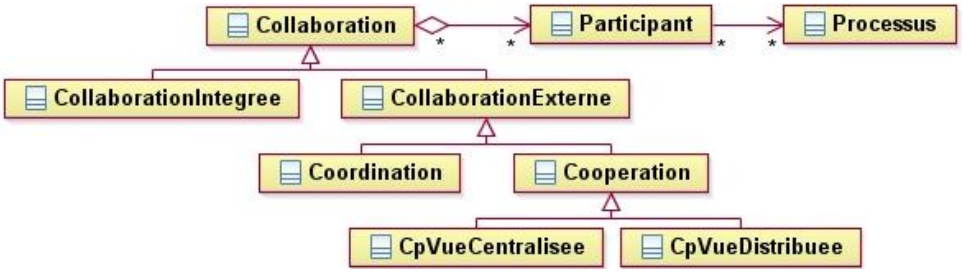


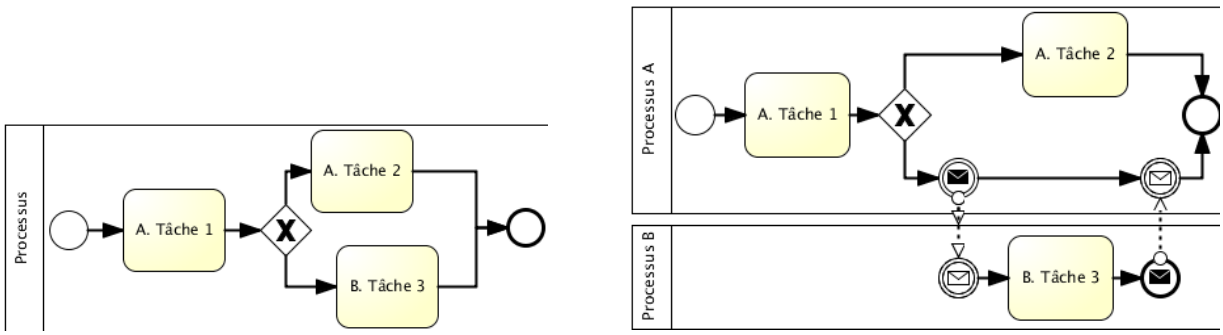
Figure III-1. Les types de Collaboration

On considère la *Collaboration* comme une collection de participants et leurs processus collaboratifs (Figure III-1). On distingue deux types de collaboration : (1) la *Collaboration Intégrée* qui signifie que les logiques métiers et applicatives sont intégrées dans un seul système monolithique

utilisé par tous les participants ; (2) la *Collaboration Externe* qui a lieu entre deux ou plusieurs systèmes autonomes.

La *Collaboration Externe* peut être aussi spécifiée avec deux types : (1) la *Coordination* dans laquelle l'exécution du processus est assurée par un seul participant (Figure III-1-a) et (2) la *Coopération*, dans laquelle l'exécution du processus est contrôlée par plusieurs participants (chaque participant contrôle uniquement l'exécution de ses propres activités). De plus, il existe deux vues distinctes pour la coopération : une vue centralisée et une autre distribuée :

1. Dans la **vue centralisée de la coopération** un seul processus contient les activités de tous les participants (Figure III-2-a). Dans cette vue proposée par (Liu et al., 2010) chacun des participants déploie et exécute le même processus collaboratif sur son propre Moteur d'Exécution de Processus (MEP). Puis, les MEP négocient entre eux pour déterminer qui exécutera l'activité suivante du processus collaboratif.
2. Dans la **vue distribuée de la coopération** les activités de chaque participant sont modélisées dans son propre processus. Dans ce cas, la collaboration entre les participants s'exprime par des échanges de messages entre leurs processus (Figure III-2-b). Cette vue a été proposée et étudiée par (Chebbi et al., 2006).



a b

Figure III-2. Les types de processus de collaboration

a : Coordination ou une vue centralisée d'une coopération | b : vue distribuée d'une coopération

La Figure III-2-a représente le processus exécuté par deux entreprises A et B dans le cadre d'une coordination ou dans une vue centralisée d'une coopération. La Figure III-2-b montre une vue distribuée du même processus dans le cadre d'une vue distribuée d'une coopération.

2.2. Interopérabilité

Dans la littérature, il existe diverses définitions de l'interopérabilité. De l'ensemble des définitions, on retient deux définitions qui couvrent les aspects techniques et métier de l'interopérabilité la définition technique de l'interopérabilité proposée par (IEEE02, 1990)

(Définition 15 - Interopérabilité technique)[22] l'interopérabilité est la capacité de deux ou plusieurs systèmes ou composants d'échanger l'information et de l'utiliser. Cette définition insiste sur l'échange d'informations entre deux systèmes différents. Puis, la définition métier proposée par (ATHENA02, 2007) **(Définition 16 – Interopérabilité métier)**[23] L'interopérabilité est la capacité opérationnelle et organisationnelle d'une entreprise pour coopérer avec ses partenaires métier pour mieux créer, diriger et développer des relations basées sur le support de l'IT au métier avec l'objectif de créer de la valeur. En se basant sur ces définitions, on constate que l'interopérabilité métier étend l'interopérabilité technique pour couvrir les aspects organisationnels et opérationnels.

2.3. L'architecture dirigée par les modèles et l'interopérabilité dirigée par les modèles

2.3.1. L'architecture dirigée par les modèles

En novembre 2000 l'OMG annonçait son initiative MDA (*Model Driven Architecture*[24])(Soley, 2000), puis la première version était proposée en 2001 et la spécification finale à été adoptée en 2003 (OMG04, 2003). L'approche MDA s'inscrit dans une tendance plus générale appelée Ingénierie Dirigée par les Modèles (IDM) qui met les modèles au centre du développement des logiciels et des systèmes d'information. Les deux principaux artefacts de l'ingénierie dirigée par les modèles sont les modèles et les transformations de modèles.

Pour permettre aux organisations de faire évoluer leurs modèles d'applications indépendamment de l'évolution des plateformes technologiques, MDA préconise l'élaboration de modèles pérennes en distinguant les modèles indépendants et spécifiques aux plateformes. Pour cela, MDA propose trois types de modèles à partir de différents points de vue (OMG04, 2003) :

- ***Computation Independent Model – CIM* :** appelé aussi modèle de domaine ou modèle métier, le CIM montre une vue qui se focalise sur l'environnement et les exigences du système.
- ***Platform Independent Model – PIM :*** modèle indépendant des plateformes techniques, le PIM représente une vue du système qui se focalise sur les entités fonctionnelles du système indépendamment des détails nécessaires à une plateforme particulière.

[22] « Interoperability is the ability of two or more systems or components to exchange information and to use the information that has been exchanged »

[23] « The organisational and operational ability of an enterprise to cooperate with its business partners and to efficiently establish, conduct and develop IT-supported business relationships with the objective to create value »

[24] Architecture Dirigée par les Modèles

- ***Platform Specific Model – PSM :*** une vue du système qui se focalise sur les informations détaillées concernant l'utilisation d'une plateforme spécifique.

Quant à la transformation de modèles, elle consiste à utiliser des techniques de transformations pour mettre en relation les trois principaux modèles CIM, PIM et PSM. Ces techniques de transformation peuvent être utilisées pour générer le PIM à partir du CIM ou le PSM à partir du PIM.

2.3.2. L'interopérabilité dirigée par les modèles

Le but de l'interopérabilité dirigée par les modèles (*Model Driven Interoperability - MDI*) est d'adapter MDA pour résoudre les problèmes d'interopérabilité. L'approche MDI peut être utilisée par deux entreprises qui ont comme but d'améliorer leurs performances et qui veulent interopérer non seulement au niveau du code, mais aussi au niveau de la Modélisation d'Entreprise en utilisant les ontologies comme support (Bourey et al., 2007).

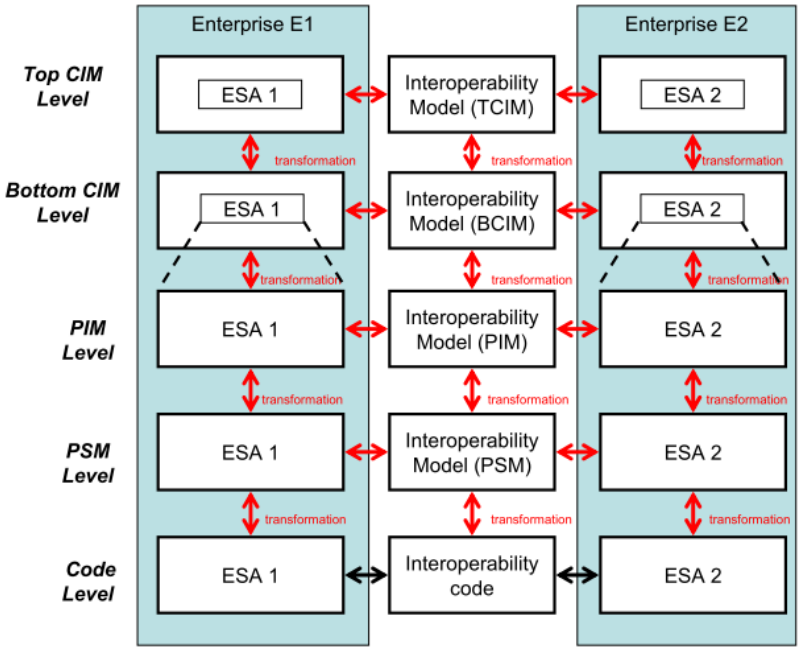


Figure III-3. Modèle de référence MDI[25] (Bourey et al., 2007)

La Figure III-3 représente le modèle de référence de l'approche MDI. Elle montre les différents types de modèles qu'il est possible d'utiliser aux différents niveaux d'abstraction et les transformations de modèles qu'il faut réaliser. Dans le modèle de référence MDI la transformation d'un niveau plus haut à un niveau plus bas est définie comme une **transformation verticale**, alors que la transformation au même niveau entre deux entreprises est définie comme une **transformation horizontale**. MDI définit 4 niveaux de modèles inspirés de la démarche MDA :

[25] ESA signifie Enterprise Software Application

1. Le niveau métier pour modéliser l'entreprise d'un point de vue global en présentant son domaine, son métier, sa stratégie, etc. C'est le domaine de la description des problèmes d'interopérabilité avec des modèles d'entreprises. Compte tenu de sa complexité et de sa richesse ce niveau est décomposé en deux sous-niveaux (Bourey et al., 2007) :
   - Le niveau CIM-Haut (*Top CIM level*) : se focalise sur la description globale de l'entreprise, sans prendre en compte ses applications logicielles.
   - Le niveau CIM-Bas (*Bottom CIM level*) : se focalise sur la partie du CIM-Haut qui fera l'objet d'une informatisation.
2. Le niveau logique (*PIM level*) : c'est à ce niveau qu'il faut créer le modèle logique de l'architecture de la solution mais de manière indépendante de toute solution technologique.
3. Le niveau technique (*PSM level*) : ce niveau est le résultat de la projection de l'architecture du niveau PIM sur une technologie donnée.
4. Le niveau code (*Code level*) : ce niveau définit la solution exécutable.

Cette description pose les grandes lignes du modèle de référence de MDI. Cependant, elle n'est pas suffisante pour aborder concrètement un problème d'interopérabilité. Il faut affiner la description de chaque niveau et les modalités de passage d'un niveau à l'autre. Une première proposition détaillée des contenus des niveaux et des informations/connaissances nécessaires pour descendre d'un niveau à l'autre ont été fournies dans le projet ISTA3[26].

2.4. Cadres d'interopérabilité d'entreprise

Il y a eu plusieurs efforts dans différents contextes, pour catégoriser et caractériser le concept d'interopérabilité. Les résultats de ces efforts sont variables et dépendent du niveau de détail technique qu'ils contiennent et des entités qui les produisent (consortiums industriels, organismes de normalisation, chercheurs, administrations, etc.). La plupart de ces travaux proposent la mise en place de cadres spécifiquement dédiés à l'interopérabilité : *(Enterprise Interoperability Framework - EIF) (Chen et al., 2006)*, (*ATHENA Interoperability Framework - AIF*) (ATHENA01, 2007), (*European Interoperability Framework - EIF*) (EIF, 2004), (*Interoperability Development for Entreprise Application and Software - IDEAS*) (IDEAS, 2003), etc.

Pour positionner nos travaux, on présente le cadre d'interopérabilité *EIF*, développé dans le cadre du projet InterOp Noe (*Interoperability Research for Networked Entreprise Applications and software*, FP6 508011) qui nous semble être le plus couvrant via à vis des différentes dimensions de la notion

[26] http://research.petalslink.org/display/ista3/ISTA3+Overview

d'interopérabilité. Ce cadre de référence distingue trois dimensions d'interopérabilité : les approches, les niveaux et les barrières.

Chen et Daclin (2006) expliquent qu'une barrière signifie une incompatibilité, une disparité ou une hétérogénéité qui empêche le partage et l'échange d'informations. Ils distinguent trois types de barrières :

- La barrière conceptuelle est due aux différentes façons de nommer, structurer et représenter les concepts. Elle concerne les différences syntaxiques (i.e., formats), schématique (i.e., schémas et modèles) ou sémantiques.
- La barrière technologique concerne les différences entre les systèmes informatiques (architecture, infrastructure, protocole de communication, standards utilisés, etc.).
- La barrière organisationnelle est due à l'incompatibilité des différentes structures d'organisations, des responsabilités ou des droits accordés à chacun dans les entreprises.

Ces trois barrières dressent des problématiques séparées : la barrière conceptuelle est orientée vers les problèmes d'informations métiers, la barrière technique est orientée vers les problèmes des machines alors que la barrière organisationnelle est orientée vers les problèmes humains (Liu, 2011).

Le cadre d'interopérabilité EIF définit aussi quatre niveaux d'interopérabilité :

- Interopérabilité métier : ambitionne de faire collaborer des organisations en dépit des variétés de cultures et les différences en termes de mode de décision et de méthodes de travail.
- Interopérabilité des processus : cherche à faire travailler différents processus ensemble
- Interopérabilité des services : concerne l'identification, la composition et la réalisation des services pour mettre en relation divers services et applications.
- Interopérabilité des données : concerne la mise en relation de différents modèles ou bases de données d'applications diverses.

Ces différents niveaux d'interopérabilité aident à supprimer les barrières d'interopérabilité sus-présentées. Il y a aussi des relations entre les quatre niveaux d'interopérabilité. Par exemple, les exigences métiers de l'organisation peuvent être exprimées sous forme du processus qui orchestre des services qui véhiculent des données.

Enfin, le cadre d'interopérabilité EIF reprend les trois approches d'interopérabilités définies dans l'ISO-14258 (1998) :

- L'approche intégrée : les organisations partagent les mêmes modèles d'information.
- L'approche unifiée : les organisations partagent le même méta-modèle.
- L'approche fédérée : les organisations n'ont pas de modèle ou méta-modèle commun, mais utilisent une ontologie pour partager les données.

Comme (Berre et al., 2004; Liu, 2011) on considère que la définition des approches d'interopérabilité basée sur les modèles des informations échangées (model, méta-model, ni model ni méta-model) n'est pas pratique dans un projet réel. On adhère au point de vue de (Berre et al., 2004) qui distinguent les trois approches d'interopérabilité en se basant sur la topologie de l'architecture du système (Figure III-4) :

- L'approche intégrée assure l'interopérabilité par le partage d'un environnement d'exécution et des conventions de communication.
- L'approche unifiée assure l'interopérabilité par le partage des spécifications d'environnements, des méta-modèles et des concepts.
- L'approche fédérée établie et maintient la collaboration entre des services locaux autonomes, chacun d'eux exécute un processus métier local.

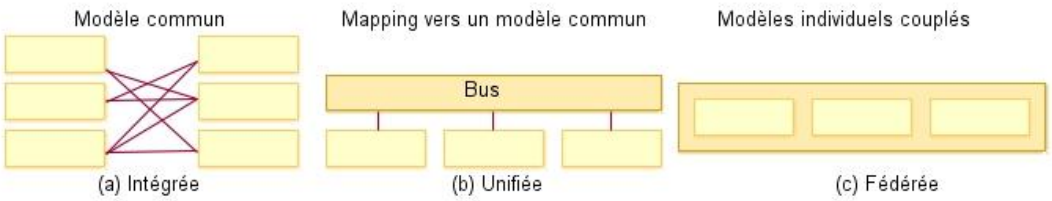


Figure III-4. Les approches d'interopérabilité intégrée, unifiée et fédérée proposées par (Berre et al., 2004)

On s'associe aussi aux précisions apportées par (Liu, 2011) qui stipule que le terme méta-modèle utilisé dans la définition de l'approche unifiée proposée par (Berre et al., 2004) désigne plutôt un modèle est non un méta-modèle au sens MDA (OMG04, 2003). C'est pourquoi, dans l'approche unifiée, chaque collaborateur peut avoir son propre modèle d'information et la collaboration est supportée par le mapping des modèles individuels vers un modèle commun. Dans cette approche tous les systèmes doivent s'enregistrer dans un Bus unifié pour faciliter la gestion et la gouvernance du système (Figure III-4). Liu (2011) propose 5 critères pour distinguer les différentes approches d'interopérabilité : (1) la portée du problème traité, (2) l'adaptabilité aux changements, (3) le résultat, (4) le nombre de connecteurs et (5) le nombre de traducteurs.

La Figure III-5 résume le positionnement de notre travail sur les trois dimensions du cadre d'interopérabilité EIF. Notre travail se focalise sur la conception des processus et des services aux niveaux d'interopérabilité processus et services afin de résoudre les problèmes d'interopérabilité

liés aux barrières conceptuelles et techniques en prenant en compte l'influence du choix de l'approche d'interopérabilité.

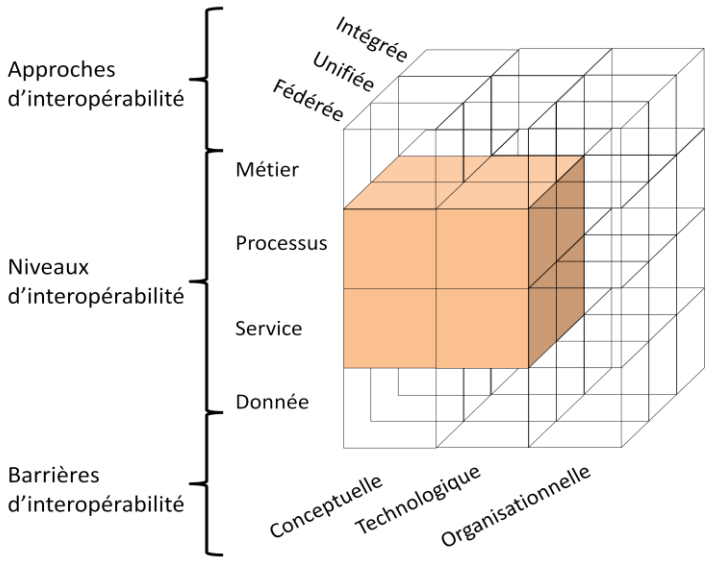


Figure III-5. Les trois dimension du cadre d'interopérabilité d'INTEROP (Chen et al., 2007)

3. SOA POUR RATIONALISER MDI

Pour tenter de répondre à la question de recherche suivante : *Comment combiner les approches MDI et SOA dans un contexte de collaboration pour améliorer l'interopérabilité et l'alignement stratégique du SI ?* on s'est basé sur les résultats du projet ATHENA, notamment le cadre interopérabilité AIF (ATHENA01, 2007). En effet, pour développer l'interopérabilité d'entreprise le projet ATHENA recommande une approche multidisciplinaire qui réunit trois disciplines de recherche : (i) la modélisation d'entreprise qui définit les exigences d'interopérabilité et supporte l'implémentation de la solution, (ii) Les architectures et les plateformes qui fournissent la base technologique de l'interopérabilité des systèmes et (iii) l'ontologie qui identifie la sémantique de l'interopérabilité dans l'entreprise. Dans nos travaux de thèse on prend en considération les deux disciplines de modélisation d'entreprise et architecture et plateformes.

La Figure III-6 représente une vue simpliste du cadre conceptuel fourni par le cadre AIF. Ce modèle de référence indique les artefacts exigés et fournis de deux entreprises collaboratives. Chaque entreprise est décrite par des modèles d'entreprise sur différents niveaux d'abstraction et différents points de vue (métier, processus, service et informations).

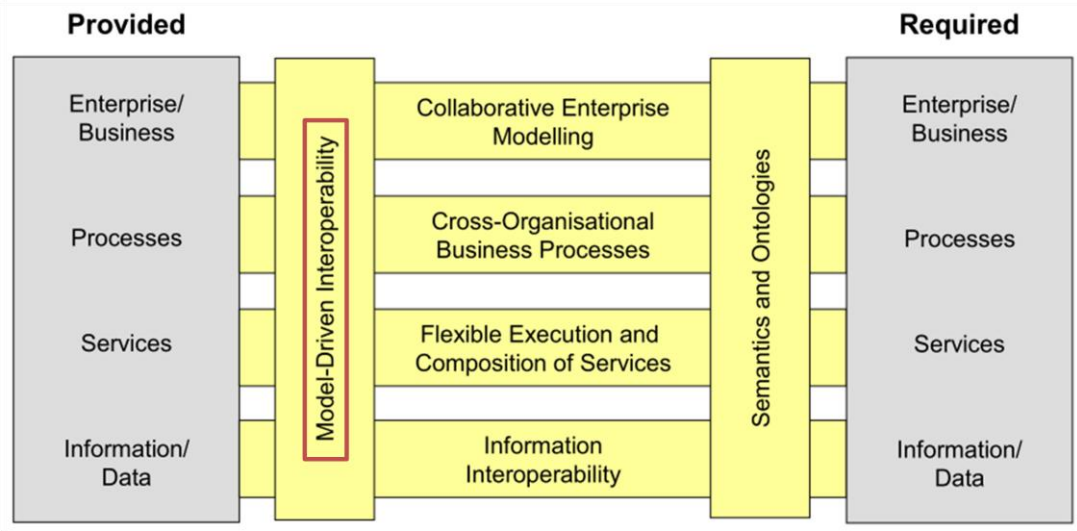


Figure III-6. Vue simplifiée du Framework conceptuel défini par AIF (ATHENA01, 2007)

L'idée est de combiner les approches AIF et MDI afin de construire une grille pour capter les bonnes pratiques à chaque niveau de MDI (CIM-Haut, CIM-Bas, PIM, PSM) pour chaque aspect de l'AIF (métier, processus, service et donnée). Ensuite, les disciplines de modélisation d'entreprise et d'architecture peuvent être étudiées au croisement des aspects d'interopérabilité et des niveaux de MDI.

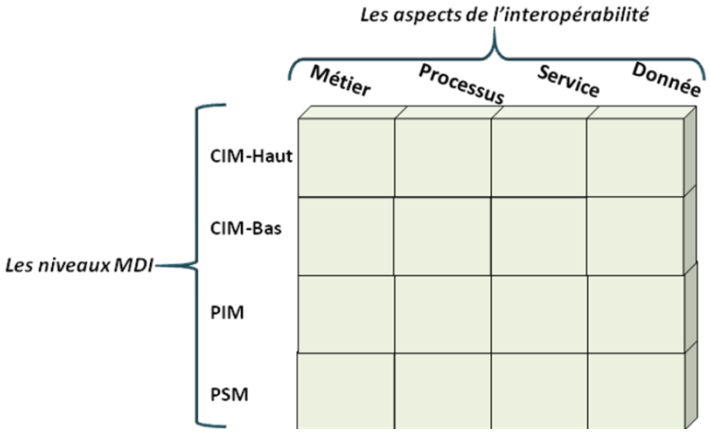


Figure III-7. La grille pour identifier les différents aspects de l'interopérabilité

La grille représentée dans la Figure III-7 offre une vue holistique de l'interopérabilité afin de permettre à chaque participant d'analyser et de comprendre ces besoins métiers et ces exigences techniques. Cette grille définie l'interopérabilité comme un ensemble de 16 sous-domaines d'interopérabilité. Ces sous-domaines facilitent la définition des domaines d'expertise entre les participants de la collaboration. Toutefois, la réalisation de tous les sous-domaines n'est pas un signe d'accomplissement de l'interopérabilité. Comme (Jochem, 2010) on considère qu'un participant est entièrement interopérable dans une collaboration s'il est capable de mettre en œuvre de nouvelles relations métier à faible coût.

On donne ici une vue d'ensemble de la grille. Puis plus tard, dans les chapitres IV et V les deux aspects processus et services seront déroulés sur les différents niveaux MDI pour illustrer les considérations à prendre en compte dans la méthode que l'on propose.

3.1. L'aspect Métier

L'interopérabilité à ce niveau est considérée comme la capacité d'une organisation de collaborer avec d'autres organisations externes qui ont des pratiques de travail, des législations, une culture ou des approches commerciales différentes (ATHENA03, 2005). Les collaborateurs doivent partager une même vision et chacun d'eux doit définir ses propres objectifs en les formalisant avec un langage comme *(Business Motivation Model - BMM)* (OMG05, 2008).
Les partenaires doivent aussi choisir un style d'architecture à mettre en œuvre (ex : SOA) et convenir à un périmètre de sa mise en place. Par exemple, pour l'architecture SOA le Gartner identifie trois styles de projets (Simon et al., 2006; Haan, 2008) :

(1) Concevoir, créer et exécuter de nouveaux artefacts SOA résultant d'une exécution complète de la démarche SOA.
(2) Assembler et déployer des applications composites et des processus. Ce type de projets concerne la combinaison de fonctionnalités existantes plutôt que la création de nouvelles fonctionnalités.
(3) Intégrer les données et la logique métier des applications.

L'effort d'engagement des participants dans chacun de ces trois styles de projet n'est pas le même ce qui affecte la mise en place de la collaboration. Enfin, c'est à ce niveau aussi que les participants doivent adopter une approche d'interopérabilité et un type de collaboration.

3.2. L'aspect Processus

Les modèles de processus contiennent ce qui il faut faire au niveau métier pour réaliser les objectifs métiers et la vision des collaborateurs. Dans un premier temps, il faut décrire globalement la collaboration en se concentrant sur les échanges entre les participants. Ensuite, il faut détailler la logique du flux des processus collaboratifs. Le formalisme BPMN 2.0 (OMG01, 2011) peut être utilisé pour spécifier les processus aux différents niveaux de l'approche MDI (cf. Chapitre IV).

3.3. L'aspect Service

Le principal objectif de cet aspect est l'identification et la conception de services réutilisables pour supporter les processus métiers. L'identification des services peut être dirigée soit par les objectifs décrits dans l'aspect métier (Han et al., 2009) soit par les processus spécifiés dans l'aspect processus. Dans un premier temps les services doivent être modélisés avec un formalisme comme

SoaML (OMG02, 2009) qui offre un haut niveau d'abstraction, puis il faut raffiner les services pour les rendre spécifiques à une plateforme donnée (ex : les web-services) (cf. Chapitre V).

Dans le cadre d'une approche d'interopérabilité unifiée les participants doivent se mettre d'accord sur une plateforme cible, alors que dans le cas d'une approche d'interopérabilités fédérée chacun des participants est libre d'adopter sa propre plateforme.

3.4. L'aspect Donnée

Les modèles de données doivent être étudiés en parallèle avec les modèles des processus et des services. BPMN ne précise pas comment spécifier les données utilisées par les processus. En revanche SoaML spécifie deux styles de message pour décrire les paramètres des opérations de services. Les services spécifiés avec SoaML doivent véhiculer des données qui respectent l'un de ces deux styles.

4. CONCLUSION

Dans ce chapitre, on présente une réflexion sur comment combiner les approches MDI et SOA pour supporter l'interopérabilité dans le cadre d'une collaboration. Pour cela, on a commencé par expliquer les différents types de collaboration. Puis, on positionne nos travaux dans le cadre d'interopérabilité *EIF* tout en précisant notre vision des approches d'interopérabilité. Et, enfin, on a proposé une grille de lecture à deux dimensions à partir des aspects du cadre AIF et des différents niveaux de l'approche MDI. Cette grille est formée de différents sous-domaines d'interopérabilité qu'il faut considérer pour permettre aux partenaires d'une collaboration d'analyser et de comprendre leurs besoins métiers et leurs exigences techniques.

L'intérêt de ce chapitre est de poser la base sur laquelle on a développé notre méthode de spécification d'une architecture orientée services dirigée par le métier. Par la suite, on utilise la grille que on propose pour expliquer en détail comment dépasser les barrières conceptuelle et technologique de l'interopérabilité sur les différents niveaux MDI des aspects processus (cf. Chapitre IV) et service (cf. Chapitre V).

CHAPITRE IV : Méthode et style pour l'identification et la spécification des processus collaboratifs dans une démarche MDI

1. INTRODUCTION DU CHAPITRE

Dans le chapitre précédent, on a expliqué comment les approches MDI et SOA peuvent être combinées dans un contexte de collaboration pour améliorer l'interopérabilité. On a aussi montré que dans une collaboration, il faut prendre en considération plusieurs aspects. Notamment, ceux qui sont proposés par le framework AIF : exigences métiers, processus, service et données. Dans ce chapitre, on zoome sur l'aspect processus du cadre AIF qui représente une partie de l'approche BPM. En effet, le but des processus collaboratifs est de contenir ce qui doit être fait pour atteindre les objectifs métiers globaux de la collaboration et les objectifs métiers individuels de chacun des participants.

L'objectif de ce chapitre est de proposer une méthode claire, complète et précise de modélisation des processus collaboratifs avec l'aide de BPMN 2.0. La méthode proposée suit une approche descendante pour découvrir les processus métiers de collaboration (cf. Chapitre II - 3.2.2). Elle est basée sur plusieurs modèles hiérarchiques et s'articule autour de 7 étapes et se déploie sur les différents niveaux de l'approche MDI. Cependant, la méthode concerne uniquement les transformations verticales de la démarche MDI (Figure IV-1).

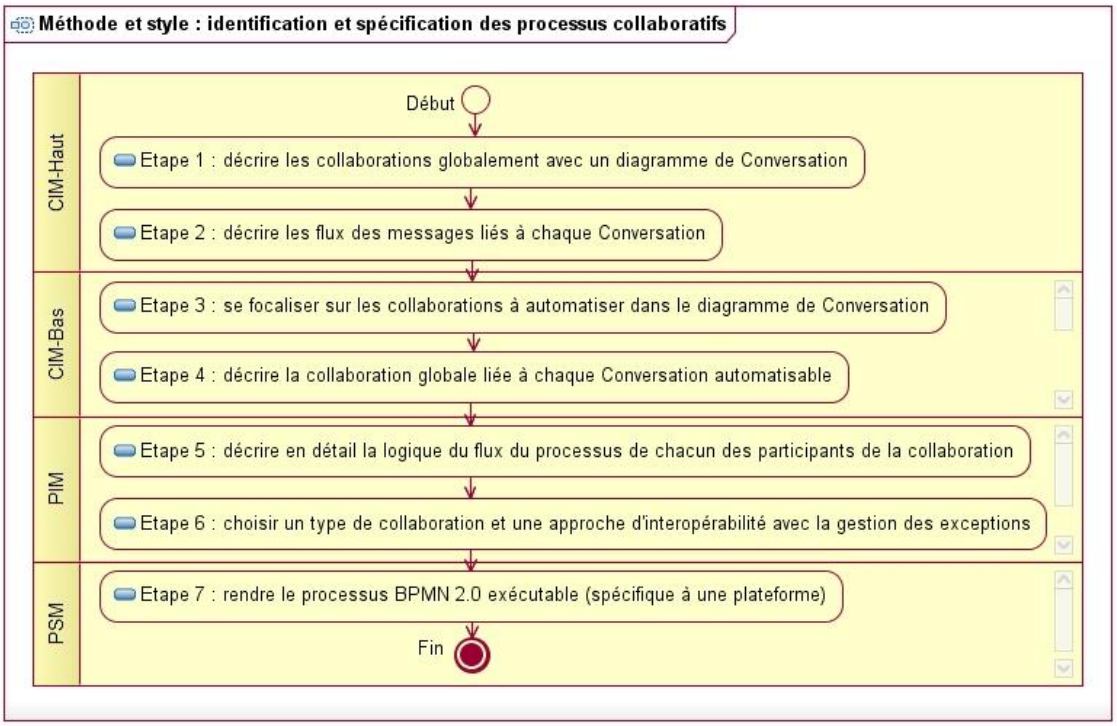


Figure IV-1. Les étapes de la méthode d'identification et de spécification des processus interopérables

La situation existante avant la collaboration est représentée avec les modèles As-Is ; alors que la nouvelle situation de collaboration est représentée avec les modèles To-Be. Jackson (1995) explique que les modèles As-Is décrivent les propriétés possédées par le système (propriétés

indicatives) alors que les modèles To-Be décrivent les propriétés que le nouveau système possédera (propriétés opératives). Dans le cas où les participants de la collaboration n'auraient jamais travaillé ensemble, il faut dresser une cartographie de l'existant (As-Is) et de la cible (To-Be). Sinon, si les participants collaborent déjà il faut prendre en considération les modèles existants. Dans le cas d'étude qu'on a choisi les participants n'ont jamais collaboré auparavant. Ainsi, il n y a pas de processus collaboratifs As-Is existants. On se limite uniquement aux modèles de processus To-Be.

2. ÉTAT DE L'ART

2.1. BPMN 2.0

2.1.1. Introduction

BPMN 2.0 est un langage de modélisation des processus dont la sémantique est supportée par un standard proposé par l'OMG (OMG01, 2011). Par conséquent, BPMN ne dépend pas d'un seul vendeur et sa compréhension n'est pas limitée aux utilisateurs d'un seul outil. De plus, l'OMG ne propose pas de méthodologie officielle pour BPMN. D'où la forte adoption de BPMN dans la communauté BPM (Recker, 2008).

La modélisation des processus métier avec une notation adaptée au métier comme BPMN contribue à faciliter l'alignement entre le métier et l'IT. En effet, BPMN fournit une notation intuitive et compréhensible à ses utilisateurs dans l'entreprise, depuis les analystes métier qui créent les ébauches initiales des processus jusqu'aux développeurs responsables de la mise en place de processus exécutables. BPMN supporte les approches de modélisations ascendantes et descendantes : soit l'analyste métier modélise le processus métier qui est ensuite raffiné par les développeurs pour le rendre exécutable, soit, le développeur crée un ou plusieurs processus de bas niveau qui sont ensuite combinés par des utilisateurs professionnels pour répondre rapidement aux exigences du marché.

On a recensé cinq améliorations significatives dans la spécification BPMN 2.0 :

1. Le méta-modèle BPMN représente l'apport majeur de la version 2.0. On discutera amplement le méta-modèle BPMN dans la section suivante.
2. Définition d'un mécanisme d'extensibilité des modèles et des éléments graphiques de BPMN. Les utilisateurs de BPMN peuvent étendre son méta-modèle en restant *BPMN-compliant*.
3. Définition de nouveaux types de diagrammes : diagramme de conversation et diagramme de chorographie.

4. Ajout des *Non-Interrupting Events* : quand un événement du type *Sub-Processes Interrupting* ou *Boundary Interrupting* est déclenché, il interrompt l'exécution normale de l'activité parente qui se termine immédiatement. Alors que le déclenchement d'un événement de type *Non-Interrupting* n'interrompt pas l'exécution de l'activité qui continue de s'exécuter normalement.
5. Extension de la définition des interactions humaines en alignant BPMN 2.0 avec la spécification BPEL4People (OASIS02, 2010).

La promesse de BPMN 2.0 est de fournir un langage de modélisation de processus de haut niveau exécutable. Cette tendance est confirmée par une étude récente (Chinosi et al., 2012) auprès d'un groupe d'experts et de professionnel BPMN. Ainsi, 37% des sondés utilisent BPMN pour modéliser des processus exécutables. Cependant, il faut noter qu'il y a une différence de taille entre BPMN 2.0 comme langage de modélisation exécutable et *Web Services Business Process Execution Language* (WS-BPEL) (OASIS03, 2007). En effet, WS-BPEL offre un niveau d'abstraction moins élevé que BPMN, il est utilisé pour la composition, l'orchestration et la coordination des web-services (Matjaz et al., 2006). En plus, contrairement à BPMN dans WS-BPEL toutes les taches sont implémentées avec des Web-services, décrites avec un WSDL et invoquées avec des messages SOAP.

2.1.2. Le méta-modèle BPMN

Le plus grand effort entre BPMN 1.2 et BPMN 2.0 s'est porté sur la définition du méta-modèle et non sur la notation. À tel point que l'OMG a changé l'acronyme de BPMN généralement connu sous le nom de *Business Process Modeling Notation* qui est devenue *Business Process Model* and *Notation*.

Un méta-modèle **(Définition 17- Méta-modèle**) est un modèle d'un langage de modélisation (Clark et al., 2008). Le méta-modèle doit remplir certaines caractéristiques : il doit incorporer l'ensemble des propriétés du langage modélisé (1), les décrire avec les syntaxes (concrète et abstraite) (2), préciser les contraintes du langage de modélisation (3) et la sémantique du langage de modélisation (4) et doit lui-même (i.e. le méta-modèle du langage de modélisation) être décrit avec un langage qui possède aussi son méta-modèle (5). Ce méta-modèle unique est appelé méta-méta-modèle dans l'architecture de méta-modélisation (Clark et al., 2008) (Figure IV-2).

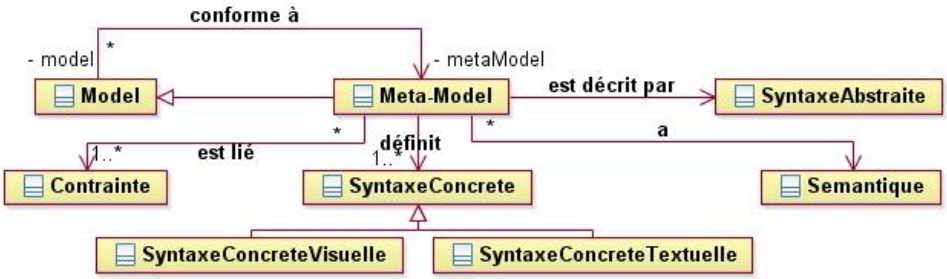


Figure IV-2. Description du méta-modèle

(Définition 18 - Syntaxe concrète) Les langages fournissent une notation pour faciliter la présentation et la construction de modèles ou de programmes dans un langage. Cette notation est connue sous le nom de syntaxe concrète. Principalement, il y a deux types de syntaxe concrète : les syntaxes textuelle et graphique (Clark et al., 2008).

Le méta-modèle BPMN 2.0 est disponible sous forme de deux syntaxes textuelles XML maintenues et synchronisées par l'OMG :

- *XML Metadata Interchange* (XMI) (OMG06, 2011) qui représente le *Complete Meta Object Facility* (CMOF) (OMG07, 2011) méta-modèle de BPMN proposé par l'OMG.
- *XML Schema Definition* (XSD) proposé par le W3C (W3C09, 2004).

Dans la spécification la syntaxe textuelle utilisée pour décrire les concepts du méta-modèle BPMN 2.0 est basée sur XSD, en effet XMI est moins populaire que XSD (Hille-Doering, 2010), notamment dans le développement des applications logicielles basées sur SOA.

Pour illustrer comment les syntaxes, la sémantique et les contraintes constituant le méta-modèle BPMN 2.0 sont utilisées, on a choisi de prendre pour exemple le concept BPMN *Task* (Figure IV-3 et Figure IV-4).

```
<xsd:element name="task" type="tTask" substitutionGroup="flowElement"/>
<xsd:complexType name="tTask">
        <xsd:complexContent>
            <xsd:extension base="tActivity"/>
        </xsd:complexContent>
 </xsd:complexType>
```

Figure IV-3. Syntaxe concrète textuelle (XSD) du concept *Task*

Figure IV-4. Syntaxe concrète graphique du concept *Task*

La syntaxe abstraite **(Définition 19 - Syntaxe abstraite)** décrit le vocabulaire des concepts fournis par un langage et la façon dont ils peuvent être combinés pour créer des modèles. Elle

permet de définir les concepts et leurs relations. Il est important de noter que la syntaxe abstraite s'intéresse uniquement à la structure des concepts du langage et à leurs relations sans prendre en compte ni leur représentation (textuelle ou graphique), ni leur signification (sémantique) (Clark et al., 2008). Cependant, généralement les noms des concepts dans la syntaxe abstraite sont porteurs de sémantique, ils fournissent des informations sur les concepts (Figure IV-5).

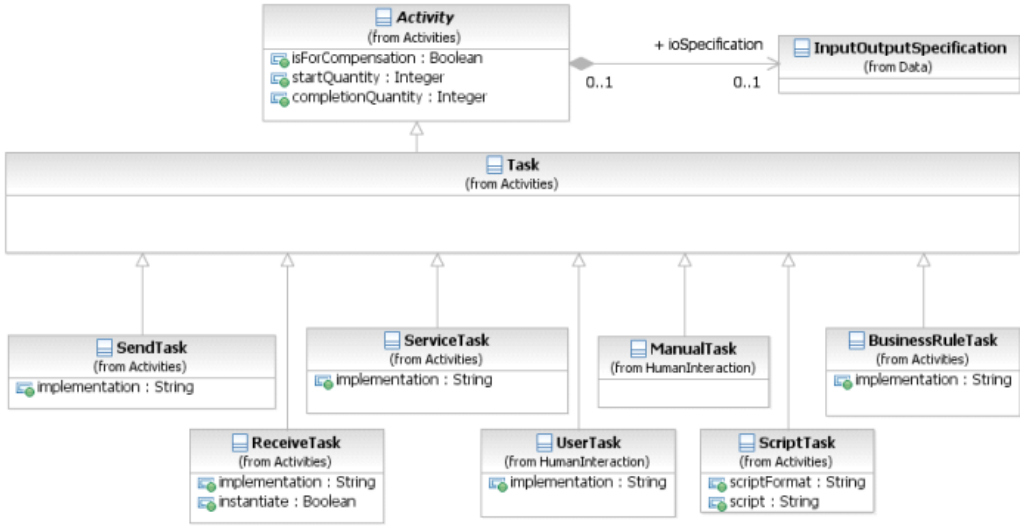


Figure IV-5. Syntaxe abstraite des relations du concept *Task*.

(Définition 20 - Sémantique) La syntaxe abstraite ne fournit pas a priori d'information sur la signification des concepts autre qu'à travers leurs noms. Plus d'informations sont nécessaires pour capter la sémantique du langage afin de clarifier ce qu'il représente et signifie. Par exemple, la sémantique apporte une réponse à la question « Qu'est-ce que c'est un *Task* ? ». BPMN 2.0 décrit un *Task* comme : « *An atomic Activity within a Process flow. A Task is used when the work in the Process cannot be broken down to a finer level of detail. Generally, an end-user and/or applications are used to perform the Task when it is executed* (OMG01, 2011).

(Définition 21 - Contraintes) Les contraintes précisent le rôle et la portée des concepts (la sémantique). Dans la spécification BPMN 2.0 les contraintes ne sont ni explicitement regroupées dans un ensemble de règles à respecter ni exprimées par un langage formel d'expression de contraintes comme *Object Constraint Language* (OCL) (OMG08, 2011). Elles sont dispersées dans les 508 pages de la spécification.

3. MÉTHODE

Les travaux sur la modélisation des processus sont variés, certains travaux se limitent à l'explication des notations des langages de modélisation des processus (Allweye, 2010) d'autres proposent une méthode de modélisation avec une notation limitée à la modélisation des processus de haut niveau (Doumeingts, 1985). D'autres encore se consacrent davantage à la

description de la modélisation de la logique du flux d'un processus et non à la collaboration de plusieurs processus (Silver, 2011). Notre méthode se focalise sur la modélisation, l'analyse et la conception de processus collaboratifs afin de supporter l'interopérabilité entre plusieurs partenaires. Elle part d'une description globale de la collaboration, pour ensuite aider à la description des processus de bout en bout de chaque participant afin de les rendre suffisamment détaillés pour être exécutés.

La méthode proposée se base sur une démarche descendante afin de favoriser une compréhension globale de la collaboration. Elle est basée sur le formalisme BPMN 2.0 qui intègre des notions comme les sous-processus qui favorisent les approches de modélisation descendantes. Cette méthode se limite à la conception des processus de collaboration et explique comment modéliser les échanges entre ces processus et leur logique interne en sept étapes ; chacune remplit un objectif précis et dénombre un ensemble de pratiques et de conseils. Dans un premier temps, elle aide à identifier les participants et les processus collaboratifs pour délimiter le périmètre de la collaboration. Puis, elle encadre la modélisation des diagrammes de collaboration globaux et explique comment les enrichir et les détailler. Enfin, elle met en évidence l'influence du choix du type de collaboration et de l'approche d'interopérabilité sur la conception des processus.

Pour répondre à la question, « comment construire les modèles de processus ? » la méthode est basée sur un style de modélisation hiérarchique, avec des diagrammes de processus de haut niveau qui sont ensuite enrichis à des niveaux plus bas de la démarche MDI. Un ensemble de diagrammes représentant les différentes vues du même modèle sémantique de collaboration est utilisé aux différents niveaux MDI. Le degré de détail des diagrammes de chaque niveau est adapté à une certaine audience. À chaque étape de la méthode, on distingue deux catégories de diagrammes : les diagrammes réutilisés aux différents niveaux de MDI (CIM-Haut, CIM-Bas, PIM, PSM) et ceux dont l'utilisation est limitée à un seul niveau MDI. Dans une démarche MDI la discutions entre les participants se situe (si elle existe) au niveau des modèles eux-mêmes. Les modèles CIM-Haut doivent être vus comme une base de discussion afin de produire les modèles CIM-Bas. Ces derniers servent par la suite à produire des modèles logiques spécifiques à une approche de collaboration et à un type d'interopérabilité donné au niveau PIM. Enfin, les modèles PIM sont enrichis avec les détails d'exécution spécifiques à une plateforme.

L'objectif final est d'aider à l'analyse et à la conception d'un système d'information collaboratif en utilisant des processus. Pour atteindre ce but la méthode proposée ici adresse des objectifs intermédiaires :

- Capturer la logique de la collaboration et des processus des participants à la collaboration.
- Décrire clairement les flux des processus : par exemple en intégrant les exceptions.

L'utilisation de toutes les notations proposées par BPMN 2.0 n'est pas nécessaire pour la modélisation des processus à tous les niveaux de la démarche MDI. La spécification BPMN 2.0 (OMG01, 2011) distingue trois niveaux de conformité de modélisation : (1) descriptive, (2) analytique et (3) exécutable. En s'inspirant des niveaux de conformités de BPMN 2.0, des travaux de (Chinosi, 2008; Chinosi et al., 2012) et de notre expérience. On propose trois palettes BPMN 2.0 qui contiennent des notations graphiques. Le nombre et le type des notations de chaque palette dépendent de l'audience à laquelle on s'adresse.

Les notations de la palette de niveau 1 (Figure IV-6) peuvent être utilisées dans la vue globale des niveaux CIM-Haut et CIM-Bas. Ils permettent de créer les diagrammes de *Conversations* qui décrivent la collaboration globale.

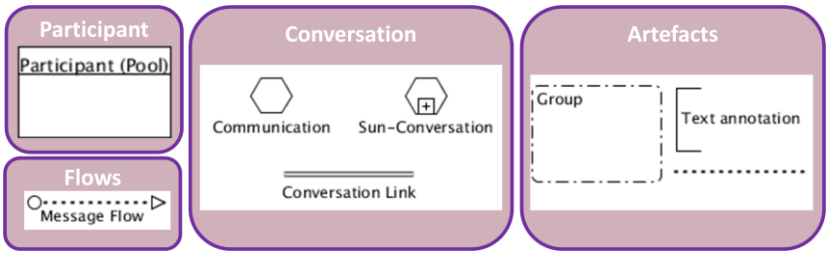


Figure IV-6. Palette BPMN 2.0 pour les diagrammes de Conversation (niveau 1)

La palette de niveau 2 (Figure IV-7) permet de décrire la vue détaillée des collaborations associées à chaque diagramme de *Conversation*. Cette palette est conforme au niveau de conformité descriptive proposé par BPMN 2.0.

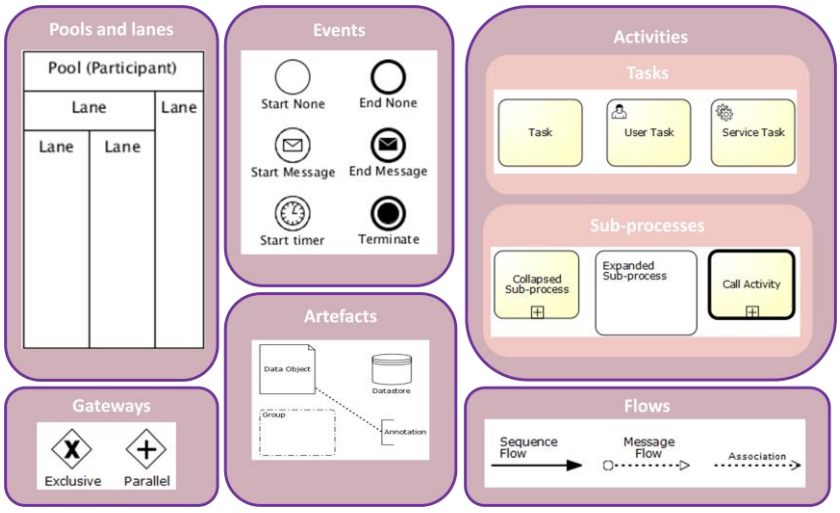


Figure IV-7. Palette BPMN pour la description des processus (niveau 2)

Quant à la palette de niveau 3 (Figure IV-8), elle complète celle du niveau 2 en ajoutant essentiellement des événements intermédiaires afin de permettre aux participants de faire une analyse approfondie de leurs processus. Cette palette est conforme au niveau de conformité analytique de BPMN 2.0.

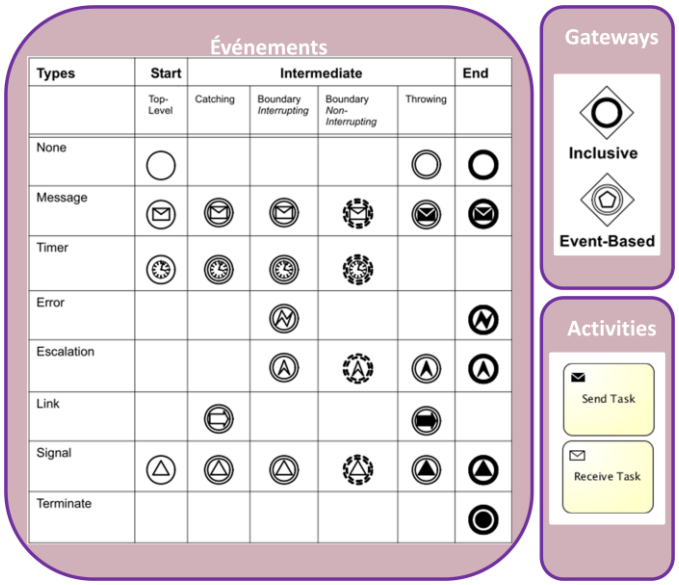


Figure IV-8. Palette BPMN pour l'analyse des processus (niveau 3)

Les palettes que l'on propose ne sont pas figées, elles doivent être adaptées à l'audience et à son niveau de familiarisation avec le langage BPMN 2.0. On illustre dans quelques exemples comment étendre la palette de niveau 3 sus-proposée avec d'autres notations BPMN 2.0. Enfin, pour modéliser des processus exécutables (i.e., PSM), il est nécessaire d'avoir une maîtrise de tous les éléments BPMN 2.0 (visuels ou pas)[27].

3.1. CIM-Haut

Les modèles CIM-Haut s'intéressent à la collaboration d'un point de vue global sans prendre en compte les problèmes d'informatisation. Ils représentent les exigences métiers du point de vue de l'utilisateur métier. Ces modèles se focalisent uniquement sur l'échange d'informations entre les participants sans rentrer dans le détail de la logique des flux des processus. Les objectifs principaux du CIM-Haut sont la compréhension du contexte et la spécification du besoin d'interopérabilité. Il s'agit donc de modéliser les partenaires qui sont concernés par la collaboration.

L'utilisation des diagrammes de *Conversation* BPMN 2.0 est pertinente. En fait, les *Pools* utilisés dans ce type de diagramme ne sont pas autorisés à contenir des processus (*Process*). Ainsi,

[27] La modélisation des processus non exécutables est réalisée avec l'outil Signavio

un diagramme de *Conversation* insiste sur les échanges entre les participants plutôt que sur la logique des flux des processus. À ce niveau, on propose d'adopter deux vues : une vue globale et une autre détaillée. La vue globale met en évidence les conversations entre les participants, alors que la vue détaillée étend chaque scénario (conversation), en représentant les échanges des principaux flux d'informations.

3.1.1. Étape (1) : décrire les collaborations globalement avec un diagramme de conversation

Dans cette étape on propose une représentation simplifiée de la collaboration avec les diagrammes de *Conversation* BPMN 2.0 (OMG01, 2011 p.124) afin de permettre aux différents partenaires d'identifier les processus de collaboration et de se mettre en accord sur le périmètre du projet et la portée de la collaboration. Le but de cette étape est de déterminer le périmètre de la future collaboration.

Dans la Figure IV-9 six participants et six conversations ont été identifiés. Dans chaque conversation, il faut distinguer les participants principaux des participants secondaires en utilisant les critères suivants : (1) la distinction entre le scénario de base (cas nominal) et les scénarios alternatifs, (2) la fréquence d'intervention de chaque participant dans une conversation et (3) le rôle joué par ce dernier. On propose d'utiliser deux couleurs pour distinguer les liens liant les participants aux conversations. Par exemple, la couleur rouge indique que le participant joue un rôle principal dans la conversation alors que la couleur jaune témoigne du rôle secondaire du participant dans la conversation. Ce type de modèle générique est d'abord destiné aux personnes qui détiennent le pouvoir politique dans les organisations partenaires.

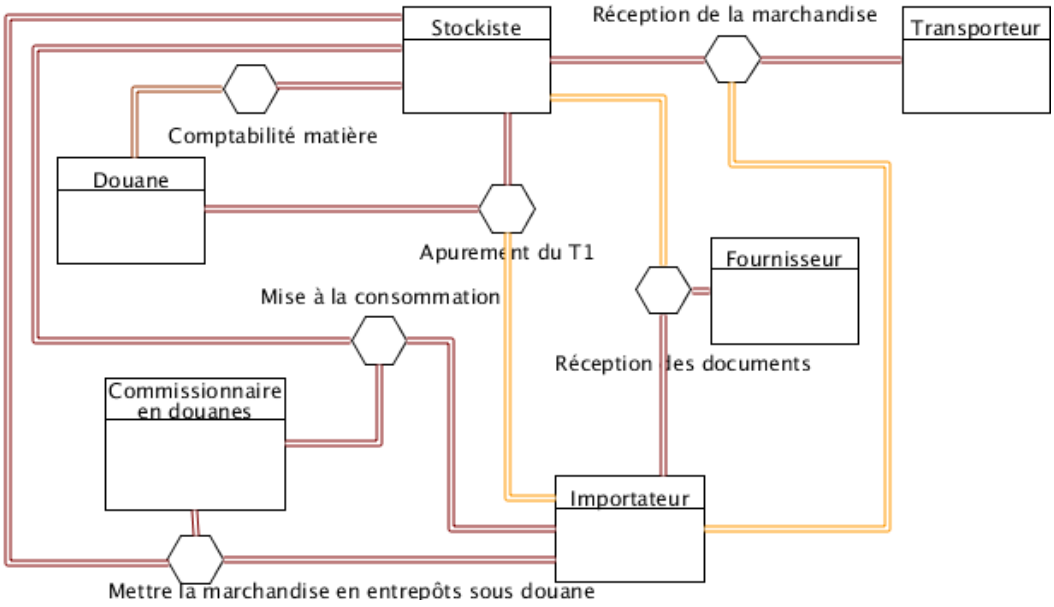


Figure IV-9. Diagramme de *Conversation* global entre les participants (CIM-Haut)

Dans la Figure IV-9 l'*Importateur* est considéré comme un participant secondaire dans les *Connections* « *Apurement du T1*[28] » et « *Réception de la marchandise* ». D'après les experts L'« *Apurement du T1* » est une simple formalité qui nécessite l'intervention de l'*Importateur* en cas de problème avec la douane. Quant à la *Conversation* « *La réception de la marchandise* » l'*Importateur* est informé uniquement à la fin du déchargement du camion.

3.1.2. Étape (2) : décrire les flux des messages liés à chaque *Conversation*

Le but de cette étape est de détailler les *Conversations* entre les participants en mettant en évidence les échanges de messages possibles entre ces derniers. Pour ce faire, on propose d'utiliser les diagrammes de *Collaboration BPMN* qui décrivent la manière dont le processus s'intègre dans l'environnement global.

On a choisi de détailler les deux conversations : « *Réception de la marchandise* » et « *Apurement du T1* » (Commission-européenne, 2010). La *Conversation* « *Réception de la marchandise* » est détaillée avec trois *Participants* et deux *messages Flows*. Lors de l'arrivée du camion chez le *Stockiste*, il est déchargé et le *Stockiste* dénombre les colis reçus et notifie l'*Importateur*. Dans le scénario de base, le nombre de colis comptés par le *Stockiste* doit correspondre au nombre de colis indiqué dans les documents de la commande qui sont envoyés par le *Fournisseur*. Sinon, si le nombre de colis reçus est différent de celui mentionné dans les documents de la commande, alors le *Stockiste* doit informer l'*Importateur* qui doit se rapprocher de la *Douane* pour résoudre ce problème.

Figure IV-10. Diagramme de Collaboration global « *Réception de la marchandise* » CIM-Bas

L'apurement du T1 permet de notifier à la *Douane* l'arrivée d'une marchandise importée à sa destination. Dans le cas nominal, le *Stockiste* apure le T1 via le système NSTI[29] (Nouveau Système de Transit Informatisé) de la douane sans problème et notifie l'*Importateur* de l'apurement. Dans ce scénario. La *Conversation* « *Apurement du T1* » est aussi détaillée avec trois *Participants* et deux *Messages Flows*. (Figure IV-11)

[28] La procédure T1 permet la circulation de marchandises non communautaires en suspension des mesures qui leur sont normalement applicables à l'importation

[29] https://pro.douane.gouv.fr/prodouane.asp

Figure IV-11. Diagramme de Collaboration global « *Apurement du T1* » CIM-Bas

3.2. CIM-Bas

Les modèles CIM-Bas s'intéressent à la partie publique de la collaboration d'un point de vue global ; ils représentent les exigences métiers des applications d'entreprise. Comme dans le CIM-Haut, on propose d'utiliser deux vues (globale et détaillée). La vue globale offre une image complète des besoins métiers à automatiser. Elle est représentée par des diagrammes de *Conversations*, alors que la vue détaillée utilise des diagrammes de *Collaboration* dérivés des diagrammes de *Collaboration* de la vue détaillée du CIM-Haut et du diagramme de *Conversation* global du niveau CIM-Bas.

À ce niveau deux aspects doivent être pris en compte : (1) se focaliser sur les *Conversations* automatisé qui font intervenir les systèmes d'information des participants ; (2) sélectionner les participants qui seront amenés à construire ou à modifier leurs systèmes dans le cadre de cette collaboration. Enfin, le passage à l'étape suivante (i.e. niveau PIM) nécessite une description des attentes des participants et des capacités de leurs systèmes d'information ou des moyens qu'ils mettront en œuvre pour la mise en place de la collaboration au niveau IT. Cet exercice est nécessaire afin de préciser les engagements de chacun des participants (cf. Introduction - 4).

3.2.1. Étape (3) : se focaliser sur les collaborations à automatiser dans le diagramme de conversation

Le but de cette étape est de mettre en évidence uniquement les *Conversations* automatisables en se basant sur les informations fournies par les analystes métier. Dans cet exemple les décisions/informations additionnelles suivantes ont été fournies. La réception de la marchandise ne sera pas automatisée. Quand le camion du *Transporteur* arrive chez le *Stockiste* les colis sont déchargés et comptés manuellement. Il n'y a pas besoin de faire intervenir le SI du *Transporteur*. Par la suite, le *Stockiste* fait l'apurement du T1 via l'application de transit douanière[30] NSTI de la douane et notifie le résultat à l'*Importateur*. En effet, le système NSTI est utilisé indirectement dans la collaboration via le *Stockiste*. Par conséquent, les participants *Douane* et *Transporteur* ne sont pas

[30] https://pro.douane.gouv.fr/transit/natweb.nw/gi_dti/LIBGIDTI.E0

retenus dans le diagramme de *Conversation*. Voilà pourquoi, il a été décidé de ne pas retenir la *Conversation* « *Réception de la marchandise* » (Figure IV-12).

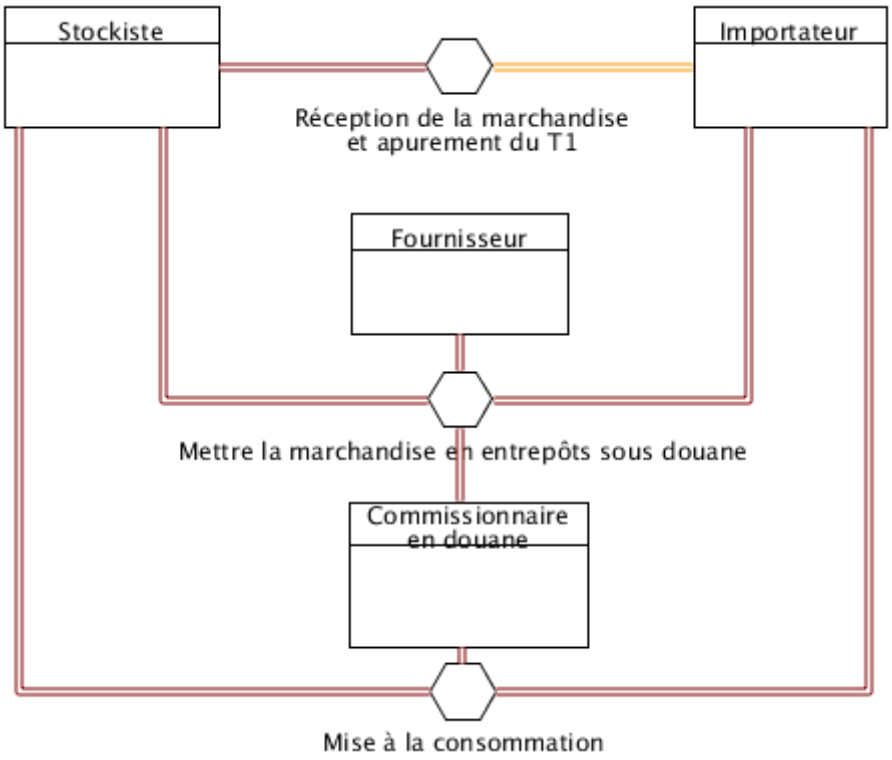


Figure IV-12. Diagramme de *Conversation* global des participants qui modifieront leurs systèmes d'information

La comptabilité matière est une photographie des mouvements de stock de l'ESD. Elle est réalisée par le *Stockiste* et doit être envoyée par mail à la *Douane*. La Conversation « *Comptabilité matière* » ne figure pas dans la Figure IV-12 car le participant *Douane* ne modifie pas son SI. La mise en place de cette conversation ne nécessite pas la collaboration entre plusieurs participants donc elle est supprimée.

3.2.2. Étape (4) : décrire la *Collaboration* globale liée à chaque *Conversation* automatisable

Les diagrammes de collaboration de cette vue détaillée doivent décrire globalement les flux métiers des processus de bout en bout. Pour produire ces modèles CIM-Bas, on suggère de prendre en considération les recommandations suivantes : (1) se focaliser sur les interactions qui seront l'objet d'une informatisation indépendamment des solutions d'implémentation, (2) intégrer plus de détails métiers dans le flux de la logique du processus, (3) se limiter aux éléments BPMN 2.0 de la palette descriptive.

Les diagrammes de collaboration doivent expliquer comment les processus de chaque collaborateur commencent et se terminent sans révéler le détail interne de la logique de ces

processus. Chacun des participants doit commencer par énumérer les activités de collaboration de son processus, puis les présenter comme une séquence d'activités BPMN. Il est fortement recommandé d'utiliser les sous-processus pour cacher les détails du processus qui seront ajoutés à des niveaux inférieurs de MDI. Ensuite, il faut créer une vue sur le processus qui regroupe uniquement les activités qui seront automatisées. Enfin, on estime que les éléments BPMN 2.0 de la palette niveau 2 sont suffisants pour décrire ces diagrammes de collaboration.

Par exemple les processus de collaboration Figure IV-13 détaillent la *Conversation* « *Réception de la marchandise et apurement du T1* » de la Figure IV-12. La collaboration est déclenchée dès l'arrivée de la marchandise chez le stockiste qui la réceptionne, puis procède à l'apurement du document T1 via le système de la douane. Ensuite, il notifie l'*Importateur* du résultat de l'apurement du T1 pour l'informer des différentes opérations d'apurement du T1 ou d'intervenir auprès de la douane en cas de problème.

Le diagramme de collaboration globale de la Figure IV-13 représente les activités identifiées par les participants dans la collaboration « *Réception de la marchandise et apurement du T1* », alors que la Figure IV-14 représente une vue restreinte aux activités automatisables de cette collaboration. L'utilisation des vues permet de se restreindre aux activités automatisables ce qui affecte le périmètre de la collaboration. C'est le diagramme de collaboration de la Figure IV-14 qui sera transmis aux niveaux inférieurs (i.e. PIM) de la démarche MDI.

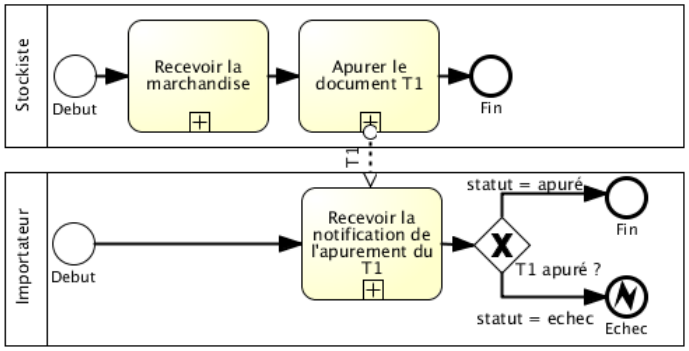


Figure IV-13. Collaboration globale: « *Réception de la marchandise et apurement du T1* »

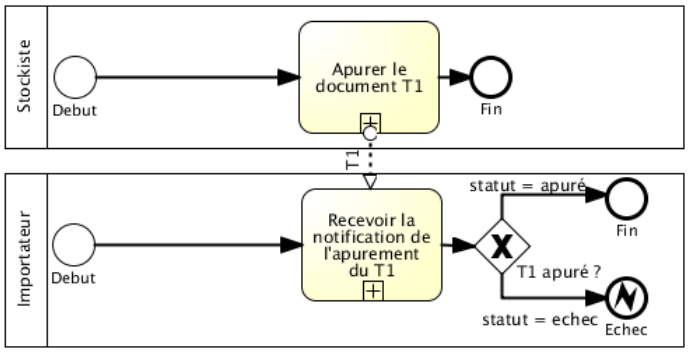


Figure IV-14. Collaboration globale automatisable: « *Réception de la marchandise et apurement du T1* »

3.3. PIM

Les modèles des processus à ce niveau doivent fournir une vue suffisamment détaillée de chaque processus, ils requièrent donc l'utilisation d'une notation BPMN plus riche. Donc, on recommande l'utilisation des éléments de la palette BPMN de niveau 3. L'intérêt de chaque participant doit être porté sur la logique du flux et des règles de décision de son processus. Des informations sur les données d'entrée et de sortie des activités du processus peuvent être rajoutées aux diagrammes pour les enrichir. Cependant, les processus ne doivent pas contenir les détails liés à une plateforme spécifique. À ce stade les participants doivent impérativement choisir un type de collaboration (cf. Chapitre III - 2.1) et une approche d'interopérabilité cibles (cf. Chapitre III - 2.2) afin d'identifier qui contrôle et exécute quels processus.

Les analystes métier doivent recueillir les informations nécessaires à la modélisation détaillée des processus. À partir de notre expérience et du livre de Bruce Silver (2011) on cite ci-après un ensemble de règles de bonne pratique sous forme de questions. Ces questions aiderent à transcrire les exigences des participants dans les modèles de processus.

1. Quel est le but ou l'objectif du processus ?
2. Qui sera responsable de l'exécution du processus ?
3. Quel est l'événement qui déclenche le processus ? Comment commence-t-il ? est-ce qu'il y a plusieurs façons possibles pour démarrer le processus ?
4. Comment savoir que le processus est terminé ? Est-ce qu'il y a plusieurs états finaux possibles ? Le processus se termine-t-il avec un ou plusieurs succès, des abondants ou des échecs ?
5. Si le flux d'exécution du processus passe d'une activité X à une activité Y, qui est censé être au courant de ce changement ? Sinon s'il n'est pas sensé aller à l'activité Y, où doit-il aller ?
6. Comment savoir que le processus est terminé ? Est-ce qu'il finit toujours de la même manière ? Est-ce qu'il y a des exceptions ? Est-ce qu'il y a des règles qui gouvernent le processus ?

3.3.1. Étape (5) : décrire en détail la logique du flux du processus de chacun des participants de la collaboration

Dans cette étape chacun des partenaires doit se concentrer sur la modélisation de la logique du flux de son propre processus. Il doit développer les sous-processus découverts dans le niveau CIM-Bas pour prendre en compte les différents cas possibles et les exceptions du processus. Un *black-box Pool* qui ne contient pas d'éléments de flux *Flow Node* peut être utilisé pour représenter

les autres participants. Prenons à titre d'exemple la modélisation du processus du *Stockiste*. Pour représenter l'*Importateur*, un *Pool* vide *black-box Pool* est utilisé pour indiquer que le *Stockiste* considère que la logique interne du processus de l'*Importateur* est invisible pour lui.

La Figure IV-15 représente le diagramme de collaboration utilisé par le *Stockiste*. Il est à noter que le processus de l'*Importateur* est représenté par un *Pool* vide. De plus, la flèche *Message flow* qui relie les deux P*ools* attache la limite de la *Pool* vide qui représente l'*Importateur* directement à l'activité « *Notifier l'Importateur* » de la *Pool* du *Stockiste*.

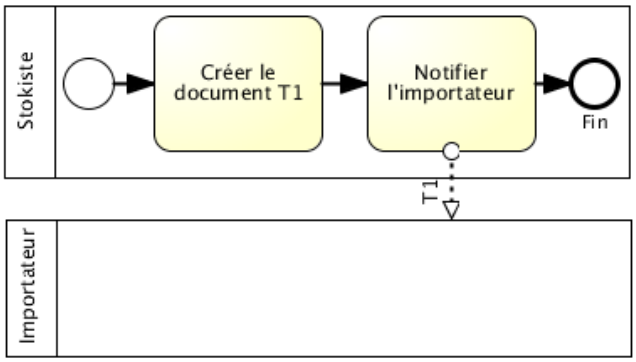


Figure IV-15. Processus du *Stockiste*

De la même manière, la Figure IV-16 représente le diagramme du processus de l'*Importateur*. Un modèle qui représente un vrai processus doit préciser ce qu'il faut faire si une exception est déclenchée. BPMN permet de visualiser ce comportement dans le diagramme lui-même.

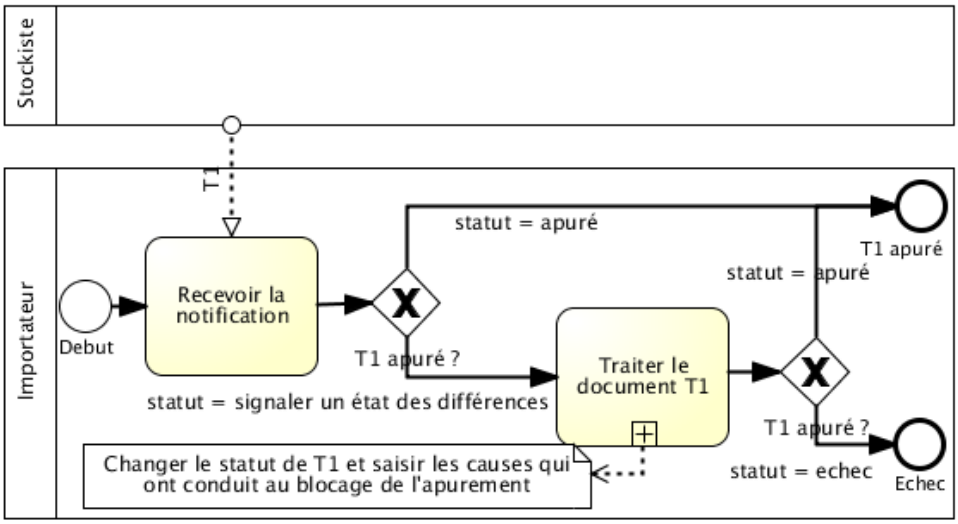


Figure IV-16. Processus de l'*Importateur*

La Figure IV-17 représente le sous-processus « *Traiter le document T1* ». L'employé se charge de résoudre le problème d'apurement du T1 en se rapprochant de la douane, alors que le manager vérifie les opérations d'apurement du T1.

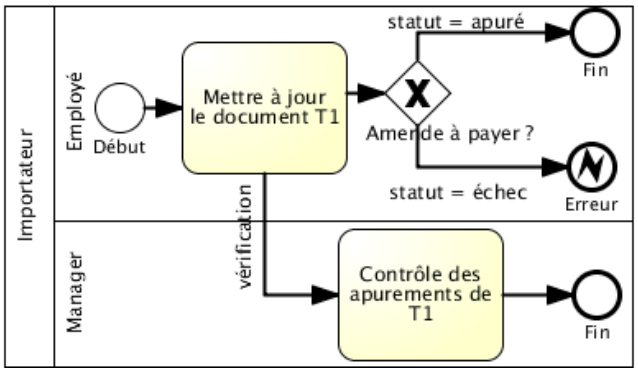


Figure IV-17. Sous-processus Traiter le document T1

3.3.2. Étape (6): choisir un type de collaboration et une approche d'interopérabilité avec la gestion des exceptions

Le but de cette étape est de prendre en compte le type de collaboration et de l'approche d'interopérabilité cible dans la modélisation des processus indépendamment d'une plateforme d'exécution spécifique.

Modélisation de la coordination dans le cadre d'une approche d'interopérabilité unifiée ou d'une vue centralisée d'une coopération dans le cadre d'une approche d'interopérabilité fédérée

Dans une coordination au sein d'une approche d'interopérabilité unifiée un seul participant contrôle l'exécution de tous les processus qui s'exécutent sur le même Moteur d'Exécution de Processus (MEP). Le MEP agit comme un médiateur entre les participants qui sont représentés par des *Lanes* dans le processus.

Dans la modélisation du processus, il faut préciser le type des activités et éviter d'utiliser des activités abstraites. Par exemple, dans la Figure IV-18 l'activité du *Stockiste* est de type *User Task*. Une erreur commune consiste à utiliser les notations BPMN *Send Task* et *Receive Task* pour transmettre un traitement entre deux *Lanes* du même processus. Dans la majorité des cas, il ne faut pas modéliser explicitement cette notification qui est implicitement représentée par le *Sequence Flow*. Il est à noter que lors de l'exécution du processus par le MEP le *Sequence Flow* ne transmet pas uniquement la notification mais aussi les instances des données valables dans le processus à ce point. Plusieurs BPMS comme BizAgi Xpress[31], Oracle BPM Suite[32], Bonita Open Solution[33] sont capables d'exécuter ce type de collaboration (Figure IV-18).

[31] http://www.bizagi.com/index.php?option=com_content&view=article&id=19&Itemid=100

[32] http://www.oracle.com/technetwork/middleware/soasuite/learnmore/vmsoa-172279.html

[33] http://www.bonitasoft.com/

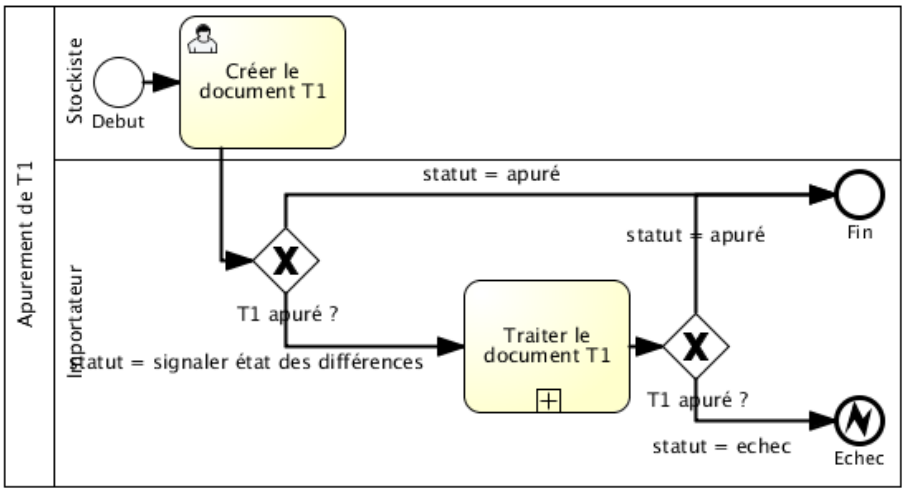


Figure IV-18. Processus « *Apurement document T1* » dans une coordination dans le cadre d'une approche d'interopérabilité unifiée

Dans une vue centralisée de la coopération avec une approche d'interopérabilité fédérée, tous les participants déploient le même modèle de processus sur leurs MEP. Cependant, chaque participant est responsable de l'exécution de ses propres activités. Les MEP négocient entre eux pour déterminer quel collaborateur a la responsabilité d'exécuter quelle activité. Ainsi, l'exécution de chaque activité est contrôlée uniquement par son exécutant. Dans ce cas l'*Importateur* et le *Stockiste* doivent disposer du même processus déployé sur leurs MEP respectives. Le MEP de l'*Importateur* exécute l'activité « *Créer le document T1* » et celui du *Stockiste* exécute le sous-processus « *Traiter le document T1*». Actuellement, il n'y a pas de solution capable d'implémenter techniquement la vue centralisée de la coopération (Figure IV-19).

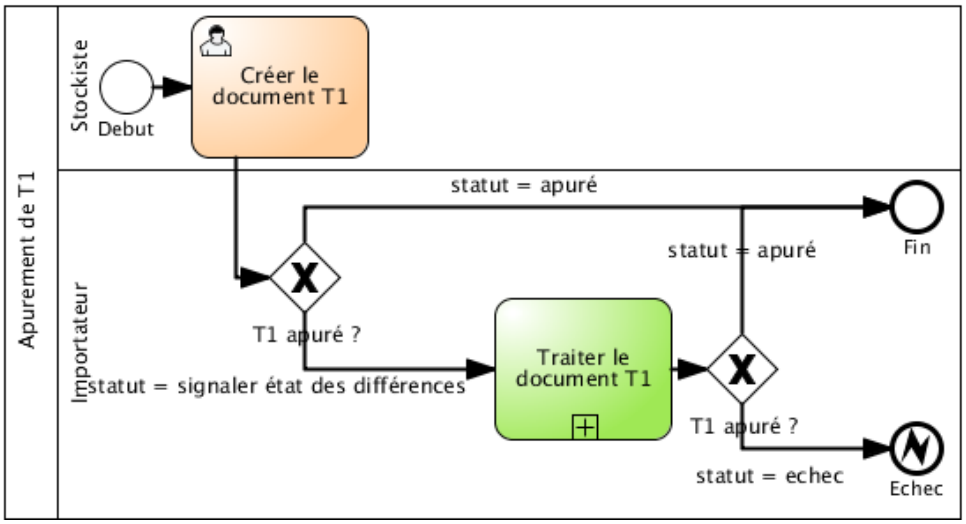


Figure IV-19. Processus « *Apurement document T1* » dans une vue centralisée de la coopération dans le cadre d'une approche d'interopérabilité fédérée

Modélisation d'une coopération avec une vue distribuée dans le cadre d'une approche d'interopérabilité fédérée

Dans une coopération avec une vue distribuée, chaque participant exécute son propre processus qui contient ses propres activités localement. Les processus coopératifs communiquent par échange de messages. La Figure IV-20 montre la coopération distribuée entre les processus du *Stockiste* et de l'*Importateur*.

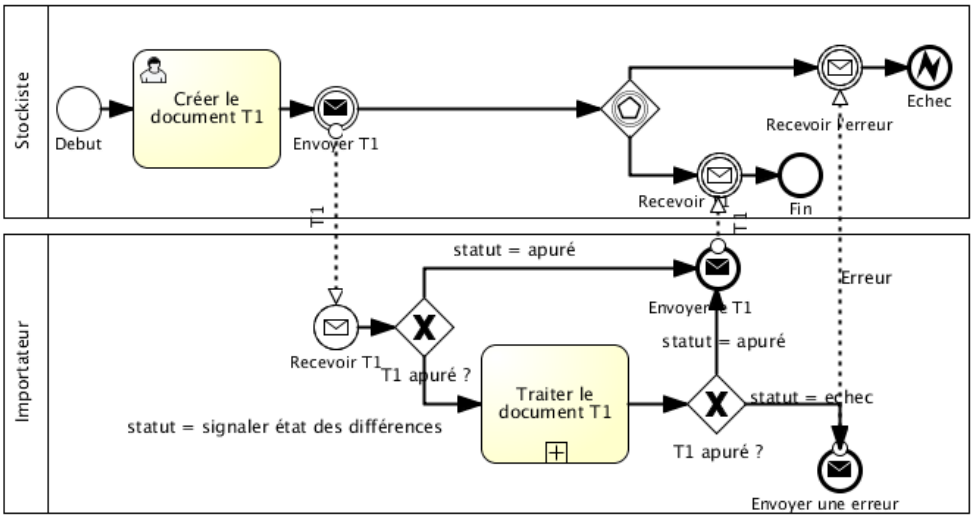


Figure IV-20. La modélisation de l'apurement de T1 dans une coopération avec une vue distribuée dans le cadre d'une approche d'interopérabilité fédérée

Les participants ont choisi le style d'interaction asynchrone représenté dans le diagramme avec des événements du type *Throw* et *Catch Message Events*. Ce type d'interactions peut aussi être représenté par les taches *Send et Receive Tasks*. En fait, résoudre le problème de l'apurement du document T1 prend du temps et ne peut être réalisé dans quelques secondes (ou millisecondes) car cette opération nécessite une intervention humaine. Au contraire, l'élément BPMN *Service Task* est plus adapté pour représenter les interactions du type synchrone. Il est important de noter que la combinaison de l'*Event Gateway* et de *l'Intermediate Message Event* permet au *Stockiste* de se mettre en attente des messages envoyés par l'*Importateur*. En fait, l'*Event Gateway* a le même comportement qu'un *XOR Gateway* mais le choix est basé sur l'événement qui se déclenche en premier et non sur les données.

Modélisation d'une coopération avec une vue distribuée dans le cadre d'une approche d'interopérabilité unifiée

L'approche d'interopérabilité unifiée assure l'interopérabilité par l'utilisation d'un environnement de spécifications et des concepts communs. Chacun des participants a ses propres modèles d'information et la coopération est supportée par la transformation des modèles de données individuels de chaque participant vers un modèle commun. On propose de créer un

nouveau processus de médiation (i.e., médiateur) pour assurer cette transformation, le médiateur peut être implémenté par un ESB (cf. Chapitre I - 4.2.1). Dans cette coopération, le médiateur prend le contrôle des partenaires impliqués et orchestre l'exécution des différentes opérations. Les participants n'ont pas besoin de savoir qu'ils participent à un processus métier d'un niveau plus élevé. Seul le médiateur est conscient de cet objectif. De plus, le médiateur lui-même peut être exécuté par un nouveau participant qui n'était pas impliqué dans la coopération initiale. Ce participant fournit la plateforme uniforme qui supporte la médiation entre les processus (Figure IV-21).

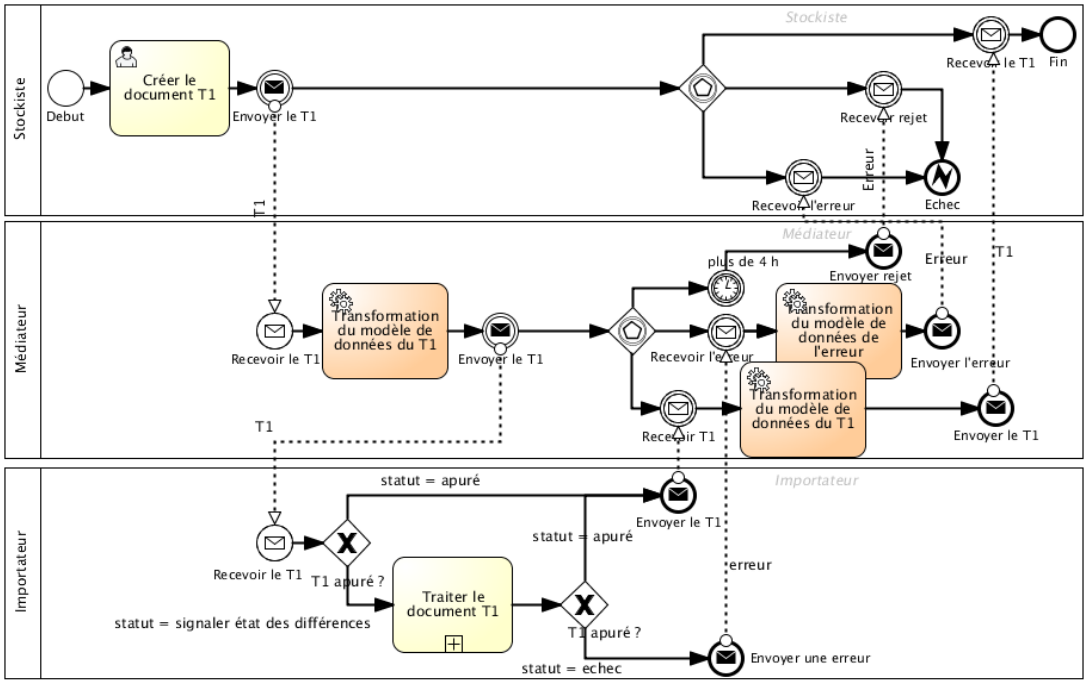


Figure IV-21. La modélisation de l'apurement du T1 dans une coopération avec une vue distribuée dans le cadre d'une approche d'interopérabilité unifiée

On suppose que le *Stockiste* et l'*Importateur* utilisent des modèles de données différents, des formats de données différents (ex : XML, CVS) et des versions du protocole SOAP différentes (SOAP 1.1 par le *Stockiste* et SOAP 1.2 pour l'*Importateur*). Si le *Médiateur* est implémenté avec un ESB, il permettra au *Stockiste* et à l'*Importateur* de communiquer entre eux malgré toutes ces différences. Cependant, seule la transformation des modèles de donnés est représentée dans le *Médiateur* (Figure IV-21).

Dans la Figure IV-20 si l'*Importateur* ne répond pas au *Stockiste*, l'instance du processus de ce dernier reste bloquée (en attente sans arrêt). Une amélioration possible de cette modélisation est représentée dans Figure IV-21. La combinaison du *Event Gateway* et du *Timer Intermediate Event* permet de résoudre ce problème. Dans la Figure IV-21 le *Stockiste* envoie un message et se met en

attente de la réponse de l'*Importateur*, s'il ne reçoit pas la réponse de l'*Importateur* au bout de 4 heures, alors il se termine avec un échec.

La Figure IV-22 met en évidence une autre capacité du *Médiateur* liée à son implémentation par l'ESB. Cette capacité découle du Pattern file d'attente asynchrone de l'ESB (cf. IV.2.a.i.). Dans ce cas le *Médiateur* agit comme un tampon intermédiaire *intermediary buffer* entre les messages envoyés par les processus de la collaboration. Par exemple, si le médiateur ne reçoit pas une réponse du transporteur dans un délai d'une heure il le relance avec un nouveau message. Pour ajouter cette nouvelle contrainte, on utilise le concept *Event Sub-Process* pour améliorer la compréhension du modèle (Figure IV-23 et Figure IV-24).

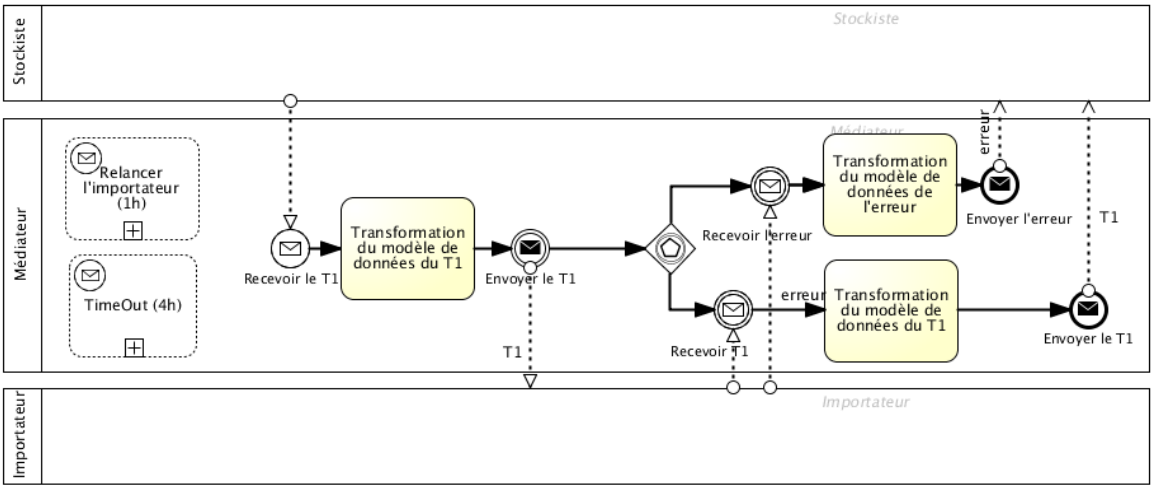


Figure IV-22. La modélisation de l'apurement du T1 dans une coopération avec une vue distribuée dans le cadre d'une approche d'interopérabilité unifiée, utilisation du *Event Sub-Process*

Figure IV-23. Sous-processus relancer l'*Importateur* (1h)

Figure IV-24. Sous-processus TimeOut (4h)

Contrairement au *Embedded Sub-Process* (sous-processus normal) un *Event Sub-Process* ne fait pas partie du flux normal du processus parent. Par conséquence, il n'a pas de *Sequence Flows* entrant ou sortant. De plus, le déclenchement du *Non-Interrupting Event Sub-Process* n'interrompt pas l'exécution du processus parent qui continue de s'exécuter en parallèle. Le processus parent se termine uniquement quand son chemin d'exécution normale arrive à terme et que le sous-processus est terminé aussi. Par contre, si le *Non-Interrupting Event Sub-Process* se termine sur une

erreur *(Error End Event)* ou une escalade *(Escalation End Event)* il entrave la continuation normale du flux du processus parent (Silver, 2011). C'est la raison pour laquelle le sous-processus « *Time Out (4h)* » se termine avec un *End Erreur Event*.

Enfin, la Figure IV-25 résume les différentes relations entre les types de collaboration et les approches d'interopérabilité qu'il faut prendre en compte dans le niveau PIM de la démarche MDI. Comme on s'intéresse uniquement aux problèmes d'interopérabilité entre plusieurs systèmes, on n'a pas abordé l'approche d'interopérabilité intégrée et la collaboration intégrée qui font référence à un seul système monolithique dans des environnements d'exécution partagés. Il est à noter que les couleurs des flèches qui relient les approches d'interopérabilité aux types de collaboration représentent les différents types de modèles de processus. Par exemple, le même modèle de processus peut être utilisé dans une coordination dans le cadre d'une approche d'interopérabilité unifiée ou dans une vue centralisée de la coopération dans le cadre d'une approche d'interopérabilité fédérée (les deux flèches rouges).

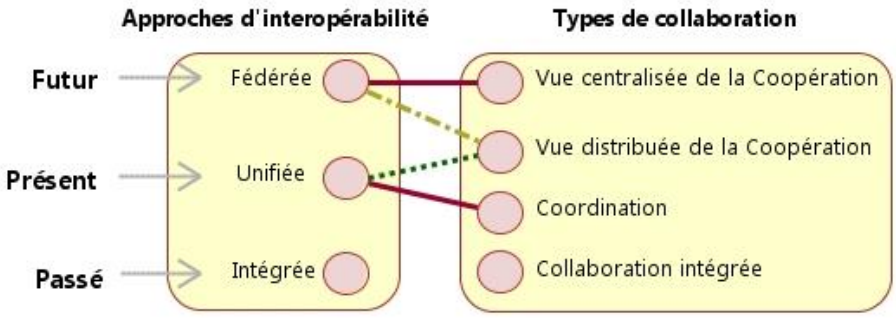


Figure IV-25. Relation entre les approches d'interopérabilité et les types de collaboration

3.4. PSM

Plusieurs solutions possibles existent pour la création d'un modèle de processus spécifique à une plateforme exécutable. Une première solution consiste à transformer les processus BPMN vers des processus WS-BPEL. D'ailleurs, la spécification BPMN propose un mapping de BPMN vers WS-BPEL. On a exploré cette possibilité en proposant une implémentation du modèle de l'apurement du T1 dans une coopération avec une vue distribuée dans le cadre d'une approche d'interopérabilité unifiée en utilisant l'ESB Petals[34] (Lemrabet et al., 2010). Cependant, WS-BPEL n'est pas adapté pour répondre aux interactions humaines (Rickayzen, 2007). Plus précisément, WS-BPEL n'est pas adapté pour interagir avec des taches humaines (appelées *User Tasks* dans

[34] http://www.petalslink.com/

BPMN), qui selon (OASIS04, 2010) **(Définition 22 - Tâche Humaine)**[35] sont des services accomplis par des utilisateurs pour permettre l'intégration des humains dans les applications orientées services.

Pour répondre à ce problème l'OASIS a proposé deux standards BPEL4People (OASIS02, 2010) et WS-HumanTask (OASIS04, 2010) afin de standardiser l'interaction entre WS-BPEL et les taches humaines. BPEL4People étend WS-BPEL en introduisant des éléments pour modéliser les interactions humaines, alors que, la motivation principale de WS-HumanTask est de permettre à des applications orientées service de créer des taches humaines d'une manière orientée services. À cet effet, WS-HumanTask introduit une grammaire de description des taches humaines et des notifications. Il fournit une API standard qui permet aux consommateurs d'accéder aux données et aux actions du contexte d'un processus qui est en attente d'une interaction humaine. N'importe quel portail ou application existante peut accéder aux informations du processus à partir de leur f*rontends* (Cheliah et al., 2012).

Une deuxième solution consiste à explorer les possibilités de BPMN 2.0 qui a comme ambition d'être exécutable. Dans ce cas, la seule différence entre les modèles BPMN 2.0 exécutables (niveau PSM) et les autres modèles, c'est qu'il y a plus de détail technique dans les modèles exécutables (Buelow et al., 2010).

On s'intéresse ici à la deuxième solution et on explique comment BPMN 2.0 permet la sérialisation et favorise la capacité d'exécution des modèles de processus. Pour cela, on s'attelle à décortiquer le méta-modèle BPMN, afin d'évaluer la sérialisation et la capacité d'exécution des modèles BPMN 2.0. Ensuite, on montre comment un BPMS tel que Bonita Open Solution (BOS) produit des modèles de processus exécutable et conforme à la spécification BPMN 2.0 dans une coordination dans le cadre d'une approche d'interopérabilité unifiée.

3.4.1. Étape (7) : rendre le processus BPMN 2.0 exécutable (spécifique à une plateforme)

Les incohérences du méta-modèle BPMN 2.0

Bien qu'un effort considérable de la spécification BPMN 2.0 soit porté sur le méta-modèle, des incohérences demeurent dans cette nouvelle spécification. Certaines de ces incohérences affectent la modélisation des diagrammes, alors que d'autres affectent l'exécution des modèles. Ces incohérences sont dues à de multiples facteurs :

[35] Human tasks are services "implemented" by people. They allow the integration of humans in service oriented applications.

- Manque de précision dans la définition sémantique de certains concepts : Par exemple, la spécification ne précise pas si un sous-processus du type *Embedded Sub-Process* peut avoir plus d'un seul *None Start Event* et s'il en a deux comment faut-il les interpréter ? Représentent-ils des événements *Start Node Event* parallèles ou alternatifs ? (Silver, 2011 p.80) (OMG01, 2011 p.173).
- L'absence de contraintes (ou règles) formelles ce qui laisse la possibilité d'avoir des interprétations différentes : les contraintes doivent être déduites des 508 pages de la spécification.
- Le non-alignement entre certains concepts de la syntaxe concrète graphique et les deux syntaxes abstraite et concrète textuelle. Par exemple, le concept de la syntaxe graphique *Multiple Event* n'a pas d'équivalent ni dans la syntaxe abstraite ni dans les syntaxes textuelles.
- L'utilisation de deux syntaxes concrètes textuelles (XSD et XMI). Bien qu'officiellement les deux syntaxes textuelles proposées par BPMN 2.0 sont supposées être équivalentes, il demeure quelques différences entre ces deux syntaxes. À titre d'exemple le concept *Sequence Flow* ne doit connecter que des éléments de type *Flow Node* selon la syntaxe abstraite comme le présente la Figure IV-26.

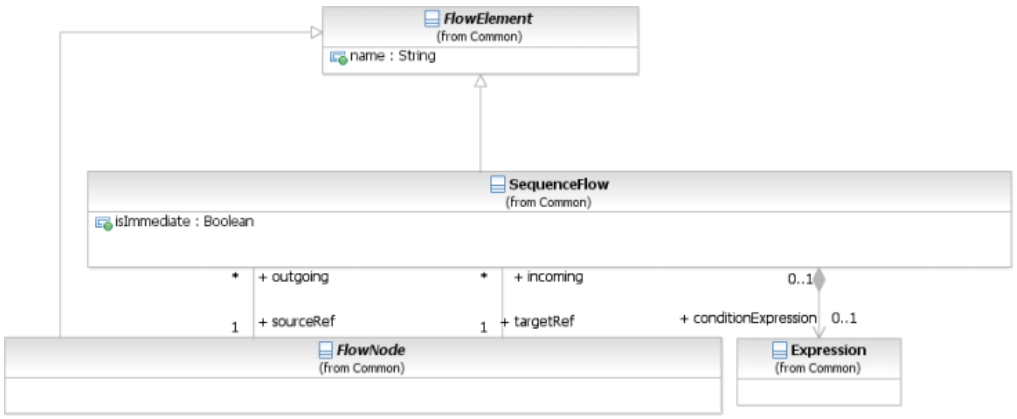


Figure IV-26. Syntaxe abstraite de l'élément *Sequence Flow*.

La syntaxe concrète textuelle décrite avec le format XMI observe cette contrainte (Figure IV-27 et Figure IV-28), par contre celle décrite avec le format XSD ne la respecte pas (Figure IV-29 et Figure IV-30).

```
…
<ownedMember xmi:type="cmof:Class" xmi:id="SequenceFlow"
name="SequenceFlow" superClass="FlowElement">
…
 <ownedAttribute xmi:type="cmof:Property" xmi:id="SequenceFlow-
```

```
sourceRef" name="sourceRef" type="FlowNode"
association="A_sourceRef_outgoing_flow"/>
 <ownedAttribute xmi:type="cmof:Property" xmi:id="SequenceFlow-
targetRef" name="targetRef" type="FlowNode"
association="A_targetRef_incoming_flow"/>
</ownedMember>

…

<ownedMember xmi:type="cmof:Class" xmi:id="FlowNode" name="FlowNode"
isAbstract="true" superClass="FlowElement">
…
</ownedMember>

…
```

Figure IV-27. Extrait de la syntaxe textuelle au format XMI de l'élément *Sequence Flow*

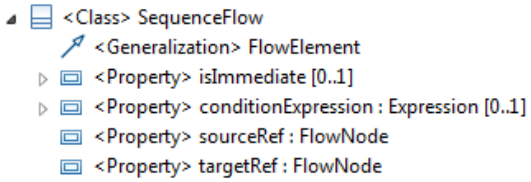


Figure IV-28. Représentation graphique de la syntaxe textuelle au format XMI de l'élément *Sequence Flow*

Dans le XSD les attributs *sourceRef* et *targetRef* sont de type IDREF, donc ils doivent pointer vers des attributs de type *ID*. À savoir, qu'une instance d'un document XML n'est valide que si et seulement si les attributs ou les éléments de type *IDREF* pointent sur des valeurs id qui existent dans ce même document et qui ne sont pas dupliqués. Donc les deux attributs *sourceREf* et *targetRef* peuvent pointer sur des éléments autres que des *FlowNode* (Figure IV-29 et Figure IV-30).

```
…
<xsd:element name="sequenceFlow" type="tSequenceFlow"
substitutionGroup="flowElement"/>
<xsd:complexType name="tSequenceFlow">
 <xsd:complexContent>
  <xsd:extension base="tFlowElement">
   <xsd:sequence>
    <xsd:element name="conditionExpression"  type="tExpression"
minOccurs="0" maxOccurs="1"/>
   </xsd:sequence>
   <xsd:attribute name="sourceRef" type="xsd:IDREF" use="required"/>
   <xsd:attribute name="targetRef" type="xsd:IDREF" use="required"/>
   <xsd:attribute name="isImmediate" type="xsd:boolean"
use="optional"/>
  </xsd:extension>
 </xsd:complexContent>
</xsd:complexType>

…
```

Figure IV-29. Extrait de la syntaxe concrète textuelle au format XSD de l'élément *Sequence Flow*

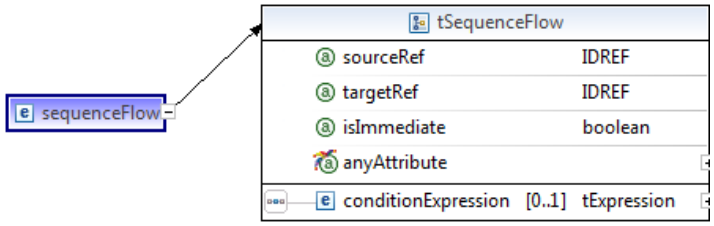


Figure IV-30. Représentation graphique de la syntaxe textuelle au format XSD de l'élément *Sequence Flow*

À la lumière de cette analyse du méta-modèle BPMN 2.0, on propose de classifier ce méta-modèle en se basant sur les cinq critères proposés par (Clark et al., 2008) pour classifier les méta-modèles dans cinq niveaux de maturité et selon cinq critères : (1) syntaxes concrètes, (2) syntaxe abstraite, (3) sémantique, (4) contraintes (règles) et (5) la possibilité d'extension du méta-modèle. Chacun de ces cinq critères représente une caractéristique du méta-modèle qui peut être définie formellement, informellement ou non définie.

Le méta-modèle BPMN 2.0 définit une syntaxe concrète textuelle partiellement formelle (certains éléments ne sont pas représentés), une syntaxe concrète graphique, plusieurs règles informelles à travers les 508 pages de la spécification, une syntaxe abstraite et une sémantique informelle relativement complètes, et enfin un mécanisme d'extension formel du méta-modèle. On estime que BPMN 2.0 atteint le niveau de maturité 3 selon (Clark et al., 2008).

BPMN 2.0 exécutable : pour l'échange de processus

C'est le respect des règles de sérialisation XML spécifiées dans le XSD qui garantit les échanges des modèles BPMN (exécutables ou pas). Cependant, les imprécisions du méta-modèle BPMN affectent la sérialisation XML des modèles de processus et par conséquent conduit à un manque d'interopérabilité entre les outils de modélisation et d'exécution des processus.

En étudiant les trois BPMS BizAgi Xpress 9.1.21020[36], Bonita Open Solution 5.4.1[37], Oracle BPM 11g[38] (Liu et al., 2011) on a constaté que chaque outil BPMS spécifie les détails indispensables à l'exécution des processus de sa propre manière. Chaque BPMS gère des éléments liés à l'instance du processus en exécution : les variables des processus, les expressions conditionnelles, les formulaires attachés au *User Task*, les exécutants des tâches et les données en entrée ou en sortie de chaque tache, etc. qui n'apparaissent pas forcément dans les diagrammes des processus. En effet, l'un des objectifs de BPMN 2.0 est de standardiser les détails d'exécution. On considère que le terme « *BPMN exécutable* » désigne la capacité d'un BPMS à spécifier, importer ou exporter les détails d'exécution d'un processus d'une manière conforme au

[36] http://www.bizagi.com/index.php?option=com_content&view=article&id=199&Itemid=119

[37] http://www.bonitasoft.com/

[38] http://www.oracle.com/technetwork/middleware/soasuite/learnmore/vmsoa-172279.html

méta-modèle BPMN 2.0. Ensuite, le BPMS peut interpréter le processus afin de l'exécuter comme Oracle BPM 11g (Buelow et al., 2010) ou le transformer en un modèle interne propriétaire avant de l'exécuter. Enfin, bien que chaque participant puisse utiliser son propre outil pour modéliser ces processus ; il faut s'assurer que les outils utilisés pour la modélisation des processus permettent l'import et l'export des modèles et respectent au moins les deux niveaux de conformité descriptive et analytique de la spécification BPMN 2.0.

Exemple d'exécution du processus d'apurement avec BOS)

Afin de mettre la lumière sur les différents aspects liés à l'exécution d'un processus et sur la façon dont ils sont définis, on propose d'étudier le modèle d'un processus exécutable et spécifique à une plateforme donnée. En effet, plusieurs aspects sont nécessaires à l'exécution du processus comme les données, les paramètres *input* et *output* de chaque tâche (*Task*) et l'assignement des acteurs aux tâches qu'ils exécutent, etc. En effet, la plupart des BPMS permettent de définir les informations concernant ces aspects à partir d'un éditeur graphique. L'éditeur est directement lié au moteur d'exécution de processus afin de permet aux BPMS de gérer l'ensemble du cycle de vie du processus : modélisation, développement, déploiement et tests. De plus, certains BPMS offrent des adaptateurs de services (appelés connecteurs par BOS) et permettent même aux développeurs de créer leurs propres connecteurs.

Dans une collaboration l'intérêt d'utiliser des BPMS qui respectent la spécification BPMN 2.0 est de permettre à tous les participants d'importer et d'exporter les modèles de processus PIM dans leurs propres BPMS. Ensuite, chaque participant peut utiliser son BPMS pour enrichir ses processus avec les informations nécessaires à leur exécution.

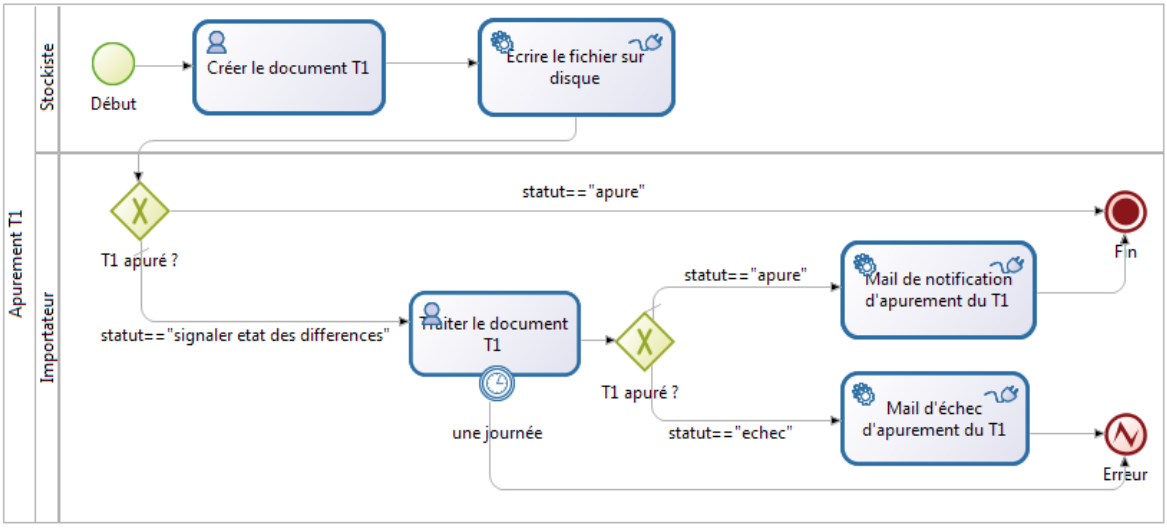


Figure IV-31. Processus « *Apurement document T1* » dans une coordination dans le cadre d'une approche d'interopérabilité unifiée avec BOS

La Figure IV-31 représente le modèle du processus « *Apurement document T1* » dans une coordination dans le cadre d'une approche d'interopérabilité unifiée. Ce processus contient de nouveaux éléments de flux (*FlowElement*) BPMN afin d'illustré certaines capacités de BOS.

Le stockiste se connecte à l'application et saisit les informations relatives à l'apurement du T1 dans un formulaire qui représente l'activité « créer document T1» (Figure IV-32). Ensuite, Les informations saisies par le stockiste sont stockées dans trois variables. La Figure IV-33 illustre comment ajouter ses variables au processus exécutable avec BOS. Les variables peuvent être typées par des types simples (texte, integer, flot, boolean, date), des objets java ou avec XML.

Figure IV-32. Formulaires d'authentification et de saisie du document T1 générés par BOS

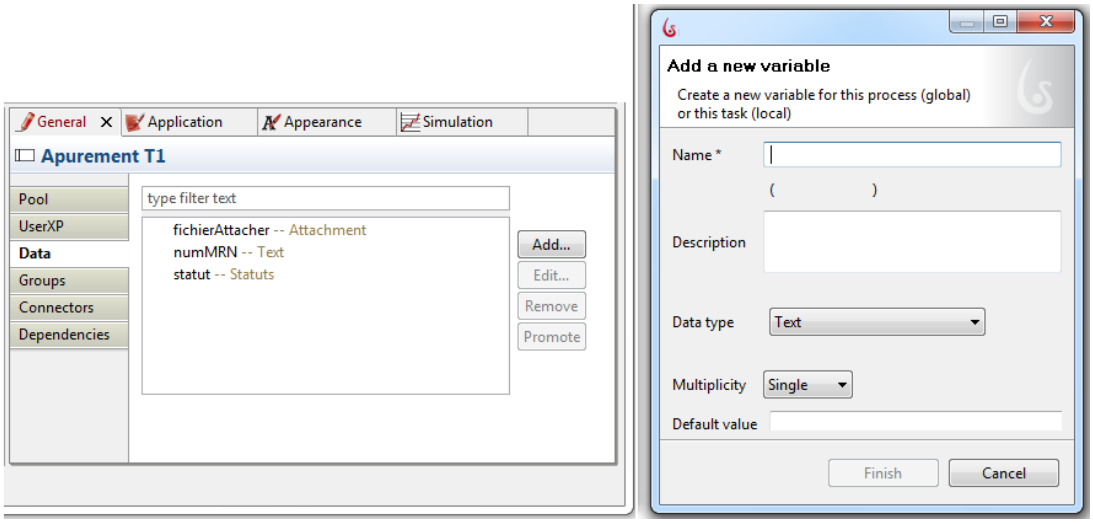


Figure IV-33. Définition des variables dans le processus

La Figure IV-34 est un extrait du code du modèle du processus exécutable. On constate que le modèle de processus référence des types de données définie avec des classes Java. En effet, il est tout à fait possible d'utiliser d'autres langages comme Java pour définir les données même s'il est préférable d'utiliser le langage XML.

```
<?xml version="1.0" encoding="UTF-8"?>
<process-definition product-version="5.4">
…
   <data-fields>
    <data-field name="statut">
     <label>statut</label>
     <value>…value>
     <description/>
     <datatype-classname>java.lang.String</datatype-classname>
     <is-transient>false</is-transient>
     <enumeration-values>
      <enumeration-value>apure</enumeration-value>
      <enumeration-value>echec</enumeration-value>
      <enumeration-value>signaler etat des differences</enumeration-value>
     </enumeration-values>
    </data-field>
    <data-field name="numMRN">
     <label>numMRN</label>
     <value>…</value>
     <description/>
     <datatype-classname>java.lang.String</datatype-classname>
     <is-transient>false</is-transient>
     <enumeration-values/>
    </data-field>
   </data-fields>
   <attachments>
    <attachment name="fichierAttacher">
     <label>fichierAttacher</label>
     <description/>
    </attachment>
   </attachments>
…
</process-definition>
```

Figure IV-34. Sérialisation des variables avec BOS

Pour illustrer l'utilisation des connecteurs, on présente deux connecteurs : le premier est un connecteur simple qu'on a développé pour écrire le document T1 sur le disque (cf. Annexe A) et le deuxième est le connecteur BOS *Email* qui permet d'envoyer des e-mails.

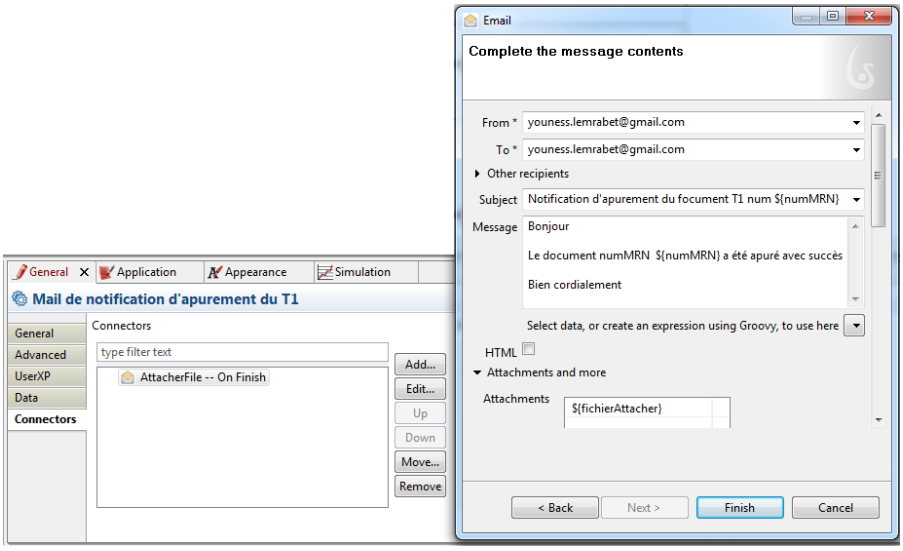


Figure IV-35 : Le connecteur Email de BOS

La Figure IV-35 montre un exemple de configuration de la tâche service « *Mail de notification d'apurement du T1*». Le connecteur Email de BOS est utilisé pour envoyer un mail avec un fichier joint contenant le document PDF chargé par le stockiste.

```
...
<activity name="Mail_de_notification_d_apurement_du_T1">
      <label>Mail de notification d'apurement du T1</label>
      …
      <connectors>
       <connector>
        <classname>org.bonitasoft.connectors.email.EmailConnector</classname>
        <description/>
        <event>automaticOnExit</event>
        <throw-exception>true</throw-exception>
        <parameters>
         <parameter name="setHeaders">…</parameter>
         <parameter name="setMessage">…</parameter>
         <parameter name="setSmtpHost">…</parameter>
         <parameter name="setAttachments">…</parameter>
         <parameter name="setSslSupport">…</parameter>
         <parameter name="setFrom">…</parameter>
         <parameter name="setTo">…</parameter>
         <parameter name="setCc">…</parameter>
         <parameter name="setBcc">…</parameter>
         <parameter name="setUserName">…</parameter>
         <parameter name="setCharset">…</parameter>
         <parameter name="setStarttlsSupport">…</parameter>
         <parameter name="setSmtpPort">… </parameter>
         <parameter name="setHtml">…</parameter>
         <parameter name="setPassword">…</parameter>
         <parameter name="setSubject">…</parameter>
        </parameters>
       </connector>
      </connectors>
      …
    </activity>

      …
```

Figure IV-36 : sérialisation du connecteur Email de BOS

4. CONCLUSION

Dans ce chapitre, on a utilisé la grille proposée dans le chapitre précédent (cf. Chapitre III - 3) pour expliquer comment spécifier les processus dans le cadre d'une collaboration. On a proposé une méthode conceptuelle descendante pour modéliser les processus en se basant sur l'approche MDI. Cette méthode est constituée de sept étapes déclinées sur les niveaux CIM-Haut, CIM-Bas, PIM et PSM de la démarche MDI. De plus, on a proposé trois palettes de notations pour faciliter la modélisation des processus avec BPMN 2.0.

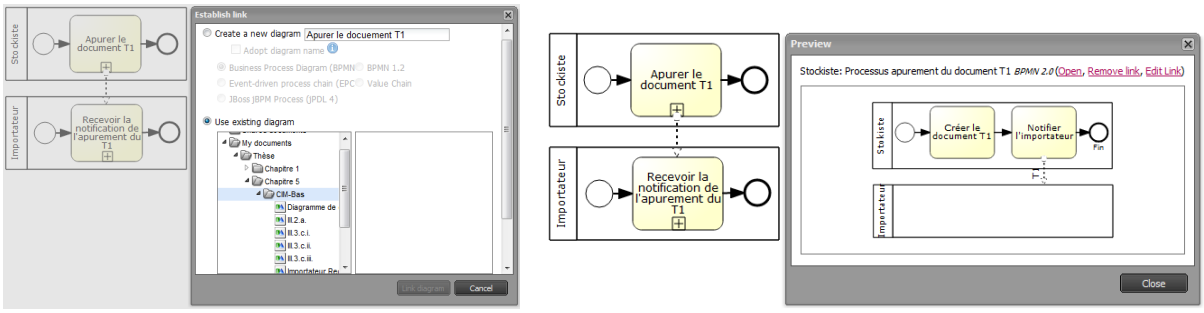
Les modèles CIM-Haut et CIM-Bas offrent une vue globale de la collaboration en utilisant les diagrammes de conversations et de collaborations. Dans ces deux niveaux de la démarche

MDI, on a suggéré d'adopter deux vues : (1) une vue globale de la collaboration avec les diagrammes de conversation. C'est une vue d'ensemble qui a pour but de déterminer le périmètre de la collaboration ; (2) une vue détaillée qui fait un zoom sur les modèles globaux et utilise les diagrammes de collaboration pour détailler chaque conversation de la vue globale. Cette vue se focalise plutôt sur la modélisation des processus et sur la description de leurs interactions.

Les modèles du niveau CIM-Haut représentent les exigences métiers du point de vue de l'utilisateur métier. Ils ont pour objectifs d'établir une compréhension partagée du contexte et de spécifier le besoin d'interopérabilité. Quant aux modèles CIM-Bas, ils représentent les exigences métiers des applications d'entreprise et se focalisent sur les besoins métiers automatisables. Alors que les modèles PIM concerne la logique du flux et des règles de décision de chaque processus de collaboration indépendamment d'une plateforme spécifique. On a démontré qu'au niveau PIM le type de collaboration et l'approche d'interopérabilité affectent la conception de la collaboration. Enfin, les modèles PSM raffinent les modèles PIM en ajoutant les détails nécessaires à l'exécution du processus (les données, les paramètres *input* et *output* de chaque tâche (*Task*) et l'assignement des acteurs aux tâches qu'ils exécutent, etc.). Ces modèles peuvent être spécifiés à partir de l'éditeur du BPMS utilisé pour exécuter le processus.

La principale limite de cette approche est liée à la difficulté de synchroniser les modèles génériques de la vue globale des niveaux CIM Haut et CIM-Bas avec les modèles de la vue détaillée des niveaux CIM-Haut et CIM-Bas et avec les modèles du niveau PIM. Toutefois, deux solutions particulièrement adaptées à une démarche descendante pallient ce problème :

1. L'utilisation d'une hiérarchie de processus avec la notion de sous-processus (Figure IV-37-a). Le sous-processus est lié au diagramme qui le représente (Figure IV-37-b).



a **b**

Figure IV-37. Créer un diagramme lié au sous-processus « *Apurer le document T1* »

a : Création du diagramme du sous-processus | b : Visualisation du diagramme du sous-processus

2. La création de différentes vues sur un même modèle. L'exemple typique est le passage du diagramme de *Collaboration* globale CIM-Bas vers le diagramme de *Collaboration* CIM-Bas automatisable (Figure IV-38). Le diagramme qui contient

uniquement les activités automatisables est une vue du modèle de la collaboration « *Réception de la marchandise et apurement du T1* » qui ne contient que les activités automatisables.

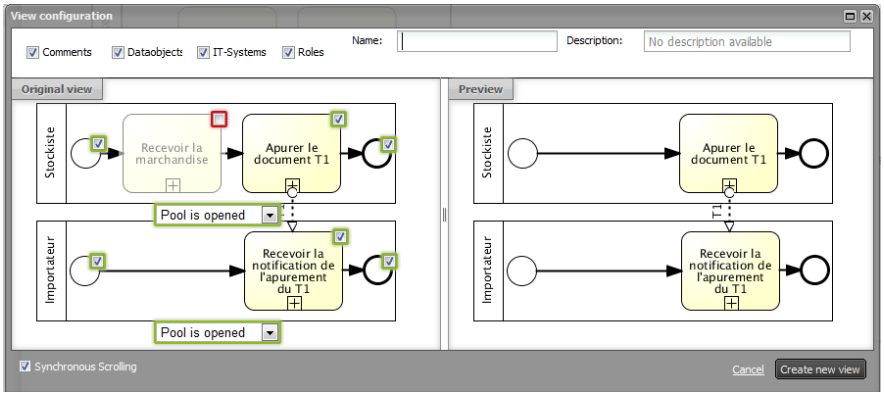


Figure IV-38. Vue sur le diagramme de collaboration « *Réception de la marchandise et apurement du T1* » contenant uniquement les activités automatisables

CHAPITRE V : Méthode et style pour l'identification et la spécification des services publics dirigés par le métier

1. INTRODUCTION DU CHAPITRE

Dans notre démarche dirigée par les modèles une plongée en abstraction est effectuée dans le cadre de la grille que l'on propose dans le chapitre III. Ainsi, les processus de collaboration découverts dans le chapitre précédent sont utilisés pour identifier et spécifier les services publics de la collaboration. Ces services supportent la mise en place des processus métiers de la solution SOA.

Dans ce chapitre on commence par présenter une transformation de modèles entre les formalismes BPMN et SoaML. Ensuite, en se basant sur la démarche MDI et à travers un cas d'étude, on explique comment concevoir des services avec SoaML. En effet, le formalisme SoaML favorise le respect de plusieurs principes de SOA : il permet de se concentrer sur la modélisation de certaines informations pertinentes du service (cf. Chapitre I – 4.1.3) en introduisant des concepts pour modéliser le contrat du service (cf. Chapitre 1 - 4.1.1) ainsi que sa composition (cf. Chapitre I - 4.1.8). Il permet aussi d'expliciter les dépendances entre les services et contribue au couplage faible des services (cf. Chapitre I – 4.1.2). Enfin, il apporte une description standardisée et pertinente des services ce qui favorise leurs découvertes (cf. Chapitre I – 4.1.7).

Contrairement à la modélisation des processus qui concerne aussi bien des processus automatisables (ex : processus métiers) ou pas (ex : processus organisationnels), la conception de services concerne uniquement les services automatisables. L'inventaire de services d'une entreprise contient ses services disponibles. C'est pourquoi la méthode de conception qu'on propose commence à partir du niveau CIM-Bas, puis s'étend sur les autres niveaux de l'approche MDI.

La méthode est déclinée en quinze étapes réparties sur les différents niveaux de MDI. Cependant, seules les étapes des niveaux CIM-Bas et PIM sont détaillées. Le but de la méthode est de montrer comment identifier, spécifier et réaliser des services au sens SOA indépendamment de toute plateforme ou technologie. Ensuite, ou bien les modèles de services du niveau PIM sont directement transformés pour générer le squelette de la solution SOA dans une plateforme cible (ex : web-services), ou bien ils sont marqués au niveau PSM pour les rendre spécifiques à une plateforme cible avant la génération du squelette de la solution SOA.

Pour maximiser le potentiel d'utilisation de SOA, on adopte deux vues d'architecture :

- **Une vue globale** qui consiste à identifier les services publics de la collaboration afin de construire l'architecture globale de services. Cette vue se limite à l'identification des rôles joués par les participants de la collaboration (fournisseurs/consommateurs) et à la description des contrats de services.

- **Une vue détaillée** qui spécifie le détail des services publics de la collaboration en précisant leurs interfaces de services et les interactions bidirectionnelles dans lesquelles ils évoluent. Cette vue explique aussi comment réaliser des services faiblement couplés et réutilisables.

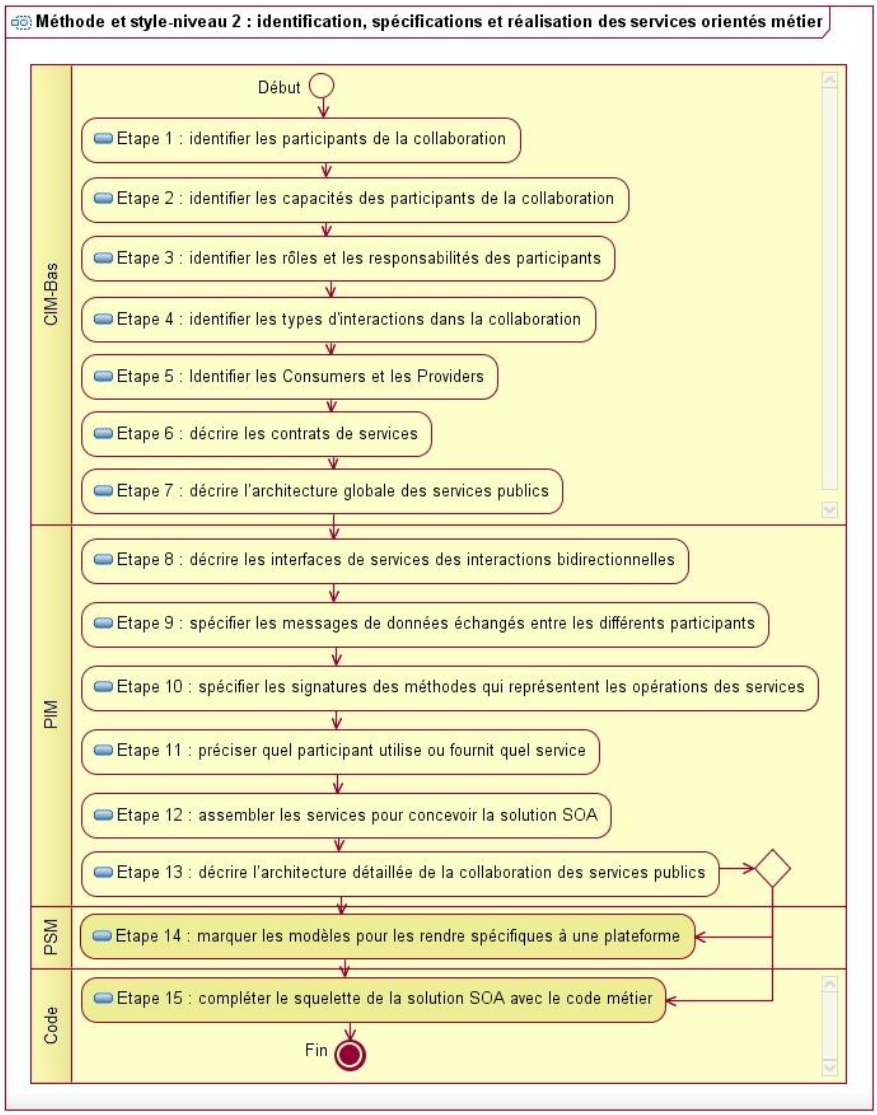


Figure V-1. Les étapes de la méthode d'identification et de spécification des services publics de collaboration

2. ÉTAT DE L'ART

Types et scénarios d'interactions

Pour spécifier un service, il faut déterminer le type d'interaction auquel il appartient. Deux types d'interaction existent : unidirectionnelle et bidirectionnelle. Chaque type regroupe un ou plusieurs scénarios d'interaction. Les quatre principaux scénarios d'interaction sont :

1. **Synchrone (requête/réponse) :** l'utilisateur du service envoie une requête au fournisseur qui traite la requête et renvoie une réponse. Dans ce scénario l'appelant est bloqué en attente de la réponse de l'appelé.
2. **Asynchrone avec Callback :** le fournisseur retourne la réponse au consommateur du service en invoquant un service de callback plutôt que de renvoyer la réponse au service qui a envoyé la requête (Swithinbank et al., 2005). Ce scénario est composé de deux scénarios requête/réponse synchrone, le premier est initialisé par l'utilisateur alors que le deuxième est initialisé par le fournisseur (Keen et al., 2004).
3. **Asynchrone one-way (aller-seul) :** dans ce mode le consommateur appelle le service du fournisseur et poursuit son traitement, il n'y a pas de callback (Smile, 2009).
4. **Asynchrone publish / subscribe (publication/abonnement) :** dans ce mode, un ou plusieurs consommateurs de services s'abonnent à un service du fournisseur. C'est en quelque sorte de l'asynchrone *one-way* à destinataires multiples. Cependant, ce n'est pas le service émetteur qui désigne les destinataires, ce sont les destinataires qui s'inscrivent à l'émetteur (Smile, 2009).

Le scénario *asynchrone avec Callback* représente une interaction du type bidirectionnelle alors que toutes les autres interactions sont unidirectionnelles.

2.1. SoaML

2.1.1. Introduction

En proposant SoaML (*Service Oriented Architecture Modeling Language*) l'OMG a adapté UML qui est fortement imprégné des principes de l'orienté objet aux principes de l'orienté service. La spécification SoaML (OMG02, 2009) définit un langage de modélisation des services au sens SOA. SoaML est avant tout destiné à modéliser des services dans des environnements distribués en prenant en compte les deux vues : métiers et IT. Le profil SoML permet de personnaliser les abstractions UML nécessaires à la modélisation des services. Il clarifie comment utiliser les méta-classes UML pour modéliser une solution SOA afin de garantir que les modèles des services reflètent les cinq principes SOA suivants : couplage faible, abstraction, réutilisation, autonomie et composition.

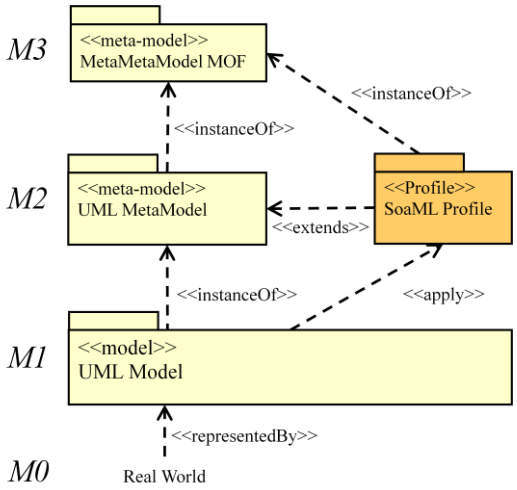


Figure V-2. Le Profil UML SoaML et le méta-modèle UML adaptation de (Bourey et al., 2011)

La Figure V-2 positionne le profil SoaML par rapport à UML et montre comment SoaML personnalise le méta-modèle UML en utilisant le mécanisme d'extension d'UML. Pour spécifier la structure des services, SoaML étend vingt-sept concepts UML, ces concepts sont regroupés dans six groupes : *Capabilities*, *Contracts*, *Services*, *Service Data*, *Milestones et BMM (OMG05, 2008) intégration*. La Figure V-3 représente les principales extensions UML définies dans la spécification SoaML.

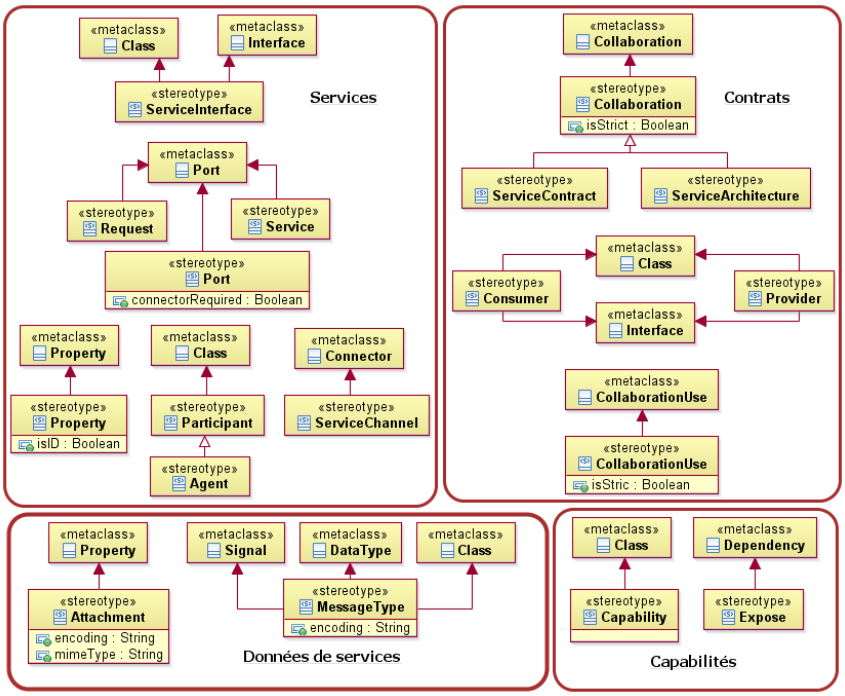


Figure V-3. Les principales extensions UML définies dans la spécification SoaML

Les participants « *Participants* » sont utilisés pour définir les fournisseurs et les consommateurs de services. Un participant peut représenter une personne, une organisation ou

un système. Il peut jouer le rôle d'un fournisseur de services, d'un consommateur de services ou des deux en même temps. Le participant peut contenir des « *Service port* » s'il est fournisseur de services ou « *Request port* » s'il est consommateur de services.

L'interface de service « *ServiceInterface* » est utilisée pour décrire les opérations fournies ou requises pour compléter la fonctionnalité d'un service. Les *ServiceInterfaces* aident à définir les interfaces et les responsabilités d'un *Participant* qui fournit ou utilise un service. Elles peuvent être utilisées pour typer les *Service ports* et les *Request ports* du *Participant*. Le *ServiceInterface* peut aussi spécifier le protocole d'utilisation des capacités d'un service avec les diagrammes de comportement UML (ex : diagrammes d'activité ou de séquence UML). Ceci permet de décrire le comportement des services et les échanges de messages entre les différentes parties. Enfin, le *ServiceInterface* n'exige pas la spécification du rôle du participant consommateur du service si ce dernier ne définit pas d'interface (i.e., interactions unidirectionnelle).

Le contrat de services « *ServiceContracts* » sert d'accord entre les différentes parties de la collaboration. Chaque rôle dans un *ServiceContract* est associé à une interface. Cependant, contrairement au *Service Interface* la spécification des rôles du fournisseur et du consommateur est obligatoire dans le *ServiceContract*. Ces rôles sont ensuite utilisés dans l'architecture des services pour relier les participants aux *CollaborationUse*. Enfin, le *ServiceContract* permet aussi de définir le comportement des services avec les diagrammes de comportements UML.

L'architecture de services « *Service Architecture* » fournit une vue de haut niveau de l'architecture orientée service. Elle définit comment un ensemble de participants qui partagent un objectif commun collaborent en fournissant et en utilisant des services exprimés avec des contrats de service (i.e., *ServiceContract*). L'architecture de services fournit le contexte dans lequel les participants collaborent, elle complète le point de vue des processus métiers qui décrivent le comportement des services (OMG02, 2009 p.104).

Les données de services spécifient les informations échangées entre les consommateurs et les fournisseurs de services. Ils regroupent deux concepts « *MessageType* » et « *Attachment* ». Le premier est utilisé pour spécifier l'échange de données entre le consommateur et le fournisseur du service, alors que le deuxième représente une partie non contenue dans le message mais attachée à ce dernier. Il est à noter que *MessageType* ne contient aucune information spécifique à un protocole ou à une implémentation donnée. Enfin, il existe deux styles de données échangés : le style orienté objet (*Remote Procedure Call - RPC*) et le style message ou document.

La Capacité « *Capability* » modélise l'aptitude à agir et à produire un résultat qui peut être fourni par un service. La *Capability* peut être utilisée pour identifier les services requis dans le

cadre de la collaboration. Elle peut aussi réaliser (au sens UML) des interfaces de services pour spécifier comment elles sont mises en œuvre par les participants.

2.1.2. Le profil UML SoaML

La spécification SoaML étend le méta-modèle UML avec un *Profil* UML qui adapte UML à la modélisation des services dans le cadre d'une architecture orientée services. Le profil UML SoaML fournit une syntaxe abstraite et trois syntaxes concrètes : une syntaxe concrète textuelle sous forme d'un fichier XMI (*XML Metadata Interchange*) et deux syntaxes concrètes graphiques avec deux formes de notations (les mots-clés et les icones). Quant aux contraintes, elles sont décrites textuellement et ne sont ni formelles ni exprimées avec un langage de spécification de contraintes comme OCL. Le Tableau V-1 illustre quelques exemples de la syntaxe concrète graphique de SoaML.

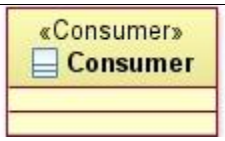
Mot clé	Icone	Exemple avec RSA
« *ServicesArchitecture* »		
« *ServiceContract* »		
« *Participant* »		
« *Provider* »	-	
« *Consumer* »	-	

Tableau V-1. Les deux syntaxes concrètes graphiques proposées par SoaML.

Le fait que la spécification SoaML ne propose pas des icônes pour tous les concepts (exemple : *Provider* et *Consumer*) prête à confusion. De plus, les exemples fournis dans la spécification sont illustrés avec deux outils différents : RSA[39] (*Rational Software Architect*) et Cameo Enterprise architecture[40], alors que seul RSA utilise des icones.

[39] http://www.ibm.com/developerworks/rational/products/rsa/

[40] https://www.magicdraw.com/cameo_enterprise_architecture

De plus, SoaML ne définit pas clairement la sémantique de certains concepts. À titre d'exemple une même phrase est utilisée pour décrire les concepts SoaML *Service « A Service is the visible point at which consumer requests are connected to providers and through which they interact in order to produce some real world effect »* (OMG02, 2009 p.101)et *Request « A Request is the visible point at which consumer requests are connected to service providers, and through which they interact in order to produce some real world effect »* (OMG02, 2009 p.80). Un autre exemple est la double utilisation de la phrase suivante dans une même page de la spécification : *« The way in which copy by value of an object is performed or how references are mapped to data types is platform technology dependent and not specified by SoaML* » (OMG02, 2009 p.65). De même trois concepts peuvent être utilisés pour spécifier un service : une interface simple (i.e., *Interface* UML), une interface de service (i.e., *ServiceInterface*) ou un contrat de service (i.e., *ServiceContract*). La spécification ne précise pas quand il est recommandé d'utiliser chacun de ces concepts. Il y a même un chevauchement sémantique entre les deux concepts SoaML *ServiceInterface* et *ServiceContract*. L'utilisation de ces deux concepts peut conduire à dupliquer l'effort de modélisation, parce qu'il est possible d'associer le comportement d'un service soit à son contrat de service (OMG02, 2009 p.90) soit à son interface du service (OMG02, 2009 p.98).

Actuellement, trois méthodes outillées utilisent le langage SoaML pour la modélisation des services. La 1ère méthode a été proposée par (Amsden, 2010), elle a été intégrée dans la version 2.9 de la méthode SOMA (*Service-Oriented Modeling and Architecture*) proposée par (Arsanjani, 2004) et supportée par RSA (*Rational Software Architect*). La 2ème méthode est proposée par (Casanave, 2009) et supportée par l'outil ModeDriven[41]. Enfin, la 3ème méthode MBDS (*Model-Based Development with SoaML*) est proposée par (Elvesæter et al., 2010) et supportée par l'outil de modélisation Modelio de SOFTEAM[42]. Les auteurs de ces trois méthodes participent activement à la spécification SoaML ce qui se traduit par l'existence de deux variantes de modélisation des services dans la spécification. La première variante est basée sur la méthode proposée par (Amsden, 2010) qui favorise l'utilisation des interfaces de services (i.e., *ServiceInterface*) alors que la deuxième variante recommande l'utilisation des contrats de services (i.e., *ServiceContrat*) et est basée sur la méthode (Casanave, 2009). D'ailleurs, la spécification fournie deux exemples : le premier « *Purchase Order Process Example* » est basé sur la méthode proposée par Amsden (2010) et le second « *Dealer Network Architecture* » est basé sur la méthode proposée par Casanave (2009). Ce constat est aussi partagé par les auteurs de la méthode MBDS (Elvesæter et al., 2011).

Enfin, le profil UML SoaML étend le méta-modèle UML de la version UML 2.1.2 (OMG10, 2007). Donc, il n'intègre pas les dernières évolutions d'UML spécifiées dans la version

[41] http://portal.modeldriven.org/

[42] http://www.modeliosoft.com/modelio-store.html?sobi2Task=sobi2Details&catid=12&sobi2Id=30

2.4. À titre d'exemple, le stéréotype SoaML *Property* étend le concept UML *Property* en ajoutant la tagged value *idID* qui a été ajoutée dans la spécification UML 2.4 (OMG03, 2011 p.128).

3. LA TRANSFORMATION ENTRE LES MODELES DE PROCESSUS (BPMN) ET DE SERVICES (SOAML)

L'objectif de la transformation est d'établir un pont entre les processus métier (approche BPM) et les services supportant le métier au niveau IT (approche SOA). Ce pont entre les processus modélisés avec BPMN et les services exprimés avec SoaML consiste à :

- Définir un mapping entre les processus de collaboration BPMN et les services SoaML en cherchant les concepts les plus proches sémantiquement.
- Réaliser le mapping à l'aide d'un langage de transformation de modèles.

Dans le cadre d'une collaboration, on propose d'utiliser deux critères pour différencier quatre catégories d'architecture de services : le premier critère concerne le niveau de détail de l'architecture des services cible. Ce critère permet de distinguer deux vues de l'architecture de services : une vue d'ensemble (i.e. globale) et une vue détaillée qui raffine les éléments de la vue globale afin de spécifier en détail les services. Le deuxième critère concerne le type des services modélisés : les services publics de la collaboration ou les services privés de chaque participant. Il est indispensable de séparer les services qui évoluent dans la collaboration inter-organisationnelle (i.e., services publics) des services privés de chaque participant. D'ailleurs, la spécification BPMN 2.0 différencie aussi les processus privés des processus publics (appelés processus abstraits dans BPMN 1.2 (OMG09, 2009 p.13) pour séparer les activités (services) publiques et privées. Cependant, on estime que la vocation première d'un processus métier est de décrire le flux de son exécution qui peut faire intervenir des services à la fois publics et privés. Donc, on réserve les notions de public/privé au niveau des services : Les quatre types d'architecture de services sont :

- **L'architecture globale des services publics de la collaboration :** définit les contrats de services entre les services métiers publics qui impliquent les différentes organisations.
- **L'architecture détaillée des services publics de la collaboration :** définit les interfaces de services entre les services métier publics qui impliquent les différentes organisations et précise le rôle joué par chaque organisation et ses points de communications.
- **L'architecture globale des services privés d'un participant :** définit les contrats de services entre les entités métiers internes (rôles) d'une organisation contrôlée par un seul participant.

- **L'architecture détaillée des services privés d'un participant :** définit les interfaces de services entre les entités métiers internes d'une organisation contrôlée par un seul participant et précise le rôle joué par chaque entité interne de l'organisation et ses points de communication.

Chacun des quatre types d'architecture de services est le résultat de la considération d'une perspective différente sur un élément du modèle source. Ce type de transformation dans lequel il est possible de générer plusieurs modèles cibles en se basant sur différents points de vue associés à un élément du modèle source a été abordé par (Grangel et al., 2009) qui proposent une transformation d'un centre de décision GRAI (Doumeingts, 1985) en diagramme de cas d'utilisation ou en diagramme d'activité UML.

3.1. Les règles de mappings entre BPMN et SoaML

La Figure V-4 résume une double transformation de modèles. Dans la première transformation exogène horizontale le méta-modèle source correspond au méta-modèle BPMN 2.0, alors que le méta-modèle cible correspond au méta-modèle UML et au profil SoaML. Les règles de mapping proposées sont basées sur l'équivalence sémantique entre les concepts BPMN et SoaML. Elles sont présentées en juxtaposant les éléments des syntaxes abstraites des méta-modèles BPMN et SoaML. Dans cette transformation certaines règles de mapping nécessitent la précision des informations suivantes :

- la vue (globale/détaillée) de l'architecture de services cible
- le type des services cibles (publics/privés).
- L'approche d'interopérabilité et le type de collaboration.

Ensuite, les éléments SoaML découverts avec la première transformation à partir des modèles BPMN sont raffinés dans une deuxième transformation endogène verticale dans laquelle les modèles source et cible sont exprimés avec SoaML.

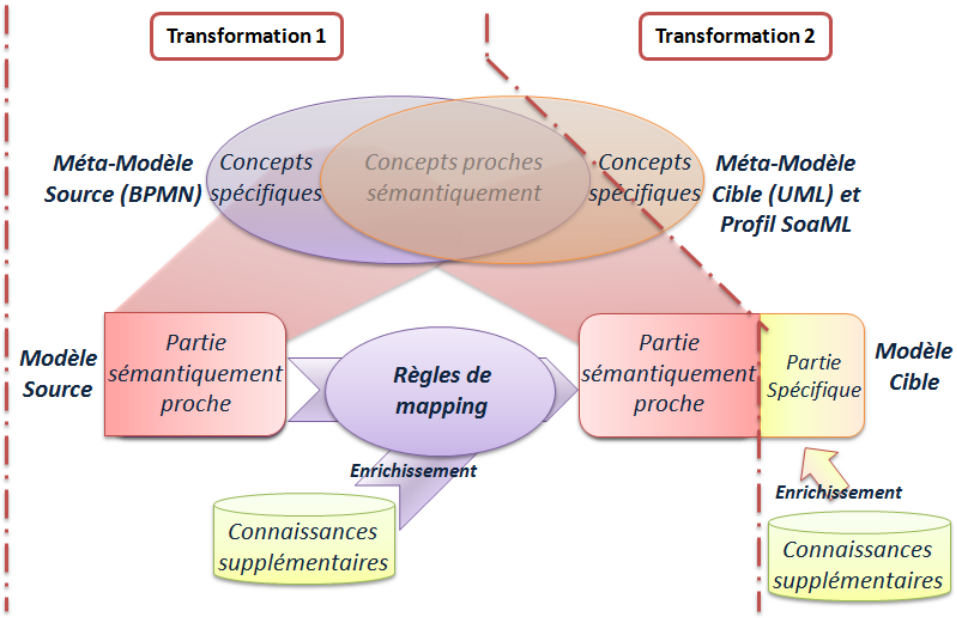


Figure V-4. Description des transformations de modèles inspiré de (Truptil et al., 2010)

Les règles de transformation que l'on présente ici concernent uniquement la première transformation. Quant à la deuxième transformation qui consiste à raffiner les concepts SoaML, elle est abordée dans la méthode de conception des services dans laquelle on explique comment spécifier en détail les services d'une solution SOA. Enfin, la deuxième transformation nécessite aussi l'ajout de certaines informations comme le type des interactions dans lesquelles le service évolue.

3.1.1. 1ère règle : mapping d'une *Collaboration* BPMN en SoaML *ServicesArchitecture*

D'une part BPMN 2.0 définit le concept *Collaboration* comme une collection de *Participants* utilisant des *MessageFlow* pour représenter leurs interactions (OMG01, 2011 p.109). D'autre part, SoaML introduit le concept *ServicesArchitecture* qui étend l'élément *Collaboration* d'UML souvent utilisé pour expliquer comment un ensemble d'instances collaborent pour achever une tâche ou un ensemble de tâches (OMG03, 2011 p.175). Le *ServicesArchitecture* décrit la manière avec laquelle des participants qui partagent un but commun utilisent des services décrits avec des contrats de services (OMG02, 2009 p.104).

BPMN 2.0	SoaML
Collaboration entre plusieurs *Participants*	Architecture de services de la collaboration

Tableau V-2. Mapping d'une *Collaboration* BPMN en SoaML *ServicesArchitecture*

3.1.2. 2ème règle : mapping d'un *Participant* BPMN qui ne référence pas de *Processus* en SoaML *Participant*

Dans BPMN 2.0 le *Participant* est le responsable de l'exécution d'un *Processus* évoluant dans une collaboration (OMG01, 2011 p.114) alors que le *Pool* est la représentation graphique du *Participant* dans la collaboration (OMG01, 2011 p.112). Un *Participant* qui ne référence pas un *Process* est alors représenté graphiquement par un *"Black box" Pool* (OMG01, 2011 p.112). L'élément *Pool* appartient à la syntaxe concrète graphique de BPMN qu'il ne faut pas utiliser dans les règles de mappings qui doivent se servir uniquement des éléments de la syntaxe abstraite des méta-modèles source et cible. On insiste sur le fait que seuls les éléments de la syntaxe abstraite doivent être utilisés dans les règles de mapping contrairement à ce qui est proposé par (Elvesæter et al., 2010) qui font usage de l'élément *Pool* de BPMN dans les règles de mapping qu'ils proposent.

La spécification BPMN 2.0 considère la *Collaboration* comme une collection de *Participants* représentés par des *Pools* (OMG01, 2011 p.109) et explique qu'un *Participant* peut référencer au plus un *Processus* (Figure V-5). Cependant la relation entre les concepts *Pool*, *Participant* et *Process* reste ambiguë. Quand l'élément graphique *Pool* est dessiné la spécification ne précise pas si son nom doit être affecté au *Participant*, au *Process* ou au deux en même temps. On considère qu'un *Participant* qui référence un *Process* est représenté graphiquement par un *Pool* non vide.

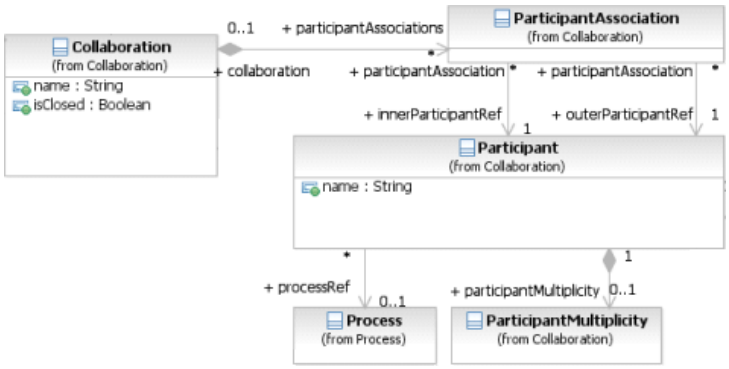


Figure V-5. BPMN 2.0 un *Participant* référence au plus un *Processus*

On a déjà précisé dans le chapitre précédent (cf. Chapitre IV - 3) que les diagrammes de collaboration doivent se focaliser sur les interactions entre les processus et non sur la logique des flux de ces derniers. Dans ce cas la collaboration peut être représentée partiellement ou exclusivement par des *"Black box" Pool*. De même, la spécification SoaML propose le concept *Participant* pour représenter une partie qui fournit ou utilise un service (OMG02, 2009 p.69). Elle précise que l'architecture de services décrit les relations entre les participants (OMG02, 2009 p.70).

BPMN 2.0	SoaML
Participant qui ne référence pas de *Processus*	Participant

Tableau V-3. Mapping d'un *Participant* BPMN qui ne référence pas de processus en SoaML *Participant*

3.1.3. 3ème règle : mapping d'un BPMN *Participant* qui ne référence pas de *Processus* en SoaML *Property*

L'élément *ServicesArchitecture* de SoaML contient des propriétés typées par des *Participants*. Ainsi, les *Participants* ne sont pas directement contenus dans la *ServicesArchitecture*. C'est pourquoi la transformation est réalisée vers une *Property*.

Dans UML une *Property* représente un état déclaré d'une ou plusieurs instances en termes de relations nommées vers une ou plusieurs valeurs. Quand la *Propery* représente une association end (i.e., *ownedAttribute*), sa valeur ou ses valeurs sont reliées à l'instance ou aux instances de l'autre bout de l'association (i.e., *Participant*) (OMG03, 2011 p.128). Enfin, l'élément SoaML *Property* étend l'élément SoaML *Property* d'UML en lui ajoutant une tagged value *isID*.

BPMN 2.0	SoaML
Participant qui référence un *Processus*	*Property*

Tableau V-4. Mapping d'un *Participant* BPMN qui ne référence pas de processus en SoaML *Property*

3.1.4. 4ème règle : mapping d'un *MessageFlow* BPMN en SoaML *CollaborationUse*[43]

Dans BPMN 2.0 le *messageFlow* est utilisé pour désigner les flux de messages échangés entre deux participants. Un *messageFlow* peut connecter soit des *"Black box" Pool* ou des *Flow Object* appartenant à différents *Participants* (OMG01, 2011 p.120). Quant au concept *CollaborationUse* de SoaML, il étend le concept *CollaborationUse* d'UML qui représente un usage particulier d'une collaboration dans le but d'expliquer les relations entre les propriétés d'un *Classifier* (i.e., *ServicesArchitecture*). En permettant d'attacher les entités d'un contexte donné (i.e., les *Participants*) aux rôles de la collaboration (OMG03, 2011 p.177). Le *CollaborationUse* indique la capacité d'un *Classifier* (i.e., *Participant*) de remplir un *ServiceContract* ou d'adhérer à une *ServicesArchitecture* (OMG02, 2009 p.59).

BPMN 2.0	SoaML
MessageFlow	*CollaborationUse*

Tableau V-5. Mapping d'un *MessageFlow* BPMN en SoaML *CollaborationUse*

[43] Circonstance de collaboration en français

3.1.5. 5ème règle : mapping d'un *MessageFlow* BPMN en SoaML *ServiceContract*

Le concept BPMN 2.0 *MessageFlow* est aussi mappé avec le concept SoaML *ServiceContract*. En effet, le *ServiceContract* est utilisé pour typer le concept *CollaborationUse* introduit dans le mapping précédent. Cependant, contrairement au *ServicesArchitecture* qui décrit les exigences pour la chorographie d'une collection de services, la *ServiceContract* décrit les exigences d'un service spécifique (OMG02, 2009 p.60). Le *ServiceContract* est utilisé pour modéliser et spécifier l'accord entre les différentes parties (le consommateur et le fournisseur du service). La spécification SoaML précise que le *ServicesContract* peut couvrir les interactions du service, sa qualité et ses exigences (OMG02, 2009 p.86). Cependant, les exemples fournis dans la spécification utilisent le *ServiceContract* seulement pour décrire les interactions entre les participants. Enfin, comme les services sont souvent complexes et imbriqués. Un *ServicesContract* peut contenir un autre *ServicesContract* pour représenter les services imbriqués. Dans ce cas le *ServiceContract* est appelé contrat de service composé (*compound service contract*) (OMG02, 2009 p.86).

BPMN 2.0	**SoaML**
MessageFlow	*ServiceContract*

Tableau V-6. Mapping d'un *MessageFlow* BPMN en SoaML *ServiceContract*

3.1.6. 6ème règle : mapping d'un *MessageFlow* BPMN en SoaML *Dependency*

Dans UML une relation du type *Dependency* signifie qu'un ou plusieurs éléments d'un modèle requièrent d'autres éléments pour leur spécification ou implémentation (OMG03, 2011 p.64). C'est-à-dire, le concept *CollaborationUse* utilise la relation *Dependency* pour être rattachée soit aux *Participants* soit aux ports du type *Request* ou *Service* des *Participants*. En effet, le *CollaborationUse* dispose de l'association end *roleBinding* pour la gestion de ses dépendances.

Contexte	**BPMN 2.0**	**SoaML**
Architecture de services globale	Début du *MessageFlow*	*Dependency* entre une *CollaborationUse* et un *Participant* qui utilise un service
	Fin du *MessageFlow*	*Dependency* entre une *CollaborationUse* et un *Participant* qui fournit un service
Architecture de services détaillée	Début du *MessageFlow*	*Dependency* entre une *CollaborationUse* et un port du type *Request* d'un *Participant* qui utilise un service
	Fin du *MessageFlow*	*Dependency* entre une *CollaborationUse* et un port du type *Service* d'un *Participant* qui fournit un service

Tableau V-7. Mapping d'un BPMN *MessageFlow* en SoaML *Dependency*

3.1.7. 7ème règle : mapping d'un *Participant* BPMN qui référence un *Processus* en SoaML *ServicesArchitecture*

D'une part, le *Participant* BPMN référence un *Process* qui décrit la séquence ou le flux des activités dans une organisation. Le *Process* représente un graphe de flux d'éléments qui définissent la sémantique d'exécution (OMG01, 2011 p.145). D'autre part, la spécification SoaML stipule explicitement qu'une architecture de services (i.e., *servicesArchitecture*) peut être dérivée ou crée à partir d'un processus métier BPMN (OMG02, 2009 p.140). En effet, la *ServicesArchitecture* de *SoaML* met un ensemble de services dans un contexte donné pour montrer comment les participants travaillent ensemble (OMG02, 2009 p.25).

Un processus BPMN peut représenter ou bien une collaboration entre plusieurs participants ou bien un processus interne d'un seul participant. Dans le premier cas l'équivalent du processus dans SoaML est l'architecture de services de la collaboration. Tandis que, dans le deuxième cas l'équivalent du processus dans SoaML est l'architecture de services privée propre à un seul participant. On propose donc deux mappings qui ont le même élément SoaML cible *ServicesArchitecture* utilisé dans deux contextes différents :

- Le 1er mapping concerne les processus modélisés dans une coordination dans le cadre d'une approche d'interopérabilité unifiée ou dans une vue centralisée de la coopération dans le cadre d'une approche d'interopérabilité fédérée (cf. Chapitre IV – 3.3.2.1). Dans ces deux types de collaboration, un seul processus représente la collaboration. C'est pourquoi, il est représenté par une architecture de services publics dans SoaML.
- Le 2ème mapping concerne les processus qui définissent les contrats de services entre les entités métiers internes (rôles) d'une seule organisation. Ces processus sont utilisés afin de créer une architecture de services privée d'un seul participant.

Contexte	**BPMN 2.0**	**SoaML**
Processus modélisé dans : • Une coordination dans le cadre d'une approche d'interopérabilité unifiée • Une vue centralisée de la coopération dans le cadre d'une approche d'interopérabilité fédérée	*Participant* qui référence un *Process*	Architecture de services de la collaboration
Processus internes a une organisation	*Participant* qui référence un *Process*	Architecture de services d'un *Participant*

Tableau V-8. Mapping d'un *Process* BPMN en SoaML *ServicesArchitecture*

3.1.8. 8ème règle : mapping d'un *Sub-Process* BPMN en SoaML *ServicesArchitecture* d'un participant

Un sous-processus *Sub-Process* BPMN est un concept utilisé pour représenter un *Process* d'un niveau plus bas ou pour définir un contexte appliqué à un groupe d'activités (OMG01, 2011 p.173-174). Un *Sub-Process* peut être mappé en *ServicesArchitecture* d'un seul participant *SoaML*.

Contexte	**BPMN 2.0**	**SoaML**
Architecture des services privés d'un participant	*Sub-process*	Architecture de services d'un participant

Tableau V-9. Mapping d'un *Sub-Process* BPMN en SoaML *ServicesArchitecture*

3.1.9. 9ème règle : mapping d'une *Lane* BPMN en SoaML *Participant*

Les propriétés *Properties* contenues dans le *ServicesArchitecture* sont du type *Participant*. Donc la *Lane* est mappée aussi avec le concept SoaML '(*Participant* (cf. 3ème règle de mapping).

BPMN	**SoaML**
Lane	*Participant*

Tableau V-10. Mapping d'une *Lane* BPMN en *SoaML* Participant

3.1.10. 10ème règle : mapping d'une *Lane BPMN* en SoaML *Property*

BPMN décrit l'élément *Lane* comme une sous-partition à l'intérieur d'un *Processus* (OMG01, 2011 p.305). Les *Lanes* sont utilisées afin d'organiser les activités (i.e., BPMN *Activities)* du *Pool*. L'usage des *Lanes* n'est pas spécifié par BPMN. Toutefois, il est précisé qu'elles sont généralement utilisées pour décrire les rôles internes d'une organisation (ex : types de responsabilité, systèmes, départements, etc.) (OMG01, 2011 p.306). Le concept *Lane* est mappé avec le concept SoaML *Property* introduit dans la 3ème règle.

BPMN 2.0	**SoaML**
Lane	*Property*

Tableau V-11. Mapping d'une *Lane* BPMN en SoaML *Property*

3.1.11. 11ème règle : mapping d'un *SequenceFlow* BPMN en SoaML *CollaborationUse*

L'élément BPMN *SequenceFlow* montre l'ordre des éléments de flux *Flow Elements* dans un *Process*. Chaque *Sequence Flow* a une seule source et une seule cible (OMG01, 2011 p.97). Ce mapping concerne uniquement les *SequenceFlows* dont la source et la cible appartiennent à deux

Lane différentes. L'élément *SequenceFlow* est mappé avec l'élément SoaML cible *CollaborationUse,* introduit dans la 4ème règle.

BPMN 2.0	SoaML
SequenceFlow	*CollaborationUse*

Tableau V-12. Mapping d'un *SequenceFlow* BPMN en SoaML *CollaborationUse*

3.1.12. 12ème règle : mapping d'un *SequenceFlow* BPMN en SoaML *ServiceContract*

Comme on l'a expliqué dans la 5ème règle le *ServiceContract* est utilisé pour typer le concept *CollaborationUse* introduit dans le mapping précédent. Donc, le *SequenceFlow* est aussi mappé vers le *ServiceContract*.

BPMN 2.0	SoaML
SequenceFlow	*ServiceContract*

Tableau V-13. Mapping d'un *SequenceFlow* BPMN en SoaML *ServiceContract*

3.1.13. 13ème règle : mapping d'un *SequenceFlow* BPMN en SoaML *Dependency*

Comme expliqué dans la 6ème règle l'élément *Dependency* est utilisé soit pour rattacher les *Participants* au *CollaborationUse* dans le cadre d'une architecture de services globale. Soit pour rattacher les ports de types *Service* et *Request* au *CollaborationUse* dans le cadre d'une architecture de services privée.

Contexte	BPMN 2.0	SoaML
Architecture de services globale	Début du *SequenceFlow*	*Dependency* entre une *CollaborationUse* et un *Participant* qui utilise un service
	Fin du *SequenceFlow*	*Dependency* entre une *CollaborationUse* et un *Participant* qui fournit un service
Architecture de services détaillée	Début du *SequenceFlow*	*Dependency* entre une *CollaborationUse* et un port du type *Request* d'un *Participant* qui utilise un service
	Fin du *SequenceFlow*	*Dependency* entre une *CollaborationUse* et un port du type *Service* d'un *Participant* qui fournit un service

Tableau V-14. Mapping d'un BPMN *SequenceFlow* en SoaML *Dependency*

3.1.14. 14ème règle : mapping d'un *SequenceFlow* BPMN en SoaML *Request* et *Service* ports

L'extrémité de départ du concept BPMN *SequenceFlows* est mapée au port *Request* et celle de l'extrémité d'arriver est mapée au port *Service*. En effet, dans SoaML les ports *Service* et *Request* héritent de l'élément SoaML *Port* qui stéréotype l'élément UML *Port*. Dans SoaML un port représente le point visible d'interaction auquel les requêtes du consommateur sont connectées aux fournisseurs du service (OMG02, 2009 p.80).

À travers le port *Request* le client (i.e., *Participant* consommateur de services) qui a des besoins s'interagit avec le fournisseur (i.e., *Participant* fournisseur de services) qui a des capacités compatibles. Quant au *Service*, il désigne un *Port* qui définit un point de connexion à travers lequel le *fournisseur* offre ses capacités en fournissant un service aux clients (OMG02, 2009 p.100). C'est le point visible par lequel les requêtes du client se connectent au fournisseur (OMG02, 2009 p.101). Le fournisseur rend disponibles des services qui répondent aux requêtes des clients conformément à ce qui est défini dans l'*Interface,* le *ServiceInterface* ou le *ServiceContrats* qui type le port.

Enfin, dans la vue détaillée d'une architecture de services l'utilisation des ports *Service* et *Request* est nécessaire afin de déterminer quels sont les services offerts par chaque *Participant,* qui peut les invoquer et quels sont leurs types.

Contexte	BPMN 2.0	SoaML
Architecture de services détaillée	Début du *SequenceFlow*	*Request*
	Fin du *SequenceFlow*	*Service*

Tableau V-15. Mapping d'un BPMN *SequeceFlow* en SoaML *Service* et *Request* ports

3.1.15. 15ème règle : mapping d'un *MessageFlow* BPMN en SoaML *Request* and *Service* ports

Dans BPMN un *MessageFlow* peut relier deux *Participants* ou deux éléments de flux *FlowElements* appartenant à deux participants différents (OMG01, 2011 p.120). Comme dans la règle précédente, l'extrémité de départ du *MessageFlow* est mappée au port *Request* alors que celle de la fin du *MessageFlow* est mappée au port *Service*. Enfin, on rappelle que l'utilisation des ports *Service* et *Request* est plutôt réservée aux architectures de services détaillés.

Contexte	BPMN 2.0	SoaML
Architecture de services détaillée	Début du *MessageFlow*	*Request*
	Fin du *MessageFlow*	*Service*

Tableau V-16. Mapping d'un BPMN MessageFlow en SoaML *Service* et *Request* ports

3.1.16. 16ème règle : mapping d'un *SequenceFlow* BPMN en SoaML *ServiceInterface* ou UML *Interface*

L'élément SoaML *ServiceInterface* (ou UML *Iinterface*) est utilisé pour définir l'interface et les responsabilités d'un participant qui fournit ou utilise un service (OMG01, 2011 p.93). Il est utilisé afin de typer les ports *Request* et *Services* des *Participants* (OMG02, 2009 p.80). L'élément BPMN *SequenceFlow* peut être mapé soit vers une *ServiceInterface* soit vers une *Interface*. C'est à l'architecte SOA de choisir l'élément cible approprié (cf. IV.).

Contexte	BPMN 2.0	SoaML
Architecture de services détaillée	*SequenceFlow*	*ServiceInterface*
	SequenceFlow	*Interface*

Tableau V-17. Mapping d'un BPMN *SequenceFlow* en SoaML *ServiceInterface* ou UML *Interface*

3.1.17. 17ème règle : mapping d'un *MessageFlow* BPMN en SoaML *ServiceInterface* ou Interface

Comme pour la règle de mapping précédente dans une architecture de services détaillée, l'élément SoaML *MessageFlow* peut être mappé soit vers une *ServiceInterface* soit vers une Interface.

Contexte	BPMN 2.0	SoaML
Architecture de services détaillée	*MessageFlow*	*ServiceInterface*
	MessageFlow	*Interface*

Tableau V-18. Mapping d'un BPMN *MessageFlow* en SoaML *ServiceInterface* ou UML Interface

3.2. Proposition de Patterns de transformation entre BPMN et SoaML

On présente ici les résultats de transformations de deux patterns *BPMN* en *SoaML* en se basant sur les règles de mapping sus-proposées. Le premier pattern est une transformation d'une collaboration BPMN en architecture globale de services publics. Alors que le deuxième pattern est une transformation d'un sous-processus d'un participant en architecture détaillée de services privés d'un participant.

3.2.1. Pattern de transformation d'une collaboration en architecture globale des services publics

Ce pattern de transformation concerne les processus collaboratifs de la vue détaillée du niveau CIM-Bas. L'intérêt de l'utilisation de SoaML est de déterminer l'ensemble des contrats de

services qui lient les partenaires de la collaboration. Le résultat de la transformation du pattern BPMN est présenté dans la Figure V-6.

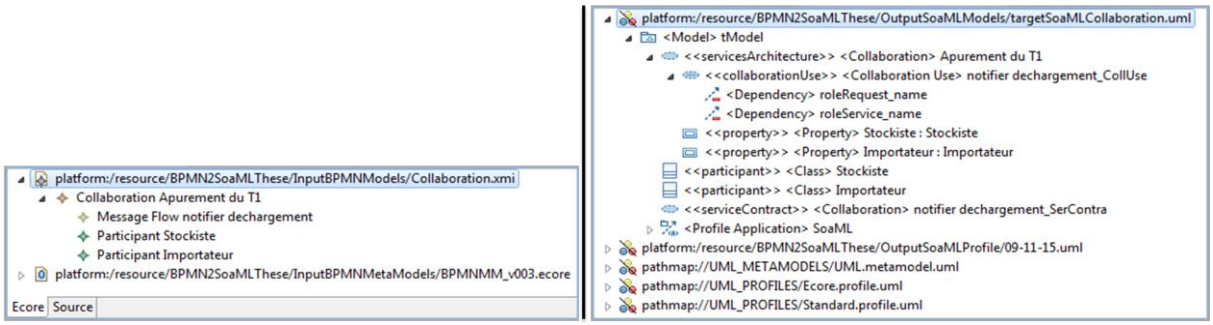


Figure V-6. Transformation d'un pattern de collaboration de haut-niveau (BPMN) en pattern d'architecture de services globale (SoaML) (syntaxe concrète textuelle)

On constate que l'architecture globale de services publics contient six éléments SoaML générés à partir d'une collaboration qui contient deux *Participants* et un *Message Flow* BPNM. La *CollaborationUse* « *notifier dechargement_CollUse* » est typée par le *ServiceContract* « *Apurement du T1* » qui est lié via deux *Dependencies* à deux propriétés typées par les participants « *Stockiste* » et « *Importateur* ». La première *Dependency* avec le rôle « *roleRequest_name* » relie le consommateur du service (i.e., le *Stockiste*) au *ServiceContract* alors que la deuxième avec le rôle « *roleService_name* » relie le fournisseur du service (i.e., l'*Importateur*) au *ServiceContract*. Enfin, il est à noter que les noms des services et leurs interfaces ne sont pas précisés dans ce type d'architecture globale (Figure V-7).

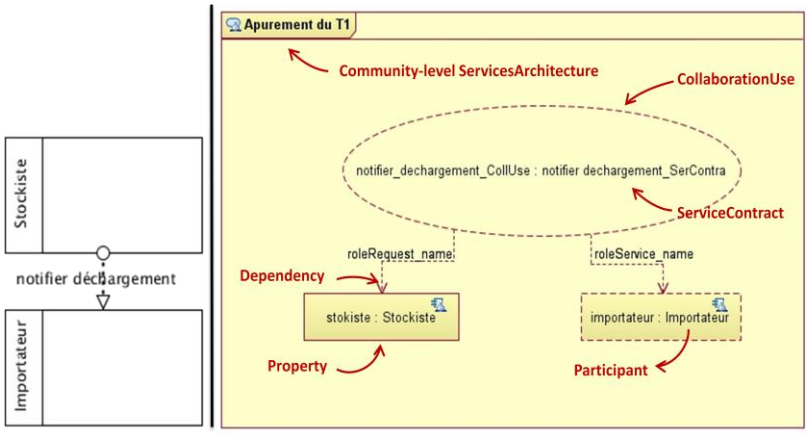


Figure V-7. Transformation d'un pattern de collaboration (BPMN) en pattern d'architecture de services globale (SoaML) (syntaxes concrètes graphique)

3.2.2. Pattern de transformation d'un sous-processus en architecture détaillée des services privés d'un participant

Ce pattern de transformation concerne la transformation des éléments d'un sous-processus en une architecture détaillée de services privés d'un seul participant de la collaboration. L'intérêt

de ce modèle d'architecture de services et d'avoir une vision détaillée des relations de dépendances et de composition entre les services privés d'un participant. Le résultat de la transformation de ce pattern est présenté dans la Figure V-8.

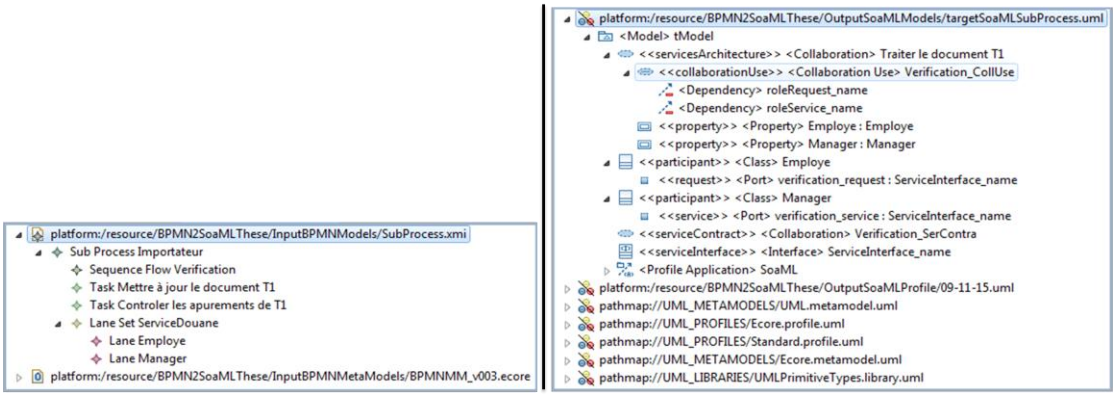


Figure V-8. Transformation d'un pattern sous-processus (BPMN) en pattern d'architecture de services détaillée (syntaxe concrète textuelle)

L'architecture détaillée des services privés met en évidence les différents rôles joués par les participants (Figure V-9). Elle permet de préciser les noms et les types des services utilisés par chaque participant dans l'architecture de services. Cette architecture contient cinq autres éléments de plus que les éléments présentés dans le pattern précédent: *Port*, *Request* port, *Service* port *Interfaces* UML simple et *ServiceInterface*.

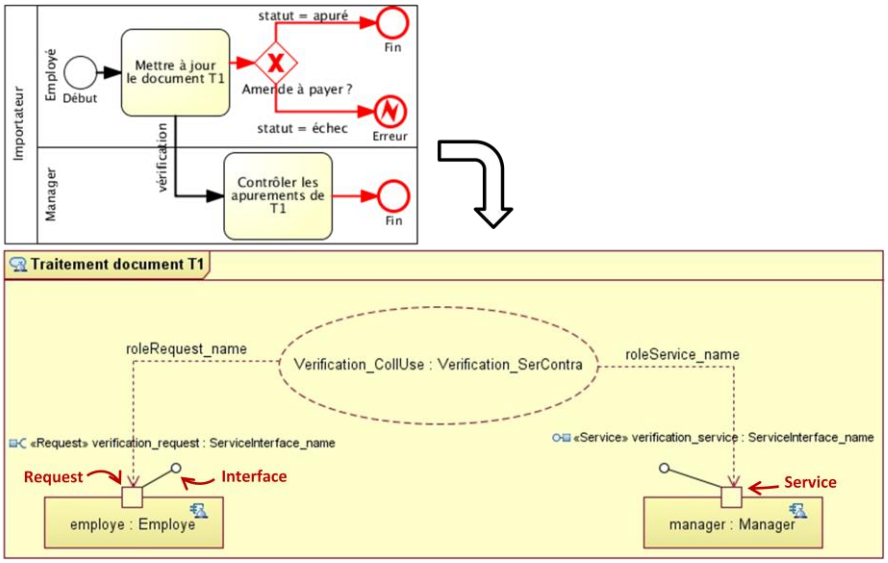


Figure V-9. Transformation d'un pattern sous-processus (BPMN) en pattern d'architecture de services détaillée (syntaxe concrète graphique)

On constate que l'architecture de services détaillée contient treize éléments générés à partir de sept éléments BPNM. La *CollaborationUse* « *Verification_CollUse* » est typée par le *ServiceContract* « *Verification_SerContra* » qui est lié via deux *Dependencies* aux ports des deux propriétés typées par

les participants « *Employe* » et « *Manager* ». La première *Dependency* avec le rôle « *roleRequest_name* » relie le port *Request* typé par l'interface simple « *ServiceInterface_name* » du consommateur du service (i.e., l'*Employe*) au *ServiceContract* alors que la deuxième avec le rôle « roleService_name » relie le port *Service* typé par l'interface simple « *ServiceInterface_name* » du fournisseur du service (i.e., le *Manager*) au *ServiceContract*. Ceci n'est possible que lorsque les parties et les rôles dans cette architecture ont le même type (OMG02, 2009 p.62) comme c'est le cas dans cet exemple.

3.3. Retour d'expérience sur la transformation entre BPMN et SoaML

On constate l'absence de mapping du type 1-1 entre les concepts BPMN et SoaML. De plus, la transformation entre ces deux formalismes nécessite l'ajout de connaissances supplémentaires. En effet, le modèle cible dépend de la vue d'architecture de services cible (cf. la 6ème, 8ème, 13ème, 14ème, 15ème, 16ème et 17ème règles de mapping), du type du service cible (cf. la 8ème règle de mapping) et du type de la collaboration ou de l'approche d'interopérabilité choisies (cf. la 7ème règle de mapping). L'automatisation de la transformation entre les modèles BPMN et SoaML nécessite donc l'ajout d'informations pour configurer la transformation. Il devient alors intéressant d'utiliser un outil de transformation configurable capable de générer différents modèles cibles à partir d'un seul modèle source. Par exemple, l'extension SOA de WebSphere de RSA version 8.0.1[44] d'IBM contient un éditeur graphique qui permet de visualiser et de configurer la transformation entre les modèles sources et cibles.

On regroupe le reste des concepts SoaML qu'on n'a pas étudiés en deux groupes : le premier groupe contient le concept *Agent* dont l'utilisation est spécifique à la mise en place d'une architecture à base d'agents. Comme la spécification SoaML précise que le concept *Agent* hérite du concept *Participant*. Alors les règles de mapping du concept *Agent* héritent des règles de mapping du concept *Participant*. Par conséquent, le concept *Agent* est proche sémantiquement des concepts BPMN *Participant, Sub-Process et Lane*. On n'aborde pas de règles de mapping pour ce concept car les architectures à base d'agents ne rentrent pas dans le périmètre de ce travail. Le deuxième groupe contient les concepts SoaML qui n'ont pas d'équivalents sémantiques dans BPMN : *Provider, Consumer, ServiceChannel, MessageType, Attachement, Milestone, Capability* et *Expose*. Ces concepts sont utilisés pour raffiner les concepts SoaML évoqués dans les règles de mapping sus-proposées.

[44] http://www-01.ibm.com/support/docview.wss?uid=swg24028087

Pour la validation des règles de mapping entre BPMN 2.0 et SoaML. On a réalisé une transformation de modèle utilisant le langage ATL (*Atlas Transformation Language*) (Bézivin et al., 2003) et l'outil Topcased 5.1.0[45]). Le code de la transformation est disponible en Annexe B).

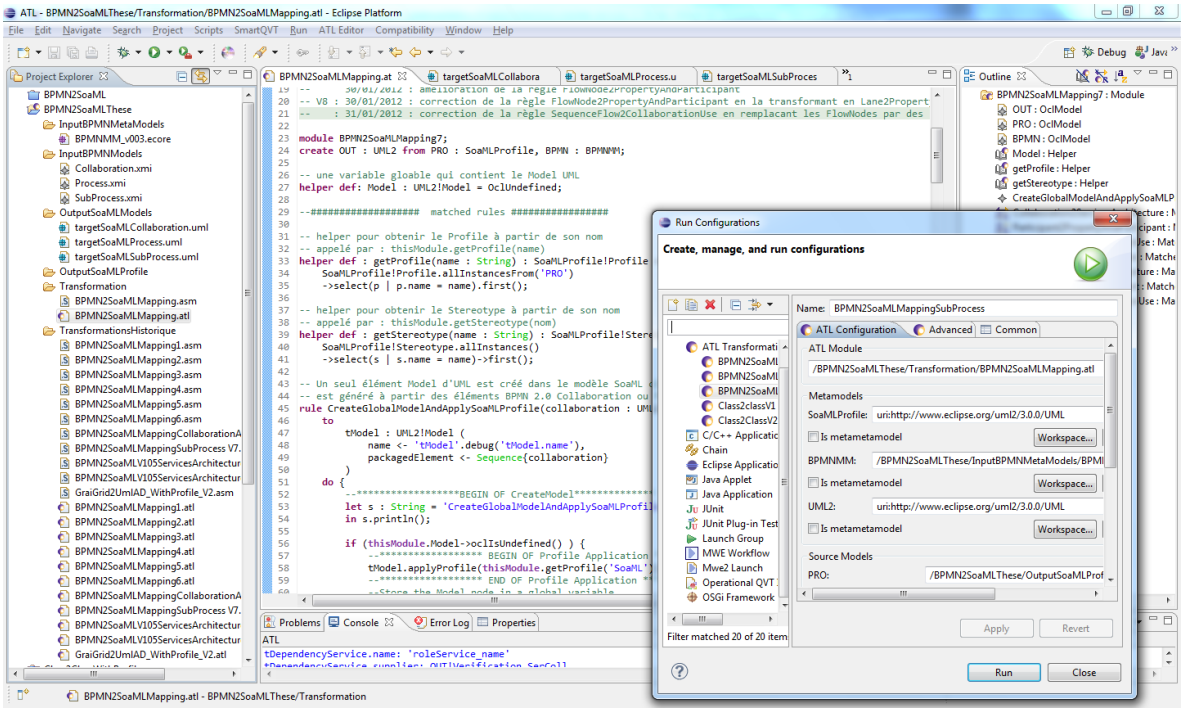


Figure V-10. Exemple des règles de transformation développées dans *Topcasesd* 5.1.0

Dans le but de faciliter l'effort de création des règles de transformation entre les éléments des méta-modèles BPMN et SoaML, on a travaillé avec (Yang, 2011) dans le cadre de son stage de Master afin d'adapter l'outil MT4MDE (*Mapping Tool for Model Driven Engineering*) proposé par (Lopes et al., 2006). Le but de ce travail était de proposer un outil graphique sous forme de plug-in Eclipse pour faire le mapping entre les éléments des deux méta-modèles BPMN et SoaML. L'outil génère les règles de transformation ATL correspondant aux éléments des méta-modèles source et cible mappés graphiquement. La Figure V-11 présente un exemple du code ATL généré après la construction d'un mapping graphique entre les concepts BPMN *Participant* et SoaML *Participant* (cf. III.1.b.).

[45] http://www.topcased.org/

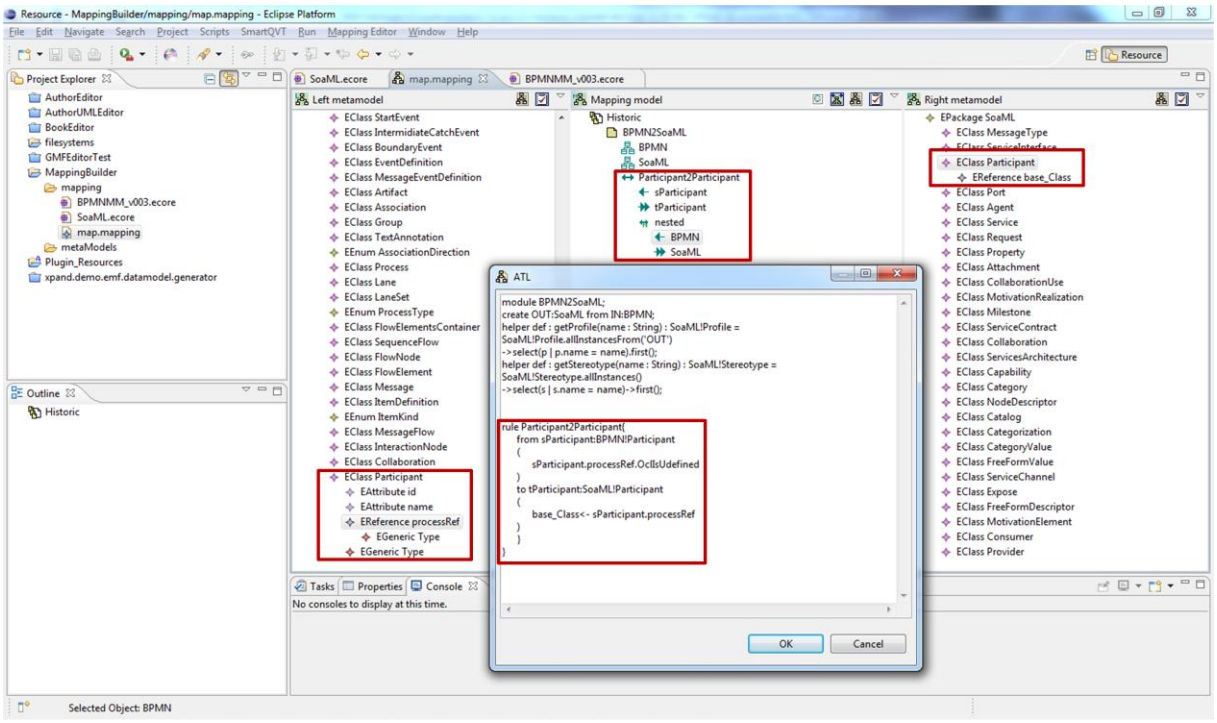


Figure V-11. Éditeur de mapping entre BPMN et SoaML et génération du code ATL

Enfin, comme on le verra par la suite les deux patterns de transformation qu'on propose ne se substituent pas à l'effort de conception de l'architecte SOA. L'utilisation d'un contrat de services n'est pas le produit d'une simple transformation d'éléments BPMN en SoaML mais le résultat des choix de conception de l'architecte SOA.

4. MÉTHODE

Ce sont les services métiers qui permettent de réduire l'écart entre les exigences métiers spécifiées avec les processus métiers et la solution SOA qui les implémentent. Il est donc essentiel que les méthodes de conception orientées services utilisent un langage de modélisation avec un niveau d'abstraction suffisamment élevé pour pouvoir décrire les services métiers. En effet, l'abstraction est l'un des principes de l'architecture orientée services (cf. Chapitre I – 4.1.3) et le langage SoaML (OMG02, 2009) permet de modéliser les trois types de services (i.e., métiers, entités et utilitaires) tout en respectant ce principe. SoaML permet de spécifier les interfaces et les contrats de services et de préciser comment les services sont utilisés (leurs consommateurs et fournisseurs). Ainsi, il devient possible d'établir les dépendances entre les participants et par la même occasion le couplage du système.

Le but de cette méthode est d'expliquer comment identifier, spécifier et réaliser les services d'une solution SOA avec SoaML à partir des exigences métiers spécifiées avec des processus de collaboration BPMN. Elle explique aussi comment faire le lien entre les exigences de

l'architecture d'entreprise et ceux des architectures d'applications dans le cadre d'une collaboration.

La méthode proposée se base sur le résultat de la transformation entre BPMN et SoaML et sur l'utilisation des deux vues d'architecture des services: globale et détaillée. Afin de déterminer les concepts SoaML à utiliser aux différents niveaux d'abstraction de la démarche MDI. De plus, la méthode se limite uniquement aux services publics de collaboration identifiés à partir des processus collaboratifs spécifiés dans le chapitre précédent. Il est à noter qu'on ne prend pas en considération le niveau CIM-Haut de la démarche MDI, car la conception des services avec SoaML concerne uniquement les services automatisables.

D'abord, le niveau CIM-Bas représente la vue métier globale qui utilise certains concepts SoaML utiles à l'identification des services. Ensuite, Le niveau PIM s'étend à d'autres concepts SoaML utilisés dans deux vues détaillées : la première vue spécifie le détail des services publics et la seconde réalise les services pour permettre la génération du code de la solution SOA spécifique à une plateforme. En effet, il est possible d'automatiser la transformation des modèles SoaML PIM vers le squelette de la solution SOA spécifique à une plateforme. La création et le marquage des modèles PSM est transparente pour le modélisateur qui doit en l'occurrence préciser une plateforme cible (ex : les web-services).

Concepts	**Vue globale**	**Vue détaillée**
ServicesArchitecture		
CollaborationUse		
Participant		
ServiceContract		
Capability		
ServiceInterface		
Provider		
Consumer		
MessageType		
ServiceChannel		

Tableau V-19. Les concepts SoaML utilisé dans cette méthode

Le Tableau V-19 représente les concepts utilisés pour spécifier l'architecture des services dans la méthode qu'on propose. Certains concepts SoaML sont utilisés uniquement dans la vue globale (ex : *Capability*, *ServiceContract*). Alors que d'autres sont utilisés uniquement dans la vue détaillée (ex : *ServiceInterface, MessageType, ServiceChannel*). D'autres encore sont utilisés dans les deux

vues (*ServicesArchitecture, CollaborationUse, Participant, Provider, Consumer*). Par l'intermédiaire d'un exemple on illustre par la suite comment utiliser ces concepts pour modéliser les services dans une architecture orientée services. Et plus particulièrement on explique comment combiner les deux concepts *ServiceContract* et *ServiceInterface* pour spécifier un service et les interfaces de services (i.e., *ServiceInterface*) pour raffiner les contrats de services (i.e., *ServiceContract*).

Notre méthode s'inspire fortement de l'exemple « *Purchase Order Process Example* » de la spécification SoaML et des trois méthodes (Casanave, 2009; Amsden, 2010; Elvesæter et al., 2010) sus-présentées.

La méthode (Casanave, 2009) utilise le concept *ServiceContract* pour décrire les services. Alors que la méthode (Amsden, 2010) recommande l'utilisation du *ServiceInterface* pour spécifier les services. On a constaté que d'une part, il est nécessaire d'utiliser le concept *ServiceContract* pour décrire le service dans l'architecture des services. En effet, Dans SoaML les services sont exprimés sous forme de contrats de services dans l'architecture de service (i.e., *ServiceArchitecture*). D'autre part, il est nécessaire d'utiliser des *ServiceInterfaces* pour typer les *Ports* des participants évoluant dans des interactions bidirectionnelles. Pour cela, on propose d'utilise les deux concepts : les contrats de services (i.e., *ServiceContract*) pour décrire les architectures de services (globale et détaillée) et les interfaces de services (i.e., *ServiceInterface*) pour spécifier en détail les services de la solution SOA.

La méthode qu'on propose se distingue par les caractéristiques suivantes :

- Elle se limite à la modélisation des services publics de la collaboration dans les vues globale et détaillée de la collaboration inter-organisationnelles.
- Elle adopte une approche descendante (*top-down*) et s'inscrit dans la continuité de la méthode d'identification et de spécification des processus (cf. Chapitre IV). L'approche descendante progresse par raffinement en commençant à partir d'un niveau de spécification de haut niveau (architecture globale des services et contrats des services), pour atteindre un niveau de spécification détaillé (les interfaces et des les composants de la solution et de ces données).
- Les services publics de la collaboration sont identifiés uniquement à partir des processus collaboratifs. L'identification des services d'une solution SOA dépasse largement le cadre de cette méthode. Les services peuvent être identifiés à partir des objectifs métiers, des modèles métiers fonctionnels, des processus métiers, des ègles métiers, des cas d'utilisation métier, des modèles de domaine ou des applications existantes (Dunnavant et al., 2011).

- Elle rend obligatoire la spécification des services publics découverts à partir des processus métier collaboratifs dans l'architecture des services publics de haut niveau. Comme la modélisation d'une architecture de services nécessite l'utilisation des contrats de services. La méthode nécessite d'abord l'utilisation des contrats de services pour spécifier les services globalement. Puis, le raffinement des contrats de services avec des interfaces de services en cas de besoin.
- Elle privilégie l'utilisation des contrats de services qui sont utilisés dans la spécification de l'architecture de services globale. Par la suite les contrats de services sont raffinés avec des interfaces de services en cas de besoin.
- Elle se base sur l'identification du type d'interaction dans lequel chaque service évolue.

Pour prendre en compte les différents scénarios d'interactions (cf. 2.1.). On a choisi l'exemple de collaboration : « *Mettre la marchandise en entrepôt sous douane* » identifié dans le diagramme de *Conversation* global du niveau CIM-Bas (cf. Chapitre V - Figure IV-12). Cette collaboration contient plus de participants et par conséquent plus d'interactions que la collaboration « Apurement du T1 » utilisée dans la spécification des processus collaboratifs dans le chapitre précédent. Le but de cette collaboration est d'obtenir l'autorisation de la douane pour stocker la marchandise dans un Entrepôt Sous Douane (ESD).

Les participants collaborent pour renseigner toutes les informations nécessaires à l'établissement d'une déclaration de mise en ESD (appelée déclaration dans la suite de ce manuscrit). Dans le cadre de cette collaboration, le *Stockiste* propose à l'*Importateur* de stocker sa marchandise dans un ESD pour pouvoir bénéficier des avantages de ce régime. Comme chacun des partenaires remplit une partie de la déclaration qui le concerne. L'*Importateur* cessera de sous-traiter la saisie de ses déclarations à son prestataire. Quant au *Commissionnaire,* il espère tester sa capacité à mettre en place un nouveau service à destination des stockistes qui voudront proposer à leurs clients de stocker la marchandise sous ESD. Dans ce scénario le *Commissionnaire* ne stocke pas la marchandise dans son ESD, il joue seulement le rôle d'intermédiaire entre le *Stockiste* et la douane. Enfin, le *Fournisseur* participe à la collaboration après avoir été sollicité par l'*Importateur*.

Les analystes métiers des participants ont modélisé les exigences métiers sous la forme d'un diagramme de coopération avec une vue distribuée dans le cadre d'une approche d'interopérabilité unifiée (cf. Chapitre IV – 3.3.2.3). C'est le *Mediateur* qui prend le contrôle des participants et orchestre l'appel aux différentes capacités des participants. D'abord, le *Fournisseur* initialise la déclaration avec des informations sur la marchandise. Notamment, les caractéristiques des articles : désignation, nomenclature, etc. Ensuite, l'*Importateu*r complète la déclaration en

rajoutant les informations qui nécessitent une connaissance des démarches de dédouanement (ex : les procédures douanières applicables, etc.). Puis, quand la marchandise arrive à destination, le *Stockiste* finalise la déclaration en précisant l'entrepôt dans lequel la marchandise sera stockée. Enfin, le *Commissionnaire* demande la déclaration auprès de la douane. Pour simplifier les processus de collaboration, seul le cas nominal est pris en compte. Dans le cas où la déclaration est refusée le processus d'orchestrateur se termine avec une erreur.

La méthode de conception s'intéresse uniquement à la modélisation des services indépendamment de toute plateforme ou technologie spécifique (niveau CIM-Bas et PIM de l'approche MDI). Le détail de l'implémentation de la solution SOA par des langages spécifiques à une plateforme donnée ne rentre pas dans le périmètre de cette approche. Cependant, dans un souci de fournir un exemple complet. On présente le résultat de la génération du code spécifique à la plateforme des web-services.

Enfin, il est à noter que l'identification des services doit être réalisée depuis les processus collaboratifs du niveau PIM. On estime que c'est à ce niveau PIM que les processus sont suffisamment aboutis pour être utilisés comme une base des spécifications métier. De plus, les processus du niveau PIM apportent une information très utile dans la modélisation des services. Ils spécifient qui parmi les participants, contrôle quels processus ou activités de processus. Ce qui permet par la suite d'identifier les fournisseurs et les consommateurs des services dans l'architecture des services. La Figure V-12 représente un exemple d'une coopération avec une vue distribuée dans le cadre d'une approche d'interopérabilité fédérée.

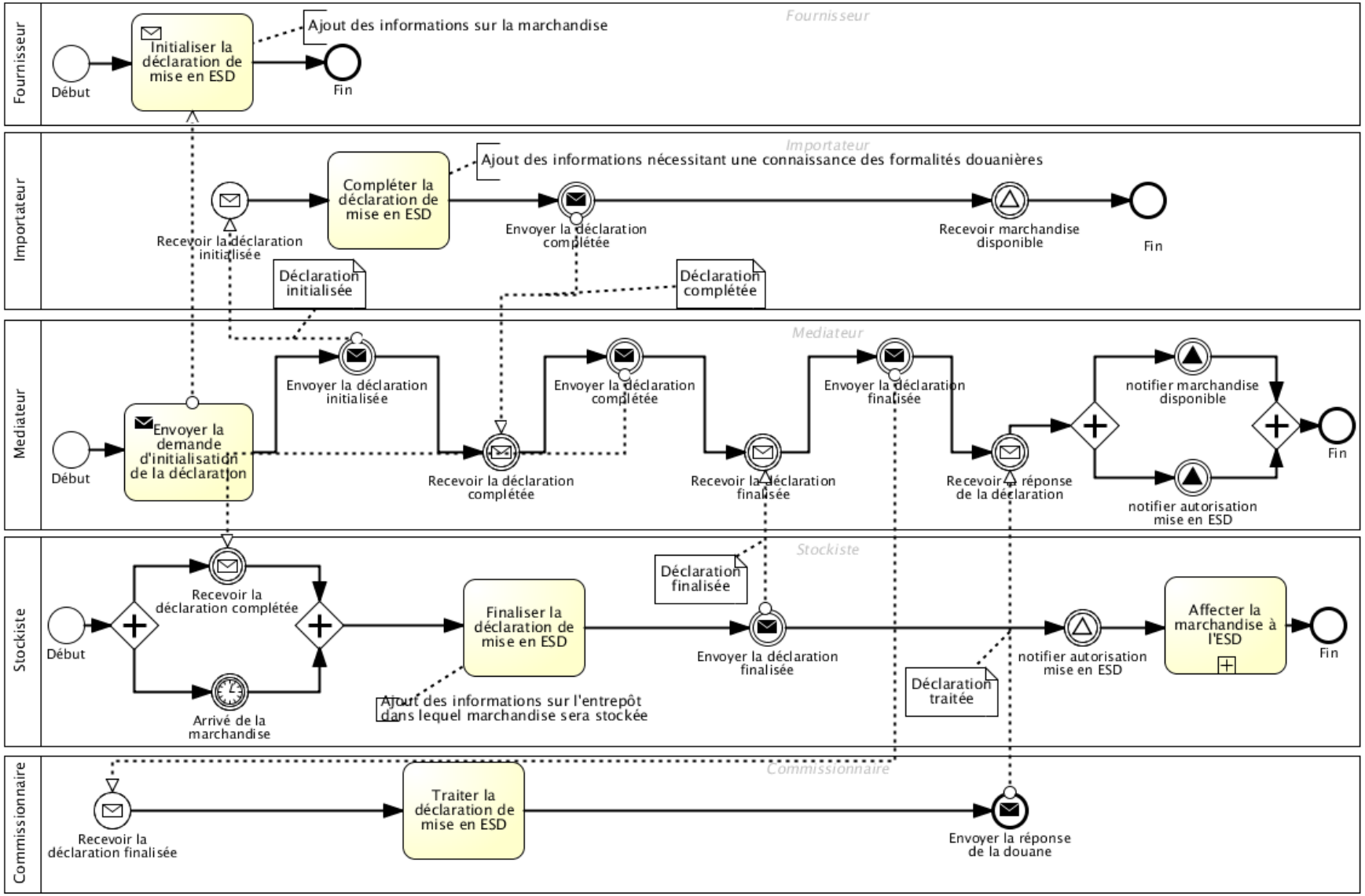


Figure V-12. La coopération avec une vue distribuée « mettre la marchandise en ESD » dans le cadre d'une approche d'interopérabilité unifiée

4.1. CIM-Bas : identification des services

À ce niveau le problème consiste à identifier les services de la solution SOA. Le but est d'identifier les services métiers publics à partir des exigences exprimées avec les processus de collaboration. La solution la plus simple consiste à créer un seul service et d'y mettre toutes les opérations découvertes. Dans ce cas tous les participants de la collaboration deviennent dépendent de ce même service. Cette solution entraîne un couplage fort entre les participants. Tout changement par le fournisseur du service affectera l'ensemble des participants qui le consomment. La solution alternative la plus complexe consiste à créer un service pour chaque opération découverte dans la collaboration. À ce stade la modélisation des services doit permettre de répondre aux questions suivantes :

- Quels sont les participants pour accomplir le résultat escompté de la collaboration ?
- Quelles sont les capacités des participants ?
- Quels sont les rôles et les responsabilités de chaque participant dans chaque interaction (i.e., les fournisseurs et les consommateurs des services) ?
- Quelles sont les interactions dans lesquelles évoluent les participants ?
- Quels sont les contrats de services associés à chaque interaction ?
- Quelles sont les dépendances entre les rôles ?

D'abord, il faut commencer par identifier les participants à partir des processus BPMN. Ensuite, il faut distinguer les capacités associées à chaque processus et à l'ensemble de la collaboration afin de déterminer les rôles et les responsabilités de chaque participant. Puis, il faut spécifier les types d'interactions de la collaboration (i.e., unidirectionnelle ou bidirectionnelle). Ainsi, l'association des capacités aux interactions permet de déterminer les *Consumers* et les *Providers*. Enfin, il convient de concevoir les contrats de service qui sont nécessaires à la spécification de l'architecture globale des services publics.

4.1.1. Étape (1) : identifier les participants de la collaboration

La première étape consiste à identifier les participants de la collaboration. Chaque participant joue le rôle d'un fournisseur ou d'un consommateur de services. La Figure V-13 représente les cinq participants SoaML identifiés à partir des *Pools* du diagramme de collaboration BPMN de la

Figure V-12. Il est à noter que dans une coopération avec une vue distribuée dans le cadre d'une approche d'interopérabilité unifiée le médiateur est considéré comme un participant.

Figure V-13. Les participants de la collaboration.

4.1.2. Étape (2) : identifier les capacités des participants de la collaboration

Dans un premier temps, il faut identifier les capacités liées à chaque processus de la collaboration. Ensuite, pour obtenir une première vision sur le couplage du système. Il faut spécifier les dépendances entre les capacités. Cependant, ces dépendances ne sont pas figées, elles peuvent être modifiées lors de la spécification détaillée des services.

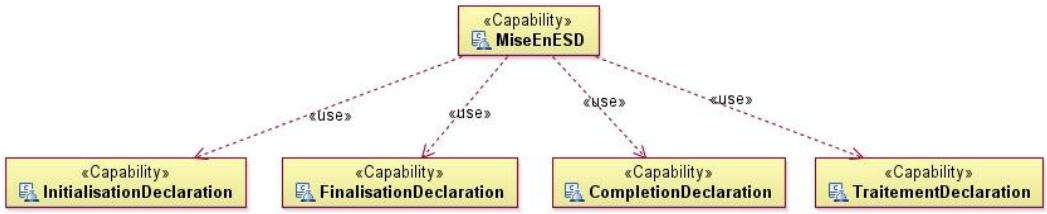


Figure V-14. Les capacités des services candidats

La Figure V-14 représente les capacités *initialisationDeclaration*, *FinalisationDeclaration*, *CompletionDeclaration* et *TraitementDeclaration* qui ont été découvertes respectivement à partir des processus *Fournisseur*, *Importateur*, *Stockiste*, *Commissionnaire*. La capacité *MiseEnESD* qui représente le *Médiateur* de la collaboration utilise les autres capacités afin d'obtenir l'autorisation de la douane nécessaire pour mettre la marchandise dans un ESD.

4.1.3. Étape (3) : identifier les rôles et les responsabilités des participants

Cette étape consiste à faire un premier effort d'identification des rôles des participants dans la collaboration. Par exemple, le *Fournisseur* joue le rôle de l'initiateur de la déclaration douanière (rôle : initiateur). Alors que l'*Importateur* est le bénéficiaire du régime ESD (rôle : bénéficiaire). Le *Stockiste* finalise la demande de déclaration et représente l'*Importateur* (rôle : représentant) auprès du *Commissionnaire* qui transmet la déclaration à la douane (rôle : déclarant). Enfin, toutes ces interactions sont orchestrées par le *Médiateur* (rôle : orchestrateur).

4.1.4. Étape (4) : identifier les types d'interactions de la collaboration

Dans cette étape, il faut déterminer les types d'interactions de la collaboration afin de décrire comment les services des fournisseurs de services sont appelés par les consommateurs. Dans BPMN une interaction bidirectionnelle est représentée par deux *Messages Flow* associés à des *Send (Throw) Message Event* et *Receive (Catch) Message Event*. Les *Signal Events* sont associés à des

interactions unidirectionnelles (scénario *asynchrone one-way*). Généralement, un *Signal* représente une diffusion (*broadcast*) quand il est utilisé entre plusieurs participants (Silver, 2011). Alors qu'un *Message Flow* entre un *Send Task* et un *Receive Task* représente une interaction unidirectionnelle (scénario de type synchrone requête/réponse).

Le rôle d'identification des interactions revient aux architectes SOA. On suggère d'utiliser un tableau pour identifier les interactions. Le Tableau V-20 représente le nom, le type et les rôles des participants dans chaque interaction : trois interactions bidirectionnelles et trois interactions unidirectionnelles ont été identifiées.

Interactions	**Types**	**Rôles du consommateur**	**Rôles du fournisseur**
Initialiser la déclaration de mise en ESD	Unidirectionnelle	Déclencheur	Initiateur
Compléter la déclaration de mise en ESD	Bidirectionnelle	Demandeur complétion	Bénéficiaire
Finaliser la déclaration de mise en ESD	Bidirectionnelle	Demandeur finalisation	Représentant
Demander autorisation de mise en ESD	Bidirectionnelle	Demandeur autorisation	Déclarant
Notifier l'autorisation de mise en ESD	Unidirectionnelle	Orchestrateur	Représentant
Notifier la marchandise est disponible	Unidirectionnelle	Orchestrateur	Bénéficiaire

Tableau V-20. Affectation des interactions aux consommateurs et aux fournisseurs

À ce stade, même s'il est difficile de prévoir toutes les interactions. Il est pertinent d'avoir une vue globale décrivant les interactions entre les rôles des différents participants (Amsden, 2010). On recommande d'utiliser un diagramme de comportement UML (*UML Behavior diagram*) pour capter une vue d'ensemble des interactions de la collaboration. Le diagramme de séquence de la Figure V-15 représente une vue globale des interactions entre les participants. La version finale de ce diagramme ne peut être que le résultat de plusieurs améliorations.

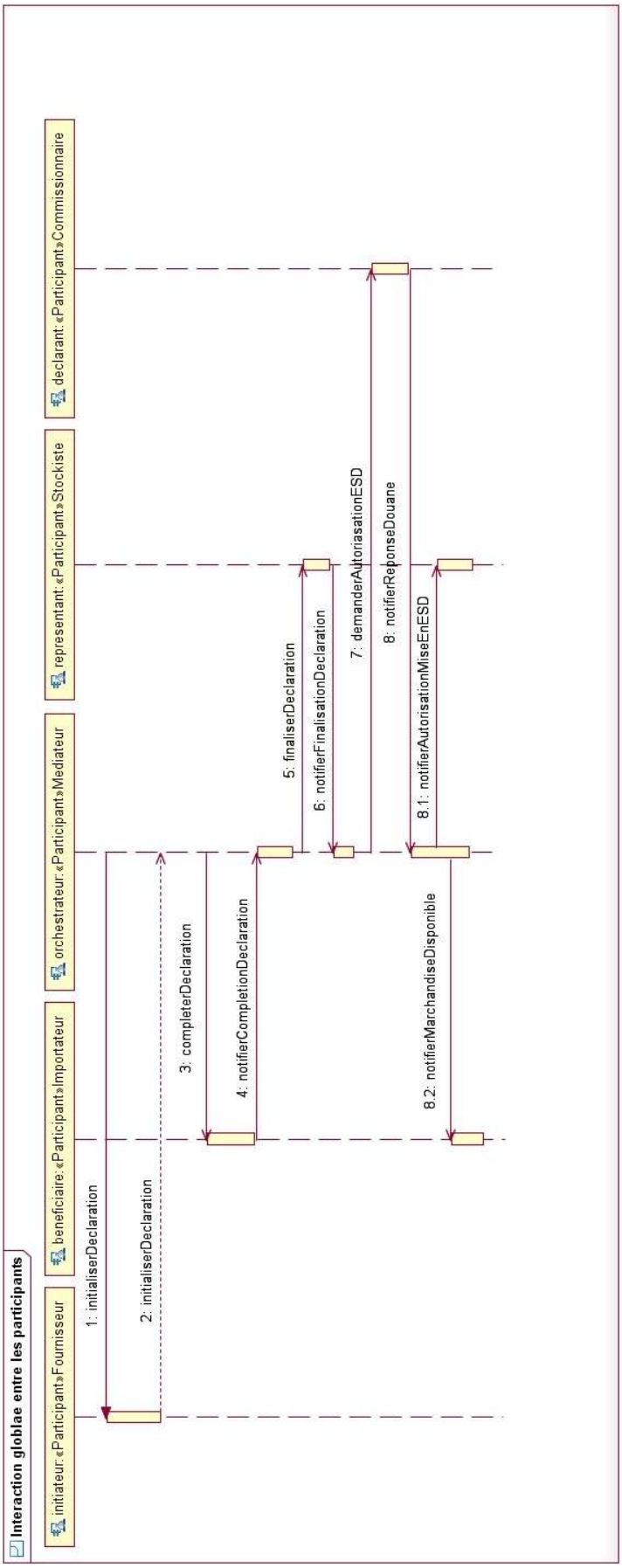


Figure V-15. Vue globale sur les interactions entre les participants

4.1.5. Étape (5) : identifier les *Consumers* et les *Providers*

Dans cette étape il faut identifier les consommateurs et les fournisseurs de services en utilisant les concepts SoaML de *Provider* et de *Consumer*. La spécification SoaML précise que le *Provider* et le *Consumer* peuvent être utilisés pour typer les rôles du contrat de service (OMG02, 2009 p.56 et 78).

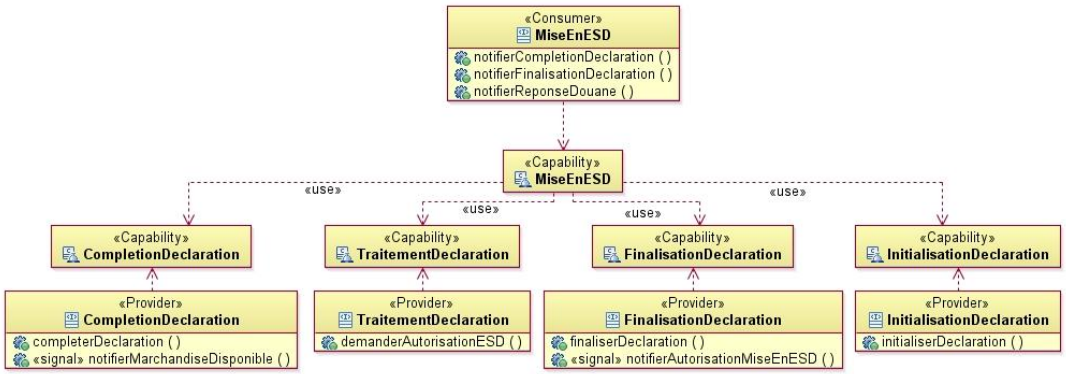


Figure V-16. Dérivation des *Consumers* et *Providers* des capacités.

La Figure V-16 spécifie plusieurs *Providers* et un *Consumer*. Chacun est dérivé à partir des capacités découvertes à l'étape 2 et regroupe les opérations identifiées dans les interactions de la collaboration à l'étape 4. Le *Cosumer* MiseEnESD dépend de la capacité du *Médiateur* qui initialise toutes les interactions. Enfin, la Figure V-17 spécifie explicitement les dépendances entre les *Consumer* et *Provider*. Il est à noter qu'à ce stade il ne faut pas spécifier les signatures des opérations ou les types des messages.

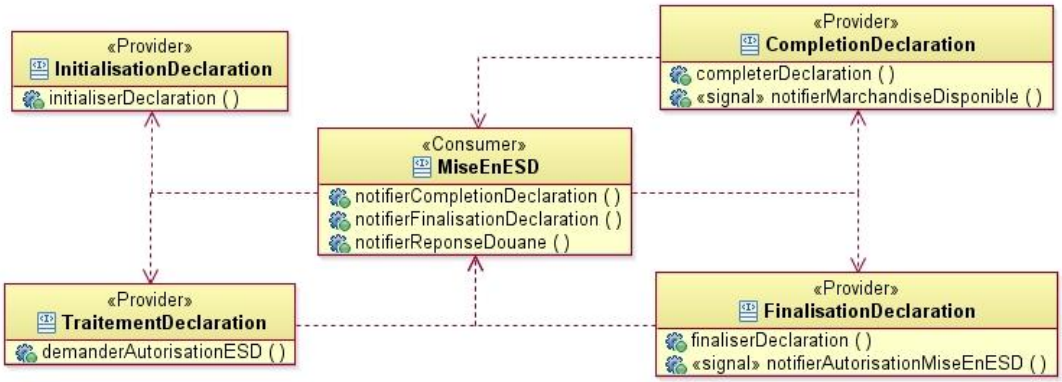


Figure V-17. Les *Consumers* et les *Providers* des services.

4.1.6. Étape (6) : décrire les contrats de services

Un contrat de service permet de modéliser une obligation contractuelle ou un accord entre différentes parties. Son utilisation est le résultat d'un effort de conception de l'architecte SOA qui décide quelles opérations regrouper dans chaque contrat de service. C'est une spécification du

service qui permet de préciser les rôles joués par le consommateur et le fournisseur et les interfaces qu'ils implémentent pour jouer ces rôles. Le contrat de service peut être binaire ou pluri-parties (en faisant intervenir plus de deux parties), simple ou composé d'autres contrats de services de faible granularité[46]. Dans ce cas, tous les contrats de services identifiés sont binaires (ils contiennent seulement deux participants) et simples (ils ne sont pas composés d'autres contrats de services).

À ce niveau, Il est possible de décrire les interactions du service avec des diagrammes de comportement UML (ex : diagramme d'activité, d'interaction, etc.) puis de les rattacher aux contrats de services. Cependant, on recommande de spécifier les interactions bidirectionnelles de chaque service au niveau PIM. C'est au moment de la spécification des interfaces de services que les opérations des services sont définitivement désignées. De plus, il est inutile de dupliquer la spécification des interactions dans les contrats et les interfaces des services.

- Spécification d'un contrat de service d'une interaction unidirectionnelle

L'initialisation de la déclaration est une interaction unidirectionnelle. Ce type d'interaction peut être utilisé avec des consommateurs anonymes. Ainsi, le consommateur ne fait pas d'hypothèse sur la chorégraphie du service qu'il invoque. Une interface UML simple est suffisante pour décrire un service évoluant dans une interaction unidirectionnelle (ex : *InitialisationDeclaration)*. Cependant, il est nécessaire de créer un contrat de service (ex : *InitialisationDeclarationSC)* pour pouvoir représenter ce service dans l'architecture des services.

Figure V-18. Contrat du service d'initialisation de la déclaration avec les deux rôles typés

Dans un contrat de service, il est obligatoire de préciser les rôles du consommateur et du fournisseur du service. La Figure V-18 montre comment le contrat de service d'une interaction unidirectionnelle doit être spécifié. Dans *InitialisationDeclarationSC* deux noms de rôles sont représentés : *declencheur* du participant *Mediateur* qui demande l'initialisation de la déclaration et *initiateur* du participant *Fournisseur qui* initialise la déclaration.

Quand l'interaction est unidirectionnelle soit les deux rôles du fournisseur et du consommateur sont typés par la même interface simple Figure V-18soit seul le rôle du

[46] Contrairement au contrat de service une interface de service ne peut pas être composée d'autres interfaces de services.

fournisseur du service est typé par l'interface simple (Figure V-19). Dans le deuxième cas le consommateur ne spécifie pas sa propre interface. De notre part, on recommande de spécifier les types de deux rôles comme présenté dans la Figure V-18.

Figure V-19. Contrat du service d'initialisation de la déclaration avec un seul rôle typé

- Spécification d'un contrat de service d'une interaction bidirectionnelle

Si un contrat de services spécifie un service qui évolue dans une interaction bidirectionnelle, alors il doit obligatoirement utiliser des rôles typés par les interfaces du fournisseur et du consommateur du service. Les trois contrats de services de la Figure V-20 font intervenir à tour de rôle le *Mediateur* et chacun des trois participants : *Importateur*, *Stockiste* et *Commissionnaire*. Ces contrats de services utilisent les rôles des participants identifiés auparavant (cf. Étape 3). Dans le contrat de service *CompleteionDeclarationSC* l'*Importateur* joue le rôle du bénéficiaire qui stock sa marchandise sous le régime ESD. Alors que dans le contrat de service *FinalisationDeclarationSC* le *Stockiste* joue le rôle du représentant de l'*Importateur* qui gère le stockage de la marchandise en ESD pour le compte de l'*Importateur*. Enfin, dans le contrat de service *TraitementDeclaration* le *Commissionnaire* joue le rôle du déclarant qui transmet la demande de mise en ESD à la douane.

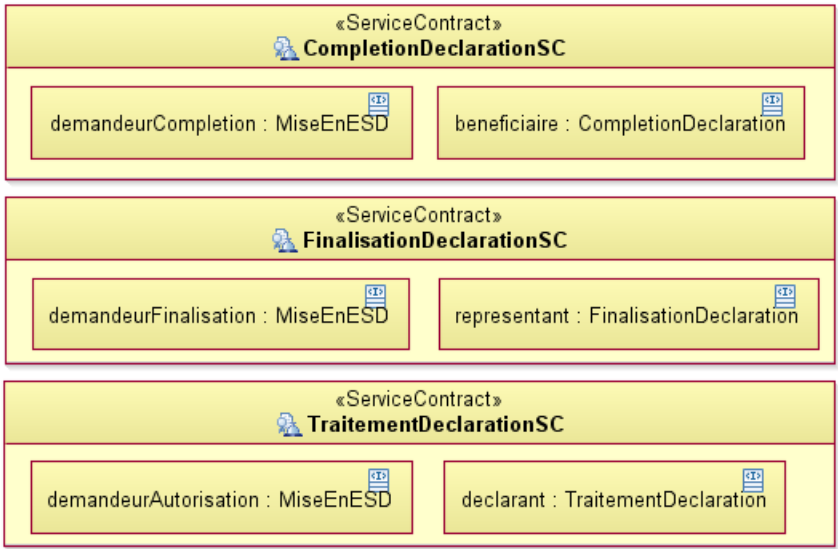


Figure V-20. Les contrats de services représentant des interactions bidirectionnelles

4.1.6.1. Étape (7) : décrire l'architecture globale des services publics

L'architecture globale des services publics représente un deuxième point de vue sur les exigences métiers exprimées avec les processus de collaboration BPMN. Alors que l'approche BPM propose une vue centrée sur les processus, l'architecture SOA offre une vue centrée sur les rôles des participants. Ceci contribue à réduire l'écart entre les processus métiers et la solution SOA qui les implémentent. L'architecture globale des services publics permet de spécifier comment réaliser les exigences métiers exprimés par la collaboration des processus. Cependant, elle ne prend pas en considération l'architecture IT cible (Amsden, 2010).

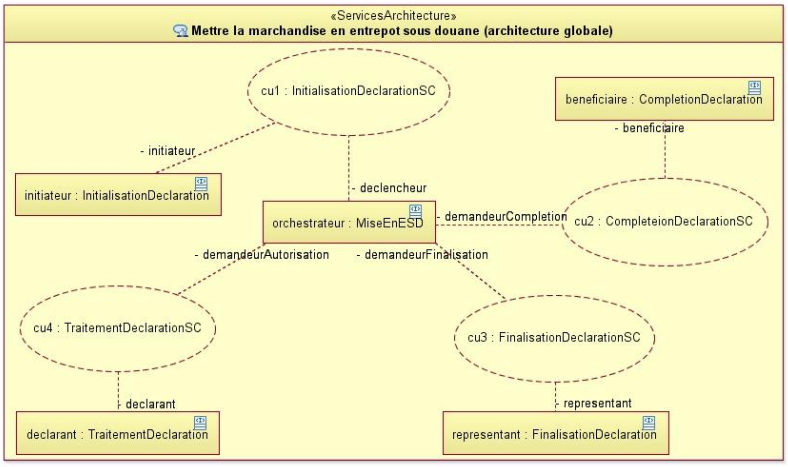


Figure V-21. Architecture globale des services publics de la collaboration

La Figure V-21 représente l'architecture globale des services publics. Cette architecture spécifie qutre contrats de services entre cinq participants. Le rôle de chaque participant est typé soit par le *Consumer* soit par le *Proivider*. Enfin, on constate qu'il y a une dépendance entre les participants (*Stockiste*, *Importateur* et *Commissionnaire*) qui utilisent la même interface *MiseEnESD* du *Mediateur*. Toute modification de cette interface affectera les trois participants. Par la suite on explique comment résoudre ce problème avec l'aide des interfaces de service *ServiceInterface*.

4.2. PIM : spécification détaillée des services

Dans la spécification détaillée, il faut raffiner les contrats de services qui spécifient les services évoluant dans des interactions bidirectionnelles avec des interfaces de services et des diagrammes de comportement UML. L'utilisation des interfaces de services précise pour chaque service comment il doit être consommé par son consommateur et implémenter par son fournisseur. Enfin, il faut aussi décrire la structure des données utilisées par les opérations des services.

4.2.1. Étape (8) : décrire les interfaces de services des interactions bidirectionnelles

L'interface de service indique les interfaces fournies et requises par le consommateur et le fournisseur du service ainsi que leurs rôles. Pour le consommateur, l'interface de service doit contenir toutes les informations décrivant comment utiliser le service. Alors que pour le fournisseur, elle doit spécifier ce qu'il faut connaître pour implémenter le service (Amsden, 2010). De plus, un diagramme de comportement UML peut être associé à l'interface de service pour spécifier comment les opérations du service sont utilisées dans l'interaction.

La spécification SoaML regroupe dans un tableau (OMG02, 2009 p. 94 et 95) les informations qui peuvent être utilisées pour spécifier une interface de service. Cependant dans cette méthode, on se limite à la description de ce que fait le service, dans quelle interaction évolue-t-il et avec quelles interfaces (i.e., interfaces fournies ou requises). On ne prend pas en compte des informations comme les contraintes de sécurité et la qualité des services qui sont des domaines qui dépassent largement le cadre de cette méthode.

Interface du service *InitialisationDeclarationSI*

La spécification SoaML autorise l'utilisation d'une interface de service pour décrire un service évoluant dans une interaction unidirectionnelle. Dans ce cas, l'utilisation d'une interface de service est redondante et n'apporte aucune information pertinente de plus qu'une interface simple. Dans la méthode qu'on propose seuls les contrats de services spécifiant des services évoluant dans des interactions bidirectionnelles sont à raffiner. Toutefois la Figure V-22 montre comment spécifier une interaction unidirectionnelle avec une interface de service. *InitialisationDeclarationSI* est une interface de service qui contient uniquement le rôle du fournisseur du service. En effet, contrairement au contrat de service la spécification du rôle du client (consommateur du service) n'est pas obligatoire dans l'interface de service.

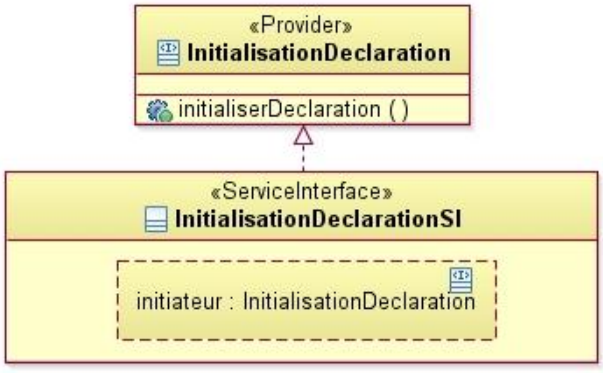


Figure V-22. Spécification du service *initialisation de la déclaration* avec l'interface de service nommée *InitialisationDeclarationSI*

Interface de service *CompletionDeclarationSI*

Après son initialisation la déclaration doit être complétée par l'*Importateur* dans un délai qui vari de quelques heures (quand la marchandise est expédiée par avion) à quelques jours (quand elle est expédiée par bateau). Donc, le *Mediateur* ne doit pas rester bloqué en attente de la réponse de l'*Importateur* . On a donc décidé d'utiliser deux scénarios d'interaction : *asynchrone avec callback* pour modéliser la demande de complétion et *asynchrone one-way* pour notifier la disponibilité de la marchandise.

Dans la Figure V-23 la relation d'usage de l'interface de service *CompletionDeclarationSI* avec l'interface *DemandeCompletionDeclaration* spécifie les exigences que le consommateur du service doit satisfaire. Alors que la relation de réalisation avec l'interface *CompletionDeclaration* spécifie les exigences nécessaires à l'implémentation du service par le fournisseur du service.

La dynamique de l'interaction est spécifiée avec un diagramme de séquence qui montre que le *Mediateur* initialise l'interaction en invoquant l'opération *completerDeclaration.* Ensuite, l'*Importateur* répond en invoquant l'opération *notifierCompletionDeclaration*. Enfin, après l'obtention de l'autorisation de mise en ESD le *Mediateur* signale la disponibilité de la marchandise a l'*Importateur* avec l'opération *marchandiseDisponible*.

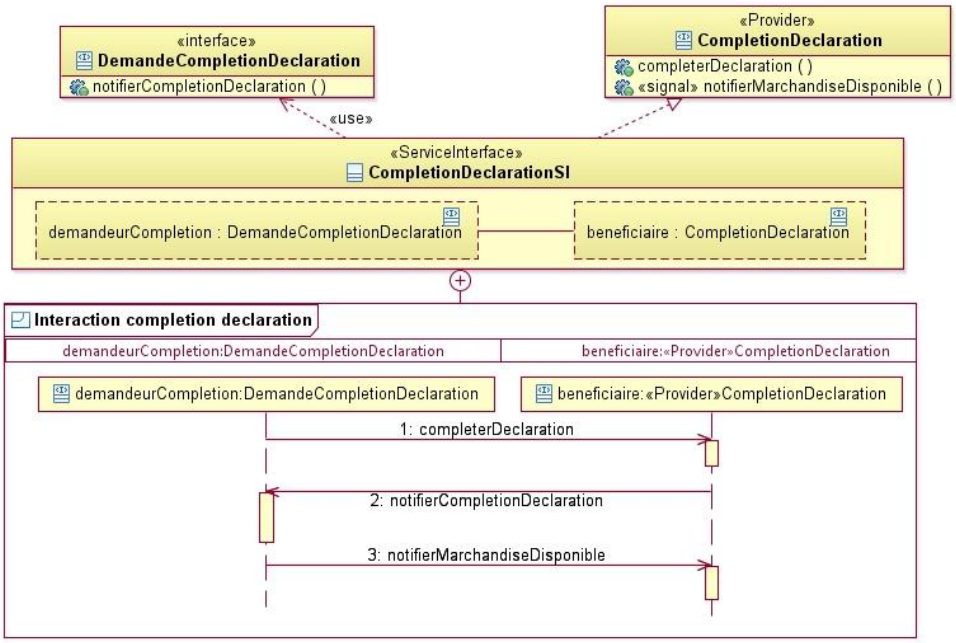


Figure V-23. Spécification du service *completion declaration* avec l'interface de service nommée *CompletionDeclarationSI*

Interface de service *FinalisationDeclarationSI*

La Figure V-24 décrit l'interface de service *FinalisationDeclarationSI*. Le *Mediateur* joue le rôle du demandeur de la finalisation *demandeurFinalisation* et le *Stockiste* joue le rôle du *representant* (de l'*Importateur*). Ce contrat de service spécifie deux scénarios d'interaction : le premier scénario a*synchrone avec Callback* est utilisé pour demander la finalisation de la déclaration. Dans cette interaction le *Stockiste* attend la réception de la marchandise pour finaliser la déclaration en indiquant l'emplacement de l'ESD dans lequel la marchandise sera stockée. Ainsi, le *Mediateur* initialise l'interaction en appelant l'opération *finaliserDeclaration* du *Stockiste*. Ensuit le *Stockiste* répond ultérieurement en invoquant l'opération *notifierFinalisationDeclaration* du *Mediateur*. Dans le deuxième scénario d'interaction *asynchrone one-way, le Mediateur* signale l'acceptation de l'autorisation de mise en ESD au *Stockiste* par l'opération *notifierAutorisationMiseEnESD*.

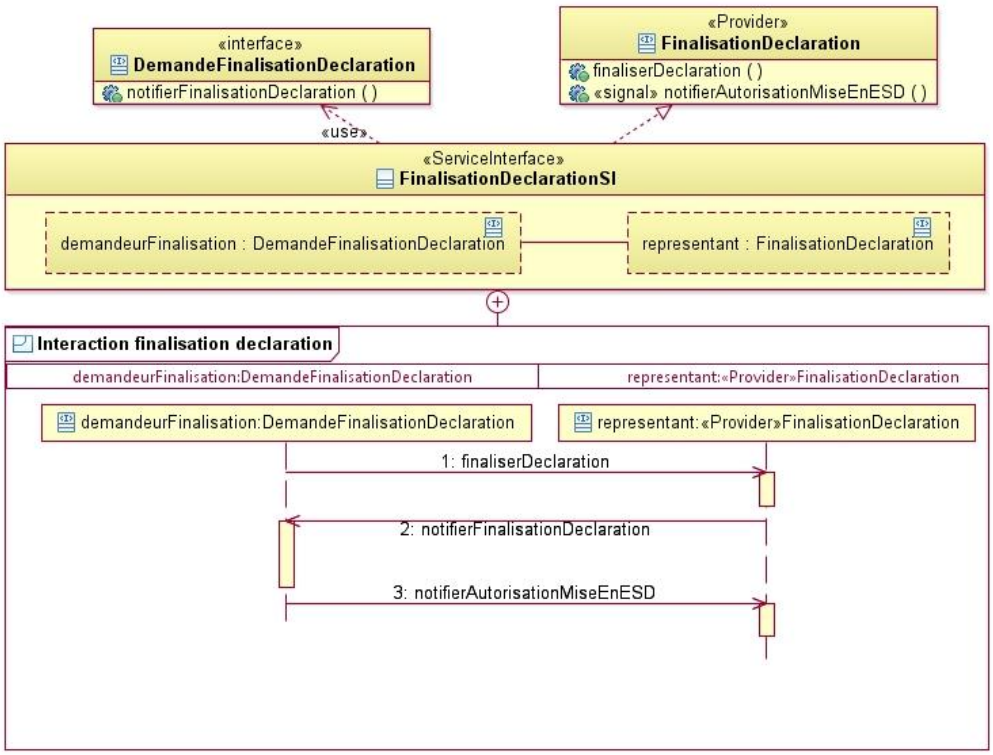


Figure V-24. Spécification du service *finalisation declaration* avec l'interface de service *nommée FinalisationDeclarationSI*

Interface du service *TraitementDeclarationSI*

Dans la Figure V-25 l'interface du service nommé *TraitementDeclarationSI* décrit une interaction bidirectionnelle à l'aide d'un diagramme de séquence. Le *Mediateur* initialise l'interaction en appelant l'opération *demanderAutorisationESD* du *Commissionnaire* qui répond après l'obtention de la réponse de douane en invoquant l'opération *notifierReponseDouane* du *Mediateur*.

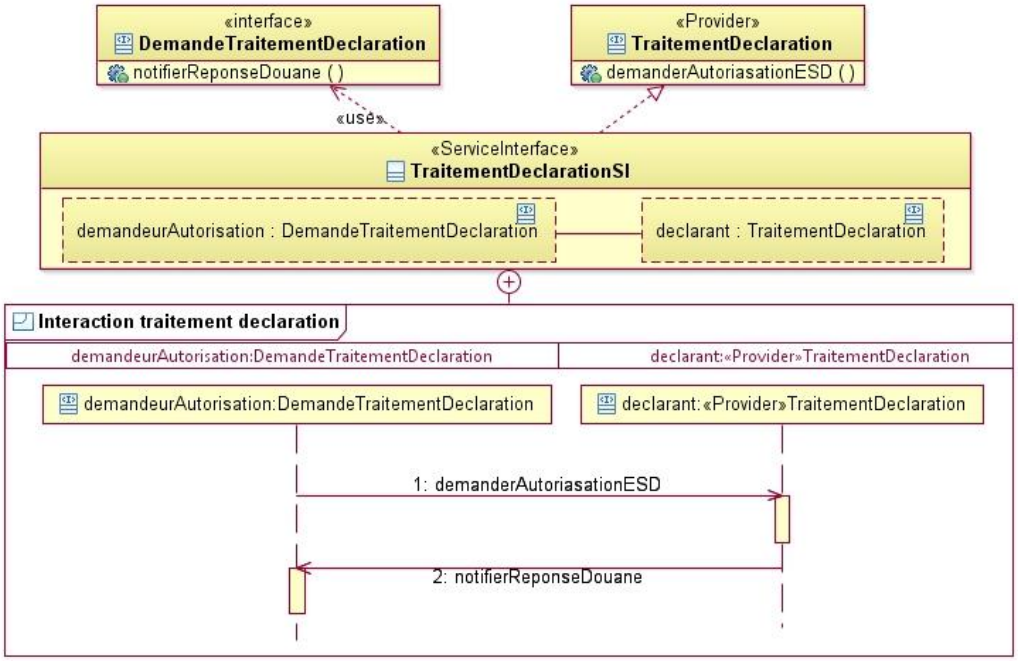


Figure V-25. Spécification du service *traitement declaration* avec l'interface de service nommée *TraitementDeclarationSI*

4.2.2. Étape (9) : spécifier les messages des données échangées entre les différents participants

Cette étape consiste à concevoir les données qui représentent les messages échangés entre les services des participants à travers les requêtes des consommateurs et les réponses des fournisseurs. L'élément SoaML *MessageType* est utilisé pour indiquer que les données ne font pas référence aux environnements spécifiques des participants (OMG02, 2009 p. 64). Les modèles de données des messages des services représentent une vue (ou une sélection) des modèles de données (généralement persistants) utilisés pour implémenter les services des participants (OMG02, 2009 p.65) Cependant, la relation entre les messages des services et les entités persistantes ne rentre pas dans le périmètre de cette méthode.

La Figure V-26 et la Figure V-27 représentent la structure des données échangées entre les participants de la collaboration. Dans cet exemple, les participants ont opté pour un format de données unifié basé sur un document technique de la douane qui décrit la structure d'une déclaration douanière (Montagne, 2007). Les *MessagesType Declaration* et *ReponseDeclaration* sont une simplification de la spécification d'une déclaration douanière. Ils contiennent les informations nécessaires pour établir une déclaration douanière. Alors que le *MessageType DeclarationInformation* contient les informations propres à la collaboration.

Il faut distinguer entre deux types de données : Les données renseignées par les participants pour établir la déclaration auprès de la douane et les données renseignées par la douane. Pour chaque participant il faut préciser les données qu'il doit fournir (ex : les données sur la marchandise qui sont renseignées par le *Stockiste).* Pour cela, on propose de mettre le nom de chaque participant devant les attributs qu'il doit renseigner. De même pour les attributs qui sont renseignés par la douane (ex : l'attribut *referenceDeclaration* qui représente une référence unique de la déclaration dans le système de la douane) on rajoute le mot douane devant eux.

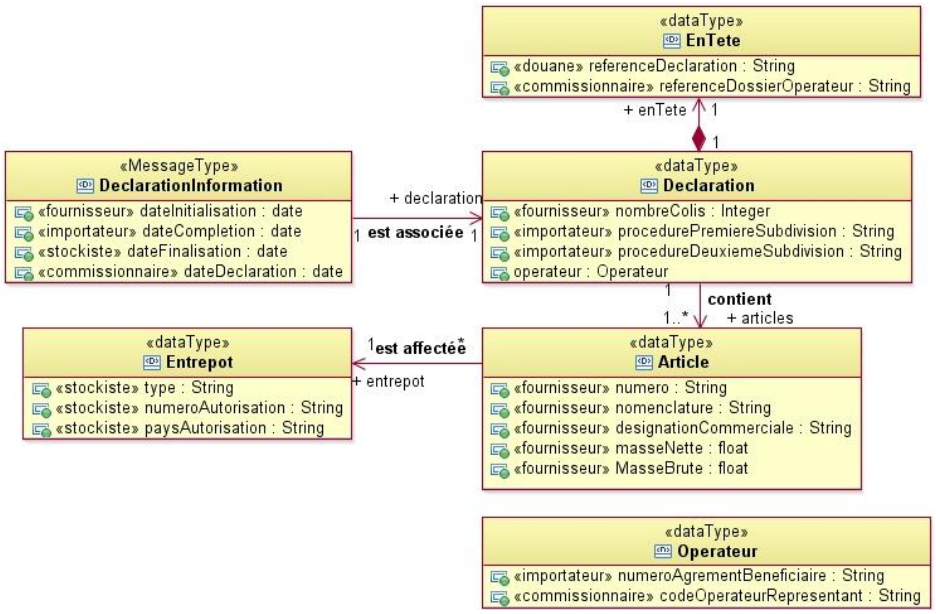


Figure V-26. Les données pour établir une de déclaration de mise en ESD

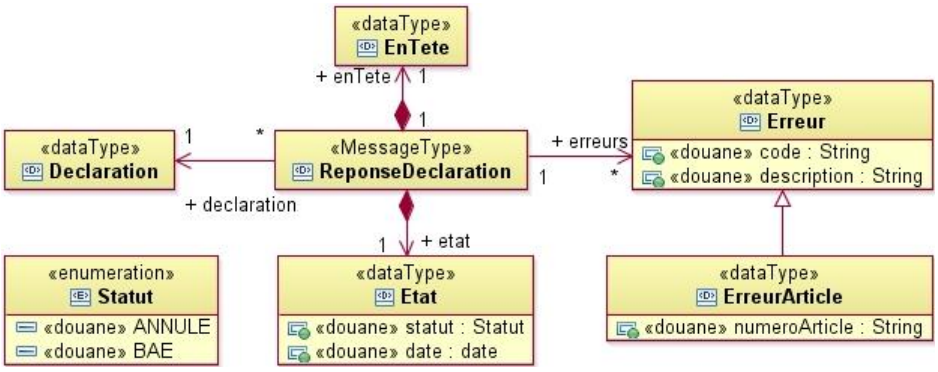


Figure V-27. Les données de la réponse de la demande déclaration de mise en ESD

Dans le document « *Delta-D documentation technique gestion des échanges XML* de la douane » (Montagne, 2007) la réponse de la demande de déclaration de la douane peut avoir plusieurs statuts. On se limite dans cet exemple à deux états : le statut BAE (*Bon À Enlevé*) si l'autorisation de mise en ESD est accordée par la douane, et le statut INVALID si la déclaration est invalidée par la douane. Le partage de l'état de la déclaration entre le service de la douane et son consommateur est une violation du principe SOA service sans état (cf. Chapitre I – 4.1.6). Car les

états d'exécution du service de douane sont préservés. Mais Amsden (2010) explique que trop souvent le principe de service sans état n'est pas respecté entre les participants qui fournissent ou consomment des services.

4.3. PIM : réalisation des services

L'objectif de la réalisation des services est de spécifier les composants de la solution SOA. Ce sont les modèles spécifiés à ce niveau qui sont transformés en modèles spécifiques à une plateforme ou en code exécutable. D'abord, il faut décrire les opérations des services. Puis, préciser les services fournis ou requis par chaque participant. Ensuite, connecter les participants fournisseurs et consommateurs des services. On commence par spécifier les services simples puis les services composites.

4.3.1. Étape (10) : spécifier les signatures des méthodes qui représentent les opérations des services

Dans le paradigme d'intégration à base de message les données sont encapsulées dans des messages. Chaque message est constitué d'un en-tête (*header*) et d'un corps (*body*) : l'en-tête contient des informations sur le message comme son origine et sa destination, etc., alors que le corps contient les données que le récepteur du message consomme. Les données contenues dans le corps du message peuvent être de différents styles (Roshen, 2009) :

- De style centré *Document* (ou *Message*) utilisé pour transférer les données d'une application à une autre.
- De style RPC (*Remote Procedure Calls)* utilisé pour invoquer une procédure (ou une méthode) dans l'application qui reçoit le message.
- De style message d'événement utilisé pour informer l'application qui reçoit le message du déclenchement d'un événement dans l'application qui envoie le message.

La spécification SoaML précise que les paramètres d'une opération de service définie dans une interface fournie ou requise par le service peuvent utiliser deux styles : (1) le style centré document dans lequel une opération du service peut avoir au plus un paramètre *In* et *Out*. Dans ce cas SoaML recommande de typer les paramètres de l'opération par des *MessageTypes* (OMG02, 2009 p.65). (2) Le style RPC dans lequel une opération du service peut avoir plusieurs paramètres du type *In*, *InOut*, *Out* ou un paramètre de retour du type UML *PrimitiveType* ou *DataType* (OMG02, 2009 p.65). Enfin, il est à noter que la spécification SoaML n'explique pas comment modéliser une opération de service dans le style de message d'événement.

En général, dans la plateforme des web-services, il est recommandé d'utiliser le style centré document. Cependant, il ne faut pas confondre le styles des données avec les styles de *Binding* (liaison) utilisés dans le WSDL pour lier le service au protocole de message (ex : SOAP). Butek (2005) élucide clairement les 5 styles de Binding existants : *RPC/encoded*, *RPC/literal*, *Document/encoded*, *Document/literal* et *Document/literal wrapped*. De ces 5 styles le dernier est le plus conforme au *Basic Profiles* fournit par l'organisation WS-I (*Web services Interoperability*) (WS-I, 2010).

4.3.2. Étape (11) : préciser quel participant utilise ou fournit quel service

L'identification des fournisseurs et des consommateurs des services est l'un des défis majeurs de la conception de services dans SOA. En effet, le choix d'un fournisseur de services plutôt qu'un autre affecte lourdement la mise en place de la solution orientée service. Dans la méthode dirigée par le métier qu'on propose l'identification des participants fournisseurs et consommateurs de services découle naturellement des processus de collaboration. Cependant, souvent d'autres considérations que les principes de conception SOA affectent le choix des fournisseurs de services comme l'emplacement, la gestion de la sécurité, la capacité d'exécution (i.e., compétences nécessaires pour mettre en place les services).

Jusqu'à maintenant, on s'est limité à l'utilisation du concept SoaML *Participant* pour décrire un fournisseur ou un consommateur de service. Cependant, ce concept ne permet pas de préciser si le participant fournit ou utilise des services. C'est pourquoi SoaML permet de caractériser un participant avec des ports *: Service* ou *Request*. Le port *Request* indique que le participant utilise un service, alors que le port *Service* indique que le participant offre un service. Enfin, un *Participant* peut posséder plusieurs ports, chacun peut être typé soit par une interface simple soit par une interface de service (i.e., *ServiceInterface*).

Le service *initilisationDeclarationService* du participant *Fournisseur*

La Figure V-28 décrit le service *initilisationDeclarationService* fournit par le *Fournisseur* via l'interface simple *InitilisaitonDeclaration*. L'utilisation de l'interface simple indique que le service évolue dans une interaction unidirectionnelle. Ce service est constitué d'une seule opération *initialiserDeclaration*.

Figure V-28. Ajout du service nommé *initialisationDeclarationService* au *Fournisseur*

Le service *completionDeclaration*Service du participant *Importateur*

La Figure V-29 représente le service *completionDeclarationService* de l'Importateur. Ce service est décrit avec l'interface de service *CompletionDeclarationSI,* il fournit l'interface *CompltionDeclaration* et requiert l'interface *DemandeCompletionDeclaration.* Ceci signifie que le fournisseur de ce service doit implémenter l'interface *CompletionDeclaration,* alors que son consommateur doit implémenter l'interface *DemandeCompletionDeclaration*.

Figure V-29. Ajout du service nommé *completionDeclarationService* à l'*Importateur*

Le service *finalisationDeclarationService* du participant *Stockiste*

Dans la Figure V-30 le service *finalisationDeclarationService* est typé par l'interface de service *FinalisationDeclarationSI* qui détermine comment le fournisseur (i.e., le *Stockiste*) doit l'implémenter et comment le consommateur (i.e., le *Mediateur*) doit l'utiliser.

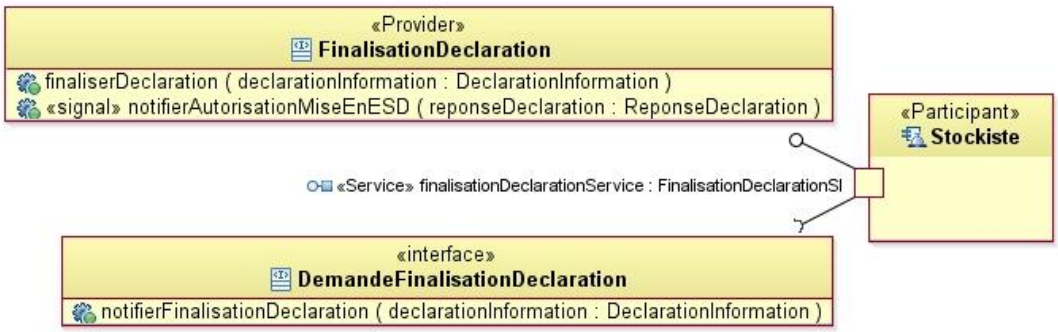


Figure V-30. Ajout du service nommé *finalisationDeclarationService* au *Stockiste*

Le service *traitementDeclarationService* du participant *Commissionnaire*

La Figure V-31 présente le service *traitementDeclarationService* du participant *Commissionnaire.* Ce service est décrit avec l'interface de service *TraitementDeclarationSI*. Le fournisseur de ce service (i.e., *Commissionnaire*) doit implémenter l'interface *TraitementDeclaration,* alors que son consommateur (i.e., le *Mediateur*) doit implémenter l'interface *DemandeTraitementDeclaration*.

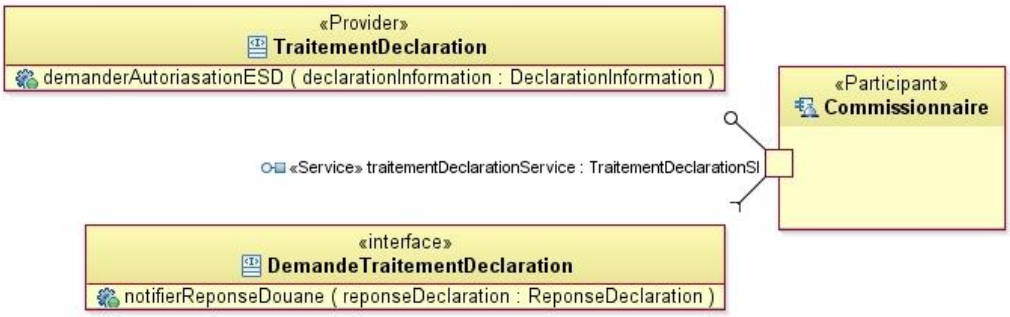


Figure V-31. Ajout du service nommé *TraitementDeclarationService* au *Commissionnaire*

Le service *declarationService* du participant *Mediateur*

Dans une coopération avec une vue distribuée dans le cadre d'une approche d'interopérabilité unifiée le médiateur orchestre l'ensemble des services de la collaboration. Donc, le participant *Mediateur* fournit un service qui orchestre les services des autres participants : *Fournisseur*, *Importateur*, *Stockiste* et *Commissionnaire*.

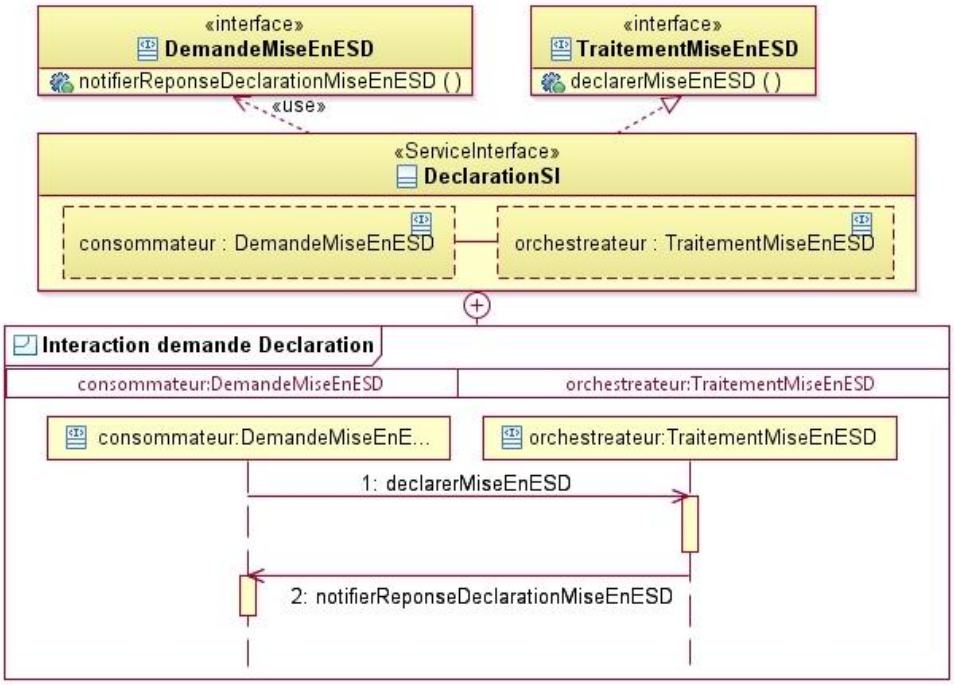


Figure V-32. Spécification du service *déclaration* avec l'interface de service *DeclarationSI*

Le participant *Mediateur* fournit l'interface de service *DeclarationSI* (Figure V-32). Pour faire une déclaration de mise en ESD le consommateur du service de déclaration passe par un port typé par cette interface de service. Ceci permet de limiter le couplage entre le consommateur de ce service et les services des participants qui collaborent pour obtenir la déclaration. Le consommateur du service de déclaration dépend uniquement du service fournit par le *Mediateur* ce qui favorise le respect du principe de couplage faible de SOA (cf. Chapitre I – 4.1.2). En effet,

l'invocation de l'opération *declarationMiseEnESD* déclenche une orchestration de service entre les participants de la collaboration ensuite le résultat de l'orchestration est notifié au consommateur du service en invoquant l'opération *notifierReponseDeclarationMiseEnESD*. Deux vues peuvent être utilisées pour représenter le participant *Mediateur :*

Une vue externe : cette vue représentée dans la Figure V-33 est destinée aux autres participants de la collaboration, elle représente ce qui est visible à l'extérieur du participant *Mediateur*.

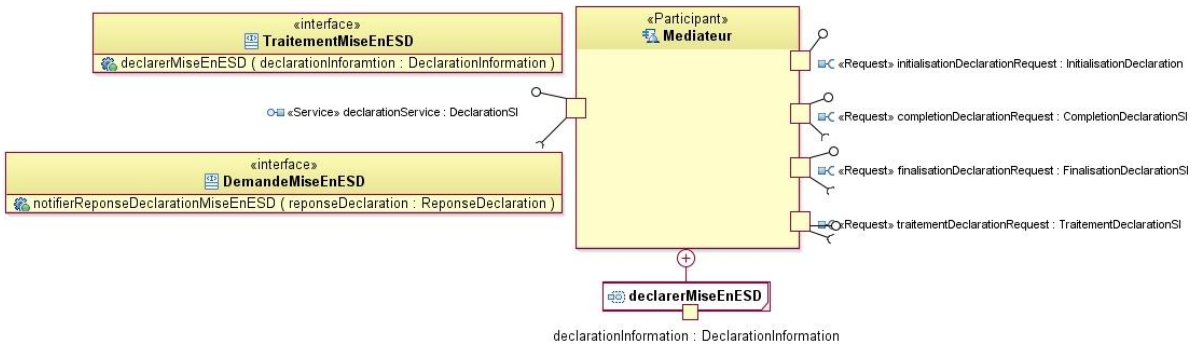


Figure V-33. Vision du participant *Mediateur (Black box vue)*

Le *Mediateur* fournit un seul service (i.e., *declarationService*) qui est défini par l'interface de service *DeclarationSI* et requiert quatre services définis par les interfaces *InitialisationDeclaration*, *CompletionDeclarationSI*, *FinalisationDeclarationSI*, *TraitementDeclarationSI*.

Une vue interne : cette vue représentée dans la Figure V-34 concerne la structure interne du participant *Mediateur*. Elle est constituée des ports *Service* et *Request* des interfaces fournies et requises et des propriétés internes du *Mediateur*. L'attribut *id* sert à identifier les instances du service. Il peut être utilisé pour gérer la corrélation entre le fournisseur et le consommateur du service. Quant à l'attribut *declarationInformation,* il contient des informations utilisées dans l'implémentation de l'opération *déclarerMiseEnESD* du service *declarationService*.

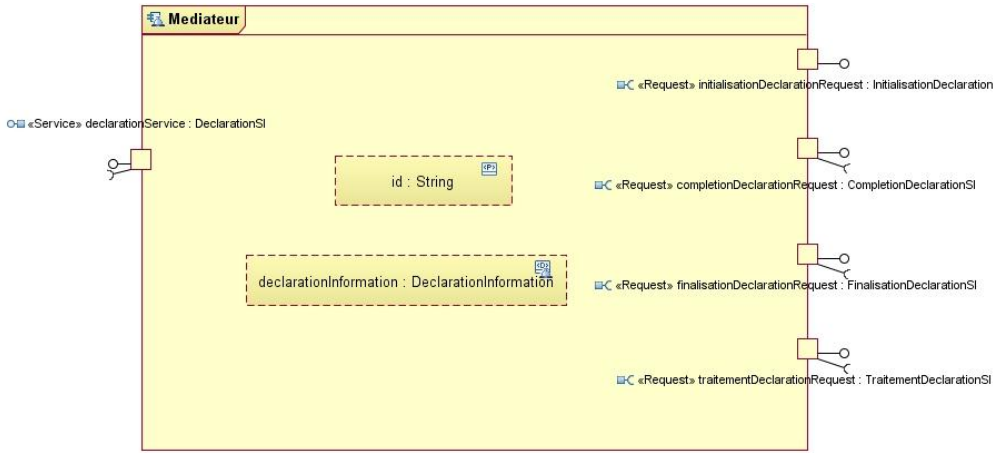


Figure V-34. Vision interne du participant *Mediateur* (*White Box View*)

Spécification de l'opération *declarationMiseEnESD* du service *declarationService*

Après avoir déterminé la structure du service *declarationService*. Il faut spécifier la méthode qui représente l'opération *déclarerMiseEnESD*. Ce diagramme permet de préciser comment l'opération *déclarerMiseEnESD* orchestre l'appel aux opérations des autres services.

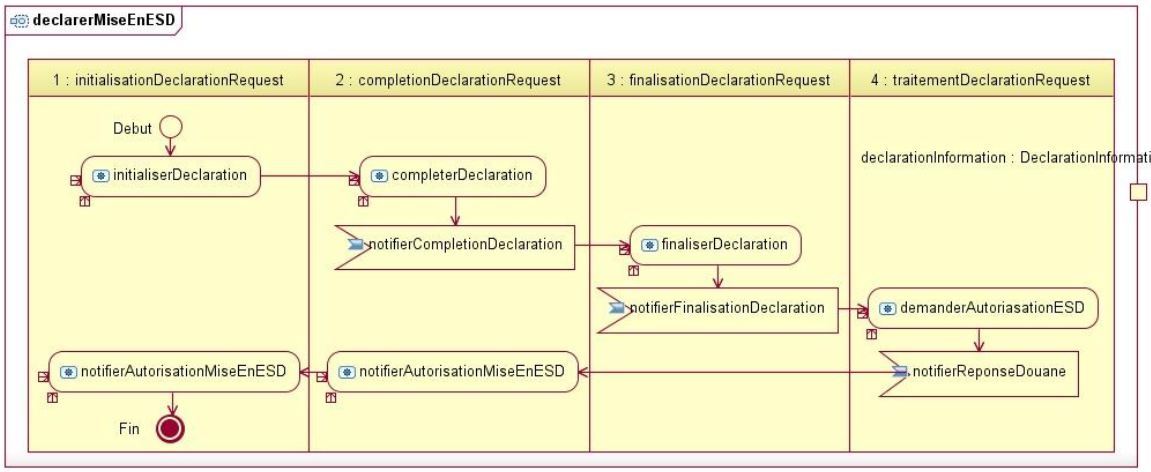


Figure V-35. L'implémentation de l'opération *declarationMiseEnESD* du service *declarationService* fournit par le *Mediateur*

Le diagramme d'activité *declarerMiseEnESD* de la Figure V-35 décrit l'opération *declarerMiseEnESD* du service *declarationService* . Chaque partition du diagramme d'activité est typée par les ports *Request* du participant *Mediateur*. Chaque opération de service fournie par un fournisseur de services doit être réalisée par un comportement ou par une action UML (OMG02, 2009 p.150) :

- Par un comportement UML pour représenter la méthode de l'opération du service de la partition consommée par l'opération *declarationService*.
- Par les éléments UML *AcceptEventAction* (pour les appels asynchrones) ou *AcceptCallAction* (pour les appels synchrones Request/Reponse) afin de représenter une activité du *Mediateur*.

Le diagramme d'activité UML de la Figure V-35 spécifie l'opération *declarationMiseEnESD* de l'interface *TraitementMiseEnESD*. L'élément UML *CallOperationAction* est utilisé pour représenter les opérations propres à chaque partition. Alors que l'élément *AcceptEventAction* est utilisé pour représenter les opérations du composant *Mediateur*.

4.3.3. Étape (12) : assembler les services pour concevoir la solution SOA dirigée par le métier

Dans le but de fournir une solution complète aux exigences métiers de la collaboration. On montre dans cette étape, comment assembler les services des participants. Ainsi, après avoir

spécifié le composant du participant *Mediateur,* on le connecte maintenant aux services des autres participants.

Comme on l'a expliqué auparavant (cf. Chapitre I – 4.1.8) la forme des composants logiciels change d'un paradigme à l'autre. Dans la conception des services composites on propose de distinguer deux types de compositions:

1. La composition inspirée du paradigme de conception orientée composante. Dans cette composition le service composite contient ces services composants.
2. La composition introduite dans le paradigme de conception orienté services. Dans cette composition le service composite orchestre des services composants externes. Dans ce cas, le service composite est réalisé via un processus d'orchestration dont le flux d'exécution appelle les services composants *stateless* externes via des interfaces bien définies. Cette composition est aussi appelée orchestration et concerne en général les services métiers. Les langages WS-BPEL et BPMN sont bien adaptés pour spécifier l'orchestration des services.

Par la suite on explique comment spécifier ces deux types de composition.

Exemple de conception d'un service composite qui contient des services composants

Dans ce type de composition, il faut créer un nouveau participant pour assembler le *Mediateur* et tous les participants qui fournissent des services à ces requêtes. Ceci permet de connecter le *Mediateur* à l'architecture de service qui modélise les exigences métiers. La Figure V-36 montre comment assembler le nouveau participant *Acheteur*. Cet assemblage permet de connecter les services (i.e., *Service*) et les requêtes (i.e., *Request*) compatibles des participants avec des *ServiceChannels*.

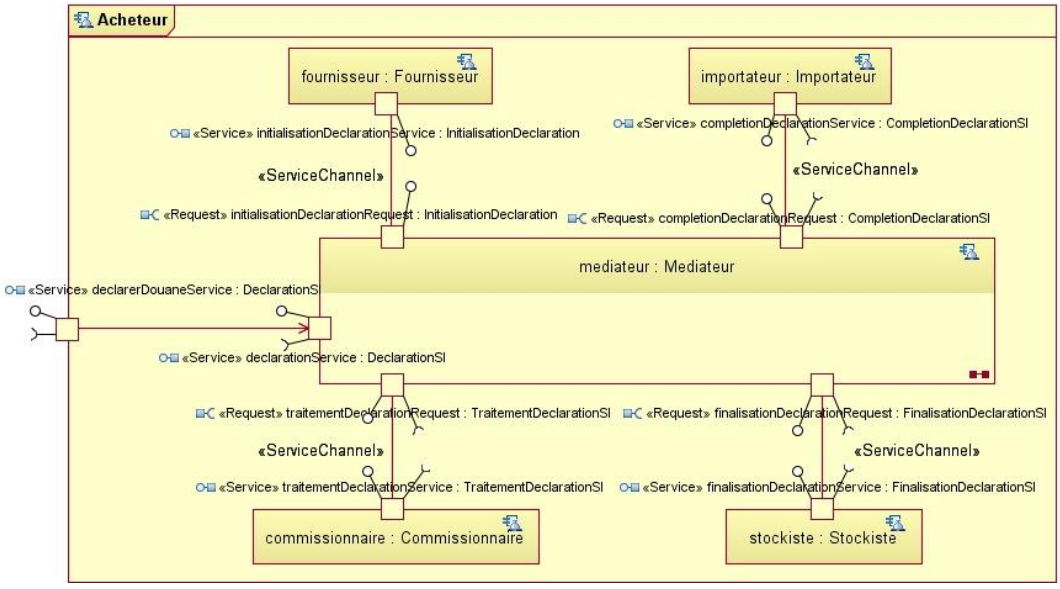


Figure V-36. Assemblage des parties du participant *Acheteur*

La Figure V-36 représente la structure interne du participant *Acheteur* qui assemble des parties référençant les instances des participants *Mediateur*, *Fournisseur*, *Importateur*, *Stockiste* et *Commissionnaire*. Le service *declarerDouaneService* de l'*Acheteur* est lié au service *declarationService* du *Mediateur* avec un connecteur de type *DelegationConnector* pour indiquer qui *l'Acheteur* délègue l'implémentation de son service au service de son participant interne *Mediateur*.

La requête *initialisationDeclarationRequest* du *Mediateur* est connectée au service *initialisationDeclarationService* du *Fournisseur* par un *ServiceChannel*. Cette connexion est valide, car l'interface *InitialisationDeclaration* est utilisée par les ports *Request* du *Mediateur* (le consommateur du service) et *Service* du *Fournisseur* (fournisseur du service). Les ports *Request* et *Service* des autres participants sont reliés de la même manière pour indiquer l'accord des participants pour s'interagir en respectant les interfaces (interfaces simples ou interfaces de services). Ceci permet de répondre aux exigences métier tout en réduisant le couplage entre les participants.

Le diagramme de la Figure V-37 montre comment le participant *Acheteur* adhère à l'architecture détaillée des services publics. Ce diagramme permet de faire le lien entre l'architecture d'entreprise et l'architecture d'application (cf. Chapitre I - 1). Les parties définies dans l'assemblage de l'*Acheteur* (cf. Figure V-36) sont attachées aux rôles qu'elles jouent dans l'architecture des services afin de garantir que la solution respecte les principes de l'architecture d'entreprise.

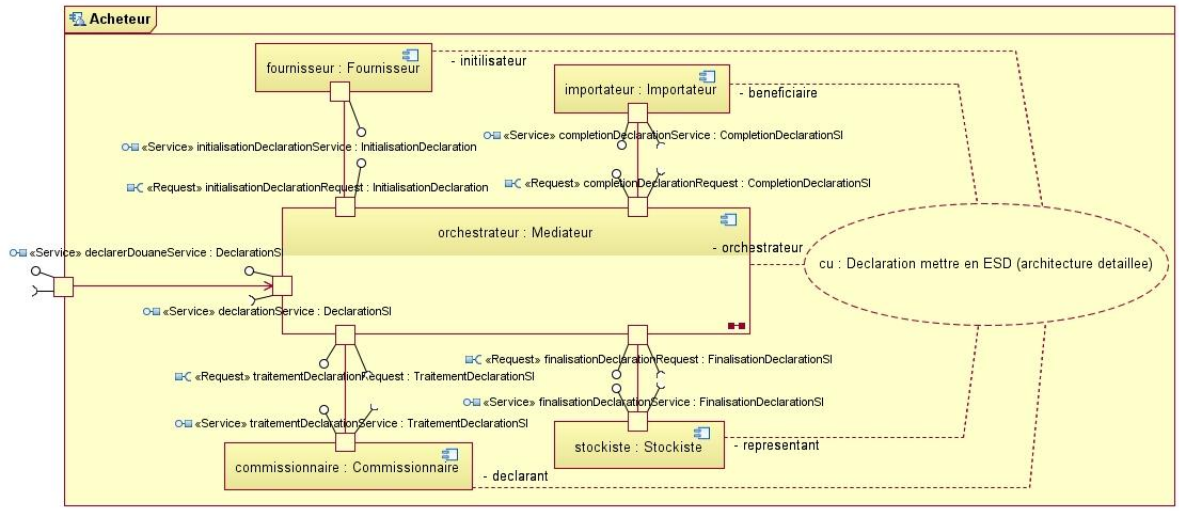


Figure V-37. Une instance de l'architecture de service de l'*Acheteur*

Exemple de conception d'un service qui orchestre d'autres services

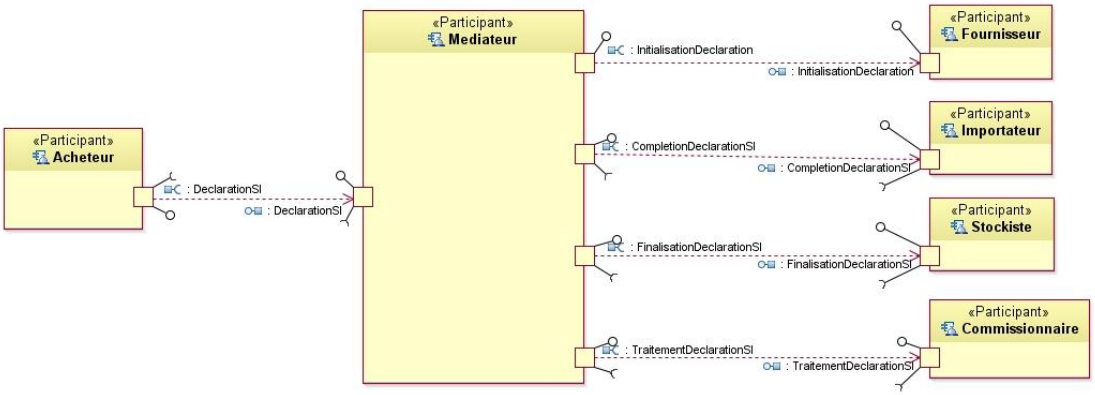


Figure V-38. Orchestration des services par le *Mediateur*

Dans la Figure V-38 l'Acheteur se contente d'invoquer le service *declarationService* fournit par le *Mediateur* au lieu d'assembler les services publics de la collaboration. C'est au *Mediateur* d'orchestrer l'invocation des services publics des différents participants. Ceci permet un couplage faible entre l'*Acheteur* et les services qui évoluent dans la collaboration. Ce type de composition (i.e., l'orchestration) respecte mieux le principe de couplage faible de SOA (cf. Chapitre 1 – 4.1.2) et donc apporte plus de flexibilité car l'ajout d'un nouveau participant dans la collaboration n'affecte pas forcément le participant *Acheteur*. Enfin, il est à noter que les flèches entre les ports des participants indiquent les relations de dépendances entre les participants.

4.3.4. Étape (13) : décrire l'architecture détaillée de la collaboration

Cette étape consiste à utiliser les composants de la solution qu'on a spécifiés dans les étapes précédentes pour raffiner l'architecture globale des services publics (cf. l'étape 7). La Figure V-39 présente l'architecture détaillée des services publics et précise comment les participants remplissent les exigences métier. Elle spécifie les ports utilisés par les participants dans chaque contrat de services.

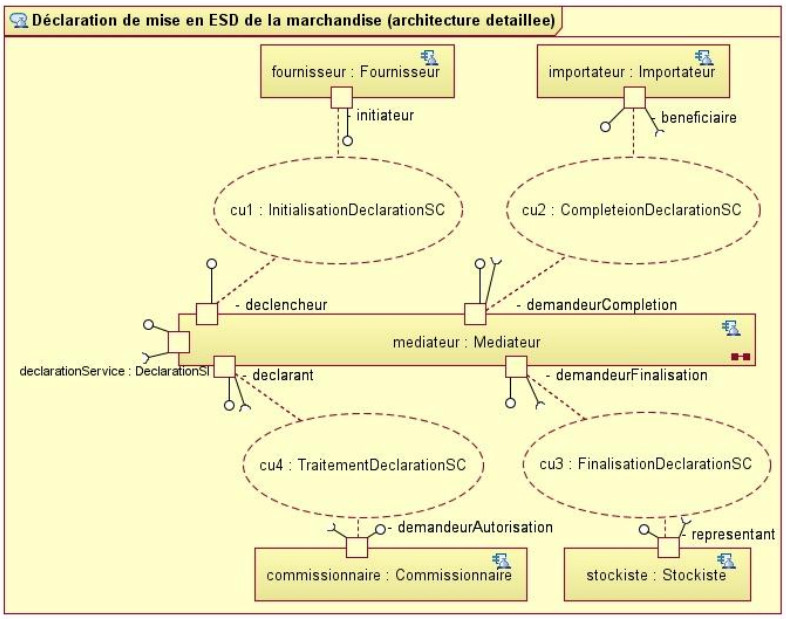


Figure V-39. Architecture détaillée des services publics de la collaboration

Puisque les dépendances entre les participants sont toutes au niveau des ports, tous les services sont fournis ou consommés via des ports afin de réduire le couplage entre les fournisseurs et les consommateurs des services. Si une interface qui définit un service est modifiée seul le port qui est typé par cette même interface doit être examiné (Amsden, 2010).

4.4. PSM : Marquage des modèles de la solution SOA pour les rendre spécifiques à une plateforme

4.4.1. Étape (14) : marquer les modèles pour les rendre spécifiques à une plateforme

Après l'identification des services à partir des exigences métier (niveau CIM-Bas), puis leur spécification et réalisation (niveau PIM), le niveau PSM consiste à produire des modèles de services spécifiques à une plateforme cible. Une technique consiste à marquer les modèles PIM avec un profil pour ajouter les informations nécessaires à la transformation des modèles dans une plateforme donnée (Bonnet et al., 2004). Ce sont ces modèles marqués qui sont par la suite utilisés pour générer le code (i.e., le squelette de la solution SOA). Cependant, l'utilisation des modèles PIM n'est pas toujours indispensable. La génération du code peut se faire directement à partir des modèles PIM vers la plateforme cible.

4.5. Code : génération du squelette de la solution SOA

Des outils comme RSA sont capable de générer le squelette de la solution SOA directement à partir des modèles du niveau PIM. Dans le but de fournir un exemple de bout en bout on présente le résultat de la transformation des modèles SoaML en code spécifique à la plateforme des web-services. L'outil RSA a été utilisé pour générer le squelette de la solution SOA (cf. Annexe C).

4.5.1. Étape (15) : compléter le squelette de la solution SOA

Trois langages de la plateforme des web-services ont été pris pour cibles : (1) Le langage XSD pour la définition des messages des services, (2) le langage WSDL pour la définition des interfaces de services et (3) le langage WS-BPEL pour la définition de l'orchestration des service. Ensuite, le code généré doit être compéter avec la logique métier et déployer dans une plateforme orientée services.

Transformation vers le langage XSD

La Figure V-40 montre les éléments XSD qui ont été générés pour chaque élément SoaML *MessageType* et *dataType* du modèle de données source.

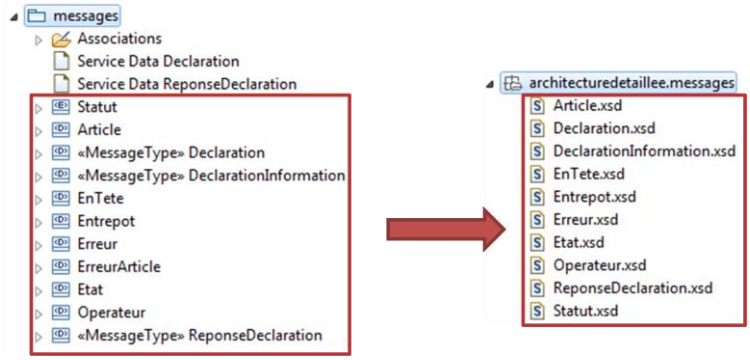


Figure V-40. Les éléments XSD générés à partir du modèles de donné

À titre d'exemple Figure V-41 représente le code de l'élément XSD *DeclarationInformation* généré à partir du *MessageType DeclarationInformation*.

```
<?xml version="1.0" encoding="UTF-8"?><xsd:schema targetNamespace="http://architecturedetaillee/messages/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:xsd_1="http://architecturedetaillee/messages/">
  <xsd:include schemaLocation="Declaration.xsd"/>
  <xsd:complexType name="DeclarationInformation">
    <xsd:sequence>
      <xsd:element name="dateInitialisation" type="xsd:date"/>
      <xsd:element name="dateCompletion" type="xsd:date"/>
      <xsd:element name="dateFinalisation" type="xsd:date"/>
      <xsd:element name="dateDeclaration" type="xsd:date"/>
      <xsd:element name="declaration" type="xsd_1:Declaration"/>
    </xsd:sequence>
  </xsd:complexType>
</xsd:schema>
```

Figure V-41. Exemple de l'éléments XSD *DeclarationInformation*

Transformation vers le langage WSDL

Une définition WSDL est générée pour chaque interface UML. Au total huit fichiers wsdl sont générés (Figure V-42) :

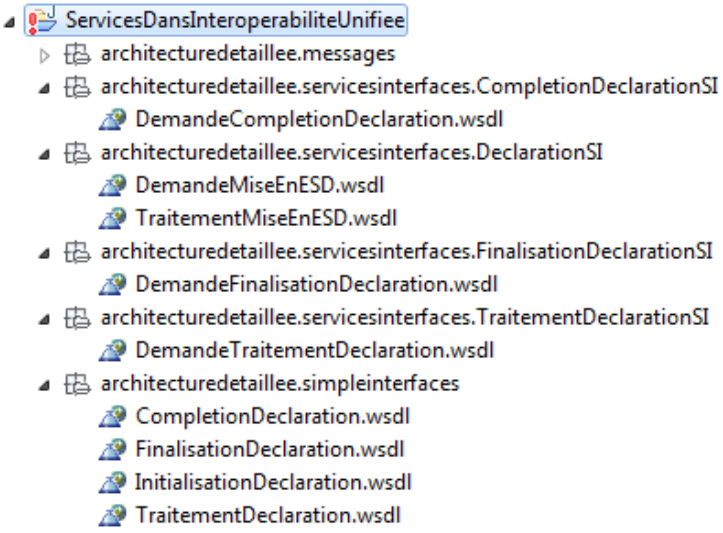


Figure V-42. Les définitions WSDL générés à des modèles de spécification SoaML

Transformation vers le langage WS-BPEL

Un diagramme d'activité a été utilisé pour décrire le comportement de l'opération *declarerMiseEnESD* du service *DeclarationService* (cf. Étape 11). Ce diagramme qui représente l'orchestration des opérations des services de la collaboration par le *Médiateur* est transformé en processus WS-BPEL exécutable. Ce processus utilise les définitions WSDL et les éléments XSD pour invoquer les opérations des services de la collaboration. Le code généré de ce processus est présenté en Annexe D.

5. CONCLUSION

Dans ce chapitre, on a voulu atteindre l'objectif de montrer un exemple complet de spécification des services au sens SOA, depuis l'identification des services à partir des exigences métiers, spécifiées avec des processus métier, jusqu'à la génération du squelette de la solution SOA.

Afin d'apporter une nouvelle preuve de la complémentarité entre les approches BPM et SOA, on a proposé un mapping entre les éléments BPMN et SoaML sémantiquement proches. Puis, on a réalisé une transformation de modèles entre ces deux formalismes. Le résultat de cette transformation a guidé le choix des concepts SoaML utilisés dans la méthode qu'on propose.

Par ailleurs, les différents types de modèles de processus du niveau PIM utilisés pour identifier les services influencent la modélisation de l'architecture des services. Par exemple, la Figure V-43 représente une architecture détaillée des services publics spécifiée à partir d'une coopération avec une vue distribuée dans le cadre d'une approche d'interopérabilité fédérée (cf. Annexe E). Ainsi, contrairement à l'architecture des services présentée auparavant le participant *Mediateur* n'apparaît pas dans cette architecture de services. On constate que le type de collaboration et l'approche d'interopérabilité influencent aussi la conception de l'architecture des services.

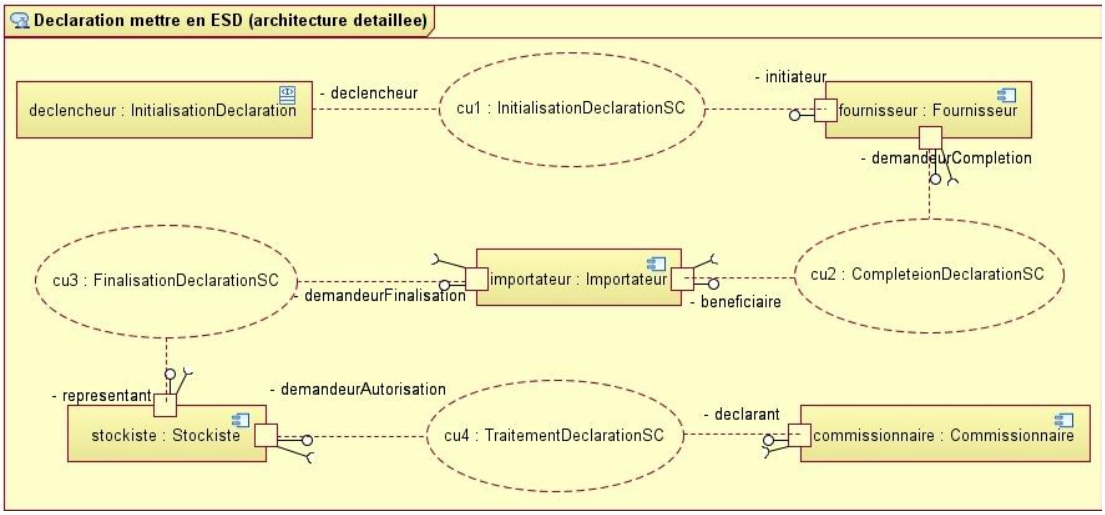


Figure V-43. Architecture détaillée des services publics

De plus, des considérations techniques indépendantes de la plateforme ou de la technologie comme le type d'interactions dans lequel évolue le service influencent la spécification détaillée des services. La Figure V-44 représente le scénario d'interaction globale dans une coopération avec une vue distribuée dans le cadre d'une approche d'interopérabilité fédérée. Ce diagramme est complètement différent de celui qu'on a étudié (cf. Figure V-21).

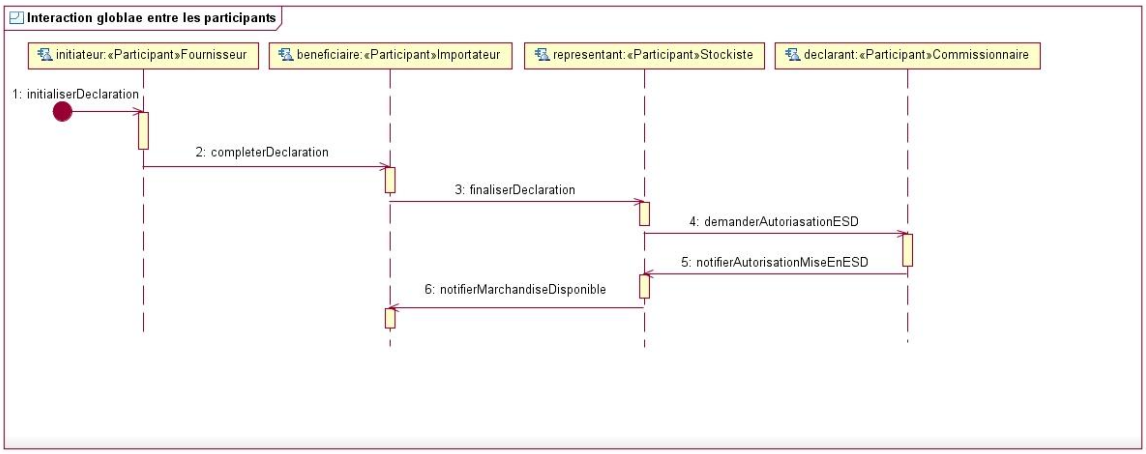


Figure V-44. Vue global sur les interactions entre les participants

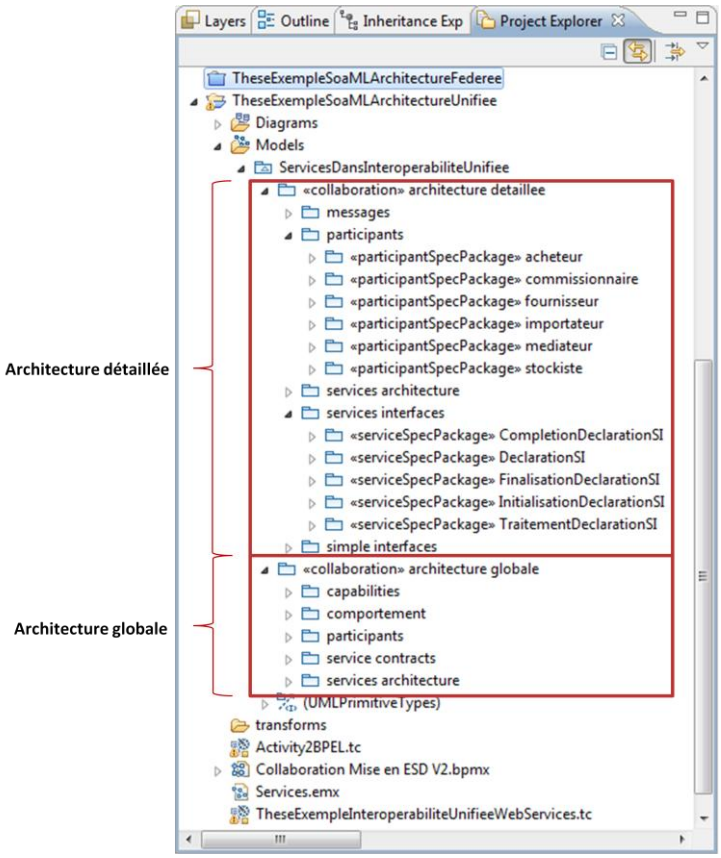


Figure V-45. La structure du projet avec les deux perspectives globale et détaillée

Enfin, la Figure V-45 représente la structure du projet de modélisation des services. Deux vues d'architecture différentes (globale et détaillée) sont utilisées. Chacune fait usage d'un ensemble d'éléments SoaML.

Conclusions et perspectives

1. CONCLUSION GÉNÉRALE

Les organisations contemporaines collaborent de plus en plus avec leurs partenaires. Cette dimension ouverte permet aux organisations d'être plus réactives face aux changements que leur imposent leurs environnements ainsi que dans leurs capacités à identifier les besoins et les attentes de leurs partenaires. La caractéristique de la collaboration moderne est due, pour les entreprises, au nouvel environnement économique mondial, qui fait de l'interopérabilité et l'agilité deux des principaux résultats que les entreprises collaboratives doivent atteindre (Bourey et al., 2007; Luftman et al., 2009). Ce contexte correspond au cadre global de nos travaux, qui porte sur la question de recherche suivante : Comment concevoir une architecture orientée services dirigée par le métier dans le cadre d'une collaboration inter-organisationnelle ?

Dans ces travaux de thèse, on décline la réponse à cette question en trois étapes :

- **Étape (1) :** On élucide comment l'architecture orientée service dirigée par le métier contribue à résoudre les défis associés aux problématiques d'agilité (alignement métier-IT) et d'interopérabilité. Pour cela, on adopte deux perspectives de l'architecture orientée services. La perspective architecture et conception pour expliquer les domaines d'application de SOA, notre vision de la notion de service et les principes et patterns de SOA (cf. Chapitre I). Puis, la perspective métier pour mettre en évidence le principe d'alignement métier-IT à travers la relation entre les notions de processus métier et de service métier (cf. Chapitre II).
- **Étape (2) :** On positionne notre travail sur les barrières conceptuelles et techniques des niveaux processus et services du cadre d'interopérabilité EIF (*Entreprise Interoperability Framework*) (cf. Chapitre III). Puis, on propose une grille qui met en évidence les dimensions à prendre en compte pour une mise en place réussie de l'architecture orientée service dirigée par le métier dans le cadre d'une collaboration.
- **Étape (3) :** À travers un cas d'étude, on propose une méthode de conception descendante d'une solution orientée services et dirigée par le métier. La méthode proposée prend en compte les principes des deux perspectives métier et architecture de conception de SOA et les deux aspects processus et services du cadre d'interopérabilité d'ATHENA. On se focalise sur l'effort d'analyse et de conception de processus et de services de collaboration (cf. Chapitres IV et V).

Nos travaux sur la spécification d'une architecture orientée service s'inscrivent dans un courant qui cherche à spécifier les services au sens SOA à partir des perspectives métier et IT. Comme les différentes méthodes proposées par (Casanave, 2009; Amsden, 2010; Elvesæter et al., 2010) on cherche à identifier et à spécifier les services au sens SOA. Cependant, dans notre

méthode on s'intéresse uniquement à la spécification des services publics dans le cadre d'une collaboration. Dans ce but, on détaille comment spécifier les processus de la collaboration puis les dériver pour identifier les services publics de la collaboration. En effet, l'un des résultats fondamentaux de ce travail de thèse est la mise en évidence de l'influence des types de collaboration (coordination, la coopération avec une vue centralisée, la coopération avec une vue distribuée) et des approches d'interopérabilité (intégrée, unifiée, fédérée) sur la conception des modèles de processus collaboratifs et des services publics de collaboration. Ces choix affectent même la mise en place de la plateforme qui supporte l'interopérabilité.

La méthode qu'on propose s'articule autour de vingt-deux étapes qui se déploient sur les différents niveaux de l'approche MDI. Elle se base sur un style de modélisation hiérarchique avec des diagrammes de haut niveau qui sont ensuite enrichis à des niveaux plus bas. Le formalisme BPMN 2.0 est utilisé pour modéliser les processus, alors que le langage SoaML est utilisé pour spécifier les services

Nos travaux portent sur la modélisation des processus de collaboration. D'abord, on explique comment modéliser des processus qui représentent la collaboration d'un point de vue global (niveau CIM-Haut). L'objectif principal de ces modèles est la compréhension du contexte et la spécification du besoin d'interopérabilité. Ensuite, on montre comment raffiner ces modèles pour représenter les exigences métiers des applications d'entreprise afin d'obtenir une image réduite aux besoins métiers à automatiser (niveau CIM-Bas). Les modèles du niveau CIM-Bas sont ensuite détaillés au niveau PIM pour prendre en compte la logique des flux des processus. À ce stade les participants doivent choisir un type de collaboration et une approche d'interopérabilité cibles afin d'identifier quel participant contrôle et exécute quels processus. Enfin, les modèles PIM peuvent être par la suite enrichis avec du code spécifique à une plateforme afin de les rendre exécutables dans un BPMS. Sinon, une autre solution consiste à transformer les modèles PIM vers un langage de modélisation de processus exécutable comme WS-BPEL.

Une autre partie de nos travaux porte sur l'identification, la spécification et la réalisation des services au sens SOA. L'identification des services doit être réalisée à partir les processus collaboratifs du niveau PIM. On estime que c'est à ce niveau que les processus sont suffisamment aboutis pour être utilisés comme une base des spécifications métier. En effet, les processus du niveau PIM spécifient qui (des participants) contrôle quel processus ou activité de processus. Ce qui permet par la suite d'identifier les fournisseurs et les consommateurs des services dans l'architecture des services.

Quant à la spécification des services, elle concerne uniquement les services automatisables qui forment l'inventaire de services de l'entreprise. C'est pourquoi la modélisation des services commence à partir du niveau CIM-Bas de l'approche MDI. C'est à ce niveau qu'il faut décrire l'architecture globale des services publics de la collaboration qui spécifie les contrats de services et les rôles des participants de la collaboration. L'architecture des services représente un deuxième point de vue sur les exigences métiers. Ensuite, au niveau PIM il faut d'abord raffiner le contrat de services de chaque service évoluant dans une interaction bidirectionnelle avec une interface de service. De plus, il faut spécifier les messages échangés entre les participants à travers les requêtes des consommateurs et les réponses des fournisseurs. Ensuite, un deuxième raffinement des modèles consiste à réaliser les services en décrivant les opérations de chaque service fourni ou requis par les participants. Ceci permet de connecter les participants fournisseurs et consommateurs des services. Enfin, ou bien les modèles de services du niveau PIM sont transformés directement pour générer le squelette de la solution SOA dans une plateforme cible (ex : web-services), ou bien ils sont marqués au niveau PSM pour les rendre spécifiques à une plateforme cible avant de pouvoir générer le squelette de la solution SOA.

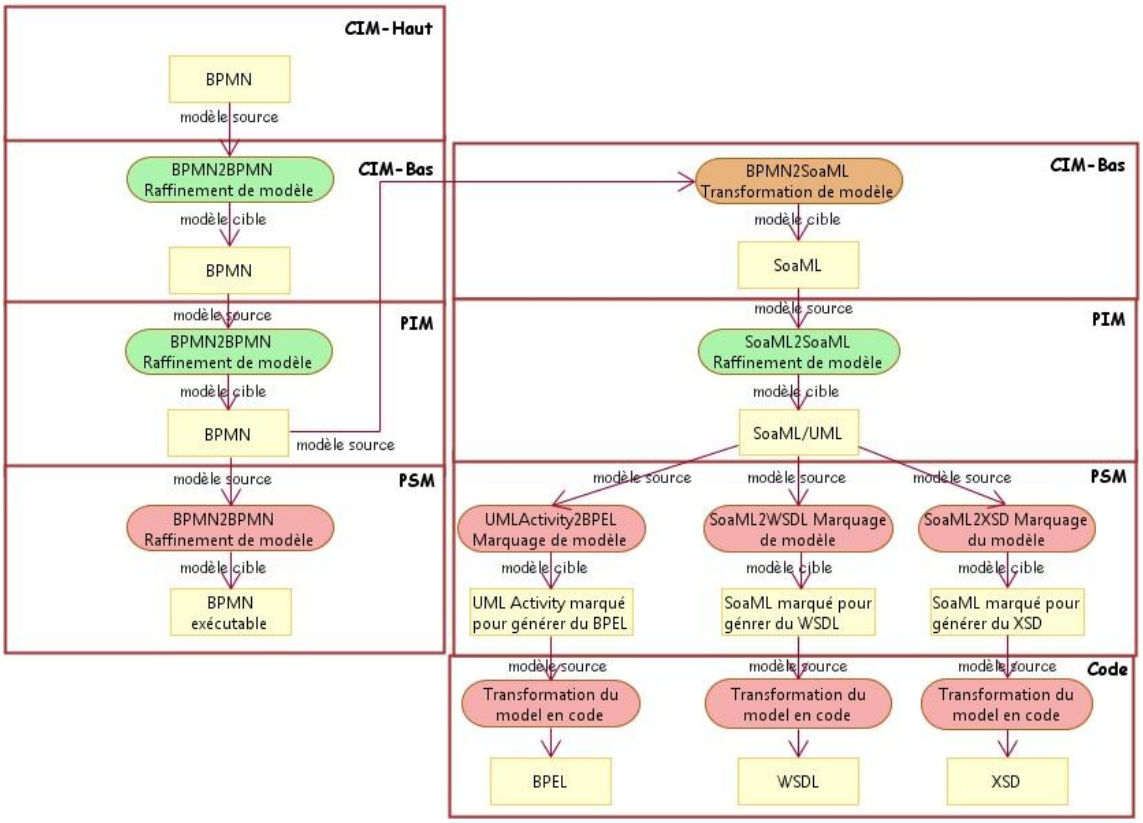


Figure 1. Vue d'ensemble des transformations de la méthode proposée

La Figure 1 représente un récapitulatif des onze transformations nécessaires pour générer le squelette de la solution SOA dans la plateforme des web-services. Certaines transformations sont

manuelles (couleur vert anis), une est semi-automatique (couleur ocre rouge) et d'autres encore sont automatiques (couleur rouge framboise). Les transformations 1, 2 et 5 sont manuelles parce qu'elles nécessitent l'intervention d'un expert pour raffiner les modèles sources en modèles cibles. Par exemple, les transformations 1 et 2 nécessitent l'intervention d'un analyste métier, alors que la transformation 5 nécessite l'intervention d'un architecte SOA. Quant-à la transformation 5 entre les formalismes BPMN et SoaML, elle est semi-automatique. Elle nécessite de préciser des informations nécessaires à la transformation des modèles de processus en modèles de services comme la vue d'architecture (globale ou détaillée) et le périmètre architectural (une architecture de collaboration ou d'un seul participant). Comme on l'a proposé dans III.2. cette transformation nécessite de choisir un pattern de transformation pour générer le modèle de services cible. Enfin, les transformations 3, 6, 7, 8, 9, 10, 11 sont automatiques. La transformation 3 consiste à enrichir le modèle des processus indépendant de la plateforme par le détail de l'exécution. Ce raffinement de modèle est le résultat de l'utilisation d'un environnement de développement du BPMS qui se charge d'enrichir le modèle BPMN 2.0. Quant-au reste des transformations (i.e., 6, 7, 8, 9, 10, 11) elles peuvent être aussi automatisées. En effet, des outils comme RSA permettent de générer le squelette de la solution SOA directement à partir des modèles du niveau PIM. Cependant il est tout à fait possible que les transformations 6, 7, 8 nécessite l'intervention humaine pour marquer les modèles.

On constate que la démarche MDI utilisé dans le cadre de la méthode proposée permet de garantir l'alignement entre les exigences métiers et l'application SOA qui les implémentent. Elle assure le respect des principes de l'architecture orientée service dans la solution SOA. Cependant, elle ne permet pas une automatisation complète du cycle de vie de développement des architectures orientées services.

2. CONCLUSION DU CAS D'ÉTUDE

On a identifié plusieurs points de vigilance qu'on estime nécessaires à prendre en considération dans un projet de collaboration :

1. Dans le but d'identifier le glissement du périmètre du projet qui peut être dû au changement des environnements des participants. Il faut dès que possible transcrire le périmètre du projet de la collaboration, ses objectifs globaux et les objectifs de chacun des participants
2. Les participants de la collaboration doivent choisir un type de collaboration et une approche d'interopérabilité (i.e., déterminer qui exécute quelle activité ou quel processus).

3. Il faut porter l'effort de modélisation sur les processus collaboratifs et les services publics de la collaboration. En effet, il est primordial de distinguer les processus collaboratifs des processus propres à chaque participant. Les premiers doivent faire l'objet d'une concertation entre les participants, alors que les seconds dépendent des considérations de chaque participant. De même, il faut distinguer les services publics qui forment l'architecture de services publics des services privés de chaque participant.
4. Les processus du niveau PIM doivent contenir le détail du flux de processus, c'est-à-dire les activités métiers et techniques indépendantes de toute plateforme ou technologie. Ces processus doivent décrire en détail la logique d'exécution du processus. Ensuite, c'est au développeur de les enrichir afin de les rendre exécutables dans une plateforme donnée. Le risque est de spécifier des processus PIM de haut niveau qui ne sont pas suffisamment détaillés.

3. LIMITES ET PERSPECTIVES

3.1. Les limites

La nature de notre recherche qui est basée sur une étude de cas porte des lacunes importantes quant à la généralisation des résultats (Gagnon, 2012). La spécificité de la description des processus et des services dans la méthode proposée ne fait pas bon ménage avec l'universalité. De plus, dans ce travail on prend en compte seulement deux des six éléments qu'on propose dans le cadre de référence pour assurer l'alignement métier-IT avec BPM et SOA: *Processus métiers* (cf. Chapitres II et IV) et *SOA et technologies IT* (cf. Chapitres I et V).

La mise en place de la méthode qu'on propose doit être précédée par le déploiement des règles de gouvernances. La gouvernance SOA est primordiale pour réussir la mise en place d'une architecture orientée services à tel point que Danny Sabbah (2007), directeur général de *Tivoli Software*[47] affirme que *« SOA is 1% services and 99% governance »*. Il est donc nécessaire d'appliquer la méthode qu'on propose dans le cadre d'une politique de gouvernance. D'où la nécessite d'aligner la méthode que l'on propose avec des règles de gouvernances SOA. Enfin, il est à noter que de nos jours aucun BPMS n'est capable d'exécuter toutes les notations BPMN 2.0.

[47] http://www-01.ibm.com/software/fr/tivoli/

3.2. Les perspectives

Le travail présenté dans cette thèse peut être poursuivi dans plusieurs directions : on distingue deux types de perspectives :

Les perspectives de la méthode de travail proposée

- La méthode se limite uniquement aux transformations verticales de l'approche MDI. Il faudrait prendre en compte les transformations horizontales de l'approche MDI.
- Proposer plus de patterns de transformation entre les formalismes BPMN et SoaML.
- Intégrer la modélisation des objectifs pour l'identification des processus et des services de la collaboration en utilisant un formalisme tel que BMM (OMG05, 2008).
- Étendre la méthode proposée pour étudier comment spécifier les processus et les services d'une entreprise qui souhaite migrer ses systèmes patrimoniaux (*legacy systems*) vers le cloud computing.

D'autres perspectives liées aux domaines étudiés dans notre travail

- Explorer l'architecture et les propriétés d'un Moteur d'Exécutions de Processus (MEP) capable d'exécuter une coopération avec une vue centralisée dans le cadre d'une approche d'interopérabilité fédérée.

Glossaire

AE	Architecture d'Entreprise
AIF	ATHENA Interoperability Framework
API	Application Programming Interface
ASICOM	Architecture de Système d'information Interopérable pour les industries du COMmerce
ATL	Atlas Transformation Language
BAE	Bon A. ENLEVÉ
BMM	Business Motivation Model
BOS	Bonita Open Solution
BPA	Business Process Analysis
BPM	Business Process Management
BPMS	Business Process Management Suite
BPMN	Business Process Modeling and Notation
CBS	Component-Based Systems
CIM	Computation Independent Model
CMOF	Complete Meta Object Facility
CSV	Comma-Separated Values
DSI	Directeurs des Systèmes d'Information
DTI	Direct Trader Interface
EAI	Enterprise Application Integration
EDI	Electronic Data Interchange
EIF	Enterprise Interoperability Framework
EMT	Economie, Méthodologie et Technologie
ERP	Enterprise Resource Planning
ESB	Enterprise Service Bus
ESD	Entrepôt Sous Douane
ETL	Extract Transform Load

HTTP	Hyper Text Transfer Protocol
JEE	Java Enterprise Edition
JMS	Java Message Service
IDM	Ingénierie Dirigée par les Modèles
IT	Information Technology
KPI	Key Performance Indicators
MBDS	Model-Based Development with SoaML
MDA	Model Driven Architecture
MDD4SOA	Model-Driven Development for Service-Oriented Architecture
MDI	Model Driven Interoperability
MEP	Moteur d'Exécution du Processus
MT4MDE	Mapping Tool for Model Driven Engineering
OASIS	Organization for the Advancement of Structured Information Standards
OCL	Object Constraint Language
OMG	Object Management Group
OSOA	Open Service Oriented Architecture
PIM	Platform Independent Model
PME	Petites et Moyennes Entreprises
PSM	Platform Specific Model
RPC	Remote Procedure Call
RSA	Rational Software Architect
RSI	Retour Sur Investissement
SAM	Strategic Alignment Model
SAWSDL	Semantic Annotations for WSDL and XML Schema
SCA	Service Component Architecture
SI	Système d'Information

SLA	Service Level Agreement
SOA	Service Oriented Architecture
SOAP	Simple Object Access Protocol
SoaML	Service Oriented Architecture Modeling Language
SOBA	Service Oriented Business Application
SOC	Service-Oriented Computing
SOE	Service Oriented Enterprise
SOMA	Service-Oriented Modeling and Architecture
TOGAF	The Open Group Architecture Framework
UDDI	Universal Description Discovery and Integration
UML	Unified Modeling Language
VAD	Vente À Distance
W3C	World Wide Web Consortium
WMS	Workflow Management System
WS-BPEL	Web Services Business Process Execution Language
WSDL	Web Services Description Language
WS-I	Web services Interoperability
XMI	XML Metadata Interchange
XML	eXtensible Markup Language
XSD	XML Schema Definition

Références Bibliographiques

Références externes

Allweye T., *Bpmn 2.0* Books on Demand GmbH, 2010, p. 156, ISBN: 3839149851, disponible sur: http://www.bpmn-introduction.com/.

Alonso G., Casati F., Kuno H., Machiraju V., *Web Services* (Data-Centric Systems and Applications), Springer, 2004, ISBN: 978-3-540-44008-6, disponible sur: http://www.springer.com/computer/database+management+%26+information+retrieval/book/978-3-540-44008-6.

Amazon, Amazon Fulfillment Web Service (Amazon FWS), 2011, disponible sur: http://aws.amazon.com/fr/fws//187-3384702-0912700#.

Amsden J., Modeling with SoaML, the Service-Oriented Architecture Modeling Language, 2010, disponible sur: http://www.ibm.com/developerworks/rational/library/09/modelingwithsoaml-1/index.html.

Arsanjani A., *Service-oriented modeling and architecture How to identify, specify, and realize services for your SOA*, IBM developerWorks, 2004, disponible sur: http://www.ibm.com/developerworks/library/ws-soa-design1/.

ASICOM, *Réponse au 5ème appel a projets R&D du fonds unique interministeriel*, 2007.

ATHENA01, *Deliverable Number : D.A4.2 : Specification of Interoperability Framework and Profiles, Guidelines and Best Practices* ATHENA European Integrated Project, 2007, p. 215.

ATHENA02, *Deliverable Number: D.B3.5 / D.B3.6 : ATHENA Contribution to Interoperability Policy Action Plan Version 3 and Long-Term Research Recommendations version 2*, ATHENA European Integrated Project, 2007.

ATHENA03, *D.A1.3.1: Report on Methodology description and guidelines definition, Version 1.0*, 2005.

atilf, CNRS, Lorraine u. d., *Le Trésor de la Langue Française informatisé*, 2012, disponible sur: http://atilf.atilf.fr/.

Avison D., Jones J., Powell P., Wilson D., *Using and validating the strategic alignment model*, *The Journal of Strategic Information Systems*, vol. 13, n° 3, 2004, p. 223-246, disponible sur: http://www.sciencedirect.com/science/article/B6VG3-4DFT6WR-1/2/d09134bf5b060dbfe8f591a47af93b4f.

Axelrod R., *Donnant Donnant - Théorie du comportement coopératif* (Collection Sciences Humaines), Editions Odile Jacob, 1992.

Bajwa I. S., Samad A., Mumtaz S., Kazmi R., Choudhary A., *BPM Meeting with SOA: A Customized Solution for Small Business Enterprises Information Management and Engineering ICIME '09*, 2009.

Ballinge K., Bisset B., Box D., Curbera F., Ferguson D., Graham S., Liu C. K., Leymann F., Lovering B., McCollum R., Nadalin A., Orchard D., Parastatidis S., Riegen C. v., Schlimmer J., Shewchuk J., Bill S., Truty G., Vedamuthu A., Weerawarana S., Wilson K., Yendlur P., *Web Services Metadata Exchange (WSMetadataExchange) Version 1.1*, 2006, p. 22, disponible sur: http://public.dhe.ibm.com/software/dw/specs/ws-mex/metadataexchange.pdf.

Banerjee U., SOA TRENDS – MINUS THE HYPE, 2010, disponible sur: http://setandbma.wordpress.com/2010/12/30/soa-trends-minus-the-hype/.

Bashein B. J., Markus M. L., Riley P., *Preconditions for BPR success and how to prevent failures*, *Information Systems Management*, vol. 11, n° 2, 1994, p. 14-23.

Berre A.-J., Hahn A., Akehurst D., Bezivin J., Tsalgatidou A., Vermaut F., Kutvonen L., F. Linington P., *State of the art for interoperability architecture approaches*, INTEROP, 2004, p. 362, disponible sur: http://interop-vlab.eu/ei_public_deliverables/interop-noe-deliverables/dap-domain-architecture-and-platforms/D91/?searchterm=d9.1.

Bessire D., Baker C. R., *The French Tableau de bord and the American Balanced Scorecard: a critical analysis*, *Critical Perspectives on Accounting*, vol. 16, n° 6, 2005, p. 645-664.

Bézivin J., Dupé G., Jouault F., Pitette G., Rougui J. E., *First experiments with the ATL model transformation language: Transforming XSLT into XQuery*, *2nd OOPSLA Workhop on Generative Techniques in the context of Model Driven Architecture*, Anaheim, USA, 2003.

Bigand M., Korbaa O., Bourey J.-P., *Integration of FMS performance evaluation models using patterns for an information system design*, *Computers & Industrial Engineering*, vol. 46, n° 4, 2004, p. 625-637, disponible sur: http://www.sciencedirect.com/science/article/B6V27-4CRY4H7-1/2/4395f2562efba1f95a6b4f5ce0c9c1ce.

Bloomberg J., *SOA strategy comparison: IBM & MICROSOFT*, zapthink, 2009, disponible sur: http://www.zapthink.com/.

Bonnet P., *Cadre de référence Architecture SOA, Meilleurs pratiques*, Orchestra Netwsorks, 2005, disponible sur: www.orchestranetworks.com.

Bonnet S., Marvie R., Geib J.-M., *Putting Concern-Oriented Modeling into Practice*, *2nd Nordic Workshop on UML, Modeling, Methods and Tools - MWUML 2004*, Turku, Finland, 2004.

Bourey J.-P., Grangel R., Doumeingts G., Berre A. J., *Deliverable DTG2.3 : Report on Model Driven Interoperability*, I. Society, InterOP, 2007, p. 91, disponible sur: www.interop-noe.org

Bourey J.-P., Grangel Seguer M. d. l. R., *Mappings and Model Transformations*, Ecole Centrale de Lille, Lille 2011.

Buelow H., Manoj D., Manas D., Palvankar P., Srinivasan M., *Getting Started with Oracle BPM Suite 11gR1 A Hands-On Tutorial*, [PACKT] PUBLISHING, 2010, p. 536 ISBN: 1849681686, disponible sur: http://www.packtpub.com/getting-started-with-oracle-bpm-suite-11gr1-hands-on-tutorial/book.

Burns T., Stalker G. M., *The Management of innovation* (Social Science Paperback), London, Tavistock Publications, 1971.

Carr N. G., *IT does not matter*, *Harvard Business Review*, 2003, p. 32.

Casanave C., *Enterprise Service Oriented Architecture Using the OMG SoaML Standard, a Model Driven Solutions, Inc. White Paper* ModelDriven.org, Model Driven Solutions, Inc for ModelDriven.org,

2009, p. 21, disponible sur: http://portal.modeldriven.org/content/soaml-presentations-white-papers

Catts A., St. Clair J., *Business Process Management Enabled by SOA Anthony*, Redbooks, IBM International Technical Support Organization, 2009, disponible sur: http://www.redbooks.ibm.com.

Chappell D., *Enterprise Service Bus*, O'Reilly, 2004, p. 274, ISBN: 0-596-00675-6.

Chebbi I., Dustdar S., *The view-based approach to dynamic inter-organizational workflow cooperation*, *Data Knowl. Eng*, vol. 56, n° 2, 2006, p. 139-173.

Cheliah P., Erl T., Gee C., Laird R., Maier B., Normann H., Shuster L., Trops B., Utschig C., Winterberg T., *Next Generation SOA: A Real-World Guide to Modern Service-Oriented Computing* (The Prentice Hall Service-Oriented Computing Series from Thomas Erl), non publié.

Chen D., Daclin N., *Framework for enterprise interoperability*, *IFAC TC5.3 workshop EI2N06*, Bordeaux, France, 2006.

Chen D., Daclin N., *Framework for Enterprise Interoperability,* (Interoperability for Enterprise Software and Applications), ISTE, 2010, p. 77-88, ISBN. 9780470612200, disponible sur: http://dx.doi.org/10.1002/9780470612200.ch6.

Chen D., Dassisti M., Elvesæter B., *Interoperability Knowledge Corpus, Intermediate Report. Deliverable DI.1b*, Network of Excellence InterOp, Contract No.IST-508011, 2006.

Chen D., Dassisti M., Elvesæter B., *Deliverable DI.3 : Enterprise Interoperability Framework and knowledge corpus Final report* INTEROP, 2007, p. 44, disponible sur: http://interop-vlab.eu/ei_public_deliverables/interop-noe-deliverables/delivlist.

Chesbrough H., *Open Innovation: The New Imperative for Creating and Profiting from Technology*, Harvard Business School Press, 2003.

Chinosi M., *Representing Business Processes: Conceptual Model and Design Methodology*, Dipartimento di Informatica e Comunicazione, Universit`a degli Studi dell'Insubria 2008.

Chinosi M., Trombetta A., *BPMN: An introduction to the standard*, *Computer Standards & Interfaces*, vol. 34, n° 1, 2012, p. 124-134, disponible sur: http://www.sciencedirect.com/science/article/pii/S0920548911000766.

Clark T., Sammut P., Willans J., *Applied Metamodelling a Foundation for Language Driven Development*, CETEVA, 2008.

Commission-européenne, *Manuel de Transit - Version consolidée*, 2010, p. 810, disponible sur: http://ec.europa.eu/taxation_customs/resources/documents/customs/procedural_aspects/transit/common_community/transit_manual_consolidation_fr.pdf.

Conner K. R., Prahalad C. K., *A Resource-Based Theory of the Firm: Knowledge Versus Opportunism*, *Organization science*, vol. 7, n° 5, 1996, p. 477-501, disponible sur: http://orgsci.journal.informs.org/cgi/content/abstract/7/5/477.

Crnkovic I., Stafford J., Szyperski C., *Software Components beyond Programming: From Routines to Services*, *Software, IEEE* vol. 28, n° 3, 2011, p. 22 - 26 disponible sur: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=5756294&tag=1.

Cummins F. A., *Building the agile enterprise with SOA, BPM and MBM.,* (Enabling the Agile Enterprise), Burlington, Morgan Kaufmann, 2009.

Cummins F. A., *BPM Meets SOA,* (Handbook on Business Process Management 1 Introduction, Methods, and Information Systems), Springer, 2010, p. 461-479, ISBN. 978-3-642-00415-5, disponible sur: http://www.springer.com/business+%26+management/business+information+systems/book/978-3-642-00415-5.

Decker G., Weske M., *Instance Isolation Analysis for Service-Oriented Architectures*, *2008 IEEE International Conference on Services Computing, SCC 2008*, Honolulu, 2008, vol. 1, p. 249-256.

Doumeingts G., *How to decentralize decisions through GRAI model in production management*, Kidlington, ROYAUME-UNI, Elsevier, 1985.

Drucker P. F., *The Practice of Management*, New York, Harper & Row, 1954.

Ducq Y., *Habilitation à diriger des recherches - Évaluation de la performance d'entreprise par les modèles*, Université Bordeaux 1, 2007, p. 124, disponible sur: http://www.univ-valenciennes.fr/GDR-MACS/hdr/HDR_Yves_Ducq.pdf.

Dunnavant T., Johnston G., Design and develop a more effective SOA, Part 2: Confidently define and design your service-oriented solutions using Rational SOMA 2.9, 2011 disponible sur: http://www.ibm.com/developerworks/webservices/library/ws-designsoapart2/index.html.

EIF, *European Interoperability Framework for pan-European eGovernment services V1.0*, La commission Européenne, Bruxelles, 2004, p. 79, disponible sur: http://ec.europa.eu/idabc/en/document/3473/5585.html.

Elfatatry A., *Dealing with change: components versus services*, *Commun. ACM*, vol. 50, n° 8, 2007, p. 35-39.

Elvesæter B., Berre A. J., Sadovykh A., *Specifying services using the service oriented architecture modeling language (SoaML)*, *The 1st International Conference on Cloud Computing and Services Science, CLOSER*, The Netherlands, 2011.

Elvesæter B., Carrez C., Mohagheghi P., Berre A.-J., G.Johnsen S., Solberg A., *Model-Based Development with SoaML*, 2010, p. 23, disponible sur: http://www.uio.no/studier/emner/matnat/ifi/INF5120/v10/undervisningsmateriale/MDSE-SoaML-INF5120.pdf.

Elvesæter B., Panfilenko D., Jacobi S., Hahn C., *Aligning Business and IT Models in Service-Oriented Architectures using BPMN and SoaML*, *First Workshop on Model-Driven Interoperabilit*, Oslo, Norway, 2010, p. 61-68.

Endrei M., Ang J., Arsanjani A., Chua S., Comte P., Krogdahl P., Luo M., Newling T., *Patterns: Service-Oriented Architecture and Web Services*, 2004, disponible sur: ibm.com/redbooks.

Erl01 T., *Service-Oriented Architecture : Concepts, Technology, and Design : 10ème édition* (The Prentice Hall Service-Oriented Computing Series from Thomas Erl), Prentice Hall, 2010, p. 760, ISBN: 0-13-185858-0, disponible sur: http://www.soabooks.com/.

Erl02 T., *SOA principles of Service Design*, Indiana, 2010, disponible sur: http://www.soabooks.com/.

Erl T., *SOA Design Patterns* (The Prentice Hall Service-Oriented Computing), The Prentice Hall, 2009, p. 814, ISBN: 978-0136135166, disponible sur: http://www.soabooks.com/.

Erl T., G.Bennette S. à. v., Gee C., Laird R., Manes T. A., Schneider R., Shuster L., Tost A., Venable C., *SOA Governance: Governing Shared Services On-Premise and in the Cloud* (The Prentice Hall Service-Oriented Computing Series from Thomas E), Prentice Hall, 2011, ISBN: 978-0138156756, disponible sur: http://www.soabooks.com/.

Etien A., *La méthode ACEM pour l'alignement d'un système d'information aux processus d'entreprise*, L'université Paris I - Pantheon - Sorbonne Paris, 2006.

Frye C., *Special Report: BPM inside the Belly of the SOA Whale* SearchSOA, 2006, Date du dernier accès: 07/02/2011, disponible sur: http://searchsoa.techtarget.com/news/1193780/Special-Report-BPM-inside-the-belly-of-the-SOA-whale.

Gagnon Y.-C., *L'étude de cas comme méthode de recherche, 2ème édition*, Presses de l'Université du Québec, 2012, p. 117.

Gamma E., Helm R., Johnson R., Vlissides J., *Design Patterns: Elements of Reusable Object-Oriented Software*, Addison-Wesley, 1995.

GAO00, *Information Technology: FBI Is Taking Steps to Develop an Enterprise Architecture, but Much Remains to Be Accomplished*, GAO-05-363, 2005, disponible sur: http://www.gao.gov/products/GAO-05-363.

GAO01, *DOD Business Systems Modernization: Long-Standing Weaknesses in Enterprise Architecture Development Need to Be Addressed*, 2005, p. 69, disponible sur: http://www.gao.gov/products/GAO-05-702.

Gartner, *Leading in times of transition. The 2010 CIO Agenda*, Stamford, 2010, disponible sur: http://www.gartner.com/DisplayDocument?resId=1280013.

Gilbert P., *A Business Oriented Architecture Combining BPM and SOA for Competitive Advantage*, Executive Vice President at Lombardi Software, 2007, disponible sur: http://www.bptrends.com/deliver_file.cfm?fileType=publication&fileName=TWO%2003-07-ART-ABusinessOrientedArchitecture-Gilbert-Final.pdf.

Grangel R., Bigand M., Bourey J.-p., *Profils UML pour la transformation de modèles décisionnels GRAI en UML*, *8ème Congrès International de Génie Industriel CIGI'2009*, Bagnères de Bigorre, France, 2009.

Gulledge T., *Architecture-driven Enterprise Integration*, *International Journal of Management and Enterprise Development*, vol. 5, n° 3, 2008, p. 265-309.

Gulledge T., *Integrated Business Process and Service Management* (Handbook on Business Process Management 1 Introduction, Methods, and Information Systems), Springer, 2010, p. 481-497, ISBN. 978-3-642-00415-5.

Gulledge T., Deller G., *Service-oriented concepts: bridging between managers and technologists*, *Industrial Management & Data Systems*, vol. 109, n° 1, 2009, p. 5-15.

Haan J. d., Architecture requirements for Service-Oriented Business Applications, 2008, disponible sur: http://www.theenterprisearchitect.eu/archive/2008/05/19/architecture-requirements-for-service-oriented-business-applications.

Han F., Moller E., Berre A. J., *Organizational interoperability supported through goal alignment with BMM and service collaboration with SoaML*, *The 2009 International Conference on Interoperability for Enterprise Software and Applications - IESA '09*, Chine, 2009, p. 268 – 274.

Harmon P., *The Scope and Evolution of Business Process Management,* (Handbook on Business Process Management 1 Introduction, Methods, and Information Systems), Springer, 2010, p. 37-81, ISBN. 978-3-642-00415-5.

Hiemstra A., Ravesteyn P., Versendaal J., *An Alignment Model For Business Process Management And Service Oriented Architecture*, *La 6ème International Conference on Enterprise Systems, Accounting ICESAL 09* Thessaloniki, Greece, 2009.

Hille-Doering R., Making of the BPMN 2.0 metamodel for Eclipse merge and conquer, 2010, p. 10, disponible sur: http://www.sdn.sap.com/irj/scn/go/portal/prtroot/docs/library/uuid/a0ec6d13-1ab4-2d10-dc87-c0a18d7f23c8?QuickLink=index&overridelayout=true.

Hohpe G., Woolf B., *Enterprise Integration Patterns, Designing, Building, and Deploying Messaging Solutions*, Addison Wesley, 2003, p. 682.

Humble J., *La direction par objectifs et ses applications*, Paris, Publi-Union, 1971.

IDEAS, *A Gap Analysis - Required Activities in Research, Technology and Standardisation to close the RTS Gap - Roadmaps and Recommendations on RTS activities*, 2003.

IEEE01, *IEEE Standard Glossary of Software Engineering Terminology*, 1990, ISBN: 0-7381-0391-8

IEEE02, *Standard Computer Dictionary- A Compilation of IEEE Standard Computer Glossaries*, New York, 1990, p. 215, ISBN: ISBN 1-55937-079-3

Infosys, *Enterprise Architecture Expands its Role in Strategic Business Transformation. Infosys Enterprise Architecture Survey 2008/2009* 2009, disponible sur: https://pub.infosys.com/.

ISO-14258, *ISO-14258. Industrial automation systems and integration – Concepts and rules for enterprise models,*, 1998.

ISO/IEC/IEEE-42010, *Systems and software engineering — Architecture description (DRAFT)*, 2011, disponible sur: http://www.iso-architecture.org/ieee-1471/.

Jackson M., *Software Requirements and Specifications*, Addison-Wesley, 1995, ISBN: 0201877120.

Jochem R., *Enterprise Interoperability assessment*, *8th International Conference of Modeling and Simulation, MOSIM'10* Hammamet-Tunisia, 2010.

Kamoun F., *A roadmap towards the convergence of business process management and service oriented architecture*, *Ubiquity*, vol. 2007, n° Avril, 2007.

Kaplan R. S., Norton D. P., *Le tableau de bord prospectif*, Paris, Editions d'Organisation, 1998.

Kaplan R. S., Norton D. P., *The strategy-focused organization*, Harvard Business School Press, 2001.

Keen M., Acharya A., Bishop S., Hopkins A., Milinski S., Nott C., Robinson R., *Patterns: implementing an SOA Using an Enterprise Service Bus*, IBM, 2004, disponible sur: ibm.com/redbooks.

Kellogg K. C., Orlikowski W. J., Yates J., *Life in the Trading Zone: Structuring Coordination Across Boundaries in Postbureaucratic Organizations*, *Organization science*, vol. 17, n° 1, 2006, p. 22-44.

Korhonen J. J., Hiekkanen K., Heiskala M., *Map to Service-Oriented Business and IT: A Stratified Approach*, *AMCIS 2010 PROCEEDINGS*, 2010, disponible sur: http://aisel.aisnet.org/amcis2010/157/.

Lawrence P., Lorsch J., *Adapter les structures de l'entreprise* (Collection Les Classiques), Paris, Editions d'Organisations, 1989.

Lemrabet Y., Clin D., Bigand M., Bourey J.-P., *From BPMN 2.0 to the Setting-Up on an ESB – Application to an Interoperability Problem,* (Collaborative Networks for a Sustainable World), Springer Boston, vol. 336, 2010, p. 722-729, disponible sur: http://dx.doi.org/10.1007/978-3-642-15961-9_85.

Liu H., *Integration of model driven engineering and ontology approaches for solving interoperability issues*, Laboratoire de Modélisation et de Management des Organisations EA 4344 Ecole Centrale de Lille, Lille, 2011.

Liu H., Jean-Pierre B., *Transformation from a Collaborative Process to Multiple Interoperability Processes*, *Enterprise Interoperability IV (Conference I-ESA'10)*, Springer, London, 2010, p. 135-144.

Liu H., Lemrabet Y., Clin D., Bourey J.-P., *Comparison between Collaborative Business Process Tools*, *Le 5ème IEEE International Conference on Research Challenges in Information Science*, Guadeloupe - French West Indies, 2011.

Lopes D., Hammoudi S., De.souza J., Bontempo A., *Metamodel Matching: Experiments and Comparison*, *Conference on Software Engineering Advances (ICSEA'06)*, Papeete, Tahiti, French Polynesia, 2006, p. du 28 octobre au 22 novembre.

Luftman J., Kempaiah R., Henrique R. E., *Key Issues for IT Executives 2008*, *MIS Quarterly Executive*, vol. 8, n° 3, 2009, p. 151-159, disponible sur: http://misqe.org/ojs2/index.php/misqe/article/view/265.

Luftman J. N., Lewis P. R., Oldach S. H., *Transforming the enterprise: the alignment of business and information technology strategies*, *IBM Syst. J.*, vol. 32, n° 1, 1993, p. 198-221.

Malinverno P., *Service-Oriented Architecture Craves Governance*, Gartner, 2006, p. 5, disponible sur: http://www.gartner.com/id=488180.

Marks E. A., Bell M., *Service Oriented Architecture (SOA): A Planning and Implementation Guide for Business and Technology*, Wiley Editions, 2006, p. 384 ISBN: 978-0-471-76894-4.

Markus M. L., Jacobson D. D., *Business Process Governance,* (Handbook on Business Process Management 2 Strategic Alignment, Governance, People and Culture), 2010, p. 201-222, disponible sur: http://www.springer.com/series/3795.

Marlon D., Thomas K., *Service-Enabled Process Management,* (Handbook on Business Process Management 1 Introduction, Methods, and Information Systems), Springer, 2010, p. 441-460, ISBN. 978-3-642-00415-5.

Matjaz B. J., Benny M., Poornachandra S., *Business Process Execution Language for Web Services, An architect and developer's guide to orchestrating web services using BPELAWS*, Packt Publishing Ltd, 2006.

Mayer P., *MDD4SOA - Model-Driven Development for Service-Oriented Architectures*, Doctoral thesis, LMU Munich, Munich, 2010.

McCoy D. W., A Personal History of "BPM, the Term", 2009.

McIlroy M. D., *Mass produced software components,* , *Software Engineering, Report on a conference sponsored by the NATO Science Committee*, Garmisch, Germany, 1968, p. 138-155, disponible sur: http://www.cs.dartmouth.edu/~doug/components.txt.

Melenovsky M. J., Sinur J., Hill B. J., McCoy W. D., *Business Process Management: Preparing for the Process-Managed Organization*, Gartner, 2005, p. 8.

Montagne D., *Delta-D, documentation technique gestion des échanges XML, version 2.1.5.1*, Direction générale des douanes et droits indirects, 2007, p. 28.

Natis Y. V., Service-Oriented Architecture Scenario, 2003, disponible sur: http://www.gartner.com/DisplayDocument?doc_cd=114358.

OASIS01, *UDDI Version 2.04 API Specification*, 2002, disponible sur: http://uddi.org/pubs/ProgrammersAPI-V2.04-Published-20020719.htm.

OASIS02, *WS-BPEL Extension for People (BPEL4People) Specification Version 1.1*, 2010, p. 57, disponible sur: http://docs.oasis-open.org/bpel4people/bpel4people-1.1.html.

OASIS03, *Web Services Business Process Execution Language Version 2.0*, 2007, disponible sur: http://docs.oasis-open.org/wsbpel/2.0/OS/wsbpel-v2.0-OS.html.

OASIS04, *Web Services – Human Task (WS-HumanTask) Specification Version 1.1*, 2010, p. 215, disponible sur: http://docs.oasis-open.org/bpel4people/ws-humantask-1.1.html.

OMG01, *Business Process Model and Notation (BPMN) Version 2.0*, 2011, p. 508, disponible sur: http://www.omg.org/spec/BPMN/2.0.

OMG02, *Service Oriented Architecture Modeling Language (SoaML) 1.0 - Beta 2*, 2009, p. 167, disponible sur: http://www.omg.org/SoaML/1.0/Beta2/PDF.

OMG03, *Unified Modeling Language TM (OMG UML), Superstructure version 2.4*, 2011, p. 742.

OMG04, *MDA Guide Version 1.0.1*, 2003, p. 62, disponible sur: http://www.omg.org/cgi-bin/doc?omg/03-06-01.

OMG05, *Business Motivation Model Version 1.0*, 2008, disponible sur: http://www.omg.org/spec/BMM/1.0/PDF.

OMG06, *OMG MOF 2 XMI Mapping Specification Version 2.4.1*, 2011, disponible sur: http://www.omg.org/spec/XMI/2.4.1/.

OMG07, *OMG Meta Object Facility (MOF) Core Specification Version 2.4.1*, 2011, disponible sur: http://www.omg.org/spec/MOF/2.4.1/.

OMG08, *Object Constraint Language Version 2.3.1*, 2011, disponible sur: http://www.omg.org/spec/OCL/2.3.1/Beta2/.

OMG09, *Business Process Model and Notation (BPMN) Version 1.2*, 2009, p. 316, disponible sur: http://www.omg.org/spec/BPMN/1.2.

OMG10, *OMG Unified Modeling Language (OMG UML), Superstructure, V2.1.2*, 2007, p. 715.

Oracle, *Adopter une approche BPM et SOA pour une intégration métier tout en souplesse*, O. Corporation, 2009.

OSOA, Service Component Architecture Home, 2007, disponible sur: http://www.osoa.org/display/Main/Service+Component+Architecture+Home.

Papazoglou M. P., Andrikopoulos V., Benbernou S., *Managing Evolving Services*, *IEEE Software*, vol. 28, n° 3, 2011, p. 49-55, disponible sur: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=5719589.

Papazoglou M. P., Traverso P., Dustdar S., Leymann F., *Service-Oriented Computing: State of the Art and Research Challenges*, *IEEE Computer Society*, vol. 40, n° 11, 2007, p. 38 - 45

PraxemeInstitute01, *Composant PxM-02, Guide général, version 1.1* 2006, disponible sur: http://www.praxeme.org/DocumentsGuidesMethodologiques/PxM02-gGen.pdf.

PraxemeInstitute02, *PxM-40, le guide de l'aspect logique, version 1.0*, PraxemeInstitute, 2007, disponible sur: http://www.praxeme.org/DocumentsGuidesMethodologiques/PxM40-gLq.pdf.

Quelin B., *Cooperation Inter-Entreprises et Création de Ressources*, Les Cahiers de Recherche 522/1994, HEC, 1994, p. 1-35.

Recker J. C., *BPMN Modeling – Who, Where, How and Why*, vol. 5, n° 3, 2008, p. 1-8, disponible sur: http://eprints.qut.edu.au/12317/.

Regev G., Wegmann A., *Remaining Fit: On the Creation and Maintenance of Fit,* (CAiSE Workshops (2)), 2004, p. 131-137.

Reix R., *Systèmes d'information et management des organisations*, Vuibert, 2004, ISBN: 978-2-7117-7568-2, disponible sur: http://www.eyrolles.com/Informatique/Livre/systemes-d-information-et-management-des-organisations-9782711775682?PHPSESSID=.

Rey-Debove J., Rey A., *Le nouveau PETIT ROBERT dictionnaire alphabétique et analogique de la langue française* Paris, DICTIONNAIRES LE ROBERT, 1993.

Rickayzen A., BPEL4People White Paper Overview, 2007 disponible sur: http://www.sdn.sap.com/irj/scn/weblogs?blog=/pub/wlg/5859.

Roshen w., *SOA-Based Enterprise Integration: A Step-by-Step Guide to Services-Based Application Integration*, 2009, p. 384 ISBN: 0071605525.

Ross J. W., Weill P., Robertson D., *Enterprise architecture as strategy: creating a foundation for business execution*, 2006, p. 234.

Rummler G., Brache A. P., *How to improve the white space in the organization chart* San Francisco, Jossey-Bass, 1990.

Sabbah D., *The future of software delivery*, IBM, 2007, p. 28.

Schmelzer R., The Seven Levels of Loose Coupling, 2007.

Schulte W. R., Predicts 2003: Enterprise Service Buses Emerge, 2002.

Segrestin B., *Innovation et coopération interentreprises - Comment gérer les partenariats d'exploration ?* (CNRS Economie), CNRS Editions, 2006.

Silver B., *Uniting Process architecture and execution*, B. Watch, 2010, disponible sur: http://www.brsilver.com/.

Silver B., *BPMN Method & Style, second Edition with BPMN implementer's Guide*, Aptos, CODY-CASSIDY PRESS, 2011, ISBN: 978-0-9823681-1-4.

Simon H., Yefim N., Application Infrastructure Reflects New Dynamics in the Software Market, 2006, disponible sur: http://www.gartner.com/id=499711.

Smile, *Plateformes web Hautes Performances --o-- Principes d'architecture et outils open source*, Smile, Smile Open Source solution, 2009, p. 180.

Smith H., Fingar P., *Business Process Management : The Third Wave*, Meghan-Kiffer Press, 2003, p. 311 ISBN: 978-0929652337.

Soley R. M., *Model driven architecture (mda), draft 3.2. Rapport technique*, 2000, p. 12, disponible sur: http://www.omg.org/cgi-bin/doc?omg/00-11-05.

Spanyi A., *Business Process Management Governance,* (Handbook on Business Process Management 2 Strategic Alignment, Governance, People and Culture), 2010, p. 223-238, disponible sur: http://www.springer.com/series/3795.

Swithinbank P., Chessell M., Gardner T., Griffin C., Man J., Wylie H., Yusuf L., *Patterns: Model-Driven Development Using IBM Rational Software Architect*, IBM, 2005, disponible sur: ibm.com/redbooks.

Szyperski C., *Component Software - Beyond Object-Oriented Programming – Second Edition*, Addison-Wesley, 2002, ISBN: 0-201-74572-0, disponible sur: http://www.amazon.com/Component-Software-Beyond-Object-Oriented-Programming/dp/0201745720.

Thevenet L.-H., *Proposition d'une modélisation conceptuelle d'alignement stratégique : La méthode INSTAL*, CRI - Centre de Recherche en Informatique, Paris I – Panthéon-Sorbonne, Paris, 2009, disponible sur: http://tel.archives-ouvertes.fr/tel-00466827/fr/.

TOGAF, *The Open Group Architecture Framework (TOGAF) Version 9*, T. O. Group, The Open Group, 2009, disponible sur: http://www.opengroup.org.

Trowbridge D., Roxburgh U., Hohpe G., Manolescu D., Nadhan E. G., *Integration Patterns*, Microsoft Press, 2004, p. 432, ISBN: 978-0-7356-1850-3.

Truptil S., Bénaben F., Couget P., Lauras M., Chapurlat V., Pingaud H., *Mediation Information System Engineering for Interoperability Support in Crisis Management*, *I-ESA'10*, Coventry, 2010.

Venkatraman N., Henderson J. C., *Strategic alignment: Leveraging information technology for transforming organizations*, *IBM Systems Journal*, vol. 32, n° 1, 1993, p. 4-16.

W3C00, *SOAP Version 1.2*, 2007, disponible sur: http://www.w3.org/TR/soap/.

W3C04, *Web Services Policy 1.5 - Framework*, 2007, disponible sur: http://www.w3.org/TR/ws-policy/#tocRange.

W3C06, *XQuery 1.0: An XML Query Language*, 2007.

W3C07, *XML Path Language (XPath) Version 1.0*, 1999.

W3C08, *Web Services Description Language (WSDL) Version 2.0 Part 1: Core Language*, 2007, disponible sur: http://www.w3.org/TR/wsdl20/.

W3C09, *XML Schema Part 0: Primer Second Edition*, 2004.

W3C10, *Semantic Annotations for WSDL and XML Schema*, 2007, disponible sur: http://www.w3.org/TR/2007/REC-sawsdl-20070828/.

WS-I, *Deliverables from the Basic Profile Working Group*, WS-I Organization, 2010, Date du dernier accès: 18-02-2012, disponible sur: http://ws-i.org/deliverables/workinggroup.aspx?wg=basicprofile.

WSO2, *WSO2 Enterprise Service Bus*, 2012, Date du dernier accès: 07/04/2012, disponible sur: http://docs.wso2.org/display/ESB403/Enterprise+Service+Bus+Documentation.

Wylie H. M., Llambros P., Connectivity Solution Patterns, 2009, disponible sur: http://www.ibm.com/developerworks/library/ws-enterpriseconnectivitypatterns/index.html#iratings.

Yang J., *Design and implementation of mechanism for automatically generating transformation from BPMN to SoaML*, Master of Engineering, Software Engineering, Harbin Institute of Technology, 2011.

Zachman J. A., *Enterprise Architecture Artifacts versus Application Development Artifacts*, vol. 28, 2000, disponible sur: http://www.brcommunity.com.

Zairi M., *Business process management: a boundaryless approach to modern competitiveness*, *Business Process Management Journal*, 1997, p. 64-88.

Zhao X., *On Supporting Collaborative Business Processes - an Organisation-Oriented Perspective*, Faculty of Information and Communication Technologies Swinburne University of Technology, 2007.

Zhong Y., Yang J., *Contract-First Design Techniques for Building Enterprise Web Services*, *International Conference on Web Services (ICWS 2009)*, Los Angeles, CA, USA, 2009, p. 591-598

Zimmermann O., *An architectural decision modeling Framework for Service-Oriented Architecture Design*, Institut für Architektur von Anwendungssytemen der Universität Stuttgart, Universität Stuttgart, Zürich, 2009.

Zur. Mühlen M., Shapiro R., *Business Process Analytics,* (Handbook on Business Process Management 2 Strategic Alignment, Governance, People and Culture), 2010, p. 137-157, disponible sur: http://www.springer.com/series/3795.

Références de l'auteur

Lemrabet Y., *Méthodologie d'implémentation d'une architecture orientée service dirigée par le métier dans les petites et moyennes entreprises*, *INFORSID'11*, Lille, 2011.

Lemrabet Y., Benkeltoum N., Bigand M., Clin D., Bourey J.-P., *Entreprise agile et l'alignement métier-IT avec les approches BPM et SOA : retour d'expérience CIGI'11, 9e Congrès International de Génie Industriel*, Saint-Sauveur, Québec, Canada, 2011.

Lemrabet Y., Bigand M., David C., Benkeltoum N., Bourey J.-P., *Model Driven Interoperability in practice: preliminary evidences and issues from an industrial project*, *First Workshop on Model Driven Interoperability, MDI'10, In conjunction with CM/IEEE 13th International Conference, MoDELS* ACM, Oslo, Norway, 2010, p. 3-9.

Lemrabet Y., Clin D., Bigand M., Bourey J.-P., *From BPMN 2.0 to the Setting-Up on an ESB – Application to an Interoperability Problem,* (Collaborative Networks for a Sustainable World), Springer Boston, vol. 336, 2010, p. 722-729, disponible sur: http://dx.doi.org/10.1007/978-3-642-15961-9_85.

Lemrabet Y., Liu H., Benkeltoum N., Bigand M., Bourey J.-P., *Business-IT alignment and organisation agility enabled by BPM and SOA approaches interplay*, *13th International Conference on Enterprise Information Systems (ICEIS'2011)*, Beijing, China, 2011.

Lemrabet Y., Liu H., Bourey J.-P., Bigand M., *Proposition of Business process modelling in Model Driven Interoperability approach at CIM and PIM levels*, *IEASA-12 : International Conference on Interoperability for Enterprise system and Applications*, Valencia, Espagne, 2012.

Lemrabet Y., Touzi J., Clin D., Bigand M., Bourey J.-P., *Mapping of BPMN models into UML models using SoaML profile*, *8th International Conference of Modeling and Simulation (MOSIM'10)*, 2010.

Liu H., Lemrabet Y., Clin D., Bourey J.-P., *Comparison between Collaborative Business Process Tools*, *Le 5ème IEEE International Conference on Research Challenges in Information Science*, Guadeloupe - French West Indies, 2011.

Annexes

Annexe A : Le Code du connecteur développé dans BOS pour écrire un fichier dur le disque

```
package org.bonitasoft.connectors.lm2o;

import java.io.File;
import java.io.FileOutputStream;
import java.io.IOException;
import java.util.List;

import org.ow2.bonita.connector.core.ConnectorError;
import org.ow2.bonita.connector.core.ProcessConnector;
import org.ow2.bonita.util.AccessorUtil;

public class WriteFileToDisk extends ProcessConnector {

	// result message
	private String result;
	// DO NOT REMOVE NOR RENAME THIS FIELD
	private org.ow2.bonita.facade.runtime.AttachmentInstance
attachedFile;
	@Override
	protected void executeConnector() throws Exception {
		if(attachedFile != null){
			String pathName = "C:\\TEMP\\" +
attachedFile.getFileName();
			writefileIntoDisk(pathName);
		}else {
			// no thing to do
		}
	}

	// create file
	public void writefileIntoDisk(String pathName) throws Exception{
		File file = new File(pathName);
		if (!file.exists()){
			if (file.createNewFile()){
				// write data into file
				writeDataIntoFile();
				result = "file "+ pathName+" was created";

			}else {
				result = "impossible to create the file" +
pathName;
			}
		}else {
			result = "another file with the name"+ pathName +"exist";
		}
	}

	// write data into file
	public void writeDataIntoFile()throws IOException {
		FileOutputStream fileOutputStream = new
FileOutputStream(attachedFile.getFileName());
		byte [] fileContent =
AccessorUtil.getQueryRuntimeAPI().getAttachmentValue(attachedFile);
		fileOutputStream.write(fileContent);
	}

	@Override
```

```
	protected List<ConnectorError> validateValues() {
		// TODO Auto-generated method stub
		return null;
	}

	/**
	 * Getter for output argument 'result'
	 * DO NOT REMOVE NOR RENAME THIS GETTER, unless you also change the
related entry in the XML descriptor file
	 */
	public String getResult() {
		return result;
	}

	/**
	 * Setter for input argument 'attachedFile'
	 * DO NOT REMOVE NOR RENAME THIS SETTER, unless you also change the
related entry in the XML descriptor file
	 */
	public void
setAttachedFile(org.ow2.bonita.facade.runtime.AttachmentInstance
attachedFile) {
		this.attachedFile = attachedFile;
	}

}
```

Annexe B : Le code de la transformation de modèles entre le méta-modèle BPMN et le profil SoaML

```
-- @nsURI SoaMLProfile=http://www.eclipse.org/uml2/3.0.0/UML
-- @nsURI UML2=http://www.eclipse.org/uml2/3.0.0/UML
-- @path BPMNMM=/BPMN2SoaMLThese/InputBPMNMetaModels/BPMNMM_v004.ecore

-- Auteur : Youness LEMRABET
-- V1 : 16/01/2012 : ajout de la règle Collaboration2ServicesArchitecture pour la modélisation de community-level ServicesArchitecture
-- V2 : 18/01/2012 : correction de la règle Collaboration2ServicesArchitecture en ajoutant une transformation vers le Modèle
-- V3 : 19/01/2012 : ajout de la règle de transformation Participant2Property
-- V4 : 20/01/2012 : ajout de la transformation MessageFlow2CollaborationUse : CollaborationUse, ServiceContract, tDependency1, tDependency2
-- V5 : 20/01/2012 : ajout des commentaire et changement de la visibilité des Dependencies
-- V6 : 21/01/2012 : ajout des Ports <<Request>> et <<Service>> associés à une Property
-- V7 : 23/01/2012 : ajout d'une règle pour transformer un Process en ServicesArchitecture : Porcess2ServicesArchitecture
--                 : externaliser la création du Model et l'application du Profile dans une matched rule : CreateGlobalModelAndApplySoaMLProfile
--                 : ajout de la règle de transformation d'une Lane en Participant et Propery : Lane2ParticipantAndProperty
--                 : ajout d'une règle pour transformer un SubProcess en ServicesArchitecture : SubPorcess2ServicesArchitecture
--          24/01/2012 : correction de la génération des Ports <<Request>> et <<Service>> dans les Participants
--          26/01/2012 : correction de la règle de transformation d'un SubProcess en ServicesArchitecture : SubPorcess2ServicesArchitecture
--          27/01/2012 : correction de la règle : MessageFlow2CollaborationUse suppression de la transformation vers le Service et le Request afin de respecter la méthode que je propose
--      30/01/2012 : amélioration de la règle FlowNode2PropertyAndParticipant
-- V8 :30/01/2012 : correction de la règle FlowNode2PropertyAndParticipant en la transformant en Lane2PropertyAndParticipant
--     : 31/01/2012 : correction de la règle SequenceFlow2CollaborationUse en remplacant les FlowNodes par des Lanes
--                 : mettre la valeur de la propriété isConjugated du «Request» à true
--    : 13/02/2012 : correction du nom du ServiceContract
--    : 07/04/2012 : changer Le service Interface par une interface
```

```
module BPMN2SoaMLMapping7;
create OUT : UML2 from PRO : SoaMLProfile, BPMN : BPMNMM;

-- une variable gloable qui contient le Model UML
helper def: Model : UML2!Model = OclUndefined;

--###################  matched rules #################

-- helper pour obtenir le Profile à partir de son nom
-- appelé par : thisModule.getProfile(name)
helper def : getProfile(name : String) : SoaMLProfile!Profile =
 SoaMLProfile!Profile.allInstancesFrom('PRO')
 ->select(p | p.name = name).first();

-- helper pour obtenir le Stereotype à partir de son nom
-- appelé par : thisModule.getStereotype(nom)
helper def : getStereotype(name : String) : SoaMLProfile!Stereotype =
 SoaMLProfile!Stereotype.allInstances()
 ->select(s | s.name = name)->first();

-- Un seul élément Model d'UML est créé dans le modèle SoaML cible. Ce dernier
-- est généré à partir des éléments BPMN 2.0 Collaboration ou Proces
rule CreateGlobalModelAndApplySoaMLProfile(collaboration : UML2!Collaboration) {
 to
       tModel : UML2!Model (
             name <- 'tModel'.debug('tModel.name'),
             packagedElement <- Sequence{collaboration}
       )
 do {
       --******************BEGIN OF CreateModel*************
       let s : String = 'CreateGlobalModelAndApplySoaMLProfile => BEGIN'
       in s.println();

       if (thisModule.Model->oclIsUndefined() ) {
             --****************** BEGIN OF Profile Application ******************
             tModel.applyProfile(thisModule.getProfile('SoaML')).debug('applyProfile : SoaML');
             --****************** END OF Profile Application ********************
             --Store the Model node in a global variable
             thisModule.Model <- tModel ;
```

```
            }

            let s : String = 'CreateGlobalModelAndApplySoaMLProfile => END'
            in s.println();
       }
      }

      --###################   règles de transformation d'une Collaboration en Community-level ServicesArchitecture
(CIM-Bas) vue détaillée #################

      --  règle de transformation d'une Collaboration BPMN en:
      --      +un Model SoaML
      --      +une ServicesArchitecture SoaML
      --  un Model est affecté à chaque Collaboration
      rule Collaboration2ServicesArchitecture
      {
       from sCollaboration : BPMNMM!Collaboration
       to   tCollaboration : UML2!Collaboration (
                  name <- sCollaboration.name.debug().debug('tCollaboration.name'),
                  ownedAttribute <- Sequence{sCollaboration.participants}.debug('tCollaboration.ownedAttribute'),
                  collaborationUse  <-  Sequence{sCollaboration.messageFlows}.debug('tCollaboration.collaborationUse')

            )
       do {
            let s : String = 'RULE:Collaboration2ServicesArchitecture => BEGIN'
            in s.println();

            thisModule.CreateGlobalModelAndApplySoaMLProfile(tCollaboration);

            --****************** BEGIN OF Stereotype Application ******************

            tCollaboration.applyStereotype(thisModule.getStereotype('ServicesArchitecture')).debug('applyStereotype
: ServicesArchitecture');
            --****************** END OF Stereotype Application ********************

            let s : String = 'RULE:Collaboration2ServicesArchitecture => END'
            in s.println();
       } -- end of 'do' section
      } -- end of 'rule' section
```

```
    -- @Todo Le traitement n'est pas le même si le Participant référence un Processus ou pas
    -- règle de transformation d'un Participant BPMN en:
    --    +une Property SoaML : chaque Property est de type Participant
    --    +un Participant SoaML
    rule Participant2PropertyAndParticipant
    {
     from sParticipant : BPMNMM!Participant
     to   tProperty : UML2!Property (
                name <- sParticipant.name.debug('tProperty.name'),
                type <- tParticipant.debug('tProperty.type')
          ),
          tParticipant : UML2!Class (
                name <- sParticipant.name.debug('tParticipant.name')
          )
     do {
          let s : String = 'RULE:Participant2Property => BEGIN'
          in s.println();

          --add participant to model
          thisModule.Model.packagedElement <- Sequence{tParticipant};

          --****************** BEGIN OF Stereotype Application ******************

          tParticipant.applyStereotype(thisModule.getStereotype('Participant')).debug('applyStereotype            :
Participant');
          tProperty.applyStereotype(thisModule.getStereotype('Property')).debug('applyStereotype : Property');
          --****************** END OF Stereotype Application *******************

          let s : String = 'RULE:Participant2Property => END '
          in s.println();
     } -- end 'do' section
    } -- end of 'rule' section

    -- règle de transformation d'un MessageFlow SoaML en:
    --    + une CollaborationUse de type ServiceContract
    --    + une ServiceContract SoaML
    --    + deux tDependency UML chacune relie un participant à une CollaborationUse
    rule MessageFlow2CollaborationUse{
```

```
	from sMessageFlow : BPMNMM!MessageFlow
	using {
		sParticipantSource : BPMNMM!Participant = sMessageFlow.sourceRef;
		sParticipantTarget : BPMNMM!Participant = sMessageFlow.targetRef;
	}
	to tCollaborationUse : UML2!CollaborationUse (
		name <- (sMessageFlow.name + '_CollUse').debug('tCollaborationUse.name'),
		type <- tServiceContract.debug('tCollaborationUse.type'),
		roleBinding <- Sequence{tDependencyRequest,tDependencyService}.debug('tCollaborationUse.roleBinding')
	),
	tServiceContract : UML2!Collaboration (
		name <- (sMessageFlow.name +'_SerContra').debug('tServiceContract.name')
	),
	tDependencyRequest : UML2!Dependency (
		name <- 'roleRequest_name'.debug('tDependencyRequest.name'),
		supplier <- tServiceContract.debug('tDependencyRequest.supplier'),
		client <- thisModule.resolveTemp(sMessageFlow.sourceRef, 'sParticipantSource'),
		visibility <- #private
	),
	tDependencyService : UML2!Dependency(
		name <- 'roleService_name'.debug('tDependencyService.name'),
		supplier <- tServiceContract.debug('tDependencyService.supplier'),
		client <- thisModule.resolveTemp(sMessageFlow.sourceRef, 'sParticipantTarget'),
		visibility <- #private
	)

	do {
		let s : String = 'RULE:MessageFlow2CollaborationUse => BEGIN'
		in s.println();

		thisModule.Model.packagedElement <- Sequence{tServiceContract}.debug('add tServiceContract to Model');

		--****************** BEGIN OF Stereotype Application ******************
		tCollaborationUse.applyStereotype(thisModule.getStereotype('CollaborationUse')).debug('applyStereotype  :
CollaborationUse');
		tServiceContract.applyStereotype(thisModule.getStereotype('ServiceContract')).debug('applyStereotype    :
ServiceContract');
		--****************** END OF Stereotype Application ******************
```

```
            let s : String = 'RULE:MessageFlow2CollaborationUse => END'
            in s.println();
      } -- end 'do' section
    } -- end of 'rule' section

    --###################  règles de transformations d'un Processus en Community-level ServicesArchitecture (PIM)
#################
    --###################  règles de transformation d'un sous Processus en Participant-level ServicesArchitecture
(PIM)  #################

    -- règle de transformation d'un Process BPMN en:
    --        + un Model SoaML s'il n'est pas déjà créé
    --        + une ServicesArchitecture SoaML (Participant-Level)
    rule Process2ServicesArchitecture
    {
     from sProcess : BPMNMM!Process
     using {
            -- @Todo vérifier que le sequenceFlows lie deux FlowNode qui appartiennent à deux Lanes différentes
            sequenceFlows    :    Sequence    (BPMNMM!SequenceFlow)    =        sProcess.flowElements->select(s    |
s.oclIsTypeOf(BPMNMM!SequenceFlow))->
                                        collect(e   |   if   e.sourceRef.lanes.first().debug('lane   source:   ')   <>
e.targetRef.lanes.first().debug('lane target: ') then e else OclUndefined endif);
            -- séléctionner seulement  les Flownode de type Gatway, Activity et Event sont récupérés
            flowNodes : Sequence (BPMNMM!FlowNode) = (sequenceFlows->collect(e | e.sourceRef)->union(sequenceFlows-
>collect(e | e.targetRef)))
                                                                        ->select(fn            |            not
fn.oclIsTypeOf(BPMNMM!ChoreographyActivity));
            --@Todo traiter les Lanes imbriqués
            lanes : OrderedSet (BPMNMM!Lanes) = flowNodes->collect(fn | fn.lanes);
      }
      to tCollaboration : UML2!Collaboration(
            name <- sProcess.name.debug('tCollaboration.name'),
            ownedAttribute <- lanes.debug('tCollaboration.ownedAttribute'),
            collaborationUse <- sequenceFlows.debug('tCollaboration.collaborationUse')
      )
      do {
            let s : String = 'RULE:Process2ServicesArchitecture => BEGIN'
            in s.println();
```

```
            thisModule.CreateGlobalModelAndApplySoaMLProfile(tCollaboration);
            --****************** BEGIN OF Stereotype Application ******************

            tCollaboration.applyStereotype(thisModule.getStereotype('ServicesArchitecture')).debug('applyStereotype
: ServicesArchitecture');
            --****************** END OF Stereotype Application ******************

            let s : String = 'RULE:Process2ServicesArchitecture => END'
            in s.println();
      } -- end 'do' section
     } -- end of rule section

     -- règle de transformation d'un SubProcess BPMN en:
     --        + un Model SoaML s'il n'est pas déjà créé
     --        + une ServicesArchitecture SoaML (Participant-Level)
     rule SubProcess2ServicesArchitecture
     {
      from sSubProcess : BPMNMM!SubProcess
      using {
            -- @Todo vérifier que le sequenceFlows lie deux FlowNode qui appartiennent à deux Lanes différentes
            sequenceFlows    :    Sequence    (BPMNMM!SequenceFlow)    =    sSubProcess.flowElements->select(s    |
s.oclIsTypeOf(BPMNMM!SequenceFlow)).debug();
            -- séléctionner seulement  les Flownode de type Gatway, Activity et Event sont récupérés
            flowNodes : Sequence (BPMNMM!FlowNode) = (sequenceFlows->collect(e | e.sourceRef)->union(sequenceFlows-
>collect(e | e.targetRef)))
                                                                  ->select(fn            |            not
fn.oclIsTypeOf(BPMNMM!ChoreographyActivity));
            --@Todo traiter les Lanes imbriqués
            lanes : OrderedSet (BPMNMM!Lanes) = flowNodes->collect(fn | fn.lanes);
      }
      to tCollaboration : UML2!Collaboration (
            name <- sSubProcess.name.debug('tCollaboration.name'),
            ownedAttribute <- Sequence{lanes}.debug('tCollaboration.ownedAttribute'),
            collaborationUse <- Sequence{sequenceFlows}.debug('tCollaboration.collaborationUse')
      )
      do {
            let s : String = 'RULE:SubProcess2ServicesArchitecture => BEGIN'
            in s.println();
            thisModule.CreateGlobalModelAndApplySoaMLProfile(tCollaboration);
```

```
            --****************** BEGIN OF Stereotype Application ******************

            tCollaboration.applyStereotype(thisModule.getStereotype('ServicesArchitecture')).debug('applyStereotype
: ServicesArchitecture');
            --****************** END OF Stereotype Application ******************

            let s : String = 'RULE:SubProcess2ServicesArchitecture => END'
            in s.println();
      } -- end of 'do' section
    } -- end of 'rule' section
    -- règle de transformation de chaque Lane BPMN en:
    --    +une Property SoaML : chaque Property est de type Participant
    --    +un Participant SoaML
    rule Lane2PropertyAndParticipant
    {
     from sLane: BPMNMM!Lane
     to   tProperty : UML2!Property (
                  name <- sLane.name.debug('tProperty.name'),
                  type <- tParticipantFromLane.debug('tProperty.type')
            ),
            tParticipantFromLane : UML2!Class (
                  name <- sLane.name.debug('tParticipantFromLane.name')
            )
     do {
            let s : String = 'RULE:Lane2PropertyAndParticipant => BEGIN'
            in s.println();

            --add participants to model
            thisModule.Model.packagedElement <- Sequence{tParticipantFromLane};
            --****************** BEGIN OF Stereotype Application ******************

            tParticipantFromLane.applyStereotype(thisModule.getStereotype('Participant')).debug('applyStereotype
Participant to Lane');
            tProperty.applyStereotype(thisModule.getStereotype('Property')).debug('applyStereotype    Property    to
Lane');
            --****************** END OF Stereotype Application ******************

            let s : String = 'RULE:Lane2PropertyAndParticipant => END'
```

```
        in s.println();
  } -- end 'do' section

} -- end of 'rule' section

-- règle de transformation d'un SequenceFlow SoaML en:
--    + une CollaborationUse de type ServiceContract
--    + une ServiceContract SoaML
rule SequenceFlow2CollaborationUse {
 from sSequenceFlow : BPMNMM!SequenceFlow
 using {
       -- récupération des Lanes source
       -- @Todo traitement des Lanes imbriquées
       sFlowNodeSource : BPMNMM!FlowNode = sSequenceFlow.sourceRef;
       laneSource : BPMNMM!Lanes = sFlowNodeSource.lanes.first();

       -- récupération des Lanes cibles
       -- @Todo traitement des Lanes imbriquées
       sFlowNodeTarget : BPMNMM!FlowNode = sSequenceFlow.targetRef;
       laneCible : BPMNMM!Lanes = sFlowNodeTarget.lanes.first();
 }
 to tCollaborationUse : UML2!CollaborationUse (
       name <- (sSequenceFlow.name + '_CollUse').debug('tCollaborationUse.name'),
       type <- tServiceContract.debug('tCollaborationUse.type'),
       roleBinding <- Sequence{tDependencyRequest,tDependencyService}.debug('tCollaborationUse.roleBinding')
 ),
 tServiceContract : UML2!Collaboration (
       name <- (sSequenceFlow.name +'_SerContra').debug('tServiceContract.name')
 ),
 tDependencyRequest : UML2!Dependency (
       name <- 'roleRequest_name'.debug('tDependencyRequest.name'),
       supplier <- tServiceContract.debug('tDependencyRequest.supplier'),
       client <- tRequest.debug('tDependencyRequest.client'),
       visibility <- #private.debug('tDependencyRequest.visibility')
 ),
 tDependencyService : UML2!Dependency(
       name <- 'roleService_name'.debug('tDependencyService.name'),
       supplier <- tServiceContract.debug('tDependencyService.supplier'),
       client <- tService.debug('tDependencyService.client'),
```

```
            visibility <- #private.debug('tDependencyService.visibility')
      ),
      -- Les Ports de chaque participant doivent être typés par une Interface ou une ServiceInterface
      -- c'est au développeur de choisir. ici nous faisons une transformation vers une ServiceInterface
      tInterface : UML2!Interface(
                  name <- 'Interface_name'.debug('Interface.name')
      ),
      tRequest : UML2!Port(
                  name <- (sSequenceFlow.name+'_request').toLower().debug('tRequest.name'),
                  type <- tInterface.debug('tRequest.tInterface')
                  -- The isConjugated property of a «Request» must be set to true (SoaML p 81)
                  -- isConjugated <- true
      ),
      tService : UML2!Port(
                  name <- (sSequenceFlow.name+'_service').toLower().debug('tService.name'),
                  type <- tInterface.debug('tService.tInterface')
      )
      do {
            let s : String = 'RULE:SequenceFlow2CollaborationUse => BEGIN'
            in s.println();

            -- ajouts des Ports dans les participants correspondants
            thisModule.resolveTemp(laneSource,  'tParticipantFromLane').ownedPort  <-  Sequence{tRequest}.debug('add
tRequest Port to sFlowNodeSource');
            thisModule.resolveTemp(laneCible,  'tParticipantFromLane').ownedPort  <-  Sequence{tService}.debug('add
tService Port to sFlowNodeTarget');
            -- ajout du ServiceContract et du Interface dans le Model UML
            thisModule.Model.packagedElement <- Sequence{tServiceContract}.debug('add tServiceContract to Model');
            thisModule.Model.packagedElement <- Sequence{tInterface}.debug('add tInterface');

            --****************** BEGIN OF Stereotype Application ******************
            tCollaborationUse.applyStereotype(thisModule.getStereotype('CollaborationUse')).debug('applyStereotype  :
CollaborationUse');
            tServiceContract.applyStereotype(thisModule.getStereotype('ServiceContract')).debug('applyStereotype   :
ServiceContract');
            tRequest.applyStereotype(thisModule.getStereotype('Request')).debug('applyStereotype : Request');
            tService.applyStereotype(thisModule.getStereotype('Service')).debug('applyStereotype : Service');
            tInterface.applyStereotype(thisModule.getStereotype('Interface')).debug('applyStereotype : Interface');
            --****************** END OF Stereotype Application ******************
```

```
        let s : String = 'RULE:SequenceFlow2CollaborationUse => END'
        in s.println();
  } -- end of 'do' section
} -- end of 'rule' section
```

Annexe C : Transformations entre SoaML et les langages spécifiques d'une plateforme web-services : XSD, WSDL, WS-BPEL

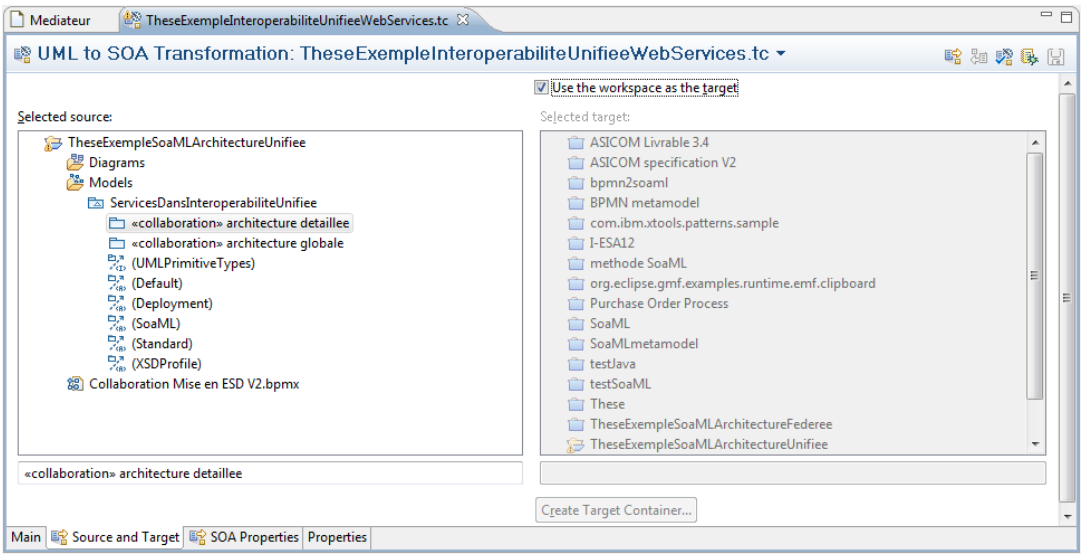


Figure 0-1. Transformation des éléments SoaML de l'architecture détaillée vers XSD, WSDL et WS-BPEL avec RSA

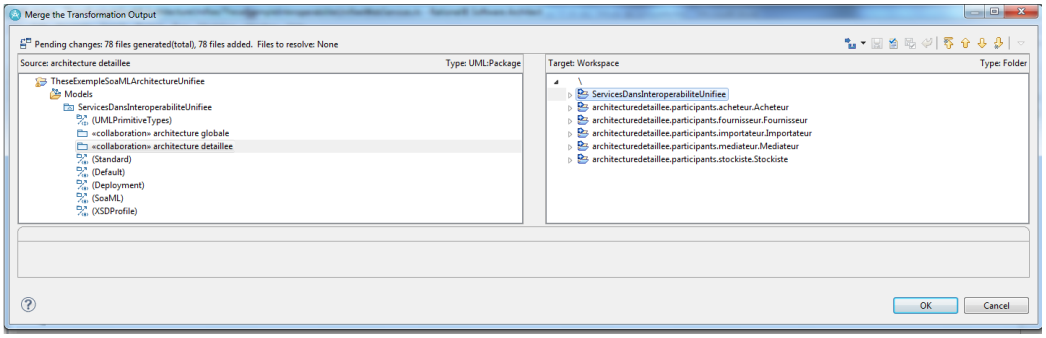


Figure 0-2. Transformation des éléments de l'architecture détaillée des services en 6 projets différents dans RSA

Pour chaque participant de la collaboration un projet est généré (i.e., 5 projets) ce qui fait un total de 5 projets. Puis, un 6ème projet contient les données métiers (XSD) et les interfaces de la collaboration.

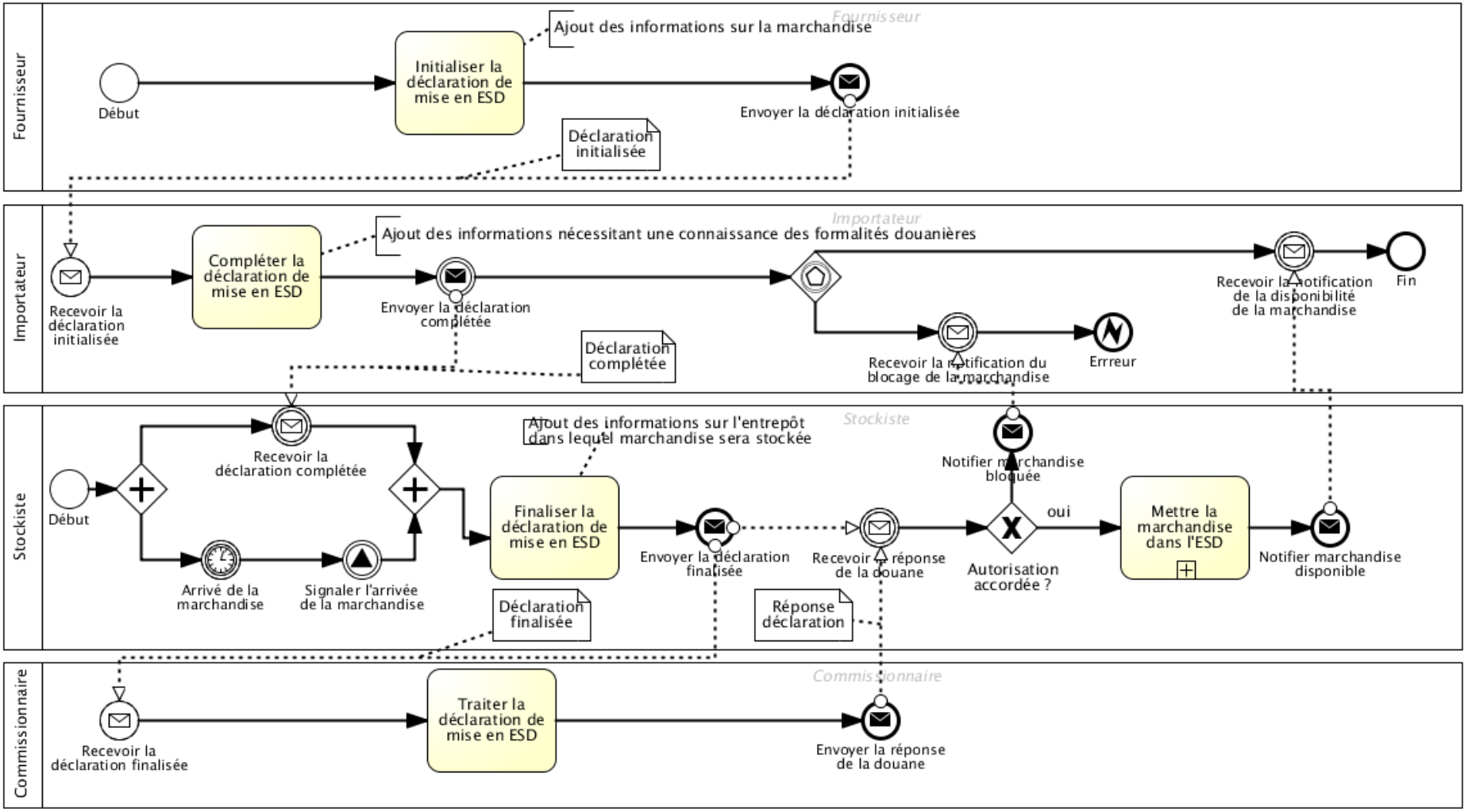
Annexe D : Code Source du WS-BPEL généré par RSA

```
<?xml version="1.0" encoding="UTF-8"?><definitions name="Mediateur"
targetNamespace="http://architecturedetaillee/participants/mediateur/MediateurArtifacts/"
xmlns="http://schemas.xmlsoap.org/wsdl/"
xmlns:ns="http://architecturedetaillee/servicesinterfaces/DeclarationSI/DemandeMiseEnESD/"
xmlns:ns0="http://architecturedetaillee/servicesinterfaces/DeclarationSI/TraitementMiseEnESD/"
xmlns:ns1="http://architecturedetaillee/simpleinterfaces/InitialisationDeclaration/"
xmlns:ns2="http://architecturedetaillee/servicesinterfaces/CompletionDeclarationSI/DemandeCompletionDeclaration/"
xmlns:ns3="http://architecturedetaillee/simpleinterfaces/CompletionDeclaration/"
xmlns:ns4="http://architecturedetaillee/servicesinterfaces/FinalisationDeclarationSI/DemandeFinalisationDeclaration/
" xmlns:ns5="http://architecturedetaillee/simpleinterfaces/FinalisationDeclaration/"
xmlns:ns6="http://architecturedetaillee/servicesinterfaces/TraitementDeclarationSI/DemandeTraitementDeclaration/"
xmlns:ns7="http://architecturedetaillee/simpleinterfaces/TraitementDeclaration/"
xmlns:plink="http://schemas.xmlsoap.org/ws/2004/03/partner-link/"
xmlns:tns="http://architecturedetaillee/participants/mediateur/MediateurArtifacts/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <plink:partnerLinkType name="DemandeMiseEnESDPLT">
    <plink:role name="DemandeMiseEnESDRole">
      <plink:portType name="ns:DemandeMiseEnESD"/>
    </plink:role>
  </plink:partnerLinkType>
  <plink:partnerLinkType name="TraitementMiseEnESDPLT">
    <plink:role name="TraitementMiseEnESDRole">
      <plink:portType name="ns0:TraitementMiseEnESD"/>
    </plink:role>
  </plink:partnerLinkType>
  <plink:partnerLinkType name="InitialisationDeclarationPLT">
    <plink:role name="InitialisationDeclarationRole">
      <plink:portType name="ns1:InitialisationDeclaration"/>
    </plink:role>
  </plink:partnerLinkType>
  <plink:partnerLinkType name="DemandeCompletionDeclarationPLT">
    <plink:role name="DemandeCompletionDeclarationRole">
      <plink:portType name="ns2:DemandeCompletionDeclaration"/>
    </plink:role>
  </plink:partnerLinkType>
  <plink:partnerLinkType name="CompletionDeclarationPLT">
    <plink:role name="CompletionDeclarationRole">
```

```
      <plink:portType name="ns3:CompletionDeclaration"/>
    </plink:role>
  </plink:partnerLinkType>
  <plink:partnerLinkType name="DemandeFinalisationDeclarationPLT">
    <plink:role name="DemandeFinalisationDeclarationRole">
      <plink:portType name="ns4:DemandeFinalisationDeclaration"/>
    </plink:role>
  </plink:partnerLinkType>
  <plink:partnerLinkType name="FinalisationDeclarationPLT">
    <plink:role name="FinalisationDeclarationRole">
      <plink:portType name="ns5:FinalisationDeclaration"/>
    </plink:role>
  </plink:partnerLinkType>
  <plink:partnerLinkType name="DemandeTraitementDeclarationPLT">
    <plink:role name="DemandeTraitementDeclarationRole">
      <plink:portType name="ns6:DemandeTraitementDeclaration"/>
    </plink:role>
  </plink:partnerLinkType>
  <plink:partnerLinkType name="TraitementDeclarationPLT">
    <plink:role name="TraitementDeclarationRole">
      <plink:portType name="ns7:TraitementDeclaration"/>
    </plink:role>
  </plink:partnerLinkType>
  <import
location="../../../../ServicesDansInteroperabiliteUnifiee/architecturedetaillee/servicesinterfaces/DeclarationSI/Dem
andeMiseEnESD.wsdl" namespace="http://architecturedetaillee/servicesinterfaces/DeclarationSI/DemandeMiseEnESD/"/>
  <import
location="../../../../ServicesDansInteroperabiliteUnifiee/architecturedetaillee/servicesinterfaces/DeclarationSI/Tra
itementMiseEnESD.wsdl"
namespace="http://architecturedetaillee/servicesinterfaces/DeclarationSI/TraitementMiseEnESD/"/>
  <import
location="../../../../ServicesDansInteroperabiliteUnifiee/architecturedetaillee/simpleinterfaces/Initialis
ationDeclaration.wsdl"
namespace="http://architecturedetaillee/simpleinterfaces/InitialisationDeclaration/"/>
  <import
location="../../../../ServicesDansInteroperabiliteUnifiee/architecturedetaillee/servicesinterfaces/Complet
ionDeclarationSI/DemandeCompletionDeclaration.wsdl"
```

```
namespace="http://architecturedetaillee/servicesinterfaces/CompletionDeclarationSI/DemandeCompletionDeclar
ation/"/>
  <import
location="../../../../ServicesDansInteroperabiliteUnifiee/architecturedetaillee/simpleinterfaces/Completio
nDeclaration.wsdl" namespace="http://architecturedetaillee/simpleinterfaces/CompletionDeclaration/"/>
  <import
location="../../../../ServicesDansInteroperabiliteUnifiee/architecturedetaillee/servicesinterfaces/Finalis
ationDeclarationSI/DemandeFinalisationDeclaration.wsdl"
namespace="http://architecturedetaillee/servicesinterfaces/FinalisationDeclarationSI/DemandeFinalisationDe
claration/"/>
  <import
location="../../../../ServicesDansInteroperabiliteUnifiee/architecturedetaillee/simpleinterfaces/Finalisat
ionDeclaration.wsdl" namespace="http://architecturedetaillee/simpleinterfaces/FinalisationDeclaration/"/>
  <import
location="../../../../ServicesDansInteroperabiliteUnifiee/architecturedetaillee/servicesinterfaces/Traitem
entDeclarationSI/DemandeTraitementDeclaration.wsdl"
namespace="http://architecturedetaillee/servicesinterfaces/TraitementDeclarationSI/DemandeTraitementDeclar
ation/"/>
  <import
location="../../../../ServicesDansInteroperabiliteUnifiee/architecturedetaillee/simpleinterfaces/Traitemen
tDeclaration.wsdl" namespace="http://architecturedetaillee/simpleinterfaces/TraitementDeclaration/"/>
  <types>
    <xsd:schema
targetNamespace="http://architecturedetaillee/participants/mediateur/MediateurArtifacts/"/>
  </types>
   </definitions>
```

Annexe E : La coopération avec une vue distribuée « mettre la marchandise en ESD » dans le cadre d'une approche d'interopérabilité fédérée



Titre : Proposition d'une méthode de spécification d'une architecture orientée services dirigée par le métier dans le cadre d'une collaboration inter-organisationnelle

Résumé : Les organisations contemporaines collaborent de plus en plus avec leurs partenaires. Cette dimension ouverte leur permet d'être plus réactives face aux changements que leur imposent leurs environnements. La caractéristique de la collaboration est due, pour les entreprises, au nouvel environnement économique, qui fait de l'interopérabilité et l'agilité deux des principaux résultats que les entreprises doivent atteindre. Ce contexte correspond au cadre global de nos travaux, qui porte sur la question suivante : Comment concevoir une architecture orientée services dirigée par le métier dans le cadre d'une collaboration inter-organisationnelle ?
L'intérêt de ce travail est de proposer une méthode qui assure l'efficacité et l'efficience d'une collaboration, en utilisant les principes de BPM et SOA pour dépasser les barrières conceptuelle et technologique de l'interopérabilité. On explique comment identifier, spécifier et réaliser les processus et les services de collaboration entre différents participants. Pour cela, on adopte une vision transversale de l'entreprise centrée sur les processus métiers. Ensuite, l'approche MDA est utilisée comme un fil conducteur pour synchroniser les modèles des processus métiers découverts à l'aide de l'approche BPM avec ceux des services identifiés avec la démarche SOA. Dans ce schéma, les processus métiers assurent l'interopérabilité au niveau métier tandis que l'utilisation des services réutilisables, des standards et des architectures préconisés par SOA soutiennent l'interopérabilité au niveau IT.
La méthode proposée s'articule autour de 22 étapes qui se déploient sur les différents niveaux de l'approche MDI. Elle se base sur un style de modélisation hiérarchique avec des diagrammes de haut niveau qui sont ensuite enrichis à des niveaux plus bas. BPMN 2.0 est utilisé pour modéliser les processus, alors que SoaML est utilisé pour spécifier les services.

Mots clés : Méthode de conception, Collaboration, Interopérabilité, Processus collaboratif, service métier, SOA, BPM, MDI, transformation de modèles, BPMN 2.0, SoaML, ESB.

Title: Proposition of a service oriented architecture methodology driven by business to support inter-organizational collaboration

Abstract: Global acceleration of exchanges in goods and services requires organizations to adopt an open view beyond their own boundaries at both business and technological levels. In the new economic environment enterprises must achieve both interoperability and agility. In this thesis the main research question is the following: How to design a service oriented architecture methodology driven by business to support inter-organizational collaboration?
To overcome the conceptual and technological barriers of interoperability, we propose a top-down model driven method based on BPM and SOA principles to ensure collaboration efficiency and effectiveness. The proposed method explains how to identify, specify and implement collaborative processes and collaborative public services. In this method business processes ensure interoperability at the business level, while reusable services, standards and SOA platform support interoperability at the IT level.
The proposed method is based on 22 steps that cover the different levels of the MDI approach. It is based on a hierarchical modeling style that use BPMN 2.0 notation to model processes and SoaML language to specify services.

Keywords: Design method, Collaboration, Interoperability, Collaborative process, business services, SOA, BPM, MDI, Model transformation, BPMN 2.0, SoaML, ESB.