INSTITUT NATIONAL POLYTECHNIQUE DE TOULOUSE

THESE

Pour obtenir le grade de

DOCTEUR DE L'INPT

Spécialité : SYSTEMES INDUSTRIELS

Préparée au Centre de Génie Industriel - Ecole des Mines d'Albi Carmaux

Présentée et soutenue publiquement

Par

Jihed TOUZI

le 09/11/2007

Titre :

Aide à la conception de Système d'Information Collaboratif support de l'interopérabilité des entreprises

Directeur de thèse : Pr. Hervé PINGAUD
Encadrant : Frédérick BENABEN

JURY

Khalid BENALI	Maître de Conférences, HDR, Nancy-Université	Rapporteur
Jean-Pierre BOUREY	Professeur Ecole Centrale de Lille	Rapporteur
David CHEN	Professeur Université de Bordeaux	Examinateur
Jean-Pierre LORRE	Responsable R&D EBM WebSourcing	Examinateur
Hervé PINGAUD	Professeur Ecole des Mines d'Albi	Directeur de thèse
Frédérick BENABEN	Maître-Assistant Ecole des Mines d'Albi	Examinateur

Dédicace

Jamais cette thèse n'aurait vu le jour, sans l'amour et l'affection que j'ai pu trouver au sein de ma famille dès mon enfance…

A mes parents qui m'ont encouragé pendant les moments difficiles de la thèse… leur présence à la soutenance reste un des meilleurs souvenirs…

A mon frère pour la période de la rédaction où sa présence à mes côtés a été capitale pour me permettre de me consacrer à mes travaux…

A ma petite chouchou dont les mots d'encouragement ont traversé la méditerranée pour me soutenir…

A ma grande mère… reposes en paix… je t'aime…

Remerciements

Mes remerciements s'adressent en premier lieu à *Monsieur Hervé PINGAUD*, directeur de thèse et *Monsieur Fréderick BENABEN*, encadrant de thèse. Leurs grandes compétences et précieux conseils ont permis l'accomplissement de ce travail. Leur confiance en mes capacités a contribué à me faire aimer ces trois ans de recherche. Je les remercie chaleureusement pour leur contact constructif, si généreux sur le plan humain comme scientifique.

Je remercie *Monsieur Jean Pierre LORRE* pour sa collaboration fructueuse pendant cette thèse et de m'avoir aidé à acquérir une vision industrielle, si intéressante, de mes travaux.

Je remercie chaleureusement *Monsieur Jean Pierre BOUREY* et *Monsieur Khalid BENALI* pour avoir accepté de rapporter cette thèse. Je leur exprime toute ma reconnaissance pour l'intérêt porté à ce travail. Leurs remarques ont apporté matière à ma réflexion.

Je suis, également, très sensible à l'honneur que m'a fait Monsieur *David CHEN*, en acceptant de présider mon jury.

Cette thèse a été réalisée au Centre de Génie Industriel de l'ENSTIMAC, tout naturellement je tiens donc à remercier son directeur, *Monsieur Lionel DUPONT* pour toute l'attention qu'il m'a portée et pour les moyens mis à ma disposition durant ces trois années.

Je remercie *Mademoiselle Isabelle FOURNIER* pour son professionnalisme dans le travail et son aide si précieuse, comme dans son accompagnement au quotidien.

Je remercie tout le personnel d'EBMWebsourcing, tout particulièrement *Monsieur Bertrand ESCUDIE* et *Monsieur Gaël BLONDELLE* pour leur accueil si chaleureux pendant les six mois passés à Toulouse.

Je remercie tous les membres du Centre de Génie Industriel qu'il m'a été donné de rencontrer au cours de ces trois ans pour la bonne ambiance et leur aide : *Jacques* pour les discussions scientifiques (et autres) intéressantes, *Didier* pour ses encouragements et ses conseils, *Matthieu* pour avoir été un bon guide dans le métro pékinois, *Paul* pour l'occasion qu'il m'adonné pour intervenir dans le cours XML, *Franck* pour sa sympathie et sa bonne humeur, *Hung* pour le thé vietnamien les samedis, *Netty* pour les beaux moments, *Carmen* pour sa joie de vivre …et tant d'autres …

Je remercie tous les amis de l'ENSTIMAC (les autres centres, service informatique, accueil, RH, etc.) mais aussi les amis footballeurs pour les moments sympathiques…

SOMMAIRE

Chapitre III : Concept de Système d'Information Collaboratif 79

Chapitre IV : Conception de Système d'Information Collaboratif 101

LISTE DES FIGURES

Chapitre I

Chapitre II

Chapitre III

Chapitre IV

Chapitre V

Chapitre VI

« rien ne se perd, rien ne se crée, tout se transforme… »
Antoine-Laurent de Lavoisier
Chimiste français

Introduction générale

Introduction générale

Les répercussions de la mondialisation ont obligé, et obligent encore actuellement, les entreprises à adopter de nouveaux schémas de comportement, à modifier en profondeur leur organisation et à s'ouvrir davantage à leur environnement. Ces entreprises s'organisent aujourd'hui en réseaux, au sein desquels interagissent différents acteurs (fournisseurs, donneurs d'ordres, sous-traitants, etc.) dans une stratégie « gagnant / gagnant ». Cependant, il existe plusieurs façons d'établir des collaborations entre entreprises. Celles-ci peuvent être caractérisées par bon nombre de facteurs, tels que la nature des relations binaires entre partenaires, la topologie ou l'objectif du réseau, la fréquence de la collaboration, etc. Dans ce contexte, nous verrons qu'il est possible de parler de niveaux de collaboration, mais également de stade de maturité collaborative.

La dynamique remarquable du contexte économique semble nécessiter de la part des acteurs d'un « réseau d'entreprises » une capacité d'adaptation et de réactivité toujours plus grande. Il est indispensable de pouvoir répondre rapidement aux évolutions du marché. La qualité et le potentiel de collaboration des entreprises deviennent par conséquent des facteurs déterminants dans l'évaluation de leur capacité de survie. L'adaptation et la réactivité de l'entreprise passent par sa capacité à interagir efficacement avec l'ensemble des acteurs. Il s'agit donc de faire tomber les barrières culturelles, organisationnelles, fonctionnelles et technologiques au sein des entreprises, afin que l'ensemble soit vu comme un **tout cohérent**.

Pour faire face à ces nouvelles exigences, les entreprises ont souvent recours au concept d'intégration et parfois, de façon plus spécifique, au concept d'interopérabilité.

L'interopérabilité peut être vue comme la capacité des entreprises à structurer, formaliser et présenter leurs connaissances et savoir-faire afin d'être en mesure de les échanger ou de les partager. Dans ce cas, l'interopérabilité est une aptitude cruciale des entreprises, si elles souhaitent s'inscrire dans une dynamique d'intégration. L'interopérabilité est désormais incontournable pour assurer la pérennité économique de l'entreprise.

La question de l'interopérabilité des entreprises et de leur capacité à interagir efficacement impacte nécessairement le domaine des Systèmes d'Information (SI). Si la définition du concept même de « système d'information » est une tâche délicate du point de vue « informatique », ce système peut être considéré comme un ensemble d'interactions entre processus, données et applications de l'entreprise. Dans ce cas, le comportement, la réactivité et la dynamique de l'entreprise reposent sur le système d'information au travers des processus qu'il supporte, des applications qu'il propose et des données qu'il gère.

Partant de ce constat de forte dépendance entre l'entreprise et son système d'information, nous considérons dans nos travaux que l'interopérabilité des entreprises passe par celle de leurs systèmes d'information. L'interopérabilité d'un système d'information avec d'autres systèmes d'information permet de former un système virtuel cohérent que nous appelons Système de Systèmes (SdS). Dans nos travaux, nous souhaitons projeter la notion d'interopérabilité sur le concept même du système d'information, afin de déduire les différentes exigences et caractéristiques qui définissent cette interopérabilité des systèmes d'information.

L'interopérabilité des systèmes d'information peut, dans un premier temps, être vue au travers d'une adaptation technologique de la projection informatique de ces SI. Cette tendance se

base sur les remarquables avancées actuelles en termes de technologies de communication (EDI, web-services, Internet, places de marché, EAI, ESB, etc.). L'usage de ces outils est aujourd'hui au cœur du questionnement sur l'évolution des systèmes d'information. Pourtant, les réponses ne résident pas seulement dans le domaine technique : il est fondamental de prendre en considération d'autres espaces d'investigation tels que les niveaux métier et sémantique. Ce sont donc trois niveaux d'interopérabilité que nous allons prendre en compte dans ce document : le niveau *métier* (qui concerne l'organisation, les objectifs, les processus et les responsabilités), le niveau *sémantique* (qui concerne les connaissances et les informations) et le niveau *technique* (qui concerne les choix et moyens technologiques accessibles). Nos travaux de thèse étudient la relation entre ces différents niveaux d'interopérabilité et les concepts vis à vis desquels nous souhaitons mettre l'interopérabilité en regard : les systèmes d'information et les solutions technologiques. Ainsi, les composants des systèmes d'information devront être positionnés vis-à-vis des trois niveaux de l'interopérabilité, tout comme les différentes solutions, approches, standards et technologies qui doivent supporter concrètement l'interopérabilité. Pour traiter de cette question de l'interopérabilité des systèmes d'information, nous allons nous baser sur deux constats.

Tout d'abord, l'interopérabilité des systèmes d'information peut être réalisée de différentes manières en se basant sur différentes architectures de systèmes d'information. Une solution directe au problème de l'interopérabilité réside dans l'adoption de standards. Si cette approche radicale résout efficacement ce problème en assurant une compatibilité des informations des systèmes d'information par uniformisation des formats, il faut se poser la question du coût et des impacts de cette solution descendante pour l'entreprise. Une autre approche, plus pragmatique, consiste à garder l'intégrité des systèmes d'information et à faire supporter l'interopérabilité de ces systèmes par un système intermédiaire. Ce système fournit l'interopérabilité au moment où il y en a besoin, en traitant l'hétérogénéité des données, processus et applications des SI. Cette approche permet d'assimiler la problématique d'interaction directe entre SI à une problématique de *médiation* entre ces SI, assurée par le système d'information intermédiaire.

Dans un second temps, nous sommes amenés à constater l'émergence d'approches d'architectures de systèmes d'information, telles que SOA (*Service-Oriented Architecture*), qui apportent une certaine simplification des interconnexions entre applications et la facilitation de l'établissement et de la reconfiguration de processus collaboratifs (ou inter-organisationnels).

Fort de ces deux constats, nos travaux s'intéressent à une approche de *médiation* entre systèmes d'information (dans le but de gérer leur interopérabilité sur tous ses niveaux) et ce, dans un contexte *SOA* (afin d'évaluer les apports de cette philosophie en tant qu'architecture améliorant l'agilité et l'accessibilité des systèmes d'information). Cependant, il est important de définir clairement la spécification d'une telle solution de médiation entre acteurs SOA : sur la base de quelles fonctionnalités et de quelles responsabilités ce système de médiation permet-il de faire interopérer les SI ? Cette solution permet-elle de supporter les nécessaires aspects complémentaires émergents de l'association de systèmes (selon l'adage : « les propriétés de la somme des systèmes sont supérieures à la somme des propriétés des systèmes »), par exemple en supportant des services émergents à la collaboration ?

Une partie essentielle de ce manuscrit de thèse s'intéresse à la conception d'un tel système de médiation vu en tant que vecteur de l'interopérabilité de systèmes d'information hétérogènes. Nous y décomposons la démarche proposée selon ses projections *métier*, *logique* et *technique*. Les pratiques proposées s'apparentent à l'approche MDA (*Model Driven Architecture*) et permettent d'anticiper sur l'outillage d'un atelier de transformation de modèles.

Au niveau *métier*, il s'agit de déterminer une modélisation (*Business Modelling*) qui permette de couvrir les spécificités, les choix et les caractéristiques de la collaboration. Nous traiterons également dans ce manuscrit de la crédibilité des modèles de processus en tant que candidats à cette tâche, du fait de leur capacité à représenter l'organisation, les échanges d'informations et les interactions entre partenaires. Il existe différents formalismes de modélisation de processus qui s'acquittent différemment (et souvent partiellement) de la représentation des différents points de vue de la collaboration.

Le niveau *logique* consiste à spécifier les fonctionnalités (services) mises en œuvre dans la collaboration (en plus des données échangées et de l'ordre des échanges entre partenaires) en respectant une architecture particulière (SOA).

Enfin, le niveau *technique* résulte d'une projection du niveau logique sur une solution technologique particulière (EAI, ESB, etc.), afin de faciliter la communication entre SI.

Néanmoins, au-delà de cette affirmation d'intention, il est particulièrement important d'avoir conscience de la réalité d'une approche MDA dans un contexte de collaboration inter-entreprises : la difficulté majeure ne réside pas dans les différents niveaux eux-mêmes (métier, logique, technique), mais plutôt dans les transitions entre ces niveaux. Nous nous posons donc les questions suivantes : comment définir les méta-modèles *métier*, *logique* et *technique* ? Quelles correspondances pouvons-nous identifier entre les différents niveaux de modélisation ? Qu'en est-il des transformations de modèles nécessaires pour la descente en abstraction entre niveaux ?

Organisation du document

Ce mémoire de thèse est organisé comme suit :

- **le chapitre I** présente la problématique de nos travaux et notre positionnement. Nous exposons, dans un premier temps, notre vision de la collaboration inter-entreprises (réseaux, structures, niveaux de collaboration, place de l'interopérabilité, place des SI). Dans un deuxième temps, nous proposons une caractérisation de l'interopérabilité des systèmes d'information en tant que solution à l'intégration des entreprises. Enfin, ce chapitre se termine par une exploration des approches de solutions pour l'interopérabilité (et de leur volet « conception »).

- **Le chapitre II** présente une exploration organisée de l'espace de nos travaux de thèse. Il comporte exactement trois parties. La première partie traite du concept de système d'information, cœur de nos travaux. Nous discutons des différentes architectures de système d'information en nous intéressant particulièrement à l'architecture orientée services (SOA). La deuxième partie présente un panorama des solutions, formalismes, standards et approches qui facilitent l'établissement de l'interopérabilité sur les niveaux que nous avons déjà identifiés dans le chapitre I. Enfin, la troisième partie s'intéresse à la conception dirigée par les modèles. Nous présentons l'approche MDA en détail, ainsi que des travaux de conception de solutions collaboratives basés sur cette approche.

- **Le chapitre III** présente notre proposition d'une solution logique permettant d'assurer une médiation entre des systèmes d'information hétérogènes, afin d'en assurer (et d'en supporter) l'interopérabilité.

- **Le chapitre IV** répond à la problématique de conception de la solution de médiation à travers la proposition d'une approche basée sur l'exploitation d'un modèle de processus collaboratif, pour générer un modèle logique de système médiateur dédié à une collaboration particulière. Nous nous intéressons plus précisément dans ce chapitre à la définition des règles de transformation entre les deux niveaux *métier* et *logique*.

- **Le chapitre V** porte sur l'implémentation et l'expérimentation de notre approche. Nous avons développé un prototype d'atelier logiciel permettant de modéliser les processus collaboratifs, de les transformer en modèles de système d'information et enfin de visualiser et d'enrichir les modèles ainsi obtenus. Nous nous intéressons particulièrement dans ce chapitre à l'implémentation des règles de transformation en utilisant le langage ATL (*Atlas Transformation Language*).

- Enfin, **le chapitre VI** clôture cette thèse en donnant les conclusions et les perspectives de ce travail. Nous proposons un bilan du travail effectué durant cette thèse et un ensemble de perspectives, liées notamment à divers travaux actuels.

Chapitre I

Problématique et positionnement

1. Introduction du chapitre

Ce chapitre montre le contexte et la problématique de nos travaux de thèse. Nous nous intéressons tout d'abord à traiter et à définir la notion de collaboration entre entreprises : comment les entreprises établissent-elles des collaborations ? Sur quels critères ? quelles sont les formes de collaboration ? Comment peut-on caractériser une collaboration ? Existe-t-il des niveaux de collaboration ?

Par la suite, nous présentons les grandes questions liées à nos travaux. Elles font directement référence à une vision « supérieure » et « idéale » de la notion de collaboration : l'interopérabilité. Après l'avoir définie, nous nous intéressons aux niveaux qui permettent de caractériser l'interopérabilité des entreprises dans le but de réduire le champ de notre travail à celle des systèmes d'information. Après avoir évoqué les approches qui nous permettent de traiter le problème de l'interopérabilité des systèmes d'information, nous présentons un concept de « médiateur », qui assure cette interopérabilité. Enfin, Nous nous intéressons également à la spécification d'un tel concept.

2. L'établissement d'une collaboration : d'une structure figée à une structure plus dynamique

La collaboration d'entreprises est un concept vaste qui peut s'appliquer à bon nombre de cas de figure : groupes de partenaires aux compétences complémentaires (augmentation de la couverture), groupements d'entreprises positionnées sur le même segment (augmentation de la puissance), rassemblements de fournisseurs d'un même donneur d'ordre (optimisation et sécurisation), plate-forme d'achats groupés (pouvoir de négociation) et toutes autres sortes de réseaux inédits, institués ou occasionnels.

La question de cette collaboration d'entreprises est nécessairement impactée par la nature de l'écosystème économique : cet environnement industriel tend vers une fluidification de sa structure, porteuse à la fois de nouvelles opportunités inhérentes à cette nouvelle souplesse, mais également de nouvelles contraintes d'agilité pour les entreprises, plus critiques et plus exigeantes. Nous pouvons alors formuler le constat suivant :

d'un paysage figé et fractionné au sein duquel il fallait s'insérer et s'implanter, le tissu industriel et économique évolue vers un environnement fluide, mouvant et chargé, dans le flot et à la surface duquel il faut naviguer (Fig I.1).

Nous pouvons illustrer cette évolution par un adage simpliste et parodique : « **plus loin, plus vite, plus fort** ». Plus concrètement, cette tendance évolutive de l'écosystème industriel se caractérise selon les trois plans :

- **rayonnement (plus loin) :** les avancées technologiques (internet, etc.) ont favorisé l'accessibilité aux informations des clients, partenaires ou fournisseurs qui peuvent être éloignés géographiquement et de la même façon à celles des concurrents.
- **Réactivité (plus vite) :** s'il est courant d'établir une relation industrielle (quelle qu'elle soit) durable et inscrite dans le fonctionnement même des

partenaires concernés, il est également pertinent de construire ponctuellement des relations opportunistes.

- **Intensité (plus fort) :** les avancées, aussi bien dans le domaine du génie industriel que dans le domaine du génie informatique, amènent nécessairement les collaborations à être plus efficaces dans leur définition et dans leur déroulement.

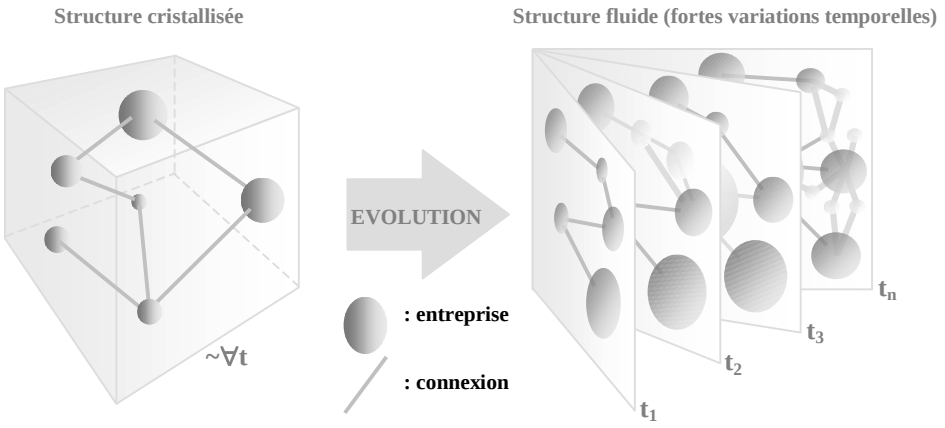


Fig. I. 1. D'une structure cristallisée à une structure fluide

La qualité des collaborations, hier principalement basée sur la solidité des connexions établies (et figées), est de plus en plus assise sur la capacité des entreprises à évoluer et à se connecter efficacement et rapidement les unes aux autres.

Ce changement s'accompagne également d'une variation dans la prise en compte des horizons stratégiques : d'une dynamique à horizons plutôt lointains, caractéristique d'un milieu stable, la structure « magmatique », vers laquelle nous pensons que s'oriente le tissu industriel, se caractérise par des horizons plus rapprochés, significatifs d'une relative instabilité. La problématique de la collaboration ajoute donc aux questions de stabilisation, de consolidation et d'optimisation des relations, celles concernant l'identification des partenaires ou la construction effective des connexions.
Nous pouvons schématiser cette transformation en parlant de *consumérisme industriel* : les entreprises ont moins le temps de construire leurs partenariats, elles recherchent des collaborations optimales au sens de la valeur ajoutée, la confiance entre les partenaires ne vient que dans un deuxième temps. Cette tendance de réactivité du milieu industriel, cette recherche de l'agilité montrent que la pérennité des entreprises dépend aujourd'hui de leur aptitude à intégrer efficacement des écosystèmes divers et variés.

3. Du concept de collaboration à l'interopérabilité

Si la partie précédente nous a principalement permis de mettre en évidence l'importance grandissante et le rôle changeant de la notion de collaboration vis-à-vis des contextes socio-économiques actuel et futur, il nous semble important de consacrer cette partie à l'étude et à la mise en relation, d'une part, des « *niveaux de*

collaboration » (*i.e.* la façon dont les entreprises peuvent envisager leurs collaborations, en termes de nature de relation entre les partenaires, de topologie, d'objectif collectif, de fréquence ou encore « d'intimité » de la relation) et d'autre part des « *niveaux de compatibilité collaborative* » des organisations (que nous qualifierons de « *niveaux de maturité collaborative* »).

3.1. Discussion sur les niveaux de collaboration

La notion de collaboration peut être abordée selon un grand nombre de points de vue [**Villareal 05**] :

- l'aspect *temporel* de la collaboration : est-elle opportuniste, occasionnelle, périodique, systématique, permanente, etc. ?
- L'aspect *relationnel* de la collaboration : concerne-t-elle des concurrents, des organisations complémentaires, un donneur d'ordres et ses fournisseurs, une combinaison de ces relations ?
- L'aspect *topologique* de la collaboration : est-elle structurée en étoile, en point à point, selon une chaîne linéaire, une combinaison de ces structures ?
- L'aspect *intimité* de la collaboration : quelle est la criticité ou la puissance des informations ou fonctions partagées par les partenaires ?
- Etc.

Nous allons nous appuyer sur certains résultats issus du domaine des systèmes multi-agents [**Bouzguenda 06**] pour en présenter une vision synthétique sous la forme d'une caractérisation des niveaux de collaboration. Nous proposons donc le nivellement du domaine de la collaboration suivant (où chaque niveau englobe le niveau précédent) :

1. **communication :** *échange de données*. Les entreprises et organisations communiquent, échangent et partagent des informations afin d'optimiser leur fonctionnement individuel. Ce niveau de collaboration consiste en un échange simple des données des entreprises.
2. **Coordination :** *partage et synchronisation de tâches*. Les entreprises et organisations réalisent des tâches dont dépendent leurs partenaires et réciproquement. Cette coordination a pour vocation l'amélioration de leurs performances individuelles. Ce niveau de collaboration, s'il semble faire apparaître un embryon de processus collaboratif (tâches partagées et synchronisées) correspond en fait à un stade individuel : l'absence d'objectif commun traduit l'impossibilité de concevoir un processus collaboratif en tant que tel. Ce sont uniquement leurs compétences (applications, fonctions, services) que les entreprises partagent ou mettent à disposition.
3. **Coopération :** *poursuite d'un objectif commun*. Les différents partenaires poursuivent un objectif collectif qui est la raison d'être du réseau d'entreprises. La poursuite de cet objectif nécessite la mise en place d'un processus collectif que les partenaires doivent définir et au sein duquel ils doivent s'intégrer.
4. **Intégration :** *appartenance transparente à une même entité*. A ce niveau de collaboration, l'échange de données, le partage des tâches et la poursuite d'un objectif commun sont inhérents au fait que les entreprises et organisations se trouvent intégrées au sein d'une même entité (virtuelle ou concrète).

3.2. Discussion sur les niveaux de maturité collaborative

On trouve dans [**Kosanke 05**] une synthèse du standard IEC TC 65/290/DC (cf. [**IEC 05**]) sur l'évaluation de la compatibilité des entreprises. En confrontant ces résultats aux niveaux de collaboration présentés précédemment, on peut en extraire les niveaux de maturité collaborative suivants :

1. **communicante :** capacité à échanger et à partager des informations. Ce niveau correspond à l'aptitude initiale des entreprises à disposer d'interfaces de communication élémentaires autorisant une communication asynchrone.
2. **Ouverte :** aptitude au partage des fonctionnalités et des services. Ce niveau correspond à la capacité des entreprises à ouvrir leurs compétences vers l'extérieur et leurs partenaires potentiels, mais également à pouvoir accéder aux fonctionnalités externes qui seraient mises à leur disposition. Les échanges doivent alors pouvoir être synchrones [**Pingaud 03**].
3. **Fédérée :** capacité à travailler selon un comportement collectif. Ce stade nécessite de l'organisation qu'elle maîtrise ses propres processus afin de pouvoir s'intégrer au processus collaboratif (on trouve ici une connexion naturelle vers les certifications de type *ISO* ou *Capacity Maturity Model*).
4. **Interopérable :** aptitude à s'immerger au sein d'une entité plus vaste et à en devenir un composant actif susceptible de participer au comportement (la circulation des données, la gestion des applications et l'exécution des processus) de l'abstraction dans laquelle il s'est fondu. L'interopérabilité peut donc être vue comme un moyen (en tant que capacité) d'atteindre l'intégration.

Ces niveaux de maturité collaborative nécessitent en outre la mise en place de certaines solutions (logiques ou techniques). Le premier niveau (Communicante) est naturellement outillé par les moyens de communication actuels (média, internet, EDI (*Electronic Data Interchange*), etc.). Les deuxième et troisième niveaux (Ouverte et Fédérée) peuvent être assurés par différentes approches telles que l' EAI (*Enterprise Application Integration*) ou encore la vision SOA (*Service-Oriented Architecture*). Le quatrième niveau (Interopérable) nécessite que les partenaires soient en mesure d'interagir les uns avec les autres de manière transparente. Il est important de noter que l'accession à cette faculté peut être caractérisée par le niveau d'effort qui aura été demandé aux entreprises pour l'atteindre. En effet, l'interopérabilité peut être atteinte par refonte profonde des entreprises (standardisation vis-à-vis d'un certain nombre de normes garantissant leur capacité d'intégration), mais cette vision « totalitaire » n'est pas la seule et nous proposerons dans la suite de ce manuscrit d'essayer d'atteindre l'interopérabilité des entreprises en limitant l'effort qui leur est demandé. Le domaine de la collaboration peut finalement être caractérisé selon les trois dimensions suivantes (profondément liées entre elles) : *niveaux de collaboration*, *maturité collaborative* et *technologies* comme présenté sur la figure I.2.

D'après [**Konstantas** *et al.* **05**], l'interopérabilité peut être définie comme *l'aptitude de systèmes à pouvoir travailler ensemble sans effort particulier pour les utilisateurs de ces systèmes*. Cette vision, issue du réseau INTEROP (réseau d'excellence européen travaillant sur l'interopérabilité des entreprises et de leurs applications) met en évidence un point crucial sur lequel nous allons nous appuyer dans la suite de ces travaux : l'interopérabilité se traduit par la capacité des

entreprises – hors intervenants humains – à supporter, de manière transparente pour les utilisateurs, les contraintes et conséquences des besoins d'intégration.

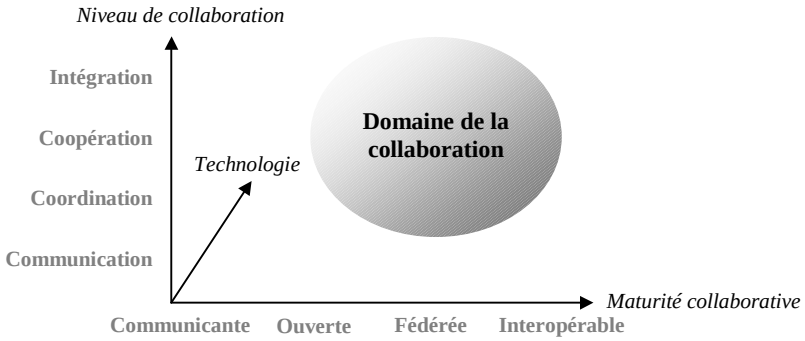


Fig. I. 2. *Un référentiel de caractérisation du domaine de la collaboration*

3.3. Un exemple illustrant la notion d'interopérabilité

Il s'agit dans cet exemple d'étudier le déroulement de *l'assemblée générale des Nations Unies*[1]. Chaque pays membre de l'ONU participe à cette assemblée par l'intermédiaire d'une délégation qui le représente (Fig I.3). L'intérêt de l'étude de cet exemple est qu'il dégage des analogies avec un système composé d'un ensemble hétérogène d'entreprises souhaitant collaborer. Nous détaillons par la suite les spécificités de cet exemple, ainsi que les analogies avec le domaine industriel de nos travaux.

Fig. I. 3. *Une réunion de l'assemblée générale des Nations Unies*

L'objectif principal (de notre point de vue) d'une telle assemblée générale des Nations Unies est de permettre à toutes les délégations de s'exprimer aisément dans le cadre d'un échange fructueux de discours et de points de vue. Nous souhaitons utiliser cet exemple pour illustrer les formes d'exigences (issues des délégations) inhérentes à ce type de collaboration :

- **une organisation des interventions :** le déroulement de l'assemblée générale doit être « orchestré » (au sens de la gestion des interventions des délégations en termes d'ordonnancement et de respect des temps de parole).
- **Un respect de la langue et de la culture de chaque délégation :** pour des raisons liées à l'identité linguistique de chaque pays, chaque délégation peut

[1] L'Organisation des Nations unies (ONU ou encore Nations Unies) est une organisation internationale fondée le 26 juin 1945 à San Francisco pour résoudre les problèmes internationaux. L'assemblée générale est le principal organe de délibération.

s'exprimer librement dans sa langue officielle[2], utiliser également les gestes qu'elle trouve convenables et s'habiller conformément à sa tradition et à sa culture.

- **Une interprétation unique des discours sans distinction de langue :** les délégations doivent pouvoir interpréter correctement (dans leur langue) le sens du discours de tout autre délégation (exprimé dans une langue différente). La capacité de l'assemblée à fournir cette interprétation est un enjeu majeur de la cohérence sémantique de l'assemblée.
- **Une sécurité et une accréditation indispensable :** la sécurisation du lieu de l'assemblée générale est indispensable : seules les personnes accréditées peuvent accéder et participer à l'assemblée générale.
- **Une possibilité d'utilisation de matériel technologique privé :** même si ce n'est pas réellement le cas, nous élargissons le domaine des contraintes identifiées en précisant que chaque délégation doit pouvoir utiliser son matériel propre (microphone, casque, etc.) et l'utiliser pour interagir avec les autres délégations. On rejoint ainsi la notion de moindre effort pour les utilisateurs du système « assemblée générale de l'ONU ».

L'interopérabilité dans cet exemple réside dans la capacité des délégations à échanger facilement leurs points de vue à travers les discours qu'elles prononcent. En effet, [**IEEE 90**] considère l'interopérabilité comme « *la capacité que possèdent deux ou plusieurs systèmes ou composants à échanger des informations puis à exploiter les informations venant d'être échangées* ».

Cette « interopérabilité de délégation » peut être déjà caractérisée selon trois niveaux : un niveau organisationnel où il faut synchroniser les interventions des délégations, un niveau sémantique où il faut traduire un discours (tout en gardant son sens) vers une autre langue et enfin un niveau technique où les délégations utilisent une infrastructure technologique (casques, micro-phones, etc.) pour communiquer.

Nous pouvons également identifier certaines exigences « complémentaires » de cette interopérabilité (sécurité des lieux de l'assemblée), car elles n'interviennent pas directement dans l'échange des discours mais permettent d'assurer une certaine qualité.

4. De la caractérisation de l'intéropérabilité des systèmes d'information

La question de l'interopérabilité des entreprises impacte nécessairement le domaine des Systèmes d'Information (SI). Dans [**Monateri** *et al.* **99**], les auteurs reconnaissent que l'interopérabilité des entreprises passe par l'interopérabilité de leurs systèmes d'information afin d'améliorer l'efficacité globale d'un réseau collaboratif. En effet, le comportement, la réactivité et la dynamique de l'entreprise reposent en grande partie sur le système d'information au travers des processus qu'il supporte, des applications qu'il propose et des données qu'il gère.

[2] Il existe en réalité six langues officielles : l'anglais, l'arabe, le chinois (mandarin), l'espagnol, le français et le russe.

L'étude de l'interopérabilité des systèmes d'information inclut, en partie, l'étude du concept même de système d'information en tant que support de cette interopérabilité. Dans cette partie de manuscrit, nous allons introduire le concept de système d'information (Section 4.1), en présentant brièvement la multitude de définitions de SI (Section 4.1.1), en parlant de l'hétérogénéité qui caractérise une mise en commun de différents SI (Section 4.1.2) et enfin en évoquant la collaboration des SI (Section 4.1.2).

Nous discutons de la caractérisation de l'interopérabilité des entreprises (Section 4.2) en présentant un ensemble de cadres limitant les niveaux à prendre en considération vis-à-vis du problème de l'interopérabilité. Enfin, nous tentons de définir des niveaux d'interopérabilité des systèmes d'information (Section 4.3).

4.1. Introduction au concept de système d'information

Définir un système d'information n'est pas une tâche facile du fait de la variété des définitions cohabitantes. Chaque définition est étroitement liée à un point de vue particulier. Nous pouvons définir un système d'information d'un point de vue systémique, vis-à-vis de son objectif dans l'entreprise, de sa conception etc.

4.1.1. Différentes définitions d'un système d'information

Une définition d'un SI d'un point de vue *systémique* essaie de le situer par rapport à d'autres systèmes existants (système opérationnel et système de pilotage). Un SI est chargé, en effet, de stocker et de traiter les informations relatives au « système opérant » afin de les mettre à disposition du « système de pilotage » [**Reix 02**]. Une définition du point de vue de son *objectif* dans l'entreprise atteste qu'un SI est un système qui doit garantir que la bonne information est disponible au bon endroit et au bon moment [**Bernus** *et al*. **96**]. Une autre définition d'un point de vue de sa *conception* considère un système d'information comme **un ensemble de données, d'applications et de processus interagissants** [**Morley 02**].

Dans cette panoplie de définitions, une question légitime est alors : « quel point de vue de SI devons nous considérer pour traiter le problème de l'interopérabilité des systèmes d'information ? ». S'il n'existe pas de réponse directe, du fait de la globalité du problème, nous souhaitons, dans ce manuscrit considérer la forte relation que le système d'information a avec son entreprise. Le domaine des travaux de notre thèse se positionne à l'intersection du génie industriel et du génie logiciel. En effet, le système d'information, dont la conception fait référence au génie logiciel, est strictement lié aux exigences et aux spécificités de l'entreprise, qui font référence au génie industriel.

4.1.2. Hétérogénéité des systèmes d'information

Il est évident que la principale raison qui motive la recherche dans le domaine de l'interopérabilité des systèmes d'information est l'hétérogénéité qui caractérise ces systèmes. En partant de la définition « structurelle » présentée ci-dessus (puisque nos travaux portent essentiellement sur la projection informatique des Si) : *un ensemble de données, d'applications et de processus interagissants*, nous pouvons déjà décomposer l'hétérogénéité des systèmes d'information en : hétérogénéité des données, hétérogénéité des applications et hétérogénéité des processus.

4.1.2.1. Hétérogénéité des données

Les données des systèmes d'information sont très « propriétaires » : elles possèdent des formats et structures non compatibles du point de vue syntaxique et appartiennent à des domaines d'entreprises différents, ce qui crée un autre problème d'intégration du point de vue sémantique. Par exemple, dans un schéma de données *A*, l'élément « id_client » désigne l'identifiant unique d'un client. Dans un autre schéma de données *B*, l'élément « code_client » désigne la même chose. Dans ce cas, il existe une correspondance sémantique entre les deux éléments avec une différence d'ordre syntaxique.

4.1.2.2. Hétérogénéité des applications

Les applications des entreprises ne sont pas faites nativement pour travailler ensemble. Ces applications peuvent être déployées dans des systèmes d'exploitation incompatibles et développées dans des plate-formes d'exécution propriétaires (.NET, Java, etc.). De plus, certaines entreprises utilisent encore des applications propriétaires et difficilement accessibles (plus connues sous le terme anglais : *legacy applications*), ce qui ne facilite pas la collaboration à ce niveau. Par exemple, le cas de deux entreprises qui possèdent deux Progiciels de Gestion Intégré (PGI) achetés auprès d'éditeurs qui travaillent avec des technologies différentes.

4.1.2.3. Hétérogénéité des processus

Les entreprises possèdent des processus internes qui sont spécifiques à leurs stratégies et modes de fonctionnement. Ces processus sont modélisés avec des formalismes différents adoptés par les entreprises (réseaux de pétri, BPMN, diagramme d'activité d'UML, etc.). De plus, dans un contexte collaboratif, le processus de collaboration doit être aligné sur les processus internes des partenaires. Cela nécessite une synchronisation du processus collaboratif global avec les processus internes des entreprises.

4.1.3. Système d'information dans un contexte de collaboration

Dans les relations humaines, et afin de préserver leurs intérêts, les individus veillent à ne présenter qu'une visibilité restreinte de leur vie privée. Dans un contexte de collaboration de systèmes d'information, les entreprises doivent gérer le degré d'ouverture de leurs SI vis-à-vis des autres partenaires. Dans [**Vanderhaeghen** *et al.* **04**] les auteurs expliquent que pour des raisons stratégiques, la plupart des entreprises d'un réseau collaboratif veillent à ne pas dévoiler leurs savoir-faire et cachent la connaissance liée à leurs processus métiers internes.

[**Bauer** *et al.* **06**] et [**Vanderhaeghen** *et al.* **06**] considèrent que c'est une partie bien définie d'un système d'information : **la partie publique**, qui présente une interface accessible d'un SI pour les autres partenaires. Une **partie privée** reste cependant protégée et dans quelques cas, accessible par des partenaires autorisés. La relation « privé / public » a certainement un impact sur la conception des systèmes d'information dans un contexte d'interopérabilité. Nous avançons que toute solution d'interopérabilité de systèmes d'information doit la prendre en considération.

4.2. Caractérisation de l'interopérabilité des entreprises

Après avoir défini le concept de l'interopérabilité d'entreprise (Section 3), la question est ici d'identifier les niveaux qui la définissent. Les entreprises se caractérisent par leurs différences d'ordre culturel, linguistique, métier et même technologique. Des travaux de recherche variés se sont intéressés à la définition de cadres caractérisant des niveaux de considération de l'interopérabilité dans les entreprises. Nous présentons trois cadres de référence.

4.2.1. Le cadre IDEAS

Le cadre IDEAS *(Interoperability Development for Enterprise Applications and Software, IST-2001-37368)* [**IDEAS 03**] est présenté comme une approche structurée pour la collecte et l'identification des visions et des challenges de recherche sur l'interopérabilité des applications d'entreprise. L'interopérabilité est considérée sur trois niveaux horizontaux et une dimension transversale (Fig I.4) :

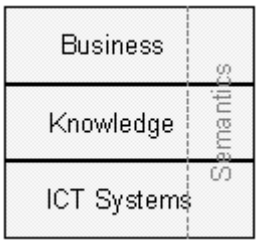


*Fig. I. 4. Représentation simplifiée de cadre IDEAS [**IDEAS 04**]*

- un niveau **métier (*business*)** : il définit le contexte métier et les processus collaboratifs des entreprises.
- Un niveau **connaissances (*knowledge*)** : il concerne l'organisation, les qualifications et les sources de connaissances dans l'entreprise.
- Un niveau **technologies de communication (*ICT systems*)** : il concerne les applications, les données et les infrastructures de communication.
- Une dimension **sémantique** (***Semantics***) : elle assure une compréhension mutuelle à tous les niveaux identifiés précédemment.

4.2.2. Le cadre AIF

Le cadre AIF (*Athena Interoperability Framework)* [**Athena 05**], [**Berre** *et al.* **07**] adopte une approche holistique de l'interopérabilité permettant de mieux comprendre et d'analyser les exigences de l'interopérabilité. Ce cadre définit trois niveaux :

- un niveau **conceptuel** (***conceptual***) : il définit les concepts, méta-modèles et langages utiles pour la modélisation de certains aspects de l'interopérabilité.
- Un niveau **applicatif (*applicative*)** : il concerne les méthodologies, les standards et les « patterns » qui interviennent dans le développement de solutions interopérables.
- Un niveau **technique** (***technical***) : il décrit les plate-formes d'interconnexion de systèmes.

4.2.3. Le cadre EIF

Le cadre EIF (*European Interoperability Framework*) [**EIF 04**] vise à structurer un ensemble de recommandations et de directives pour l'interopérabilité entre les administrations européennes d'une part et entre les administrations et les citoyens d'autre part. L'intérêt est qu'il présente trois niveaux (appelés dimensions dans ce cadre) pour le traitement de l'interopérabilité.

- Un niveau **organisationnel (*organisational*)** : il concerne les objectifs métiers, la modélisation des interactions entre partenaires (modèles de processus) et la description de leur entreprise.
- Un niveau **sémantique** (***semantic***) : il concerne le sens des informations et des connaissances des entreprises.
- Un niveau **technique** (***technical***) : il concerne les solutions technologiques et les outils d'interconnexion de systèmes.

4.2.4. Analyse

Les cadres cités regroupent des approches de considération de problème d'interopérabilité dans les domaines industriel (IDEAS et AIF) et administratif (EIF). D'autres cadres sont plus spécifiques au domaine métier (E-commerce[3], industrie des chaussures[4], etc.).

Nous pouvons remarquer une forte similitude entre les niveaux cités par les trois cadres : tout d'abord, le niveau organisationnel d'EIF et le niveau métier d'IDEAS qui définissent l'échange d'objectifs, de stratégies, de responsabilités et de modes de fonctionnement des entreprises. Cet échange est basé sur l'utilisation de modèles et de langages compréhensibles par les entreprises (définis dans le niveau conceptuel d'AIF). En conclusion, si EIF et IDEAS définissent en quoi les entreprises peuvent échanger au niveau organisationnel / métier, le niveau conceptuel de AIF décrit les moyens qui permettent cet échange.

Le niveau sémantique d'EIF et la dimension de même nom d'IDEAS concernent la signification et l'interprétation des informations et des connaissances des entreprises. Le cadre IDEAS donne cependant une valeur plus importante à la sémantique en la positionnant transversalement par rapport aux autres niveaux.

Enfin, les trois cadres tombent d'accord sur un niveau technique de l'interopérabilité des entreprises. [**Chituc** *et al.* **05**] expliquent que l'interopérabilité repose en grande partie sur les **infrastructures technologiques des entreprises**. Les avancées en la matière sont importantes. Le langage XML (*eXtended Markup Language*) a facilité l'intégration des données. Les technologies de type EAI et ESB (*Enterprise Service Bus*) ont facilité l'intégration des applications quelles que soient leurs spécificités techniques. Cependant, Il nous semble démesuré de considérer que l'interopérabilité repose sur les seules infrastructures techniques. D'après [**Berre** *et al.* **07**] « *Interoperability should not only be considered a property of ICT systems, but also concerns the business processes and the business context of an enterprise* »[5].

[3] Le cadre E-Commerce Integration Mata-Framework (ECIMF) http://www.ecimf.org/.

[4] Le cadre Euro Shoe http://www.euro-shoe.net/.

[5] L'intéropérabilité ne doit pas être considérée uniquement sur un niveau technique, car elle concerne aussi les processus métiers et le contexte métier d'une entreprise.

Nous pouvons remarquer une convergence des cadres cités à prendre en compte d'une manière plus ou moins poussée les trois niveaux : organisationnel, sémantique et technique d'EIF. En effet, les niveaux métier et connaissances de IDEAS correspondent à notre avis au niveau métier d'EIF. Le niveau sémantique d'EIF reprend le problème de la sémantique cité dans IDEAS. Le niveau technique est présent dans les trois cadres. Dans la suite, nous considèrerons donc l'interopérabilité selon les trois points de vue suivants (faisant la synthèse des dimensions rencontrées) : *métier*, *sémantique* et *technique*.

4.3. Une tentative de caractérisation de l'intéropérabilité des systèmes d'information

Le système d'information joue un rôle capital pour permettre à une entreprise d'être interopérable. Assurer l'interopérabilité des systèmes d'information passe d'abord par une évaluation de la capacité d'un SI à être interopérable. Dans cette section, nous tentons d'identifier les différents niveaux qui caractérisent l'interopérabilité des systèmes d'information. Nous proposons d'établir certaines liaisons possibles (Fig I.5) entre les différents niveaux constatés pour l'interopérabilité des entreprises (Section 4.2.4) et les dimensions qui caractérisent l'hétérogénéité structurelle des systèmes d'information (Section 4.1.2).

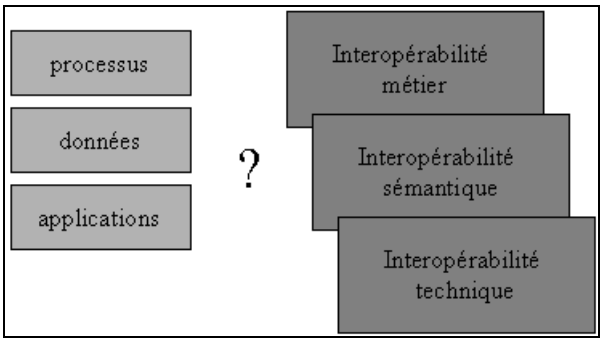


Fig. I. 5. *Quelles liaisons entre les dimensions des SI et les niveaux d'interopérabilité ?*

4.3.1. Niveau métier pour l'interopérabilité des systèmes d'information

En reprenant la définition de [**Panetto 06**], qui définit comme suit le niveau métier pour l'interopérabilité des entreprises : « *il regroupe les responsabilités, autorisations, confiances, aspects légaux, propriétés intellectuelles et structures organisationnelles nécessaires à l'acceptation des échanges d'information…* », nous pouvons constater que cette description semble globalement inappropriée au domaine des systèmes d'information. Nous devons filtrer cette définition et n'en garder que ce qui touche directement aux systèmes d'information.

Nous pouvons imaginer deux classes d'informations : la première classe concerne les stratégies et les décisions relatives aux entreprises et une deuxième classe qui contient les autres informations, portant sur les processus, données et applications de l'entreprise (à un niveau métier). L'interopérabilité des systèmes d'information (en particulier au sens de sa problématique informatique) se réfère alors à tout ce qui a trait à la seconde classe en y associant l'aide à la décision. A ce niveau, l'interopérabilité entre systèmes d'information peut être décomposée comme suit :

- **interopérabilité métier des processus** : deux SI qui veulent être interopérables doivent converger vers des modèles de processus collaboratifs, expliquant les échanges qui vont avoir lieu.
- **Interopérabilité métier des données** : il faut décrire les données à échanger entre les applications des SI. Par exemple, un schéma de données XSD (*XML Schema Definition*) est nécessaire pour pouvoir échanger efficacement entre deux applications des fichiers XML (qui correspondent à ce schéma).
- **Interopérabilité métier d'applications** : nous pouvons imaginer un échange de modèles qui décrivent les applications à mettre à disposition par toutes les entreprises souhaitant faire interopérer leur SI.

Enfin, pour schématiser ces considérations et la notion de classes d'information sur laquelle nous nous sommes appuyés, nous pouvons schématiser la différence entre l'interopérabilité des systèmes d'information et celle des entreprises (au niveau métier) en nous basant sur la définition systémique d'un SI et en gardant la partie « *système de traitement de l'information* » d'un SI [**Morley 02**] (Fig I.6).

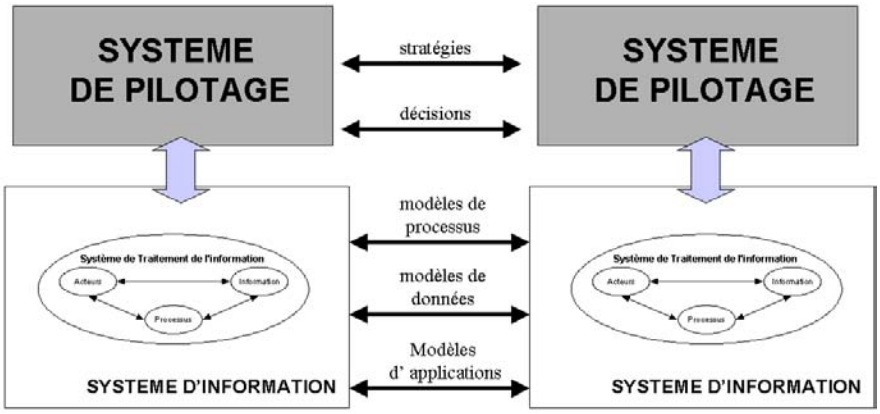


Fig. I. 6. Interopérabilité des systèmes d'information sur un niveau métier

4.3.2. Niveau sémantique pour l'interopérabilité des systèmes d'information

Dans le domaine des systèmes d'information, la sémantique réfère plus précisément au sens des différents éléments d'un système d'information. Le sens d'un élément peut dans ce cas différer d'un système à un autre. En effet, on doit s'assurer que les informations échangées entre systèmes d'information sont compréhensibles du point de vue de leur signification et de leur interprétation. F. Vernadat [**Vernadat 06**] considère l'interopérabilité sémantique comme « *la possibilité de partager, agréger ou synchroniser l'information dans des systèmes d'information hétérogènes...* ». C'est donc une capacité à partager les informations d'un système d'information (processus, données et application) avec un autre système d'information. Nous en déduisons que les entreprises doivent donc bien décrire et définir leurs SI afin de pouvoir partager leurs informations avec d'autres entreprises. Les conflits sémantiques d'après [**Izza 06**] « *ne concernent pas seulement les données, mais peuvent aussi concerner les autres couches de l'intégration dont notamment les fonctions et les processus…* ». A ce niveau, l'interopérabilité entre systèmes d'information peut être décomposée en :

- **interopérabilité sémantique des processus** : concerne la possibilité de faire interopérer des processus appartenant à des domaines métiers différents, qui possèdent une structuration différente (contrôle de flux, structures itératives, etc.) ou modélisés avec des formalismes de processus divers.
- **Interopérabilité sémantique des données** : concerne la mise en commun des données hétérogènes de systèmes d'information, qui sont décrites dans des référentiels différents.
- **Interopérabilité sémantique des applications** : concerne la mise en commun des fonctions proposées par les applications des entreprises. Il peut exister, de la même manière que pour les données, une difficulté pour l'interprétation de la fonction exacte qui peut être rendue par une application. Comme application de ce type d'interopérabilité, nous citons l'approche orientée services sémantiques [**Izza 06**].

Les conflits sémantiques peuvent faire référence à une homonymie (un même nom recouvre plusieurs concepts) ou à une synonymie (plusieurs noms recouvrent un même concept). La résolution des conflits sémantiques peut faire appel à des mécanismes de médiation afin de rapprocher les différentes interprétations sémantiques des éléments de SI. Cette médiation doit utiliser un référentiel pivot, qui identifie précisément le sens et la portée de tout le vocabulaire utilisé par les systèmes d'information souhaitant interopérer à ce niveau.

4.3.3. Niveau technique pour l'interopérabilité des systèmes d'information

Ce niveau concerne la capacité à mettre en communication les systèmes informatiques qui composent les systèmes d'information[6]. En effet, s'il existe une forte tendance à considérer que l'interopérabilité des systèmes d'information est principalement basée sur ce niveau, cela est essentiellement lié à la courante réduction du système d'information à sa projection informatique. L'interopérabilité technique des systèmes d'information concerne la différence entre les matériels, les infrastructures réseaux, les systèmes d'exploitation et les plate-formes des applications des SI. A ce niveau, l'interopérabilité entre systèmes d'information peut être décomposée en :

- **interopérabilité technique des processus** : concerne la possibilité de faire exécuter un processus collaboratif qui fait intervenir les applications, les données et les processus internes des SI. Les moteurs d'exécution de processus (BPEL, etc.) et les systèmes de gestion de workflow permettent d'assurer cette tâche.
- **Interopérabilité technique des données** : concerne les interfaces et les protocoles de communication nécessaires à l'échange des données entre SI. Nous citons les « middleware » comme solution générique permettant de remplir cette exigence.
- **Interopérabilité technique des applications** : des applications qui s'exécutent sous des systèmes d'exploitation différents (UNIX, Windows, etc.) ou développées dans des environnements incompatibles (.NET, JAVA,

[6] **[Morley 02]** considère qu'un système d'information est composé de deux sous-systèmes : un système de traitement de l'information et un système informatique.

etc.) doivent interopérer à ce niveau. Les Services-web ou l'EAI présentent des solutions pour ce point précis.

En traitant ce niveau, nous devons évoquer le cas des entreprises qui possèdent des applications propriétaires. Une application propriétaire est prisonnière de l'environnement technologique dans lequel elle a été développée [**Sneed 06**]. Cependant, on peut envisager de communiquer avec les applications qui sont basées en partie sur un SGBD (Système de gestion de base de données) par le biais des données échangées. Les applications qui proposent des traitements sont les plus délicates à faire communiquer.

L'interopérabilité technique est capitale pour l'interopérabilité des systèmes d'information. L'interopérabilité technique permet de faciliter la mise en œuvre des autres niveaux d'interopérabilités : métier et sémantique. Si nous ne pouvons pas démontrer cette assertion, nous donnons cependant l'exemple des portails et des plate-formes collaboratifs qui ont facilité l'interopérabilité du point de vue organisationnel (échange d'idées, de stratégies, etc.) en dépit des différences culturelles et linguistiques entre les entreprises.

4.3.4. Conclusion

Nous avons essayé, dans cette section, de caractériser l'interopérabilité des systèmes d'information en nous basant d'une part sur une caractérisation des niveaux d'interopérabilité des entreprises (*métier*, *sémantique* et *technique*) et d'autre part sur l'identification des dimensions liées à l'hétérogénéité dans les systèmes d'information (*processus*, *données* et *applications*). Nous avons conservé les niveaux *métier*, *sémantique* et *technique* identifiés pour les entreprises pour le cas de l'interopérabilité des SI, car le problème n'est pas seulement technologique, mais également un problème organisationnel et sémantique. L'interopérabilité des SI est finalement assez proche d'une définition de l'interopérabilité des composants électroniques donnée par l'IEC 62390, *« l'interopérabilité d'un système n'est autre qu'un niveau de compatibilité qu'il est possible d'obtenir en garantissant certaines caractéristiques régissant la qualité de la communication entre les composants de systèmes »* [**IEC 05**]. La qualité de la communication entre composants électroniques est assimilable à la qualité des échanges d'information entre SI.

Dans un contexte d'interopérabilité de systèmes d'information, l'information doit être échangée d'une manière **organisée** (niveau organisationnel), **compréhensible** (niveau sémantique) et **accessible** (niveau technique).

5. Exploration des solutions pour l'interopérabilité des sytèmes d'information

Dans cette section, notre objectif est d'évaluer les approches qui permettent de traiter l'interopérabilité des systèmes d'information aux niveaux que nous avons présenté (Section 4.3). Nous présentons deux approches différentes pour assurer une interopérabilité des SI. L'analyse des avantages et inconvénients des approches nous permet d'introduire la solution que nous proposons pour l'interopérabilité des SI. Nous parlons enfin des pistes pour sa conception et sa mise en œuvre.

5.1. Approches pour l'intéropérabilité des systèmes d'information

Assurer l'interopérabilité entre des systèmes d'information hétérogènes n'est pas un objectif facile. Une partie des problématiques de nos travaux de thèse concerne l'étude des concepts, outils et solutions qui facilitent l'interopérabilité des SI. L'analyse de la littérature sur l'interopérabilité des entreprises, issue des travaux du réseau d'excellence européen INTEROP [**Konstantas** *et al.* **05**], montre qu'il existe deux approches pour traiter le problème de l'interopérabilité des systèmes d'information :

- **une approche d'unification et de standardisation** : dans cette approche, les entreprises se mettent d'accord pour unifier la manière de présenter leurs données, processus et applications, mais aussi leurs outils, méthodes et stratégies pour faciliter l'interopérabilité de leurs SI. Comme exemple, l'EDI [**EDI 07**] illustre bien cette approche en unifiant les formats d'échange de données entre les systèmes d'information par domaine d'activité. Sur un autre niveau, le langage UEML [**Panetto 06**], [**Opdahl** *et al.* **06**] *(Unified Enterprise Modeling Language)* est un autre exemple de cette approche. UEML est un langage unifié de modélisation d'entreprise. Il offre en effet un moyen d'entente aux partenaires, qui présentent pourtant leur entreprise selon des formalisme de modélisation incompatibles et qui ne couvrent pas les mêmes vues (CIMOSA, GRAI, etc.). UEML est un langage pivot pour l'échange de modèles d'entreprises différents.

- **Une approche de « non standardisation »** : dans cette approche, les systèmes d'information collaborent sans se soucier de leur hétérogénéité et en gardant leurs formats de données, modèles de processus, applications et matériels existants. C'est donc une collaboration sans effort particulier de la part des partenaires. Cette approche nous semble plus réaliste, plus pragmatique et plus en phase avec les fondements idéologiques de l'interopérabilité.

L'approche d'unification et de standardisation est très coûteuse en termes d'investissement afin d'adopter les standards proposés : les entreprises ne peuvent pas intégrer si facilement ces standards avec leurs outils, formats et stratégies en place. Cependant, et à un certain niveau (basique), nous constatons une tendance de la part des entreprises pour s'approprier les standards de marché, ce qui augmente leur capacité à devenir interopérables. Dans cette approche, ce sont donc les systèmes d'information qui payent le coût de l'interopérabilité en s'adaptant. Dans l'exemple des Nations-Unies, présenté dans la section 3 de ce chapitre, la langue anglaise présente le standard qui peut être adopté par toutes les délégations. Cependant, cela nécessite un effort considérable de la part de toutes les délégations pour apprendre et maîtriser cette langue (jusque dans ses nuances et ses subtilités)

L'approche de « non standardisation » semble être plus attractive pour les entreprises, puisqu'elle offre plus de liberté. Les entreprises peuvent utiliser leurs SI durant une collaboration et s'intégrer efficacement dans le réseau collaboratif tout en gardant leurs habitudes de travail. Dans l'exemple des Nations-Unies présenté dans la section 3 de ce chapitre, cette approche est équivalente à la préservation des identités linguistiques des délégations en leur permettant de s'exprimer dans leur

langue maternelle. Pour que cela devienne réalisable, d'efficaces traductions d'une langue à une autre sont nécessaires.

En projetant cette « non standardisation » sur les composants d'un système d'information (*données, applications et processus interagissant*), cela ressemble à un réseau de systèmes d'information où chaque partenaire possède ses propres données, applications et processus et arrive à collaborer sans effort particulier avec les autres. Cela nécessite, en effet, que chaque partenaire connaisse les systèmes d'information (*données, applications et processus interagissant*) de tous les autres partenaires. Pratiquement, ceci apparaît évidemment comme très difficile à réaliser, surtout avec un nombre important de systèmes d'informations. Il faut autant de connexions que de systèmes partenaires. C'est une interopérabilité bi-latérale entre les SI (Fig I.7)

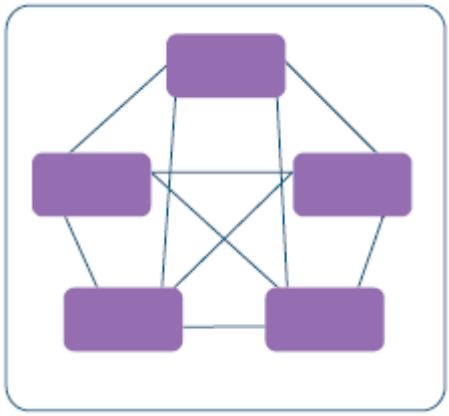


Fig. I. 7. *Solution bi-latérale pour l'interopérabilité des systèmes d'information*

L'autre solution est une solution multi-latérale (Fig I.8) où une autre partie, indépendante et tiers de confiance, doit gérer cette mise en compatibilité des systèmes d'information des partenaires. Une médiation entre les différents partenaires est nécessaire. Cette médiation doit être gérée par un système intermédiaire qui connaît tous les systèmes d'information du réseau collaboratif.

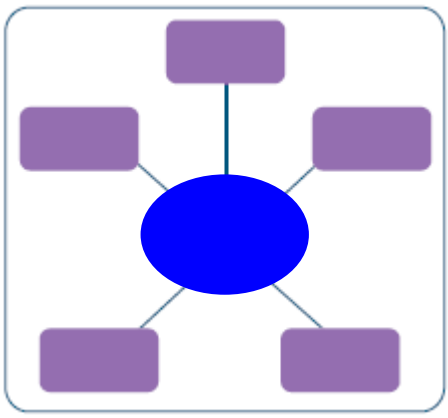


Fig. I. 8. *Solution multilatérale pour l'interopérabilité des systèmes d'information*

Dans le domaine juridique, la médiation est un terme qui fait référence à la résolution des conflits humains, en rapprochant des positions de personnes en conflit. Elle implique l'intervention d'un tiers neutre, impartial et indépendant, le

médiateur, lequel est un intermédiaire dans les relations. D'après J.-L. Lascoux[7] [**Lascoux 07**] la médiation « *propose un cadre, avec ses propres repères, constitués de règles de fonctionnement et de communication, et un processus avec des étapes. Elle commence par la reconnaissance de la position des parties, en termes de légitimité, jusqu'à la formalisation d'un accord le plus satisfaisant possible pour les parties* ».

Par rapport à la citation de J.-L. Lascoux, la source et la cause de médiation entre les systèmes d'information est leur hétérogénéité même qui rend compliquée leur interopérabilité. La médiation doit réussir à les faire interopérer en suivant « *un processus avec des étapes* » et des « *règles de fonctionnement* » bien définies. La « *reconnaissance des spécificités* » de chaque système d'information faisant partie du réseau est indispensable. C'est sur la base de ces spécificités que le médiateur de SI va proposer des solutions satisfaisantes.

5.2. Un médiateur pour l'interopérabilité des systèmes d'information

Dans ces travaux de thèse, nous nous orientons vers une approche de médiation pour assurer l'interopérabilité entre des SI hétérogènes. Dans ce cadre, le concept de **médiateur** ou d'un **système intermédiaire,** qui supporte et gère l'hétérogénéité des systèmes d'information de partenaires sur les niveaux : organisationnel, sémantique, syntaxique et technique, semble être prometteur. Le médiateur a pour fonction de permettre aux SI des partenaires d'un réseau collaboratif de collaborer sans se soucier de l'hétérogénéité de leurs SI. Ces partenaires ne doivent pas avoir à faire d'effort particulier.

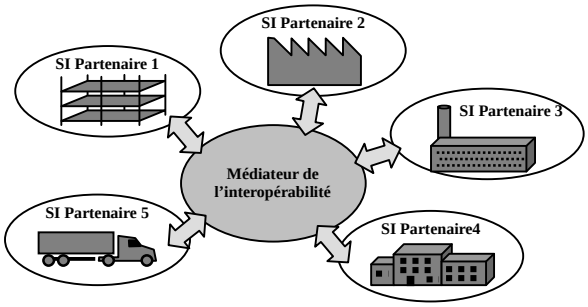


Fig. I. 9. *Un système médiateur pour supporter l'interopérabilité des systèmes d'information*

Quelques travaux se sont intéressés au concept de médiation entre systèmes d'information : [**Aubert** *et al.* **02**] définit un système d'information inter-organisationnel comme « *un système qui a pour fonction particulière de supporter des processus qui traversent les frontières d'une organisation* ». Dans [**Bauer** *et al*. **05**], on explique qu'une entité centrale pourra bien gérer un processus collaboratif en organisant l'intervention des partenaires. L'architecture de réseau dans ce cas est une architecture en étoile. L'exécution de processus collaboratif est gérée par un partenaire (partenaire fort) vis-à-vis des autres partenaires (partenaires faibles).

[7] Secrétaire Général de l'Union nationale des Médiateurs.

Les travaux cités mettent en avant la dimension collaborative du problème de l'interopérabilité : c'est le **processus collaboratif** qui définit cette dimension. Une définition plus spécifique du médiateur peut être déclinée par rapport à chacun des niveaux liés à l'interopérabilité des système d'information. Le médiateur est un outil de traduction et de mise en relation de données, applications et processus des partenaires. Il doit orchestrer le processus collaboratif tout en gérant les correspondances syntaxiques et sémantiques des données et applications des partenaires. C'est ainsi que le médiateur rentre dans l'architecture d'interopérabilité, qui est une architecture de système de systèmes (SoS pour *System of Systems*).

Nous montrerons l'intérêt de disposer d'un médiateur indépendant et intermédiaire (tiers de confiance), positionné au cœur du réseau des partenaires, maîtrisant les spécificités de chaque partenaire et les conventions associées au réseau. L'interaction avec ce médiateur est assurée par la partie publique de chaque système d'information (Section 4.1.3). Les parties publiques des SI forment avec le médiateur le support de l'interopérabilité des SI. Nous désignons cet ensemble sous la dénomination *Système d'Information Collaboratif* (SIC) ou *Système d'Information Médiateur* (SIM).

5.3. Conception de Système d'Information Collaboratif

A partir de cette vision du système d'information collaboratif en tant que pierre angulaire de l'interopérabilité des SI, il est légitime de poursuivre par une réflexion sur le contenu de ce système : comment peut-on le concevoir efficacement ? De quelle connaissance doit-on disposer pour pouvoir définir précisément ce SIC ? Et plus concrètement, quels éléments caractéristiques du réseau de partenaires doit-on rassembler afin de disposer du savoir nécessaire à la construction du SIC ?

Le développement d'un système d'information est certainement une tâche complexe. Historiquement, plusieurs paradigmes sont apparus à cet effet. Citons *le paradigme procédural* qui est basé sur une définition des procédures (traitements) du logiciel à développer pour l'entreprise, *le paradigme objet*, qui consiste à intégrer les traitements et données dans une seule entité appelée « objet » et *le paradigme composant* où un système d'information est défini par l'interaction d'un ensemble de composants. Un composant est une boîte noire accessible via une interface.

Plus récemment, l'OMG (*Object Management Group*) a proposé l'architecture dirigée par les modèles MDA (*Model Driven Architecture*) pour supporter le développement de systèmes distribués complexes. MDA est une nouvelle façon de spécifier les systèmes informatiques basée sur différents points de vue et leur mise en correspondance. C'est une approche d'ingénierie mettant le modèle au cœur du problème. Tout d'abord, les fonctionnalités et le comportement d'un système sont modélisés sans tenir compte des caractéristiques technologiques. Puis, cette description des fonctionnalités et des comportements est projetée sur une proposition de support technologique pour créer le système informatique conforme à une technologie spécifique.

La première phase est réalisée par l'intermédiaire de deux modèles : le premier (qui formalise le besoin) est indépendant de toute implémentation alors que le deuxième, construit à partir du premier, est indépendant de toute plate-forme d'exécution. La deuxième phase est réalisée par le biais d'un modèle décrivant une plate-forme

spécifique sur lequel on projette le modèle issu de la première phase. Dans la deuxième phase, les outils conformes à l'approche MDA doivent supporter la génération de code, les fichiers de déploiement et les fichiers de configuration. Enfin, les passages entre modèles constituent un aspect important de cette approche.

La question de l'utilisation de ce type d'approche d'ingénierie dans le cadre de notre problématique de conception de SIC nous semble légitime. La possibilité de disposer d'un modèle qui le décrit indépendamment des choix technologiques prendrait alors toute son importance. La génération de tels modèles doit prendre en compte les spécificités des SI partenaires et la manière avec laquelle ils collaborent et échangent des informations.

En raffinant ces considérations, nous pouvons prétendre que nous proposons, dans ce manuscrit, d'adopter cette approche d'ingénierie afin de permettre, en partant d'une spécification de la collaboration (processus collaboratif) de spécifier un modèle de SIC. Cette approche, qui est basée sur une architecture de modèles est bien placée pour maîtriser des couplages entre différents stades d'élaboration du SIC : du modèle de processus : CIM (*Computation Independent Model*) à un modèle de SIC : PIM (*Platform Independent Model*), puis à l'implémentation de la solution : PSM (*Platform Specific Model*). Ce sont des mécanismes de transformation de modèles qui réalisent les transitions entre les niveaux métiers, logiques et techniques dans le cycle de conception du SIC. Nous voulons nous intéresser dans ce manuscrit à la génération d'un modèle PIM de système d'information collaboratif. Ce modèle pourra par la suite être projeté sur plusieurs plate-formes technologiques (ESB, EAI, etc.).

6. Conclusion du chapitre

Dans ce chapitre, nous avons voulu atteindre deux objectifs principaux : le premier concerne la détermination de l'espace dans lequel ont évolué nos travaux. Le deuxième concerne une exposition des problématiques à traiter et une présentation des solutions que nous proposons.

Les entreprises collaborent pour améliorer leur productivité, pour s'ouvrir à de nouveaux marchés et pour faire face à l'agressivité des groupements déjà existants. Cependant, cette collaboration ne peut être réussie et efficace que s'il existe une réelle capacité d'ouverture et d'intégration de l'entreprise dans son écosystème. Tout ce qui relève d'une volonté d'intégration d'un système dans un système de systèmes se trouve naturellement facilité par l'émergence de l'interopérabilité. L'interopérabilité est dans ce cas une capacité qui facilite l'intégration d'une entreprise dans un réseau d'entreprises. Toutefois, la coexistence de partenaires de culture et d'habitudes de fonctionnement potentiellement différentes au sein d'un réseau hétérogène ne facilite pas la conception et l'usage de l'interopérabilité. C'est l'un des enjeux du domaine de l'interopérabilité de définir des cadres méthodologiques pour chercher des formes de consensus en fonction du besoin du réseau, tout en préservant les intérêts de chacun.

Un des constats que nous avançons dans nos travaux est que l'interopérabilité des entreprises passe forcément par l'interopérabilité de leurs systèmes d'information. Ces entreprises possèdent des systèmes d'information hétérogènes qui ne sont pas faits au départ pour collaborer. C'est à ce niveau que l'interopérabilité des systèmes d'information prend tout son sens. Nous avons tenté de définir les différents

niveaux d'interopérabilité qui permettent à des systèmes d'information d'être interopérables, en nous basant sur les travaux de caractérisation des niveaux d'interopérabilité des entreprises.

Nous avons introduit le concept de système d'information collaboratif. Ce système est basé sur un médiateur chargé de supporter les processus collaboratifs du réseau tout en garantissant l'interopérabilité de leurs applications et données. Ce système doit répondre aux exigences de l'interopérabilité des systèmes d'information que nous avons définies.

Enfin, nous ambitionnons dans ce travail de thèse d'évaluer l'apport de MDA afin de permettre à partir de la simple modélisation des processus collaboratifs, de générer la spécification d'un système d'information collaboratif qui permette de supporter l'interopérabilité entre systèmes d'information hétérogènes.

Chapitre II :

Cadre méthodologique pour l'intéropérabilité des systèmes d'information

Introduction du chapitre

Dans le chapitre précédent, nous avons introduit le concept de médiateur (Chapitre I, Section 5.2) dont le but est de supporter l'interopérabilité des systèmes d'information aux niveaux organisationnel, sémantique et technique. L'objectif de ce chapitre est de mener une exploration organisée de l'espace lié à cette problématique de l'interopérabilité des systèmes d'information. Nous définissons trois axes d'analyse bibliographique, qui forment un cadre méthodologique approprié pour notre étude.

1. Définition des axes d'analyse bibliographique

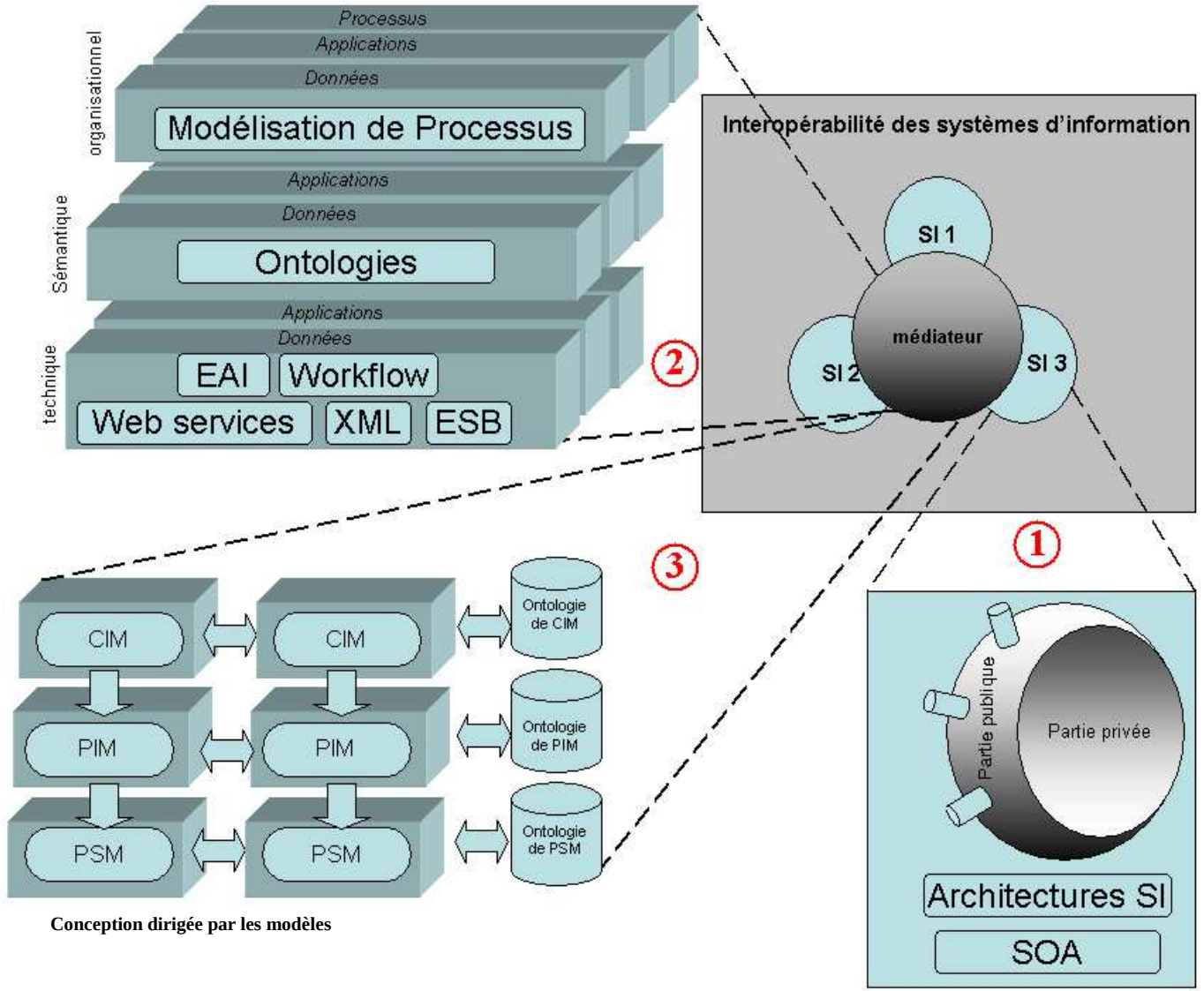
Le cadre classique pour l'étude de l'interopérabilité des entreprises est le cadre présenté par le projet INTEROP (http://interop-vlab.eu/) qui contient trois axes de recherche :

- **modélisation d'entreprise** : concerne les approches de modélisation d'entreprise et les architectures de référence permettant de caractériser l'interopérabilité,
- **Ontologies** : concerne les aspects sémantiques de l'interopérabilité d'entreprises.
- **Architectures et plate-formes** : concerne les solutions technologiques.

Les deux volets de notre problématique sont *l'interopérabilité des systèmes d'information* et la *conception de solutions* permettant de l'atteindre. Le système d'information est donc au cœur de nos travaux. Fort de ce constat, nous proposons un cadre méthodologique composé des trois axes d'analyse bibliographique suivants (Fig II.1) :

- **l'axe système d'information, architectures et interopérabilité (1) :** se focalise sur le concept de SI. Nous présentons une analyse bibliographique des différentes définitions d'un SI et considérations attenantes. Nous discutons de la position du SI vis-à-vis d'une collaboration (partie publique et partie privée). Nous définissons également la notion d'architecture de SI et nous nous intéressons notamment à SOA (*Service Oriented Architecture*).
- **L'axe concepts, approches et standards pour l'interopérabilité des SI (2) :** se focalise sur les concepts, approches et standards qui sont concernés par l'interopérabilité des SI. Cet axe est structuré selon les niveaux identifiés dans le chapitre précédent (Chapitre I, Section 4.3) : *organisationnel (métier)*, *sémantique* et *technique*.
- **L'axe conception dirigée par les modèles (3) :** se focalise sur l'approche MDA (*Model Driven Architecture*), qui est basée sur les transformations de modèles et ses principes de base. Nous discutons aussi des travaux liés à l'interopérabilité en relation avec MDA.

Les trois axes sont directement liés à notre proposition de SIC (Système d'Information Collaboratif) (Chapitre I, Section 5.2). Le premier axe s'intéresse aux systèmes d'information, cible de notre étude. Le deuxième axe détaille les approches, solutions et standards liés aux niveaux constatés de l'interopérabilité des SI (Chapitre I, Section 4). Enfin, le troisième axe concerne la conception de solutions pour l'interopérabilité. Le médiateur en est une. La figure (Fig II.1) présente le cadre que nous avons défini.



Conception dirigée par les modèles

Système d'information, architectures et interopérabilité

Fig. II. 1. Cadre méthodologique pour l'analyse bibliographique liée au concept de SIC

PARTIE I :

AXE SYSTEMES D'INFORMATION, ARCHITECTURES ET INTEROPERABILITE

2. Systèmes d'information, architectures et intéropérabilité

Dans cette section, nous allons présenter le concept de système d'information d'entreprise en nous basant sur différentes approches traitant de ce sujet. Nous discutons également des architectures d'un tel système.

2.1. Qu'est-ce qu'un système d'information ?

Au début des années soixante-dix, le concept de système d'information apparaît avec la proposition de G.-B. Davis [**Davis 74**] qui introduit le concept de MIS (*Management Information System*) comme étant le système qui fournit les informations supportant les opérations, la gestion et les prises de décision dans une organisation. Cette approche primaire du SI n'est pas à jour, puisque elle n'évoque pas assez la mémorisation des informations et surtout l'automatisation des « procédures » dans l'entreprise (processus de l'entreprise).

Le concept de système d'information est toujours lié au concept d'organisation. La bonne gestion de l' « information » au sein d'une organisation est capitale pour sa survie. L'organisation attend de son système d'information une augmentation de son efficacité de l'action en assurant une bonne gestion de ses informations. Les informations de l'entreprise portent sur les biens et les services produits ou rendus par l'entreprise, mais aussi sur les moyens (humains, techniques et financiers) que l'organisation utilise pour parvenir à ses objectifs. Cette affirmation est cohérente avec la définition suivante d'une entreprise[1] : *« Tout système socio-économique donné visant la production de biens ou de services pour satisfaire un marché (sa mission) en utilisant au mieux ses moyens (financiers, techniques et humains) ».* Dans la suite, nous présentons trois approches différentes qui ont caractérisé les systèmes d'information.

2.1.1. Une approche systémique d'un système d'information

L'approche systémique positionne un système d'information dans une entreprise comme un sous-système interagissant avec d'autres sous-systèmes. J.-L. Le Moigne [**LeMoigne 77**], [**LeMoigne 90**] définit en effet une entreprise comme la composition de trois sous-systèmes (Fig II.2) :

- **le système opérant** transforme des intrants (entrées) en extrants (sorties) en fonction d'une finalité donnée. Ce système transforme des ressources ou des flux primaires (flux d'information, flux financier, etc.) en d'autres flux ou ressources.
- **Le système de décision** (appelé aussi système de pilotage) élabore des commandes (décisions d'action) en fonction d'informations de suivi. C'est le système qui pilote l'entreprise en fonction des objectifs choisis et des stratégies adoptées.
- **Le système d'information** relie les deux systèmes cités précédemment. Il a pour rôle d'acquérir, mémoriser et transmettre les informations nécessaires au bon fonctionnement de l'entreprise :
    - le comportement du système opérant vers le système de décision,
    - les actions à réaliser par le système opérant, définies par le système de décision.

[1] Définition proposée par le groupe de travail « modélisation d'entreprise » du Groupement De Recherche « Modélisation, Analyse et Conduite des Systèmes Dynamiques ».

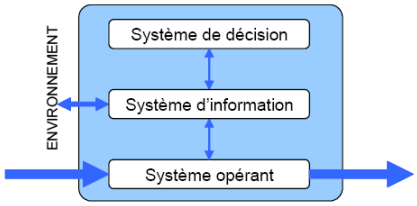


Fig. II. 2. Vue systémique d'un système d'information

2.1.2. Une approche fonctionelle d'un système d'information

Dans cette approche, on s'intéresse aux fonctions qu'un système d'information propose à ses utilisateurs. [**Bernus** *et al*. **96**] considèrent qu'un « *SI doit garantir que la bonne information est disponible au bon endroit et au bon moment* ». Le système d'information a naturellement pour objet principal l'information. C'est l'échange d'information qui permet aux utilisateurs de travailler ensemble. Pour que ce travail soit optimal et efficace, la bonne information doit être disponible au bon endroit (bon utilisateur) et au bon moment. C'est la valeur ajoutée d'un système d'information dans l'entreprise.

[Reix 02] détaille davantage ce qu'offre un système d'information à un utilisateur : un système d'information est un ensemble organisé de ressources :

- matériels (ordinateurs, papier, etc.),
- personnel (clients, responsables, etc.)
- connaissances (modèles, règles, etc.)
- logiciels et procédures (applications informatiques, méthodes de travail, etc.)

qui permet de :

- acquérir des informations,
- traiter (organiser) des informations,
- stocker (mémoriser) des informations,
- communiquer des informations.

Nous ajoutons à ces fonctions celle d'aider les acteurs de l'organisation dans l'exercice de leur métier, en particulier accompagner la prise de décision au sein de l'organisation. Selon [**Saadoun 00**], « *la raison d'être d'un système d'information est l'accès au bon moment à la bonne information pour* ***prendre la bonne décision*** **»**. Cette dernière définition montre que le système d'information s'inscrit dans un processus d'aide à la décision.

2.1.3. Une approche structurelle d'un système d'information

La définition d'un système d'information de C. Morley (Fig II.3) [**Morley 02]** [**Morley** *et al*. **05**] propose de voir un SI comme la composition de deux sous-systèmes : *le système de traitement de l'information* (comprenant les *acteurs*, les *données* et les *processus*) et *le système informatique* (comprenant les *ressources matérielles* et *logicielles*, *les bases de données* et *les fonctions*). L'originalité de cette définition du système d'information est qu'elle met l'accent sur les liens forts entre processus et système d'information. En effet, dans le *système de traitement de l'information* qui est composé d'acteurs, information et processus « *Le* ***processus*** *est un plan d'ensemble indiquant comment les* ***acteurs*** *collaborent au moyen des* ***informations*** *gérées pour accomplir l'objectif de production...* » [**Morley** *et al*. **05**].

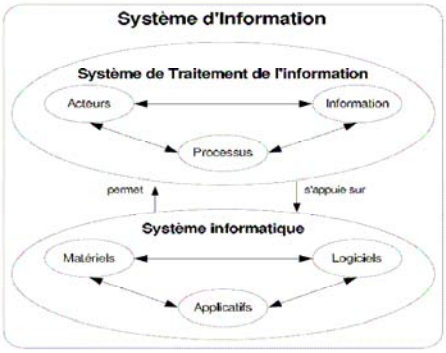


*Fig. II. 3. Une approche structurelle d'un système d'information [**Morley 02**]*

Nous allons préciser les différentes définitions du terme « processus » et son rôle capital dans la définition d'un système d'information dans la suite de ce manuscrit. L'approche de [**Morley 02**] positionne le système informatique comme l'infrastructure du système d'information. L'informatique joue naturellement un rôle important dans le développement des systèmes d'information, mais elle ne présente que la partie physique (applicative) d'un SI. Un système d'information d'entreprise contient aussi une partie non-informatique (archive papier, téléphone, etc.).

2.1.4. Synthèse

Nous avons présenté dans cette section trois approches différentes liées à la compréhension du concept de système d'information. L'approche systémique du SI montre la séparation nette entre les parties *opérationnelle, informationnelle* et *décisionnelle* dans l'entreprise. L'approche fonctionnelle positionne les fonctions rendues par le SI dans l'entreprise autour de l'information. Si cela reste toujours vrai, l'approche structurelle que nous avons présentée, met en avant une notion essentielle pour la définition d'un système d'information : le processus. Celui-ci décrit où, quand et comment les acteurs de l'organisation sont supposés utiliser les informations et les ressources pour atteindre les objectifs fixés par l'organisation. Cette dernière définition traduit le fait qu'un système d'information « *a dépassé le stade d'outil* (gestion de l'information) *pour devenir l'élément structurant d'une organisation* (les processus) » [**Morley** *et al*. **05**]. Nous voulons appuyer dans nos travaux ce rôle structurant des processus pour la définition d'un système d'information.

Dans le reste de ce manuscrit, nous proposons d'utiliser une vision adaptée de système d'information comme **un ensemble de données, applications et processus interagissant**. Si cette vision de SI semble être « technique » et ne pas contenir les dimensions gestion et organisation de SI, cela provient du fait que nous nous intéressions prioritairement dans nos travaux à la conception et la construction d'une projection informatique du SI apte à garantir l'interopérabilité. De plus, les processus qui tiennent une part centrale dans cette vision amènent une certaine organisation du système d'information car ils incluent une description des acteurs impliqués et des échanges qu'ils peuvent avoir.

Enfin, le système d'information a un rôle principal pour faire communiquer l'entreprise avec son environnement et pour faciliter son intégration. Nous pouvons parler d'une évolution du besoin informationnel de l'entreprise. Les informations de l'entreprise doivent être intégrées dans un contexte informationnel qui regroupe d'autres acteurs externes à l'entreprise. Réussir l'intégration de l'entreprise dans son environnement peut

devenir vital pour cette dernière. Nous pouvons qualifier la collaboration de simple communication jusqu'à une intégration complète (Chapitre I, Section 3). La complexité de la mise en œuvre de l'interopérabilité provient essentiellement de l'hétérogénéité des SI. Cette hétérogénéité caractérise les données (présentation, sémantique différente, etc.), les applications (développées dans des environnements différents, etc.) et enfin les processus (organisation différente, divers formalismes de modélisation, etc.).

Cette réalité de la complexité de l'interopérabilité des SI a fait apparaître de nouvelles stratégies, **architectures** et **mécanismes** pour la conception de SI, qui peuvent être qualifiés d'interopérables. Nous nous intéressons dans la section 3.2 à cette notion d'**architecture** de système d'information.

2.2. Architecture de système d'information

L'architecture des systèmes d'information est une discipline multiforme et transversale qui traite de la démarche, des modèles, des outils et des langages permettant de concevoir et construire un système d'information. Elle vise à décrire la structuration d'un système d'information en termes de constituants et d'organisation de ces constituants en tenant compte des multiples niveaux de conception, des aspects de modélisation, mais aussi du degré de généricité du système.

Dans [**Morley** *et al*. **05**], on identifie une architecture de système d'information comme « *une représentation abstraite des différentes parties du système d'information qui permet des décisions globales et de s'assurer de la pertinence de l'assemblage, notamment la cohérence et l'efficience technique…* ».

L'architecture représente alors un enjeu fondamental en vue de permettre une certaine maîtrise de la conception et de l'évolution des systèmes d'information. La maîtrise de la conception passe par une description détaillée des composants, niveaux et organisations mais également par différents modèles de conception d'un système d'information. La maîtrise de l'évolution, comme la définit [**Vernadat 06**] passe par une gestion efficace des changements qui interviennent durant le cycle de vie du système.

Nous ne pouvons pas parler d'architecture de système d'information sans évoquer l'architecture de l'entreprise. Nous dirons synthétiquement que l'architecture de l'entreprise et l'architecture de son système d'information sont étroitement liées. En effet, pour évoluer sur un marché fortement concurrentiel, les entreprises doivent s'adapter aux évolutions du contexte technique et du contexte métier simultanément. Ceci passe nécessairement par la définition d'une **architecture flexible** de leurs systèmes.

Différents types d'architectures d'entreprises (ODP [**Putman 01**], IEEE 1471 [**IEEE 00**], Zachman [**Zachman 87**], TOGAF [**Open group 05**], ARCHIMATE [**Archimate 07**], etc.), de méthodologies ou approches d'ingénierie (PERA [**Pera 07**], CIMOSA [**Vernadat 96**], GERAM [**Geram 97**], Urbanisme [**Lonpégé 02**], etc.), de langages (UML [**Fowler** *et al*. **04**], UEML [**Panetto 06**], BPML [**BPML 07**], etc.) et d'outils (Eclipse [**Eclipse 07**], ARIS [**Sheer 07**], Mega [**Mega 07**], etc.) structurent la discipline des architectures.

Si ces différents éléments forment un ensemble de savoirs conséquents, force est de constater qu'ils sont souvent peu ou mal adaptés à la problématique de l'interopérabilité. En effet, peu d'auteurs ont cherché à développer la propriété de flexibilité d'une architecture. Dans une grande majorité de cas, les composantes du système sont supposées parfaitement définies. La problématique d'intégration est alors formulée comme un besoin

d'assemblage pour garantir la fonctionnalité globale attendue (intégrité des données, compatibilité des traitements, ordonnancement des tâches, etc.).

2.2.1. Exemples d'architecture de système d'information

Il existe une typologie riche d'architectures de système d'information. Ces architectures présentent des niveaux d'abstraction différents d'un système d'information. Nous distinguons :

- **architecture fonctionnelle** : permet de spécifier les besoins fonctionnels des utilisateurs de système d'information. Ainsi, un système d'information est présenté en termes de fonctions, liées entre elles par des flux d'entrée/sortie (*SADT*) ou liées aux acteurs qui les manipulent (*diagramme de cas d'utilisation UML*). Nous parlons aussi d'urbanisme de système d'information, qui consiste à découper le SI en modules autonomes faisant référence aux fonctions de SI. Cette architecture intervient directement pour faire évoluer ou refondre un système d'information.

- **Architecture logique** : permet d'identifier la structuration d'un système d'information en adoptant une logique indépendante des considérations techniques. Les architectures logiques les plus connues sont : l'architecture orientée composants et l'architecture orientée services (SOA). Les composants ou les services sont caractérisés par le fait qu'ils sont des entités autonomes, réagissent par échange de messages et sont définis sur différents niveaux de granularité. La différence principale entre un service et un composant réside dans le fait qu'une architecture orientée services se concentre sur les exigences déterminées au niveau de la stratégie et du processus métier (nous parlons de *service métier*), alors qu'une architecture orientée composants se concentre sur les composants d'un logiciel (nous parlons de *composant logiciel*). Ces composants peuvent être utilisés pour livrer des services. Un service **expose** dans ce cas un composant.

- **Architecture informatique (ou physique)** : décrit la structuration d'un système informatique en termes d'organisation de fonctions et des constituants qui le composent. Cette architecture définit d'autres « sous-architectures » :
  - **architecture logicielle** : l'agencement et l'interaction des composants logiciels. Ces derniers peuvent être définis sur plusieurs couches (couche présentation, couche applicative, couche données).
  - **Architecture matérielle** : l'agencement et l'interaction des composants physiques (disque dur, unité centrale, etc.).
  - **Architecture d'intergiciel (Middleware)** : l'agencement et l'interaction des composants servant à faire communiquer plusieurs applications entre elles.
  - **Architecture réseau** : l'architecture permettant la communication au sein d'un système d'information.

2.2.2. Synthèse

Les architectures de système d'information ont un rôle capital pour la définition d'un système d'information et pour maintenir son évolution. Il existe donc naturellement un lien fort entre l'architecture de système d'information (conception de SI) et la capacité d'interopérabilité que ce dernier peut avoir. Dans nos travaux, nous voulons explorer les apports de deux architectures en matière de facilitation de l'interopérabilité de SI : SOA et MDA.

S'il est tout à fait légitime de présenter MDA dans cette section de chapitre consacrée à l'étude des architectures de système d'information, nous avons préféré la présenter dans l'axe « **conception dirigée par les modèles »**. MDA a permis, en effet, de valoriser l'ingénierie dirigée par les modèles en offrant une architecture structurante. MDA permet de séparer les exigences métiers des spécifications techniques dans une démarche de conception de système d'information. L'idée fondamentale est que les fonctionnalités du système à développer sont définies dans un modèle indépendant de la plate-forme technologique : PIM (*Platform Independent Model*), en utilisant un langage de spécification approprié, puis traduites dans un ou plusieurs modèles spécifiques à la plate-forme : PSM (*Platform Specific Model*) pour l'implémentation concrète du système. Par la suite, nous présentons SOA et ses apports en matière d'interopérabilité.

2.3. L'architecture orientée services

SOA est une architecture logique de système d'information basée sur la notion de « service », qui constitue la brique de base de cette architecture. D'après [**Raymond 07**] « *SOA est un style d'architecture organisé à partir de services métiers communs mutualisés pour un ensemble de lignes métiers ou d'applications* ». Un système d'information qui respecte l'architecture SOA est défini en termes de services proposés. SOA peut être utilisée dans un cadre interne à l'entreprise, ainsi les services peuvent être classés par domaine fonctionnel (achat, finance, marketing, etc.), mais surtout SOA présente un moyen efficace pour faciliter la communication avec d'autres entreprises. Dans ce cas, l'entreprise expose (publie) ses services à ses partenaires. Nous pouvons donner l'exemple d'une agence de voyages qui expose ses services de réservation d'hôtel, de vol, de séjour, etc.

2.3.1. Les éléments de base de l'architecture

F. Vernadat [**Vernadat 06**] identifie trois concepts de base pour une structuration d'un modèle d'entreprise pour SOA : **évènement**, **service** et **processus**. Selon le même auteur, ces concepts doivent être compris et maîtrisés aux niveaux métier (organisationnel) et applicatif (technique). Au niveau métier, ces concepts peuvent être désignés *évènement métier*, *service métier* et *processus métier*. Nous présentons ces concepts tels qu'ils sont définis par [**Vernadat 06**] :

- ***un évènement métier** : est un fait qui se produit en relation avec les opérations de l'entreprise. C'est un changement dans l'état de l'entreprise, qui doit avoir une réponse. Il existe plusieurs types d'évènements : un évènement sollicité (réception d'une nouvelle requête, etc.), un évènement d'exception (panne de machine, etc.), évènement programmé (délai de temps passé, etc.) ou un évènement de synchronisation (début ou fin d'une activité).*
- ***Un processus métier** : un processus métier est un séquencement partiellement ordonné d'un ensemble d'activités et/ou de services dans le but de réaliser un objectif de l'entreprise.*
- ***Un service métier** : est une fonctionnalité de l'entreprise qui apparaît atomique du point de vue de l'appeleur de service. Un service doit être identifié d'une manière unique dans l'entreprise.*

Cette vision de SOA montre le lien fort entre les services et les processus métier d'une entreprise qui a adopté cette architecture. Dans ce cas, un processus métier peut faire appel à un ensemble de services pour s'exécuter. L'orchestration de services (dite aussi composition) réfère à un processus métier qui est composé uniquement de services. Toutefois, un service peut exister d'une manière indépendante (cela ressemble à un processus métier composé d'une seule activité). A ces concepts de base, présentés par [**Vernadat 06**], nous voulons ajouter le concept de **message,** en effet les services communiquent entre eux en utilisant des messages. Un message présente une structure bien définie. Il est important de respecter cette structure pour pouvoir invoquer un service. Les messages illustrent le point de vue informationnel de SOA.

De nos jours, les Services-web[2] fournissent les technologies les plus adaptées pour rendre possible la création de l'architecture orientée services. Cependant, il ne faut pas confondre les Services-web avec SOA. Les Services-web fournissent le support pour la description et l'infrastructure de communication des services, alors que SOA décrit comment un système composé de services peut fonctionner.

2.3.2. Les apports de SOA et l'intéropérabilité

SOA contribue à l'évolutivité, la flexibilité et la pérennité du SI en :

- **encapsulant la complexité de ses applications** : il s'agit de substituer la découpe strictement applicative d'un SI par une structuration en services plus réduits et potentiellement plus simple à faire évoluer [**Raymond 07**] (Fig II.4).

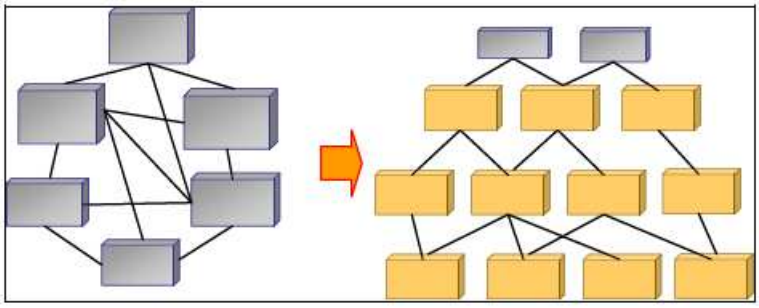


*Fig. II. 4. SOA pour encapsuler la complexité des applications [**Raymond 07**]*

- **favorisant son agilité** : SOA permet de structurer d'une manière dynamique un système d'information. Ce dynamisme est lié à la possibilité de changer facilement une configuration de services : principe de couplage faible (ajouter ou enlever un service, modifier l'ordre de communication de services). De plus, il est possible d'intervenir sur le contenu de service sans toucher à son interface d'accès (changement interne).
- **améliorant son accessibilité :** le système d'information est désormais plus facile d'accès, une fois l'exposition des services réalisée. On parle aussi de publication de services. SOA simplifie les échanges inter-entreprises [**Monfort** *et al.* **04**].

Les trois apports cités ci-dessus (réduction de la complexité, agilité et accessibilité) sont de nature à réduire la difficulté de communiquer entre systèmes d'information. Une difficulté réduite veut dire un moindre effort pour la communication. Rappelons les définitions de

[2] Les Services-web seront présentés dans une partie suivante de ce chapitre.

l'interopérabilité de [**Vernadat 96**] : « *c'est la capacité de communiquer avec d'autres systèmes et d'accéder à leurs fonctionnalités* » et de [**Konstantas** *et al.* **05**] « *c'est la capacité de systèmes à collaborer efficacement sans effort particulier de la part des utilisateurs* ». SOA est une réponse conceptuelle adaptée à la problématique de l'interopérabilité.

2.3.3. Scénario d'exploitation de SOA

La figure (Fig II.5) explique le scénario d'exploitation de SOA. Deux acteurs interviennent dans ce scénario : l'agent fournisseur qui publie les services et l'agent demandeur (ou consommateur) de services.

Les services sont décrits à travers une représentation structurée : XML (*eXtensible Markup Language*), par l'agent fournisseur. Ensuite, ce dernier enregistre les descriptions de ses services dans un annuaire. L'agent demandeur cherche dans l'annuaire des services selon des critères spécifiques au mécanisme de recherche. L'annuaire retourne à l'agent demandeur les informations d'un service demandé. L'agent demandeur récupère la description de ce service et l'utilise pour échanger des messages et communiquer avec le service.

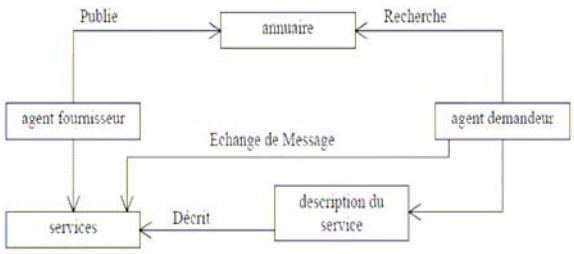


Fig. II. 5. Représentation de scénario d'exploitation de SOA

2.3.4. Synthèse et limites de SOA

Nous avons montré les apports bénéfiques de SOA quant à l'agilité, l'accessibilité et la pérennité d'un système d'information. SOA est une solution attractive pour l'interopérabilité des systèmes d'information. Cependant, le passage des entreprises d'une architecture classique (applications difficilement accessibles) vers une architecture SOA reste toujours une opération délicate et coûteuse. Dans un autre contexte, les limites de SOA quant à la vue sémantique de l'interopérabilité restent un point faible de cette approche. La question de la sémantique des applications peut néanmoins être traitée par l'approche SOA : de quel service a-t-on exactement besoin pour la réalisation de cette tache ? Parlons-nous d'un même service ? Quelques travaux de recherche se sont intéressés au problème [**Zaidat 05**], [**Izza 06**] en proposant des approches d'enrichissement sémantique des services, qui permettent d'offrir des mécanismes d'interopérabilité sémantique.

2.4. Structuration adaptée de système d'information pour la collaboration inter-entreprises

Le but de cette section est de présenter une structuration adaptée d'un système d'information dans un contexte de collaboration inter-entreprises ainsi que les mécanismes

qui aident l'entreprise à mieux gérer son ouverture vers l'extérieur. L'entreprise doit ouvrir l'accès d'une partie de son système d'information à ses partenaires, suivant le contrat de collaboration signé par ces derniers. La contribution de son système d'information à cette collaboration consiste à :

- communiquer avec les partenaires (messagerie instantanée, e-mail, etc.),
- proposer des services à rendre,
- proposer l'accès (en lecture et dans des cas fréquents en écriture) aux données et ressources de l'entreprise,
- intégrer ses processus internes aux processus collaboratifs définis par les contrats de collaboration de l'entreprise.

Cependant, l'ouverture et l'intégration du système d'information de l'entreprise dans un réseau collaboratif doivent se faire d'une manière contrôlée et sécurisée. [**Vanderhaeghen** *et al.* **04**] explique que « *pour des raisons stratégiques liées à la compétitivité de l'entreprise, la plupart des entreprises partenaires d'un réseau collaboratif ne dévoilent pas directement leurs savoir-faire et connaissances liés à leurs processus métiers internes* ».

Dans le même registre, [**Panetto 06**] atteste que « *le partage d'information entre les applications de l'entreprise pose encore des problèmes liés au secret professionnel et à la propriété intellectuelle. Le processus d'interopérabilité dépend aussi des applications mises en place, de leur ouverture et de la disponibilité de leurs modèles internes* ».

Dans un contexte d'interconnexion de procédés d'entreprises, d'après K. Benali ([**Benali 04**]) : « *Bien que les entreprises aient besoin de passerelles d'interconnexion entre leurs procédés, elles peuvent être concurrentes directement ou indirectement. De ce fait, tout modèle d'interconnexion doit prendre en compte les aspects de confidentialité des procédés (i.e. distinction entre informations privées et informations partagées, etc.)* ».
En conclusion et dans un contexte de collaboration d'entreprises, le système d'information doit prendre en compte ces nouvelles exigences de confidentialité de l'information.

2.4.1. Une représentation privée/publique de système d'information

Nous considérons qu'un SI dans un contexte de collaboration inter-entreprises doit être représenté suivant deux parties (Fig II.6) :

- **la partie privée** d'un système d'information comprend une description strictement propriétaire de ce dernier. Cette description est uniquement accessible à l'entreprise [**Bauer** *et al.* **06**]. C'est une approche **boîte noire**, où les données, les processus, les applications, l'organisation et même les stratégies de l'entreprise sont cachées aux partenaires externes à l'entreprise [**Vanderhaeghen** *et al.* **04**].
- **La partie publique** d'un système d'information (appelée aussi abstraite [**Bauer** *et al.* **06**]) est une abstraction de la partie privée de l'entreprise. Dans cette partie, on décrit la valeur ajoutée apportée à la collaboration par l'entreprise sans rentrer dans les détails. C'est une approche **boîte blanche**, où l'on détaille les fonctionnalités et les informations relatives à la collaboration et dont les partenaires ont besoin. Des approches comme SOA permettent de faciliter cette abstraction de système d'information [**Vanderhaeghen** *et al.* **06**]. Les services font abstraction du système d'information : les partenaires communiquent avec les services en termes de données d'entrée et de sortie, sans connaître les spécificités du système avec lequel

ils communiquent. L'entreprise se connecte alors au réseau collaboratif par le biais de la partie publique de son système d'information.

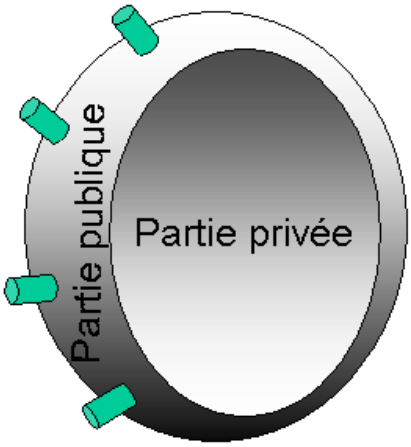


Fig. II. 6. Représentation d'un système d'information dans un contexte de collaboration

2.4.2. Les mécanismes de publication/privatisation

Suite à des changements stratégiques de l'entreprise ou des changements qui peuvent survenir dans le contrat de la collaboration, l'entreprise peut décider la restriction ou l'élargissement du périmètre de la collaboration avec ses partenaires. En effet, l'entreprise peut restreindre l'accès à certaines ressources de son système d'information ou au contraire, mettre plus de ressources dans la collaboration. Nous définissons deux mécanismes que l'entreprise peut utiliser afin de réagir efficacement avec son environnement (Fig II.7) :

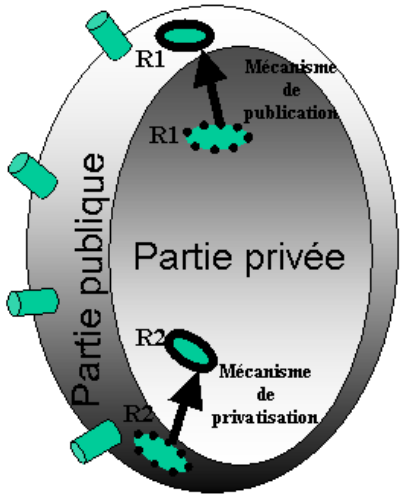


Fig. II. 7. Les mécanismes de publication et de privatisation

- **mécanisme de publication :** ce mécanisme permet de rendre publique une ressource : donnée, accès à un service ou à une application. C'est un passage de la ressource du domaine privé du SI (restreint à l'entreprise) au domaine public (destiné pour la collaboration). L'accès à cette ressource peut concerner un certain nombre de partenaires identifiés (au détriment d'autres).

- **Mécanisme de privatisation :** ce mécanisme permet de rendre privée une ressource : donnée, accès à un service ou application. C'est un passage du domaine public du SI au domaine privé.

2.5. Conclusion

Nous avons défini dans cette partie du manuscrit le concept de système d'information. Les entreprises qui possèdent des SI interopérables sont celles qui réussissent le mieux leur intégration car, rappelons le, l'interopérabilité est un moyen de tendre vers l'intégration. Le système d'information présente dans ce cas le pilier de l'interopérabilité de l'entreprise. Cette interopérabilité des systèmes d'information est loin d'être un objectif facile à atteindre. La coexistence de partenaires de culture et d'habitude de fonctionnement potentiellement différentes au sein du réseau, que l'on peut assimiler à de l'hétérogénéité, ne facilite pas la conception et l'usage de l'interopérabilité des systèmes d'information. Des architectures de système d'information tendent à faciliter l'interopérabilité. SOA permet d'ajouter une certaine agilité pour la structuration de SI, faciliter l'accès à ses services et réduire la complexité de ses applications. Enfin, l'entreprise doit définir d'une manière claire l'ouverture de son système d'information vers l'extérieur dans un but d'intégration. Elle doit contrôler l'accès à ses ressources. En ce qui concerne l'interopérabilité, elle doit être dans ce cas contrôlée et maîtrisée.

PARTIE II :

AXE CONCEPTS, APPROCHES ET STANDARDS POUR L'INTEROPERABILITE DES SYSTEMES D'INFORMATION

3. Concepts, approches et standards pour l'interopérabilité des systèmes d'information

Dans cette partie, nous nous intéressons aux concepts, standards et approches qui ont contribué à faciliter l'interopérabilité des systèmes d'information sur ses trois niveaux (organisationnel (métier), sémantique et technique).

3.1. Le niveau métier pour l'interopérabilité des systèmes d'information

Ce niveau décrit l'organisation des acteurs, des interactions et des échanges entre les différents systèmes d'information du réseau collaboratif. Cette interopérabilité métier (indépendante de toute considération technologique) des systèmes d'information fait référence à la notion de **processus collaboratif**. Dans cette section du manuscrit, nous nous intéressons à cette notion, qui permet d'expliquer la synchronisation des échanges d'information et de ressources entre partenaires.

3.1.1. Notion de processus

Parmi les définitions existantes d'un processus, nous retenons :

- Une définition essentielle : « *un processus est un ensemble partiellement ordonné d'**étapes exécutées** en vue de réaliser au moins un objectif »* **[Vernadat 99]**.
- Une autre définition, qui s'intéresse plus à la structure d'un processus : *« un processus représente l'organisation d'un ensemble finalisé d'activités effectuées par des acteurs et **mettant en jeu des entités** »* [**Morley 02**].
- La norme ISO 9001 (2000) [**ISO 00**] définit, quant à elle, le processus comme : « *un ensemble d'activités corrélées ou interactives qui transforment des éléments d'entrée en éléments de sortie* ». Cette définition intègre la réalisation des produits ou des services, mais elle n'explicite pas l'objectif associé au processus.
- La définition issue de **[Théroude 02]**, nous semble la plus complète : « *un processus est défini comme un enchaînement partiellement ordonné d'exécution d'**activité**s qui, à l'aide de **moyens techniques et humains**, transforme des éléments d'**entrée** en éléments de **sortie** en vue de réaliser un **objectif** dans le cadre d'une **stratégie** donnée ».*

Sur la base de ces définitions, nous pouvons caractériser un processus par :

- des **activités**, qui sont les éléments d'action atomiques. Une activité exprime la transformation d'une ressource d'entrée en une ressource de sortie.
- Un **graphe d'activités**, qui représente l'enchaînement des activités nécessaires à la réalisation de l'objectif.
- Des **rôles**, qui expriment l'organisation dans le processus.
- Un **objectif**, qui représente l'expression de la finalité du processus.
- Une **fonction de transition**, qui contrôle le déroulement du processus.
- Des **ressources**, qui peuvent être des moyens, des informations ou des outils utilisés par une activité. La norme ISO9004 précise que *« toute activité utilisant des **ressources** et gérée de manière à permettre la transformation d'éléments d'entrée en éléments de sortie est considérée comme un **processus** »*.

3.1.2. Notion de « processus collaboratif »

La notion de processus **inter-organisationnel** ou **collaboratif** est particulièrement importante dans un contexte d'entreprise en réseau. [**Morley** *et al*. **05**] précise que ce sont les avancées technologiques qui ont permis des organisations « globales », favorisées par l'ouverture des marchés et faisant coopérer des entités et des ressources différentes. Un processus collaboratif fait alors intervenir des organisations différentes. [**Aubert** *et al.* **02**] définit un processus collaboratif comme « *un processus dont les activités peuvent appartenir à des organisations différentes*. Cette vision, si elle semble simple, n'est pas anecdotique dans la mesure où elle est porteuse de la notion d'externalisation du traitement au sein d'un processus collaboratif. [**Morley** *et al*. **05**] définit un processus inter-organisationnel comme un processus où « *les partenaires ont une **maîtrise partielle** sur ce dernier* ». La maîtrise partielle sur le processus est due à un partage rigoureux des activités de processus entre les partenaires. Ces derniers sont responsables uniquement de l'exécution de leurs activités. Nous pouvons donc parler de « contrat » ou « règlement » afin d'assurer le bon déroulement d'un processus. Nous pouvons donner un exemple classique d'un processus qui regroupe un client et un fournisseur. Dans ce cas, le client gère des activités liées au domaine d'achat. Le fournisseur gère des activités liées au domaine de vente. Chacun des acteurs n'a dans ce cas qu'une vue limitée de l'activité de l'autre. D'où l'importance d'une explication claire des modalités de partage d'information. D'après [**Morley** *et al*. **05**], **le contrôle de processus global ne peut qu'être réparti entre les acteurs**.

Les activités des partenaires peuvent faire référence à leurs processus internes. Du point de vue de l'entreprise, une **activité collaborative** fait référence à un processus interne et présente une interface dédiée à la collaboration, sans que les détails de ce processus interne ne soient visibles pour les autres partenaires. Cette assertion est compatible avec la structuration d'un système d'information dans un contexte de collaboration telle que nous l'avons présentée (Section 3.4). Dans ce cas, l'activité collaborative appartient au domaine public du SI, le processus interne (lié directement à l'activité collaborative) appartient au domaine privé du SI.

Nous retenons qu'un processus collaboratif est **un ensemble partiellement ordonné d'activités spécifiquement organisées chez les partenaires de la collaboration et leur exécution est hébergée chez ces derniers.**

3.1.3. Caractérisation de formalismes de modélisation de processus collaboratif

[**LeMoigne 90**] définit la modélisation comme *« l'élaboration et la construction intentionnelle par composition de symboles, de modèles susceptibles de rendre intelligible un phénomène perçu complexe, et d'amplifier le raisonnement de l'acteur projetant une intention délibérée au sein du phénomène »*. Un modèle possède alors une syntaxe définie et chaque élément de modèle véhicule une sémantique particulière. [**Zaidat 05**] ajoute que la modélisation est basée sur des techniques appropriées : *« le processus de modélisation dans les entreprises est une tâche très complexe. Il consiste à décrire un agencement d'un nombre d'éléments important dont la nature est différente. La réalisation de ce processus requiert l'utilisation de techniques de modélisation appropriées. C'est dans ce sens que les architectures de référence ont proposé des cadres de modélisation pour la conduite du processus de modélisation »*.

Ces techniques de modélisation sont basées sur des formalismes de modélisation bien spécifiques au contexte de la problématique. Pour notre objectif de modélisation de processus collaboratif, plusieurs **formalismes de modélisation de processus**, sont utilisés.

Afin de comparer les différents formalismes de modélisation de processus présentés par la suite, nous nous basons sur un ensemble de critères, organisés en différentes catégories. La plupart des critères sont dérivés de [**Nurcan 07**], [**Morley** *et al*. **05**] et [**Mougin 02**]. Ils sont présentés dans la liste suivante :

- **représentation de la granularité** : un processus peut se décomposer en sous processus.
- **Description de l'enchaînement d'activités** : un processus métier doit être décrit en termes d'enchaînement d'activités.
- **Traitement de conditions** : possibilité de définir les conditions d'exécution d'une activité en fonction de variables internes au processus (coûts, délais, etc.).
- **Gestion des évènements** : prise en compte des évènements externes et des exceptions dans le processus.
- **Gestion documentaire** : possibilité d'assigner des documents à une activité métier.
- **Gestion de la performance** : mettre en œuvre un suivi de performance d'un processus. Il est important d'attacher à chaque activité des critères de mesure de la performance (coûts, délais, qualité, etc.).
- **Description des ressources** : une ressource doit pouvoir être typée (système, homme, machine) et décrite (capacité, fiabilité, localisation…).
- **Intégration des vues** : une intégration d'autres vues d'entreprise (informationnelle, organisationnelle, décisionnelle, de ressources) doit être possible ou facilitée.
- **Contrôle avancé de flux de processus** : possibilité d'inclure des symboles de contrôle avancés tels, qu'une horloge, pour décrire une période de temps passée.
- **Lien avec une définition exécutable** : existence d'un lien direct avec une définition exécutable de processus : BPEL (*Business Process Execution Language*), BPML (*Business Process Modelling Language*), etc.

3.1.4. Présentation des formalismes de modélisation de processus

Nous présentons dans cette section deux catégories de formalismes de modélisation de processus, qui ne sont pas au même stade d'évolution :

- **les formalisme primaires** : présentent uniquement une représentation comportementale (évènementielle) pour décrire les processus. Il s'agit de formalismes simples qui sont basés sur une approche élémentaire du processus. La modélisation du processus est alors réduite à cette vision : *un enchaînement d'un ensemble d'activités*.
- **Les formalismes évolués** : peuvent inclure, en plus, une représentation informationnelle (donnée) ou/et organisationnelle (acteurs). De plus, les formalismes évolués peuvent offrir une typologie riche d'activités, d'évènements, de contrôles de flux, etc. Ces formalismes peuvent être dédiés à des domaines particuliers : processus de système d'information : UML (*Unified Modelling Language*), processus métiers : BPMN (*Business Process Management Notation*), etc.

3.1.4.1. Formalismes primaires

- **Organigrammes (Flow chart)** :

présentent probablement les premières notions de modélisation de processus. Ce sont des représentations graphiques d'une séquence logique d'activités : opérations, données, flux, équipements, etc. Les caractéristiques des organigrammes sont flexibles et simples.

- **Réseau de Pétri (RdP)** :

fortement utilisé dans le domaine industriel, il sert à traiter les problèmes de synchronisation d'activités. Les notions graphiques sont : les places (nœuds), les arcs (flèches) et les transitions (contrôles). Les places correspondent aux activités des processus modélisées. Les arcs sont associés aux évolutions du processus et des flux d'informations. Les transitions représentent les évènements ou les conditions à vérifier pour avancer dans le processus.

- **IDEF3 et SADT (IDEF0)** :

formalismes basés sur les mêmes symboles et concepts, mais relevant de méthodes différentes. SADT est basé sur le concept de « boîte », qui peut représenter une activité (actigramme) ou une donnée (datagramme). Dans le premier cas, la boîte permet de présenter les données en entrée et en sortie, les ressources et les données de contrôle. IDEF3 reprend les notions de base de SADT et ajoute les opérateurs logiques comme « AND » et « OR » qui permettent de diriger le flux dans un processus.

3.1.4.2. Formalismes évolués

- **Diagrammes d'enchaînement de processus d'ARIS**

ARIS [**Scheer 94**] est une méthode de modélisation d'entreprise fortement utilisée dans le monde industriel (SAP[3], BaanERP[4], etc.). La vue de processus dans cette méthode est basée sur les diagrammes d'enchaînement de processus ou *Process Chain Diagram* (PCD).

Les PCD peuvent exister sous deux formes : *Tableau* ou *Diagramme*. Le *Tableau* réunit les entités provenant des différents modèles de vue, ce qui permet d'offrir une représentation « étendue » des processus de l'entreprise.

Ce formalisme contient aussi des contrôles de flux et permet de décrire une typologie des activités (batch, interactive, manuelle). Le *Tableau* permet de séparer les concepts (Evénement, Fonction, Données, Système applicatif et Unité d'organisation) et de les présenter dans des couloirs.

Cette représentation offre une vue générale de la composition structurelle des processus et de leurs relations avec les autres vues de l'entreprise. Le *Diagramme* reprend exactement les mêmes concepts, mais avec une représentation « à plat ». Ce formalisme est bien adapté à la représentation des « workflows » et offre une clarté indéniable sur l'intégration de processus dans les autres vues de l'entreprise (Fig. II.8).

[3] http://www.sap.com/index.epx.
[4] http://www.baan.com/.

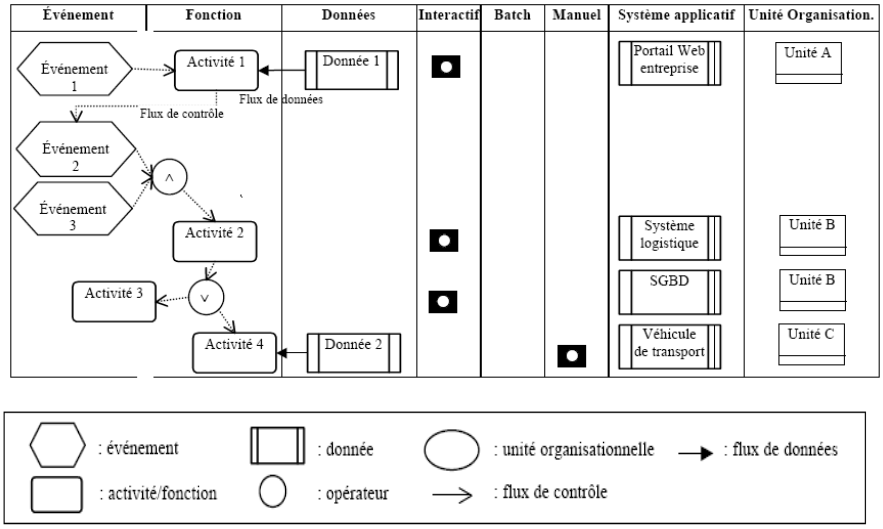


Fig. II. 8. Un diagramme d'enchaînement de processus ARIS

- **UML**

Le langage UML est une référence en matière de modélisation de système selon une approche orientée-objet [**Booch** *et al.* **00**] [**Booch** *et al.* **04**] qui peut s'adapter au monde de l'entreprise. UML offre bien, au travers des diagrammes d'activité et de séquence une possibilité de modéliser des processus de l'entreprise.

Chantal Morley [**Morley 00**] considère qu'UML n'intègre pas, nativement, le concept de processus, vu qu'il n'est pas destiné, à l'origine, à ce type de modélisation. Elle propose cependant une démarche de modélisation de processus avec UML. La première étape consiste à présenter d'une manière générale tous les processus à l'aide des **diagrammes de cas d'utilisation**. Chaque processus est considéré comme un cas d'utilisation à part et il sera détaillé à l'aide d'autres diagrammes, par la suite. Un **diagramme d'état transition** permet de définir les différents états d'une entité impliquée dans le processus (par exemple, les états possibles d'une commande sont : validée, envoyée, reçue, etc.). Le **diagramme de séquence** permet de traduire la communication entre les acteurs des processus. **Le diagramme d'activité** joue le rôle le plus important en détaillant les activités des étapes de processus, l'organisation mise en œuvre et les entités utilisées. **Le diagramme de classe** permet de donner des détails relatifs aux entités impliquées dans le processus.

L'intégration des diagrammes UML entre eux permet de construire des vues complémentaires du système. La figure ci-dessous (Fig. II.9) extraite de [**Sakli 00**] montre bien, à travers un exemple, la facilité de lier les différentes vues entre elles. Le nombre de diagrammes constitue une critique principale d'UML pour son utilisation dans un contexte de modélisation de processus. Les profils UML fournissent un mécanisme capable de spécialiser le langage UML au contexte métier. Un profil UML[5] est un « package » stéréotypé contenant des éléments adaptés à un domaine spécifique en utilisant des mécanismes d'extension. le profil UML pour EDOC (*Entreprise Distributed Object Computing Systems*) [**OMG 02**], permet en partie de décrire un ensemble d'extensions du langage UML pour formaliser les processus métiers. Ce profil fournit les concepts permettant de composer des activités, de sélectionner des critères sur les entités qui supportent ces activités, de coordonner les processus (évènement, communication, etc.).

[5] Les profils UML seront présentés plus en détail dans le quatriéme chapitre

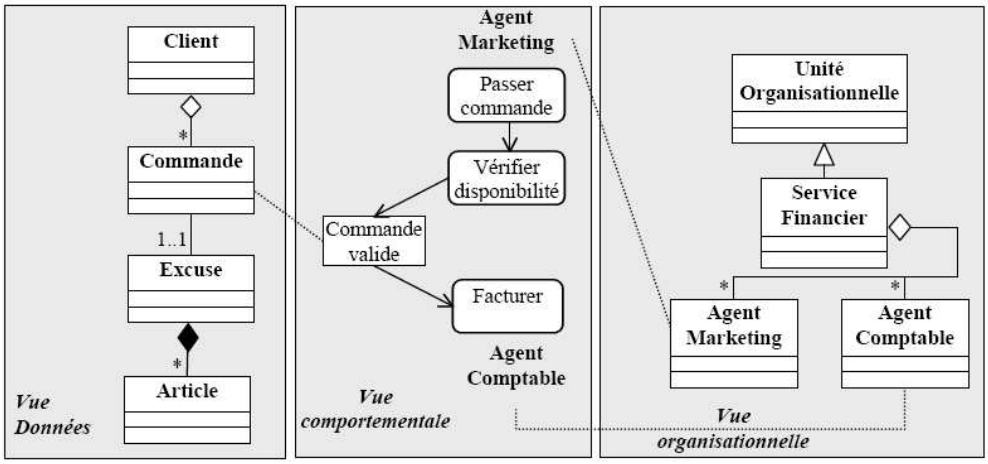


Fig. II. 9. *L'intégration des vues dans UML à travers un exemple de processus*

- **BPMN**

BPMN est un langage semi-formel de modélisation de processus [**BPMI 04**] défini par le BPMI[6] (*Business Process Management Initiative*). Ce dernier est un organisme qui anime un courant de standardisation dans le domaine du management par les processus métiers.

Les origines de BPMN reviennent à une réponse à la RFP – *Request for Proposal* – soumise par l'OMG[7] (*Object Management Group*), pour une spécification permettant la définition et la représentation des processus métiers. La spécification proposée devait permettre la modélisation des processus à différents niveaux : métier et technique. BPMI a proposé la spécification BPMN dans sa première version 1.0.

BPMN a pour but d'offrir une notation explicite, facile d'emploi et accessible à tous les utilisateurs métiers. Mais, de façon plus ambitieuse, BPMI souhaite que ce formalisme devienne ***un standard*** utilisable par les **concepteurs de systèmes d'information**. En effet, BPMN n'est pas destiné à un acte simple de modélisation dans le but unique de capitaliser des connaissances, il propose des passerelles pour l'exécution automatique des processus (sous format BPEL) à l'aide de moteurs BPMS (*Business Process Management System*).

Cette facilité le positionne comme un candidat sérieux pour toute problématique de collaboration de systèmes. Par la suite, nous présentons succinctement des éléments de base de ce formalisme, en les utilisant pour décrire notre exemple. Les éléments de base de représentation graphique avec BPMN sont de trois types :

- **les conteneurs** (point de vue *organisationnel*) que sont les couloirs (*« pool »*) et les bandes (*« lane »*). Ce sont des partitions du processus qui relèvent d'un acteur ou d'une entité organisationnelle particulière.
- **Les nœuds du graphe** (point de vue *fonctionnel*) représentent les activités (*« activities »*, symbole rectangulaire), les évènements (*« events »*, symbole circulaire) et les branchements conditionnels (*« gateways »*, symbole losange).
- **Les arcs** (point de vue *informationnel*) représentent les flux d'information. Il en existe trois types : les flux de contrôle séquentiel, qui caractérisent une

[6] www.bpmi.org.

[7] www.omg.org.

communication interne (dans un même couloir ou une même bande, *« sequence flow »*, trait plein), les flux de message, qui caractérisent une communication inter-conteneurs (*« message flow »*, trait pointillé) et les associations (« *Association* », trait incliné). Cette dernière est utilisée, par exemple, pour associer un élément de documentation d'une entité, pour identifier des données en tant qu'entrées ou sorties d'une activité ou pour associer une compensation à une activité, etc.

La figure (Fig II.10) présente un exemple de modélisation BPMN d'un processus de collaboration Client-Fournisseur. Le diagramme BPMN est réalisé avec l'outil *Process Modeler for Visio©* [8] :

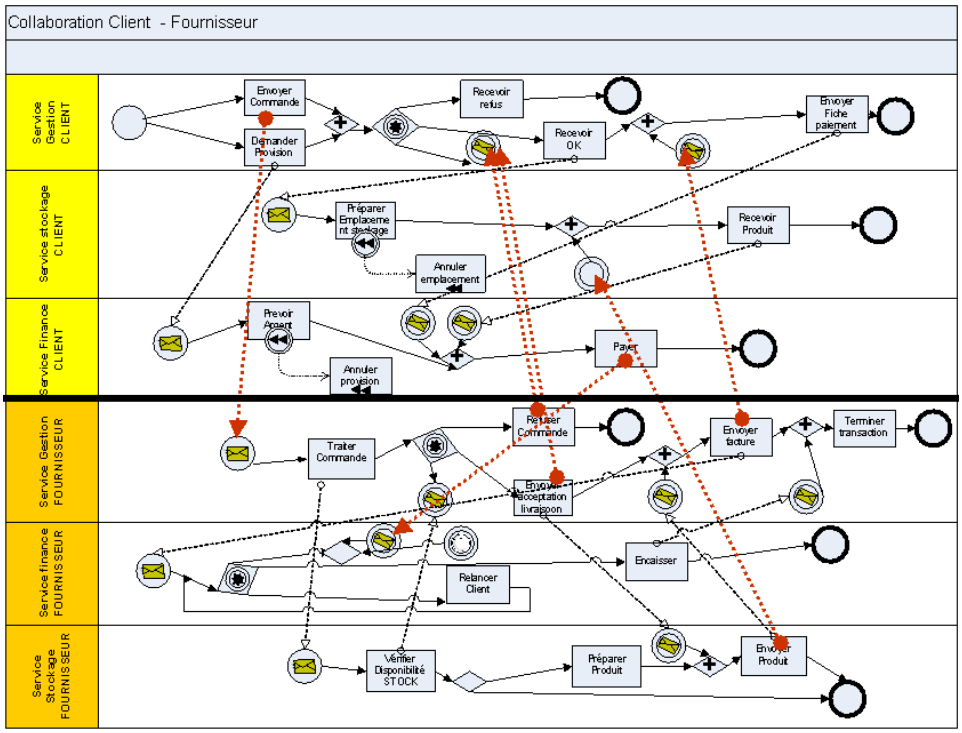


Fig. II. 10. *Un exemple de modèle BPMN (processus de collaboration Client-Fournisseur)*

Dans cet exemple, les couloirs sont utilisés pour délimiter chaque partie (deux couloirs représentent respectivement le client *en haut* et le fournisseur *en bas*), et les bandes permettent de localiser le processus parmi les services de chaque partie (trois bandes représentent les services stockage, finance et gestion des parties client et fournisseur).

Les évènements dans BPMN sont typés et à chaque type correspond un symbole particulier normalisé : début et fin, arrivée d'un message, échéance d'une temporisation, exception, nécessité de mettre en œuvre une compensation (défaire ce qui a été fait précédemment pour revenir à un état stable).

Sur notre exemple, la réponse du Service Stockage du fournisseur à la requête de son Service de Gestion pour vérifier le niveau des stocks est représentée par un évènement particulier : l'arrivée d'un message (lettre à l'intérieur du symbole circulaire). De la même manière, un évènement peut être provoqué par la fin d'une temporisation (horloge à l'intérieur du symbole circulaire), qui correspond au délai limite pour le paiement du fournisseur. Dans ce cas, le fournisseur doit relancer le client en envoyant un message

[8] http://www.itpearls.com/.

(nouvel évènement). Le processus se poursuit selon une ou plusieurs branches conditionnelles contrôlées par des « *gateways* ». Lorsqu'il est sollicité, un tel branchement reçoit des flux d'entrée, les évalue suivant une condition logique et préside à la poursuite de la séquence d'exécution sur une partie donnée du graphe en aval.

BPMN peut mettre en évidence les relations de synchronisation entre les processus de chaque organisation. Ce sont des flux de message *« message flow »* qui représentent ce lien particulier. Les deux parties communiquent par des échanges d'informations entre deux couloirs respectifs. Une identification des points de communication est donc possible, en repérant les flux qui traversent la frontière des deux organisations. On identifie ainsi clairement l'interface entre les systèmes d'information des deux entreprises. Ces synchronisations entre les deux parties correspondent à l'acte même de coopération. Lorsque ce processus est pris en charge par voie informatique, les systèmes d'information du client et du fournisseur sont donc en relation. Une partie de chaque système doit être visible par l'autre, afin d'assurer le bon déroulement des activités. Autrement dit, chaque partenaire présente une interface « publique » pour la collaboration avec l'autre partie. BPMN peut modéliser un processus où les activités correspondent à des services au sens de SOA. En résumé, les points forts du formalisme BPMN sont :

- une notation intuitive et plus ergonomique à l'usage des acteurs de l'organisation et de la gestion d'entreprise.
- Un vocabulaire riche et adapté aux besoins de conception de processus métiers complexes – ensemble de concepts et de relations – rigoureusement défini pour fournir un socle robuste à l'outillage des approches processus.
- Un lien fort avec le format d'échange BPEL.

3.1.5. Synthèse

Nous nous sommes attachés à présenter le concept de « processus collaboratif » et les formalismes les plus courants et les plus fréquemment utilisés, pour le modéliser. Nous avons pu constater l'évolution de la présentation graphique et des techniques de contrôle de flux d'activités entre les formalismes primaires et les formalismes évolués. Pour être plus général dans notre étude, il aurait aussi fallu inclure les formalismes de modélisation de processus évolués suivants : OSSAD (*Office Support Systems Analysis and Design*) [**Dumas** *et al.* **90**], PSL (*Process specification Language*) [**Schlenoff** *et al.* **99**] et POP* (Product, Organization, Process and Systems) [**Athena 05**]. Utiliser un formalisme de modélisation de processus évolué permet d'avoir une couverture globale du processus, en maîtrisant ses acteurs (vue organisationnelle) et ses informations (vue informationnelle). Cette couverture reste partielle, car c'est toujours le point de vue fonctionnel qui reste dominant dans ces formalismes.

Le tableau suivant (Tab. II.1) propose une comparaison basée sur les critères que nous avons identifiés (Section 4.1.3) des formalismes que nous avons étudiés. Nous pouvons voir que les organigrammes sont le formalisme qui répond le moins à nos critères (critère de base pour l'enchaînement d'activités). PCD, UML et BPMN semblent être proches par rapport à nos critères. Cependant, la différence principale est que BPMN a été construit spécifiquement pour la modélisation des processus métier (contrairement à UML). De plus, BPMN possède un lien direct avec un langage d'exécution BPEL. BPMN pourra alors permettre de réconcilier modélisation des processus métiers et besoin de l'informatique. Même si certains outils le font de manière semi-automatique, le passage d'une modélisation des processus métiers à une modélisation pour l'exécution de ces processus

demande un travail d'adaptation manuelle. Ces adaptations peuvent entraîner des erreurs et même rendre difficile la compréhension, pour les modeleurs, des évolutions de leurs propres processus.

Critère/ Formalisme	Organigrammes	Réseau de Pétri	IDEF3 /SADT	PCD d'ARIS	UML	BPMN
Représentation de granularité	Non	Oui, dans des versions évoluées	Oui	Non	Oui	**Oui**
Description de l'enchaînement d'activités	Oui	Oui, dans des versions évoluées	Oui	Oui	Oui	**Oui**
Traitement de conditions	Non	Oui	Non	Oui	Oui	**Oui**
Gestion des évènements	Non	Oui, dans des versions évoluées	Non	Oui	Oui	**Oui**
Gestion documentaire	Non	Non	Non	Non	Oui	**Oui**
Gestion de la performance	Non	Non	Non	Non	Non	Non
Description des ressources	Non	Non	Non	Oui	Oui	**Oui**
Intégration des vues	Non	Non	Non	Oui	Oui	**Oui, des liens d'intégration**
Contrôle avancé de flux de processus	Non	Non	Non	Non	Non	**Oui**
Lien avec définition exécutable	Non	Non	Non	Oui, BPML	non	**Oui , BPEL**

Tab. II.1 : comparatif des formalismes de modélisation de processus.

3.2. Le niveau sémantique pour l'interopérabilité des systèmes d'information

Nous avons mentionné dans ce manuscrit l'importance du niveau sémantique en évoquant l'interopérabilité des systèmes d'information. L'hétérogénéité sémantique d'après [**Izza 06**] « *ne concerne pas seulement les données, mais peut aussi concerner les autres couches de l'intégration dont notamment les fonctions et les processus…*». L'interopérabilité sémantique consiste alors à « *s'assurer que l'information échangée* (qui concerne les données, les processus ou les applications) *a le même sens du point de vue de son expéditeur et du point de vue de son destinataire* » [**Pokarev** *et al.* **06**]. Dans ce cas, la résolution des conflits sémantiques doit faire appel à un rapprochement des différentes interprétations sémantiques des éléments du SI. Mais avant d'établir ce rapprochement sémantique entre systèmes d'information, les entreprises doivent d'abord bien **définir**, **décrire** et **formaliser** la connaissance liée à leurs systèmes d'information, afin de pouvoir

la partager avec d'autres. Dans ce cadre, les **ontologies** présentent le candidat idéal pour cette tâche de conceptualisation de la connaissance de système d'information.

3.2.1. Présentation de la notion d'ontologie

Le terme « **ontologie** », qui dans son acception initiale a un sens philosophique (la théorie de l'être), a été appliqué au discours scientifique, au langage et dans des domaines professionnels précis, tels que la documentation, par exemple. Dans le domaine industriel, une ontologie d'entreprise est une matérialisation explicite et persistante des connaissances et informations cruciales d'une entreprise pour faciliter leur accès, partage et réutilisation par les acteurs de l'entreprise dans leurs tâches individuelles et collectives. L'ontologie est « *une composante de la mémoire d'entreprise qui capture une connaissance potentiellement intéressante en elle-même pour l'entreprise* » [**Pokarev** *et al.* **06**].

Les ontologies sont devenues des éléments fondamentaux dans toute une gamme d'applications faisant appel à des connaissances. La structure d'une ontologie permet de représenter les connaissances d'un domaine sous un format informatique en vue de les rendre utilisables pour différentes applications [**Guarino 99**]. Une ontologie peut être utilisée soit (i) pour partager des connaissances entre agents humains et / ou artificiels, (ii) pour raisonner sur des bases de connaissances ou (iii) pour faciliter la recherche d'information au sein d'un même système ou de systèmes hétérogènes. Elle est aussi primordiale pour la médiation des systèmes d'information ayant des modèles et des sémantiques hétérogènes. C'est donc une clé de toute première importance dans l'étude de l'agilité des partenaires et dans le développement de l'activité réseau.

Afin de répondre à toutes ces finalités, une ontologie ne doit pas uniquement considérer le volet terminologique d'un domaine comme peut le faire un thésaurus, mais doit intégrer toutes les connaissances de ce dernier. Ainsi, les ontologies actuelles, qui correspondent pour la plupart à des ontologies qualifiées de légères, car incluant seulement quelques propriétés de structuration telles que la subsomption et les propriétés algébriques, doivent évoluer vers des ontologies plus « denses » sémantiquement parlant, incluant tous les axiomes permettant de représenter la sémantique du domaine considéré [**Grüninger** *et al.* **95**].

La définition d'une méthodologie **de construction et de validation** des ontologies est nécessaire. La construction de l'ontologie fait partie d'un processus, qui permet d'évaluer et de faire évoluer cette ontologie lorsque de nouvelles connaissances sont détectées dans le système ou suite à une modification des propriétés d'un de ses éléments (Fig II.11). Cela signifie que l'ontologie doit être disponible et refléter l'existant à tout moment [**Pokarev** *et al.* **06**].

Une ontologie peut s'exprimer sous la forme d'un ensemble de schémas XML ou en utilisant OWL. OWL (*Ontology Web Language*) [**OWL 04**] est un méta-langage XML dédié à la description des ontologies standardisé par le W3C (*World Wide Web Consortium*) basé sur RDF (*Resource Description Framework*) et RDFS (*RDF Schema*). OWL permet de décrire des relations et contraintes complexes sur des ontologies (exclusion, dépendance, etc.) et est composé de trois sous-langages de complexité croissante : OWL-Lite, OWL-DL et OWL-Full. RDF et OWL sont au cœur des évolutions futures du « semantic Web », qui ajoute aux informations disponibles sur le Web une composante sémantique, ce qui permet des traitements plus intelligents de l'information et donc une intégration facilitée.

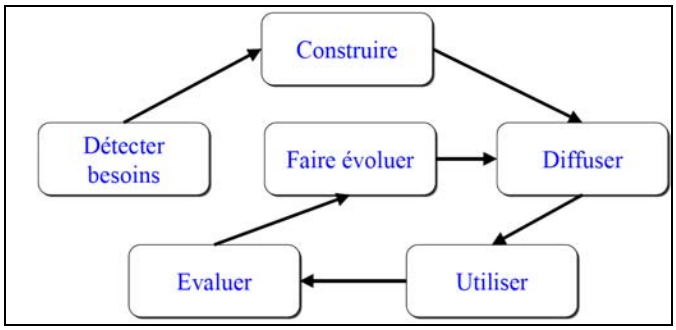


*Fig. II. 11. Le cycle de vie de la construction de l'ontologie entreprise [**Pokarev** et al. **06**].*

3.2.2. Exemples d'ontologies standardisées

Un grand nombre d'initiatives ont été développées afin de spécifier des ontologies en utilisant des schémas XML. Elles sont classées généralement suivant deux axes :

- **vertical** : s'il s'agit d'initiatives dédiées à un métier (domaine) particulier : MathML[9] (*Mathematic Markup Language*) pour le domaine des mathématiques, GML[10] (*Geography Markup Language*) pour le domaine de la géographie, etc.
- **Horizontal** : s'il s'agit d'initiatives pouvant être déclinées pour des métiers (des domaines) différents : UBL[11] (*Universal Business Language*) est un standard de l'OASIS visant à normaliser en XML les documents commerciaux courants.

3.2.3. Architectures des ontologies pour l'interopérabilité des SI

Dans [**Wache 01**] et [**Izza 06**], on décrit trois approches d'architecture d'ontologies spécifiques au domaine de l'intégration des données : l'approche mono-ontologie, l'approche multi-ontologies et l'approche hybride, qui combine les deux :

- **approche mono-ontologie** : elle permet de mettre en œuvre une ontologie unique et partagée entre les différents systèmes d'information. D'après [**Vernadat 06**], cette ontologie est utilisée comme un langage pivot pour lier sémantiquement les concepts d'un système avec les concepts d'un autre. Cette ontologie regroupe une connaissance commune et partagée par différents SI qui participent à la collaboration. C'est une ontologie où chacun des partenaires se reconnaît et peut communiquer facilement avec les autres. La mise en œuvre de cette architecture est simple lorsque les sources de données font référence à des domaines similaires. En effet, il devient difficile de s'entendre sur une seule ontologie lorsque les métiers des partenaires sont trop éloignés.
- **Approche multi-ontologies** : chaque système d'information est décrit par sa propre ontologie. Des mécanismes de transformation et de correspondance entre les différentes ontologies sont alors nécessaires pour lier leurs différents concepts (*mapping inter-ontologies*). La mise en œuvre de ces mécanismes est difficile. Cette difficulté augmente avec le nombre de partenaires et la distance entre leurs domaines respectifs. Cette architecture est plus avantageuse, car elle permet de garder l'intégrité et l'indépendance des connaissances d'un système d'information.

[9] http://www.w3.org/Math/.
[10] http://www.opengeospatial.org/.
[11] www.oasis-open.org/committees/ubl/.

- **approche hybride** : tire avantage des deux approches précédentes : chaque source est décrite par sa propre ontologie, et toutes les ontologies des sources (ontologies locales) sont reliées à une ontologie globale permettant de partager un vocabulaire commun.

3.3. Le niveau technique pour l'interopérabilité des systèmes d'information

Cette partie s'intéresse à un panorama des technologies et des standards qui ont facilité l'intégration entre les systèmes d'information sur le niveau technique. La technologie de l'EAI permet une intégration des applications en assurant entre autres des mécanismes de transformation syntaxique de données. Cette technologie peut permettre aussi une intégration par les processus métiers. Les outils de gestion de « workflow » et de BPM (*Businnes Process Management*) s'intéressent à l'automatisation et à l'exécution des processus métiers de l'entreprise dans un cadre interne et du réseau collaboratif dans un cadre plus large. Le langage XML adopté par la communauté industrielle a permis l'apparition de plusieurs technologies connexes. Nous citons les intergiciels et les Services-web. La technologie ESB (*Enterprise Service Bus*) permet un support efficace pour la réalisation d'une approche SOA en communiquant les Services-web des entreprises.

3.3.1. Le langage XML

Nous l'avons vite constaté, la structuration de données prend une place prépondérante dans le domaine des systèmes d'information. En effet, un système d'information a pour objectif premier d'organiser l'échange de données entre acteurs d'entreprise. Dans cette optique, XML constitue un langage de structuration de données, adapté à des contextes aussi divers que les applications distribuées, la configuration de produits, les annuaires, l'édition de documents, la diffusion de contenu sur le web ou la gestion de la connaissance.

XML est une recommandation[12] du W3C[13] (*World Wide Web Consortium*). C'est un langage de balises, définissant un format universel de représentation des données. Un document XML contient à la fois des données et des indications sur le rôle que jouent ces données. Ces indications (balises) permettent de déterminer la structure du document. La structure logique d'un document XML est obtenue en analysant ses balises. Un document XML est composé d'un ensemble d'éléments fils contenant éventuellement un ensemble d'attributs. Un document XML peut être défini par deux normes. La première est la spécification XML, qui stipule les règles applicables par défaut à la création de tout document XML (document bien formé). La deuxième norme, qui est facultative, est créée par les auteurs du document et spécifiée dans la définition du type de document : XSD (*Xml Schema Definition*). Tout document XML qui respecte les règles définies dans le document XSD se définit comme étant un document XML valide. Un document XSD est lui-même un document XML. Un document XSD définit :

- des éléments qui peuvent apparaître dans le document.
- Des attributs qui peuvent apparaître dans le document.
- Les éléments fils d'une balise.

[12] XML est une recommandation de W3C depuis 1998. Une recommandation est un document stable pouvant être utilisé comme document de référence ou cité comme une référence normative.

[13] http://www.w3.org/.

- L'ordre des éléments fils,
- Le nombre d'éléments fils,
- Des types de données pour les éléments et attributs.

La figure (Fig II.12) montre un exemple de document XML qui décrit un e-mail. Les fils de la balise racine <email> sont : <a> (expéditeur), <de> (destinataire), etc. Le document XSD décrit la structure du fichier XML (l'élément <a> est de type chaîne de caractères, etc.). Les documents DTD (*Document Type Definition)* permettent aussi de décrire les documents XML. La puissance de description des DTD est faible par rapport aux XSD : une DTD permet uniquement de décrire la structure d'un document XML (liste des balises et organisation des balises), et non la typologie des données contenues (chaîne de caractère, date, entier, etc).

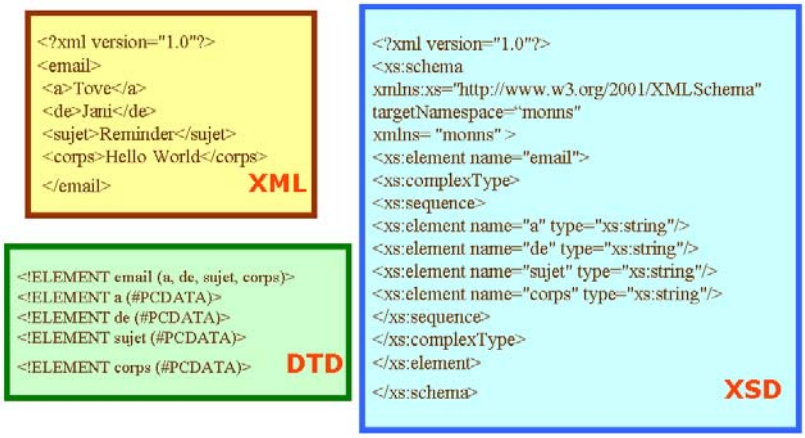


Fig. II. 12. Un exemple de document XML et son schéma de données XSD

La manipulation des documents XML est réalisée par des parseurs (*parser*). Ces parseurs peuvent explorer les documents XML afin de lire l'information recherchée ou la modifier. Ces parseurs peuvent être appelés par des API (*Application Programming Interface*) intégrées dans des langages de programmation tels que JAVA. Parmi les parseurs les plus connus, nous distinguons SAX (*Simple API for XML*)[14] et DOM (*Document Object Model*) [15].

3.3.2. Les outils de workflow

L'acquisition d'outils permettant la modélisation, la gestion et le suivi des processus et de l'information est capitale pour améliorer les performances de l'entreprise. Cela passe obligatoirement par une maîtrise de l'information, une meilleure coopération des acteurs de l'entreprise et une optimisation de ses processus métiers. L'objectif est d'aboutir à une meilleure vue globale de l'ensemble des processus métiers de l'entreprise et de leurs interactions afin d'être en mesure de les optimiser et, dans la mesure du possible, de les automatiser au maximum à l'aide d'applications métier. Le concept de workflow est standardisé par l'organisme WfMC (*Workflow Management Coalition*)[16] **[WfMC 99]**.

[14] http://sax.sourceforge.net/

[15] http://www.w3.org/DOM/

[16] WfMC (Workflow Management Coalition) : organisme regroupant des industriels de l'informatique, des chercheurs, et des utilisateurs dans le domaine du workflow. Le but de WfMC est de définir des standards pour le développement d'applications Workflows.

Plusieurs définitions du terme workflow existent, mais un problème de confusion persiste entre les termes : *workflow* (processus workflow), *technologie workflow* et *système workflow*. Par la suite, nous allons définir chacun de ces termes en nous appuyant sur des définitions issues de WfMC **[WfMC 99]**.

3.3.2.1. Définition d'un workflow

- Un **workflow** est la forme exécutable d'un processus d'une organisation, gérable par un système workflow **[WfMC 99]**. Il permet l'automatisation de l'exécution du processus ou encore sa simulation.
- Un **système workflow** (ou SGWF pour Système de Gestion de WorkFlow) est un système informatique dédié à la gestion des processus métiers. Les services proposés par un SGWF sont au minimum l'exécution d'un processus et sa gestion (contrôle et suivi) ainsi que la mise à disposition des documents et outils nécessaires à la réalisation des étapes de processus **[WfMC 99]**.
- La **technologie workflow** est la technologie informatique du TCAO (*Travail Coopératif Assisté par Ordinateur*), qui s'occupe de la gestion des processus de l'organisation. C'est l'ensemble des moyens mis en œuvre pour automatiser et gérer un processus. Cette gestion repose sur la possibilité de représenter un modèle de processus par une forme exécutable. **[WfMC 99]**

La relation entre les trois termes est simple. Une entreprise peut introduire une **technologie workflow** dans son système d'information en mettant en place un **système de gestion de workflow** qui gère ses processus, automatisés en **workflow**.

3.3.2.2. Les concepts de base du workflow

La WfMC propose un méta-modèle[17] pour la définition du workflow. Ce méta-modèle permet de faciliter la compréhension des concepts de base du workflow. Ce méta-modèle identifie un ensemble élémentaire d'objets fondamentaux qui entrent dans la définition d'un processus géré par un système workflow. Ces concepts sont quasiment les mêmes que ceux d'un processus (Section 4.1.1). Nous citons : *activité*, *acteur*, *rôle* et *condition de transition*. La différence majeure concerne le concept de *ressource* (de processus) qui se trouve réduit dans le méta-modèle de workflow de la WfMC à *application externe*. En effet, dans la technologie des workflow les activités échangent des *données* et sont exécutées par *applications externes* gérées par des *acteurs*.

3.3.2.3. Description de l'architecture d'un SGWF

Un SGWF est composé d'un ensemble de modules qui permettent la définition, l'exécution et le suivi d'un workflow. Cependant, nous pouvons découper l'architecture d'un SGWF en trois parties [**WfMC 99**] :

- **une partie responsable de la modélisation de workflow :** Il s'agit d'un outil informatique basé sur une interface graphique permettant la modélisation d'un workflow sous une notation existante (par exemple : BPMN) ou propriétaire. De tels outils permettent de générer des modèles de workflow exploitables.
- **Une partie responsable de la gestion complète des modèles réalisés :** cette gestion comprend l'exécution des workflow (*workflow engine*), la distribution des tâches aux rôles appropriés, la mise à disposition de l'ensemble des données et des outils nécessaires, la supervision et le contrôle de la cohérence etc.

[17] un méta-modèle permet de décrire les différents élements d'un modèle, ainsi que les relations entre eux.

- **Une partie responsable de la gestion des connexions externes avec d'autres applications ou modules :** cette partie contient des fonctions API qui assurent ce genre de connexions. Ces fonctions permettent d'adapter le système et ses services en fonction de leurs besoins spécifiques.

La figure (FigII.13) illustre l'architecture décrite ci-dessus. La partie « *API Workflow* » s'occupe de la gestion des connexions externes avec d'autres applications ou autres services. La partie « *service d'exécution* » assure la gestion des modèles workflow réalisés. Enfin, les « *outils de définition des processus* » assurent la modélisation des workflows. L'architecture présentée est une architecture généraliste des SGWF établie par la WfMC. Il existe en effet plusieurs types d'architectures de systèmes, chacun basé sur des concepts spécifiques en plus des concepts de base, le plus souvent invariants. Nous avons donc plusieurs critères, sur lesquels nous pouvons fournir une classification des SGWF.

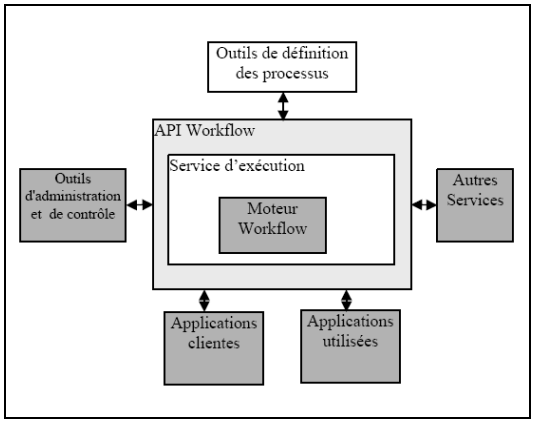


*Fig. II. 13. Architecture d'un SGWF [**WfMC 99]***

3.3.3. L'EAI

Les systèmes d'information actuels doivent s'appuyer sur différentes applications logicielles, qu'ils doivent faire communiquer rapidement et de manière efficace. Cela montre une évolution de l'ingénierie du système d'information de l'entreprise. En effet, d'une logique d'ingénierie qui visait à réaliser des systèmes d'information « clés en main » répondant spécifiquement à un besoin, nous passons à une nouvelle logique au sein de laquelle il est de plus en plus nécessaire d'intégrer des applications existantes qui n'ont pas été conçues en vue de cette intégration.

Au début des années 1990, les connecteurs et les protocoles de transport de données n'étaient pas basés sur des standards : ils étaient spécifiques à l'interaction entre deux applications. La plupart des échanges entre applications se faisaient par l'intermédiaire de fichiers non structurés, avec tous les risques d'erreurs que cette solution basique pouvait engendrer. Ce type d'intégration a donné lieu à des architectures dites « accidentelles », résultant d'un amalgame de connexions propriétaires hétérogènes, comparable à un plat de spaghettis, construites au fur et à mesure de l'intégration de nouvelles applications dans le SI (partie gauche de Fig II.14).

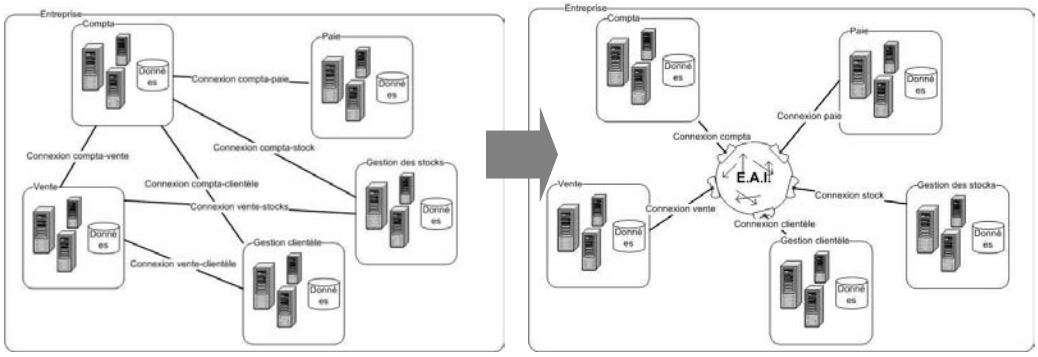


Fig. II. 14. *Une approche EAI d'intégration des applications d'entreprise*

Pour que les différentes applications interopèrent correctement malgré leur hétérogénéité, il est nécessaire qu'elles présentent une **bonne complémentarité fonctionnelle**, qu'elles **se synchronisent correctement** (notion de processus) et qu'elles **interprètent de la même manière** les contenants et les contenus des structures de données échangées. Ceci suppose également la résolution des problèmes techniques à tous les niveaux d'interfaces, tant des applications que des infrastructures technologiques (plate-formes, SGBD, systèmes d'exploitation, réseaux) qui les supportent. L'EAI (Fig II.14) offre une solution technologique à l'intégration des applications en simplifiant les échanges entre les différentes applications d'un ou plusieurs systèmes d'information. Cette gestion permet une meilleure maîtrise de la complexité et de l'évolutivité des SI.

Parmi les définitions existantes de l'EAI, nous retenons : « *l'EAI est un ensemble d'outils, de progiciels d'intégration qui offrent un moyen de connecter et de faire communiquer entre elles les applications de l'entreprise, existantes ou à venir* » [**Hostachy 00**]. Selon cette définition, nous constatons que l'EAI permet d'offrir un support pour l'évolution de l'entreprise en matière d'applications à intégrer dans son système d'information en fonction du changement de marché.

Les processus d'une entreprise (ou d'un réseau d'entreprises) font nécessairement appel à bon nombre d'applications. l'EAI, en tant que support à l'intégration des applications de l'entreprise, présente nécessairement une relation forte avec ces processus et le Système de Gestion de WorkFlow (Fig II.16). L'exécution des activités des processus est basée sur différentes applications hétérogènes de l'entreprise. L'EAI fonctionne en les faisant interagir, suivant un processus défini, afin de proposer une vision unifiée de l'information.

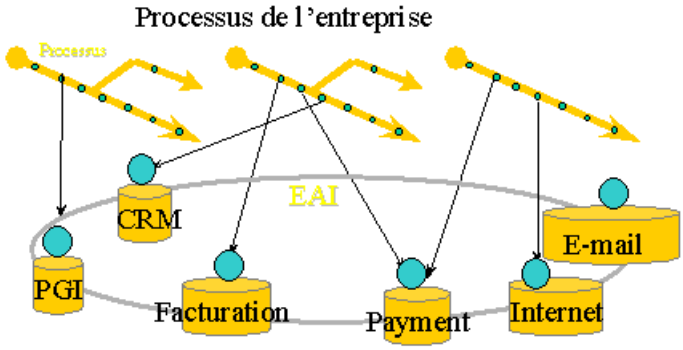


Fig. II. 15. *Une intégration par les processus métiers des applications de l'entreprise*

Après avoir présenté la définition de l'EAI ainsi que ses objectifs, nous allons présenter l'architecture relative à cette technologie.

3.3.3.1. L'architecture EAI

L'architecture, qui peut être considérée comme idéale, est une architecture qui permet à chaque application de se connecter au SI de façon indépendante et unique, sans avoir de connaissance *a priori* du domaine global. L'application obéit aux ordres du SI en traitant des tâches et en retournant des informations suivant l'exécution des processus métiers. Dans ce type d'architecture, le SI de l'entreprise peut être modulé de façon triviale, les ajouts, modifications ou suppressions d'applications n'ayant aucun impact sur le reste du domaine. L'architecture d'une EAI est composée de quatre couches (Fig II.16), qui permettant la liaison entre les processus de l'entreprise et ses applications [**Blanc 04**] :

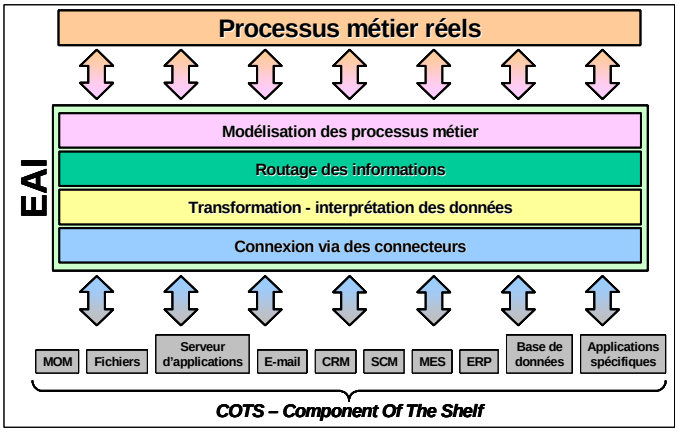


Fig. II. 16. L'architecture EAI

- **modélisation des processus métier :** les outils d'EAI sont généralement couplés à des outils de BPM[18] (*Business Process Management*), qui automatisent les processus de l'entreprise. On définit alors les échanges entre les différents départements de l'entreprise. Cette couche illustre le positionnement stratégique des systèmes de workflow dans le contexte EAI. Les workflow permettent d'automatiser l'exécution d'une partie des processus de l'entreprise. Cette automatisation consiste à traiter les activités nécessaires, les informations manipulées et les applications requises pour l'exécution de ces processus.
- **Routage des informations :** cette couche a la mission d'aiguiller les différents flux d'information vers les bonnes applications. Cet aiguillage est rendu possible grâce à la représentation des processus métier fournie par la couche modélisation décrite ci-dessus. D'un point de vue technique, cet aiguillage peut notamment être rendu possible grâce à la mise en place d'outils logiciels permettant de gérer des files d'attentes de messages (les outils Middleware).
- **Transformation-interprétation des données :** cette couche sert à structurer le format des données véhiculées au sein de la plate-forme EAI. Le but de cette couche est donc de transformer (dans les deux sens) le format des messages (données des diverses applications) en un format pivot propre à la plate-forme EAI. A l'heure actuelle, ce format pivot est généralement basé sur XML.

[18] BPM : analyse et modélisation logicielle des processus mis en place par l'entreprise pour réaliser ses activités

- **Connexion via des connecteurs :** cette couche propose des connecteurs spécifiques pour les différentes applications de SI afin de les faire communiquer dans une approche processus. Ces connecteurs permettent d'interroger les différentes applications pour récupérer et / ou insérer des données. Cette couche gère également la définition des **protocoles de transport** qui permettent d'acheminer et de distribuer les données.

Le principal reproche que l'on peut cependant faire aux outils d'EAI est leur aspect propriétaire. La logique d'intégration d'un EAI étant propriétaire, les éléments de l'outil (connecteurs, transformateurs de données, orchestrateur des processus) ne sont pas standardisés, ce qui lie l'entreprise à l'éditeur qu'elle aura choisi, avec les conséquences coûteuses (coût du conseil et des interventions, pérennité de la solution, etc.).

3.3.4. Les Services-web

Les Services-web sont des composants logiciels encapsulant des fonctionnalités métier de l'entreprise et accessibles via des protocoles standards du web [**Zhao 05**]. Une définition plus précise des Services-web est fournie par le dictionnaire Webopedia (http://www.webopedia.com). Il définit un service-web comme « *une manière standardisée d'intégration des applications basées sur le web en utilisant les standards ouverts XML, SOAP, WSDL, UDDI et les protocoles de transport de l'Internet. XML est utilisé pour représenter les données, SOAP pour transporter les données, WSDL pour décrire les services disponibles, et UDDI pour lister les fournisseurs de services et les services disponibles* ». Les Services-web présentent l'application la plus concrète de SOA.

D'après cette dernière définition, nous retenons qu'un Service-web est décrit dans un document WSDL[19] (*Web Service Description Language*), précisant les méthodes (fonctionnalités) pouvant être invoquées, leur signature et les points d'accès du service (URL, port, etc.). Ces méthodes sont accessibles via des requêtes supportées par une couche de transport de messages : SOAP[20] *(Simple Object Access Protocol)*. La requête et la réponse sont des messages XML transportés par HTTP (*Hyper Text Transfer Protocol*). Un Service-web peut être référencé dans une sorte d'annuaire UDDI[21] (*Universal Description, Discovery and Integration*) qui facilite l'accès à ce service aux utilisateurs. On peut utiliser un Service-web pour exporter des fonctionnalités d'une application d'entreprise et les rendre accessibles via des protocoles standard. Le Service-web sert alors d'interface d'accès et de dialogue avec l'application (Fig II.17).

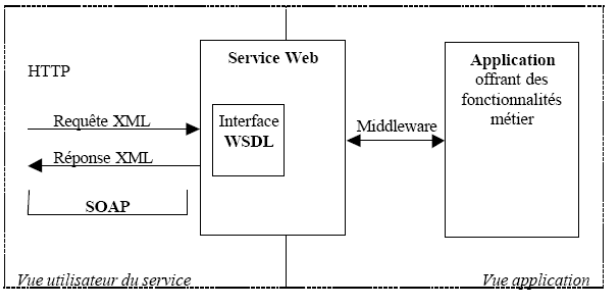


Fig. II. 17. La publication des fonctionnalités des composants industriels sur le web

[19] http://www.w3.org/TR/wsdl.
[20] http://www.w3.org/TR/soap/.
[21] http://www.uddi.org/.

Les Services-web relèvent de la technologie la plus récente utilisée pour supporter le développement des systèmes d'information distribués, en particulier les applications B2B (*Business to Business*) sur Internet. Aujourd'hui, ils semblent être la solution la plus adaptée pour assurer l'interopérabilité sur Internet. Les Services-web sont basés sur des technologies standardisées, ce qui réduit l'hétérogénéité et fournit un support pour l'intégration d'applications.

La technologie des Services-web vise l'indépendance maximale des services. Un service doit être le plus indépendant possible de la structure des autres services et doit également imposer le minimum de contraintes aux entités amenées à l'utiliser. Il est alors possible de définir rapidement et facilement de nouveaux processus métiers, par assemblage de services existants. On permet ainsi à l'entreprise de réagir rapidement aux évolutions de son contexte de marché.

Pour réaliser ce couplage faible, l'effort de conception de l'architecture doit porter sur la normalisation des interfaces des services, définissant une manière de communiquer la plus souple et la plus riche possible. Cette normalisation entraîne également une indépendance entre cette couche de communication et la couche de transport effective. En résumé, les services-web sont aujourd'hui associés à trois spécifications XML (Fig II.18) :

- **SOAP** : pour le transport des données et l'infrastructure de communication.
- **WSDL** : pour la description des services offerts.
- **UDDI** : annuaire pour le référencement des services par les fournisseurs et leur découverte par les utilisateurs.

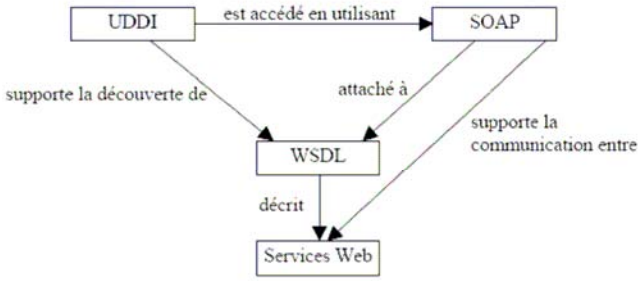


Fig. II. 18. Les spécifications XML liées aux services-web et leurs relations

Nous pouvons ajouter un quatrième standard, qui est de plus en plus utilisé pour l'orchestration des Services-web: **BPEL**. Par la suite, nous présentons la technologie des ESB basée principalement sur les Services-web.

3.3.5. L'ESB

Un ESB est une solution d'intégration implémentant une architecture totalement distribuée et fournissant des services comme la transformation des données ou le routage basé sur le contenu, ainsi qu'une interopérabilité accrue par l'utilisation systématique des standards comme XML, les Services-web et les normes WS-*[22] **[EBM 05]**. L'ESB est une solution « packagée » qui permet de mettre en œuvre l'approche SOA.

[22] WS-* : Ensemble de normes associées aux Web Services et traitant par exemple la sécurité ou l'orchestration de services.

3.3.5.1. Architecture et caractéristiques

Comme le présente la figure (Fig II.19) la technologie ESB est centrée sur la notion de bus, qui permet d'assurer une intégration distribuée des différents services métiers connectés. Nous pouvons remarquer qu'un ESB gère plusieurs types de connecteurs : les Services-web, les API spécifiques, etc. Par la suite, nous présentons les principales caractéristiques de la technologie ESB, qui sont issues de [**EBM 05**] :

- **Distribution**

La notion de distribution est centrale pour un ESB. En effet, par essence les applications à intégrer sont réparties sur différentes machines ou systèmes d'information. Par la mise en œuvre de ce principe de distribution, il existe des données de configuration et d'administration pour chaque application à intégrer. Ceci permet de contourner les problèmes de l'architecture centralisée « *hub and spoke* », proposée classiquement par les solutions EAI, et amène à des architectures sans SPOF (*Single Point Of Failure*)[23].

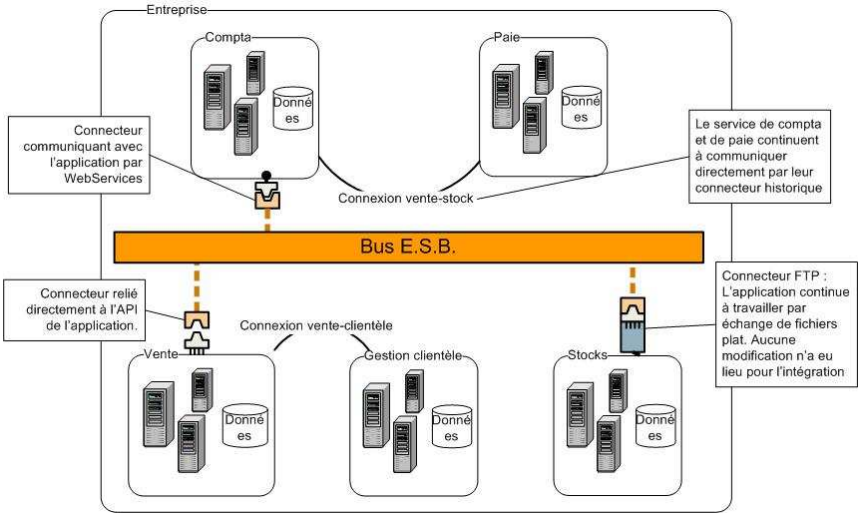


*Fig. II. 19. Architecture du SI d'une entreprise bâtie autour d'un ESB [**EBM 05**]*

- **Fiabilité**

Comme toute solution d'entreprise, un ESB doit apporter des garanties de fiabilité. Dans cette optique, la plupart des ESB sont construits sur des MOM (*Message Oriented Middleware*), et tous permettent de l'utiliser comme moyen de transport. L'utilisation d'un MOM garantit dans certaines configurations que les messages sont bien transmis. L'autre point important concernant la fiabilité est la possibilité de construire des architectures sans SPOF, comme décrit ci-dessus. Ainsi, quand un serveur tombe en panne, le reste du système peut continuer à fonctionner.

- **Interopérabilité et ouverture du système d'information**

Les ESB sont basés sur des standards reconnus, ce qui facilite leur interopérabilité et l'interconnexion des SI de deux entreprises partenaires utilisant des ESB, même différents (Fig II.20). Les messages circulant dans un bus peuvent être transmis au bus du partenaire. A terme, les aspects techniques de transaction, sécurité, etc. seront supportés complètement par la mise en œuvre des normes WS-*, spécifiques à chaque problématique.

[23] SPOF : point unique par lequel passent tous les traitements et qui paralysent l système en cas de panne.

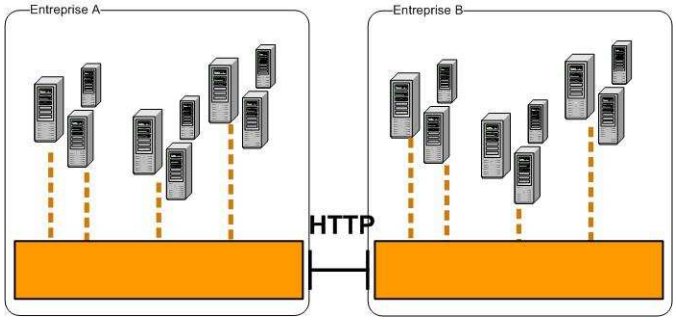


Fig. II. 20. Deux entreprises reliées par leurs bus ESB

- **Connectivité**

Pour connecter les différentes ressources applicatives à intégrer, l'ESB propose un ensemble de connecteurs basés sur la norme J2CA (*J2EE Connector Architecture*)[24]. Ainsi, tous les ESB proposent des connecteurs techniques vers la plupart des formats techniques d'échange : formats XML, RMI (*Remote Method Invocation*)[25], RPC (*Remote Procedure Call*)[26], etc. Certaines offres ESB incluent des connecteurs « métiers », permettant d'intégrer des progiciels du marché. Dans tous les cas, l'utilisation de J2CA permet d'utiliser des connecteurs fournis par les éditeurs de logiciels, selon ce standard.

- **Services techniques**

Un ESB doit offrir des services techniques, comme la transformation des messages, le routage basé sur le contenu et éventuellement l'orchestration des services (Fig II.21). Ces services techniques proposés par les ESB permettent de mettre en œuvre des intégrations par couplage faible, c'est-à-dire qu'une application n'a pas à s'adapter aux formats ou aux spécificités des applications qu'elle intègre via l'ESB : les adaptations éventuelles sont traitées au niveau de l'ESB.

- **Normes et standards**

Les normes pertinentes dans le cadre des ESB relèvent de trois domaines :

- *Standards W3C* **:** il s'agit des standards relatifs aux Services-web comme XML, SOAP, WSDL.
- *Standards OASIS*[27] : Normes WS-* comme WS-Security, WS Adressing, etc. ainsi que les normes BPEL et UDDI.
- *JSR* (*Java Specification Request*)**[28]** : J2CA et JBI (*Java Business Integration*)[29].

[24] fournit une architecture standard pour l'intégration d'applications réparties.

[25] est une API pour le langage Java qui permet d'appeler des objets distants.

[26] permet à des processus s'exécutant dans des environnements différents de faire des appels de méthodes à travers un réseau.

[27] www.oasis-open.org.

[28] décrit les demandes des utilisateurs de JAVA pour faire évoluer, modifier ou ajouter les fonctionnalités fournies par la plate-forme officielle de SUN.

[29] définit une architecture qui permet la mise en place de solutions d'intégrations basées sur l'utilisation de composants.

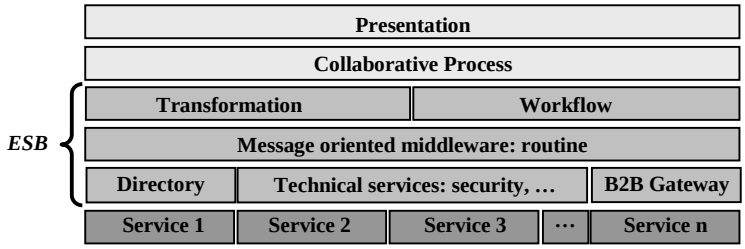


Fig. II. 21. Les couches d'un ESB

3.4. Conclusion

Dans cette deuxième partie de chapitre, nous avons présenté les outils, standards et approches qui contribuent à faciliter l'interopérabilité des systèmes d'information sur ses trois niveaux (métier, sémantique et technique).

Au niveau métier, nous nous sommes attachés à présenter les « processus collaboratifs », qui ont une place centrale dans un contexte d'entreprise en réseau. Ces processus expliquent l'organisation des acteurs, des interactions et des échanges entre les différents systèmes d'information du réseau collaboratif ; d'où l'importance de bien assurer une modélisation rigoureuse des processus collaboratifs de l'entreprise, afin de réussir son interopérabilité métier avec ses partenaires.

Sur le niveau sémantique, nous avons présenté la notion d' « ontologie » comme moyen permettant de faciliter l'établissement de l'interopérabilité sémantique entre systèmes d'information hétérogènes. Nous avons également présenté des approches différentes d'architecture d'ontologie, dans un contexte d'interopérabilité. Ces architectures sont basées sur l'utilisation de plusieurs ontologies également hétérogènes qu'il faut mettre en correspondance. Cette hétérogénéité peut être décrite aux niveaux : syntaxique, terminologique, conceptuel et pragmatique [**Izza 06**]. Dans ce cas, des mécanismes de mise en correspondance doivent être définis et implémentés. Cela reste un domaine de recherche ouvert, même si des travaux ([**Izza 06**], [**Zaidat 05**]) ont étudié ces aspects et proposent des solutions pour résoudre les différentes hétérogénéités qui peuvent exister au niveau des ontologies.

Enfin, **au niveau technique**, il existe un large choix d'outils et de technologies, facilitant la communication entre différents systèmes d'information. XML apporte une structuration des informations qui peuvent être échangées entre SI. Cela a déclenché l'émergence d'un grand nombre de technologies et d'initiatives fondées sur XML, comme norme de représentation et d'échange de l'information. Les systèmes de gestion de workflow permettent le contrôle de l'envoi des informations entre les participants d'un processus collaboratif ainsi que la définition de la logique métier nécessaire pour intégrer des systèmes d'information hétérogènes et distribués. Nous avons introduit, par la suite, les technologies (EAI, Services-web et ESB) basées sur les apports de XML (structuration et échange des données) et de workflow (gestion de processus). Les ESB reprennent les grands principes de l'EAI (intégration d'applications) mais évitent la centralisation de la logique d'intégration en se basant sur une architecture totalement distribuée. Enfin, le point commun entre ces technologies est qu'elles utilisent toutes une technologie de base : l'intergiciel (« *Middleware* »). Ce dernier est un intermédiaire de communication entre plusieurs applications, généralement complexes ou distribuées sur un réseau informatique [**Lopes 05**].

PARTIE III :

AXE CONCEPTION DIRIGEE PAR LES MODELES

4. Conception dirigée par les modèles

Dans la dernière partie de ce chapitre, nous abordons le contexte lié à la conception de la solution que nous proposons afin d'assurer l'interopérabilité des entreprises : le Système d'Information Collaboratif [30]. Le choix d'une approche d'ingénierie appropriée dans une démarche de conception d'un système d'information est évidemment prépondérant. La nature et la complexité de la solution (faire interopérer des systèmes d'information complexes, hétérogènes et répartis) nous obligent à suivre une démarche rigoureuse et structurée, prenant en compte la spécificité du contexte de développement.

Le choix de l'approche de développement basée sur les modèles MDA [**OMG 03**] s'est imposé rapidement : elle facilite la descente en abstraction et permet une séparation nette et délimitée entre le métier et la technique. Dans cette partie de manuscrit, nous discutons le choix de cette approche et ses avantages. Nous définissons ses principes de construction d'applications réparties et ses apports quant à l'interopérabilité des systèmes d'information. Nous montrons aussi les mécanismes de transformation de modèles qui relèvent de l'Ingénierie Dirigée par les Modèles (IDM). En effet, l'intérêt pour l'IDM a été fortement amplifié, lorsque l'OMG a rendu publique son initiative MDA (qui vise à la définition d'un cadre normatif pour l'IDM). Il existe cependant bien d'autres alternatives technologiques aux normes de l'OMG, par exemple Ecore[31] dans la sphère Eclipse. L'IDM dépasse largement MDA, pour se positionner plutôt à la confluence de différentes disciplines.

4.1. Pourquoi une approche de conception de SIC basée sur MDA ?

La conception des applications informatiques, de plus en plus complexes et consommatrices de ressources, a subi une grande mutation marquée par l'apparition de nombreux paradigmes différents. Le paradigme procédural a ainsi laissé la place au paradigme objet, puis au paradigme composant.

Cependant, ces évolutions, bien que majeures, ne permettent plus de faire face à la complexité croissante des systèmes logiciels développés. Les logiciels ne sont pas seulement complexes parce qu'ils doivent manipuler une grande quantité de données, mais aussi parce qu'ils doivent répondre parfaitement aux besoins spécifiques de leurs utilisateurs. De plus, dans un contexte de collaboration, cette complexité s'accroît du fait de l'hétérogénéité des partenaires. En conséquence, le développement de logiciels informatiques ne peut plus être fait en utilisant une démarche intuitive. D'après **[Lopes 05]**, « *Il devient largement recommandé d'utiliser des méthodologies différentes pour concevoir, analyser, modéliser et produire ces logiciels* ». Le génie logiciel a donc connu sa première vague de structuration de la conception d'applications. Plusieurs méthodologies ont été proposées pour supporter le développement logiciel dans les années 1980, telles que MERISE, OOD (*Object Oriented Design*), OMT (*Object Modeling Technique*) et OOSE (*Object Oriented Software Enginnering*) [**Morley** *et al.* **02**]. En 1997, ces méthodologies ont convergé vers le formalisme UML. Ce langage pseudo-formel est devenu une référence pour le développement orienté-objet d'un système. Cependant, son

[30] Le SIC est présenté dans le troisième chapitre de ce manuscrit.
[31] www.eclipse.org\emf\.

usage s'est principalement orienté vers la documentation des logiciels, délaissant quelque peu le domaine du « développement ».

L'approche MDA propose alors d'aller encore plus loin avec les modèles de type UML par la création et le maintien des logiciels à partir de ces modèles, avec moins d'efforts que dans le passé. Cependant, l'approche MDA n'est pas une rupture avec l'existant. **[Lopes 05]** affirme que MDA n'est pas une apparition d'idées ou de technologies nouvelles, mais la normalisation de fait, d'un certain nombre de notations consensuelles et de démarches caractérisant les bonnes pratiques du développement du logiciel en technologie des objets et des composants. En effet, MDA met en avant la mise en relation entre un modèle « *exploitable par l'homme* », qui décrit une certaine connaissance métier en capitalisant les besoins spécifiques, en lien avec le logiciel à développer et un modèle « *exploitable par l'informatique* », qui définit la spécification technique de logiciels (Fig II.22).

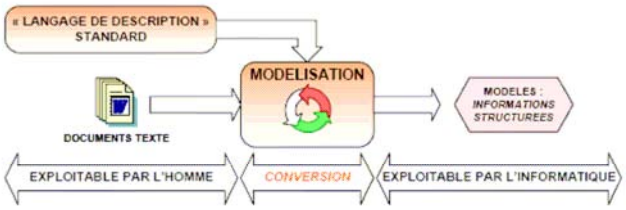


Fig. II. 22. *D'une connaissance exploitable par l'homme à une connaissance exploitable par l'informatique*

Dans **[Garcia 04]**, on énumère trois raisons pour le choix d'un développement de logiciel avec une approche MDA :

- « *tout d'abord pour séparer logique métier et technologies informatiques. Les éléments du métier sont stables et capitalisent le savoir-faire de l'entreprise. Il faut donc* ***modéliser les processus métiers*** *pour les rendre indépendants des technologies et aussi faciliter leur adaptation aux nouvelles technologies.*
- *Ensuite parce que la technologie des objets et des composants n'a pas tenu ses promesses en termes de simplification de conception. La complexité de ces technologies est croissante et fait régulièrement appel à des techniques (« design patterns », « frameworks »,etc.) pour y remédier.*
- *Enfin, il devient urgent de stopper l'empilement des technologies. Des systèmes hétéroclites, qu'il devient difficile de faire communiquer et qui nécessitent des compétences nombreuses et variées pour maintenir et faire vivre le système d'information* ».

En ce qui concerne notre problématique de conception de SIC et en se basant sur les raisons de **[Garcia 04]** citées ci-dessus, nous souhaitons évaluer les apports de l'approche MDA pour mener efficacement la démarche de définition du notre système.

4.2. Présentation de l'approche MDA

Le MDA est une approche présentée par l'OMG en novembre 2000. L'objectif à l'époque était ambitieux : avoir la possibilité de spécifier et de faire évoluer les modèles des applications au rythme requis par l'organisation et non pas par l'évolutions des plateformes technologiques. Une plate-forme est un ensemble de sous-systèmes et de technologies, qui fournit aux applications supportées un ensemble cohérent de fonctionnalités au travers

d'interfaces spécifiques. Les plate-formes peuvent supporter plusieurs technologies (CORBA, Java, .NET, etc.) et sont implémentées par plusieurs fournisseurs (Orbix, JDK, WebSphere, WebLogic, etc.).

D'une manière plus détaillée, MDA promeut une nouvelle voie pour le développement des applications, basée sur l'élaboration d'un modèle indépendant des plate-formes **(PIM - *Plateform Independent Model*)** à partir d'un modèle métier **(CIM - *Computation Independent Model*)** qui décrit le besoin métier de l'application à développer, et sur la transformation du PIM en un ou plusieurs modèles dépendants de plate-formes **(PSM - *Plateform Specific Model*)**, correspondant à chaque plateforme sur laquelle l'application va être déployée. D'après [**Bézivin** *et al.* **03a**] MDA fournit un processus de conception et met en œuvre des outils pour :

- spécifier un système indépendamment de la plate-forme qui le supporte, et donc réaliser un PIM,
- enrichir ce modèle par étapes successives,
- spécifier les plate-formes,
- choisir une plateforme particulière pour le système,
- transformer la spécification du système (PIM) en une autre spécification pour une plateforme particulière (PSM),
- transformer un CIM en un PIM et un PIM en un autre PIM,
- raffiner le PSM jusqu'à obtenir une implémentation exécutable.

4.3. Les types de modèle dans MDA

Dans le processus MDA, tout est considéré comme modèle, aussi bien les spécifications des applications que le code source ou le code binaire. Les quatre types de modèles identifiés sont donc les CIMs, les PIMs, les PDMs (ou PMs) et les PSMs [**Miller** *et al.* **03**] [**Bézivin** *et al.* **02**] :

- **CIM (*Computation Independent Model*)** : appelé aussi modèle de domaine ou modèle métier. Il montre le système dans l'environnement organisationnel dans lequel il va être exécuté. Son but est d'aider à la compréhension du problème. Les exigences exprimées dans le CIM doivent être traçables dans le PIM et le PSM. Comme exemple de CIM : un modèle de processus.
- **PIM (*Platform Independent Model*)** : sont des modèles qui n'ont pas de dépendance avec les plateformes techniques. Les PIMs représentent par exemple les différentes entités fonctionnelles d'un système avec leurs interactions, exprimées uniquement en termes de la logique d'entreprise. Ils représentent l'intérêt de l'application : la logique métier. Ces modèles sont mis au point par un architecte spécialisé dans le domaine de l'application. Comme exemple de PIM : un modèle d'architecture logique.
- **PDM (*Platform Description Model*)** ou **PM (*Platform Model*)** : il décrit la plate-forme sur laquelle le système va être exécuté (modèles de composants à différents niveaux d'abstraction : CCM, C#, EJB, EDOC, etc). Actuellement, il est souvent sous forme de manuels de logiciels et de matériels. Dans une démarche MDA, on se base sur les PDMs pour générer les PSMs à partir des PIMs.
- **PSM (*Platform Specific Model*)** : les PSM quant à eux sont dépendants de plate-formes techniques. Les PSM servent essentiellement de base à la génération de code exécutable vers ces mêmes plate-formes techniques. Comme exemple de PSM : un modèle de Services-web.

4.4. La transformation de modèles en MDA

La transformation de modèles est le processus qui transforme un modèle en un autre modèle du même système, mais à un niveau différent. Certaines étapes du processus MDA correspondent à la transformation d'un modèle en un autre modèle (de même type ou non), en utilisant éventuellement d'autres modèles. Plus précisément, un PIM suffisamment détaillé est projeté, par l'intermédiaire d'un PDM, vers un PSM. Quatre types de transformations différentes sont ainsi identifiés [**Miller** *et al.* **03**] (Fig II.23) :

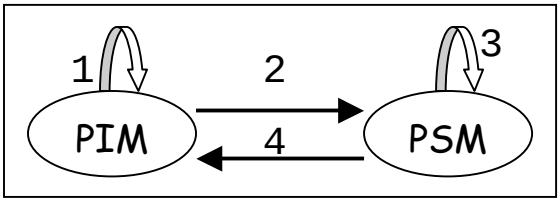


Fig. II. 23. Les transformations de modèles dans le processus MDA

4.4.1. Les transformations en MDA

1. *Transformation de PIM vers PIM* : ces transformations visent à enrichir, filtrer ou spécialiser le modèle sans utiliser d'informations dépendantes d'une plateforme. Les transformations PIM vers PIM sont généralement utilisées pour le raffinement de modèle et ne sont pas toujours automatisables.
2. *Transformation PIM vers PSM* : ces transformations sont effectuées lorsque les PIMs sont suffisamment raffinés pour pouvoir être projetés vers une plateforme d'exécution. Les caractéristiques de cette plateforme peuvent être décrites à l'aide d'un profil UML. L'opération qui consiste à ajouter des informations propres à une plateforme technique pour permettre la génération de code est une transformation PIM vers PSM. A l'heure actuelle, les plate-formes techniques visées sont : .Net, J2EE, XML et CORBA. Il apparaît clairement que ce sont les règles qui permettent ces transformations qui sont importantes et qui doivent être généralisées et capitalisées. Ces transformations peuvent donc être fortement automatisées.
3. *Transformation PSM vers PSM* : ces transformations s'appliquent sur un modèle spécifique et génèrent un autre modèle spécifique à la même plate-forme. En fait, une unique transformation PIM vers PSM n'est pas toujours suffisante pour permettre la génération de code, il faudra alors parfois transformer les PSM en PSM en utilisant des formalismes intermédiaires. Ces transformations sont donc effectuées pour la réalisation et le déploiement de composants.
4. *Transformation PSM vers PIM* : ces transformations permettent d'obtenir un PIM à partir d'une implantation existante sur une plateforme spécifique. Bien que celles-ci ne fassent pas directement partie du processus MDA, au sens où les spécifications OMG de ce processus ne les montrent pas explicitement, elles sont nécessaires à considérer dans toute stratégie de migration. Actuellement, ces transformations sont les plus difficiles à établir et à automatiser. Idéalement, le résultat de cette transformation devrait correspondre à la transformation inverse PIM vers PSM.

Nous pouvons aussi ajouter *la transformation de CIM vers PIM*. Cette transformation peut nécessiter **un enrichissement de modèle** CIM pour obtenir le modèle de PIM. En effet, nous pouvons dire que la distance entre CIM (exigences métiers) et PIM (exigences logiques de système) est plus grande que la distance entre PIM et PSM (exigences

logiques et techniques d'un même système), ce qui rend cette transformation plus compliquée à définir.

4.4.2. Notions de modèle et méta-modèle

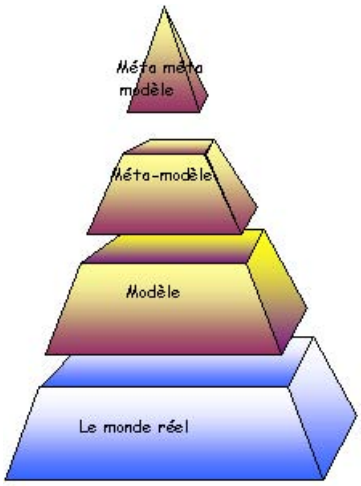


Fig. II. 24. Modèle et méta-modèle

D'après [**Terasse** *et al.* **03**], « *le modèle d'un système est la spécification formelle des fonctions, de la structure et / ou du comportement de ce système dans son environnement, dans un certain but* ». Un modèle est souvent représenté par des schémas et du texte. Le texte peut être exprimé dans un langage de modélisation ou en langage naturel.

Un méta-modèle d'un modèle est aussi un modèle qui décrit le modèle (Fig II.24). Il existe une relation bien précise entre les concepts de modèles et de méta-modèles. Chaque élément du modèle est une instance d'un élément de son méta-modèle. La relation d'instanciation implique que toutes les informations contenues dans une instance soient décrites dans l'élément correspondant du méta-modèle. Réciproquement, toutes les caractéristiques décrites dans l'élément d'un méta-modèle peuvent être instanciées par les éléments correspondants du modèle. Nous pouvons dire que :

- un méta-modèle correspond à la description d'un modèle,
- un modèle correspond à l'instanciation d'un méta-modèle.

4.4.3. Mise en œuvre d'une transformation

Nous distinguons trois étapes dans le processus de mise en œuvre d'une transformation MDA [**Bézivin** *et al*. **03**], [**Benguria** *et al*. **06**] :

1. la définition des règles de transformation,
2. le choix d'un outil de transformation,
3. l'exécution des règles de transformation.

1. la définition des règles de transformation

Etant donné un modèle source dans un langage $\mathbf{l_s}$ (par exemple BPMN), et un modèle cible dans un langage $\mathbf{l_c}$ (par exemple UML), il s'agit dans cette étape d'élaborer une mise en correspondance des concepts de $\mathbf{l_s}$ et ceux de $\mathbf{l_c}$ (par exemple une activité BPMN correspond à un cas d'utilisation en UML). Dès lors, on établit des correspondances sémantiques entre les deux méta-modèles. L'ensemble de ces correspondances va former une base de règles exhaustive et générique.

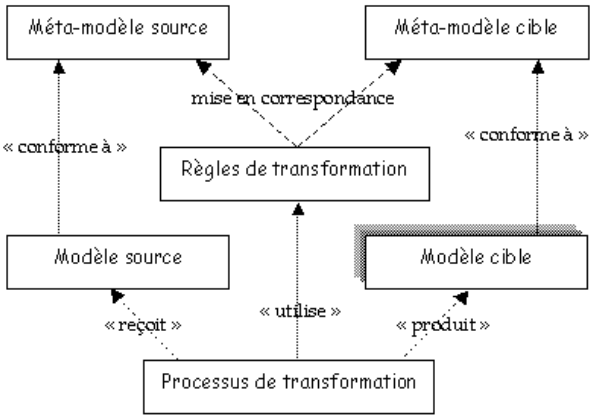


Fig. II. 25. *Architecture d'un système de transformation basé sur la méta-modélisation*

Les règles de transformation sont établies entre le méta-modèle source et le méta-modèle cible, c'est-à-dire entre l'ensemble des concepts du modèle source et celui du modèle cible. Le processus de transformation prend en entrée un modèle conforme au méta-modèle source et produit en sortie un (ou plusieurs) autre(s) modèle(s) conforme(s) au méta-modèle cible, en utilisant les règles préalablement établies (Fig II.25).

La figure suivante (Fig II.26) montre un exemple de transformation qui explicite le mécanisme de la transformation d'un diagramme de classe UML en des tables d'une base de données. La règle montre que l'élément *Table* de modèle cible est déduit à partir de l'élément *Class* de modèle source.

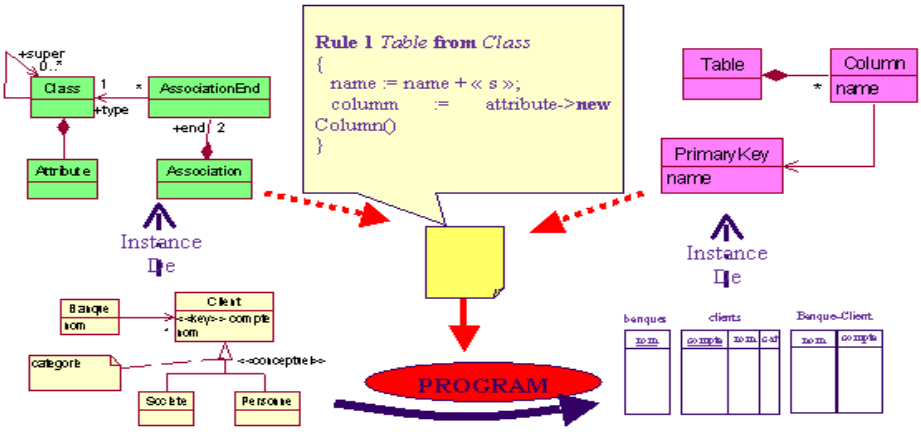


Fig. II. 26. *Principe du mécanisme de transformation de modèles* [32]

2. le choix d'un outil de transformation

Une fois les règles de transformation définies, il est nécessaire de disposer d'un langage de spécification de ces règles. Quelques langages en cours de standardisation existent. *ATL* (Atlas Transformation Language)) [**Jouault 06**], [**Bézivin** *et al*. **03**], *GreAT* [**Karsai** *et al.* **03**] et *AndroMDA* [**Andro 07**] sont des exemples. Nous nous intéressons particulièrement à ATL dans le cadre de nos travaux, car il répond à nos exigences en matière de simplicité de définition de règles et d'intégration de profil UML dans les règles. Un langage de transformation peut être déclaratif, impératif ou hybride :

[32] M. Belaunde - France Télécom R&D

- dans la programmation déclarative, on décrit d'une part les données du problème à traiter et d'autre part les contraintes sur ces données. Le programme s'exécute à partir de la situation courante décrite par les données, tout en respectant les contraintes.
- Par opposition à un programme déclaratif, un programme impératif décrit comment le résultat est obtenu en imposant une suite d'actions, que la machine doit effectuer.
- Un langage hybride regroupe à la fois les paradigmes de programmations déclarative et impérative. ATL est un langage hybride.

3. l'exécution des règles de transformation

Une fois spécifiées et exprimées, les règles requièrent le moteur d'exécution de l'outil de transformation choisi pour être exécutées. Ce moteur prend comme entrée le modèle et le méta-modèle source, le méta-modèle cible, ainsi que le modèle de transformation (les règles de transformation écrites dans le langage de transformation, basées sur les correspondances entre les deux méta-modèles source et cible) et son méta-modèle (représentant la grammaire du langage de transformation). Il produit en sortie le modèle cible. Ce mécanisme est décrit par la figure suivante (Fig II.27) :

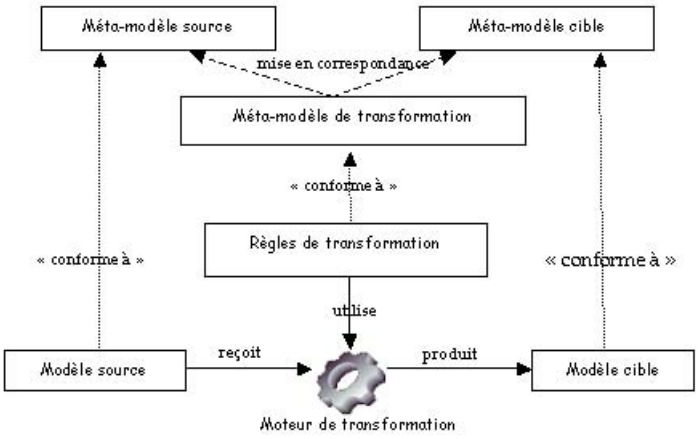


Fig. II. 27. Le moteur d'exécution des règles de transformation

4.5. Méthodologie de développement MDA

Une méthodologie de développement MDA est « *un processus de développement logiciel basé sur la transformation de modèles et utilisant l'architecture MDA* » [**Bézivin** *et al*. **03**] [**Bézivin** *et al*. **03a**]. Ce processus propose une structuration entre PIM et PSM en application du paradigme de séparation des préoccupations dissociant la logique métier et le code technique.

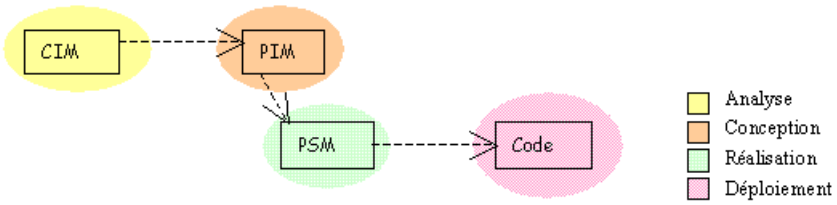


Fig. II. 28. De l'analyse au déploiement en MDA

Cette méthodologie est basée sur quatre étapes :

- l'**analyse** : elle exprime la logique métier à travers les CIM.
- La **conception** : elle révèle les caractéristiques fonctionnelles du système à l'aide des PIMs.
- La **réalisation** : elle décrit l'architecture technique et fonctionnelle du système par des PSMs.
- Le **déploiement** : il concerne la génération des codes exécutables et de certains artefacts (composants logiciels, documentation, etc.).

4.6. Model Driven Interoperability (MDI)

MDA vise à faire remonter le problème de l'interopérabilité des systèmes d'information au niveau des modèles, qui représentent ces systèmes. MDA propose des mécanismes de transformation, visant à faire interopérer des modèles de système d'information qui appartiennent aux différents stades de cycle du développement logiciel : CIM, PIM et PSM. MDI [**Athena 04**], [**Grangel** *et al.* **06**], [**Benguria** *et al.* **06**], [**Bourey** *et al.* **05**] propose de lier les trois niveaux conceptuels de MDA qui correspondent aux CIM, PIM et PSM. Le but de MDI est « *d'obtenir des applications informatiques interopérables et qui respectent les besoins spécifiques exprimés dans les niveaux conceptuels* » [**Grangel** *et al.* **06**]. Chaque niveau doit être identifié par un méta-modèle obéissant à une ontologie de référence. Le CIM est présenté comme un modèle qui collecte les besoins et les caractéristiques de l'entreprise en utilisant un modèle d'entreprise. Ce modèle permet de représenter et comprendre comment l'entreprise fonctionne, afin de capitaliser ses connaissances et son savoir-faire pour une utilisation ultérieure. Le PIM est un modèle obtenu à partir des modèles d'entreprise définis dans le CIM. Le PSM représente plusieurs modèles d'application suivant la technologie choisie. Afin de dépasser la difficulté inhérente à la mise en place de la méthodologie MDI, on propose de partir du niveau des modèles d'entreprises et d'utiliser des transformations horizontales pour l'interopérabilité au même niveau et des transformations verticales pour arriver au niveau de l'application informatique. Cette idée de base de MDI a été reprise par [**Roser** *et al.* **07**] pour illustrer le développement d'applications interopérables dans un contexte collaboratif.

La figure (Fig II.29) montre que les modèles des entreprises A et B souhaitant collaborer doivent être partagés à tous les niveaux de la démarche de développement. Au niveau CIM, l'entreprise A utilise un modèle d'entreprise ARIS, alors que l'entreprise B utilise au même niveau un modèle d'entreprise MO2GO. Des règles de transformation horizontales permettent la mise en commun des deux modèles au même niveau. Les règles de transformation verticales permettent par des étapes successives, de générer un modèle BPEL qui sert à implémenter la collaboration entre les SI des partenaires.

Puisqu'une démarche de conception d'un SI dans une approche orientée-modèles commence par la définition d'un CIM, il est alors légitime de s'interroger à propos de la connaissance nécessaire pour pouvoir définir précisément le CIM du SIC ? Et plus concrètement, quels éléments caractéristiques du réseau de partenaires doit-on rassembler afin de disposer du savoir nécessaire à la construction du CIM d'un SIC ?

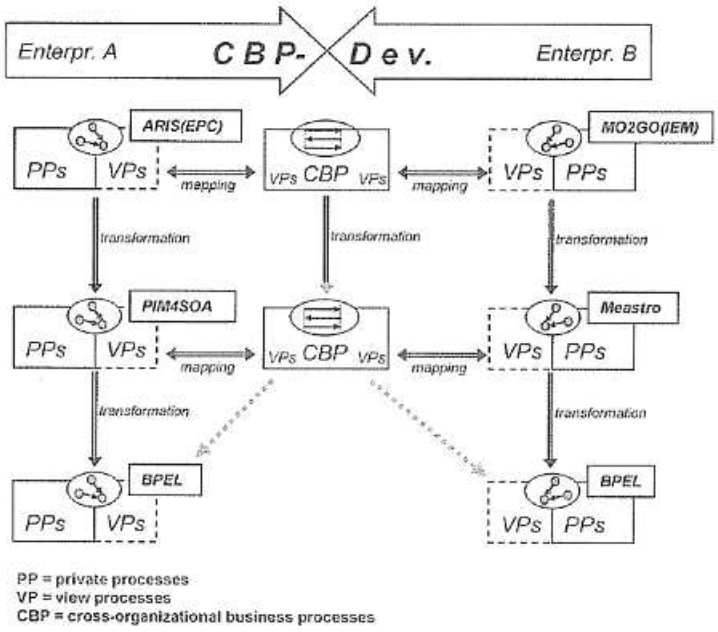


*Fig. II. 29. L'application d'une approche modèle pour l'interopérabilité des entreprises [**Roser** et al. **07**]*

Nous constatons qu'une démarche élaborée comme le MDI dépend fortement de la définition des besoins métiers (modèles d'entreprise). De plus, la relation modèle d'entreprise / système d'information doit être mieux étudiée. Nous proposons par la suite l'étude de quelques travaux qui se sont intéressés à l'utilisation des modèles d'entreprise pour la spécification d'applications informatiques.

4.7. Etude des travaux sur la correspondance modèle d'entreprise / modèle de système d'information

Dans cette section, nous présentons trois études qui illustrent l'utilisation de la modélisation d'entreprise comme moyen pertinent pour fournir la connaissance nécessaire à la conception de système d'information adapté aux spécificités de l'entreprise.

4.7.1. Cadre de référence pour un projet de conception d'ERP [Darras 04]

Dans ce travail, on définit un cadre de référence (Fig II.30) permettant principalement de positionner les modèles qui interviennent dans un projet de conception de système d'information (à savoir le modèle d'entreprise et le modèle de référence de la solution), mais aussi de positionner la solution logicielle à un niveau générique et le système d'information à un niveau particulier. Une mise en adéquation des modèles a été proposée en utilisant le langage UML pour modéliser les différentes vues de l'entreprise (ENV 40003). Utiliser un langage commun permet de mettre plus facilement en correspondance le modèle d'entreprise et le modèle de référence. Des mécanismes à développer dans ce cadre permettent la transition d'un modèle à un autre.

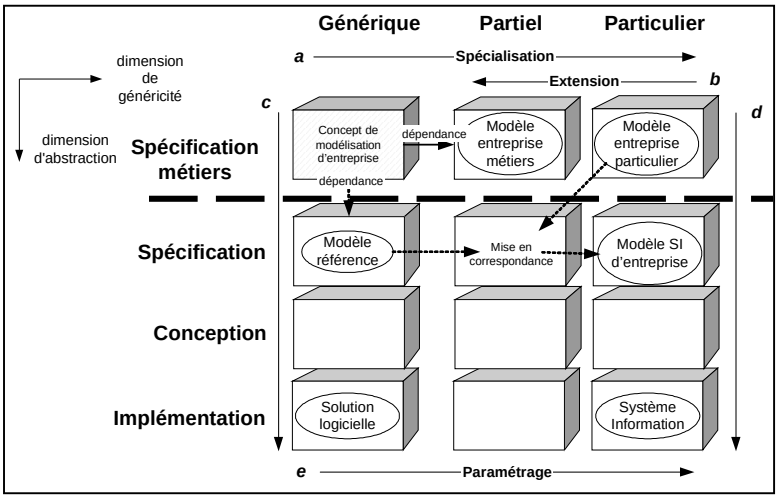


*Fig. II. 30. Cadre de référence pour un projet de conception d'ERP [**Darras 04**]*

Nous distinguons deux dimensions dans ce cadre : la dimension de généricité et la dimension d'abstraction (héritées de CIMOSA). La première dimension correspond au niveau de détail, auquel on associe le modèle. Il y a trois niveaux : *le niveau générique*, *le niveau partiel* et *le niveau particularisé*. Nous distinguons dans cette dimension deux mécanismes :

- *mécanisme de spécialisation*, qui vise la réduction du modèle de référence chez l'éditeur pour obtenir le modèle spécifique à l'entreprise cible.
- *Mécanisme d'extension*, qui permet d'étendre le modèle propre à l'entreprise vers le modèle de référence de l'application. Ce mécanisme permet de remédier aux problèmes des trous fonctionnels dans le modèle de l'entreprise.

La dimension d'abstraction fait référence à l'utilisation du modèle en introduisant quatre niveaux : *le modèle métier*, *le modèle de spécification*, *le modèle conceptuel* et *le modèle d'implémentation*. L'analogie avec les niveaux conceptuels de la méthodologie MDI est directe. Nous pouvons également nous déplacer dans cette dimension en définissant deux mécanismes d'implémentation, l'un de ces mécanismes permet de développer l'application logicielle de l'éditeur à partir de modèle de référence de ce dernier. L'autre mécanisme permet d'obtenir le système d'information de l'entreprise à partir de modèle d'entreprise relatif. Un mécanisme supplémentaire est ajouté à ce cadre pour permettre un paramétrage de la solution logicielle, afin de correspondre aux besoins particuliers de client. Plus précisément, ce travail ambitionne de définir un configurateur [**Dolidon** *et al.* **06**] basé sur les mécanismes cités ci-dessus et permettant de spécifier une solution ERP à partir d'un modèle d'entreprise décrivant le client.

4.7.2. Paramétrage d'un ERP en utilisant des modèles d'entreprise [Grangel *et al.* 06], [Bourey *et al.* 05]

Ce travail qui se situe, directement dans une approche MDI, vise à assurer une automatisation de paramétrage d'un ERP en utilisant des modèles d'entreprise (au niveau CIM). Les ERP sont toujours présentés comme des solutions standard qui ne répondent pas directement aux besoins spécifiques des clients. Le paramétrage d'un ERP consiste alors à aligner une solution générique avec les besoins spécifiques d'une entreprise dans le but d'améliorer sa performance. Nous retenons prioritairement de ce travail l'utilisation d'un ensemble de modèles d'entreprise (GRAI, IDEF, etc.) associé à un modèle de paramétrage

permettant d'identifier les paramètres spécifiques de client. Nous pouvons citer : les règles métier, la spécification des codifications des produits du client et des aspects techniques liés à la solution logicielle (Fig II.31).

L'apport de ce travail concerne entre autres l'identification claire du besoin de définir des méta-modèles (d'entreprise, de paramétrage et de solution informatique) qui respectent une ontologie de référence. Les auteurs identifient aussi le besoin de développer des mécanismes de transformation entre les différents méta-modèles et mettent l'accent sur la complexité de définition de tels mécanismes.

Dans un autre travail, connexe à ce sujet, les auteurs [**Grangel** *et al.* **07**] définissent des transformations au sens de MDA pour la génération d'un diagramme d'activités UML (PIM) à partir d'un actigramme GRAI (CIM). Les auteurs établissent des correspondances sémantiques entre le diagramme d'activité UML et l'actigramme GRAI.

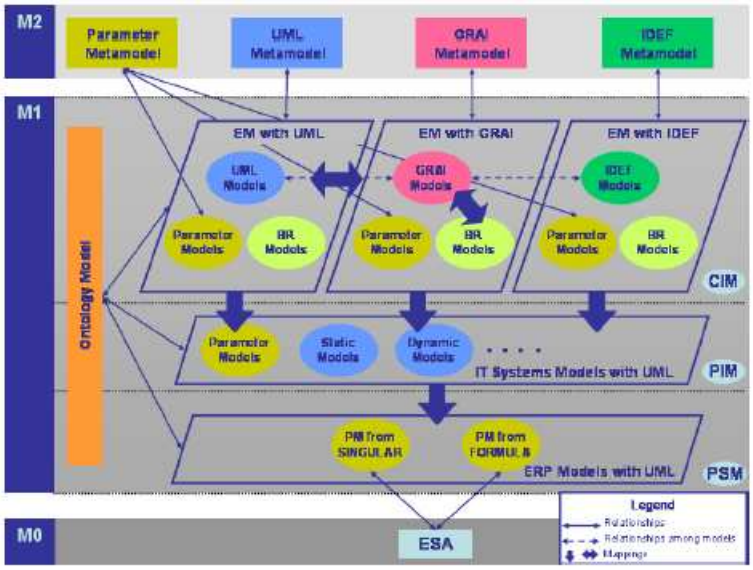


*Fig. II. 31. Paramétrage d'un ERP dans une approche MDI [**Grangel** et al. **06**]*

4.7.3. Introduction d'un choix architectural logique de SOA dans une démarche MDI [Benguria *et al.* 06]

Ce travail décrit une démarche qui permet d'aider à faire communiquer les analystes métier d'une entreprise avec les développeurs techniques dans le but de concevoir une solution informatique orientée services[33]. On définit notamment un méta-modèle appelé PIM4SOA (*Platform Independent Model For Service-Oriented Architecture*). Ce dernier est un méta-modèle qui facilite la génération d'un modèle de système d'information dans une approche orientée-services (SOA). La génération du modèle PIM4SOA présente une étape intermédiaire entre un modèle d'entreprise qui décrit la spécificité de l'organisation[34] et un modèle d'implémentation de la solution technologique (Fig II.32).

La différence majeure entre ce travail et les autres travaux cités précédemment est qu'il mentionne le besoin d'une connaissance additionnelle (qui peut être ajoutée d'une manière

[33] Les services Web présentent une plate-forme possible pour l'implémentation finale.

[34] Les auteurs utilisent le modèle POP*: Product, Organization, Process and Systems, un modèle qui décrit plusieurs points de vue de l'entreprise.

manuelle) pour mener à terme le processus de définition du modèle de SI. En effet, le modèle PIM4SOA généré par la transformation de modèle d'entreprise est incomplet, car il ne contient pas toutes les informations nécessaires pour générer le modèle spécifique à la plateforme technologique (Services-web). Par exemple, l'information sur la structure et les formats des messages échangés entre les services ne peut pas être définie à un niveau métier. Une intervention humaine est proposée par les auteurs. Une fois le modèle PIM4SOA complet, il est possible de générer un certain nombre de fichiers pour l'implémentation de la solution finale (fichier BPEL, fichier WSDL, fichier XSD, etc.)

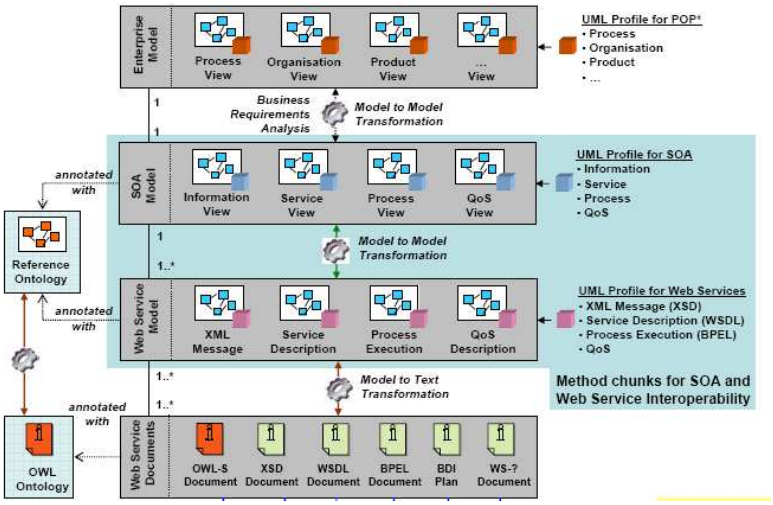


*Fig. II. 32. PIM4SOA pour le développement d'une solution SOA [**Benguria** et al. **06**]*

4.7.4. Synthèse

Les travaux que nous avons étudiés montrent que la modélisation d'entreprise présente incontestablement une matière exploitable pour la génération d'applications informatiques pour supporter les activités des entreprises. Nous identifions quelques critères principaux, sur lesquels nous pouvons classifier ces travaux :

- **définition du cadre de la démarche** : existe-t-il dans la démarche un cadre qui définisse bien les modèles et les transformations ?
- **Degrés d'utilisation d'UML dans la définition de modèle** : quel est le niveau d'utilisation d'UML pour définir les modèles de la démarche ?
- **Besoin d'une connaissance extérieure dans la démarche** : une connaissance extérieure est-elle citée dans la démarche ? pour quelle intervention humaine ?
- **Utilisation d'un modèle de paramétrage** : existe-t-il un modèle servant à paramétrer le résultat final ?
- **Définition des règles de traduction** : les règles de traductions sont-t-elles clairement détaillées ?
- **Définition des méta-modèles** : les méta-modèles sont-t-ils définis clairement ?
- **Implémentation d'un prototype** : y a-t-il un prototype qui implémente la démarche citée ?
- **Utilisation d'un choix logique de conception** : existe-t-il un choix logique de conception de SI (SOA, agents, etc.) ?
- **Utilisation d'un choix technique de conception** : existe-t-il un choix technique de conception de SI (web services, EAI, etc.) ?

- **Contrôle de résultat final** : existe-t-il des mécanismes de contrôle de la qualité des modèles générés ? Correspondent-ils vraiment aux besoins des utilisateurs ?

Critère	[**Darras 04**]	[**Grangel** *et al.* **06**]	[**Benguria** *et al.* **06**]
Définition du cadre de la démarche	Oui, cadre de référence inspiré de GERAM	MDI(CIM→PIM→PSM)	MDI(CIM→PIM→PSM)
Degrés d'utilisation d'UML	Utilisation d'UML pour les modèles d'entreprises et les modèles de SI	Des diagrammes GRAI sont aussi utilisés	Utilisation de profils UML pour les niveaux CIM, PIM et PSM
Besoin d'une connaissance extérieure	Pas évoquée	Pas évoquée	Oui, pour compléter le modèle PIM
Modèle de paramétrage	Non, mais des mécanismes de paramétrage	Oui, des modèles de paramétrage utilisés sur les trois niveaux	Non
Définition de règles de traduction	Non	GRAI→diagramme d'activité UML	Non, les règles ne sont pas développés
Définition de méta-modèles	Oui	Oui	Oui
Implémentation d'un prototype	Non	oui, GRAI→diagramme d'activité UML avec ATL	Oui, avec MTF
choix logique de conception	Non	Non	Oui (SOA)
choix technique de conception	Non	Non	Oui (Web services)
Contrôle de résultat final	Non	Non	Non

Tab II.2. Étude comparative des travaux sur la conception de SI à partir de modèles d'entreprise.

Les travaux de [**Darras 04**] et [**Grangel** *et al.* **06**] montrent en effet la possibilité d'utiliser UML pour cette tâche de modélisation de CIM. Cependant, UML et ses diagrammes restent prioritairement destinés à la modélisation d'un système d'information. Des profils UML sont apparus pour l'extension des capacités d'UML quant à la modélisation d'autres vues que celles d'un système d'information. Par exemple, le *UML Profile for Business Modeling*[35] (profile UML pour la modélisation métier), créé par Rational, permet de disposer de stéréotypes permettant la modélisation des processus métiers et d'autres concepts métiers de l'entreprise. Pour que cette adaptation soit partagée, il faut une documentation jointe aux profils UML. On peut donc se poser la question de la capacité d'UML à modéliser efficacement l'entreprise ? Si la réponse est positive, qu'en est-il de la qualité de cette modélisation, sachant que d'autres formalismes sont plus spécifiques aux vues de la modélisation d'entreprise : GRAI pour la vue décisionnelle, IDEF pour la vue fonctionnelle, BPMN pour la vue des processus, etc. ?

Enfin, nous notons que les travaux que nous avons consultés ne fournissement pas les règles de transformation en détail. La plupart de travaux, définissent uniquement les méta-modèles source et cible de leurs transformations, voire certains principes majeurs de transformation.

[35] http://www-128.ibm.com/developerworks/rational/library/4476.html

4.8. Conclusion

Nous avons montré que l'approche MDA permet de réduire la complexité de la conception de systèmes d'information en séparant les besoins métiers des spécifications techniques. La question à laquelle nous avons été confrontée est celle de la détermination de la connaissance métier nécessaire (au niveau CIM) pour la génération de modèle de système d'information (au niveau PIM). L'étude des travaux qui lient modèles d'entreprise et modèles de système d'information ont montré la faisabilité d'une telle approche en définissant des transformations de modèles entre niveaux CIM et PIM.

Dans un contexte collaboratif, la modélisation métier (*business modelling*) concerne tous les partenaires de la collaboration. Elle est donc importante en masse ; d'où la question : **de quelle connaissance a-t-on besoin exactement pour générer un modèle de SI** ? Nous apporterons des réponses à ces questions dans le chapitre IV de ce manuscrit, après avoir présenté en détail le concept de SIC (Chapitre III).

5. Conclusion du chapitre

Dans ce chapitre, nous avons présenté trois parties qui correspondent à une vision, que nous pensons globale et couvrante, des éléments de connaissance nécessaires pour l'étude du concept de SIC en tant que support de l'interopérabilité des systèmes d'information. Chaque partie fait référence à un élément principal de cette étude :

La première partie s'intéresse au concept de système d'information : le cœur de nos travaux. Plusieurs définitions positionnent ce concept. Nous avons choisi de nous focaliser sur la vision que nous pensons adaptée à notre problématique, qui représente un SI comme un ensemble d'applications, données et processus interagissant. La notion d'architecture de SI a également été présentée. En particulier, l'architecture SOA qui constitue une candidature légitime et crédible pour supporter l'intégration de systèmes d'information. Nous nous sommes également intéressés aux particularités de la structuration d'un SI dans un contexte collaboratif.

La deuxième partie de ce rapport s'est intéressée à présenter un panorama des solutions, et approches standards qui concernent les niveaux métier, sémantique et technique de l'interopérabilité. Le niveau métier concerne la mise en correspondance des organisations et la modélisation de leurs interactions et échanges. Nous avons conclu que les processus collaboratifs peuvent assurer cette tâche. Nous avons par ailleurs présenté une étude des formalismes de modélisation de processus et montré les évolutions constatées quant à la capacité de modélisation. Le niveau sémantique concerne le problème de la représentation et de la compréhension mutuelle des connaissances des entreprises partenaires. Ces entreprises utilisent des données, des applications et des processus incompatibles. Les ontologies associées à des mécanismes de mise en correspondance constituent une solution au problème de l'hétérogénéité sémantique des SI. Le niveau technologique concerne l'intégration des différentes plate-formes et solutions logicielles des partenaires d'une collaboration. Des solutions comme l'EAI, les Services-web et les ESB offrent des moyens reconnus pour ce type d'intégration.

Dans la troisième partie, nous nous sommes intéressés aux approches et démarches de conception de SI. Nous avons en particulier présenté le concept de MDA en détaillant les principes et les caractéristiques de cette approche de construction d'applications réparties et ses apports quant à l'interopérabilité des systèmes d'information. Des travaux montrent

la possibilité de profiter d'une capitalisation de connaissances de l'entreprise pour générer un modèle de SI adapté aux exigences spécifiques de cette dernière.

Notre challenge dans les chapitres suivants est de montrer que la conception d'un système d'information collaboratif, en charge de la gestion de l'interopérabilité des SI des partenaires, peut être réalisée en utilisant une approche d'ingénierie basée sur les modèles. Cette démarche devra permettre de lier les modèles de processus collaboratifs (description métier de la collaboration) à un paramétrage ou un développement d'une solution technologique (tel qu'un ESB) qui assure l'interopérabilité des systèmes d'information des partenaires. L'approche SOA nous permet de contourner les difficultés rencontrées pour connecter les applications des différents SI.

Chapitre III

Concept de Système d'Information Collaboratif : une médiation pour l'interopérabilité des systèmes d'information

1. Introduction du chapitre

L'objectif de nos travaux de thèse est d'accompagner (conceptuellement et techniquement) la transition qui amène les acteurs du domaine industriel à passer d'un *isolement compensé par l'ouverture technologique de leur système d'information à une intégration optimale de ces mêmes systèmes d'information en un système de systèmes agile et pertinent* (Fig III.1). Au sein de ce système de systèmes, les systèmes d'information collaborent d'une manière optimale, car ils sont interopérables. Cette interopérabilité considère plusieurs niveaux liés au concept même de système d'information.

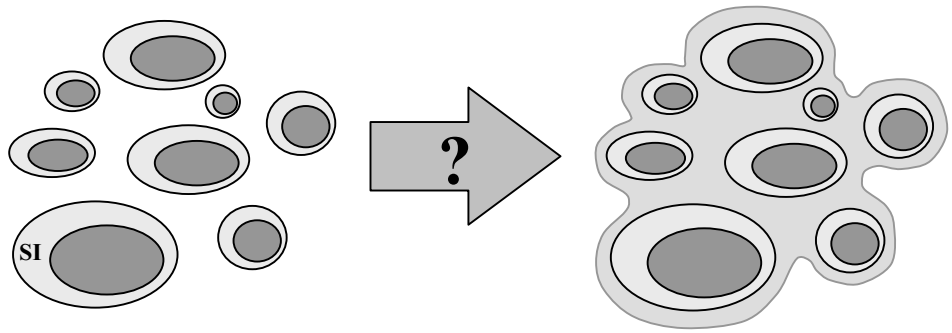


Fig. III.1. Objectif : des entreprises isolées mais connectables au système de systèmes

L'exemple de l'assemblée générale des Nations Unies, que nous avons évoqué dans le premier chapitre (Chapitre I, Section 3), montre un certain nombre d'exigences que nous estimons nécessaires pour assurer l'interopérabilité entre les différentes délégations (organisation des interventions, gestion des différences linguistiques, gestion des problèmes d'interprétation, etc.). Dans une première partie de ce chapitre, nous proposons de formuler une réponse adaptée à ces exigences. Cela nous permettra de disposer, dans un second temps, d'une base de réflexion pour explorer les approches de solution pour le problème de l'interopérabilité des SI. Nous discuterons ensuite de la notion de médiation avant de revenir sur le concept de SIC, sa vision SOA ainsi que l'illustration des principes évoqués au travers d'un exemple.

2. Réponse aux exigences d'intéropérabilité de l'exemple des Nations Unies

Afin de répondre aux exigences de l'interopérabilité décrites dans l'exemple de l'assemblé générale des Nations Unies (Chapitre I, Section 3), nous pouvons imaginer décrire le système collaboratif en place composé d'un ensemble d'équipes travaillant sous l'autorité du président de l'assemblée générale. Nous détaillons par la suite les caractéristiques de ce système.

2.1. Proposition d'une réponse aux exigences

Une réponse aux exigences de l'interopérabilité des délégations peut être basée sur :

- **un président de l'assemblée générale qui** s'occupe du pilotage du déroulement de l'assemblée générale. Il contrôle les interventions des délégations suivant un programme détaillé et établi à l'avance. Il s'intéresse

aussi au bon déroulement de l'assemblée : le respect du temps accordé pour les interventions, le comportement des délégués, la gestion des imprévus, etc.

- **Une équipe (cellule) de traducteurs expérimentés** qui veille à traduire les discours prononcés dans une langue particulière dans les autres langues des auditeurs. Par exemple, un discours prononcé en chinois se voit traduit en même temps en arabe, français, anglais, etc. Ce choix de traduction présente un avantage et un inconvénient : *chaque délégation peut ainsi s'exprimer dans sa propre langue*, peu importe qu'elle maîtrise ou non d'autres langues. En revanche, l'interprétation des autres délégations dépend fortement de la capacité des traducteurs à transmettre d'une manière fidèle le sens du discours. Même si ce choix de traduction s'avère coûteux (traducteurs, experts linguistiques, etc.), l'avantage de la possibilité d'expression dans sa langue constitue un atout considérable. Nous pouvons aussi discuter d'un autre choix de communication : *l'adoption d'une langue commune.* Ce choix permet à toutes les délégations de s'exprimer et de se comprendre, à condition qu'elles maîtrisent toutes cette langue commune : l'anglais par exemple. Cela nécessite un certain effort à fournir en matière d'apprentissage de cette langue par les délégations non anglophones(jusque dans ses moindres subtilités et nuances). L'autre inconvénient de ce choix est lié à l'affirmation de l'identité linguistique des pays participants.

- **Une équipe d'experts linguistiques** qui vérifie et contrôle le travail des traducteurs. L'interprétation d'une phrase traduite en une autre langue peut changer son sens. Ces experts doivent avoir une connaissance suffisante des langues traduites et des cultures associées et assurer une mise à jour continue du vocabulaire et des règles d'une langue particulière. Cette équipe veille finalement à s'assurer que la traduction d'un discours dans une langue particulière est fidèle dans son sens au discours d'origine.

- **Une équipe de sécurité et d'accréditation** qui contrôle la sécurité des lieux de l'assemblée générale, ainsi que l'accréditation des personnes autorisées à participer. Seules les personnes munies de badges officiels sont autorisées à accéder aux lieux de l'assemblé générale.

- **Une équipe d'experts en infrastructure et technologie de communication** qui permet d'assurer la connexion de divers matériels avec la plate-forme technologique de l'assemblée.

2.2. Proposition d'un système collaboratif

L'ensemble des équipes et unités citées dans la section précédente forme à notre avis un **système collaboratif**, chargé de faciliter ce que nous pouvons appeler l'« *interopérabilité des délégations* » (Fig III.2). Une des définitions de l'interopérabilité consiste à considérer une collaboration « sans effort particulier » de la part des partenaires. Les délégations doivent, dans ce cas, pouvoir communiquer ensemble en s'exprimant dans leurs propres langues, en ne s'intéressant pas à l'interprétation de leurs discours par les autres délégations, en utilisant leur propre matériel technique, etc. donc à moindre effort.

C'est le système collaboratif (Fig III.2) mis en place qui prend en charge ces spécificités liées à la nécessaire interopérabilité. La multitude des exigences des délégations nécessite la présence de plusieurs équipes, qui interviennent ensemble pour répondre aux besoins du groupe. Cela montre la complexité du problème de l'interopérabilité, qui doit être résolu sur plusieurs niveaux.

Notre cas d'étude se révèle être particulièrement parallèle au problème de l'interopérabilité de systèmes d'information hétérogènes dans un groupement d'entreprises qui collaborent. Par la suite, nous discutons des analogies entre cet exemple étudié et le monde industriel.

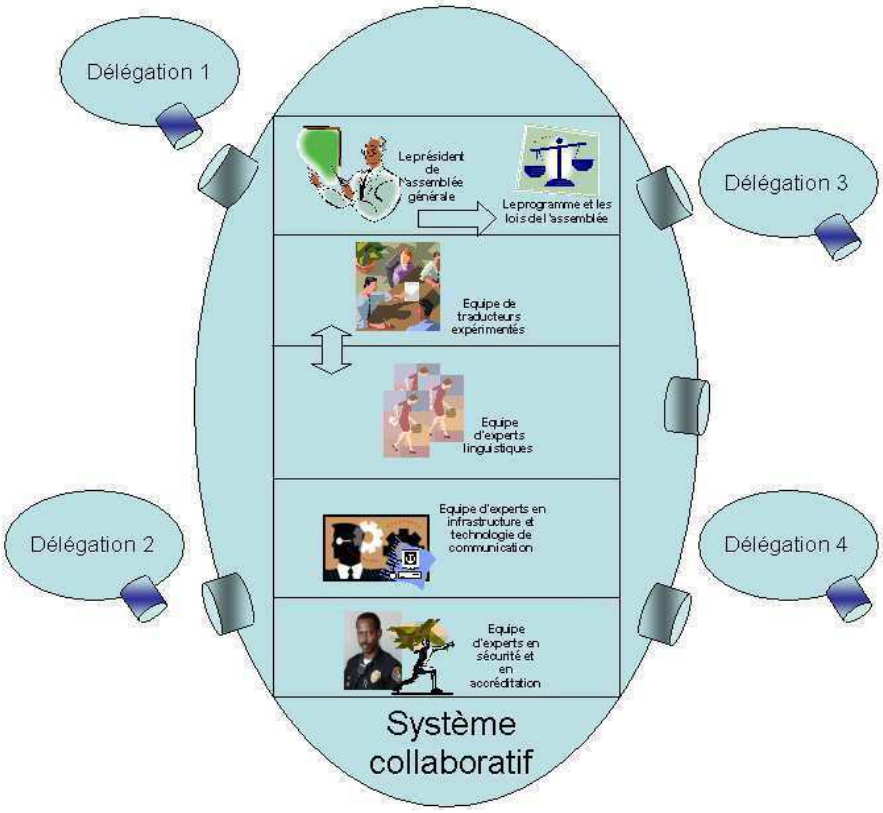


Fig. III.2. Un système collaboratif pour assurer l'interopérabilité des délégations

3. Carcactérisation des solutions d'intéropérabilité des systèmes d'information

L'exemple étudié exprime clairement un système évolué (assemblée générale) composé de sous-systèmes autonomes (délégations). L'ensemble des sous-systèmes vise à communiquer d'une manière efficace et fiable. L'obstacle de la langue est ici la composante majeure du problème d'hétérogénéité.

3.1. Projection du concept de système collaboratif dans le domaine des systèmes d'information

Dans cette partie, nous essayons de projeter la solution du système collaboratif pour l'interopérabilité des délégations dans le domaine de l'interopérabilité des systèmes d'information.

- **Gestion de l'hétérogénéité syntaxique** : les différences de langues entre les délégations peuvent être assimilées à des différences de formats de données, d'applications et de processus entre les partenaires d'un réseau collaboratif. Nous avons mentionné deux solutions possibles pour tenter de résoudre ce problème :
    - la solution adoptée dans notre exemple consiste à préserver les identités linguistiques des délégations en leur permettant de s'exprimer dans leur langue maternelle. Pour que cela devienne réalisable, des traductions d'une langue à une autre sont nécessaires. L'équipe de traducteurs multilingues doit travailler sur ces aspects. Par analogie avec le domaine industriel, cette solution correspond à un cas où les systèmes d'information gardent leur intégrité durant la collaboration et ne changent rien de leurs formats de données, de leurs applications et processus en place : ils communiquent d'une manière transparente avec leurs partenaires en dépit de leur hétérogénéité. Dans ce cas, un module de traduction des formats d'échange entre systèmes d'information hétérogènes est à définir. Ce module doit être hébergé par un système extérieur, qui connaît tous les formats de tous les SI des partenaires.
    - Une solution alternative consiste à adopter une langue commune et unifiée (l'anglais) avec les inconvénients que cette simplification entraîne. Par analogie avec notre problème industriel, les entreprises peuvent adopter un standard (ou plusieurs s'il y a plusieurs écosystèmes) commun(s) pour faire communiquer leurs systèmes d'information. Cela nécessite une adaptation des entreprises en intégrant le standard adopté avec leurs systèmes d'information. Cette approche d'intégration de systèmes d'information est relativement coûteuse pour l'entreprise (intégration et adaptation nécessaires). La mise en œuvre de cette approche prend beaucoup de temps pour harmoniser le standard adopté avec l'existant de l'entreprise. Comme nous l'avons montré dans le premier chapitre de ce manuscrit, les entreprises peuvent être amenées à établir des collaborations d'une manière rapide et occasionnelle (Chapitre I, Section 2). Dans ce cas, une solution optimale où l'entreprise fournit un effort minimum pour se rendre interopérable est capitale.

- **Gestion de l'hétérogénéité sémantique** : l'équipe de traducteurs linguistiques dans l'exemple de l'assemblée générale doit travailler avec une deuxième équipe, composée d'experts linguistiques. Cette dernière a la charge de contrôler le travail de traduction de l'équipe de traducteurs. Dans le domaine industriel, ceci s'apparente au problème sémantique d'échange d'information entre partenaires. La connaissance d'une langue correspond à la connaissance

d'un système d'information. Dans ce cas, l'équipe d'experts linguistiques correspond à un module de mise en correspondance (à jour) entre connaissances sur les systèmes d'information (ontologies), afin de vérifier la fiabilité des échanges entre ces derniers.

- **Gestion de processus collaboratif** : la notion de « processus collaboratif » est présente d'une manière simpliste dans notre exemple de l'assemblée générale. En effet, le président de l'assemblée a normalement une double fonction : la première est de donner la parole d'une manière séquentielle aux différentes délégations. La deuxième, d'exercer une certaine autorité dans l'assemblée générale : il peut modifier l'ordre des interventions, gérer des perturbations, décider de la fin de l'assemblée, etc. Cela relève de deux fonctions distinctes (ces deux rôles étant assumés par le même acteur dans notre exemple) : le pilotage (gestion) et la prise de décisions (autorité). Dans le cas du domaine industriel, le pilotage est l'affaire de la gestion de workflow (qui exécute le processus en orchestrant les activités collaboratives des partenaires) alors que l'autorité peut être répartie ou confiée à un ou plusieurs acteurs.
  Le président de l'assemblée s'appuie sur un ordre du jour et sur les règles de l'assemblée générale pour assurer le bon déroulement des séances. L'ordre du jour correspond dans ce cas à la définition du processus collaboratif. Chaque intervention d'une délégation correspond à une activité de processus. Le délégué correspond à l'acteur qui réalise l'activité. Le président (moteur du workflow) coordonne l'exécution des différentes activités (il est bien évident que le processus collaboratif peut être beaucoup plus compliqué qu'un simple séquencement). Les lois de l'assemblé générale correspondent au contrat de collaboration établi entre les partenaires, qui précise leurs droits et leurs devoirs. Par exemple, un temps est toujours accordé pour l'intervention d'un délégué. De même dans un processus collaboratif, il est important que les partenaires respectent les engagements pris quant au respect de contraintes temporelles.

- **Gestion de la sécurité des transactions** : la sécurité des transactions dans la collaboration est aussi une composante importante pour établir une interopérabilité fiable entre systèmes d'information. Dans l'exemple, l'équipe de sécurité et d'accréditation correspond à un module de contrôle des transactions entre partenaires de réseau collaboratif. En effet, nous pouvons comparer le lieu de l'assemblée générale à l'espace collaboratif (périmètre du système de systèmes d'information) des partenaires. Pour pouvoir entrer dans le lieu de l'assemblée, les membres des délégations doivent posséder des badges officiels. Dans le domaine industriel, cela s'apparente à une identification préalable, ainsi qu'une définition des droits d'accès à l'espace collaboratif, aux ressources partagées, etc.

- **Gestion de l'hétérogénéité technique** : les partenaires de la collaboration doivent aussi pouvoir faire interopérer leurs plate-formes technologiques. C'est un autre niveau de l'interopérabilité des systèmes d'information (niveau technique). Dans notre exemple, nous supposons que les délégations peuvent utiliser leur propre matériel technique. Dans le domaine industriel, cela correspond à une utilisation d'une solution d'intégration de technologies informatiques de type (EAI, ESB, etc.).

Le tableau suivant (Tab III.1) résume les analogies constatées entre l'exemple des nations unies et l'interopérabilité des systèmes d'information.

Interopérabilité	***Eléments de l'exemple étudié***	***Domaine industriel***
Bénéficiaire de l'interopérabilité	Délégation	Système d'information d'entreprise
Espace de l'interopérabilité	Assemblée générale	Espace collaboratif
Interopérabilité métier	Programme de l'assemblée générale	Définition du processus collaboratif
Interopérabilité métier	Les lois de l'assemblée générale	Les règles de contrat de collaboration entre partenaires
Interopérabilité syntaxique	Traducteur expérimenté	Module de traduction entre formats de données applications et processus
Interopérabilité sémantique	Expert linguistique	Module de mise en correspondance entre connaissances
Interopérabilité technique	Président de l'assemblée	Moteur workflow
Interopérabilité technique	Intervention d'une délégation	Exécution d'une activité de processus collaboratif
Interopérabilité technique	Expert en technologie et infrastructure de communication	EAI, ESB,etc.

Tab III.1 : Les analogies entre l'exemple étudié et le domaine industriel

3.2. Synthèse

Nous avons montré, en nous appuyant sur un exemple qui nous a servi de base de réflexion, qu'une réponse admissible à l'ensemble des exigences compliquées de l'interopérabilité correspond à un ensemble de fonctionnalités (gestion de l'hétérogénéité syntaxique, gestion de l'hétérogénéité sémantique, gestion de l'hétérogénéité technique, gestion de processus collaboratif et gestion de la sécurité des transactions). Cet ensemble forme un système collaboratif qui prend en charge le problème de l'interopérabilité en permettant de garder l'intégrité des systèmes pendant leur collaboration. Ce système collaboratif doit disposer d'informations sur tous les systèmes collaborant. Nous pouvons ajouter alors que ce système doit être un tiers de confiance : donc indépendant de tous les partenaires de la collaboration. L'approche par système intermédiaire, **le médiateur de la collaboration**, qui propose les fonctionnalités adaptées pour traiter les différents niveaux de l'interopérabilité des SI semble alors pertinente.

Ce système doit permettre de passer d'une addition de systèmes hétérogènes à une fusion de ces systèmes en un seul système virtuel. L'interopérabilité des systèmes d'information n'est acquise que si et seulement si elle prend en compte tous les aspects : métier, sémantique et technique (cf. chapitre I de ce manuscrit) liés aux différentes couches d'un système d'information : données, applications et processus.

Par la suite et avant de présenter les principes de notre solution d'interopérabilité des SI, nous citons quelques travaux sur la médiation des systèmes.

4. Travaux sur la médiation des systèmes

Dans le chapitre I (chapitre I, Section 5.1), nous avons rappelé que le terme médiation fait référence, dans le domaine juridique, à la résolution des conflits humains. Dans cette section, nous montrons deux approches de médiation qui ont marqué le domaine de l'intégration de systèmes d'information : les bases de données fédérées et les architectures de médiation.

4.1. Bases de données fédérées

L'approche des *bases de données fédérées* [**Heimbigner** *et al.* **85**], [**Sheth** *et al.* **90**] concerne plutôt une médiation de données. Cette approche permet de fournir aux utilisateurs une vue unique des données implémentées sur plusieurs bases de données locales *à priori* hétérogènes. Cette vue unique est représentée par un schéma global établi à partir de l'ensemble des schémas locaux. En effet, chaque schéma local se voit traduit dans un modèle de données commun (pivot, appelé aussi canonique) et intégré par la suite au sein du schéma global. Cependant, cette approche n'existe que théoriquement. Aucune réelle implémentation complète prenant en compte la lecture et l'écriture des données, n'a pu exister. Par exemple, si une base de données A contient une table *Employé* avec un attribut *ADRESSE* et une base de données B contient une table *Personnel* avec un attribut *EMAIL,* mais pas d'attribut *ADRESSE*. Une vision unifiée de ces deux bases de données suivant cette architecture doit contenir une table *Employé-Personnel* qui a pour attributs *ADRESSE* et *EMAIL*. Ainsi, chaque instance de A peut être vue dans cette table pivot avec un attribut *EMAIL* vide et chaque instance de B peut être vue dans cette table pivot avec un attribut *ADRESSE* vide.

4.2. Architectures de médiation

L'approche d'*architecture de médiation* [**Wiederhold 92**], [**Jouanot 00**] concerne une médiation syntaxique des schémas (intégration syntaxique) et une médiation sémantique des contextes (intégration sémantique).

Pour la médiation des schémas, le principe est le même que dans l'approche des bases de données fédérées : un schéma conceptuel global (ou schéma de médiation), auquel doivent s'apparier les différents schémas locaux (contextes locaux) : il y a donc intégration des schémas locaux au schéma global. La construction de ce dernier repose en général sur l'analyse préalable des schémas locaux et ceux-ci sont intégrés d'une manière statique au schéma global.

Concernant la médiation sémantique des contextes appartenant à des ontologies différentes, [**Jouanot 00**] précise qu'un contexte constitue *un élément d'information associé à un objet de système ou à une portion de l'objet et spécifie de façon explicite un ensemble de connaissances sur la structure, les valeurs ou les fonctionnalités de l'objet.* Ce type de médiation prend en charge la gestion des correspondances sémantiques entre différents concepts grâce aux mécanismes d'**unification** ou de **réconciliation** des contextes. L'unification des contextes peut avoir lieu quand les ontologies utilisent des contextes exprimés dans le même format.

Ce mécanisme utilise des règles de logique formelle pour manipuler les contextes. La réconciliation (ou aussi rapprochement sémantique) est utilisée surtout avec des ontologies non structurées, peu structurées ou structurées, mais aux formats disparates. Ce mécanisme est basé sur un calcul de la distance sémantique (appelée aussi affinité par certains auteurs) entre deux concepts. Ce calcul prend en compte plusieurs facteurs : ressemblance linguistique des termes, les taxonomies des termes, les différentes relations et propriétés des termes, etc.

5. Concept de Système d'Information Collaboratif

Notre proposition pour l'interopérabilité des systèmes d'information s'appuie sur le concept de ***système d'information collaboratif*** [**Touzi** *et al.* **06**], [**Benaben** *et al.* **06**], [**Touzi** *et al.* **06a**] et [**Touzi** *et al.* **07**]. Ce SIC constitue un système d'information inter-organisationnel susceptible de supporter les processus collaboratifs des partenaires. Le principe de cette option est de faire jouer au SIC le rôle de système intermédiaire assurant la cohésion entre les partenaires potentiels. Le SIC devient alors le garant de l'**interopérabilité** des SI des partenaires.

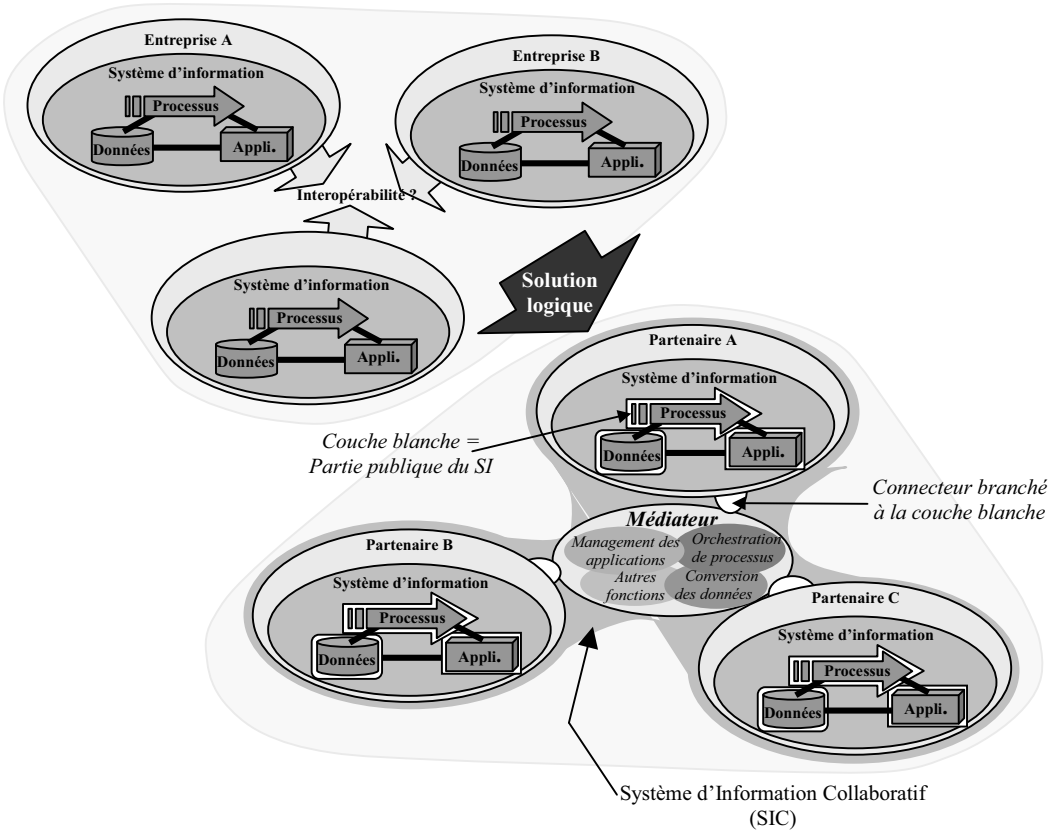


Fig. III.3. Le Système d'Information Collaboratif

Le SIC peut être vu plus largement comme le système qui inclut les parties publiques des systèmes d'information des entreprises (Chapitre II, Section 3.4.1) et le médiateur (Fig III.3). Le médiateur présente le cœur du SIC. Il permet de faire interopérer les différents systèmes d'information en utilisant des connecteurs directement liés à leurs parties publiques. Les connecteurs permettent d'assurer les échanges d'information entre le médiateur et les parties publiques des SI. Enfin, le

médiateur intervient sur les trois couches : données, applications et processus d'un système d'information. Il gère les différents niveaux de l'interopérabilité des SI que nous avons identifiés : métier, sémantique et technique. *Alimenté d'une définition de processus collaboratif, le médiateur assure une exécution de ce processus en échangeant les différentes données entre SI et en se connectant à leurs différentes applications.* Cela nous rappelle la définition de l'interopérabilité de [**Chen** *et al.* **03**] : « *c'est l'interaction sur les trois niveaux : données, applications et processus, en accord avec une sémantique métier convenue entre les partenaires* ».

5.1. Architecture de base du Système d'information Collaboratif

Le SIC est composé d'un ensemble de modules interagissants (Section 3). Chaque module s'occupe d'un niveau de problème de l'interopérabilité des systèmes d'information. Si nous pouvons stigmatiser l'intervention du SIC sur les trois couches d'un système d'information : processus, applications et données, il existe également d'autres modules qui traitent des aspects secondaires, mais complémentaires, pour optimiser la qualité de l'interopérabilité entre SI (sécurité des transactions, gestion des droits, etc.). Par la suite, nous détaillons les choix que nous avons faits pour gérer les différentes niveaux identifiés de l'interopérabilité.

5.1.1. Gestion de processus collaboratif

La plupart des collaborations établies dans le domaine industriel peuvent être cadrées par des processus collaboratifs (Chapitre II, Section 4.1.2). Un processus collaboratif présente les coordinations, interactions et échanges entre les différents partenaires. Il définit clairement les acteurs impliqués, les données et les fonctionnalités à mettre en œuvre dans la collaboration. Certaines recherches actuelles s'occupent de la **flexibilité des processus métiers** en la présentant comme un critère primordial pour l'agilité des échanges dans un environnement industriel en perpétuelle évolution [**Bouzguenda 06**], [**Nurcan 07**]. Dans nos travaux, nous nous concentrons sur la gestion de processus collaboratifs **stabilisés**. Ce choix de collaboration rigide semble justifiée, à notre avis, par le fait qu'une collaboration demeure aujourd'hui (sans que ce constat soit pour autant voué à demeurer valable très longtemps) une certaine prise de risque pour les partenaires qui n'ont pas d'autres moyens concrets de se rassurer que de s'appuyer sur une définition ferme de la dynamique de la collaboration (qui va faire quoi ?). Néanmoins, notons également que la « réputation » des partenaires et l'enjeu financier constituent également des critères de persuasion conséquents..

Le Système d'Information Collaboratif doit gérer l'exécution du processus collaboratif en synchronisant ce dernier avec les processus internes des partenaires. Une spécificité de notre vision du processus collaboratif réside dans le fait que le SIC lui-même peut participer à la collaboration en proposant des services métiers. Ces services de support ou à valeur ajoutée participent pleinement aux objectifs du processus global et justifient également le choix d'un système médiateur (porteur de ces compléments, tels que la mise à disposition d'un algorithme de choix performant ou le contrôle des opérations de paiement électronique, etc.).

La figure suivante (Fig III.4) montre la structure d'un processus collaboratif (en BPMN) en positionnant le SIC comme acteur médiateur de la collaboration. C'est l'acteur principal de ce processus (une *pool* lui est réservée). Ce système gère la

collaboration et véhicule tous les échanges entre partenaires. Cependant, ce système doit travailler d'une manière transparente, de sorte que les partenaires ne se rendent pas compte de son existence.

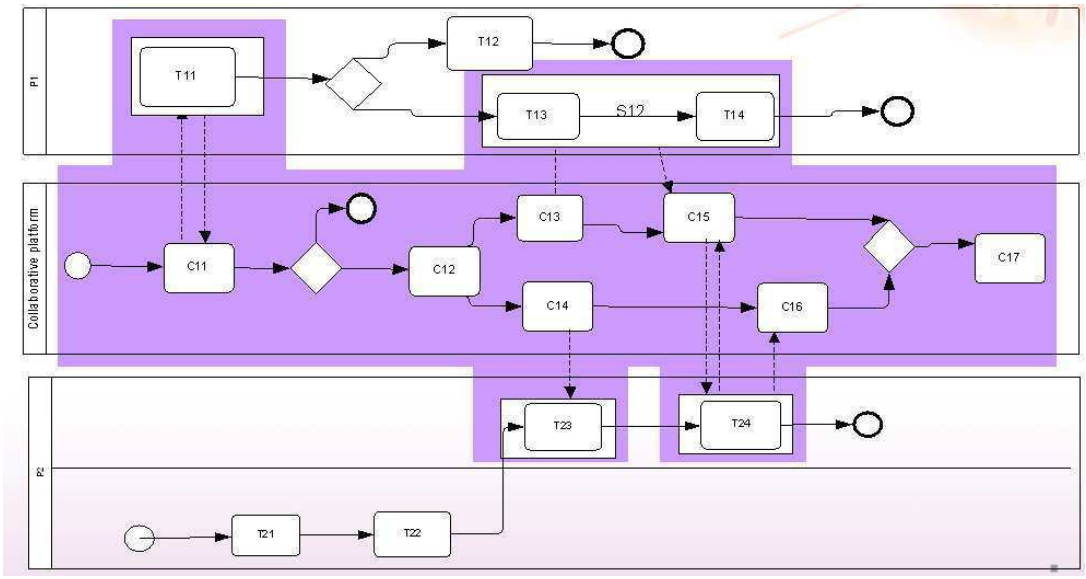


Fig. III.4. La portée de processus collaboratif exécuté par le SIC

On peut constater sur la figure précédente que les interactions entre deux activités du SIC (*pool* collaborative platform), représentées par des *sequence flow* (flèches pleines) diffèrent de celles entre activité du SIC et activité d'un partenaire, représentées par des *message flow* (flèches pointillées). Les processus situés dans les pools des partenaires forment les processus internes des partenaires. Ces processus contiennent naturellement des activités qui sont directement impliquées dans la collaboration (liées par un message flow au pool de SIC). D'où une question légitime sur le périmètre du processus collaboratif : Jusqu'à quel niveau (de granularité) doit-on descendre (chez les partenaires) pour délimiter la couverture exacte du processus collaboratif ? Nous proposons par la suite d'étudier deux choix :

- le premier choix appréhende la collaboration dans sa globalité. Nous considérons le processus collaboratif comme l'union des processus métiers internes des entreprises avec la partie de synchronisation du SIC. C'est un choix qui permet au SIC de disposer des détails de l'exécution des processus métier internes des partenaires (*le comment*). Cependant, cela nécessite une intégration des processus métiers internes des partenaires avec le processus global de collaboration. Dans [**Vanderhaeghen** *et al.* **04**], les auteurs considèrent que ce genre d'intégration pose deux problèmes : l'un syntaxique et l'autre sémantique. En effet, les partenaires utilisent des formalismes et des outils différents pour modéliser leurs processus. Les auteurs proposent des mécanismes de traduction entre les différents formalismes en identifiant les différentes correspondances sémantiques entre leurs éléments. La figure (Fig III.5) montre un aperçu des correspondances entre les formalismes BPMN et EPC. L'implémentation de ces traductions se fait en utilisant le langage XSLT (eXtensible Stylesheet Language Transformation), après avoir transformé les représentations graphiques des modèles en XML.

EPC type	uni- /bi-directional relation	BPMN type
function	←→	activity
aggregated function	←→	subprocess
event	←(→)	start event
event	←(→)	intermediate event
...	...	...

Fig. III.5. Aperçu des règles de correspondance entre des modèles EPC et BPMN ***[Vanderhaeghen*** et al. ***04]***

- Le deuxième choix consiste à réduire les processus métiers internes des partenaires aux seules activités représentant des fonctionnalités « parties prenantes » du processus collaboratif (visibles depuis le SIC : par exemple, qui reçoivent et / ou fournissent des données au médiateur). Dans ce cas, les processus internes sont représentés selon une approche « boîte noire ». C'est une forme d'application de l'approche orientée services (SOA) afin de monter en abstraction et de représenter les processus internes des SI en termes de fonctionnalités basiques (activité collaborative). Dans la figure (Fig III.4), la portée du processus collaboratif suivant ce choix est représentée par la partie encadrée, avec une zone délimitée grisée.

Pour la suite de nos travaux, nous adoptons le deuxième choix de délimitation du processus collaboratif. Nous constatons que du point de vue de la collaboration, les entreprises sont, en effet, libres de gérer comme elles veulent leurs processus internes. Ce qui compte, du point de vue collaboratif, ce sont les interfaces métiers qui caractérisent l'entreprise dans le domaine de la collaboration. Cette approche facilite la gestion des processus collaboratifs en se focalisant sur l'orchestration des activités fournies par le SIC et leur synchronisation avec les activités des partenaires. Nous montrerons que l'approche SOA et la technologie des services-web permettent de faciliter la mise en œuvre de cette approche.

Le processus collaboratif réagit aux évènements envoyés par les différents systèmes d'information des partenaires et à des évènements internes gérés par le SIC : *réception d'un message*, *délais de temps dépassé*, etc. Suite à ces évènements, le processus collaboratif s'exécute en appelant et en communiquant avec les différentes applications des entreprises. Cette communication avec les applications soulève le problème de l'interopérabilité au niveau technique.

5.1.2. Gestion de l'hétérogénéité syntaxique

L'interaction des systèmes d'information nécessite de gérer efficacement les échanges de données entre les applications hétérogènes de ces derniers. La difficulté provient essentiellement de l'incompatibilité des **schémas de données** des différentes applications. Un schéma de données décrit la structure (format) d'un **objet métier** manipulé par une application. Un objet métier est un concept utilisé par un acteur d'une entreprise (partie prenante interne), qui décrit son métier. Nous pouvons citer les exemples d'objets métiers suivants : facture, bon de livraison, compte bancaire, client etc. Les entreprises décrivent et structurent leurs objets métier d'une manière différente. *Ainsi, deux entreprises peuvent parler par exemple d'une même facture, alors que cette dernière est représentée d'une manière différente (deux schémas). Une fois le lien sémantique établi entre les deux, un mécanisme de*

traduction doit faire correspondre syntaxiquement les deux schémas de la facture. D'une manière générale, pour remédier au problème de l'hétérogénéité syntaxique de l'information, des mécanismes de traduction entre schémas de données doivent être mis en œuvre. Nous distinguons deux approches possibles :

5.1.2.1. Une approche de traduction de données point à point

Dans cette approche (Fig III.6), les traductions sont directement établies entre deux schémas. Pour lier deux schémas (Sh1, Sh2), deux traductions *T12* (de schéma 1 au schéma 2) et *T21* (de schéma 2 au schéma 1) effectuent les correspondances syntaxiques.

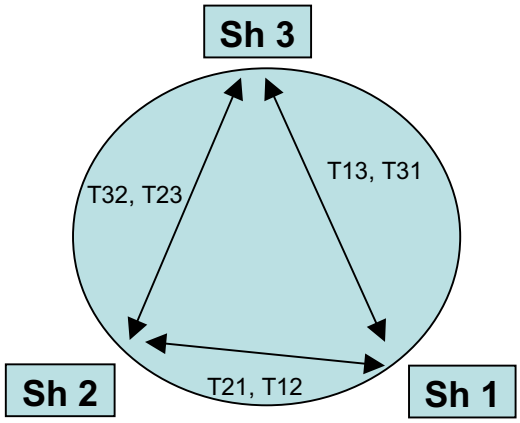


Fig. III.6. Une traduction des données par une approche point à point

Le principe de cette approche est que pour **n** schémas il doit exister **n*(n-1)** mécanismes de traduction.

5.1.2.2. Une approche de traduction de données par médiation

Cette approche [1] définit une médiation entre schémas en s'appuyant sur un schéma global (Fig III.7). C'est un schéma intermédiaire de données, unifié et commun, sur lequel se mettent d'accord tous les partenaires de la collaboration. Un partenaire peut ainsi fournir facilement au médiateur les mécanismes de traduction : de ses schémas locaux vers le schéma global et du schéma global vers ses schémas locaux.

Pour traduire un schéma de données en un autre schéma de données, il faut d'abord traduire le schéma source vers le schéma global (pivot) et ensuite traduire à nouveau ce schéma global vers le schéma cible. En résumé, pour chaque schéma local (*Sh n*) il existe deux mécanismes de traduction depuis et vers le schéma global (*Sh g*) : *Tng* (du schéma local au schéma global) et *Tgn* (du schéma global au schéma local). Plusieurs offres technologiques d'intégration de données utilisent cette architecture de médiation de schémas. Nous citons entre autres l'EAI et l'ESB. Le principe de cette approche est que pour **n** schémas il doit exister **2*n** mécanismes de traduction.

[1] Médiation des schémas

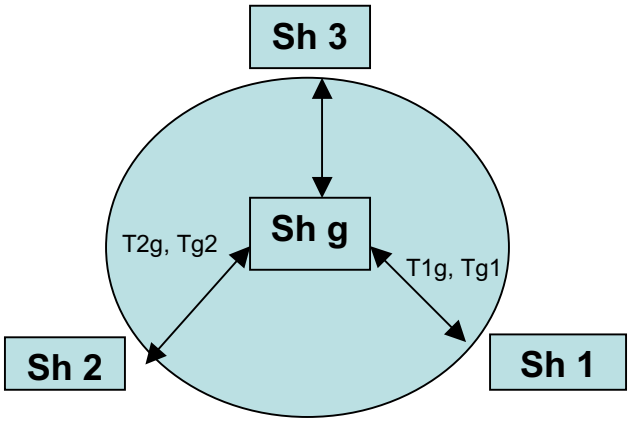


Fig. III.7. Une traduction des données par une approche de médiation

5.1.2.3. Synthèse et choix d'une approche pour le SIC

Nous avons présenté ci-dessus deux approches d'intégration syntaxique des données d'entreprises. Notre choix est d'adopter l'approche par médiation, car avec un nombre important de partenaires, la première approche devient compliquée à mettre en œuvre : chaque entreprise doit établir ses mécanismes de traduction avec tous les partenaires. Dans la seconde approche, l'entreprise s'occupe uniquement des mécanismes de traduction vis-à-vis du schéma global de la collaboration.

L'interopérabilité syntaxique est suffisante pour l'intégration des données des entreprises partenaires d'un réseau collaboratif **à condition que ce réseau soit stable.** Un réseau est considéré stable quand *le nombre de partenaires est fixe dans la collaboration et que chaque partenaire définit d'une manière unique les informations, applications et processus qu'il met en jeu dans la collaboration* [**Wiederhold 92**]. Dans ce cas, les correspondances sémantiques peuvent être établies directement (manuellement) au moment de l'établissement de la collaboration. Le système d'information collaboratif n'a qu'à gérer les transformations syntaxiques entre les ressources des partenaires, qui sont déjà bien identifiées. Le succès des solutions EAI à faire interopérer les applications des entreprises sans présenter de composante sémantique [**Blanc 04**], [**Hostachy 00**] et la montée en puissance des solutions ESB (qui, pour la plupart des cas, ne proposent pas d'intégration sémantique) confirment notre constat.

5.1.3. Gestion de l'hétérogénéité sémantique

L'approche sémantique est principalement utilisée pour ajouter du dynamisme à la découverte (non établie à l'avance) des différentes ressources des SI dans un réseau collaboratif. Cette approche permet également aux entreprises une gestion dynamique et flexible de leurs ressources dans la collaboration. Nous avons montré dans le chapitre précédent qu'une ontologie permet de structurer la connaissance d'un système d'information (Chapitre II, Section 4.2.2). Nous avons aussi discuté (Chapitre II, Section 4.2.3) de différentes approches, destinées à mettre en correspondance ces ontologies : l'approche mono-ontologie, l'approche multi-ontologies et l'approche hybride. Nous optons pour l'approche hybride comme solution pour interconnecter les différentes ontologies des partenaires (Fig III.8) : l'approche mono-ontologie est simple à mettre en œuvre, mais il est difficile de

l'adopter du fait de la difficulté d'amener les partenaires à un consensus (grandes différences sémantiques entre les systèmes). L'approche multi-ontologies permet une flexibilité et une indépendance des entreprises pour créer leurs propres ontologies, mais il est compliqué de mettre en correspondance par la suite les différentes ontologies. L'approche hybride combine alors les avantages des deux autres approches en permettant une flexibilité et une simplicité de mise en œuvre [**Wache 01**], [**Izza 06**]. Des mécanismes de *mapping* permettent la mise en correspondance des différents éléments des ontologies des partenaires (Fig III.8) .

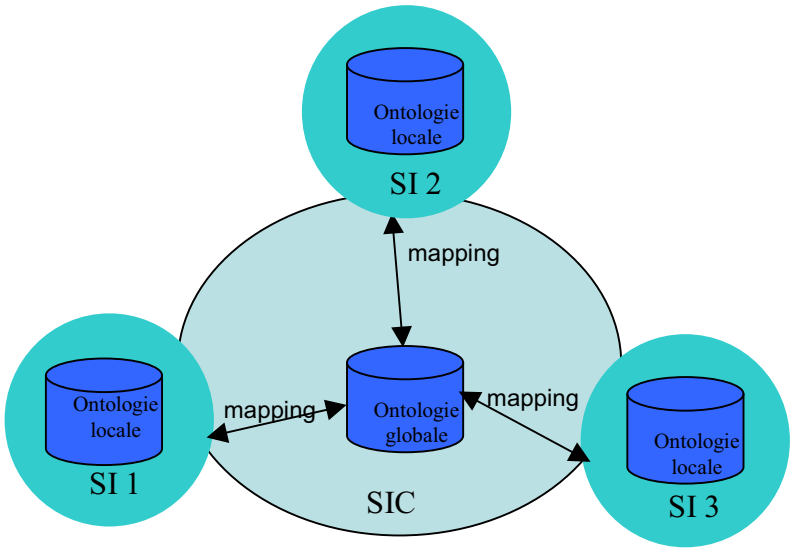


Fig. III.8. Une approche ontologique pour le Système d'Information Collaboratif

5.1.4. Gestion de l'hétérogénéité technique

L'hétérogénéité technique des systèmes d'information concerne la différence entre les matériels, les infrastructures réseaux, les systèmes d'exploitation et les plateformes d'applications des entreprises. De plus, nombreuses sont les entreprises qui possèdent des applications propriétaires (*legacy system*). Une application propriétaire est prisonnière de l'environnement technologique dans lequel elle a été développée [**Sneed 06**]. Cependant, les applications qui sont basées en partie sur un SGBD peuvent communiquer par le biais des données échangées. Les applications qui proposent des traitements sont les plus délicates à faire communiquer. Dans l'approche technologique EAI, on développe des connecteurs spéciaux pour chaque environnement. On ainsi parle de connecteur Visual Basic, connecteur J2EE, connecteur .NET, etc. [**Blanc 04**].

La technologie des services-web permet une encapsulation des applications en dépit de leur hétérogénéité technique. Les services-web interagissent ensemble en échangeant des messages. L'utilisation des services-web permet de s'appuyer sur le fait qu'ils soient basés sur des standards XML, SOAP, etc. (Chapitre II, Section 4.3.4) pour établir facilement des interactions efficaces entre les services. Les entreprises trouvent ainsi un moyen efficaces d'améliorer leur interopérabilité. Cependant Le problème engendré concerne la relation de ces entreprises avec leur existant. Un des grands défis industriels et académiques est de gérer la restructuration d'un SI classique en un ensemble de services-web [**Sneed 06**].

5.2. Synthèse

Nous avons montré, dans cette section de chapitre, les différentes couches de base qui composent le médiateur, solution de l'interopérabilité des SI (Fig III.9). Nous avons détaillé les mécanismes et les solutions (liés à ces couches) à mettre en place afin que le SIC puisse gérer l'interopérabilité à tous les niveaux évoqués. Cependant, la connexion des systèmes d'information au SIC et l'accès de ce dernier aux ressources des SI des partenaires reste une tâche délicate.

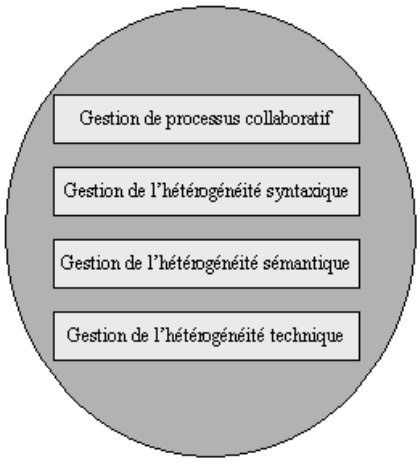


Fig. III.9. Couches de base de l'architecture de SIC

Nous avons montré, dans le deuxième chapitre de ce manuscrit (Chapitre II, Section 3.3.2), que l'architecture orientée-services facilite l'intégration de systèmes d'information hétérogènes. Les trois apports que nous avons identifiés (réduction de la complexité, agilité et accessibilité d'un système d'information) sont de nature à réduire la difficulté de communiquer entre systèmes d'information. SOA est en passe de devenir le standard universel de la conception de système d'information. Cette approche présente aujourd'hui un moyen efficace et fiable afin de contourner les problèmes de l'interopérabilité des systèmes d'information [**Touzi** *et al.* **06**, [**Vernadat 06**], [**Lopes 05**]. L'idée sous-jacente est de cesser de construire « la vie de l'entreprise » autour d'applications isolées pour faire en sorte de construire une architecture logicielle globale décomposée en services correspondant aux fonctionnalités métiers de l'entreprise. L'adoption presque « de facto » de cette architecture orientée services dans la conception des systèmes d'information actuels, a influencé la définition même de l'interopérabilité de systèmes. D'après [**Pokarev** *et al.* **06**] : « *c'est la capacité des différents systèmes à utiliser efficacement leurs services, à pouvoir s'échanger des messages, les analyser et les interpréter correctement* ».

Par la suite, nous voulons profiter des apports constatés de SOA afin de faciliter l'accès aux systèmes d'information des partenaires et faciliter également leur interopérabilité avec le médiateur. Dans ce cas, une interopérabilité réussie des SI des partenaires avec le SIC engendre naturellement une interopérabilité réussie entre ces mêmes systèmes d'information. Nous présentons une vision SOA de notre solution de l'interopérabilité et nous montrons un exemple concret de collaboration gérée suivant cette vision.

6. Une version orientée services (SOA) du système d'information collaboratif

L'objectif d'une architecture orientée services est principalement de décomposer un système d'information en un ensemble de fonctions basiques (appelées « services »), fournies par des composants logiciels et par la même occasion de décrire finement le schéma d'interaction entre ces services. Dans ce sens, le SIC est perçu comme une solution collaborative qui implémente les concepts SOA.

Dans cette vision SOA, le SIC peut alors être considéré comme composé du médiateur (qui fournit les différents modules pour l'intéropérabilité) et des différents services (publiés) des partenaires. La figure suivante (Fig III.10) montre une description de la collaboration en utilisant le SIC : un ensemble de **partenaires** de la collaboration (se présentant chacun comme un ensemble de services) utilise le **médiateur** pour interopérer.

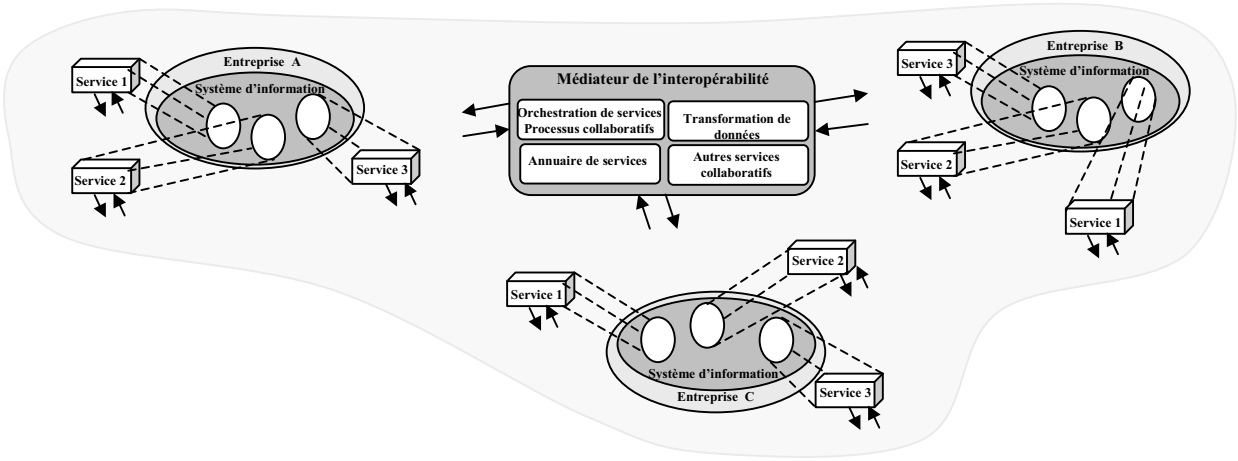


Fig. III.10. Une version SOA du Système d'Information Collaboratif

Ce système de médiation permet de répondre aux besoins suivants :

- comment interconnecter les services des partenaires ?
- comment orchestrer l'exécution de ces services ? Suivant quel ordre ?
- quel service répond parfaitement aux besoins d'un partenaire ?
- comment gérer la sécurité des transactions entre les différents services ?

Par la suite, nous détaillerons l'architecture de l'utilisation du SIC en version SOA.

6.1. Définition du Système d'Information Collaboratif

Le système d'information collaboratif peut être décrit comme un ensemble de modules (fonctionnalités) regroupés pour faciliter l'interopérabilité entre les services des partenaires (Fig III.11). Ces modules reprennent les concepts de gestion de l'interopérabilité (détaillés dans ce chapitre). C'est une autre vision d'un même système tout en conservant le même principe : « *assurer la médiation pour permettre l'interopérabilité* ». Nous distinguons trois modules de base liés à la gestion du

triplet : données, services et processus, auxquels s'ajoute un quatrième correspondant aux services dits « collaboratifs » apportés par le SIC.

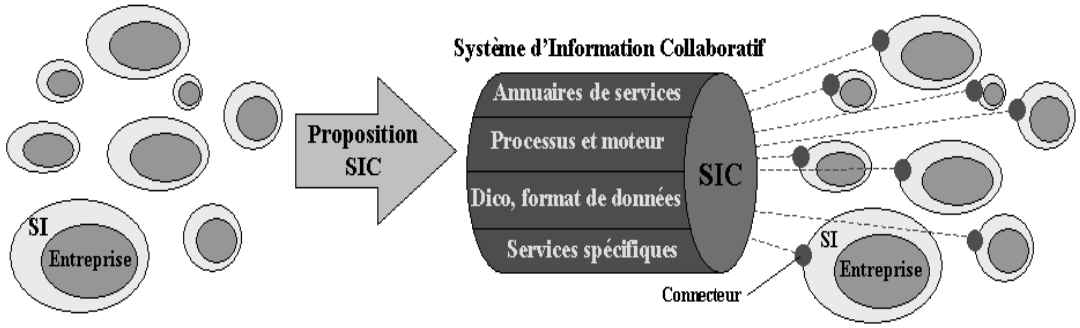


Fig. III.11. Le concept du SIC comme support du système de systèmes

6.1.1. Les modules de base du SIC

6.1.1.1. Module de gestion des messages échangés

Les services communiquent en échangeant des messages. Ces derniers peuvent être des objets métiers : facture, ordre de mission, etc. Les messages des différents services ne sont pas structurés de la même manière. Un mécanisme de transformation de formats est géré par le SIC pour faire communiquer les services. De plus, chaque entreprise partenaire doit pouvoir décrire ses services en définissant pour chacun la structure des messages qui lui sont liés (d'invocation et de réponse). Dans une technologie web-services, le principal format de description des messages de services est XSD, normalisé par le W3C.

6.1.1.2. Module de gestion des services des partenaires

Le SIC gère l'annuaire des services des partenaires (afin de pouvoir y accéder). Cette gestion consiste en deux mécanismes : la *publication* et la *découverte* des services. Concernant la publication, le partenaire publie les informations relatives à l'accès aux services (protocole, adresse, etc.), la structure des messages liés au services et une brève description des rôles des services dans la collaboration. Le mécanisme de découverte recouvre la possibilité de rechercher un service parmi ceux qui ont été publiés. Le principal standard utilisé est UDDI, normalisé par OASIS. Ce module gère une autre contrainte : les services doivent être facilement accessibles pour les partenaires (par le biais du SIC). On parle d' « invocation » de service. Le principal protocole utilisé est SOAP.

6.1.1.3. Module de gestion de processus collaboratif

Le SIC gère l'exécution d'un processus collaboratif qui décrit le schéma d'interaction entre les services des différents partenaires. Dans une technologie web-services, le langage BPE permet de fournir une définition exécutable d'un modèle graphique de processus. Un moteur BPEL (*BPEL engine*) permet d'orchestrer les différents services selon la définition du processus.

6.1.2. Les modules optionnels du SIC

Aux modules de base du SIC dans sa version SOA, nous pouvons également ajouter un ensemble de modules propres au SIC et lui permettant d'améliorer la qualité de la

collaboration. Ces modules peuvent être perçus comme des services que le SIC apporte à la collaboration. Nous distinguons deux catégories de services (ou modules) : les modules techniques (nous pouvons également les appeler modules *supports*) et les modules métiers (ou services à valeur ajoutée). La différence entre ces deux catégories est directement liée au processus collaboratif des partenaires. Les modules techniques interviennent en arrière plan pour traiter des aspects techniques à la collaboration. L'intervention de ces modules n'est pas forcément représentée dans le modèle de processus collaboratif. Les modules métiers interviennent en proposant des services à rendre aux partenaires, dont seule une partie tierce de confiance peut s'occuper. L'intervention de ces modules est représentée dans le processus collaboratif par des « activités » spéciales, ainsi le SIC peut synchroniser l'intervention de son module métier avec les autres services des partenaires.

6.1.2.1. Les modules techniques (supports)

Le SIC peut proposer un ensemble de modules techniques suivant les demandes des partenaires. Ces modules servent à améliorer la qualité de l'interopérabilité entre les SI. Nous présentons un petit aperçu de ces modules techniques :

- un **module de sécurisation des échanges** entre les différents services. Il gère et contrôle les accès aux différents services des partenaires. Ce service gère un annuaire des collaborateurs, qui spécifie les droits d'accès et les autorisations.
- Un **module de gestion des transactions** gère les différents états du système, suite à des échanges entre services. Dans le cas d'échec d'un échange entre services, le SIC qui a sauvegardé l'état du système avant l'opération va pouvoir restaurer l'état initial.
- Un **module de gestion des performance** des échanges entre services. Il effectue une gestion des statistiques des échanges entre les différents services des partenaires. Ainsi, il peut gérer certains indicateurs de performance qui aident les entreprises à améliorer la qualité de leurs échanges.

6.1.2.2. Les modules métiers

Le SIC gère une bibliothèque de modules métiers génériques. Nous les appelons génériques, car ils peuvent être utilisés dans un bon nombre de cas de processus collaboratif. Nous présentons un petit aperçu de ces modules métiers :

- un module **de sélection de fournisseurs** pour des plate-formes d'achats groupés. Il permet l'évaluation de la requête d'un client et le choix d'un fournisseur adapté aux critères du client (prix, date de livraison, nature de la commande, pérennité, confiance, etc.).
- Un module **de gestion de paiement,** qui peut gérer des opérations de paiement bancaire entre un ensemble de partenaires.

6.2. Définition des partenaires de la collaboration

Un partenaire de la collaboration, comme nous l'avons déjà expliqué, est considéré comme un ensemble de services variés, proposés par l'entreprise dans la collaboration. Les services encapsulent les applications ou les processus internes d'un partenaire. Les relations entre applications, processus (internes et collaboratifs)

et services de l'entreprise sont très dépendantes (Fig III.12). Nous détaillerons les relations qui peuvent exister entre ces éléments.

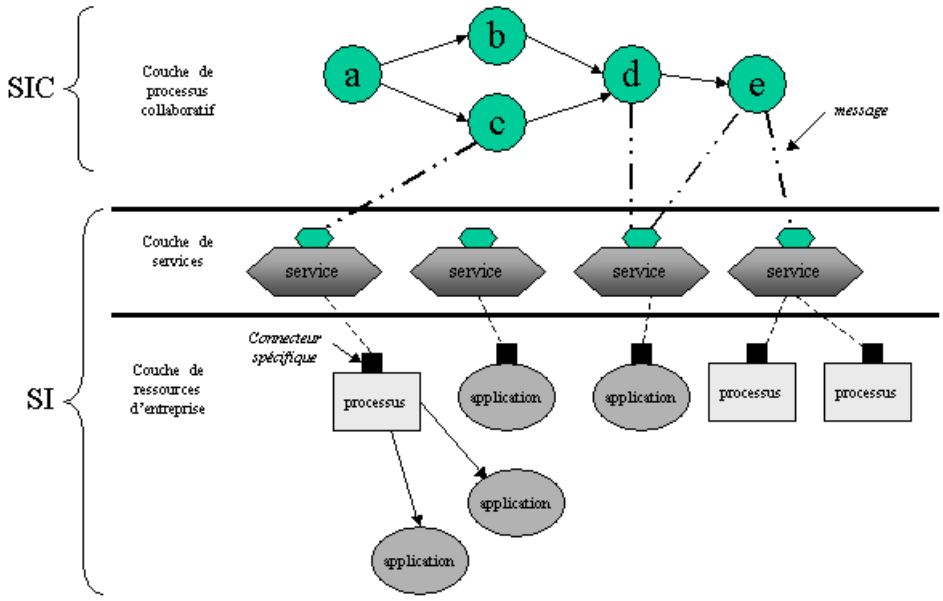


Fig. III.12. Architecture SOA d'un système d'information partenaire de la collaboration

- **Relation [service, processus interne]** : un service peut encapsuler un processus interne de l'entreprise ou même une coordination de plusieurs processus. Par exemple, un processus de réservation d'un vol (d'une compagnie aérienne) peut être perçu comme un service de réservation de vol du point de vue du SIC.
- **Relation [service, processus collaboratif]** : un processus collaboratif fait appel à un ensemble de services des partenaires pour son exécution. Par exemple, un processus Client-Fournisseur fait appel aux services de vente du fournisseurs et aux services d'achat des client.
- **Relation [application, processus interne]** : un processus de l'entreprise peut faire collaborer plusieurs applications de l'entreprise. Par exemple un processus d'achat de matériel peut faire appel à une application de facturation.
- **Relation [application, service]** : un service peut encapsuler directement une application de l'entreprise. Par exemple, une application de facturation peut être perçue comme un service de facturation, du point de vue du SIC.

Pour qu'un partenaire réussisse sa collaboration en utilisant ses services, ces derniers doivent être indépendants les uns des autres afin de garantir leur réutilisabilité et leur atomisation au sein du processus collaboratif.

7. Exemple de processus collaboratif géré par le SIC dans sa version SOA

Dans cette section, nous présentons un exemple de processus collaboratif afin de mettre en évidence la capacité du SIC à gérer l'interopérabilité entre les partenaires, acteurs de ce processus.

7.1. Présentation de processus collaboratif

Le processus collaboratif que nous étudions est une version évoluée du processus présenté dans le chapitre II (Chapitre II, Section 4.1.4.2.c) lors de la discussion sur les capacités de modélisation du langage BPMN. Le SIC permet de gérer une plateforme d'aide à l'achat à laquelle se connectent divers fournisseurs potentiels. Le clients envoie ses demandes d'achat au SIC et ce dernier est chargé de le mettre en contact avec le(s) fournisseur(s) qui répond(ent) le mieux à sa demande.

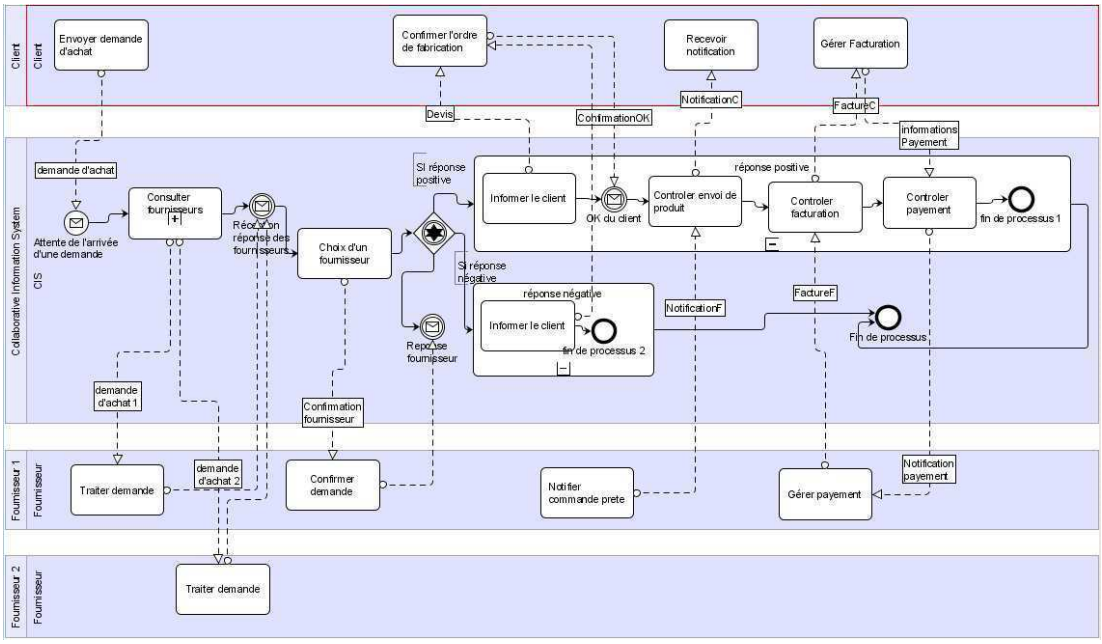


Fig. III.13. Exemple de processus collaboratif de la plate-forme d'achats groupés

La figure précédente (Fig III.13) présente un modèle BPMN qui exprime ce processus de collaboration entre un client et un ensemble de fournisseurs. Le client envoie une demande d'achat au SIC. Cette demande décrit :

- les articles demandés,
- la quantité,
- une fourchette de prix,
- une date de livraison souhaitée.

7.2. Gestion de processus collaboratif

Le SIC analyse cette demande et contacte les fournisseurs susceptibles de répondre à sa requête (activité « *consulter fournisseur* »). Chacun des fournisseurs répond au SIC en proposant une offre (devis) quant aux prix et délais nécessaires pour la fabrication. Le SIC, par la suite, fait intervenir le service à valeur ajoutée qui permet de sélectionner parmi les offres des fournisseurs (activité « *choix d'un fournisseur* »). Ce service du SIC choisit l'offre qui répond le mieux à la demande du client (selon des critères propres qui font justement toute la valeur ajoutée de ce service). Une fois

que le fournisseur a confirmé son acceptation de fabrication, le SIC doit gérer la livraison (activité « *contrôler l'envoi de produit* »), la facturation (activité « *contrôler facturation* ») et le paiement de la commande (activité « *contrôler paiement* »).

7.3. Gestion des services

Les services des partenaires doivent être publiés dans l'annuaire géré par le SIC, afin que ce dernier puisse y accéder. Un service est appelé, une fois que l'activité qui le présente dans le processus est exécutée. Nous pouvons représenter les services de ce processus sous ce format :

messages de sortie (type de message) = Serv n (messages d'entrée (type de message))
avec n un numéro de service dans le processus.

- **Les services du client :**
  - **un service de confirmation de fabrication de la commande** : le SIC utilise ce service pour tenir le client informer du résultat de sa recherche de fournisseur qui réponde à son besoin. Si tel est le cas, le devis du fournisseur est transmis au client. Le SIC attend une réponse positive de ce service pour demander au fournisseur de commencer la production.
    - ***rep (OUI | NON)***
      = Serv1 (FournTrouvé (OUI | NON), DevisFournisseur (Devis))
  - **un service de Facturation :** c'est un service classique de gestion de factures reçues. Ce service permet de répondre à une facture reçue en envoyant au SIC les coordonnées bancaires nécessaires pour le paiement du fournisseur.
    - ***coord (CoordPayement)***
      = Serv2 (Facture (Facture Client))
- **Les services des fournisseurs :**
  - **un service de traitement de demande client :** ce service reçoit la demande d'achat du client et répond par un devis (prix, délais de fabrication, etc.).
    - ***DevisFournisseur (Devis)***
      = Serv3 (demande (demandeclient))
  - **un service de confirmation de demande client :** c'est un simple service de confirmation au fournisseur (sélectionné pour la production de la commande de client).
    - ***rep1 (OUI | NON)***
      = Serv4 (rep2 (OUI))

7.4. Gestion des messages

Le SIC gère la traduction syntaxique entre différents formats d'objets métiers utilisés dans le processus : demande d'achat, facture, paiement, etc. Nous pouvons voir dans notre exemple BPMN les objets *FactureC* (facture au format client) et *FactureF*

(facture au format fournisseur). Ces deux objets se réfèrent sémantiquement à une même entité, mais sont représentés différemment (schémas de données différents). Le SIC, en utilisant un schéma pivot, peut ainsi stocker les différentes instances des deux objets dans le processus. Par la suite, il pourra envoyer au format voulu l'objet à son destinataire (client ou fournisseur).

8. Conclusion du chapitre

Nous avons montré par un exemple (*assemblée générale des Nations Unies*) que l'interopérabilité d'un ensemble de systèmes hétérogènes et autonomes doit être traitée sur plusieurs niveaux. Cela implique qu'une solution collaborative puisse disposer d'un ensemble de modules qui travaillent ensemble pour offrir cette capacité d'interopérabilité aux différents éléments demandeurs. Dans ce cas, chaque module traite un niveau particulier de l'interopérabilité. Le Système d'Information Collaboratif, support de l'interopérabilité des systèmes d'information hétérogènes, respecte ce principe. Il est composé d'un ensemble de modules qui permettent de gérer les niveaux métier, sémantique et technique. L'approche SOA permet de faciliter l'interopérabilité des systèmes d'information. Nous avons proposé d'adopter cette approche, dans un contexte où les systèmes d'information des partenaires peuvent être représentés par des services.

Pour que le SIC puisse fonctionner, il doit s'appuyer sur un modèle de système d'information qui détaille les données, les services et les processus de la collaboration. Nous ambitionnons dans le chapitre suivant d'étudier la génération automatiques par déduction d'un tel modèle, à partir du modèle du processus collaboratif mis en jeu par la collaboration.

Chapitre IV

Aide à la conception de système d'information collaboratif orienté-services

Introduction du chapitre

Nous avons présenté dans le chapitre précédent le concept de Système d'Information Collaboratif (SIC) en tant que support de l'interopérabilité des systèmes d'information. Dans ce chapitre, nous nous focalisons sur la conception de ce système. L'intérêt principal de ce chapitre réside dans le fait qu'il détaille une approche d'ingénierie qui permet, à partir d'une connaissance métier sur la collaboration (processus BPMN), de déduire la spécification d'un modèle de SIC respectant un choix architectural orienté services (SOA). Avant de présenter les détails de notre approche (en particulier les règles formelles de traduction), nous discutons des correspondances entre un modèle de processus en BPMN (niveau métier) et les diagrammes UML (niveau SI).

1. Positionnement des processus au sein de l'entreprise et du système d'information

Dans cette partie, nous montrons la place que les processus occupent dans la modélisation d'entreprise d'une part et dans la conception de système d'information d'autre part. Nous présentons deux approches, qui illustrent l'intérêt apporté aux processus métiers pour la conception de système d'information : la démarche d'urbanisation de SI et la démarche Business Process Management (BPM).

1.1.1. La place des processus dans la modélisation d'entreprise

Il est démesuré d'affirmer que toutes les vues de la modélisation d'entreprise ont le même poids dans une démarche de conception de système d'information. F. Vernadat [**Vernadat 96**], [**Vernadat 97**], [**Vernadat 99**] positionne la vue fonctionnelle (qui fait directement référence aux processus) au cœur des autres vues d'entreprise (Fig IV.1).

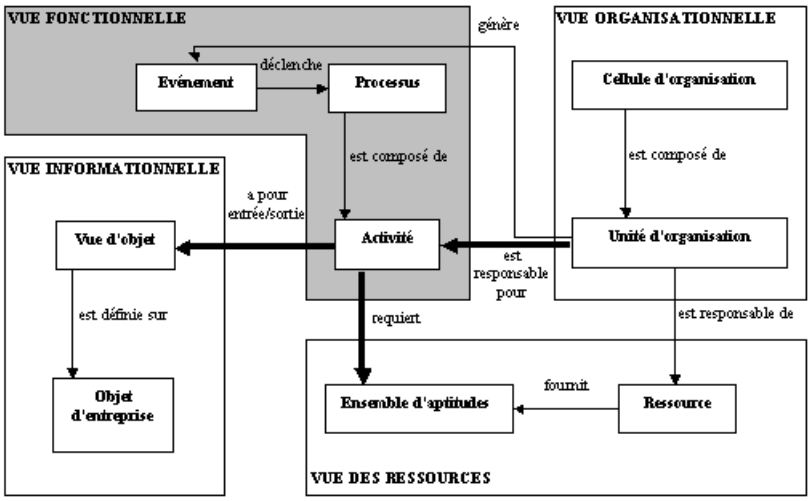


Fig IV. 1. La vue fonctionnelle : au cœur de la modélisation d'entreprise

En effet, la vue fonctionnelle est la seule vue qui est connectée à toutes les autres vues (Fig IV.1). L'élément *activité* de cette vue assure cette connexion avec les autres vues : une *activité* a pour entrée / sortie des *objets* (vue informationnelle), il existe une *unité d'organisation* (vue organisationnelle) de l'entreprise, responsable de l'*activité* et l'*activité* requiert pour son exécution un *ensemble d'aptitudes* fournies par des

ressources (vue des ressources) de l'entreprise. Si nous devons caractériser un modèle d'entreprise et nous poser la question de la vue sur laquelle s'appuyer pour commencer cette caractérisation, la réponse est certainement la vue fonctionnelle.

1.1.2. La place des processus dans la conception du système d'information

La notion de « processus » joue un rôle majeur dans la définition et la gestion des systèmes d'information. L'utilisation d'une « approche processus » pour la conception de SI et, pour reprendre les termes de [**Morley** *et al.* **05**] « *a déplacé le centre des préoccupations de la statique vers la dynamique de la structure. C'est l'identification des processus qui oriente la construction d'une architecture de système d'information…».* En effet, la définition d'un système d'information a évolué d'une focalisation sur son rôle à fournir de l'information à une focalisation sur son rôle à gérer les processus de l'entreprise. D'après [**Reix 02**] « *un système d'information a dépassé le stade d'outil pour devenir l'élément structurant de l'organisation* ». La prise en compte de cette nouvelle réalité a laissé apparaître différentes approches de conception de SI, basées sur l'identification préalable des processus métiers de l'entreprise. La majorité de ces approches respectent une architecture de système d'information, qui peut être composée de trois facettes indépendantes : l'architecture métier, l'architecture fonctionnelle et l'architecture informatique [**Morley** *et al.* **05**].

Nous présentons deux démarches, qui respectent cette architecture : la démarche d'urbanisation de système d'information et la démarche Business Process Management (BPM).

1.1.3. La démarche d'urbanisation des systèmes d'information

L'urbanisation des systèmes d'information [**Sassoon 98**], [**Chelli 03**] considère qu'un système d'information présente deux aspects : d'un côté celui de ses processus et acteurs, de l'autre celui de ses fonctions logiques et entités informationnelles. Une démarche d'urbanisation cherche alors à établir des correspondances entre ces deux aspects. Il s'agit de partir des processus de l'entreprise pour arriver à l'identification des fonctionnalités logiques qui les supportent et enfin déduire les composants logiciels nécessaires pour l'implémentation du système final. Il existe en réalité plusieurs approches d'urbanisation des systèmes d'information, qui présentent de fortes similitudes entre elles. La plupart des approches respecte trois étapes :

- **identification des processus** : consiste à identifier les processus de l'entreprise en se basant par exemple sur les objectifs stratégiques de celle-ci [**Chelli 03**] ou sur ses évènements internes et externes [**Longépé 01**].

- **Identification des activités des processus** : une fois les processus identifiés dans l'entreprise, ils doivent être modélisés en termes d'interactions entre activités.

- **Mise en correspondance entre activités et composants informatiques** : consiste à lier dans un premier temps les activités identifiées dans la phase précédente avec les blocs fonctionnels de l'entreprise. Enfin, les fonctions logiques établies vont être mises en correspondance avec les composants informatiques du SI.

1.1.4. La démarche de Business Process Management (BPM)

D'après [**Soulier** *et al.* **05**], le BPM est un concept de gestion qui affiche l'objectif de permettre aux utilisateurs et gestionnaires métiers de mieux collaborer avec les équipes techniques, pour rendre les processus de l'entreprise exécutables et contrôlables. Ayant la même vision, [**Crusson 03**] et [**Debauche** *et al.* **04**] expliquent que l'objectif de BPM est de permettre aux décideurs, analystes métiers, équipes fonctionnelles et équipes techniques de collaborer pour la définition et l'évolutivité des processus métiers via un seul outil agrégeant les différentes visions.

Le défi actuel de BPM réside dans sa capacité à fournir des outils permettant d'automatiser le passage d'une modélisation de processus (niveau métier) à une définition exécutable (niveau technique) de ce dernier. Ces outils sont connus précisément sous le terme Business Process Management System (BPMS). Ils peuvent aussi offrir des fonctionnalités de supervision de l'exécution de processus : le Business Activity Monitoring (BAM). Le BPM peut désormais être lié à la technologie des web-services. Les propriétés de « réutilisabilité » et de « couplage lâche » [**Bouzguenda 06**] des services-web permettent de concevoir efficacement les processus à partir d'une description de ces derniers. Ainsi, un service-web peut être utilisé dans plusieurs processus de l'entreprise à la fois. Par la suite, l'orchestration des appels aux services assure l'exécution des processus.

L'outil libre Intalio Designer©[1] permet de transformer un processus modélisé avec BPMN en un ensemble de définitions exécutables BPEL, WSDL et XSD. Cet outil nécessite une intervention humaine au milieu de la procédure de transformation, afin d'enrichir les processus désignés par une connaissance additionnelle portant sur les données, l'organisation et d'autres aspects liés au processus.

Des travaux de recherche sur la transformation de BPMN vers BPEL existent. L'équipe de W. Van Dar Aalst [**Ouyang** *et al.* **06**] a formalisé des règles de traduction entre un diagramme BPMN et le code BPEL. Les travaux récents de [**Recker** *et al.* **07**] viennent contredire ceux de [**Ouyang** *et al.* **06**]. Ils ouvrent le débat sur les équivalences sémantiques entre BPMN et BPEL. Les auteurs affirment qu'il paraît illusoire de croire que tout processus BPMN est convertible en un code BPEL. Ils justifient cela par le fait que BPMN dispose d'un ensemble de signes de modélisation plus riche sémantiquement que l'ensemble des balises BPEL. Certains signes BPMN ne peuvent pas être traduits en BPEL. Les auteurs montrent que tout code BPEL est convertible en un diagramme BPMN. La faisabilité dans le sens contraire est source de plusieurs interrogations.[107]

La figure suivante (Fig IV.2) montre un aperçu des règles de traduction de BPMN vers BPEL établies par [**Ouyang** *et al.* **06**] : nous pouvons voir qu'un élément BPMN *gateway* est transformé en des balises BPEL différentes (<*SWITCH*>, <*PICK*>, <*WHILE*>, <*REPEAT*>, etc.) suivant le type de *gateway* : (*PARALLEL, EXCLUSIVE, INCLUSIVE*, etc.). On passe ainsi d'une définition métier des éléments (BPMN) à une définition technique (BPEL).

[1] www.intalio.com

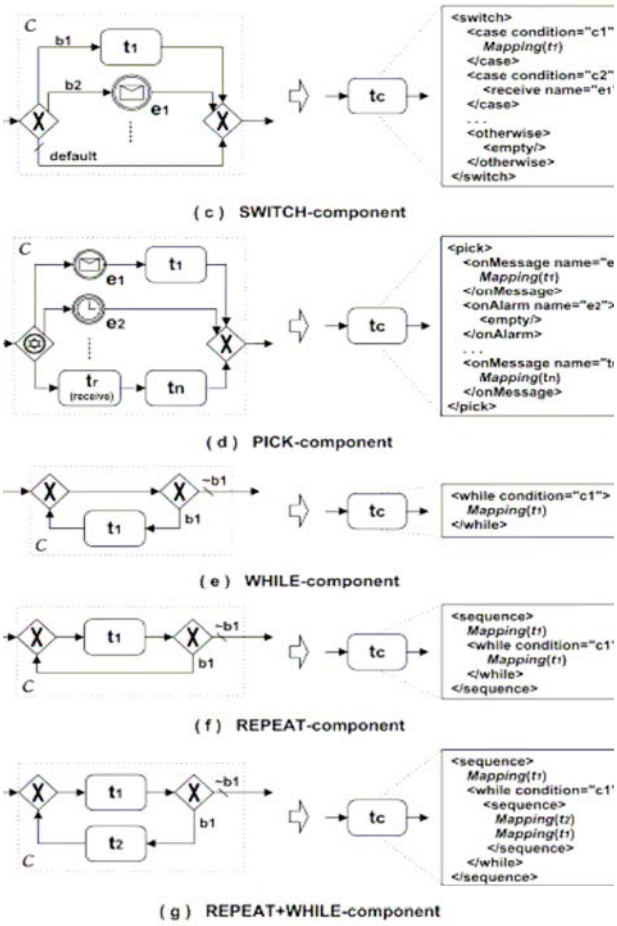


Fig IV. 2. Un exemple de transformations BPMN-BPEL ***[Ouyang** et al. **06]***

1.1.5. Synthèse

La conception de système d'information a parcouru des étapes importantes pour arriver à adopter une démarche mettant « les processus » au cœur du problème. La démarche d'urbanisation de SI présente un cadre idéal structurant le passage d'une modélisation de processus métier à une identification des composants informatiques qui supportent l'exécution de processus. Le BPM est à notre avis une concrétisation de la démarche d'urbanisation dans les entreprises, en proposant des outils qui automatisent le passage entre les niveaux métier et technique. Cependant, les outils BPMS actuels ressemblent à des boîtes noires et ne dévoilent pas leurs mécanismes de transformation. De plus, la démarche BPM semble négliger la génération d'un modèle logique intermédiaire (PIM) entre le modèle métier et le modèle technique. Ce modèle logique devrait permettre par la suite d'assurer une implémentation de processus dans plusieurs plate-formes technologiques. Les travaux académiques sur la transformation de BPMN vers BPEL semblent ne pas se mettre d'accord sur la qualité des fichiers BPEL obtenus : respectent-ils sémantiquement les processus BPMN de départ ? le traitement des correspondances entre langages différents reste toujours compliqué et ouvert à plusieurs interprétations.

Nous présentons par la suite, une analyse sur la possibilité de transformation d'un modèle BPMN (niveau métier, CIM) vers un ensemble de diagrammes UML (niveau logique, PIM). Cette étude nous fournit des réponses quant à la faisabilité d'une telle approche. Sur la base de ces résultats, nous présentons notre approche pour la conception de SIC, une approche basée sur la traduction d'un modèle de processus collaboratif établi

par les partenaires (et modélisé avec un formalisme de modélisation de processus spécifique : BPMN) vers un modèle de système d'information collaboratif en UML, qui supporte ce processus et respecte SOA.

2. Etude de la transformation de langage BPMN vers le langage UML

2.1. Présentation de la problématique

Un des objectifs de nos travaux de thèse est d'expérimenter la transition d'un modèle BPMN, qui exprime *la réponse au besoin de la collaboration souhaitée par les partenaires,* vers un modèle UML qui exprime *la spécification du Système d'Information Collaboratif,* qui supporte cette collaboration. Nous considérons que la réalisation de cette expérimentation exige, dans un premier temps, d'effectuer une comparaison entre les capacités de modélisation des deux langages : BPMN et UML.

Nous considérons qu'un modèle d'un langage particulier exprime une certaine connaissance en combinant un ensemble de signes d'une manière appropriée, en respectant les règles de langage et en définissant d'une manière claire la sémantique liée aux signes et aux règles [**Morley** *et al.* **05**]. En étudiant les deux langages BPMN et UML, nous voulons répondre à la question suivante : **« la connaissance contenue dans des modèles de processus en BPMN constitue-t-elle une « matière première » suffisante pour construire un modèle de SIC en UML ? »**. C'est finalement une exploration des relations qui existent entre les couvertures de connaissance des deux langages qui va nous permettre d'avoir un aperçu sur la possibilité de passage d'un niveau métier (processus BPMN) vers un niveau logique (système d'information UML).

2.2. Différence de couverture entre les deux langages BPMN et UML

2.2.1. Couverture du langage BPMN

BPMN, langage de modélisation de processus métiers, ne peut pas être restreint à la seule couverture de la vue fonctionnelle de la modélisation d'entreprise. Au contraire, il nous semble que compte tenu de la richesse du formalisme BPMN (Chapitre I, Section 4.1.4.2.c), des éléments caractéristiques d'autres vues de la modélisation d'entreprise sont véhiculés (au moins partiellement) par ce type de modèle. Ce langage « déborde », en effet, sur les autres vues : vue organisationnelle, vue informationnelle et vue de ressources :

- **la vue organisationnelle** : qui sont les acteurs de processus ? Quelle relation existe entre eux ? La disposition des « *pool* » et des « *lane* » dans un processus BPMN permet de déduire une certaine organisation des acteurs des processus.
- **La vue informationnelle** : quelles sont les données échangées dans le processus ? une analyse des « *message flow* » des processus permet d'avoir une idée sur les données échangées entre les partenaires.
- **La vue de ressources** : quelles ressources utilisent les activités de processus pour être exécutées ? Pour les ressources mises à disposition dans le processus, leur

identification semble être plus compliquée et un commentaire à joindre avec la description d'une « task » est nécessaire.

Nous avons utilisé le terme « déborder », car nous ne pouvons pas être en mesure de maîtriser les limites et l'étendue de la couverture de BPMN des autres vues de la modélisation d'entreprise. Nous considérons que le langage BPMN couvre essentiellement et principalement la vue fonctionnelle (comme tout modèle de processus). En conclusion, le formalisme BPMN entre dans la catégorie des langages dont R. Saven [**Saven 04**] précise qu'*un modèle de processus peut fournir une définition complète du processus et le lier à d'autres vues d'entreprise*. Dans notre travail, nous souhaitons vivement profiter de la richesse du langage BPMN (couverture de la vue fonctionnelle et débordement sur d'autres vues) pour modéliser la connaissance nécessaire à la conception du SIC.

2.2.2. Couverture du langage UML

Nous avons choisi le langage UML pour modéliser le système d'information collaboratif. UML permet de décrire un système selon différentes vues complémentaires, supportées par des diagrammes. Chaque type de diagramme UML possède une structure (les types des éléments de modélisation qui le composent sont prédéfinis) et véhicule une sémantique précise. Les vues du système que supportent ces diagrammes peuvent être classées de bien des façons (statique / dynamique, logique / physique, etc.), mais nous pouvons proposer une découpe inspirée par [**Booch** *et al.* **04**] et [**Roques 04**] et organisée en quatre vues : fonctionnelle, structurelle, comportementale et architecturale :

- **la vue fonctionnelle** : quelle hiérarchie et quelle arborescence pour les différentes fonctions disponibles dans le système ? Les diagrammes de *cas d'utilisation* permettent de couvrir cette vue.
- **La vue structurelle** : quel agencement pour les composants logiques du système ? Les diagrammes de *classes* et d'*objets* permettent de couvrir cette vue.
- **La vue comportementale** : comment peut-on décrire la dynamique des échanges entre les composants du système ? Les diagrammes de *séquence*, d'*activité*, de *collaboration* et d'*états-transitions* permettent de couvrir cette vue.
- **La vue architecturale** : quel agencement pour les composants physiques du système ? Les diagrammes de *composants* et de *déploiement* permettent de couvrir cette vue.

Il est évident que l'objectif avec lequel on construit le modèle dicte le choix des vues du système que l'on cherche à couvrir. Nous considérons que nous ne serons pas obligés d'utiliser tous les diagrammes dans le cadre de notre objectif de conception d'un modèle (de type PIM) d'un système d'information particulier. Cette proposition relève en fait de deux constats :

Tout d'abord, plusieurs diagrammes de UML couvrent une même vue des système d'information et parfois même selon des approches extrêmement similaires (on peut citer par exemple, les diagramme de *collaboration* et de *séquence*). D'autre part, notre objectif est extrêmement précis et délimité : nous souhaitons obtenir une vue logique du SIC (niveau PIM) sur la base d'un modèle métier de la collaboration (niveau CIM).

Il est donc indispensable de choisir clairement les diagrammes pertinents pour notre approche et de définir la méthode en rapport avec l'obtention de ces diagrammes.

2.2.3. Analyse et comparaison des couvertures de BPMN et UML

Dans notre contexte de « génération de modèle UML par traduction de modèle BPMN », nous nous trouvons face à deux modèles différents : *un modèle BPMN* qui est dédié à la seule description des processus inhérents à la collaboration (vue **comportementale** principalement) même s'il déborde sur les autres vues et *un modèle UML* qui comporte différents diagrammes décrivant tous les aspects du système d'information de cette collaboration (**comportemental,** mais également **structurel**, **fonctionnel** et **architectural**). Notre objectif de traduction doit donc être nuancé, d'une part par la question de la faisabilité d'une approche exhaustive et d'autre part par le positionnement extrêmement précoce de ces travaux dans le cycle de conception du SI (passage du CIM au PIM). Par exemple, la prise en compte de la vue **architecturale** (diagrammes UML de composants et de déploiement) ne semble pas pertinente vis-à-vis du niveau *logique* auquel nous nous trouvons. Afin de clarifier cette situation, nous pouvons représenter notre problématique par le schéma de la figure IV.3 où les flèches entre points de vue signifient « fournit de la connaissance pour ».

BPMN permet d'exprimer un comportement de l'ensemble des partenaires de processus. Nous pouvons alors déduire que les modèles de processus collaboratifs BPMN peuvent permettre la construction de diagrammes relatifs à la vue *comportementale* (A). Quant aux vues *fonctionnelle* (B), *structurelle* (C, E et F) et *architecturale* (D), il sera visiblement indispensable de compléter la connaissance extraite des modèles BPMN pour obtenir les diagrammes correspondants.

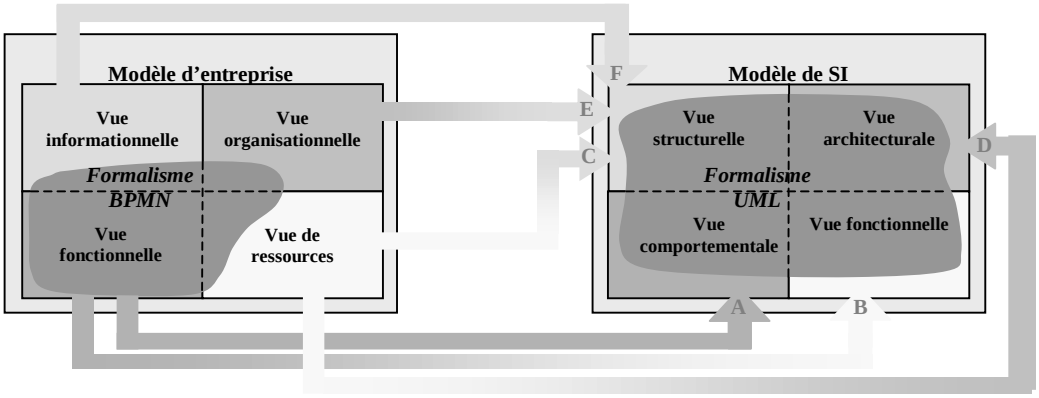


Fig IV. 3. Espace des modèles d'entreprises et espace des modèles de SI

Ces observations semblent logiques : il est difficile d'imaginer construire une description de l'architecture d'un SI (qu'elle soit logique pour la vue *structurelle* ou physique pour la vue *architecturale*) à partir d'une description de processus sans apporter un savoir sur ces structures (type d'architecture, nature des composants). De plus, le « débordement » du formalisme BPMN sur les vues *informationnelle*, *organisationnelle* et *de ressources* ne couvre pas suffisamment les attributs du partenariat : il faut compléter le savoir extrait des diagrammes BPMN à l'aide de connaissances issues de modèles décrivant plus précisément le réseau d'entreprises.

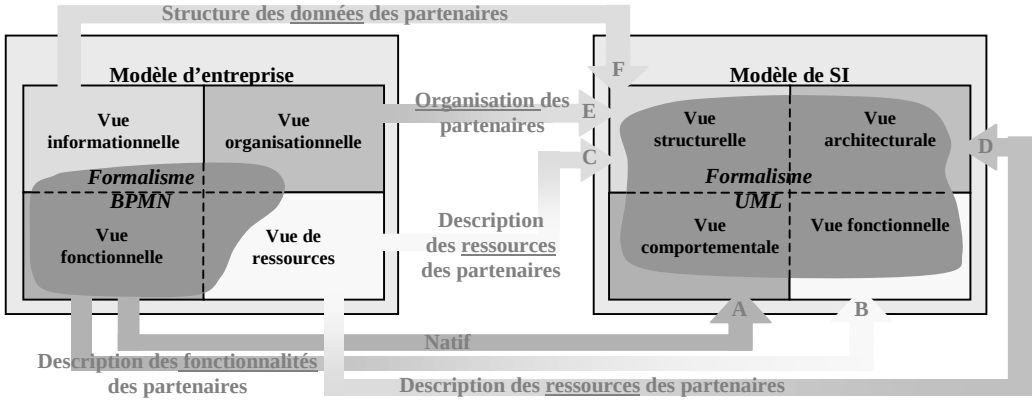


Fig IV. 4. De l'espace des modèles d'entreprises à l'espace des modèles de SI

En outre, les exigences formulées précédemment sur la nécessité pour les partenaires de décrire leurs données (vue *informationnelle*), leurs ressources (vue des *ressources*) et leurs fonctionnalités (vue *fonctionnelle*) permettent de palier les manques vis-à-vis de ces trois vues. Enfin, une connaissance sur l'organisation des partenaires apporte une structuration utile sur le plan *organisationnel*. L'équation précédente peut alors être représentée par la figure (Fig IV.4).

2.3. Génération des diagrammes UML à partir de processus BPMN

Afin de mesurer à quel niveau un modèle BPMN peut fournir de la connaissance pour la construction d'un modèle UML, nous présentons dans l'**annexe A** de ce manuscrit un essai de génération de quelques diagrammes UML à partir d'un processus BPMN et qui correspondent aux vues **structurelle** et **comportementale** :

- le **diagramme de classes,** qui permet de fournir la structure logique du système (organisée en éléments de modélisation statiques tels que les classes ou les paquetages, etc.),
- le **diagramme d'états transitions** qui permet de définir le comportement des entités du système sous la forme d'un *statechart* (par exemple, les états possibles d'une commande peuvent être : validée, envoyée, reçue, etc.),
- le **diagramme de séquence** qui permet de traduire un scénario de communication entre les acteurs et les composants du système,
- le **diagramme d'activité**, qui joue un rôle important en détaillant les activités des différents processus du système.

3. Méthode de conception orientée « processus » de système d'information collaboratif dans une approche SOA

3.1. Présentation de l'approche

Notre objectif dans ce chapitre est de présenter une démarche d'ingénierie qui nous permette de générer un modèle de SIC qui respecte une approche orientée services (SOA), à partir d'un modèle de processus qui exprime la collaboration entre partenaires. Le choix d'une approche de conception orientée « processus » est justifié par les raisons suivantes :

1. dans un contexte collaboratif, les entreprises établissent des collaborations selon des processus stabilisés. Cette stabilité est recherchée par l'entreprise. Ce qui change, ce sont prioritairement les ressources et les informations que l'entreprise aura à gérer dans la collaboration. L' « approche processus », que nous souhaitons concrétiser, capitalise donc cette stabilité des processus en générant des modèles de SI qui peuvent supporter par la suite un enrichissement par des informations sur la collaboration et les partenaires.

2. Nous avons montré au début de ce chapitre le positionnement stratégique des processus dans la modélisation métier de l'entreprise (*Business Modelling*) d'une part et d'autre part dans la conception des systèmes d'information (urbanisation, BPM, etc.). La modélisation métier des processus (*Business Process Modelling*) peut donc être un moyen qui permette de guider la démarche de construction d'un capital de connaissance métier. Ce capital exprime les besoins spécifiques (au processus) pour la construction de modèles de SI. Nous rappelons que la modélisation des processus métier ne permet pas (seule) de recueillir l'ensemble des informations nécessaires à l'expression des besoins en système d'information. D'autres aspects liés à la modélisation d'entreprise sont nécessaires.

3. Nous avons étudié et comparé un certain nombre de démarches, qui mettent en avant l'utilisation de modèles d'entreprise pour définir ou configurer un système d'information (Chapitre II, Section 5.7). L'obtention d'un modèle de SIC **spécifique** à une collaboration particulière est un critère important dans nos travaux. La conception de système d'information est un processus fastidieux et il est important de pouvoir l'achever à moindre coût en se basant uniquement sur la connaissance métier nécessaire pour cela. Cette connaissance ne doit pas dépasser le périmètre défini de la collaboration.

Sans revenir sur les avantages de l'adoption d'une approche orientée-services (SOA), qui permet de faciliter l'établissement de l'interopérabilité entre systèmes d'information hétérogènes, cette section s'intéresse à la présentation d'une méthode de conception de système d'information collaboratif dans une approche SOA. Notre méthode vise à automatiser la génération d'un modèle de SIC dans la collaboration à partir d'un simple modèle de processus. Notre méthode apparaît comme compatible avec l'approche MDI (*Model Driven Interoperability*). Elle comporte quatre étapes indispensables (Fig IV.5) :

- **étape 1** : un modèle BPMN de processus collaboratif décrivant la collaboration des partenaires est présenté comme entrée indispensable pour la méthode. Ce modèle correspond à un niveau métier de la démarche (CIM).
- **Etape 2** : cette connaissance est alors transformée en un modèle UML respectant une architecture orientée services (SOA). Ce modèle correspond à un niveau logique de la démarche (PIM).
- **Etape 3** : le résultat obtenu est enrichi à l'aide d'informations extérieures. Cette étape nécessite une intervention humaine concrète pour compléter efficacement la structure à l'aide des services identifiés, des formats des messages échangés, etc.
- **Etape 4** : le modèle logique complet obtenu est finalement projeté sur une architecture technologique cible, afin de fournir un modèle UML exploitable. Ce modèle correspond au niveau technique de la démarche (PSM).

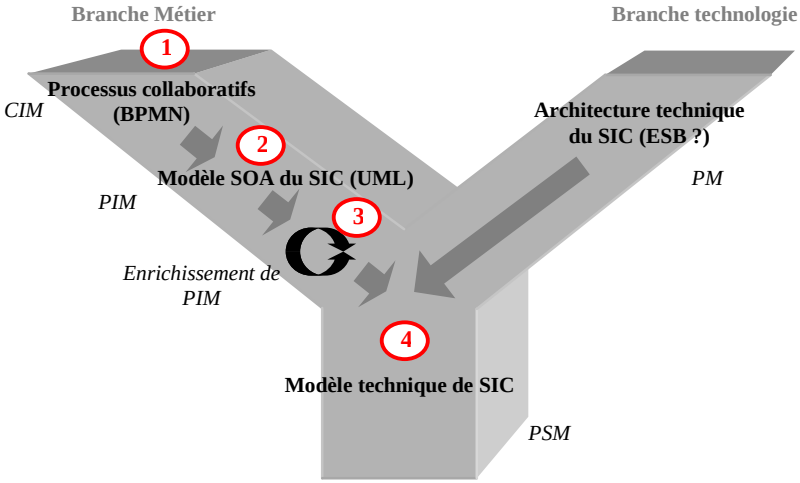


Fig IV. 5. Approche adoptée pour la conception du SIC

Dans nos travaux, nous nous focalisons sur les étapes 1 et 2. Nous définissons en détail les méta-modèles de processus collaboratif et de SIC. Ces deux méta-modèles permettent de prendre en considération en amont un certain nombre de remarques issues des chapitres précédents de ce manuscrit. Ainsi, le méta-modèle de processus collaboratif, s'il respecte intégralement la syntaxe BPMN, incorpore en plus les spécificités liées à notre vision du processus collaboratif. De même, le méta-modèle de SIC prend en considération les spécificités de l'architecture logique choisie :

- le modèle UML doit décrire le SIC avec une architecture SOA. Les diagrammes classiques d' UML ne permettent pas de refléter un choix architectural particulier pour la conception de système d'information. Les éléments de modélisation de base d' UML paraissent, en effet, très génériques et abstraits à ce niveau. Nous devons utiliser des éléments plus spécifiques pour désigner un choix architectural tel que SOA. Un **profil UML** particulier permet une extension des capacités de modélisation standard d' UML. Ainsi, pour SOA par exemple, nous définissons les éléments de modélisation « *service* », « *message* », « *annuaire* », etc. au lieu de tout modéliser à l'aide de « *classe* » dans un diagramme de classe UML. Ce constat est directement lié à l'apparition des DSL (*Domain Specific Language*) qui permettent de définir des langages « à la volée » spécifiques à un domaine particulier. UML (et ses profils) est considéré comme un DSL parmi d'autres.

- Le modèle BPMN doit refléter la présence du SIC comme médiateur de la collaboration. Les échanges entre partenaires doivent passer obligatoirement par ce système. De plus, et vu que le modèle d'arrivée UML obéit à une architecture de services (SOA), et afin de minimiser la perte d'information et garantir la spécificité de modèle généré par rapport au modèle BPMN de départ, le modèle de processus collaboratif opte aussi pour un choix orienté-services pour la présentation des tâches des partenaires.

Nous montrons également les différentes règles de transformation qui permettent le passage entre ces deux méta-modèles. L'étape 3, qui concerne la possibilité d'enrichir les modèles UML obtenus, se fait manuellement et nécessite une intervention humaine. L'étape 4 peut concerner plusieurs architectures techniques, qui peuvent implémenter une approche SOA : ESB, EAI, etc. Nous discutons de cette étape dans les perspectives à envisager pour nos travaux.

3.2. Modélisation de processus collaboratif BPMN

Le modèle BPMN, qui présente l'entrée de l'approche proposée, s'inspire de la définition formelle (qui présente les éléments d'un diagramme BPMN) que nous avons proposée dans l'annexe A (**Définition 1**). Tout d'abord, ce modèle doit obéir à la grammaire du langage BPMN (**Définition 1a**) pour qu'il soit valide. Par la suite, nous souhaitons spécifier le modèle BPMN pour qu'il respecte le méta-modèle spécifiquement conçu.

3.2.1. Spécification de processus collaboratif

Le modèle de processus collaboratif est le modèle qui doit être supporté par le Système d'Information Collaboratif. Le modèle de processus doit donc respecter notre vision du processus collaboratif (chapitre III, Section 5) et refléter le fait que la collaboration est gérée par le SIC (SOA). Un ensemble de contraintes, précisions et restrictions doivent être définies :

- C 1 : une seule « *pool* », appelée « *Système d'information Collaboratif* », doit être présente dans le modèle de processus.
- C 2: une seule « *pool* » est définie pour chacun des partenaires de la collaboration.
- C 3: les partenaires de la collaboration ne peuvent pas communiquer directement. Ils doivent passer par le CIS.
- C 4: à chaque « *message flow* » correspond au moins une donnée « *data* ».
- C 5: une « *pool* » d'un partenaire ne peut contenir que des « *task* ».
- C 6: les « *task* » dans une « *pool* » partenaire ne peuvent être que de type « *receive* » *(attente d'un partenaire de la réponse d'un service),* « *invoke* » *(l'appel d'un service par un partenaire)* ou « *response* » *(quand le partenaire reçoit une requête et doit y répondre).*

3.2.2. Détermination de méta-modèle de processus collaboratif

Le méta-modèle de processus collaboratif décrit les éléments de modélisation qui constituent un modèle de processus collaboratif, mais aussi les contraintes que nous avons identifiées précédemment.

Nous exprimons ces contraintes en utilisant des associations et des cardinalités. Par exemple, la règle C1 (une seule « *pool* » appelée « *Système d'information Collaboratif* » doit être présente dans le modèle de processus) se traduit dans le méta-modèle par deux classes « *Collaborative process* » et « *CIS pool* » et une association qui les lie avec une cardinalité (1,1). Un modèle instancié à partir de ce méta-modèle est donc un modèle de processus collaboratif, qui peut être une entrée valide pour notre démarche de traduction. La figure (Fig IV.6) montre le méta-modèle que nous avons conçu pour l'entrée de notre démarche. Ce méta-modèle décrit les éléments suivants :

- la classe « *BPMN process* » est une classe abstraite qui désigne un processus qui respecte la grammaire de langage BPMN, notre processus (la classe « *collaborative process* ») respecte aussi la grammaire BPMN (lien de spécialisation),
- une classe « *partner pool* » désigne la « *pool* » d'un partenaire,
- une classe « *CIS pool* » désigne la « *pool* » du Système d'Information Collaboratif (SIC),
- une classe « *partner lane* » désigne les « *lane* » appartenant à une « *pool* » de partenaire « *partner pool* »,
- une classe « *CIS lane* » désigne les « *lane* » appartenant à la « *pool* » de CIS « *CIS pool* »,
- une classe « *message flow* » permet de lier une « *partner pool* » avec une « *CIS pool* » et véhicule toujours au moins une donnée « *data* »,
- une classe « *sequence flow* » permet de lier des éléments BPMN appartenant à la « *CIS pool* » et peut véhiculer une donnée « *data* ».
- « *partner task* » désigne une tâche appartenant à « *partner pool* »,
- « *CIS task* » désigne une tâche appartenant à « *CIS pool* »,
- « *start event* », « *intermediate event* », « *end event* », « *gateway* » et « *sub process* » sont les éléments BPMN (définis dans **Définition1** de l'annexe A de ce manuscrit) qui appartiennent uniquement à la « *CIS pool* »,
- « *CIS component* » est une classe abstraite désignant tout élément qui appartient à la « *CIS pool* ».

Afin de faciliter la gestion de la liaison des éléments BPMN entre eux, nous avons défini les éléments suivants :

- « *mf IN* » et « *mf OUT* » sont respectivement les extrémités d'entrée (*in*) et de sortie (*out*) d'un « *message flow* ».
- « *sf IN* » et « *sf OUT* » sont respectivement les extrémités d'entrée (*in*) et de sortie (*out*) d'un « *sequence flow* ».

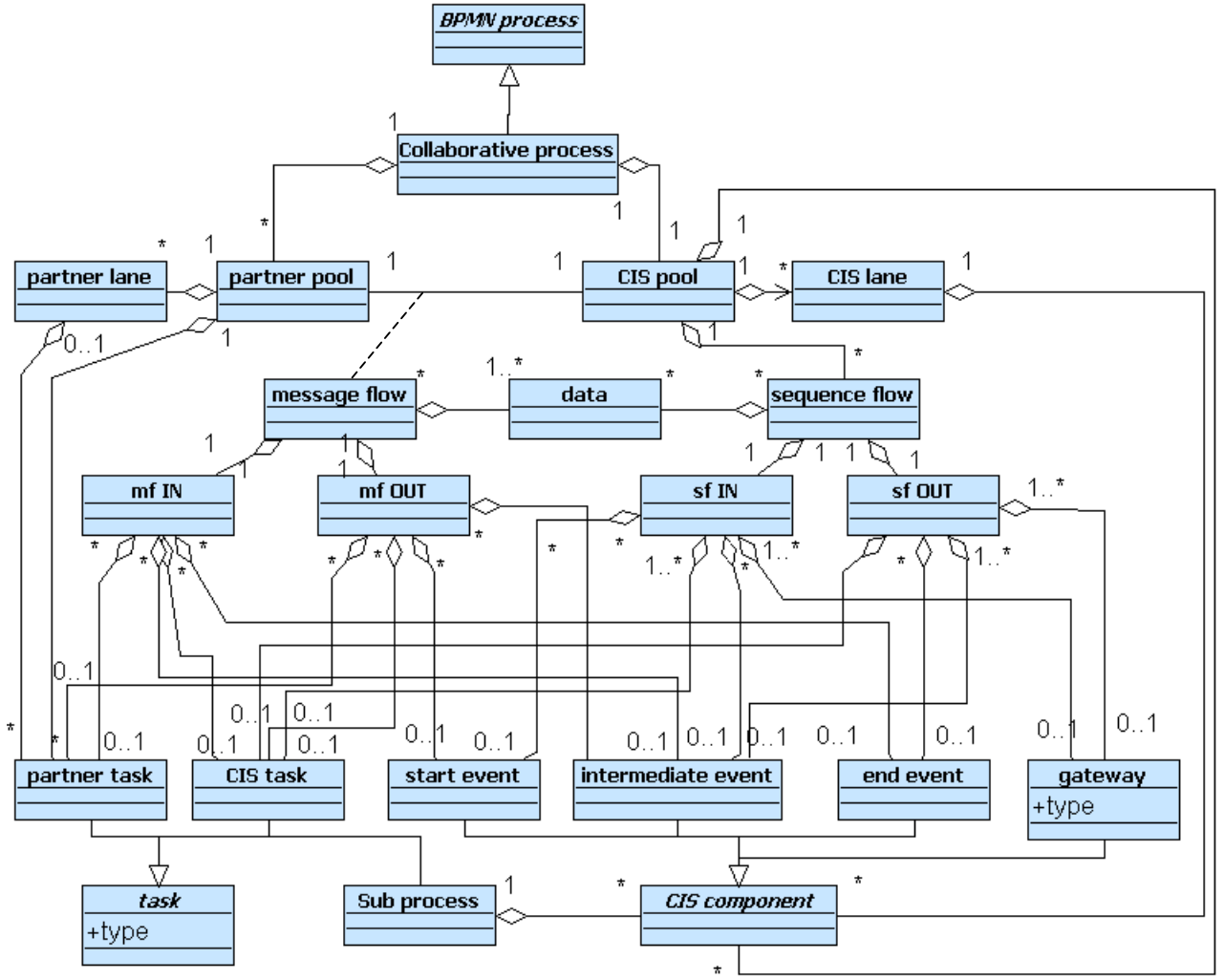


Fig IV. 6. Méta-modèle de processus collaboratif

3.3. Modèlisation de Système d'information Collaboratif orienté SOA

Le langage de modélisation UML possède un ensemble riche de concepts qui ont été créés pour permettre la modélisation de tous types d'applications logicielles et tous domaines. Mais chaque domaine a des notions particulières et des besoins particuliers. De plus, il est fréquent que les utilisateurs souhaitent employer des caractéristiques supplémentaires à celles existantes. Pour cela, UML possède un mécanisme d'extension, afin de le spécialiser et de le personnaliser. Ce dernier autorise la définition de stéréotypes, de contraintes et de propriétés adaptant UML au domaine d'application spécifique. Cet ensemble d'extensions regroupé de manière cohérente est appelé **profil UML**. Cependant, dans une démarche de transformation, il faut définir un méta-modèle qui identifie et décrit le domaine d'application spécifique. Le profil UML fournit dans ce cas une notation pouvant être effectivement outillée par un atelier de transformation de modèles.

Tout comme il nous a fallu définir le méta-modèle source, il faut également définir le méta-modèle cible de la transformation. Ce dernier méta-modèle doit refléter une description de SIC selon une approche SOA. Dans ce but, nous avons étudié un certain nombre de travaux de recherche sur la définition d'un modèle de système d'information respectant une architecture SOA. Citons le projet PIM4SOA [**Benguria** *et al*. **06**], qui définit quatre vues pour la modélisation d'un système d'information selon une approche SOA. Dans chaque vue, on définit un ensemble d'éléments de modélisation :

- **une vue de services :** les services présentent une abstraction et une encapsulation des fonctionnalités fournies par une entité autonome. On décrit les rôles des services, ainsi que les moyens pour y accéder.
- **Une vue de processus :** les processus décrivent l'enchaînement et la coordination des appels aux services.
- **Une vue d'informations :** les informations sont liées aux messages et aux objets métier (facture, bon de livraison, etc.) échangés entre les services.
- **Une vue qualité de services :** la qualité des services comprend des aspects non fonctionnels, tels que la sécurité des transactions, la performance, etc.

D'autres travaux, comme le *UML 2.0 Profile for Software Services*[2] (profile UML 2.0 des services logiciels), permet d'offrir un moyen puissant pour la modélisation et la description des services. Les concepts de base de la modélisation des services en utilisant ce profil sont : un « *service* » possède un élément « *spécification* » qui décrit les différentes « *opération* » et « *protocole* » d'accès. Les « *opération* » utilisent des « *message* » pour communiquer, etc.

Le modèle de SIC en approche SOA s'inspire fortement des travaux cités. Nous considérons trois vues différentes, qui sont indispensables pour modéliser notre système :

- **la vue services :** décrit les services des partenaires et du SIC, utilisés dans la collaboration.
- **La vue informationnelle** : décrit les messages échangés entre les différents partenaires de la collaboration.

[2] http://www-128.ibm.com/developerworks/rational/library/05/419_soa/.

- **La vue processus :** décrit l'enchaînement et la coordination des services en termes d'interactions et des flux de contrôle de processus.

3.3.1. La vue « services »

Cette vue décrit les services rendus disponibles d'une part par les partenaires de la collaboration et d'autre part par le SIC lui-même. Dans cette vue, nous décrivons les informations qui concernent l'utilisation et l'appel aux services par le SIC. La figure suivante (Fig IV.7) montre le méta-modèle relatif à la vue « services » :

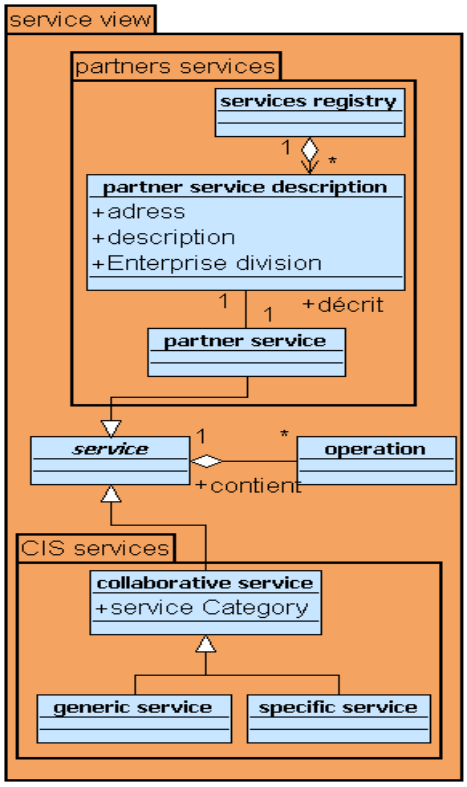


Fig IV. 7. La vue services du méta-modèle de SIC

- la classe « *service* » est une classe abstraite qui désigne un service impliqué dans la collaboration.
- La classe « *operation* » désigne une opération offerte par un service.
- Le package « *Partners services* » désigne la gestion des services des partenaires.
- La classe « *services registry* » désigne un registre accessible par le SIC et qui gère un ensemble de fiches décrivant les services.
- La classe « *partner service description* » désigne une fiche qui décrit un service : ses opérations et une présentation de son rôle dans la collaboration.
- La classe « *partner service* » désigne un service offert par un partenaire dans la collaboration.
- Le package « *CIS services* » désigne la gestion des services de SIC.
- La classe « *collaborative service* » est une classe abstraite qui désigne un service offert par le SIC (Chapitre III, Section 6.1.1.2).
- La classe « *generic service* » est une classe qui désigne un service générique qui peut être utilisé dans un bon nombre de processus collaboratifs. Ce service existe déjà dans une bibliothèque de services gérée par le SIC.
- La classe « *specific service* » est une classe qui désigne un service spécifique à concevoir pour une collaboration particulière.

3.3.2. La vue « information »

Cette vue décrit les données impliquées dans la collaboration et gérées par les différents services. Ces données sont échangées entre les services sous forme de messages. La figure suivante (Fig IV.8) montre le méta-modèle relatif à la vue information :

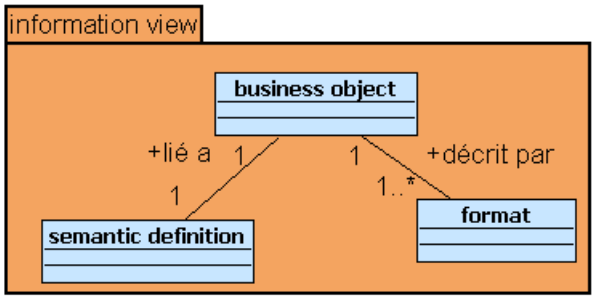


Fig IV. 8. *La vue d'information du méta-modèle de SIC*

- la classe « *business object* » désigne un objet échangé entre deux services.
- La classe « *Format* » décrit le format de message.
- La classe « *semantic definition* » désigne une description sémantique de l'objet. Cette description permet d'identifier facilement l'objet.

3.3.3. La vue « processus »

Cette vue décrit l'orchestration des appels aux services en décrivant les différentes interactions entre les services. La définition de cette vue s'inspire des normes BPML et BPEL. La figure suivante (Fig IV.9) montre le méta-modèle relatif à la vue processus :

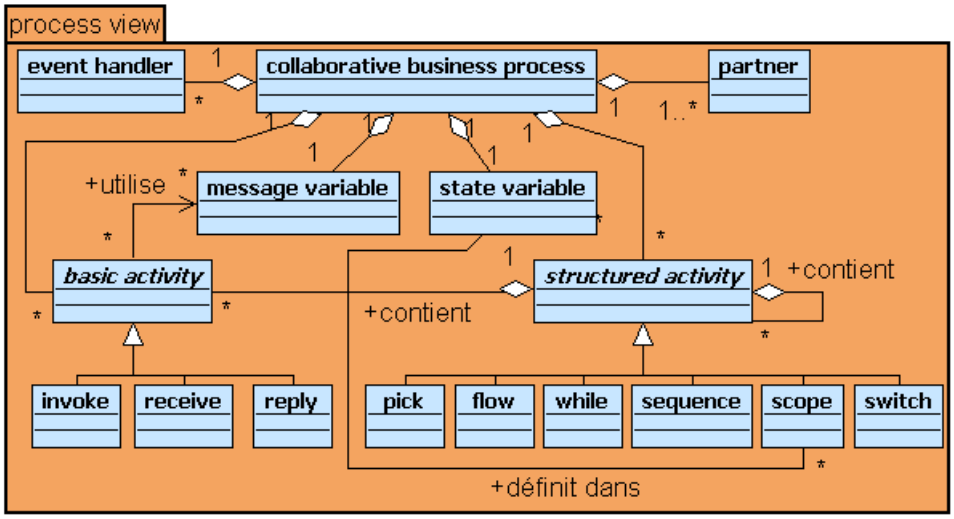


Fig IV. 9. *La vue processus de méta-modèle de SIC*

- la classe « *collaborative business process* » désigne le processus collaboratif géré par le SIC,
- la classe « *partner* » désigne un partenaire de processus,
- la classe « *message variable* » désigne une variable qui présente un message échangé dans le processus,

- la classe « *state variable* » désigne une variable qui peut être utilisée au cours de l'exécution du processus. Cette variable peut prendre plusieurs valeurs suivant l'exécution du processus,
- la classe « *basic activity* » est une classe abstraite qui désigne les classes de base dans un processus :
    a. la classe « *receive* » désigne l'attente de l'arrivée d'un message,
    b. la classe « *invoke* » désigne l'appel à un service,
    c. la classe « *reply* » désigne la réponse à un appel de service,
- la classe « *structured activity* » est une classe abstraite qui désigne les classes qui permettent de structurer le flux d'exécution de parties de processus :
    a. la classe « *while* » permet l'exécution d'une partie de processus en boucle jusqu'à la satisfaction d'une condition,
    b. la classe « *pick* » permet d'attendre la production d'un événement (arrivée d'un message, etc.) pour continuer l'exécution du processus,
    c. la classe « *scope* » permet de regrouper un ensemble d'activités. Ce regroupement peut avoir ses propres variables et ses propres gestionnaires d'évènements,
    d. la classe « *switch* » permet de raisonner sur la valeur d'une variable pour le choix de l'exécution d'une branche d'activités parmi plusieurs,
    e. la classe « *sequence* » permet de lier directement deux parties dans le processus. L'exécution de la deuxième partie suit l'exécution de la première,
    f. la classe « *flow* » permet l'exécution en parallèle de plusieurs activités de processus,
- la classe « *event handler* » permet de gérer la production d'un évènement qui est associé à tout le processus ou uniquement à un « *scope* ».

3.3.4. Connexions entre ces différentes vues

Les connexions entre les différentes vues présentées sont indispensables pour assurer une compréhension efficace de tout le modèle de SIC. La figure précédente (Fig IV.10) montre le méta-modèle complet de notre Système d'Information Collaboratif en version SOA et les connexions existantes entre les différentes vues :

- deux associations « *in* » et « *out* » entre « *business object* » et « *service* » reflètent le fait que les services communiquent entre eux via des messages contenant des objets métier,
- une association entre « *service* » et « *basic activity* » reflète le fait que l'appel à un service est assuré par l'exécution d'une activité de processus,
- une association entre « *business object* » et « *message variable* » reflète le fait qu'un objet métier de la vue informationnelle est représenté par une variable dans la vue de processus,
- une association entre « *invoke* » et « *services registry* » montre que l'appel d'un service de partenaire doit passer par l'annuaire qui gère les moyens d'accès à ces services.

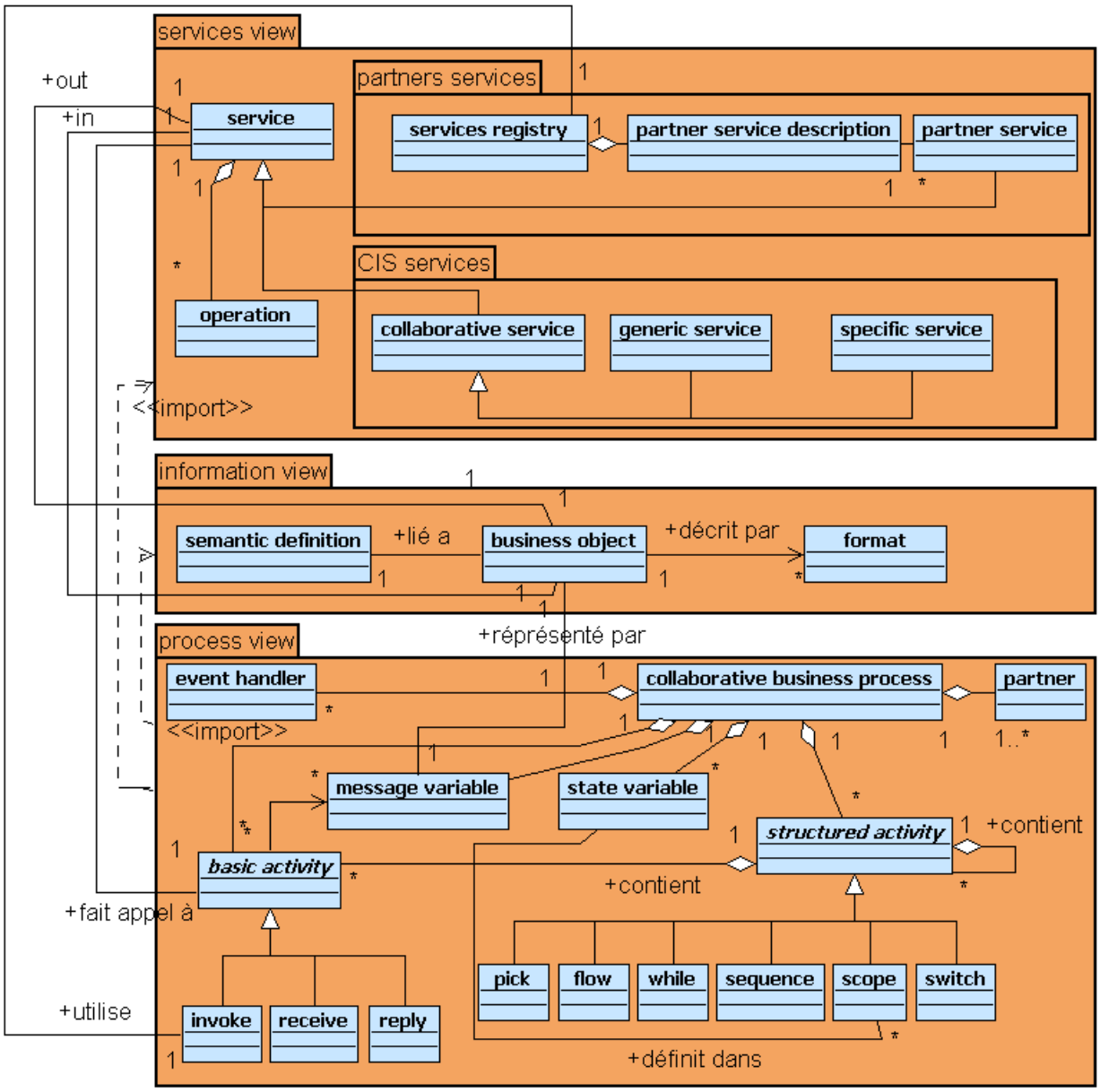


Fig IV. 10. Les liaisons entre les différentes vues du modèle de SIC

3.4. Définition d'un morphisme : processus collaboratif BPMN→ modèle SIC (SOA)

3.4.1. Différence entre « transformation » et « mapping »

Dans nos travaux, nous souhaitons établir un morphisme de modèles entre le modèle de processus collaboratif et le modèle de SIC (SOA) que nous avons définis. La figure (Fig IV.11) représente schématiquement un morphisme entre A et B. A et B sont respectivement les modèles source et cible et M est un morphisme.

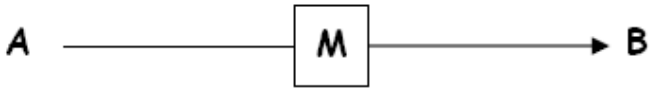


Fig IV. 11. Morphisme entre A et B

Un morphisme de modèle est basé sur les concepts de *mapping* et de *transformation*. La différence entre un mapping et une transformation est la suivante [**D'antonio 05**] :

- un *mapping* est une **relation** qui tend à faire correspondre les éléments de deux modèles sans les modifier. La définition d'un *mapping* nécessite la présence des deux modèles.
- Une *transformation* de modèles est une **fonction** qui transforme (ou modifie) un modèle source en un modèle cible. La modèle source reste intact alors qu'il y a génération d'un nouveau modèle, résultat de la transformation.

Dans notre objectif de *transformation* du modèle de processus collaboratif en un modèle de SIC (SOA), nous établissons tout d'abord un *mapping* entre les différents éléments des méta-modèles de processus collaboratif et de SIC (SOA). Ce *mapping* est basé sur des correspondances sémantiques que nous avons identifiées entre les deux niveaux (processus et système d'information). Le *mapping* que nous souhaitons établir entre les méta-modèles que nous avons défini peut se décomposer selon trois types :

- 1→n : quand un seul élément du modèle source correspond à plusieurs éléments du modèle cible.
- m→1 : quand plusieurs éléments du modèle source correspondent à seul élément du modèle cible.
- m→n : quand plusieurs éléments du modèle source correspondent à plusieurs éléments du modèle cible.

La *transformation* de modèles nécessite l'identification d'un certain nombre de règles basées sur le *mapping* que nous avons identifié auparavant. Le travail sur les règles de transformation correspond finalement au « cœur applicatif » de nos travaux. Ces règles présentent les mécanismes de passage d'un niveau métier (processus BPMN) au niveau logique du SIC (modèle de système d'information).

3.4.2. Justification des mappings

Le *mapping* que nous allons définir, se justifie sur la base des deux considérations suivantes :

- la première concerne l'architecture d'urbanisation de systèmes d'information [**Chelli 03**]. Cette architecture permet analyser le besoin métier (processus collaboratif) et de déduire les composants fonctionnels qui doivent supporter ce besoin. La démarche de *mapping* que nous proposons se conforme en partie à cette vision de l'urbanisation de SI.
- La seconde concerne la possibilité d'utiliser les liens existants entre les différentes vues de la modélisation d'entreprise (vue fonctionnelle, vue organisationnelle, vue de ressources et vue informationnelle) pour justifier le passage d'un élément BPMN vers un élément UML (Fig IV.12). La modélisation d'entreprise peut jouer, dans ce cas, le rôle de pivot pour justifier le passage entre les deux modèles BPMN et UML.

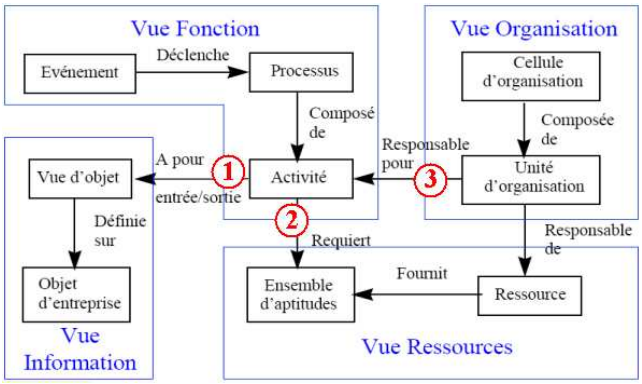


Fig IV. 12. Les liens de passage entre les vues de la modélisation d'entreprise

La figure (Fig IV.12) met en évidence les différents liens conceptuels suivants, entre les vues de la modélisation d'entreprise :

- **lien 1** : les entrées / sorties d'une activité (*vue fonctionnelle*) font référence à des objets de l'entreprise (*vue informationnelle*).
- **Lien 2** : une activité (*vue fonctionnelle*) requiert une ou plusieurs ressources (*vue de ressources*) pour pouvoir s'exécuter.
- **Lien 3** : une activité (*vue fonctionnelle*) possède une unité d'organisation (*vue organisationnelle*) responsable de son exécution.

3.4.3. Catégories de règles

Les règles que nous allons présenter dans la suite de ce manuscrit sont de deux types : les règles de génération de base et les règles de liaison :

- les *règles de génération de base* sont les règles qu'on applique dans un premier temps pour assurer la génération directe des éléments du modèle de SIC (SOA). Cette première couche de règles définit les correspondances directes (*mapping*) entre les éléments des deux niveaux (BPMN-SIC). Nous présentons par la suite ces correspondances en utilisant une modélisation graphique adaptée.

- Les *règles de liaison* sont les règles qu'on applique dans un second temps pour lier les éléments déjà créés par les premières règles. Ce deuxième type de règles est plus complexe que le premier. Ce sont des transformations indirectes des relations existantes entre plusieurs éléments du modèle de départ pour obtenir des relations dans le modèle d'arrivée.

3.4.4. Règles de génération de base de modèle de SIC

Ces règles peuvent être décrites de plusieurs façons. Nous avons choisi de les structurer selon les trois packages à générer dans le modèle de SIC : le package « services », le package « information » et le package « processus ». Les figures (Fig IV.13, Fig IV.14 et Fig IV.15) montrent graphiquement les différentes règles de transformation que nous avons considérées (rond gris) et qui sont basées sur des *mappings* identifiés (deux flèches en pointillé) entre méta-modèle source et méta-modèle cible.

3.4.4.1. Génération du package « services »

La figure (Fig IV.13) montre les règles de correspondances directes entre une partie du méta-modèle de processus (nous retenons les éléments impliqués dans ces règles) et les éléments du package « *services* » du méta-modèle du SIC. Cinq règles définissent ces correspondances :

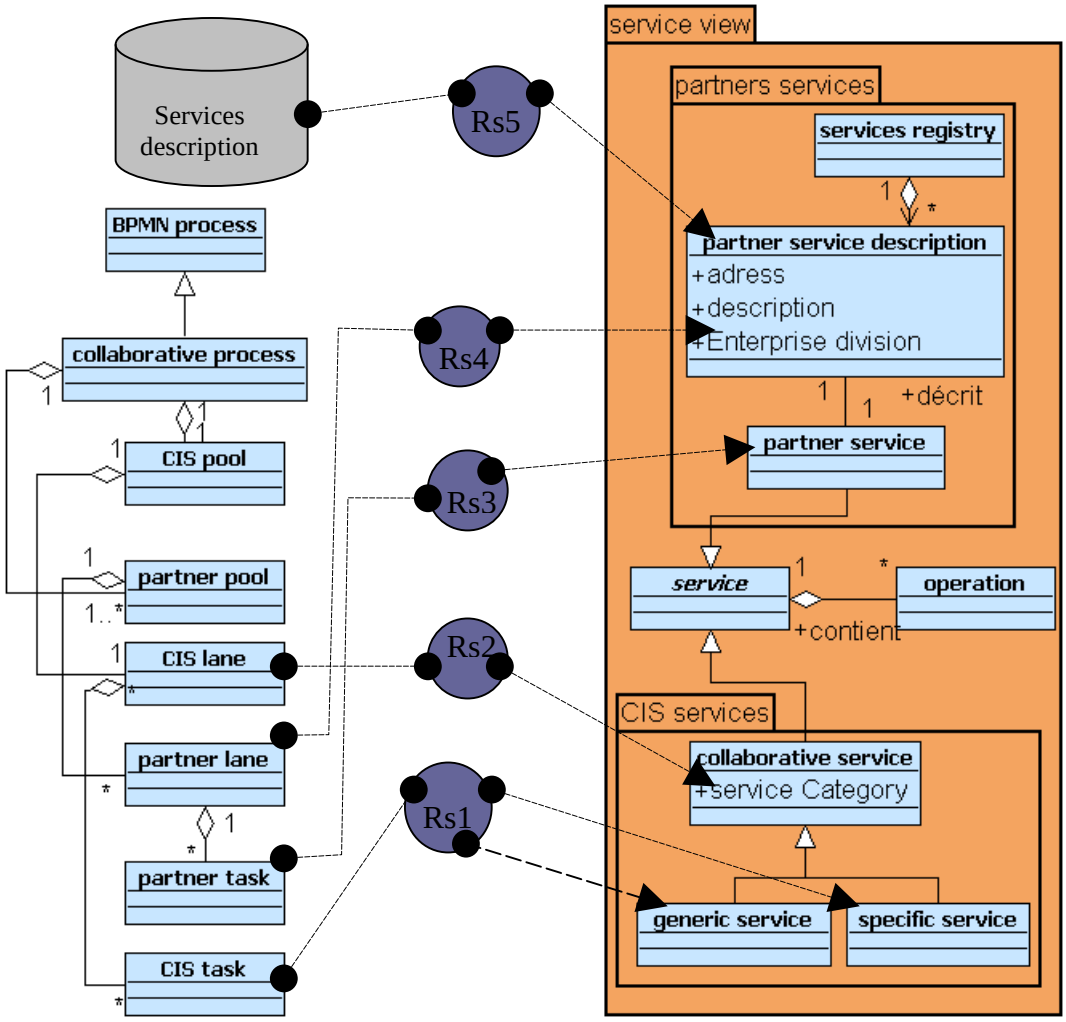


Fig IV. 13. Les règles de transformation pour la génération de la vue de services

➢ **règle Rs1 :** « *CIS task* » → « *generic service* » | « *specific service* »
Cette règle exprime la déduction des services proposés par le SIC à partir des tâches BPMN. Cette règle montre la relation « *tâche* » → « *service* ». Un service au sens de SOA correspond à une fonctionnalité du système. Les services proposés par le SIC appartiennent à deux catégories : les services spécifiques à une collaboration particulière et les services génériques, qui peuvent être utilisés dans plusieurs processus collaboratifs. Le choix d'utiliser une catégorie ou l'autre des services se fait au moment de la conception du processus collaboratif. Une annotation particulière (« *specific* » ou « *generic* ») s'ajoute à la tâche créée.

- **Règle Rs2 :** « *CIS lane* » → « *services category* »

Cette règle exprime une organisation des services proposés par le SIC suivant des catégories de service. Dans ce cas, la « *CIS Pool* » doit être composée de plusieurs « *CIS Lane* ». Une « *CIS Lane* » correspond alors à un attribut « *service Category* » de la classe « *collaborative service* ».

- **Règle Rs3 :** « *partner task* » → « *partner service* »

Cette règle ressemble à la règle Rs1 (« *CIS task* » → « *service* »), mais elle concerne la déduction des services des partenaires impliqués dans la collaboration. Un service d'un partenaire est représenté dans le processus par une tâche.

- **Règle Rs4 :** « *partner lane* » → « *enterprise division* »

Cette règle exprime l'organisation des services selon les partenaires de la collaboration. L'attribut « *enterprise division* » de la classe « *Partner service description* » fait référence au partenaire qui détient le service. Il est déduit à partir de l'élément « *Partner lane* ».

- **Règle Rs5 :** *services description* → « *partner service description* »

Cette règle montre le besoin de disposer d'une source additionnelle de description des services des partenaires. Cette description concerne l'implémentation des services (leurs adresses, protocole d'accès) et leurs opérations.

3.4.4.2. Génération du package « information »

La figure suivante (Fig IV.14) montre les règles de correspondances directes entre une partie du méta-modèle de processus (nous retenons les éléments impliqués dans ces règles) et les éléments du package « *Information* » du méta-modèle du SIC. Comme il en a été exposé précédemment, le processus BPMN ne peut pas contenir toute la connaissance nécessaire pour la génération de la vue informationnelle de notre modèle de SIC. Les règles présentées ci-dessous prennent en compte ce fait en ajoutant une source additionnelle de connaissance : « *information description* ». Cette connaissance est nécessaire, surtout pour la définition des structures des données utilisées par les services des partenaires (schéma de données).

- **Règle Ri1 :** « *data* » → « *business object* »

Cette règle concerne l'élément « *data* » qui est associé aux éléments « *message flow* » (tout « *message flow* » est porteur d'une information « *data* ») et « *sequence flow* ». L'élément « *data* » se traduit par l'élément « *business object* » de modèle de SIC. Cette règle est une spécialisation de la règle n°3 des règles de correspondance BPMN-UML présentées dans l'annexe A.

- **Règle Ri2 :** *information description* → « *format* », « *semantic definition* »

Cette règle montre le besoin d'une source additionnelle de connaissance sur les informations des partenaires pour déterminer les éléments « *format* » (structure d'un « *business object* ») et l'élément « *semantic definition* » (pour exprimer le sens sémantique de l'objet).

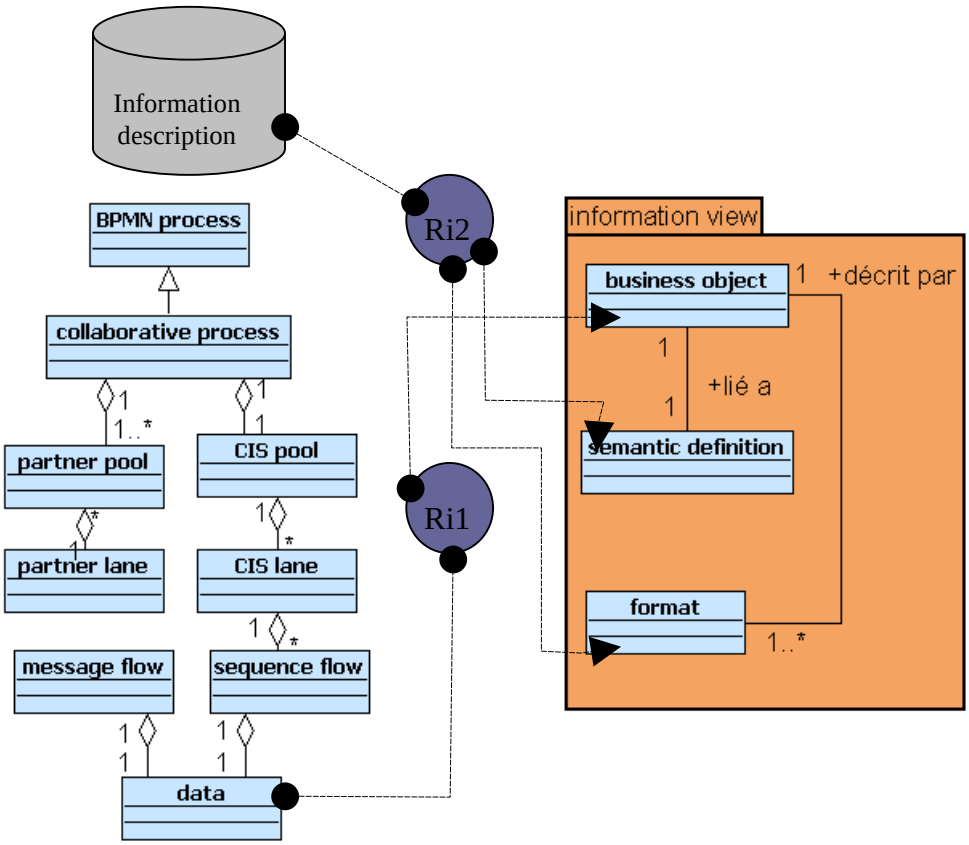


Fig IV. 14. Les règles de transformation pour la génération de la vue « informations »

3.4.4.3. Génération du package « process »

La figure suivante (Fig IV.15) montre les règles de correspondance directe entre une partie du méta-modèle de processus (nous retenons les éléments impliqués dans ces règles) et les éléments du package « *processus* » du méta-modèle du SIC. Comme nous l'avons précisé, le package « *processus* » de notre méta-modèle de SIC s'inspire fortement du méta-modèle du langage BPEL, qui présente le standard actuel en matière d'exécution de processus composé de services-web. Une partie des règles de transformation décrites dans cette partie sont adaptées d'une part des documents officiels sur le langage BPMN [**BPMI 04**] (qui expliquent en partie la traduction vers BPEL) et d'autre part, des travaux de recherche sur la transformation BPMN - BPEL. Nous citons [**Ouyang** *et al.* **06**] et [**Recker** *et al.***07**] :

- **règle Rp1 :** « *partner lane* » → « *partner* »

  Cette règle concerne la déduction de l'élément « *partner* » de modèle du SIC à partir de l'élément « *partner pool* » de modèle de processus collaboratif.

- **Règle Rp2 :** « *data* » → « *message variable* »

  Cette règle permet d'avoir une variable qui présente une donnée échangée dans le processus collaboratif.

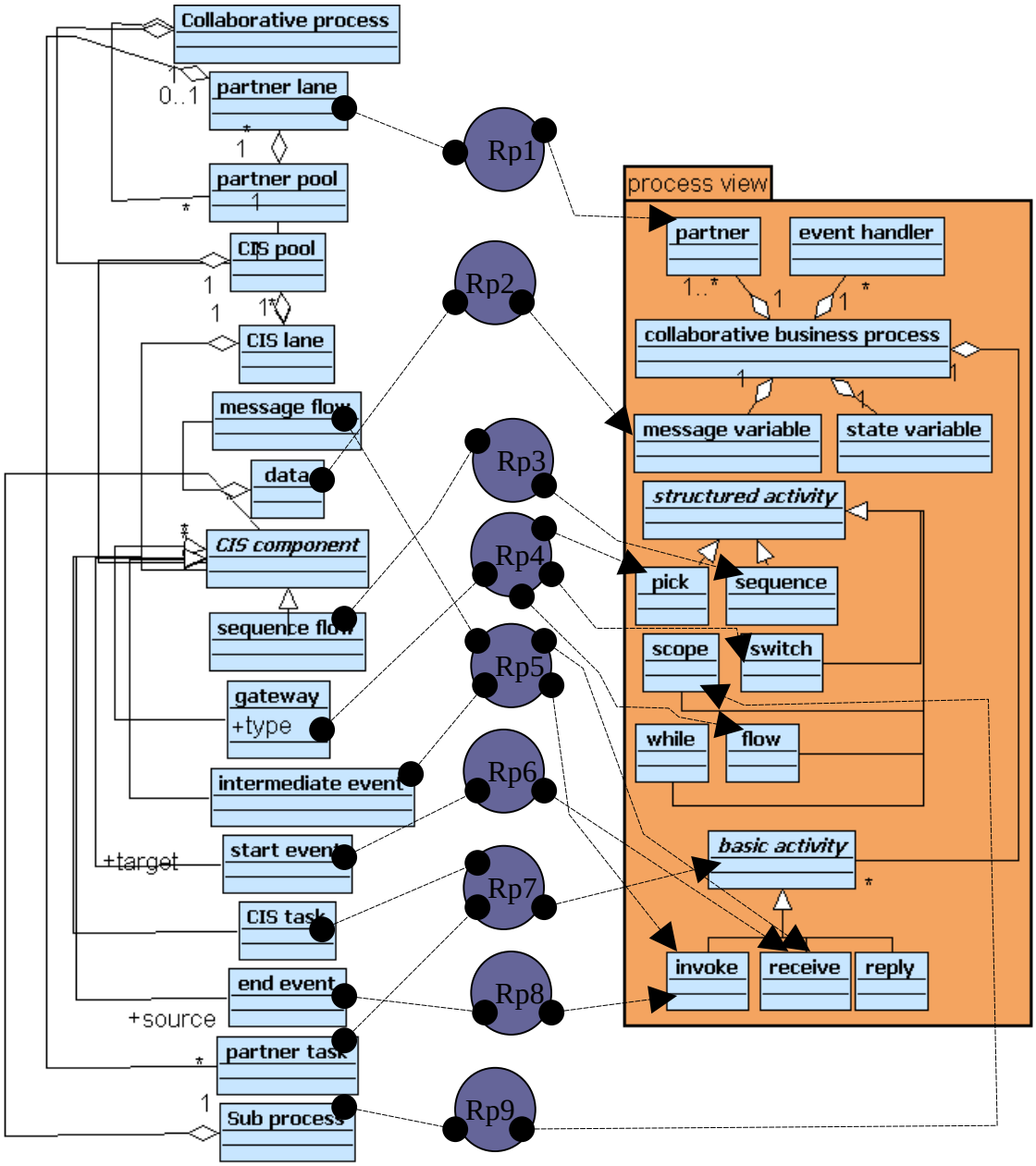


Fig IV. 15. Les règles de transformation pour la génération de la vue processus

➢ **Règle Rp3 :** « *sequence flow* » → « *sequence* »

L'opérateur « *sequence* » désigne dans BPEL une succession logique de l'exécution de deux parties de processus. En BPMN un « *sequence flow* » permet d'une manière similaire la succession de deux parties de processus BPMN.

➢ **Règle Rp4 :**« *gateway* » → « *pick* »|« *flow* »|« *switch* »|
« *flow* »+« *switch*»

Selon la nature de « *gateway* », un élément différent est généré :

- s'il s'agit d'une « *parallel gateway* », alors c'est une classe « *flow* » qui est générée pour exprimer une exécution en parallèle des différentes branches liées à la « *parallel gateway* »,

- s'il s'agit d'une « *data based inclusive gateway* », alors c'est une classe « *flow* » qui est générée, associée à une classe « *switch* » pour chacune des branches liées à la « *data based inclusive gateway* »,
- s'il s'agit d'une « *event based exclusive gateway* », alors c'est une classe « *pick* » qui est générée pour exprimer le raisonnement sur la production d'un évènement pour la poursuite de l'exécution du processus,
- s'il s'agit d'une « *data based exclusive gateway* », alors c'est un « *switch* » qui est générée pour exprimer le raisonnement sur la valeur d'une donnée pour la poursuite de l'exécution d'un processus.

- **Règle Rp5 :** « *message flow* » + « *intermediate event* » → « *receive* » | « *invoke*»

Une analyse préalable de l'« *intermediate event* » est nécessaire avant de décider si l'on génère un « *receive* » ou un « *invoke* ». Il faut en effet voir si le « *message flow* » lié est entrant ou sortant :
- s'il s'agit d'un « *message flow* » entrant par rapport à « *intermediate event* », alors on génère un « *receive* », car il y a réception d'un nouvel évènement à partir d'un éventuel service.
- s'il s'agit d'un « *message flow* » sortant par rapport à « *intermediate event* », alors on génère un « *invoke* », car il y a production d'un nouvel évènement vers un autre service.

- **Règle Rp6 :** « *start event* » → « *receive* »

Un « *start event* » est un évènement qui désigne la réception d'un message pour commencer le processus. Un « *start event* » se traduit alors par un « *receive* ».

- **Règle Rp7 :** « *CIS task* » | « *partner task* » → « *basic activity* »

Les tâches BPMN « *partner task* » et « *CIS task* » se transforment naturellement vers des « *Basic Activity* » dans le méta-modèle du SIC. Selon la valeur type des éléments « *partner task* » et « *CIS task* », on décidera de la transformation :

- sous la forme d'un « *receive* » si l'attribut type est *Receive,*
- sous la forme d'un « *reply* » si l'attribut type est *Service,*
- sous la forme d'un « *invoke* » si l'attribut type est *Send***.**

L'identification de type des tâches BPMN du processus collaboratif peut être déterminée d'une manière manuelle par le concepteur de processus, mais nous avons pu mettre en place une identification automatique du type de la tâche en analysant les « *message flow* » entrants et sortants de cette tâche. L'algorithme suivant permet de mettre en place ce mécanisme :

POUR une tâche X **FAIRE**
SI il existe un « mf OUT » égal à X --*X possède un « message flow » sortant*
ET SI il n'existe pas de « mf IN » égal à X --*X ne possède pas un « message flow » entrant*
ALORS X.type = « Send »
SINON
SI il existe un « mf IN » égal à X --*X possède un « message flow » entrant*
ET SI il n'existe pas de « mf OUT » égal à X --*X ne possède pas un « message flow » sortant*
ALORS X.type = « Receive»
SINON
SI il existe un « mf IN » égal à X --*X possède un « message flow » entrant*
ET SI il existe un « mf OUT » égal à X --*X possède un « message flow » sortant*
ALORS X.type = « Service»
FIN FAIRE

- **Règle Rp8 :** « *end event* » → « *invoke* »

Un « *end event* » est un évènement qui désigne l'envoi d'un message pour signaler la fin de processus. Un « *end event* » se traduit alors par un « *invoke* ».

- **Règle Rp9 :** « *sub process* » → « *scope* »

« *scope* » désigne dans BPEL une partie de processus qui peut être composée d'un ensemble d'éléments : « *basic activity* », « *gateway* », « *message variable* », etc. un sous-processus se transforme alors en « *scope* » dans le modèle du SIC.

3.4.5. Définition des règles de liaison

Ces règles interviennent une fois que le modèle cible du système d'information collaboratif est peuplé par des éléments en application des règles de génération de base. Les règles de liaison ont pour objectif de lier par des associations (au sens d' UML : simple, composition, etc.) des éléments qui appartiennent à un même package (services, informations ou processus) ou à des packages différents. Ces règles illustrent la transformation d'un lien logique de dépendance entre des éléments du modèle de départ (inclusion, liaison, généralisation, spécification, etc.) vers un lien de dépendance entre les éléments du modèle d'arrivée. Avant d'identifier les différentes règles de liaison, nous définissons la fonction *Y = Equivalent (X, pa)* ou *X* est un élément qui appartient au modèle BPMN, *Y* est un élément équivalent qui appartient au modèle de SIC et *pa* désigne le package cible de la transformation (services, informations ou processus). Par exemple :

« *CIS service* » = *Equivalent* (« *CIS task* », *services*)
et
« *basic activity* » = *Equivalent* (« *CIS task* », *processus*)

Nous définissons ensuite l'ensemble des règles de liaison qui permettent de finir la structuration de notre modèle de système d'information collaboratif.

- **Règle de liaison 1 :** gestion des « *sequence* »

Afin de pouvoir modéliser le séquencement de l'exécution des activités dans la vue « *processus* », cette règle permet de lier un élément « *sequence* » généré par application de la règle « Rp3 » à deux éléments du package « *processus* ».

SI
x et *y* deux éléments du modèle BPMN liés par un « *sequence flow* » *sf* ET
x' = *Equivalent (x, processus)*
y' = *Equivalent (y, processus)*
sf' = *Equivalent (sf, processus)*
ALORS *sf'* qui est un élément « *sequence* » doit être lié par deux associations (source et destination) à *x'* et *y'*.
FIN SI

- **Règle de liaison 2 :** gestion de la relation entre « *service* » (package services) et « *business object* » (package informations)

Cette règle permet de lier les « *service* » de la vue « *services* » avec les « *business object* ») de la vue « *information* ».

SI
x est une « *partner task* » ou « *CIS task* » qui manipule une donnée *y* « *data* » par le biais d'un « *message flow* » ET
x' = *Equivalent (x, services), x'* est un « *service*»,
y' = *Equivalent (y, informations), y'* est un « *business object* »
ALORS il existe une association (*utilise*) entre *x'* et *y'*.
FIN SI

- **Règle de liaison 3 :** gestion de la relation entre « *basic activity* » (package « *processus* ») et « *Service* » (package « *services* »)

Cette règle permet de lier les deux vues « *processus*» et « *services* » en montrant que les interactions entre services sont représentées par le biais d'activités dans la vue « *processus* ».

SI
x est une « *partner task* » ou « *CIS task* » ET
x' = *Equivalent (x, processus), x'* est une « *basic activity* »,
y' = *Equivalent (x, services), y'* est un « *service* »
ALORS il existe une association (*fait appel*) entre *x'* et *y'*.
FIN SI

- **Règle de liaison 4 :** gestion de la relation entre « *invoke* » (package « *processus* ») et « *Services registry* » (package « *services* »)

Cette règle permet d'illustrer le fait que l'appel à un service passe par une consultation de l'annuaire « *Services registry* »

SI
x est un élément « *invoke* » généré dans la vue processus et *y* l'élément « *Services registry* »
ALORS il existe une association (*utilise*) entre *x* et *y*.
FIN SI

4. Conclusion du chapitre

Ce chapitre, qui concerne la conception de Système d'Information Collaboratif, a été présenté en deux grandes parties. La première partie montre le rôle central que jouent les processus dans la modélisation métier d'une part et dans la conception de systèmes d'information d'autre part. Nous avons évalué les différences en matière d'apport de connaissance entre un modèle BPMN et UML. Dans la deuxième partie, nous avons proposé notre approche de conception d'un modèle de SIC en prenant en compte un choix architectural orienté service. Notre démarche est basée sur une exploitation de la connaissance liée à un modèle de processus collaboratif qui montre les interactions entre les différents partenaires. Un des avantages de notre approche est la garantie que le modèle de SIC généré est créé spécifiquement pour supporter le processus identifié à l'entrée de la démarche. Nous avons identifié en détail les méta-modèles de processus collaboratif et de SIC (en SOA) et les règles de transformation. Nous ambitionnons, à travers notre approche, de faciliter la communication entre les analystes métier (modélisation de processus collaboratifs) et développeurs de système d'information (mise en œuvre des fonctionnalités du système).

Afin de mettre en pratique notre travail sur la génération d'un modèle PIM de SIC en approche SOA, nous nous proposons dans le chapitre suivant de présenter une maquette d'atelier logiciel qui permette, à partir de la modélisation d'un processus en BPMN, de générer un modèle UML de SIC, facilitant ainsi le paramétrage et l'implémentation finale de notre solution, support de l'interopérabilité des systèmes d'information.

Chapitre V

Implantation d’un atelier de transformation de modèles de processus : Traducteur 1.0

1. Introduction du chapitre

Afin de valider une partie de notre travail, nous avons conçu et développé une maquette d'atelier logiciel de transformation de modèles. Cet outil permet la définition d'un modèle de système d'information en SOA (UML) à partir d'un modèle de processus collaboratif (BPMN). La première version de cet outil est dénommé **Traducteur V1.0**. En fin de chapitre, nous présentons un cas d'application de notre travail, basé sur un exemple de processus collaboratif.

2. Présentation de l'atelier

Dans cette section, nous présentons les différents éléments qui caractérisent le développement de notre atelier. Après avoir rappelé l'objectif et les fonctionnalités principales de l'atelier, nous présenterons l'architecture technique générale de l'atelier, ensuite nous détaillerons l'implémentation des principaux modules le constituant.

2.1. Objectif de l'atelier logiciel

L'objectif principal de notre atelier est de valider les règles de traduction (que nous avons définies dans le chapitre précédent) entre un processus collaboratif (BPMN) et un modèle de SIC SOA (UML). Nous voulons démontrer la possibilité de disposer d'un outil permettant la facilitation de la communication entre un analyste métier chargé de la modélisation des processus collaboratifs et un expert de système d'information chargé d'accomplir la tâche de conception du système d'information collaboratif qui va supporter l'exécution de ces mêmes processus collaboratifs. Notre travail permet donc de raccourcir la distance qui sépare le domaine de la modélisation de processus et le domaine de la conception de système d'information (Fig V.1).

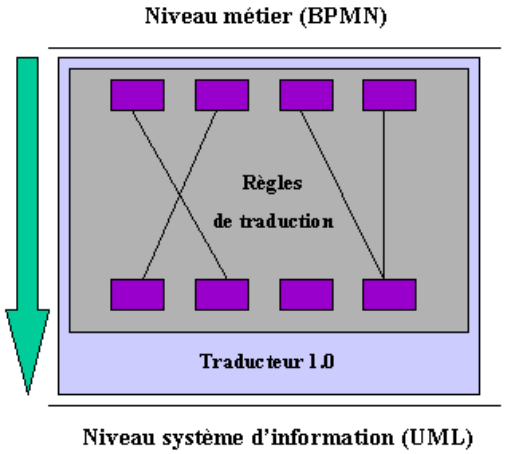


Fig. V.1. Objectif principal de l'atelier logiciel à développer

2.2. Fonctionnalités de l'atelier logiciel

Afin de répondre à l'objectif décrit dans la section précédente, nous proposons une maquette d'atelier logiciel, qui est, en fait, une combinaison de trois outils différents. Chaque outil propose une fonctionnalité indispensable dans notre atelier :

- **modélisation du processus métier** : l'utilisateur qui joue le rôle d'analyste métier utilise cette fonctionnalité pour dessiner le processus collaboratif, entrée de la démarche.
- **Transformation : modèle de processus → modèle de SI** : cette fonctionnalité permet de générer un modèle UML de SI, respectant un profil UML SOA à partir d'un modèle de processus BPMN. Cette fonctionnalité inclut une « sous-fonctionnalité » de vérification de la compatibilité du modèle de processus collaboratif vis-à-vis du méta-modèle décrit dans le chapitre précédent. En effet, la transformation de modèles ne peut avoir lieu sans cette vérification préalable.
- **Visualisation et enrichissement du modèle UML** : l'utilisateur qui joue le rôle de concepteur de système d'information collaboratif peut visualiser le résultat de la fonctionnalité décrite précédemment. Par ailleurs, il peut également modifier le modèle et l'enrichir avec des informations qui ne peuvent pas être fournies par le modèle BPMN de départ. Cet enrichissement peut permettre par la suite de générer un modèle PSM qui dépende de la spécification d'une plateforme technologique.

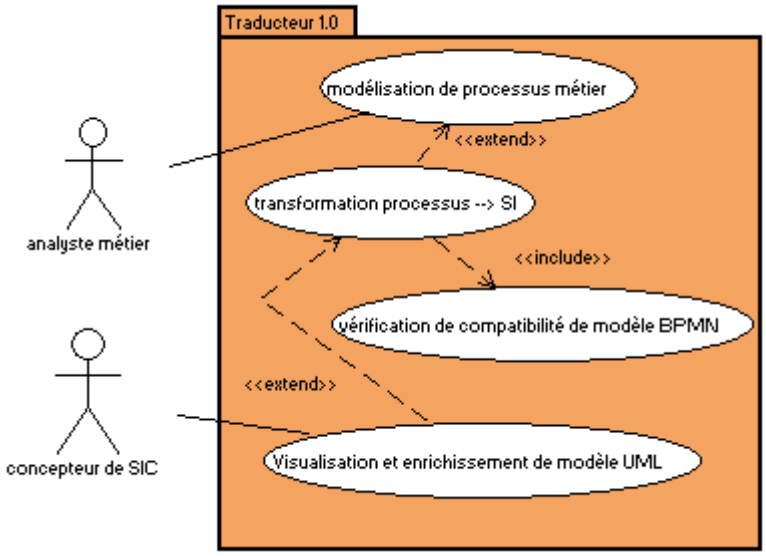


Fig. V.2. Diagramme de cas d'utilisation de l'outil Traducteur V1.0

La figure précédente (Fig V.2) montre le diagramme de cas d'utilisation relatif à notre outil Traducteur V1.0. Les trois fonctionnalités citées ci-dessus sont représentées par des cas d'utilisation. La deuxième fonctionnalité (transformation processus → SI) présente notre contribution majeure en matière de définition et d'implantation des méta-modèles et des fichiers de transformation. Nous détaillons ce travail ultérieurement.

2.3. Architecture technique générale

La figure suivante (Fig. V.3) montre l'architecture technique générale de notre atelier. Ce dernier est basé sur trois outils « logiciel libre » qui s'exécutent tous dans un IDE (*Integrated Development Environment*)[1] **Eclipse©** : l'outil **Intalio Designer©** permet, en utilisant une palette de symboles BPMN, la modélisation de processus collaboratif, entrée de notre démarche de transformation. L'outil **Atlas Transformation Language (ATL©)**

[1] Environnement de développment intégré.

permet de récupérer la définition exploitable d'un modèle de processus (format XML) et de générer un modèle UML de système d'information collaboratif (fichier .uml), résultat de l'exécution des règles de transformation que nous avons définies et implantées. Enfin, l'outil **TOPCASED©** nous permet de visualiser le modèle généré et de l'ouvrir dans un éditeur de diagramme de classes UML. Par la suite, il sera facile d'intervenir sur le modèle afin de l'enrichir et de le compléter.

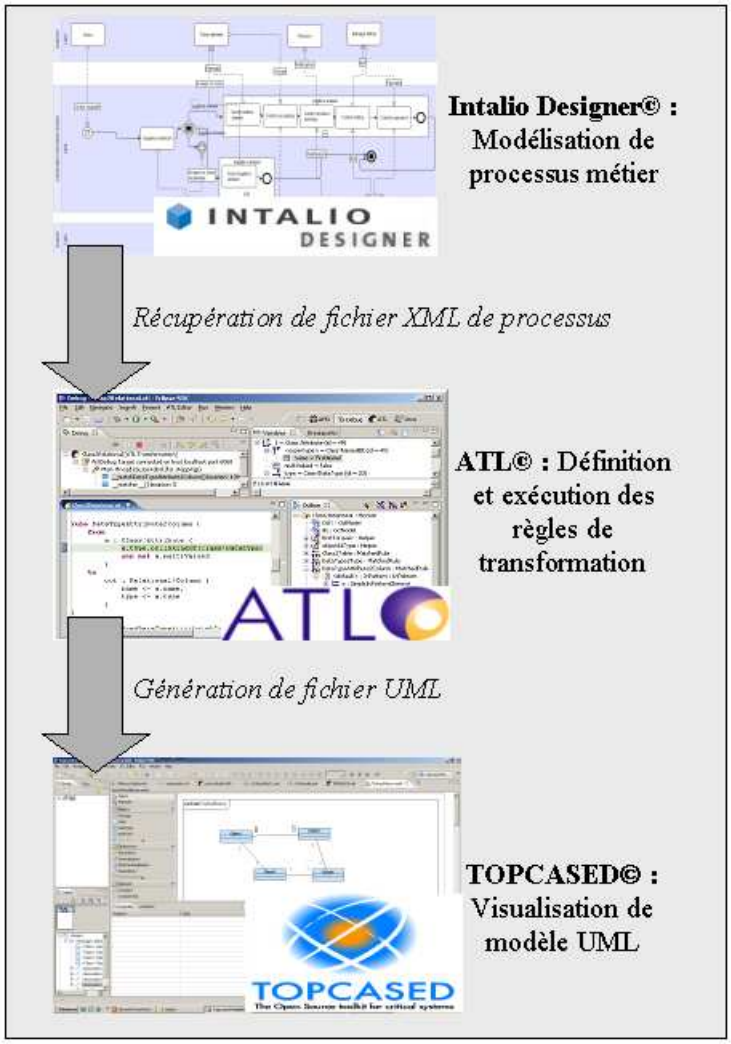


Fig. V.3. Architecture technique générale de l'atelier logiciel

2.4. Présentation des composants de l'atelier

Dans cette section, nous présentons brièvement les outils composant notre atelier de transformation de modèles et justifions les choix que nous avons faits par rapport à des critères que nous avons été amenés à définir.

2.4.1. L'outil Intalio designer©

2.4.1.1. Choix de l'outil

Une des premières difficultés quant à la validation de nos résultats de recherche a été de trouver un éditeur permettant de construire des modèles de processus en BPMN. Le site web officiel du langage BPMN (*www.bpmn.org*) propose une liste d'outils supportant la

modélisation avec ce langage. Nous avons étudié en particulier les outils : *Process Modeler for Visio©*[2], *Together Architect 2006©*[3] et *Intalio Designer©*[4]. Nous avons considéré deux critères principaux quant au choix de notre outil :

- **gestion de la grammaire BPMN** : l'outil doit permettre de contrôler le respect de la grammaire BPMN au cours de la conception d'un processus. Par exemple, l'éditeur doit interdire la liaison de deux *tâches* appartenant à deux *pools* différentes par un *sequence flow*. Ce critère est important pour l'obtention de diagrammes BPMN valides.
- **Génération d'un fichier exploitable de processus** : nous cherchons dans l'atelier à utiliser une définition exploitable d'un processus BPMN pour pouvoir la traduire en un modèle UML. Cela nécessite que l'éditeur BPMN puisse générer un fichier qui définisse le processus et qui soit décrit à l'aide d'un format non propriétaire, mais standardisé et exploitable (tel que XML).

Outil/critère	Gestion de la grammaire BPMN	Génération d'un fichier exploitable de processus	Logiciel libre
Process Modeler for Visio©	Non, ne permet qu'une simple modélisation graphique	Non	Non
Together Architect 2006©	Oui	Non	Non
Intalio Designer©	Oui	Oui, XML	Oui

Tab V.1. Étude comparative d'outils de modélisation BPMN

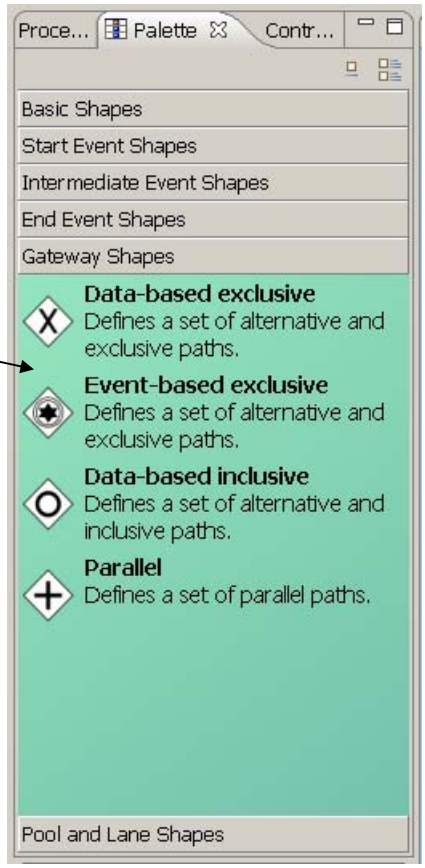
Le tableau comparatif (Tab V.1) montre que le choix d' *Intalio Designer©* est judicieux pour la modélisation de processus collaboratif, en entrée de notre démarche de transformation (d'autant plus si l'on tient compte du critère secondaire relevant de l'utilisation d'un logiciel libre).

2.4.1.2. Modélisation des processus collaboratifs avec Intalio Designer©

2.4.1.2.a. Modélisation de processus

Cet outil permet, moyennant une palette graphique qui comporte la plupart des symboles BPMN, de concevoir des processus conformes à la grammaire BPMN. La figure (Fig V.4) montre un aperçu de cette palette.

[2] http://www.itpearls.com/.
[3] http://www.borland.com/us/products/together/index.html.
[4] www.intalio.com.



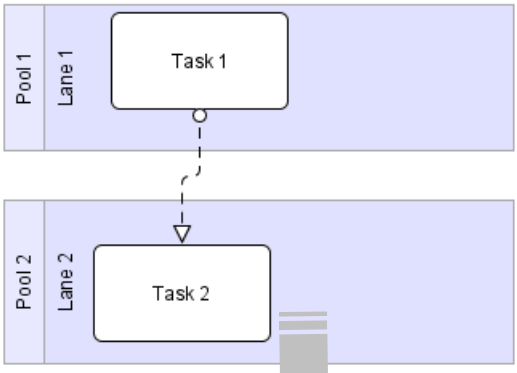
Palette de modélisation proposée par l'outil *Intalio Designer*© (développée au niveau des *gateways*)

Fig. V.4. Exemple de symboles BPMN pour la modélisation de processus collaboratif

Pour la suite du processus de traduction, cette phase de modélisation doit impérativement respecter le méta-modèle de processus collaboratif (défini dans le chapitre précédent). Cette contrainte est extrêmement délicate dans la mesure où elle est laissée au bon vouloir de la personne en charge de la modélisation (l'outil *Intalio designer*© est dédié à la modélisation BPMN et non pas à la modélisation de processus collaboratifs respectant notre méta-modèle). Nous reviendrons sur cet aspect de la conception du CIM dans le Chapitre VI de ce manuscrit. Le respect de la grammaire BPMN est quant à lui assuré par l'outil lui-même (via des messages d'alerte qui guident l'utilisateur).

2.4.1.2.b. Génération de fichier XML

Intalio Designer© permet de générer une définition XML à partir de processus conçus graphiquement. La figure (Fig V.5) montre un exemple de génération de fichier XML à partir d'un modèle graphique. Cet exemple illustre deux tâches BPMN liées par un message flow. Nous pouvons constater que chaque élément BPMN du modèle graphique (*task*, *pool*, *lane*, *message flow*, etc.) correspond à une balise XML dans le fichier XML généré (<task>…</task>, <pool>…</pool>, etc.). Chaque balise contient des informations sur l'élément (nom, position dans le modèle, etc.).




```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<?sdm orientation="horizontal"?>
-<SDM>
  - <Pool id="ID-86e6c398-c323-1004-8aca-88d79c61ebf6">
      <property name="Name">Pool 1</property>
      <property name="x">161.59766</property>
      <property name="Size">200.0</property>
      <property name="Length">319.40234</property>
      <property name="y">25.0</property>
    </Pool>
  - <Lane id="ID-86e6c39b-c323-1004-8aca-88d79c61ebf6">
      <property name="Name">Lane 1</property>
      <property name="x">187.29883</property>
      <property name="MaxLevel">1</property>
      <property name="Size">200.0</property>
      <property name="Length">293.70117</property>
      <property name="Lane">ID-86e6c398-c323-1004-8aca-
          88d79c61ebf6</property>
      <property name="y">25.0</property>
    </Lane>
  - <Task id="ID-86e6c39e-c323-1004-8aca-88d79c61ebf6">
      <property name="Name">Task 1</property>
      <property name="x">284.8506</property>
      <property name="Lane">ID-86e6c39b-c323-1004-8aca-
          88d79c61ebf6</property>
      <property name="y">125.0</property>
    </Task>
  + <Pool id="ID-1f24c9a0-c370-1004-83a4-d9c76a2b9f2b">
  + <Lane id="ID-1f24c9a3-c370-1004-83a4-d9c76a2b9f2b">
  + <Task id="ID-1f2e9b48-c370-1004-83a4-d9c76a2b9f2b">
  - <Message from="ID-86e6c39e-c323-1004-8aca-88d79c61ebf6" id="ID-
      1f3c44f0-c370-1004-83a4-d9c76a2b9f2b" islink="true" to="ID-
      1f2e9b48-c370-1004-83a4-d9c76a2b9f2b">
      <property name="Name" />
      <property name="hasSchema">false</property>
      <property name="Message">Message</property>
    </Message>
</SDM>
```

Fig. V.5. Exemple de génération d'un fichier XML à partir d'un modèle BPMN par l'outil Intalio Designer ©

2.4.2. L'outil ATL©

2.4.2.1. Choix de l'outil

La deuxième difficulté rencontrée lors de la mise en place de notre atelier est le choix d'un langage de transformation qui permette de répondre parfaitement à nos exigences. Si le

domaine de la transformation de modèles n'en est qu'à ses débuts, cela n'a pas empêché un nombre important d'outils de transformation d'apparaître. En effet, durant ces dernières années, la création d'un langage de transformation normalisé est devenue le centre de réflexions autour de MDA [**Lopes 05**]. L'OMG a lancé en 2002 un appel à travers son RFP QVT [**OMG 02**], pour que des propositions d'un langage de transformation soient faites. Le RFP QVT pour *Query, View and Transformation* n'est pas seulement destiné à la transformation, mais aussi à l'expression des requêtes (*Query*) et la définition de vue (*View*). Plusieurs langages s'inscrivent dans la recommandation QVT. Nous citons *MOLA* (MOdel transformation LAnguage) [**Kalnins** *et al.* **04**], *ATL©* (Atlas Transformation Language) [**Jouault 06**], *GreAT* [**Karsai** *et al.* **03**], *AndroMDA* [**Andro 07**].

Ces outils de transformation peuvent être classés suivant plusieurs critères : la nature des règles (impératives, déclaratives ou hybrides), la gestion des modèles (textuels ou graphiques), la gestion de la traçabilité des transformations, le fait que l'outil soit libre ou pas, etc. Il est alors possible qu'un langage de transformation soit plus adapté qu'un autre dans un contexte spécifique. En conclusion, Il apparaît difficile de converger vers un langage unique. Nous fixons deux critères liés aux transformations de modèles que nous souhaitons établir : la simplicité d'établissement des règles et la possibilité de gérer des modèles ou des profils UML (SOA).

ATL© semble répondre à ces deux critères. En plus d'être un outil libre, les règles ATL© sont simples à mettre en œuvre. Son langage à base de variables et d'affectation le rend accessible. De plus, ATL©, qui est défini dans un IDE d'Eclipse, permet de générer des modèles UML (moyennant le méta-modèle d'UML) qui sont compatibles et utilisables dans *UML2 Tools* (un environnement Eclipse pour la gestion de modèles UML qui fait partie du projet Eclipse MDT[5] pour *Model Development Tools*). Enfin, un autre avantage d'ATL©, qui revêt toute son important dans la cadre de notre démarche, concerne la gestion des profils UML. En effet, ce langage permet d'introduire efficacement un profil UML (réalisé avec un éditeur UML) en entrée de la transformation (avec le fichier source de la transformation). Des instructions particulières d'ATL© permettent par la suite d'utiliser le profile UML pour typer les classes UML en sortie de la transformation.

2.4.2.2. La transformation de modèles avec ATL©

Dans cette section, nous présentons uniquement les caractéristiques de l'outil ATL©, qui nous ont été utiles dans l'implémentation de notre atelier. Une présentation plus détaillée du langage ATL©, est fournie dans [**Jouault 06**].

2.4.2.2.a. Langage déclaratif et impératif

Une des caractéristiques du langage ATL©, relève de son langage de programmation qui présente la particularité d'être hybride (déclaratif et impératif).

- La partie déclarative permet de faire correspondre directement un élément du méta-modèle source de la transformation avec un élément du méta-modèle cible de la transformation (*matched rule*). L'exemple ATL© suivant montre une transformation complètement déclarative :

[5] http://www.eclipse.org/modeling/mdt/?project=uml2

```
1  module UML2JAVA ;
2  create OUT : JAVA from IN : UML ;

3  rule Class2Jclass
4  {

5  from uclass : UML !Class
6  to jclass : JAVA !JClass(
7  uclass.name <- jclass.name
8  )
9  }
```

En ATL© une transformation s'appelle *module*. Le mot-clé OUT montre le méta-modèle cible *JAVA* (ligne 2). Le mot clé IN montre le méta-modèle source (*UML*) (ligne 2). La règle (*Class2Jclass*) du fichier de transformation (*UML2JAVA*) est déclarative. L'élément *class* (classe UML) du méta-modèle source va être traduit vers l'élément *jclass* du méta-modèle cible (ligne 6). On précise ensuite en utilisant l'opérateur de liaison ←, que l'attribut « *name* » de la nouvelle classe va être égal au nom de la classe UML source (ligne 7).

- La partie impérative d'ATL© complète ces correspondances directes entre éléments. Elle comporte des déclarations conditionnelles (*if…then…else...endif*), des déclarations de variables (*let VarName : varType = initialValue*), des déclarations de boucle (*while* (*condition*)…*do*), etc. Cette partie permet également de manipuler les éléments générés par les règles déclaratives (modification d'attributs, etc.).

De plus, ATL© fournit le mécanisme des *Helpers*. Ce sont simplement des mots-clés faisant références à des procédures stockées (qui peuvent ainsi être appelées dans une règle lorsqu'il y en a besoin). Les *helpers* servent à éviter la redondance de code et la création de grandes expressions dans une règle. Ceci induit aussi une meilleure lisibilité des programmes ATL©. Un *helper* en ATL© peut recevoir des paramètres et retourner une valeur. Les *helpers* constituent donc une bibliothèque de fonctions.

Les règles appelées (*called rule*) ressemblent aux *helpers*. L'exécution d'une règle appelée est déclenchée par son appel explicite depuis une autre règle. Ces règles s'exécutent pour générer des éléments dans le modèle cible après une première génération assurée par les règles déclaratives. Ces dernières s'exécutent en effet en même temps lors de l'exécution de la transformation. Il n'y a pas d'ordre d'exécution pour les règles déclaratives.

2.4.2.2.b. Gestion de méta-modèles avec ATL©

ATL© doit permettre de gérer les méta-modèles liés à la transformation. Il gère notamment les méta-modèles écrit sous *Ecore* [**Budinski** *et al.* **03**]. Ce dernier est un langage de méta-modélisation qui fait partie d'EMF (*Eclipse Modeling Framework*), résultat des efforts du projet Eclipse (*Eclipse Tools Project*). *Ecore* ressemble dans sa structure à un diagramme de classe UML. Il est basé sur des classes, des attributs, des associations pour lier les classes, des généralisations / spécifications entre classes, etc. *Ecore* présente donc un outil performant pour la réalisation de méta-modèles pour la transformation avec ATL©. Quelques outils permettent l'édition de méta-modèles conformes à *Ecore*, tels que : *Omondo*[6] ou bien l'éditeur *Ecore* fourni par EMF [**Budinski** *et al.* **03**].

[6] www.omondo.com

Une fois les deux méta-modèles de la transformation (source et cible) et le modèle source réalisés, il faut les déclarer en utilisant une fenêtre de configuration de la transformation. La figure (Fig V.6) montre l'utilisation de la fenêtre de configuration pour l'initialisation de la transformation *SOAgeneration BPMN2UML*. Les champs du haut *IN* et *OUT* font correspondre les modèles et méta-modèles de la transformation à leurs déclarations dans le code ATL©, alors que le champ du bas *Path Editor* fait correspondre ces déclarations à des fichiers portant une extension *.ecore* .

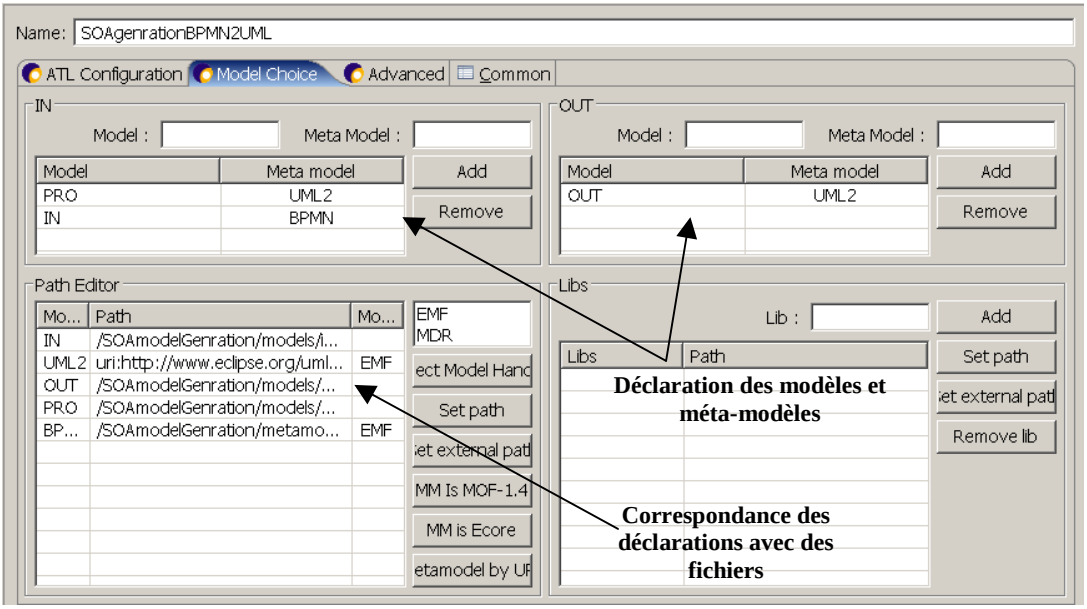


Fig. V.6. Initialisation d'une transformation avec ATL©

2.4.2.2.c. Gestion des profils UML avec ATL©

Une caractéristique importante, qui nous a orienté vers le langage ATL©, est sa capacité à gérer les profils UML. Dans ATL©, un profil UML se présente comme un fichier créé par un éditeur UML (TOPCASED©, etc.) et portant l'extension (.uml). En ATL©, on utilise un fichier profil UML conjointement avec le fichier du méta-modèle cible, qui doit être celui d'UML. La déclaration du profil UML se fait en même temps que la déclaration des méta-modèles de la transformation (ligne 2) :

```
   1  module nomDeModule; -- Module Template
   2  create OUT : UML2 from IN:Meta-modèle source, NomDeProfile : UML2;
…
```

L'application du profil ne peut commencer qu'une fois la partie déclarative du fichier de transformation exécutée (lorsque les classes UML sont générées). Elle se fait dans la partie impérative limitée par l'instruction do {…} (ligne 3) : la commande *applyProfile* permet d'appliquer le profil UML sur le modèle UML généré (ligne 5) :

```
   3  do{

   4  -apply the profile to the model created
   5  out.applyProfile(UML2!Profile.allInstances()->select(e | e.name =
      nomDeProfile').
   6  asSequence().first());
   7  ...
   8  }
```

Enfin, la commande *applyStereotype* permet d'appliquer un stéréotype de profil sur une classe UML générée au sein du modèle cible (ligne 9) :

```
…
   9  nomDeLaClasseUML.applyStereotype(nomDeLaClasseUML.
   10 getApplicableStereotype('nomDeStereotype'));
…
```

2.4.2.2.d. Injection de fichier XML

L'outil ATL© est fourni avec un module externe de *gestion globale de modèles*. Ce module est connu sous l'acronyme AM3 (*ATLAS MegaModel Management* ou gestion de méga-modèle ATLAS). Parmi les fonctionnalités offertes par ce module, nous nous intéressons à celle de l'injection d'un fichier XML externe dans l'espace technique d'une transformation (Fig V.7). En effet, ATL© ne peut pas manipuler directement un fichier XML, mais il peut manipuler un modèle XML qui obéit au méta-modèle de langage XML. L'injecteur sert donc à transformer directement un fichier XML (un fichier avec l'extension *.xml*) en un modèle XML (un fichier avec l'extension *.ecore*). Nous allons le voir plus loin, cette fonctionnalité est capitale dans l'implémentation de notre atelier.

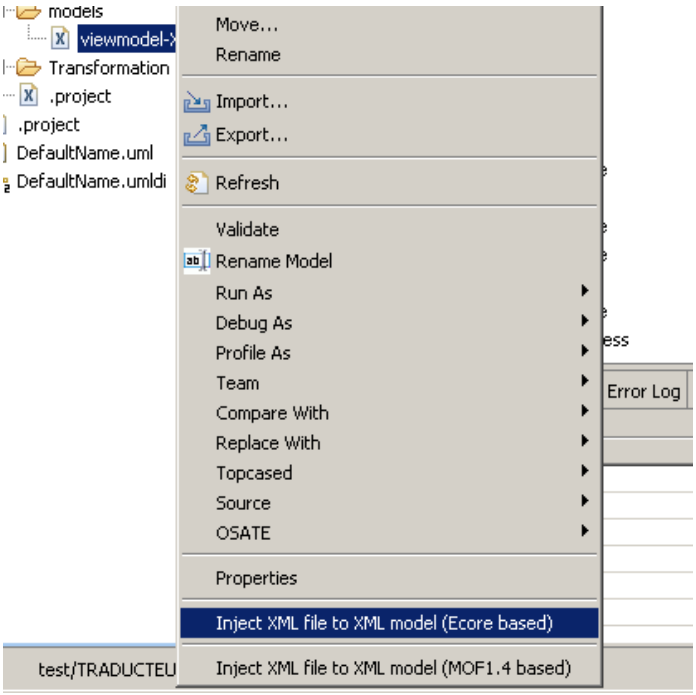


Fig. V.7. Injection d'un fichier XML avec ATL

2.4.3. L'éditeur UML de TOPCASED©

2.4.3.1. Choix de l'outil

Par choix global de promotion des logiciels libres (et donc par souci de cohérence avec les autres outils de cette chaîne logicielle), les éditeurs UML dits « produits commerciaux » (*IBM Rational rose*, *Borland Together*, etc.) ont été éliminés de notre liste dès le départ. Dans le monde du logiciel libre, plusieurs outils proposent un éditeur UML de qualité. Citons par exemple *StarUML*[7], *ArgoUML*[8], l'éditeur UML de *TOPCASED*© [9], etc. Ces

[7] www.staruml.com/

outils sont toujours en évolution, afin d'améliorer des aspects liés au graphisme et à la gestion des différents diagrammes UML. Un critère principal pour le choix de notre outil est la capacité de l'outil à lire nativement les fichiers *Eclipse UML2* portant l'extension *.uml*.

En effet, nous utilisons le méta-modèle d'UML proposé par Eclipse et intitulé *Eclipse UML2* comme méta-modèle de sortie pour nos transformations à travers l'outil ATL©. L'outil TOPCASED© est le seul à permettre de générer un diagramme de classe à partir d'un fichier *.uml*. Cela permet d'enrichir et d'intervenir facilement sur le fichier généré, via l'éditeur graphique de l'outil. L'autre avantage de cet outil est sa capacité à prendre en charge graphiquement les profils UML. Ainsi, les classes liées à un profil UML et visualisées avec TOPCASED© apparaissent avec l'adjonction d'un « stéréotype » les différenciant des classes UML non stéréotypées.

2.4.3.2. La visualisation de modèles avec TOPCASED©

Comme nous l'avons dit précédemment, la caractéristique déterminante pour le choix de TOPCASED© comme éditeur UML de notre atelier est sa capacité à lire les modèles UML, générés par ATL© et qui possèdent un profil associé, dans un diagramme de classe (Fig V.8). Comme tout éditeur UML, il est possible d'intervenir sur le modèle afin de l'enrichir avec de nouvelles classes, de nouveaux attributs, etc. Le résultat peut être enregistré dans un fichier diagramme de TOPCASED© qui porte l'extension *.umldi* (*UML Diagram Interchange*). La figure suivante (Fig V.8) montre la génération d'un diagramme de classe UML (.umldi) à partir d'un modèle UML (.uml) ouvert avec l'éditeur de base d' *Eclipse*. Cependant, une intervention manuelle est nécessaire pour l'organisation des éléments de diagramme de classe (le positionnement des différentes classes UML dans le diagramme). Nous avons la conviction que l'implantation d'un algorithme bien défini peut bien servir à organiser automatiquement les éléments d'un diagramme TOPCASED©.

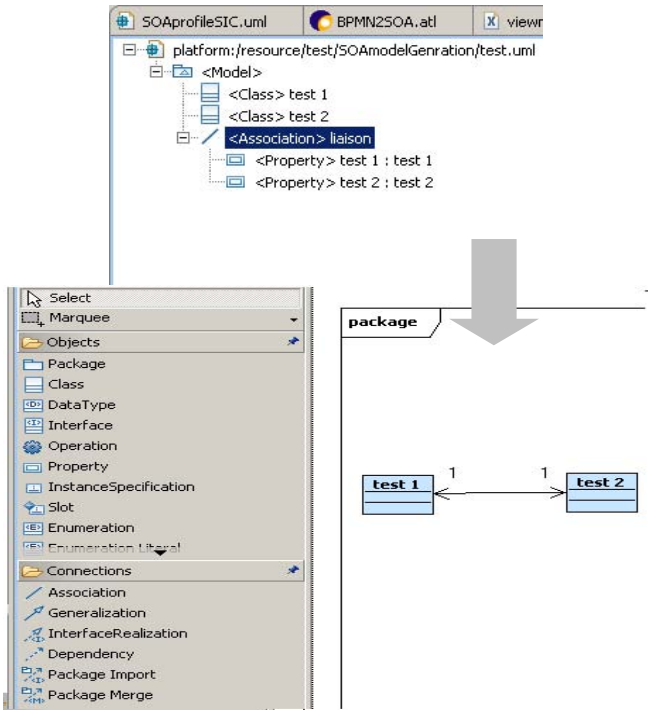


Fig. V.8. D'un modèle UML (.uml) à un diagramme de classe (.umdi)

[8] http://argouml.tigris.org/
[9] www.topcased.org

3. Implémentation des règles de transformation : du processus collaboratif au modèle UML

Après avoir défini, dans la section précédente, les outils composant notre atelier (Traducteur V1.0), nous présentons dans cette section le scénario détaillé de l'implémentation de notre atelier. Ce scénario décrit les différentes étapes (à partir de la modélisation du processus collaboratif jusqu'à l'obtention du modèle UML de Système d'Information Collaboratif en SOA). Nous détaillons également les différentes transformations que nous avons à réaliser au travers de l'outil ATL©.

3.1. Définition de scénario

Le scénario détaillé de fonctionnement de l'atelier est défini dans la figure suivante (Fig V.9). Ce scénario est décrit en quatre étapes :

- **première étape** : elle concerne l'injection de fichier XML du processus collaboratif (généré par Intalio©) dans l'espace technique des transformations ATL© en utilisant la fonctionnalité d'injection de fichier XML fournie par ATL© (Section 2.4.2.2.d). Le résultat de cette injection est un modèle XML bien structuré (qui respecte le méta-modèle standard de XML).

- **Deuxième étape** : elle correspond à la première transformation MDA de notre atelier, via l'outil ATL©. Nous transformons le modèle XML (fichier *ecore*) obtenu à l'issue de la première étape (et qui correspond au fichier XML obtenu à partir du modèle de processus BPMN réalisé avec Intalio©) en un modèle de processus collaboratif (fichier *ecore*) qui respecte le méta-modèle de processus collaboratif que nous avons défini dans le précédent chapitre de ce manuscrit.

- **Troisième étape** : elle correspond à la transformation MDA principale de notre atelier. C'est l'exécution des règles de transformation détaillées dans le chapitre précédent entre le méta-modèle de processus collaboratif et le méta-modèle de SIC (SOA). Le résultat de cette étape est un fichier UML (fichier *uml*) qui respecte le méta-modèle standard d'UML (fourni par *Eclipse*) et qui est « coloré » selon un profil UML de SIC (profil que nous avons conçu).

- **Quatrième étape** : le modèle UML, obtenu comme résultat de la troisième étape, peut être visualisé dans l'éditeur UML d' *Eclipse* sous forme d'une arborescence d'éléments UML (voir Fig V.9). Dans cette étape, nous convertissons le fichier UML en un diagramme de classes, visualisable et modifiable par l'éditeur UML de l'outil TOPCASED©.

La première étape correspond à l'application directe de la fonctionnalité native de l'outil ATL© d'injection d'un fichier XML, décrite dans la section *2.4.2.2.d* de ce chapitre. La quatrième étape est aussi l'application directe d'une fonctionnalité de base fournie par l'outil TOPCASED© et décrite dans la section *2.4.3.2* de ce chapitre. Nous nous intéressons dans la suite aux deuxième et troisième étapes du scénario de fonctionnement de l'atelier.

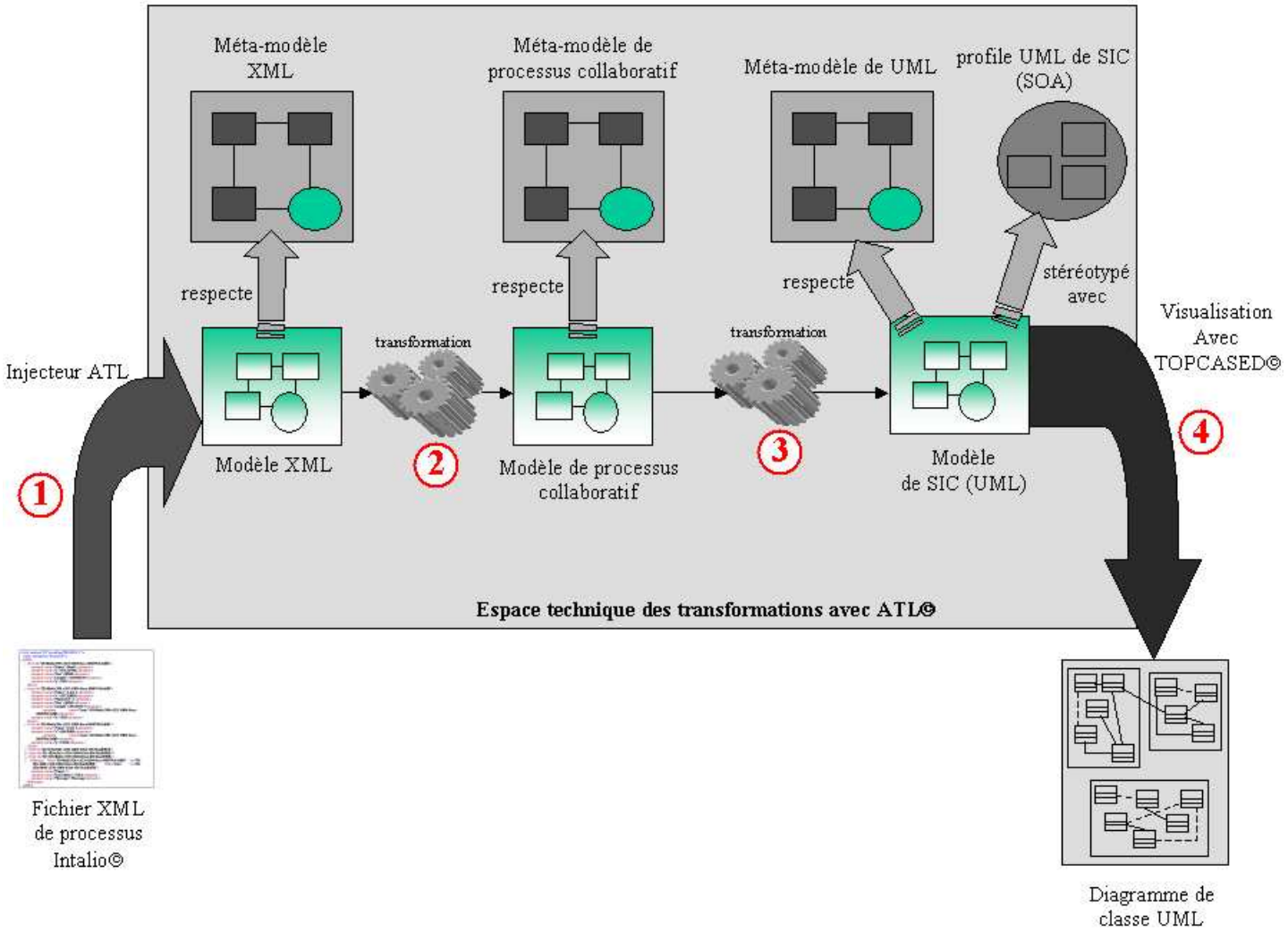


Fig. V.9. Scénario détaillé de l'implémentation de l'atelier des transformations Traducteur v1.0

3.2. Etape 2 : transformation de modèle XML vers le modèle de processus collaboratif

La transformation de modèle XML (obtenu à partir du modèle BPMN conçu sous Intalio©) vers un modèle de processus collaboratif est basée sur un fichier de transformation ATL© appelé *XML2Proc.atl*. L'exécution de ce fichier nécessite un ensemble de pré-requis.

3.2.1. Pré-requis de la transformation *XML2Proc.atl*

Les pré-requis de cette transformation sont :

- **le modèle de processus XML d'Intalio©** : c'est le fichier *viewmodelXML.ecore* que nous obtenons en utilisant l'injecteur d' ATL© (première étape du scénario).
- **Le méta-modèle standard de XML** : c'est le fichier *XML.ecore* qui décrit la structure d'un document XML, téléchargeable depuis la bibliothèque des méta-modèles *ecore* de projet *Generative Modeling Technologies* (GMT) d' *Eclipse* [10].
- **Le méta-modèle de processus collaboratif** : nous optons pour la réalisation de ce méta-modèle en utilisant l'éditeur *Eclipse Modeling Framework* (EMF) d'*Eclipse*[11]. Le fichier *MMprocCollaboratif.ecore* correspond au méta-modèle de processus collaboratif décrit dans le chapitre IV (Fig IV.6). La figure suivante (Fig V.10) montre un aperçu de ce fichier, ouvert dans l'éditeur EMF.

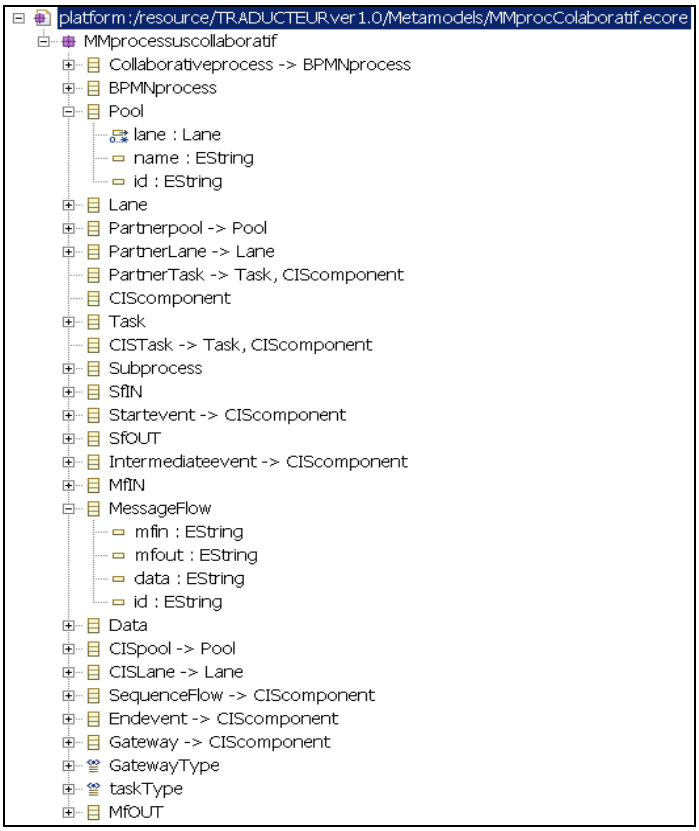


Fig. V.10. Méta-modèle de processus collaboratif modélisé avec l'éditeur EMF d'Eclipse

[10] http://www.eclipse.org/gmt/am3/zoos/atlantEcoreZoo/#XML.

[11] Nous aurions pu utiliser un éditeur UML et exporter par la suite dans le format *Ecore*.

L'initialisation de la transformation doit être faite en déclarant les différents fichiers cités ci-dessus, moyennant la fenêtre de configuration de la transformation (Fig V.6) de l'outil ATL©.

3.2.2. Le fichier de transformation *XML2Proc.atl*

Le but de ce fichier de transformation ATL© est de transformer le modèle XML d'Intalio© vers notre modèle de processus collaboratif. Le modèle XML respecte, en effet, le méta-modèle de XML, alors que l'exécution des règles de transformation décrites dans le chapitre précédent, nécessite de disposer d'un modèle conforme au méta-modèle de processus collaboratif. Le fichier de transformation *XML2Proc.atl* contient un ensemble de règles déclaratives et impératives ainsi que l'utilisation d'un certain nombre de *helpers*. Le modèle source est composé d'un ensemble d'éléments standard de BPMN (*pool*, *lane*, *task*, etc.), cf. Fig V.11.

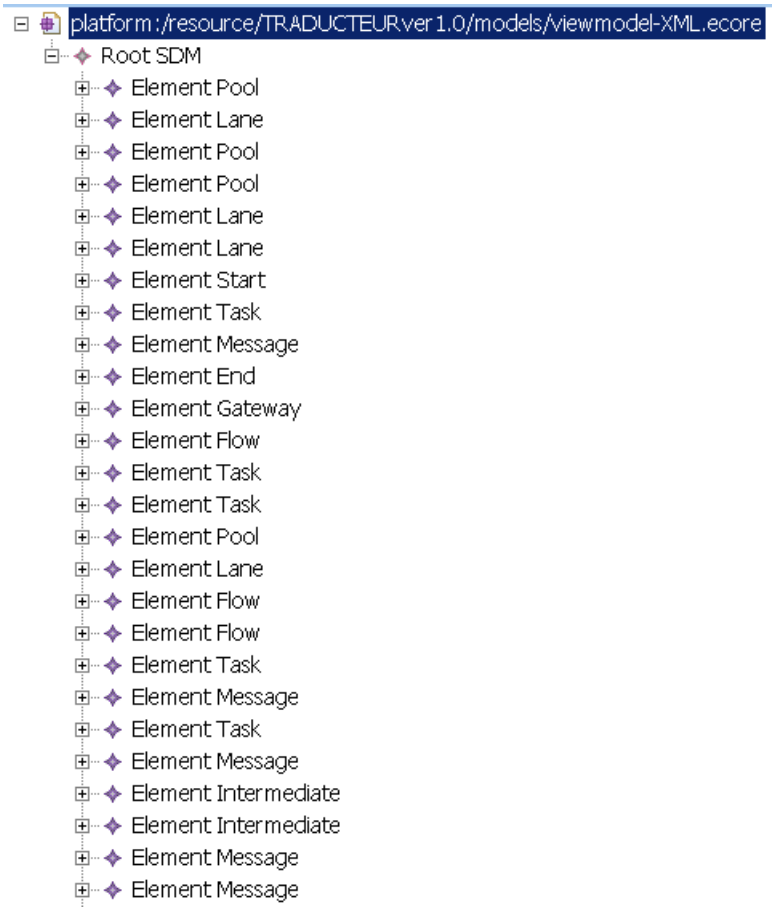


Fig. V.11. Aperçu d'un exemple de modèle XML d'Intalio

Nous identifions trois catégories de règles de transformation, que nous avons utilisées dans ce fichier de transformation :

3.2.2.1. Règles déclaratives

Un ensemble d'éléments du modèle cible est déduit directement (par correspondance directe) des éléments du modèle source. Nous citons *start event*, *intermediate event*, *gateway*, *message flow* etc. Nous présentons la règle déclarative d'ATL©, qui sert à générer l'élément *start event* dans le modèle cible :

```
1  --génération des startevent
2  rule startEvent {
3  from
4  e : XML!Element (
5  e.name = 'Start'
6  )
7  to
8  a : BPMN!startEvent (
9  id <- e.getId()
10 ,name<-e.getName3()
11 )
12 }
```

Comme nous pouvons le constater, l'élément *start event* du méta-modèle de processus collaboratif est déduit directement à partir de l'élément *Start* (ligne 5) du modèle source. Les *helpers getId()* et *getName3()* permettent respectivement de déduire à partir du modèle XML l'identifiant de l'élément et son nom dans le diagramme BPMN (lignes 9 et 10).

3.2.2.2. Règles déclaratives avec évaluation de condition

Un ensemble d'éléments du modèle cible est déduit directement d'un ensemble d'éléments du modèle source, (après avoir éventuellement validé une condition particulière). Le tableau suivant (Tab V.2) montre les éléments qui appartiennent à cette catégorie.

Eléments du méta-modèle source	Eléments du méta-modèle cible
pool	*CIS pool* ou *partner pool*
lane	*CIS lane* ou *partner lane*
task	*CIS task* ou *partner task*

Tab V.2. Éléments de modèle source et leurs correspondances

Nous présentons la règle déclarative d' ATL©, qui sert à déduire l'élément *CIS pool.*

```
--génération de la CISpool
1  rule CISPool {
2  from
3  e : XML!Element (
4  e.name = 'Pool' and
5  --si le nom est égale à Collaborative Information System
6  e.getName3()= 'Collaborative Information System'
7  )
8  to
9  a : BPMN!CISpool

10 (
11 name <- 'Collaborative Information System',
12 id <- e.getId()
13 )}
```

Ce code permet d'évaluer chaque élément *pool* trouvé dans le modèle source (lignes 3 et 4). Si le nom de cet élément est « *Collaborative Information System* » (ligne 6), alors il correspond à l'unique élément *CIS pool* de modèle cible (lignes 9 à 13).

3.2.2.3. Règles appelées

Un ensemble d'éléments du modèle cible est créé explicitement par le moyen d'une règle, appelée *called rule.* Le tableau suivant (Tab V.3) montre certains de ces éléments.

Eléments de méta-modèle source	Eléments créés de méta-modèle cible
message flow	*mf IN , mf OUT* et *data*
Sequence flow	*sf In* et *sf OUT*

Tab V.3. Éléments de modèle source et leurs correspondances

Nous présentons les règles (i) déclarative et (ii) appelée d'ATL© qui servent à déduire l'élément *data* dans le modèle cible :

```
-- règle déclarative pour création de message flow
1  rule MessageFlow{
2  from
3  e : XML!Element (
4  e.name = 'Message'
5  )
6  to
7  a : BPMN!MessageFlow (

8  …
9  id <- e.getFlowAttribute ('id'),
10 data <- e.getFlowName()

11 )
12 do {
13 --création de l'élement data
14 thisModule.CreateDataElement(a);
15 …
--fin do
16 }
--fin règle
```

La première règle qui est déclarative vise a créer un *message flow* dans le modèle cible à partir de l'élément XML *Message*. Dans le bloc impératif de cette règle *do {…}* (ligne 12), il est explicitement fait appel (ligne 14) à la règle *CreateDataElement()* (dont la syntaxe est décrite ci-dessous) afin de créer l'élément *data*.

```
-- Called rule pour créer les éléments data
19 rule CreateDataElement(a :BPMN!MessageFlow ) {
20 to
21 data: BPMN!Data (
22 name <-a.data,
23 messageflow <-a.id
24 )

25 }
```

La règle appelée (ligne 19), définie séparément, permet alors de créer l'élément *data* (ligne 22) et de le lier avec l'élément *message flow* (ligne 23).

3.3. Etape 3 : transformation du modèle de processus collaboratif vers le modèle de SIC (SOA)

La transformation du modèle de processus collaboratif vers le modèle de Système d'Information Collaboratif (SIC) est basée sur un fichier de transformation ATL© appelé *Proc2SIC.atl*. L'exécution de ce fichier nécessite un ensemble de pré-requis.

3.3.1. Pré-requis de la transformation *Proc2SIC.atl*

Les pré-requis de cette transformation sont :

- **le méta-modèle source de processus collaboratif :** c'est le même méta-modèle que celui identifié dans la section 3.2.1 de ce manuscrit.
- **Le méta modèle cible standard d'UML2 :** ce méta-modèle est écrit en *Ecore* et il décrit le langage UML2 dans sa dernière version[12]. Le fichier *UML2.ecore* est téléchargeable également depuis la bibliothèque des méta-modèles *Ecore* du projet *Generative Modeling Technologies* (GMT) d' *Eclipse* [13].
- **Le profil de SIC en UML :** le résultat habituel d'une transformation qui utilise le méta-modèle cible de UML2 est un modèle UML composé d'éléments : *classe*, *package*, *association*, etc. L'utilisation d'un profil UML associé à la transformation permet d'avoir des classes stéréotypées qui reflètent mieux la spécificité du système cible (dans notre cas, le SIC dans sa version SOA). Pour cette raison, nous avons inclus un profil UML du SIC. Ce profil est un fichier UML qui contient la liste des différents stéréotypes. Nous avons réalisé ce profil avec l'éditeur UML de TOPCASED©. Le résultat est le fichier *SOAprofileSIC.uml*. La figure (Fig V.12) donne un aperçu de ce profil, visualisé à l'aide de l'éditeur UML d'*Eclipse*.
- **Le modèle de processus collaboratif :** résultat de l'exécution du premier fichier de transformation ATL© (étape 2).

Tout comme lors de l'étape précédente, l'initialisation de la transformation doit être faite en déclarant les différents fichiers cités ci-dessus, moyennant la fenêtre de configuration de la transformation (Fig V.6) de l'outil ATL©.

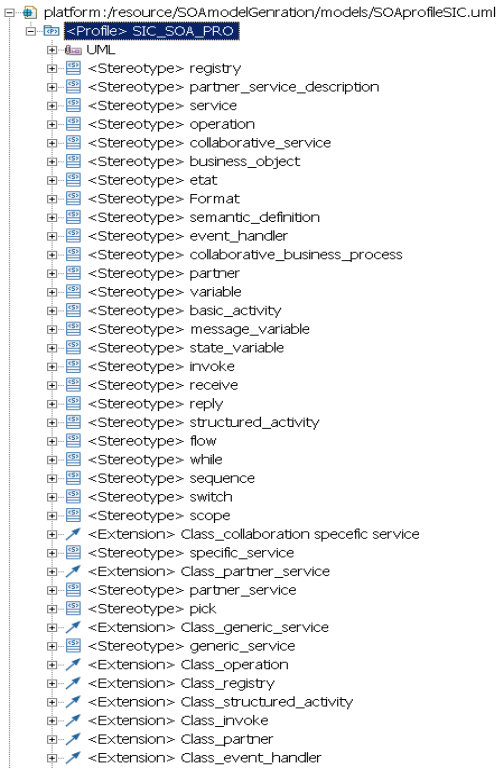


Fig. V.12. Aperçu du profil UML du SIC

[12] www.omg.org

[13] http://www.eclipse.org/gmt/am3/zoos/atlantEcoreZoo/#UML2

3.3.2. Le fichier de transformation Proc2SIC.atl

Le but de ce fichier de transformation ATL© est de générer le modèle du SIC à partir d'un modèle de processus collaboratif. Ce fichier contient un ensemble de règles déclaratives et impératives avec utilisation d'un certain nombre de *helpers*. La figure suivante (Fig V.13) présente les étapes cruciales de la construction du modèle de SIC, qui seront réalisées par l'exécution de ce fichier. Nous détaillons ces étapes dans la suite.

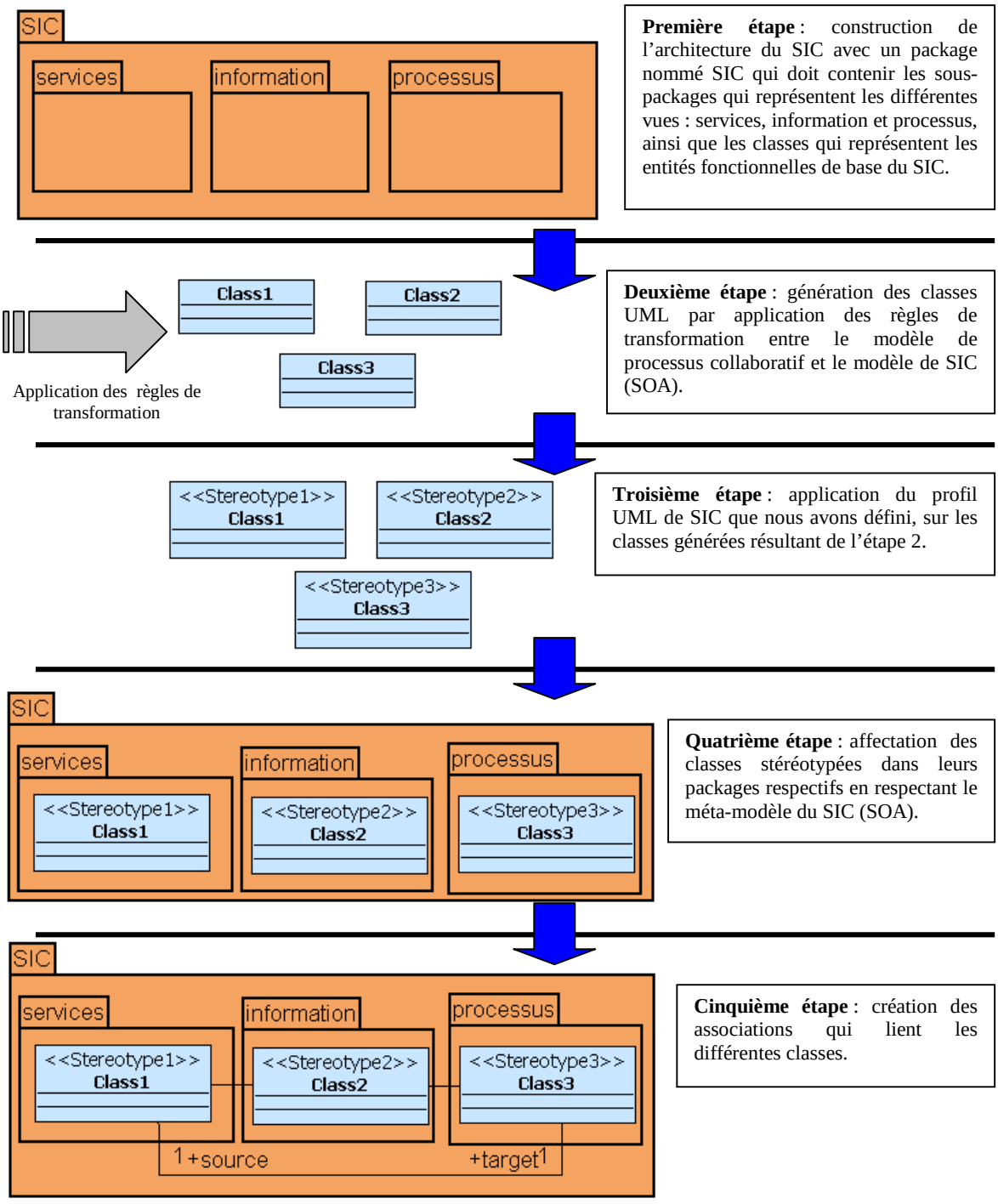


Fig. V.13. Cinq étapes cruciales pour la création du modèle du SIC à partir de modèle de processus collaboratif

3.3.2.1. Construction de l'architecture de SIC

L'inconvénient technique pour la réalisation de profil UML avec les versions actuelles de l'éditeur TOPCASED© et d'ATL© est qu'il est impossible d'y inclure l'architecture complète de notre modèle de SIC (les trois packages services, informations et processus et les différentes associations entre eux). Le profil UML que nous utilisons actuellement est un ensemble de stéréotypes qui ne sont pas liés entre eux. Le choix que nous avons fait est donc de coder directement cette architecture de SIC (en « dur ») directement dans le fichier de transformation. En effet, c'est une règle déclarative qui permet de faire correspondre l'élément racine de notre modèle de processus collaboratif (*collaborative process)* avec une composition de packages UML qui désigne l'architecture de notre SIC. L'avantage de ce choix est que la partie de code qui concerne cette architecture est facilement modifiable, car elle est totalement indépendante des autres règles du fichier (par exemple pour modifier les noms des packages UML).

```
--génération de l'architecture de SIC
1  rule generatePackages {
2  from
3  a : BPMN!Collaborativeprocess
4  to
5  out :UML2!Model
6  (
7  name<-'Collaborative information system',
8  packagedElement <- OrderedSet {services, information, processus}
9  ),
10 --génération de package service
11 services : UML2!Package(
12 name <- 'Service view'
13 , packagedElement <- OrderedSet {partners,cis}
14 ),
15 partners : UML2!Package (
16 name <- 'Partners services'
17 , packagedElement <- registry
18 ),
19 registry : UML2!Class (
20 name <- 'services registry'
21 ),
22 cis : UML2!Package (
23 name <- 'CIS services'
24 ),
25 --génération de package information
26 information: UML2!Package (
27 name <- 'Information view'
28 ),
29 --génération de package process
30 processus : UML2!Package  (
31 name <- 'Process view'
32 , packagedElement <- OrderedSet {basic,structured}
33 …
```

La règle *generatePackages* (ligne 1) permet d'assurer la génération des différents packages UML qui composent notre SIC (*services*, *information*, *process*, etc.). De plus, nous pouvons voir que ce code permet de créer une classe UML *services registry* (lignes 19 et 20) dans le sous-package *partners services* de package *services.* Cette classe désigne l'annuaire pour la description des services et elle fait partie directement de l'architecture de notre SIC comme entité de base.

3.3.2.2. Génération des classes UML de modèle de SIC

Comme il est décrit dans les règles de transformation, détaillées dans le chapitre précédent, un élément du méta-modèle source peut correspondre à plusieurs éléments dans le méta-modèle cible (par exemple *partner task* → *basic activity* | *partner service*). Pour illustrer ces correspondances, nous mettons en place, une couche de règles déclaratives, qui font correspondre directement les éléments des deux méta-modèles entre eux. A titre d'exemple, nous présentons la règle déclarative d'ATL©, qui sert à traduire l'élément *partner task* du modèle source de la transformation.

```
--génération des éléments cibles à partir de partner task
1  rule generatePartnerservices {
2  from
3  a : BPMN!PartnerTask
4  to
5  task :UML2!Class
6  (
7  name <- a.name
8  ),

9  service :UML2!Class
10 (
11 name <- a.name
12 ),

13 fiche :UML2!Class
14 (
15 name <- 'fiche__'+a.name
16 )

17 }
```

Cette règle montre que l'élément *partner task* génère trois classes UML du modèle SIC : la classe « *task* », qui correspond à *basic activity* du SIC (ligne 5), la classe « *service »,* qui correspond à *partner service* du SIC (ligne 9) et enfin la classe « *fiche »,* qui correspond à *partner service description* du SIC (ligne 13). Nous pouvons remarquer qu'à ce stade, les classes n'appartiennent pour le moment à aucun package du SIC. De même, la « coloration » avec un stéréotype n'a pas encore eu lieu.

3.3.2.3. Application du profil UML de SIC

Le profil de SIC que nous avons conçu sert à « stéréotyper » les simples classes UML que nous avons générées à l'aide de règles déclaratives. La déclaration de ce profil et son application dans le modèle UML sont conformes à ce qui a été présenté dans la section 2.4.2.2.c de ce manuscrit. L'application d'un stéréotype sur une classe UML permet à cette dernière de représenter un élément de modèle cible de nos transformations. Comme exemple, *partner task* se transforme en une classe UML portant l'un de ces trois stéréotypes (*receive*, *invoke*, *reply*) suivant la valeur de l'attribut *type* de *partner task*. Le code suivant illustre la transformation équivalente en ATL©.

```
-- application des profils pour l'élément partner task
  1  for (m in BPMN!PartnerTask.allInstances()->select(a|true))
  2  {
  3  --application de stéréotypes pour les activités de la vue processus
  4  if (m.type=#receive){
  5  thisModule.resolveTemp(m,'task').applyStereotype(thisModule.resolve
     Temp(m ,'task').getApplicableStereotype('SIC_SOA_PRO::receive'));
  6  }
  7  else if (m.type=#send){
  8  thisModule.resolveTemp(m,'task').applyStereotype(thisModule.resolve
     Temp(m ,'task').getApplicableStereotype('SIC_SOA_PRO::invoke '));
  9  }
  10 else if (m.type=#service)    {
  11 thisModule.resolveTemp(m,'task').applyStereotype(thisModule.resolve
     Temp(m ,'task').getApplicableStereotype('SIC_SOA_PRO::reply'));
  12 }
```

Une boucle *for*(){...} (ligne 1) permet de parcourir tous les éléments *partner task* du modèle source. Pour chaque élément et selon la valeur de l'attribut *type* (test avec *if*() *else*()) (lignes 4, 7 et 10), on déduit le stéréotype à appliquer sur la classe UML équivalente dans le modèle cible (obtenue par l'application de la première couche des règles déclaratives). Cette classe est récupérée à l'aide de la méthode native d'ATL© *resolveTemp()*. Cette méthode prend en entrée un élément du modèle source et retourne en sortie son équivalent dans le modèle cible. L'application de stéréotype se fait avec la méthode *applyStereotype()* (lignes 5, 8 et 11).

3.3.2.4. Affectation des classes UML dans leurs packages

Les classes UML générées à l'aide des règles déclaratives (voir *3.3.2.2*) et stéréotypées avec le profil que nous avons défini (voir *3.3.2.3*) doivent être localisées par rapport à l'architecture de SIC que nous avons définie (voir *3.3.2.1*). Pour cette tâche, nous utilisons l'opérateur *including*, qui permet d'inclure une classe UML particulière dans le contenu d'un package UML. Nous présentons un exemple afin de placer dans leur package les éléments résultant de la transformation de *partner task* du modèle source.

```
--placement des éléments résultats de la transformation de partner task

  1  for (m in BPMN!PartnerTask.allInstances()->select(a|true))
  2  {
  3  --affectation des basic activity dans la vue processus
  4  basic.packagedElement<-basic.packagedElement-
     >including(thisModule.resolveTemp(m
  5  ,'task').debug('tt'));
  6  --affectation des fiches de description de service dans la vue
     services
  7  partners. packagedElement <-  partners.packagedElement-
     >including(thisModule.resolveTemp(m ,'fiche').debug('tt'));
  8  affectation des partner service dans la vue services
  9  partners.packagedElement <-  partners.packagedElement-
     >including(thisModule.resolveTemp(m ,'ser').debug('tt'));
  10 }
```

Une boucle *for*(){...} permet de parcourir tous les éléments *partner task* (ligne 1) du modèle source. Pour chaque élément, on déduit l'élément équivalent dans le modèle cible (*resolveTemp*) et on le place dans le package UML associé (ligne 4,7 et 9).

3.3.2.5. Association entre deux classes générées

Une fois les classes UML générées, stéréotypées et incluses dans l'ensemble des packages qui forment l'architecture de SIC, l'étape suivante consiste à lier les différentes classes entre elles par le biais d'associations *UML*. Les associations illustrent, en effet, la transformation d'un lien logique de dépendance entre des éléments du modèle de processus collaboratif (inclusion, liaison, etc.) en un lien de dépendance entre des éléments du modèle *UML* de SIC. Par exemple, deux tâches *BPMN*, qui se suivent au travers d'un *sequence flow*, se transforment en un élément *sequence* et deux éléments *basic activity* de modèle SIC. Le but ici est de lier les deux éléments *basic activity* avec l'élément *sequence*, moyennant deux associations *UML*. Les associations *UML* permettent alors de concrétiser **les règles de liaison** que nous avons détaillées dans le chapitre précédent. Nous définissons une règle appelée (*called rule*) *CreateAssociation*, qui prend en paramètres les deux classes source et cible à lier par une association. Cette règle sera appelée à chaque fois qu'il y'a besoin, pour lier deux classes de notre modèle de SIC. Toutefois, nous avons voulu proposer une version avancée de cette règle en gérant aussi les cas de composition entre éléments *UML*. Le code ATL© suivant, présente cette règle.

```
--Régle de base pour la création des associations--

1  rule CreateAssociation(src : UML2!Class, target : UML2!Class, mo:
   UML2!Model,
2  namee: String, firstagreg: Integer, secondagreg: Integer,
3  nameSource: String, nameTarget: String)

4  {
5  to
6  association: UML2!Association (
7  name <- namee,
8  memberEnd <- Set {srcEnd,targetEnd},
9  ownedEnd <- srcEnd,
10 ownedEnd <- targetEnd
11 ),

12 srcEnd: UML2!Property (
13 name <- nameSource,
14 type <- src,
15 aggregation <- #none
16 ),

17 targetEnd: UML2!Property (
18 name <- nameTarget,
19 type <- target,
20 aggregation <- #none
21 )
22 do {

23 if (firstagreg=1)
24 srcEnd.aggregation <- #composite;
25 if (secondagreg=1)
26 targetEnd.aggregation <- #composite;

      a. add associations to the source's class package
27 mo.packagedElement <-mo.packagedElement.append(association);}}
```

Nous pouvons constater qu'une association *UML* est définie par ses deux extrémités *srcEnd* et *targetEnd* (lignes 12 et 17). Ces deux extrémités définissent les rôles de la relation (lignes 13 et 18) et permettent de définit sa nature (association ou agrégation). Par

défaut les associations sont simples. Mais si les variables *firstagreg* (respectivement *secondagreg*) sont égales à « 1 », la première (resp. seconde) extrémité devient porteuse d'une agrégation (lignes 23 à 26). La règle *CreateAssociation* doit être appelé à partir de la partie impérative d'une autre règle. Par la suite, nous proposons de présenter le code qui permet d'associer l'élément *services registry* du modèle du SIC (annuaire décrivant les services des partenaires) avec les éléments *partner service description* (fiche descriptive de service) qui sont obtenus directement par le biais des règles déclaratives, à partir des éléments *partner task* du modèle source. Cette relation est en fait une agrégation. En effet, l'annuaire des services est composé d'un ensemble de fiches descriptives des services.

```
1  for (m in BPMN!PartnerTask.allInstances()->select(a|true))
2  {
3  --lier l'élément registry avec les fiches générées
4  thisModule.CreateAssociation(registry,thisModule.resolveTemp(m
   ,'fiche'), partners,'appartient a',1,0,'contient','appartient à');
5  }
```

Une boucle *for*(){...} (ligne 1) permet de parcourir tous les éléments *partner task* du modèle source. Pour chaque élément, on déduit l'élément *partner service description* équivalent dans le modèle cible (à l'aide de la commande *resolveTemp*) (ligne 4). L'appel à la règle *CreateAssociation* permet finalement d'associer l'élément *partner service description* avec l'élément *registry* (ligne 4). Nous pouvons voir que le paramètre *firstagreg* de la règle *CreateAssociation* est égal à 1 (ligne 4). Cela implique que cette association est une agrégation.

4. Expérimentation avec un exemple de processus collaboratif : plate-forme d'aide à l'achat

Dans cette section de chapitre, nous présentons une expérimentation de notre atelier avec un exemple de processus collaboratif. Cet exemple a été présenté dans le deuxième chapitre du manuscrit et décrit une collaboration entre un client et deux fournisseurs, dans le cadre d'une plate-forme d'aide à l'achat[14] qui permet l'externalisation de la fonction achat. Nous commentons des impressions d'écran relatives au fonctionnement de notre atelier (décrit dans *3.1*).

4.1. Modélisation de processus collaboratif

La figure (Fig V. 14) montre le modèle de processus collaboratif de la plate-forme d'aide à l'achat, une fois que le modèle de processus est finalisé. Le fichier XML est injecté dans l'espace technique des transformations ATL©. Par la suite, nous exécutons les deux transformations citées dans ce chapitre, qui correspondent aux deux fichiers de transformation *XML2Proc.atl* et *Proc2SIC.atl*.

[14] Afin de simplifier la réprésentation du processus, nous considérons un client et deux fournisseurs dans le processus collaboratif.

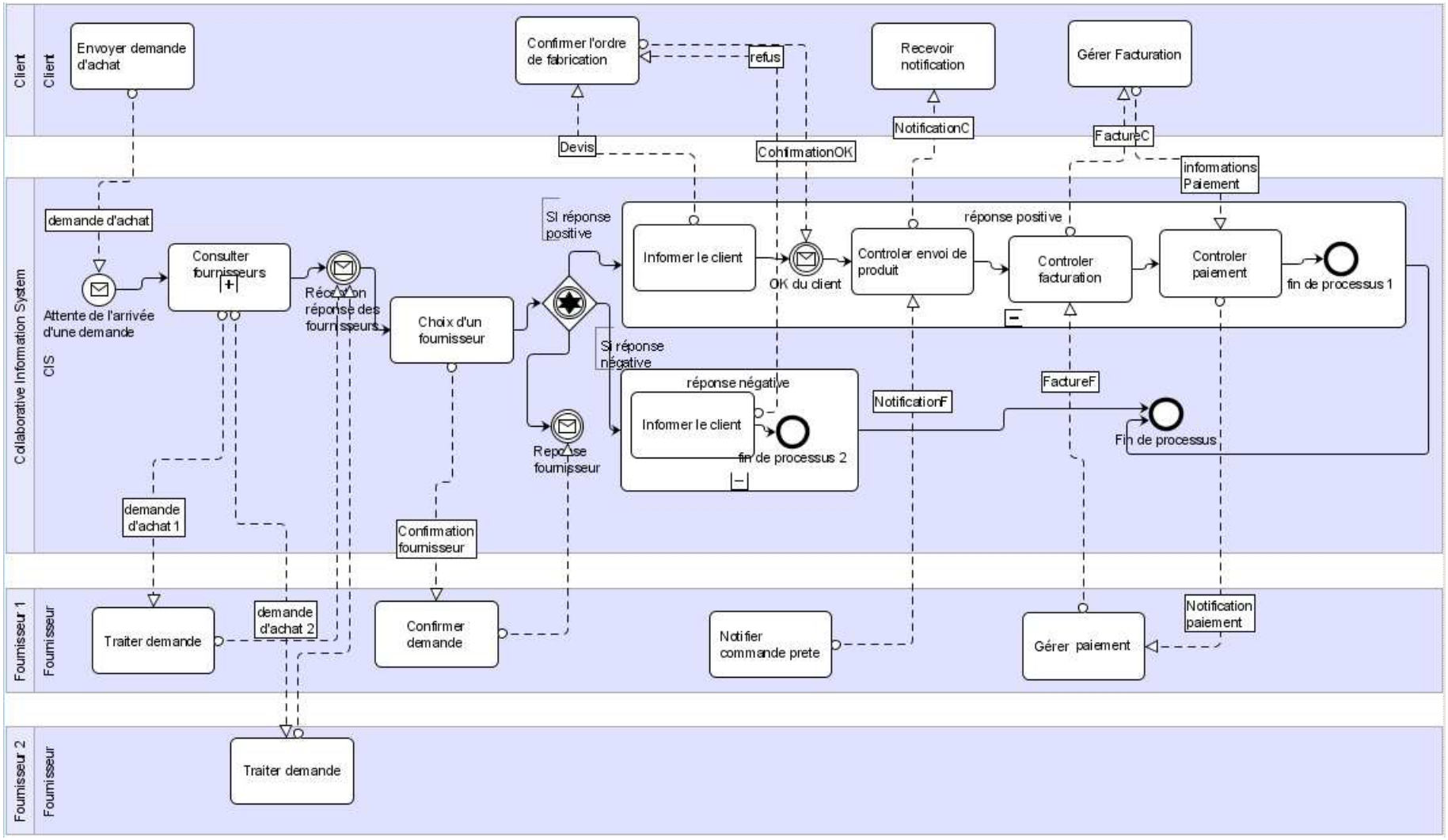


Fig. V.14. Copie d'écran de modèle de processus réalisé avec l'outil Intalio Designer©

4.2. Génération de modèle SIC (UML)

La figure (FigV.15) montre un aperçu du résultat généré après application des deux transformations ATL© : le fichier UML *SOAmodel.uml* décrit le modèle de SIC qui supporte l'exécution du processus collaboratif. Ce fichier est ouvert dans l'éditeur UML d'*Eclipse*. Ce dernier montre une description en arborescence des différents éléments UML (stéréotypés) qui composent le modèle. La figure (FigV.16) ne montre pas cependant la totalité du modèle, à cause de nombre important des éléments UML du modèle. Nous pouvons voir l'architecture de celui-ci : le package racine *Collaborative information system*, qui est composé des autres packages *service view*, *information view* et *process view*. Comme exemple de l'application de nos règles de transformation, le package *CIS services*, qui fait partie du package *service view*, contient les différents services (spécifiques à cette collaboration) et gérés par le SIC. C'est l'application de la règle de transformation **Rs1** *CIStask* → *CISservice* définie dans le chapitre précédent (voir Chapitre *IV.4.4.1.1*). La tâche *contrôler facturation*, par exemple, de processus BPMN devient un service spécifique de même nom dans le modèle UML.

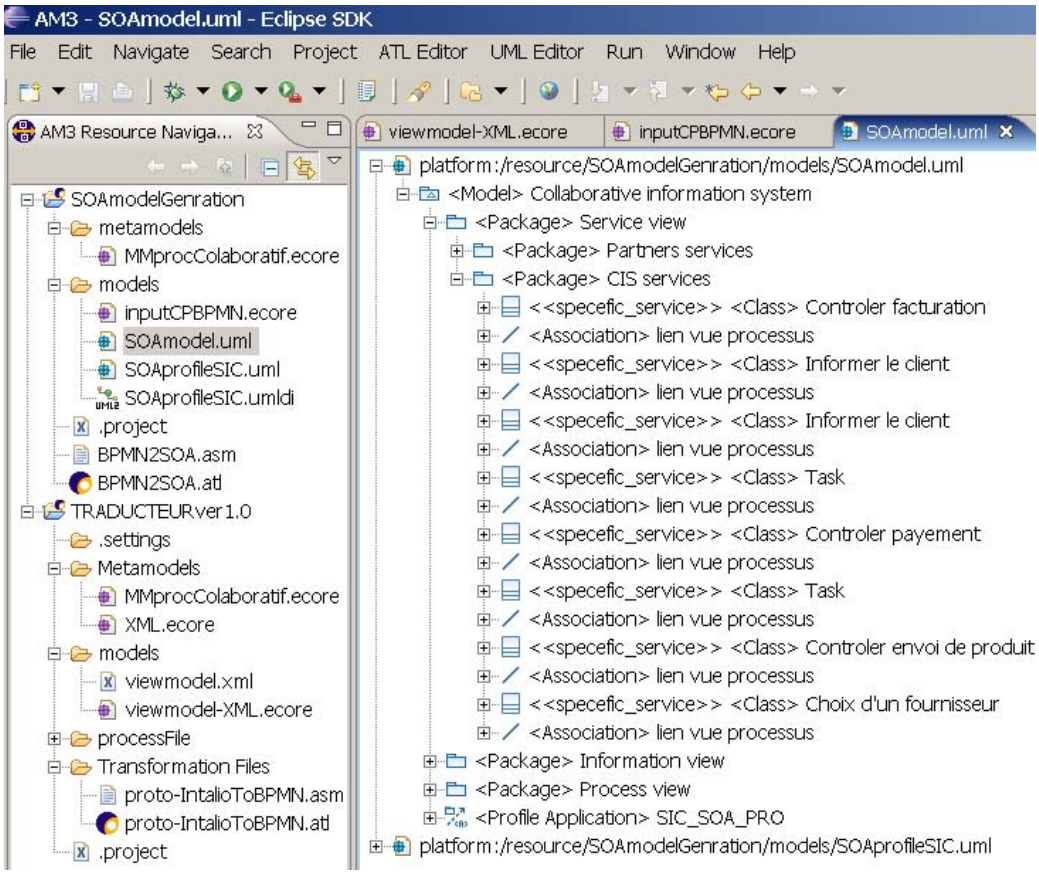


Fig. V.15. Impression d'écran du modèle UML de SIC (SOA) visualisé avec l'éditeur UML d'Eclipse

La figure (Fig V.16) montre un aperçu du résultat de l'ouverture du fichier *SOAmodel.uml* avec l'éditeur UML de TOPCASED© en tant que diagramme de classe. Cependant, TOPCASED© ne permet pas d'organiser automatiquement les éléments d'un diagramme de classe. Un effort manuel est à fournir pour accomplir cette tâche. Le diagramme de classe complet, qui correspond à notre modèle SIC, est très chargé, vu le nombre important de classes et d'associations (*effet spaghetti*). Nous avons supprimé une partie des classes

UML de notre modèle, afin d'améliorer la lisibilité de ce diagramme qui représente le modèle de SIC.

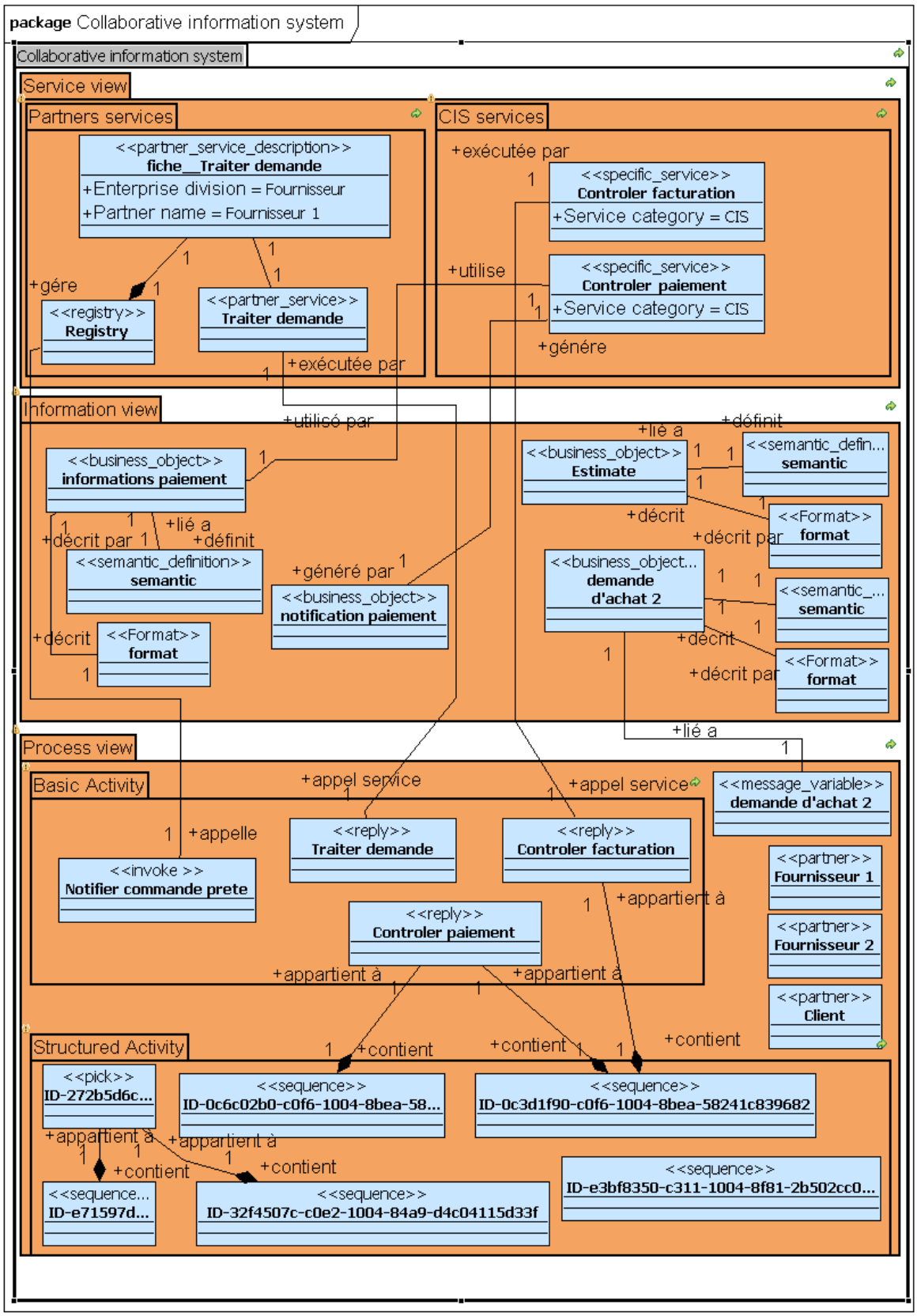


Fig. V.16. *Copie d'écran de modèle UML de SIC ouvert avec l'éditeur UML de TOPCASED*

L'intérêt principal de ce diagramme de classe est qu'il représente les différentes associations existantes entre les différentes classes UML :

- les activités « Contrôler facturation » et « Contrôler paiement » du package *Basic Activity* sont liées par l'élément *sequence* du package *Structured Activity*. Cela explique le séquencement de leur exécution dans le processus collaboratif, en conformité avec le modèle BPMN de départ.
- L'activité « Contrôler facturation » du package *Basic Activity* est lié avec le service spécifique du SIC (stéréotype *specific service*) « Contrôler facturation ». Cela montre que l'exécution de l'activité fait appel au service du SIC directement associé.
- L'activité « Traiter demande » du package *Basic Activity* est liée avec le service d'un partenaire de la collaboration (stéréotype *partner service*) « Traiter demande ». Cela montre que l'exécution de l'activité fait appel au service de partenaire directement associé.
- Le service spécifique du SIC « Contrôler paiement » est lié (association *utilise*) avec l'objet métier (stéréotype *business object*) « informations paiement », afin de montrer que ce service a besoin de cet objet métier pour pouvoir être exécuté. Le même service est lié (association *génère*) avec l'objet métier (stéréotype *business object*) « notification paiement », pour montrer que ce service génère en retour cet objet métier.
- L'activité « Notifier commande prête » du package *Basic Activity*, qui est de type *invoke*, est liée avec la classe « Registry » du package *partners services*. Le SIC a besoin de consulter le registre des services des partenaires pour pouvoir exécuter ce service.

Nous présenterons ensuite un aperçu détaillé de chacun des packages composant le diagramme. L'inconvénient est qu'il est impossible, en descendant de niveau, de voir les associations avec les autres packages.

4.2.1. Génération de la vue « services »

La figure (Fig V.17) donne un aperçu du package *CIS services*, qui montre les services gérés par le SIC. Les différentes tâches BPMN appartenant à la *pool* du SIC sont transformées en des services spécifiques.

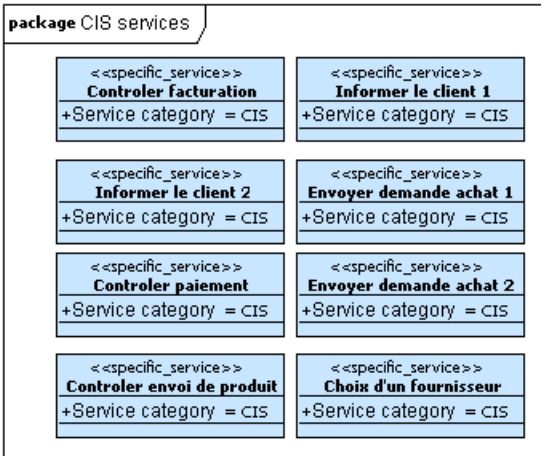


Fig. V.17. Le package « CIS services » du modèle de SIC

La figure (Fig V.18) donne un aperçu du package *partners services*, qui montre les différents services des partenaires de la collaboration. Nous pouvons voir que l'élément « Registry » (qui fait partie directement de l'architecture de SIC) est lié à des fiches descriptives (stéréotype *partner_service_description*) de tous les services (stéréotype *partner_service*) gérés par les partenaires. Chaque fiche décrit le propriétaire du service (attributs *Partner name* et *Enterprise division*) et doit être complétée par des informations additionnelles.

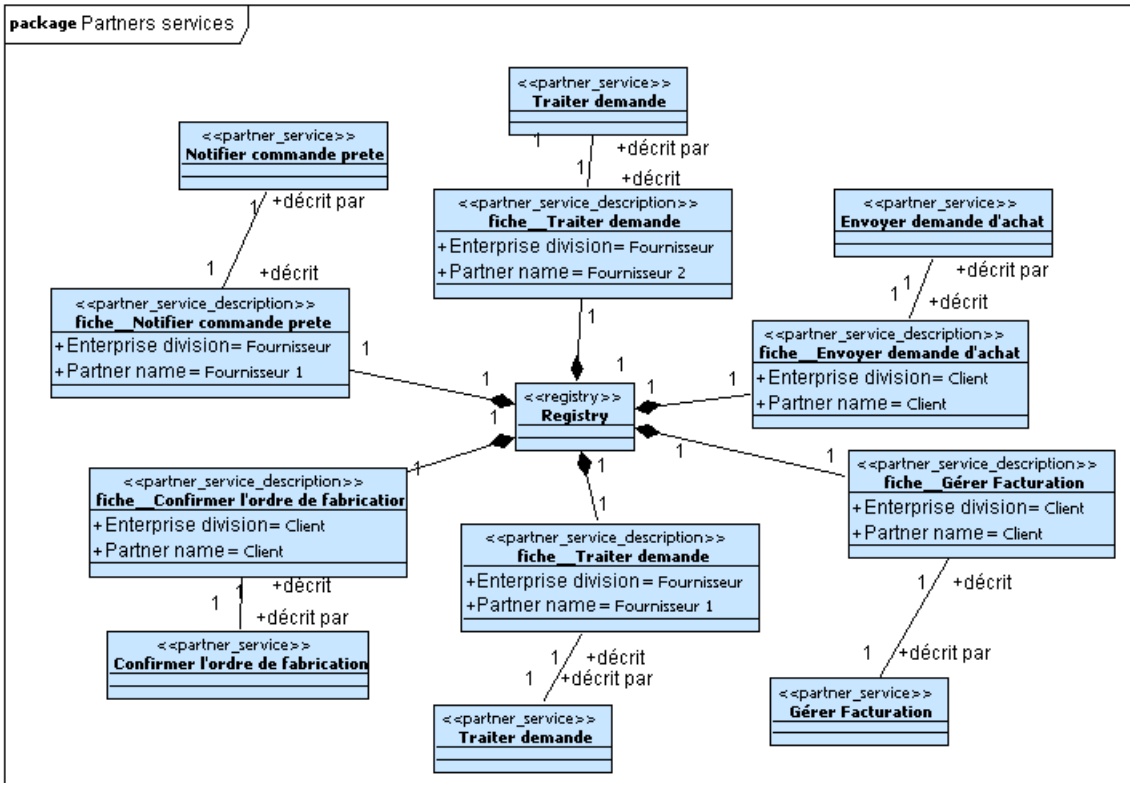


Fig. V.18. *Le package « Partners services » du modèle de SIC*

4.2.2. Génération de la vue « informations »

La figure (Fig V.19) donne un aperçu du package *Information view* qui décrit les objets métier manipulés lors de l'exécution de processus collaboratif. Chaque objet métier (stéréotype *business_object*) est lié à une classe « format », qui désigne la structure de l'objet et une classe « semantic », qui désigne une description sémantique de l'objet. Ces informations additionnelles sont à compléter ultérieurement par le concepteur du SIC.

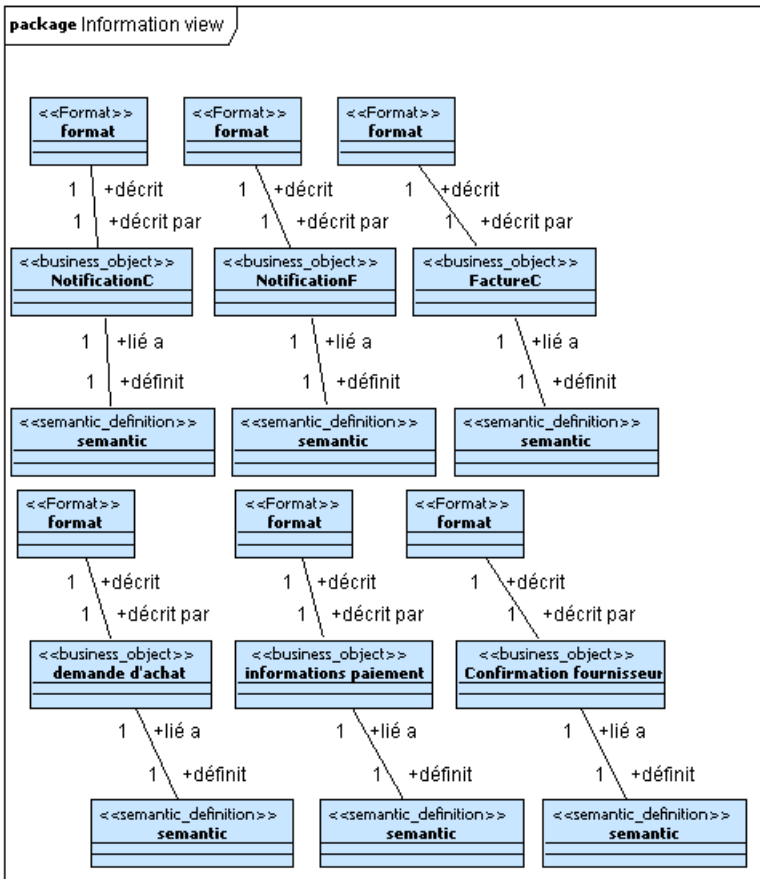


Fig. V.19. *Le package « Information view » du modèle de SIC*

4.2.3. Génération de la vue « processus »

La figure (Fig V.20) donne un aperçu du package *Basic Activity,* qui décrit les activités de base, nécessaires pour l'exécution du processus collaboratif. Nous pouvons voir ici l'application de l'algorithme décrit dans la section *3.3.2.3* qui concerne la déduction du stéréotype à appliquer (*invoke*, *reply* ou *receive*) sur l'activité selon le type des *task* BPMN du processus. De plus, nous pouvons voir la transformation de *start event* « Attente de l'arrivée d'une demande » en une activité (stéréotype *receive*). Les *intermediate event* du processus (« Ok du client », « Réponse fournisseur », etc.) sont aussi transformés conformément aux règles en activité (stéréotype *receive*). Quant aux *end event* (« fin du processus »), ils sont transformés en activités (stéréotype *invoke*).

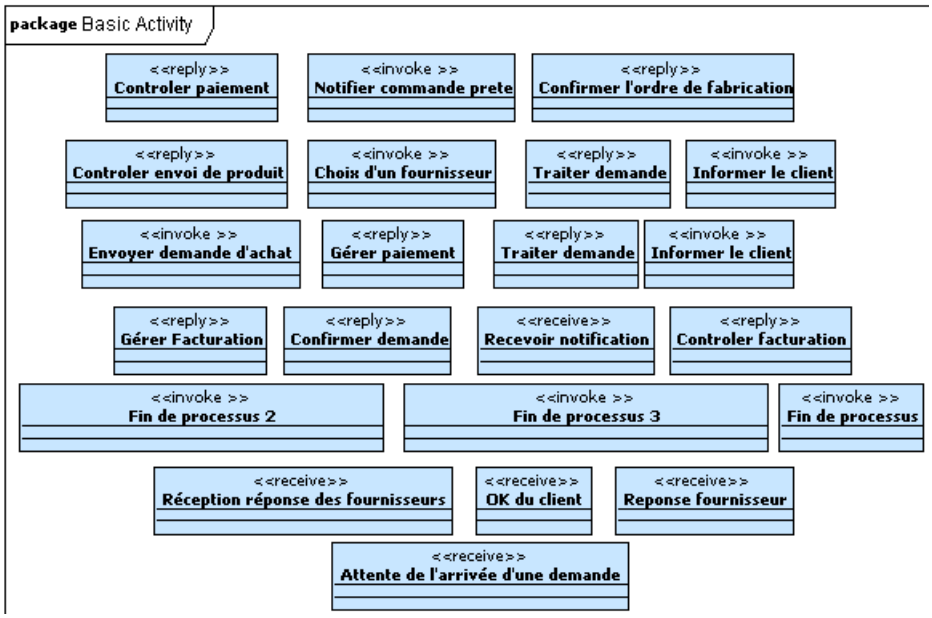


Fig. V.20. Le package « Basic Activity » du modèle de SIC

La figure (Fig V.21) donne un aperçu de package *Structured Activity*, qui décrit les activités qui structurent le flux d'exécution du processus collaboratif. Nous pouvons voir que les *sequence flow* du processus BPMN se sont transformés en des éléments « sequence » qui lient les différents éléments du modèle de SIC. Les élements « scope » désignent les différents sous-processus du processus. Enfin, le *gateway* de type *event based exclusive gateway* est transformé, conformément à la règle **Rp4** définie dans le chapitre précédent (voir Chapitre *IV.4.4.1.3*), en l'élément « pick ».

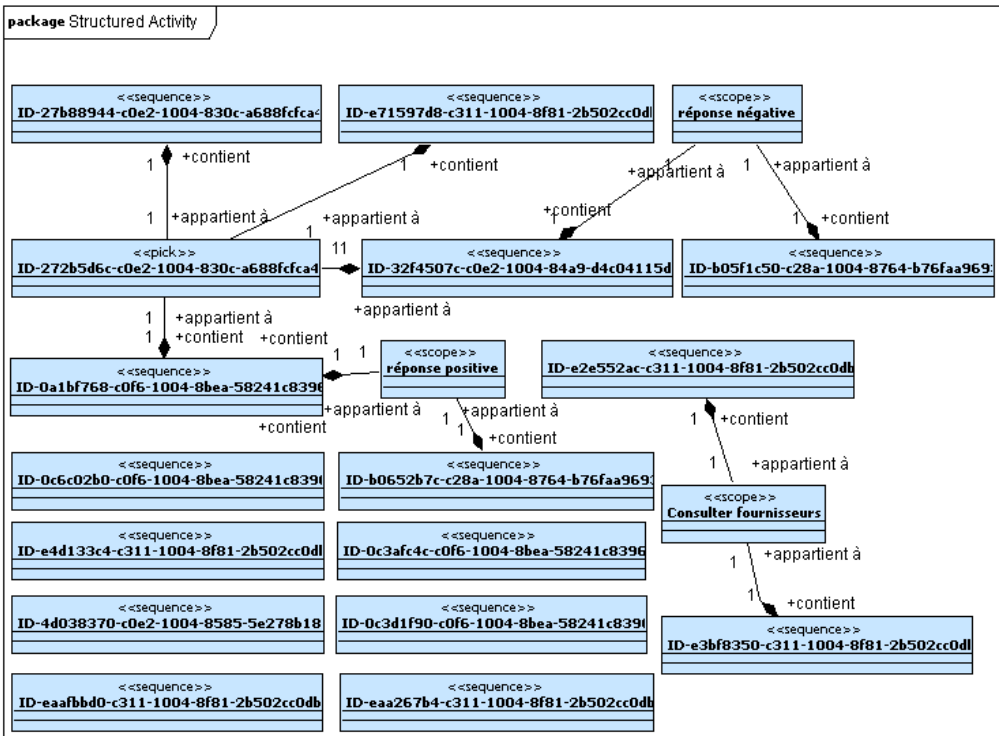


Fig. V.21. Le package « Structured Activity » du modèle de SIC

5. Conclusion du chapitre

Dans ce chapitre, nous avons présenté une implémentation et une expérimentation de l'atelier qui permettent de mettre en application nos résultats présentés dans le chapitre précédent sur la génération d'un modèle de SIC à partir d'une définition de processus. L'atelier logiciel est basé sur l'utilisation de trois outils différents : Intalio Designer© pour la modélisation des processus collaboratifs, ATL© pour la transformation des modèles et l'éditeur TOPCASED© pour la visualisation des diagrammes UML générés. Ces outils sont tous basés sur l'environnement de développement intégré IDE Eclipse©. De plus, ces outils sont des logiciels libres (leurs codes sources sont disponibles en accès libre).

Nous avons détaillé dans ce chapitre le scénario de fonctionnement de notre atelier (transformation, manipulation de fichier XML, etc.). Les caractéristiques principales des transformations ATL© que nous avons développées ont été présentées (construction de l'architecture de SIC, utilisation d'un profil UML, etc.). Enfin, nous avons également expérimenté notre prototype sur un exemple de processus collaboratif, celui d'une plateforme d'aide à l'achat. Nous avons présenté et commenté des impressions d'écran qui illustrent les résultats générés par notre outil. Toutefois, il devient intéressant et pertinent de tester et d'expérimenter notre atelier sur un périmètre plus élargi.

Concernant les améliorations / évolutions à introduire dans l'atelier, nous pouvons citer les deux suivantes :

- la première amélioration à faire concerne surtout la mise en place d'un « *plugin* » *Eclipse*, qui permette de générer le modèle du SIC dans le même répertoire que le modèle du processus. Actuellement, nous effectuons manuellement l'injection du fichier du processus dans l'espace technique de ATL©.

- La deuxième amélioration concerne l'utilisation d'un éditeur BPMN, qui corresponde directement à notre méta-modèle de processus collaboratif. Cela nous éviterait d'effectuer la première transformation ATL© détaillée dans ce chapitre. L'utilisation de l'outil *Graphical Modeling Framework* [15] (GMF) d'*Eclipse* semble être une solution. L'outil Intalio© reste cependant avantageux (qualité de la modélisation graphique, vérificateur de grammaire BPMN, etc.).

[15] www.eclipse.org/gmf.

Chapitre VI

Conclusion et perspectives

1. Rappel du cadre des travaux

Le cadre général de nos travaux porte sur la notion de l'intégration d'entreprises, en tant que stade ultime de la collaboration d'organisations. Après nous être intéressés, d'une part, aux différents aspects de la collaboration (en termes de niveaux, mais également de stades de maturités ou de concepts logique / technique participant de son établissement), et d'autre part à la relation étroite, et parfois fusionnelle, que l'entité entreprise entretient avec son système d'information (tout au moins d'un point de vue externe), nous avons porté l'essentiel de notre effort sur l'interopérabilité des systèmes d'information. La réponse à cette question nous semble effectivement constituer une clé pertinente et légitime de la résolution de la problématique supérieure que constitue l'intégration des entreprises.

Nous avons décidé d'aborder ce sujet de l'interopérabilité des SI selon un double questionnement (hérité du principe « *produit / processus* »): *quelle solution conceptuelle pour répondre théoriquement à ce besoin ?* Et *comment concevoir, à un niveau logique tout au moins, cette proposition de solution ?*

Notre réponse à la première question propose de permettre aux SI des entreprises de se fondre au sein d'un système de systèmes d'information en limitant les contreparties au respect, par ces SI, des principes SOA. Ce systèmes de SI présente la particularité de se baser sur un système d'information médiateur, désigné par l'acronyme SIC (système d'information collaboratif) qui, également basé sur les préceptes SOA, constitue le réel fournisseur d'interopérabilité en s'occupant du partage des données, de la gestion des applications et de l'orchestration des processus collaboratifs.

Pour répondre à la seconde question, nous nous sommes donc, naturellement, tournés vers les moyens de réussir la réalisation de ce SIC. En tant que système d'information, toute la question revient à réussir la transition allant du niveau d'abstraction métier jusqu'aux niveaux logique et physique. Conscients de l'intérêt que représente une approche comme le *Model-Driven Approach* pour la conception de ce type de système, nous avons essayé de nous concentrer sur le cheminement permettant de passer d'un modèle *métier* « *Computer-Independent* » (CIM) à un modèle *logique* « *Platform-Independent* » (PIM). Cette ambition nous a amené à considérer les correspondances conceptuelles existant entre les domaines des modèle de processus, exprimés à l'aide du formalisme BPMN (vus tant que CIM), et les modèles de systèmes informatiques formalisés en UML (vus en tant que PIM). Nous avons ainsi pu proposer un certain nombre de règles de traduction, implémentées au sein d'une maquette d'atelier de traduction « MDA », permettant au final de proposer la génération d'une architecture logique de SI Médiateur spécifique, assurant l'interopérabilité de SI SOA partenaires mais hétérogènes dans le cadre d'un processus collaboratif déterminé.

2. Synthèse sur les apports de ces travaux

1. Caractérisation de l'interopérabilité des systèmes d'information

Dans cette thèse, nous avons montré que l'interopérabilité des systèmes d'information est connexe à celle des entreprises. En effet, un système d'information présente la couche de liaison entre les décisions / opérations de l'entreprise (dans un cadre interne) et l'interface de communication de l'entreprise (dans un cadre externe). Fort de ce constat, nous pensons que l'interopérabilité des systèmes d'information n'inclut pas les aspects liés à la décision ou à l'exécution interne des opérations de l'entreprise (ils font partie de l'interopérabilité des entreprises). Nous avons considéré dans nos travaux trois niveaux différents pour l'interopérabilité des systèmes d'information : un niveau métier (organisation des partenaires et description de leurs échanges), un niveau sémantique (correspondance conceptuelle entres les ressources des partenaires : données, processus et applications) et un niveau technique (capacité à mettre en communication les systèmes informatiques qui composent les systèmes d'information). Dans un contexte d'interopérabilité des systèmes d'information, l'information doit être échangée d'une manière **organisée** (niveau métier), **compréhensible** (niveau sémantique) et **accessible** (niveau technique).

2. Solution pour l'interopérabilité des systèmes d'information : le Système d'Information Collaboratif (SIC)

Dans notre étude sur les solutions pour l'interopérabilité des systèmes d'information, nous avons identifié deux approches principales. La première consiste en une refonte profonde des entreprises (standardisation vis-à-vis d'un certain nombre de normes garantissant leur capacité d'intégration), mais cette vision radicale est coûteuse à cause de l'investissement à faire pour réussir la standardisation des SI. La deuxième approche consiste à conserver l'intégrité des systèmes d'information, tout en assurant leur interopérabilité. Dans nos travaux, nous avons opté pour la deuxième approche en proposant un système intermédiaire dénommé « médiateur » pour supporter les exigences de l'interopérabilité de ces systèmes. Le médiateur forme avec les parties publiques des systèmes d'information collaborant le Système d'Information Collaboratif (SIC). Notre approche permet en effet d'assimiler la problématique de l'interopérabilité directe entre SI à une problématique de médiation entre ces SI. Ce dernier fournit l'interopérabilité au moment où il y en a besoin, en traitant en particulier l'hétérogénéité des données, processus et applications des SI.

Nous avons montré dans nos travaux qu'une architecture logique de base de ce médiateur comporte des modules destinés à traiter l'hétérogénéité des systèmes d'information (gestion de processus collaboratif et gestion de l'hétérogénéité syntaxique, sémantique et technique). Enfin, nous avons proposé une adaptation de notre concept de SIC en adoptant une approche orientée services (SOA). Nous avons montré que cette architecture de systèmes d'information facilite l'intégration des systèmes d'information partenaires lorsqu'ils sont hétérogènes. Les trois apports que nous avons identifiés (réduction de la complexité, agilité et accessibilité d'un système d'information) sont de nature à réduire la difficulté de communiquer entre systèmes d'information. Le SIC, dans cette nouvelle vision, est chargé de gérer les différents échanges entre services de partenaires. Nous avons également élargi le champ d'action du SIC : il intervient dans le processus collaboratif en proposant des services à valeur ajoutée pour les partenaires,

mais il peut également proposer des services techniques afin d'améliorer la qualité de l'interopérabilité.

3. Conception de système d'information collaboratif dans une approche orientée services

L'apport principal de nos travaux réside dans la proposition d'une démarche qui s'inscrit dans l'approche de l'interopérabilité dirigée par les modèles (MDI). Notre démarche part d'une spécification du besoin de la collaboration au travers d'un modèle de processus collaboratif dans l'objectif de générer un modèle logique de système d'information collaboratif. Pour justifier notre approche, nous nous appuyons d'une part sur le rôle central qu'occupent les processus dans la caractérisation d'une modélisation métier de la collaboration et d'autre part sur la conception de système d'information. Toutefois, nous avons mentionné que les processus ne peuvent pas fournir toute la connaissance nécessaire pour la spécification d'une solution qui va supporter leur exécution. Leur définition permet finalement de guider la démarche qui vise à identifier spécifiquement le système cible. Le cadre global de notre travail est définit dans le « Y » représentatif de l'approche de conception MDA. Notre contribution dans ce cadre global (Fig VI.1), se positionne dans la branche *métier* du « Y » représentatif de l'approche de conception MDA. Il s'agit plus précisément d'une descente en abstraction depuis la couche *métier* jusqu'à la couche *logique* (passage du CIM au PIM) concrétiser par la traduction d'un modèle de processus collaboratif exprimé (BPMN) en un modèle de SIC (UML). Il est possible de schématiser les principes mis en œuvre en parlant d'un « Y secondaire » de conception logique (notion de *Logical Model*). Dans nos travaux, ce LM concerne une architecture SOA, définie selon trois vues principales : services, informations et processus.

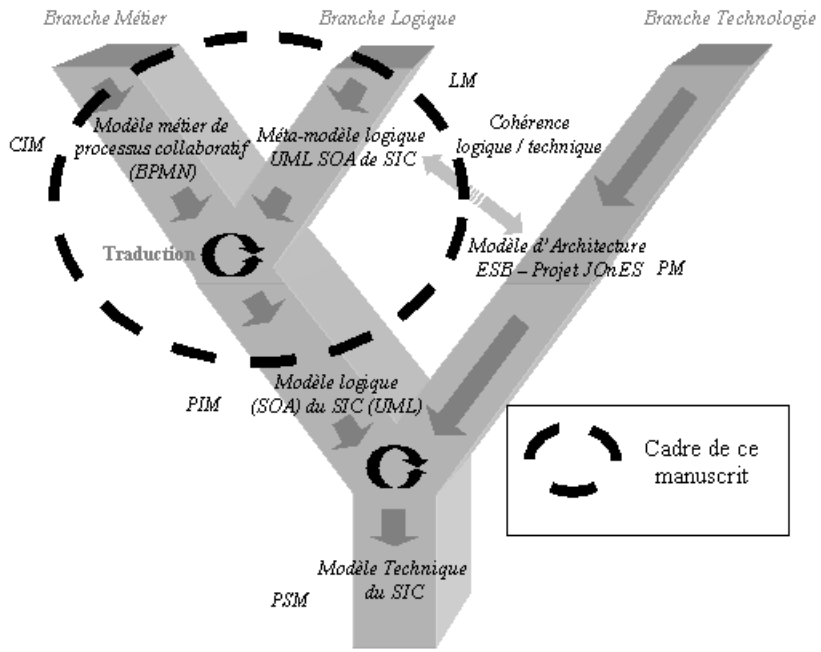


Fig. VI.1. Notre participation au sein du « double Y » de l'approche MDA

La mise en œuvre de notre proposition s'est concrétisée avec la définition en détail des méta-modèles de processus collaboratif et de SIC (selon une approche SOA).

L'identification des règles de correspondance entre les différents éléments des deux méta-modèles constitue également une étape importante. A travers les règles de correspondance, nous avons voulu raccourcir la « distance conceptuelle » qui sépare le niveau métier (CIM) du niveau logique (PIM). Cette problématique nous semble d'autant plus importante que nous pensons que la distance conceptuelle qui sépare un PIM d'un CIM est plus grande que celle qui sépare un PIM d'un PSM. Cette affirmation repose en particulier sur le nécessaire apport en « créativité » qu'impose la modélisation d'une architecture logique à partir des seuls besoins métier, à contrario d'une projection logique / technique qui bien que loin d'être triviale, semble tout de même moins problématique (car moins disparate en ses deux extrémités).

4. Implémentation d'un atelier de transformation logiciel « Traducteur V1.0 »

Notre travail s'est achevé sur l'implémentation d'une maquette d'atelier logiciel permettant, à partir d'une modélisation de processus collaboratif (BPMN), de générer un modèle de SIC qui respecte l'architecture logique spécifique qui a été choisie pour notre solution d'interopérabilité. Ce modèle peut être enrichi par la suite à l'aide d'une connaissance technique sur les services et données des partenaires. S'il y a eu un effort conséquent de choisir et d'assembler les différents outils complémentaires et compatibles pour former notre atelier (Intalio©, ATL© et TOPCASED©), notre principale effort a porté sur la maîtrise de l'opérationnalité des transformations avec ATL©. Nous avons réussi à manipuler correctement les différents types de règles offerts par ce langage (règle appelée, règle déclarative, etc.) et nous sommes également parvenus à intégrer les profils UML (sous la forme du profil de notre architecture spécifique) dans les fichiers de transformation afin de générer des modèles UML pertinents, légitimes et parlant.

nos travaux ne prétendent pas présenter une solution parfaite et indiscutable à une problématique complexe qui est celle de l'interopérabilité des systèmes d'information. Nous avons essentiellement essayé, dans ces travaux, de mettre en valeur l'utilisation combinée de SOA et de MDA, afin de proposer une approche bien définie, basée sur la médiation de systèmes d'information, dans l'objectif de faciliter leur collaboration.

3. Limites de ces travaux de thèse

1. Dépendance à SOA

Notre approche repose sur une vision orientée services et il est à ce titre indispensable que les partenaires de la collaboration soient en mesure de présenter leur système d'information selon cette même vision vis-à-vis de l'extérieur. Cette exigence préalable d'une mise en conformité avec les principes SOA, constitue évidemment une limite tangible à nos propositions (et ce malgré le développement incontestable de ce type d'approche à l'heure actuelle). Il est clairement impensable d'exiger des partenaires, sous prétexte de leur volonté affichée de s'inscrire dans une démarche de collaboration, une remise en question profonde de leur système d'information (dans le cas où leur SI ne correspondrait pas à ces attentes). Néanmoins, dans le cadre des niveaux de maturité collaborative qui ont été évoqués dans ce document, nous souhaitons rappeler le fait que les stades d'*entreprise ouverte* et d'*entreprise fédérée* (i.e. les niveaux inférieurs à celui d'*entreprise interopérable*) exigent déjà du SI des caractéristiques technologiques avancées. On peut alors se demander si ce type de technologies (et SOA en particulier) loin d'être, comme nous l'avons répété, le vecteur unique de l'interopérabilité, ne constituent pas pour autant un marqueur significatif du niveau de maturité collaborative des entreprises.

2. Gestion de l'enrichissement des modèles

Nous avons clairement identifié le besoin de disposer d'un enrichissement de la connaissance fournie par le modèle BPMN par une connaissance additionnelle. Les processus ne peuvent pas fournir à eux seuls des modèles complets et exploitables pour la description de SIC. Nous supposons donc que les modèles générés doivent être complétés par des connaissances techniques additionnelles après la génération du modèle PIM. Cependant, il existe une autre approche, qui aurait pu être mieux étudiée. Une connaissance métier additionnelle pourrait être introduite, en même temps que le processus collaboratif, dans la démarche de transformation. Par connaissance métier, nous entendons principalement :

- les informations de toute sorte que l'entreprise utilise (produits, procédés, ressources…),
- description des services ainsi que les règles qui décrivent leur fonctionnement,
- les acteurs et les rôles qui peuplent l'organisation,
- les relations entre ces éléments.

Que la connaissance soit au cœur de la problématique d'interopérabilité n'est pas un résultat surprenant, quand on considère son importance croissante dans le métier d'architecte de système d'information dans une entreprise.

4. Perspectives

La question de l'adaptation des entreprises à la nécessaire mise en collaboration avec d'autres partenaires constitue un enjeu majeur de la survie des acteurs industriels. Et plus particulièrement, de la **conception d'un SI Collaboratif**, capable de supporter la collaboration entre SI des partenaires (pourvu que ceux-ci soient orientés services). Comme nous l'avons vu, ces travaux se positionnent dans la branche *métier* du « Y », représentatif de l'approche de conception MDA. Nous souhaitons revenir sur ce positionnement en discutant en particulier de travaux connexes menés actuellement au sein de notre laboratoire et qui constituent un certain nombre de perspectives à ce manuscrit. Nous proposons également de les positionner sur le « Y » de l'approche MDA.

Nous allons voir que nous pouvons identifier trois sujets de recherche qui permettent de compléter le schéma de la figure VI.1. Ces travaux ambitionnent de proposer une couverture additionnelle aux « zones du Y MDA », qui n'étaient pas (ou partiellement) couvertes par les travaux exposés dans ce manuscrit. A ces trois sujets, nous pouvons associer un quatrième qui n'est pas encore d'actualité (car non initié).

La première de ces perspectives concerne la génération du modèle BPMN de processus collaboratif (le modèle métier – CIM) servant de base à la définition du modèle UML du SIC (le modèle logique – PIM). Ce sujet s'avère incontournable pour deux raisons : (i) il est finalement peu courant de rencontrer un groupement d'entreprises souhaitant collaborer qui soient capables de formaliser, en amont, les termes de leur collaboration jusqu'à être en mesure de fournir un modèle du processus collaboratif sur lequel ils souhaitent baser leur réseau, (ii) la démarche de conception de SIC proposée dans ce manuscrit nécessite que le modèle de processus collaboratif servant de base à la traduction soit formalisé en BPMN et respecte le méta-modèle présenté au chapitre 4 (Fig. IV.6) : il est délicat d'attendre qu'un modèle de processus collaboratif établi par les partenaires (s'il existe) soit conforme à ces contraintes. Dans [**Rajsiri** *et al.* **07**] on présente une démarche ontologique de construction de ce type de processus collaboratif (en BPMN et qui respectent le méta-modèle présenté dans le chapitre IV) à partir de la connaissance dont le groupe de partenaires dispose et sur la collaboration à venir. Pour ce faire, deux ontologies ont été définies : la première pour les collaborations (définissant et positionnant les notions de *partenaires*, *rôles*, *objectifs*, *domaine métier*, *tâches*, etc.) et la seconde pour les processus collaboratifs (sur la base de classifications de processus telles que le *Process Handbook* [**Malone** *et al.* **99**] et en conformité avec le méta-modèle du chapitre IV). Ces deux ontologies ont ensuite été connectées par des liens sémantiques et structurels permettant de conduire une démarche d'inférence. Ces déductions offrent à la fois la possibilité de vérifier la cohérence du modèle de la collaboration et de le compléter, mais surtout de proposer un modèle de processus adapté à la collaboration modélisée (en BPMN et conforme au méta-modèle de processus collaboratif présenté au chapitre IV). Les travaux de thèse de V. Rajsiri se positionnent donc en amont des travaux présentés dans ce manuscrit et viennent compléter la partie haute de la branche métier du « Y » MDA (Fig. VI.2).

La deuxième perspective sur laquelle nous proposons de nous attarder concerne la réalisation d'un (ou plusieurs) modèle de plate-forme technique (PM) susceptible de s'interfacer efficacement (lors d'une phase d'urbanisation) avec le PIM proposé par une traduction, telle que celle que nous présentons dans nos travaux. Il est nécessaire que ce modèle de plate-forme technologique soit cohérent avec le modèle logique choisi. Le

projet **JOnES** (ANR/RNTL2005, étalé sur une période allant de janvier 2006 à janvier 2008) auquel nous participons, s'intéresse à la définition d'une architecture technique d'Enterprise Service Bus (*ESB*) respectant une structure SOA. La technologie ESB, relativement jeune, est actuellement une piste prioritaire de l'outillage de la philosophie SOA. Si certains courants proposent d'utiliser certaines approches plus anciennes (telles que l'EAI, voire l'adaptation spécifique de progiciels de gestion intégrée[1]) pour assurer la couverture technologique des principes SOA, le projet **JOnES** préfère étudier la définition d'un ESB. Ce projet est particulièrement adapté à une convergence vers le type de PIM que notre approche de traduction peut générer. Les connexions du projet JonES, avec les travaux présentés dans ce manuscrit et les perspectives qu'elles représentent, se positionnent sur la branche technique du « Y » MDA de la figure VI.2 (en tant qu'architecture technique compatible candidate).

Le troisième volet de ces perspectives concerne un autre projet en cours. Le projet **ISyCri** (ANR/CSOSG2006, étalé sur une période allant de mai 2007 à mai 2009) [**Bénaben** *et al.* **07**] s'intéresse à la conception de système d'information collaboratif dans un contexte de crise. Les points particuliers de ce projet, qui constituent des perspectives tangibles à ce manuscrit, sont les suivants : (i) la démarche d'urbanisation de système d'information (permettant en particulier la mise en correspondance des activités à effectuer avec l'ensemble des services disponibles) doit être concrètement mise en œuvre dans le cadre de différents cas d'usage (tels que le tremblement de terre de Yogyakarta en Inde en mai 2006 ou une fuite chimique virtuelle dans un exercice de la préfecture du Tarn en 2004), (ii) en outre, ce projet ambitionne de s'intéresser à la faculté du PSM généré de permettre la conception d'un SIC **agile**. Ce SIC devra alors être à la fois **réactif** (la définition même de ce SIC doit être extrêmement rapide compte-tenu de la nature dramatique situation) et **flexible** (la situation de crise est par définition instable et il est crucial que le SIC reste pertinent et s'adapte à l'évolution de l'environnement). Ces travaux sont en particulier supportés par la thèse de S. Truptil sur la définition de modèles de SIC agile en contexte de crise. Ce troisième niveau de perspective se positionne dans la partie basse du « Y » MDA (Fig. VI.2).

L'ensemble de ces travaux doit, à terme, constituer une concrétisation de l'approche MDA dans un contexte de support de l'intégration de partenaires. Cette ambition se base sur notre conviction que l'interopérabilité des SI constitue une clef essentielle à ce verrou de l'intégration, mais également sur la « contrainte minimale » que nous avons identifiée : la nature SOA des SI des partenaires. Conscient de cette contrainte, nous proposons une quatrième piste de perspectives allant dans cette direction. Concrètement, l'objectif de ces travaux complémentaires serait de nous intéresser à la façon de permettre aux acteurs de la collaboration de faire apparaître leur SI comme « orientés services » (en les modifiant selon une démarche à définir ou en les équipant de connecteurs jouant le rôle d'interface SOA). Ce quatrième point des perspectives ne se positionne pas concrètement sur la figure VI.2. En revanche, il constitue un moyen de permettre aux partenaires potentiels d'entrer dans le processus d'intégration que nous proposons.

[1] Progiciel de Gestion Intégrée (PGI) : système informatique assurant la couverture des activités de l'entreprise en intégrant ses données (au moins), ses applications (souvent) et certains de ces processus (parfois). On les désigne également sous le terme ERP (Enterprise Resource Planning).

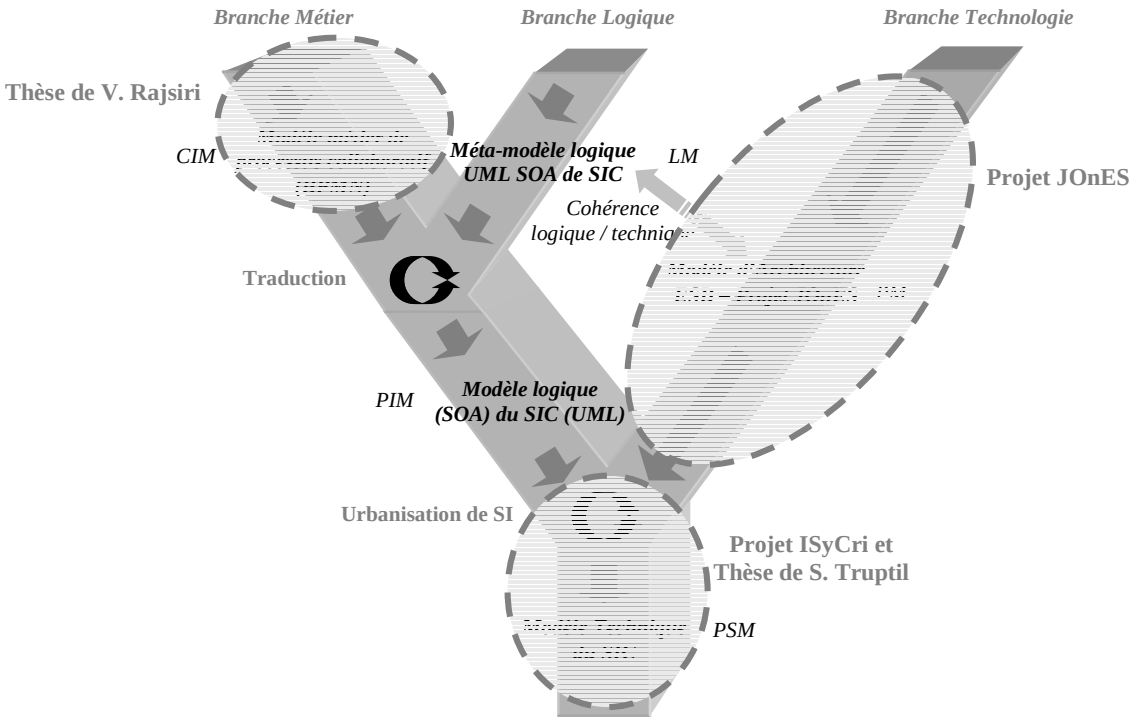


Fig. VI.2. *Positionnement de nos perspectives de travaux au sein du « double Y » de l'approche MDA*

Glossaire

Glossaire

AIF	*Athena Interoperability Framework*
API	*Application Programming Interface*
B2B	*Business to Business*
BPEL	*Business Process Execution Language*
BPM	*Business Process Management*
BPMI	*Business Process Management Initiative*
CIM	*Computation Independent Model*
EAI	*Enterprise Application Integration*
EDI	*Electronic Data Interchange*
EIF	*European Interoperability Framework*
ESB	*Enterprise Service Bus*
HTTP	*Hyper Text Transfer Protocol*
IDEAS	*Interoperability Development for Enterprise Applications and Software*
J2CA	*J2EE Connector Architecture*
JBI	*Java Business Integration*
JSR	*Java Specification Request*
MDA	*Model Driven Architecture*
MDI	*Model Driven Interoperability*
MOM	*Message Oriented Middleware*
OMG	*Object Management Group*
OMT	*Object Modelling Technique*
OOD	*Object Oriented Design*
OOSE	*Object Oriented Software Engineering*
PIM	*Platform Independent Model*

PIM4SOA	*Platform Independent Model For Service-Oriented Architecture*
PSM	*Platform Specific Model*
RMI	*Remote Method Invocation*
RPC	*Remote Procedure Call*
SGWF	*Système de Gestion de WorkFlow*
SIC	*Systéme d'Information Collaboratif*
SOA	*Service Oriented Architecture*
SOAP	*Simple Object Access Protocol*
SPOF	*Single Point Of Failure*
TCAO	*Travail Coopératif Assisté par Ordinateur*
UDDI	*Universal Description, Discovery and Integration*
UEML	*Unified Enterprise Modeling Language*
UML	*Unified Modeling Language*
W3C	*World Wide Web Consortium*
WfMC	*Workflow Management Coalition*
WSDL	*Web Service Description Language*
XML	*eXtended Markup Language*
XSD	*XML Schema Definition*

Références bibliographiques

Références bibliographiques

[Andro 07] AndroMDA. Disponible sur *htpp://www.andromda.org/*

[Archimate 07] Archimate, *Deliverables of Archimate Project*, https://doc.telin.nl/dscgi/ds.py/Index/Collection-4483.

[Athena 04] Athena Consortium. *ATHENA General description v10*, public document: http://www.athena-ip.org/, 2004.

[Athena 05] ATHENA, *D.A1.3.1: Report on methodology description and guidelines definition Version 1.0*, ATHENA Integrated Project, deliverable D.A1.3.1, March 2005.

[Aubert *et al.* 02] B. Aubert, A. Dussart, *Systèmes d'Information Inter-Organisationnels*, Rapport Bourgogne, CIRANO, mars 2002.

[Baina *et al*. 06] S. Baina, H. Panetto, K. Benali, *Apport de l'approche MDA pour une interopérabilité sémantique*, revue Ingénierie des Systèmes d'Information (ISI), Numéro spécial Ingénierie des processus d'entreprise et des SI, Vol. 11, Num. 3, pp 11-29, Hermès Science Publications, Lavoisier, Paris, 2006.

[Bauer *et al*. 05] B. Bauer, J.P. Müller, S. Roser, *Adaptive design of cross organizational business processes using a model-driven architecture*, International Conference on Business Information Systems (Wirtschaftsinformatik 2005), Physica-Verlag, pp. 103-121, 2005.

[Bauer *et al*. 06] B. Bauer, J.P. Müller, S. Roser, *A Decentralized Broker Architecture for Collaborative Business Process Modelling and Enactement*, Enterprise Interoperability: New Challenges and Approaches- Springer Verlag - ISBN-10: 1846287138, 2006.

[Bénaben *et al.* 06] F. Bénaben, J. Touzi, N. Rajsiri, H. Pingaud, *Collaborative Information System Design* , Série GI LNI, Lecture Notes in Informatics, AIM'06, Luxembourg, Grand-duché de Luxembourg, 7-9 juin 2006.

[Bénaben *et al.* 07] F. Bénaben, , J.P. Pignon, C. Hanachi, J.P. Lorre, et V. Chapurlat. *Interopérabilité des systèmes en situation de crise*, Workshop Interdisciplinaire sur la Sécurité Globale, Troyes, 2007.

[Benali 04] K. Benali, *Quelques briques et concepts de base pour la coopération et les systèmes coopératifs*. Habilitation à Diriger des Recherches en informatique de l'Université Nancy 2, 1er décembre 2004.

[Benguria *et al*. 06] G. Benguria, X. Larrucea, B. Elveseater, T. Neple, A. Beardsmore, M. Friess. *A Platform Independent Model for Service Oriented Architectures*. 2éme conférence sur l'interopérabilité, IESA'06-Bordeaux, Mars 2006.

[Bernus *et al*. 96] P. Bernus, G. Schmidt, *Architectures for enterprise integration*, Chapman and Hall, London, 1996.

[Bernus *et al*. 98] P. Bernus, G. Schmidt, *Architectures of Information Systems, handbook on architectures of information systems*, Springer Verlag, ISBN 3-540-64453-9, 1998

[Berre *et al.* 07] A-J. Berre, B. Elveaster, N. Figay, C. Guglielmina, S. G. Johnsen, D. Karlsen, T. Knothe and S. Lippe, *The ATHENA Interoperability Framework*, Enterprise interoperability: New challenges and approaches II, Springer edition. ISBN:978-1-84628-857-9, 2007.

[Bézivin *et al.* 02] J. Bézivin et X. Blanc, *MDA : standards et travaux*, développeur Référence, 2 octobre 2002.

[Bézivin *et al*. 03] J. Bézivin, G. Dupé, F. Jouault, G.Pitette,J. E. Rougui, *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, October 2003.

[Bézivin *et al.* 03a] J. Bézivin et S. Gérard, *A Preliminary Identification of MDA Components*, 2003.

[Blanc 04] L. Blanc dit Joli Cœur, *Modélisation des processus métier dans une approche EAI*, thèse de doctorat, Université de Savoie, 2004

[Booch *et al.* 00] G.Booch, I. Jackobson, le guide de l'utilisateur UML, Eyrolles, 2000

[Booch *et al.* 04] G. Booch, I. Jacobson, J. Rumbaugh, UML 2.0 *Guide de référence*, Paris, CampusPress, 2004.

[Bourey *et al.* 05] J.-P. Bourey, R. Grangel, A.Berre, G. Doumeingts, K. Kalampoukas, M. Bertoni, L. Pondrelli, and N. Daclin, *DTG2.1: Report on model establishment*, Interoperability Research for Networked Enterprises Applications and Software Network of Excellence, n° IST 508-011, 2005

[Bouzguenda 06] L. Bouzguenda, *Coordination multi-agents pour le Workflow inter-organisationnel lâche*, thèse de doctorat, IRIT, 2006.

[BPMI 04] BPMI, *Business Process Modeling Notation (BPMN)*,Version 1.0 - May 3, 2004

[BPML 07] BPMI, *BPML - Business Process Modeling Language*, www.bpmi.org.

[Budinski *et al.* 03] F. Budinsky, D. Steinberg, E. Merks, R. Ellersick et T.J. Grose, *Eclipse Modeling Framework: A Developer's Guide*. Addison-Wesley Pub Co, 1st edition, August 2003.

[Burlat *et al*. 01] P. Burlat, D. Vila, B. Besombes, V. Deslandres, *Un cadre de modélisation des*

trajectoires d'évolution des groupements d'entreprises, 4éme congrès international de Génie Industriel 12-15 juin 2001, Aix-Marseille, France.

[Chelli 03] H. Chelli, *Urbaniser l'entreprise et son système d'information*, Vuibert, ISBN : 978-2-7117-4817-4, 201 p, 2003

[Chen *et al.* 03] D. Chen, G. Doumeingts, *European initiatives to develop interoperability of enterprise applications basic concepts, framework and roadmap*, Annual Reviews in Control, vol. 27, p. 153-162, 2003.

[Chituc *et al.* 05] C.M. Chituc and A. Azvedo, *Towards a Self-Forming Business Networking Environment*, Proceedings of the Seventh International Conference on Enterprise Information Systems, Miami, USA, May 25-28, 2005

[Crusson 03] T.Crusson, *BPM, De la modélisation à l'exécution : Positionnement par rapport aux Architectures Orientées Services*, livre blanc- Intalio, 2003.

[D'antonio 05] D'Antonio, *TG MoMo Roadmap*, Delivrable Interop, MoMo.2, 2005

[Darras 04] F. Darras., *Proposition d'un cadre de référence pour la conception et l'exploitation d'un progiciel de gestion intégré*, thèse de doctorat, Ecole des mines d'Albi-Carmaux, 2004.

[Davis 74] D.-B. Davis, *MIS: conceptual, foundation, structure and development*, Mc Graw Hill, 1974.

[Debauche *et al*. 04] B. Debauche, P. Megard, *BPM - Business Process Management : pilotage métier de l'entreprise* - Paris : HERMES, - 212 p. ISBN 2-7462-0852-0, 2004.

[Dolidon *et al.* 06] F. Dolidon, H. Pingaud, *Modèles et méta-modèles pour la conception des ERP*, Communication à la journée "Pratiques et processus en ingénierie des ERP", conjointe au GT ECI et ERP du CNRS GdR MACS, Toulouse 6 décembre, 2006.

[Dumas *et al.* 90] P. Dumas, G. Charbonnel, *La méthode OSSAD pour maîtriser les technologies de l'information*, Volume 1 ,Principes, éditions d'Organisation, 1990

[EBM 05] EBM Websourcing, *Nouvelles technologies pour l'intégration : les ESB*, White Paper, Mai 2005

[Eclipse 07] Eclipse Foundation, *Eclipse - an open development platform*, www.eclipse.org.

[EDI 07] EDI x12 standards, http://www.x12.org/

[EIF 04] European Interoperability Framework, White Paper, Brussels, http://www.comptia.org, Feb 2004.

[Fowler *et al*. 04] M. Fowler, *UML 2.0 : initiation aux aspects essentiels de la notation*, CampusPress, 2004.

[Garcia 04] A. Garcia, *L'approche guidée par les modèles de l'OMG*, Tectosages, rapport interne, 2005.

[Geram 97] GERAM, *Generalised Enterprise Reference Architecture and Methodology*, Task Force on Enterprise Integration, 1997.

[Grangel *et al.* 06] R. Grangel Seguer, J.-P. Bourey, R. Chalmeta, M. Bigand, *UML for Enterprise Modelling: a basis for a Model-Driven Approach*. Enterprise Interoperability: New Challenges and Approaches- Springer Verlag edition, ISBN-10 : 1846287138, 2006.

[Grangel *et al.* 07] R. Grangel Seguer, R. Ben Salem, J.-P. Bourey, N. Daclin, Y. Ducq, *Transforming GRAI Extended Actigrams into UML Activity Diagrams: a First Step to Model Driven Interoperability*, Enterprise interoperability: New challenges and approaches II, Springer Verlag edition, ISBN : 978-1-84628-857-9, 2007.

[Grüninger *et al.* 95] M. Grüninger and M.S Fox, *Methodology for the design and evaluation of ontologies*. Workshop on Basic Ontological Issues in Knowledge Sharing, Montreal, Canada, 1995.

[Guarino 99] N. Guarino, *Formal Ontology and Information Systems*, Formal Ontology in Information Systems. IOS Press, 1999.

[Heimbigner *et al.* 85] D. Heimbigner, D. McLeod, *A federated architecture for information management*. ACM Transactions on Office Information Systems (TOIS), Vol. 3, No. 3, p. 253-278, juillet 1985.

[Hostachy 00] E. Hostachy, *Le système d'information doit être centré sur l'EAI*, Informatiques Magazine, pages 40-44, mai 2000.

[IDEAS 03] IDEAS, *A gap Analysis –Required activities in Research, Technology and standardisation to close the RTS Gap- Roadmaps and Recommendations on RTS activites*, IDEAS, Deliverables, 2003.

[IEC 05] IEC 62390, Common automation device. *Profile guideline*, IEC TC 65/290/DC Device Profile Guideline (2002), TC65: Industrial Process Measurement and Control. IEC TR 62390, IEC, Geneva, Switzerland, 2005.

[IEEE 00] IEEE, *IEEE Recommended Practice for Architecture Description of Software-Intensive Systems*, IEEE Recommended practice for architectural description of software-intensive systems, 2000.

[IEEE 90] IEEE, *IEEE: Standard Computer Dictionary - A Compilation of IEEE Standard Computer Glossaries*, 1990.

[Iso 00] Norme européenne NF EN ISO 9001 version 2000, *Systèmes de management de la qualité – Exigences*, AFNOR, 2000.

[Izza 06] S. Izza, *Intégration des systèmes d'information industriels, une approche flexible basée sur les services sémantiques*, thèse de doctorat, Ecole des Mines de Saint-Étienne, 2006.

[Jouanot 00] F. Jouanot, *Un modèle sémantique pour l'interopérabilité de systèmes d'information*, actes du 13e Congrès Inforsid, p. 16-19, Lyon, France, mai 2000.

[Jouault 06] F. Jouault, Contribution à l'étude des langages de transformation de modèles, thèse de doctorat, Université de Nantes, 2006.

[Kalnins *et al.* 04] A. Kalnins, J. Barzdins et E. Celms, *Basics of Model Transformation Language MOLA*. Workshop on Model Driven Development (WMDD 2004) at ECOOP 2004, June 2004.

[Karsai *et al.* 03] G. Karsai, A. Agrawal, F. Shi et J. Sprinkle, *On the Use of Graph Transformation in the Formal Specification of Model Interpreters*. J. UCS, 9(11) : 1296.1321, 2003.

[KnoInt 07] Knowledge map of research in interoperability, INTEROP NoE http://www.interop-noe.org/.

[Konstantas *et al.* 05] D. Konstantas, J.-P. Bourrières, M. Léonard, N. Boudjlida, 2005. Preface de : *Interoperability of Enterprise Software and Applications*, Actes de INTEROP-ESA'05, Genève, Suisse, , Springer-Verlag, p. v-vi, 21-25 février 2005.

[Kosanke 05] K. Kosanke, *ISO Standards for Interoperability: a comparison*, INTEROP-ESA'05, Proceedings of the First International Conference on Interoperability of Enterprise Software and Applications (IFIP/ACM SIGAPP INTEROP-ESA'2005).

[Lascoux 07] J.-L. Lascoux, *Pratique de la médiation*, Editeur ESF, ISBN-10: 2710118599, 2007.

[Le Moigne 77] J.-L. Le Moigne, *La théorie du système général, théorie de la modélisation*, Presses Universitaires de France, (3ème édition, 1990), 1977

[Le Moigne 90] J.-L. Le Moigne, *La modélisation des systèmes complexes*, Afcet-systèmes, Dunod, 1990.

[Lonpégé 02] C. Lonpégé, *Le projet d'urbanisation du système d'information*, Dunod, 2002. AM, Agile Modeling, www.agilemodeling.com.

[Lopes 05] D. Lopes, *Étude et applications de l'approche MDA pour des plate-formes de Service Web*, thèse de doctorat, Université de Nantes, 2005

[Malone *et al.*99] T.W. Malone, K. Crowston, J. Lee, B. Pentland, C. Dellarocas, G. Wyner, J. Quimby, C. Osborn, A. Bernstein, G. Herman, M. Klein, and E. O'Donnell. *Tools for inventing organizations: Toward a handbook of organizational processes*. Management Science 45 (3):425-443, 1999.

[Mega 07] Mega, *Mega Suite*, www.mega.fr.

[Miller *et al.* 03] J. Miller et J. Mukerji, *MDA Guide Version 1.1*, , Document number omg/,12 Juin 2003.

[Monateri *et al.* 99] J.C. Monateri, M. Sapina, *Dynamique des relations entre entreprises, stratégies manufacturières et arrangements contractuels durables* , 3° Congrès international de génie industriel, L'intégration des ressources humaines et des technologies : le défi, , mai 1999.

[Monfort *et al.* 04] V. Monfort, S. Goudeau, *Web services et interopérabilité des SI*, Dunod/01 Informatique, Collection InfoPro - ISBN:210008240X, 2004.

[Morley 00] C. Morley, *Changement organisationnel et Modélisation des processus*, 5éme congrés de l'Association Information et Management(AIM), Montpellier 2000.

[Morley 02] C. Morley, *La modélisation des processus : typologie et proposition utilisant UML*. Processus et Systèmes d'information – Journées ADELI, Paris, France, 2002.

[Morley *et al*. 02] C. Morley, J. Hugues et B. Leblanc, *UML pour l'analyse d'un système d'informations*, 2éme édition, Dunod, 2002.

[Morley *et al*. 05] C. Morley, J. Hugues, B. Leblanc, O. Hugues, *Processus métiers et S.I : évaluation, modélisation, mise en œuvre*, éditions DUNOD, ISBN 2100070991, mars 2005.

[Mougin 02] Y. Mougin, *La cartographie des processus*, économica, p.1-8., 2002.

[Nurcan 07] S. Nurcan, *Business process modeling and flexibility*, Enterprise interoperability: New challenges and approaches II, Springer edition. ISBN:978-1-84628-857-9, 2007.

[OMG 02] OMG. Request for Proposal: MOF 2.0 Query/Views/Transformations RFP, October 2002. Disponible sur http://www.omg.org/docs/ad/02-04-10.pdf.

[OMG 02a] OMG, *UML Profile for Enterprise Distributed Object Computing Specification*, février 2002.

[OMG 03] OMG, *MDA guide version 1.0.1*, document number : omg/2003-06-01 edition. Object Management Group, 2003.

[Opdahl *et al.* 06] A. Opdahl, G. Berio, *A Roadmap for UEML, Enterprise interoperability:New challenges and approaches*, Springer edition. ISBN: 978-1-84628-713-8, p. 189-198, 2006.

[OpenGroup 05] The Open Group, TOGAF - *The Open Group Architecture Framework*, Version 8, The Open Group, 2005.

[Ouyang *et al.* 06] C. Ouyang, W. Van Der Aalst, M. Dumas, A. Hofstede, *Translating BPMN to BPEL*, Technical report - BPM group of Queensland University of Technology Brisbane (QUTB), 2006.

[OWL 04] OWL, *OWL: Web Ontology Language*, http://www.w3.org/TR/owl-features/. 2004.

[Panetto 06] H. Panetto, *Meta-Modèles et Modèles pour l'Intégration et l'Interopérabilité des Applications d'Entreprises de Production*, HDR, Université Nancy 1, 2006.

[Pera 07] PERA, *The Purdue Enterprise Reference Architecture*, http://www.pera.net.

[Pingaud 03] H. Pingaud, *Logistiques et technologies de l'information et de la communication : les guides experts* /. WEKA. 100 p., ISBN 2-7337-0232-7, 2003.

[Pokarev *et al.* 06] S. Pokraev, D. Quartel, M. W.A. Steel, M. Reichert, *Semantic Service Modeling: Enabling System Interoperability*, Enterprise Interoperability : New Challenges and Approaches- Springer Verlag -ISBN-10 : 1846287138, 2006

[Putman 01] J. R., Putman, *Architecting with RM-ODP*, Prentice Hall, 2001.

[Rajsiri *et al.* 07] V.Rajsiri, J.-P. Lorré, F. Bénaben and H. Pingaud, *Cartography for Designing Collaborative Process*, Enterprise interoperability : New challenges and approaches II, Springer edition. ISBN: 978-1-84628-857-9, 2007

[Raymond 07] G. Raymond, *SOA : architecture logique, principes, structures et bonnes pratiques*, www.softeam.fr, 2007.

[Recker *et al.* 07] J. Recker, J. Mendling, *On the Translation between BPMN and BPEL: Conceptual Mismatch between Process Modeling Languages*. In: T. Latour and M. Petit, the 18th International Conference on Advanced Information Systems Engineering. Proceedings of Workshops and Doctoral Consortium. Namur University Press, Luxembourg, Grand-Duchy of Luxembourg, pp. 521-532, 2007.

[Reix 02] R. Reix, *Système d'information et management des organisations*, Vuibert, 4e édition, 2002.

[Roques 04] P. Roques, *UML 2 par la pratique*, Paris, Eyrolles, 2004.

[Roser *et al.*07] S. Roser and B. Bauer, *Improving Interoperability in collaborative Modelling*, Enterprise interoperability: New challenges and approaches II, Springer edition. ISBN : 978-1-84628-857-9, 2007.

[Saadoun 00] M. Saadoun, *Technologies de l'information et management*, Hermès, 2000.

[Sakli 00] K. Sakli, *Flexibilité des Workflows par l'approche Objet : 2flow, un framework pour workflows flexibles*, thèse de doctorat, Ecole centrale de lyon, 2000.

[Sassoon 98] J. Sassoon, *Urbanisation des systèmes d'information*, Hermès, 1998.

[Saven 04] R. Saven, *Business process modelling : review and framework*, International Journal of Production Economics, vol. 90, issue 2, , p. 129-149, 28 July 2004.

[Schlenoff *et al.* 99] C. Schlenoff, M. Gruninger, F. Tissot, L.Valois, , and J. Lubell, *The Process Specification Language (PSL): Overview and Version 1.0 Specification*, NIST Internal Report (NISTIR) 6459, National Institute of Standards and Technology, Gaithersburg, MD, USA, 1999.

[Sheer 07] IDS Scheer, *ARIS Tool Set*, http://www.ids-scheer.fr.

[Sheer 94] A.W. Sheer, *ARIS Toolset: A software product is born*, Information Systems Journal, Volume 19, Issue 8, December 1994, Pages 607-624.

[Sheth *et al.* 90] A.P. Sheth, J. Larson, *Federated database systems for managing heterogeneous, distributed and autonomous Databases*. ACM Computing Surveys, Vol. 22, No. 3, 1990.

[Sneed 06] H. M. Sneed, *Integrating legacy Software into a Service oriented Architecture*, Discussion Paper. In: F. Lehner, H. Nösekabel, P. Kleinschmidt, eds.: Multikonferenz Wirtschaftsinformatik 2006, Band 2, XML4BPM Track, GITO-Verlag Berlin, ISBN 3-936771-62-6, pages 345-360, 2006.

[Soulier *et al.* 05] E. Soulier, M. Lewkowicz, *Simulation des pratiques collaboratives pour la concpetion des SI basés sur les processus métiers*, Revue Ingénierie des Systèmes d'Information (ISI), Volume 11- n°3/2006, Lavoisier, 2006.

[Terasse *et al.* 03] M.N. Terrasse, M. Savonnet, G. Becker, and E. Leclercq. *UML-Based Metamodeling for Information System Engineering and Evolution*. In Proc. of the 9th International Conference on Object-Oriented Information Systems, OOIS'03, LNCS 2817, pp. 83–94, 2003.

[Théroude 02] F. Théroude, *Formalisme et système pour la représentation et la mise en œuvre des processus de pilotage des relations entre donneurs d'ordre et fournisseurs*, thèse de Doctorat de l'Institut National Polytechnique de Grenoble, 2002.

[Touzi *et al.* 06] J. Touzi, F. Bénaben, H. Pingaud, *Interoperability through model based generation: the case of the collaborative IS*, Enterprise Interoperability: New Challenges and Approaches- Springer verlag -ISBN-10: 1846287138, 2006.

[Touzi *et al.* 06a] J. Touzi, F. Bénaben, H. Pingaud, *Collaborative Information System Design: From Process Model to Information System Model*. Proceedings of the 12th IFAC Symposium on Information Control Problems in Manufacturing (INCOM'2006), May 17-19, 2006, Saint Etienne, France, Elsevier, 2006.

[Touzi *et al.* 07] J. Touzi, F. Bénaben, J.-P. Lorré, H. Pingaud, *A Service Oriented Architecture approach for Collaborative Information System design*, IESM'07, Pekin, Chine, 30 mai-2 juin 2007.

[Vanderhaeghen *et al.* 04] D. Vanderhaeghen, S. Zang, A. Hofer, O. Adam, *XML-based Transformation of Business Process Models – Enabler for Collaborative Business Process Management* -, Proc. of the 2nd GI Workshop XML4BPM , XML Interchange Formats for Business Process Management, 2004.

[Vanderhaeghen *et al.* 06] D. Vanderhaeghen, D. Werth, T. Kahl, P. Loos, *Service-and Process-Matching- An Approach towards Interoperability Design and Implementation of Business Networks*, Enterprise interoperability : new challenges and approaches, Springer edition. ISBN : 978-1-84628-713-8, p. 189-198, 2006.

[Vernadat 96] F. Vernadat, *Enterprise modelling and integration, Principles and applications*, Chapman & Hall, 1996.

[Vernadat 97] F. Vernadat, *Enterprise Modelling Languages*, ICEIMT'97 Entreprise Integration - International Conscensus, EI-IC ESPRIT Project 21.859.

[Vernadat 99] F. Vernadat, *techniques de modélisation en entreprise : Applications aux processus Opérationnels*, économica 1999.

[Vernadat 06] F. Vernadat, *Interoperable enterprise systems : architecture and methods*, plenary lecture, IFAC/INCOM conference, Saint-Etienne, May 2006.

[Villareal 05] C. L. Villareal, *Contribution au pilotage des projets partagés par des PME en groupement basée sur la gestion des risques*, thèse de doctorat, Ecole des mines d'Albi-Carmaux, 2005.

[Wache 01] H. Wache, *Ontology-Based integration of information, a survey of existing approaches,*workshop on ontologies and information sharing, IJCAI 2001

[WfMC 99] The Workflow Mangement Coalition, *Workflow Management Coalition Terminology and Glossary*, technical report WfMC-TC-1011, Febrary 1999

[Wiederhold 92] G. Wiederhold, *Mediators in the architecture of future information systems*. IEEE Computer Magazine, Vol. 25, No. 3, 3849, mars 1992

[Zachman 87] J. Zachman, *A Framework for Information Systems Architecture*, IBM Systems Journal, Vol. 26, No 3, 1987.

[Zaidat 05] A. Zaidat, *Spécification d'un cadre d'ingénierie pour les réseaux d'organisations*. Thèse de doctorat en Génie Industriel de l'Ecole des Mines de Saint-Etienne et l'Université Jean Monnet. Septembre 2005.

[Zhao 05] J. L Zhao, H. K. Cheng, *Web services and process management : a union of convenience or a new area of research ?* Decision Support Systems, Volume 40, Issue 1, July 2005.

Annexe A

Règles de transformation BPMN→UML

Génération de diagrammes UML à partir d'un modèle BPMN

Nous présentons dans cet annexe nos essais de traduction d'un modèle BPMN vers un modèle UML (Fig A.1). Tout d'abord nous souhaitons présenter une définition formalisée du langage BPMN qui a été présenté dans le deuxième chapitre de ce manuscrit.

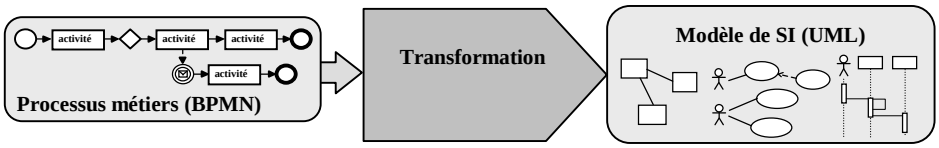


Fig A. 1. D'un modèle BPMN vers un modèle UML

1.1. Définition d'un diagramme BPMN valide

Définition 1.
Un diagramme BPMN *D* est composé de[1] :

- un ensemble *T* de « *task* »,
- un ensemble $T^{sp} \subset T$ de « *sub-process* »,
- un ensemble *E* d' « *event* », composé des sous-ensembles : E^s « *start event* », E^i «*intermediate event* », E^e « *end event* »,
- l'ensemble E^i peut être composé de deux sous-ensembles E^{im} « *intermediate message event* » et E^{tm} « *intermediate timer event* »,
- un ensemble *G* de « *gateway* » (paserelle), composé des sous-ensembles : G^p « *parallel gateway* », G^{dbi} « *data based inclusive gateway* », G^{ebe} « *event based exclusive gateway* » et G^{dbe} « *data based exclusive gateway* »,
- un ensemble *SF* de « *sequence flow* »,
- un ensemble *MF* de « *message flow* »,
- un ensemble *x.CMP.y* de « *compensation flow* », x et y∈ T, la tache *y* est exécutée pour compenser un problème d'exécution de la tache *x.*
- un ensemble *P* de « *pool* »,
- et un ensemble *L* de « *lane* ».

Pour que le diagramme BPMN soit valide, il doit respecter la grammaire du langage BPMN. Cette grammaire qui est détaillée dans le document officiel du BPMN dans sa version 1.0 [**BPMI 04**] peut être considéré comme un ensemble de contraintes à respecter pour s'assurer d'une construction correcte d'un diagramme BPMN. Nous montrons un aperçu de ces contraintes qui sont inspirées des travaux de **[Ouyang** *et al.* **06]** qui traitent, partiellement, la validité des diagrammes BPMN pour une traduction vers BPEL.

Définition 1a.
Un diagramme BPMN **valide** doit respecter les contraintes suivantes :

- C 1 : Un *« start event* » ne peut avoir de « *sequenceflow* » entrant et possède un seul « *sequenceflow* » sortant.

[1] Nous utilisons des termes anglais pour identifier quelques éléments BPMN (pool, lane, etc.) car il n'existe aucune traduction officielle. Nous préférons utiliser les termes originaux.

- C 2 : Un « *end event*» ne peut avoir de « *sequenceflow* » sortant et possède un seul « *sequence flow* » entrant.
- C 3 : Une « *task*» et un « *intermediate event* » possèdent un seul « *sequenceflow*» entrant et un seul « *sequence flow*» sortant.
- C 4 : Une « *parallel Gateway*» et une « *data based inclusive Gateway*» possèdent un seul « *sequence flow* » entrant et plus qu'un « *sequenceflow* » sortant.
- C 5 : Une « *event based exclusive Gateway* » doit être suivie par un « *intermediate event* » ou par une « *task* » de type « receive ».
- C 6 : Une « *pool* » doit commencer obligatoirement par un « *start event* » et finir par un « *end event*».
- C 7 : Une « *data based exclusive Gateway* » possède une branche conditionnelle par défaut.

1.2. Génération de diagramme de classes

1.2.1. Présentation de diagramme

Un diagramme de classe décrit la structure statique du système. Il décrit d'une manière abstraite les objets logiques du système en les assimilant à des classes. Ce diagramme décrit aussi les liens potentiels entre ces classes. Dans la suite, nous présentons une définition formelle de base de diagramme de classes :

Définition 2.
Un fragment de diagramme de classes D^{cl} est composé de :

- un ensemble *Pk* de « *package* »,
- un ensemble *Cl* de « *classe* »,
- un ensemble *At* d' « *attribut de classe* »,
- un ensemble *Op* d' « *opération de classe* »,
- un ensemble *Ass* d' « *association* » pour lier deux classes *x* et *y* : *x.Ass.y*
- un ensemble *Com* de « *composition* » pour lier deux classes *x* et *y* : *x.Com.y*
- un ensemble *Agr* d' « *agrégation* » pour lier deux classes *x* et *y* : *x.Agr.y*
- un ensemble *Gen* de « *généralisation* » pour lier deux classes *x* et *y* : *x.Gen.y*

1.2.1.1. La traduction vers le digramme de classe

La détermination d'un diagramme de classe relatif au système d'information à concevoir, relève d'une grande importance. La problématique spécifique à la réalisation de ce type de diagramme relève de la vue « structurelle » (et en moindre mesure de la vue« architecturale ») d'un tel système. Comme nous l'avons vu lors de l'analyse de la figure (Fig IV.3) et des relations de projection entre BPMN et UML, les diagrammes de classes nécessitent de s'appuyer sur une connaissance complémentaire et en particulier sur une architecture spécifique de SI compatible avec la collaboration étudiée. Ces architectures peuvent être logiques comme l'architecture orientée services. Dans ce cas, le diagramme de classe correspond à un PIM dans le sens de MDA. Mais, ces architectures peuvent être techniques comme des architectures ESB ou EAI. Dans ce cas, le diagramme de classe correspond à un PSM dans le sens de MDA.

Dans le but de générer des éléments de diagramme de classe, nous nous intéressons aux éléments : « *pool* », « *lane* » et « *message flow* ». En effet les « *pool* » et les « *lane* » servent dans un processus BPMN à répartir les tâches entre les acteurs du processus. Cette

connaissance sur l'organisation dans le processus peut alors être traduite en une autre connaissance sur l'organisation des classes dans le diagramme de classes. Ce sont les *« package »* qui occupent ce rôle.

Les *« message flow »* échangés entre les partenaires dans un processus collaboratif sont porteurs d'objets métier (facture, confirmation, bon de commande, etc). Dans un contexte collaboratif, ces objets ne possèdent pas la même structure de coté de l'expéditeur de message et de coté de son destinataire. Pour cette raison, pour un « *message flow* » correspondent deux classes (pour l'expéditeur et le destinataire du message). Nous pouvons alors déduire les trois règles de traduction suivantes :

Règle 1 (R1) :

$\forall x \in (P \cup L), \exists y \in Pk, x \xrightarrow{cor} y$, Pour tout élément « *pool* » ou *« lane »* de processus BPMN il existe un package « *package* » dans le diagramme de classe pour l'organisation des classes de diagramme.

Justification :

- Les éléments « *pool* », « *lane* » et « *package* » correspondent sémantiquement à une description organisationnelle.

Représentation graphique :

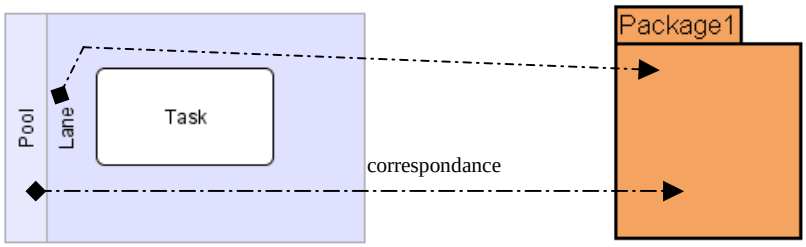


Fig A. 2. Présentation graphique de la règle 1

Règle 2 (R2) :

$\forall x \in L, \exists! y \in P, GetPool(x) = y \xrightarrow{cor} \exists (z, z') \in Pk, (z = R1(x), z' = R1(y)) \wedge z \subset z'$, toute inclusion entre une « *lane* » et une « *pool* » va être traduite en une inclusion de leurs packages relatifs obtenus en appliquant la règle 1.

Justification :

- Le lien d'inclusion entre une « *pool* » et une « *lane* » est hérité en passant dans le diagramme de classes. Les packages relatifs héritent cette relation d'inclusion.

Représentation graphique :

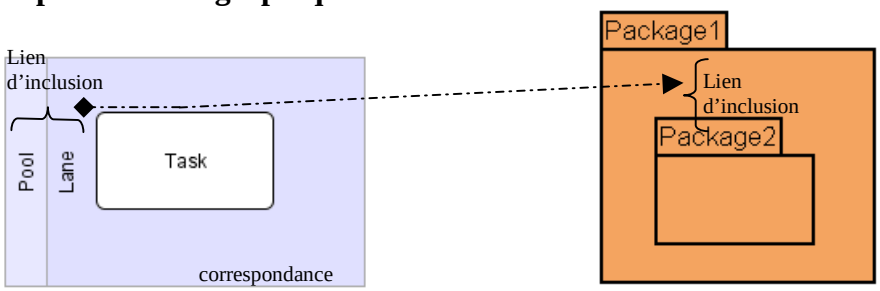


Fig A. 3. Présentation graphique de la règle 2

Règle 3 (R3) :

$\forall x \in Mf \xrightarrow{cor} \exists (y,z) \in Cl \wedge y \subset R1(getLaneSource(x))$

$\wedge z \subset R1(getLaneDestination(x))$

Pour tout « *message flow* » il correspond deux classes dans le diagramme de classe. Chaque classe appartient à un package déduit à partir de « *lane* » source ou destinatrice de « *message flow* ».

Justification :

- Lien (1) de passage entre vues de modélisation d'entreprise (vue fonctionnelle→vue informationnelle).
- Les « *message flow* » servent à lier deux « *pool* » différentes. Ces « *pool* » représentent deux entreprises différentes de la collaboration. Donc, il existe deux instances de classe : une pour le destinataire et l'autre pour l'expéditeur de message.

Représentation graphique :

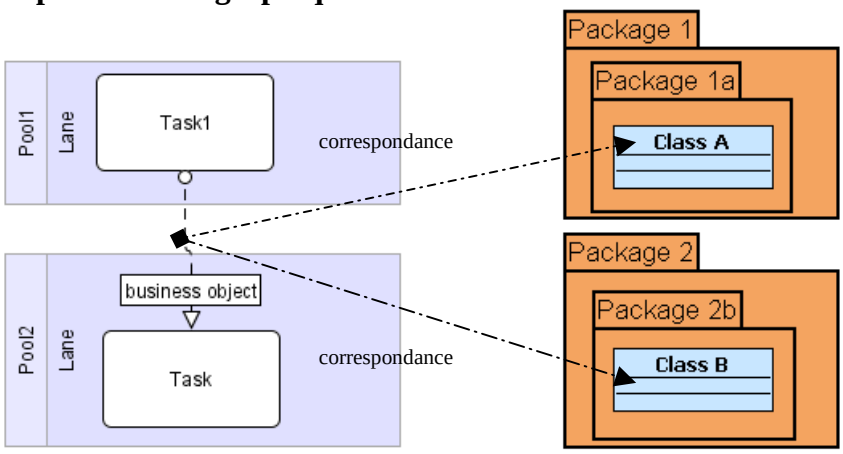


Fig A. 4. Présentation graphique de la règle 3

1.3. Génération de diagramme d'états-transitions

Les diagrammes d'états-transitions permettent de décrire les changements d'états d'un objet, en réponse à l'occurrence d'évènements, aux interactions avec d'autres objets ou avec des acteurs. Un diagramme d'états-transitions possède un seul état initial et plusieurs états finaux. Le changement d'état d'un objet correspond à la modification (d'une au moins) de ses caractéristiques internes. Dans la suite nous présentons une définition formelle de base de ce diagramme :

Définition 3.

Un fragment de diagramme d'états- transitions D^{et} est composé de :

- un ensemble *Et* de « *état* »,
- un ensemble *Tr* de « *transition* »,
- un ensemble *Fr* de « *fork* » de passerelles ouvrantes,
- un ensemble *Jn* de « *join* » de passerelles fermantes,
- un seul élément *Is* de « *Initial state* », pour l'état initial du diagramme,
- un ensemble *Fs* de « *final state* », pour les états finaux du diagramme,

- un ensemble *Ch* de « *choix* », pour le choix parmi un ensemble de transitions en se basant sur l'évaluation de conditions.

1.3.1.1. La traduction vers le diagramme d'états-transitions

L'exécution d'une tâche traitant avec un objet donné ou réalisé par cet objet constitue une source potentielle de changement d'état pour l'objet concerné. Ce constat est à l'origine du principe sur lequel nous proposons de nous appuyer afin de générer un modèle de diagramme d'états-transitions à partir d'un modèle BPMN.

La génération d'un diagramme d'états-transitions nécessite donc dans un premier temps de déterminer les objets qui seront décrits dans leur comportement. Ces objets sont crées, manipulés, véhiculés ou modifiés par les tâches de processus. Cette étape de choix peut être déterminée quasi-automatiquement sur la base de critères de sélection à partir du diagramme de classes [2] : classement des classes par nombre d'associations, par nombre d'attributs et/ou de méthodes, etc. Ce choix peut également (et sans doute plus justement) être fait par intervention du concepteur du SI (questions, propositions, etc.). Nous pouvons déduire alors ces trois règles de traduction :

Règle 4 (R4) :

$x \in T, IsLinkObject(Ob, x) = 1 \xrightarrow{cor} y \in Et$, Pour toute tâche « *task* » liée à l'objet dont nous voulons présenter les états dans le processus, il existe un « *état* » dans le diagramme qui correspond à l'exécution de cette tâche.

Justification :

- La simple exécution d'une tâche qui est fortement liée à un objet, change l'état de l'objet par rapport à son état avant l'exécution de la tâche. Exemple pour un processus de gestion d'une commande. L'objet « *commande* » subit plusieurs états : validée, traitée, facturée, payée, etc.

Représentation graphique :

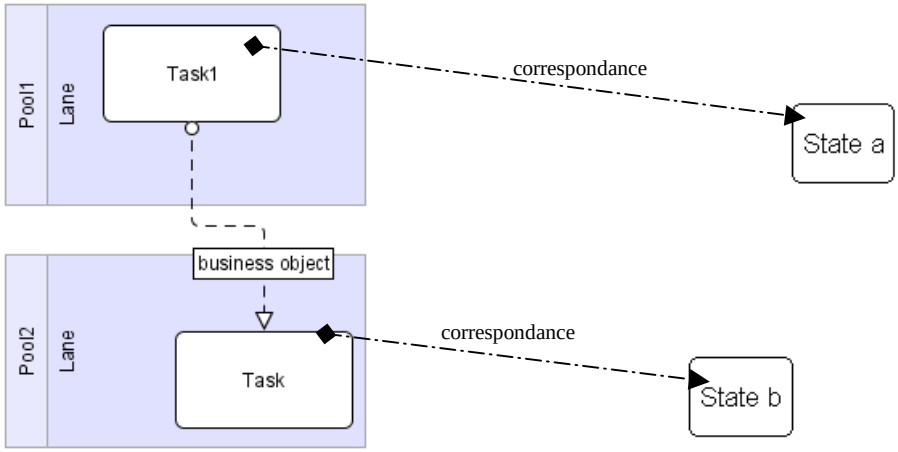


Fig A. 5. Présentation graphique de la règle 4

[2] Nous respectons ici l'ordre de création de digrammes UML dans un projet de conception de système d'information tel qu'il est décrit dans le livre de P. Roques **[Roques 04]**.

Règle 5 (R5) :

$x \in G^{dbe} \xrightarrow{cor} y \in Ch$, une « *gateway* » de type *data-based exclusive* correspond à un choix entre transitions en héritant exactement la même condition que celle permettant de choisir entre les tâches du processus en question.

Justification :
- Si les « *task* » de processus deviennent des « *état* » alors les « *gateway* » qui gèrent les conditions de passage entre « *task* » correspondent à des « *choix* » pour gérer les conditions pour passer entre « *état* ».

Représentation graphique :

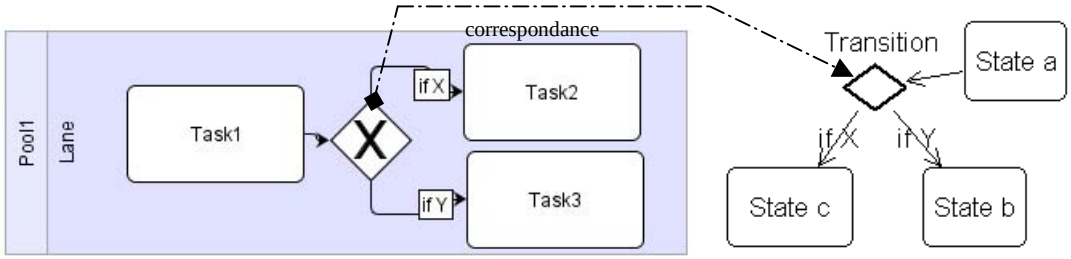


Fig A. 6. Présentation graphique de la règle 5

Règle 6 (R6) :

$x \in (Mf \cup Sf), IsLinkObject(Ob, x) = 1 \xrightarrow{cor} y \in Tr$, Pour tout lien (« *message flow* » ou « *sequence flow* ») entre taches liées à l'objet dont nous voulons présenter les états dans le processus, il existe une transition qui lui correspond dans le diagramme états-transitions.

Justification :
- Seules les « *message flow* » ou les « *sequence flow* » permettent de passer d'une tâche à une autre. De l'autre coté, seules les « *transition* » permettent de passer d'un « *état* » (qui correspond à une « *task* ») à un autre.

Représentation graphique :

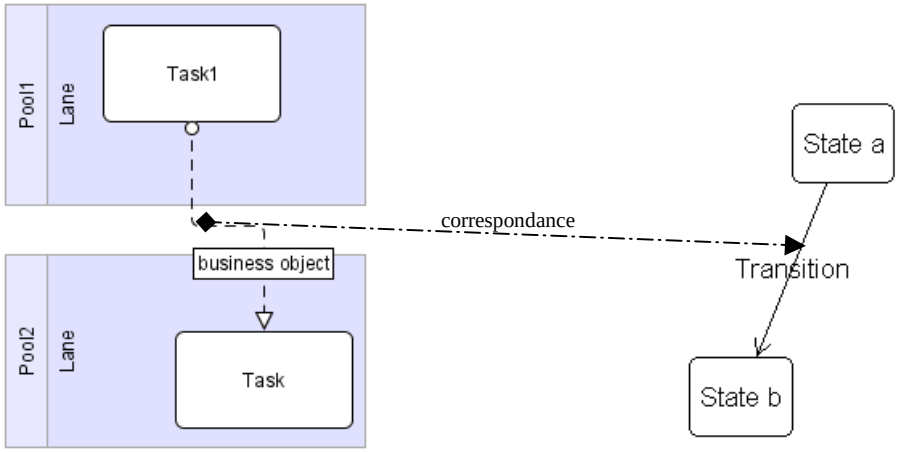


Fig A. 7. Présentation graphique de la règle 6

Règle 7 (R7) :

$x \in G^p \wedge IsLinkObject(Ob, x) = 1 \xrightarrow{cor} y \in Fr \vee y \in Jn$, un « *gateway* » de type *parallel* correspond à un « *fork* » ou à un « *join* » selon la manière avec laquelle elle est utilisée dans le processus.

Justification :
- Sémantiquement un « *gateway* » de type *parallel* peut jouer le rôle d'un « *fork* » pour commencer une synchronisation de « *task* » ou le rôle de « *join* » pour finir une synchronisation.

Représentation graphique :

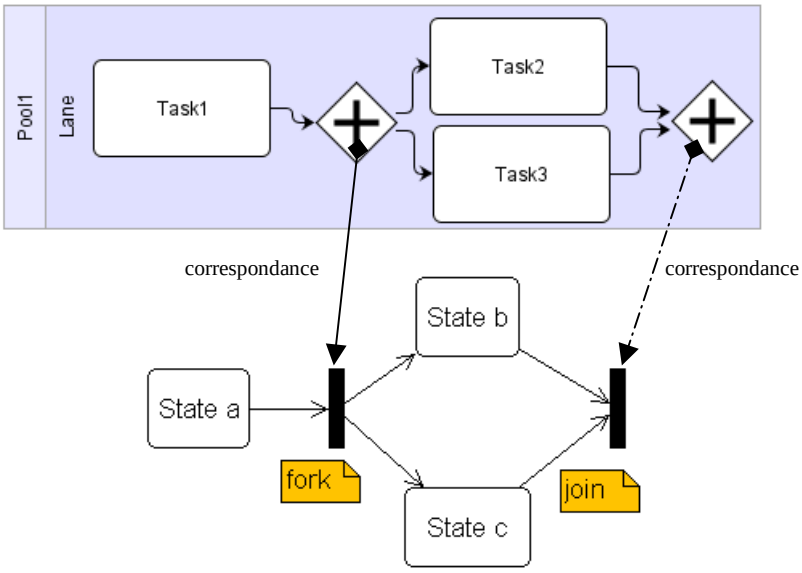


Fig A. 8. Présentation graphique de la règle 7

1.3.2. Génération de diagramme de séquence

1.3.2.1. Présentation de diagramme

Les diagrammes de séquence montrent un scénario d'interactions entre objets de système d'information selon un point de vue temporel. Le contexte des objets n'est pas représenté de manière explicite, comme dans les diagrammes de collaboration UML où la représentation se concentre sur l'expression des interactions. Dans ce diagramme, une barre verticale : ligne de vie, est construite à partir d'un acteur ou d'un objet.

Les communications ou messages[3] entre acteurs ou objets sont représentés par des flèches entre les lignes de vie sui correspondent concrètement à des appeles de méthode et objets. Ce diagramme permet d'identifier facilement l'ordre temporel de l'envoi des messages en suivant un ordre de lecture de haut en bas. A partir de l'envoi des messages[4], il est

[3] Un message est un élement de communication entre objets et acteurs qui déclenche une activité dans l'objet destinataire ; la réception correspond à un évenement.
[4] L'envoi des messages reléve essentiellement de l'activation de méthodes.

possible de créer ou détruire des objets ou d'avoir une période d'activités pendant laquelle un certain nombre de changements sont effectués.
Dans la suite nous présentons une définition formelle de base de ce diagramme :

Définition 5 :
Un diagramme de séquence D^{seq} est composé de :

- un ensemble *Lv* de « *ligne de vie* », qui représentent des objets.
- un ensemble *x.Mg.y* de « *message* », pour lier deux lignes de vie entre elles d'une manière asynchrone ou d'une manière synchrone,
- Un ensemble *Ex* d' « *exécution* », pour définir la période d'activité d'un objet ou d'un acteur.
- Un ensemble *Sig* de « *signal* »

1.3.2.2. La transformation vers le diagramme de séquence

Sur le plan métier, les partenaires collaborent ensemble (échange de données et de services) pour réaliser leur objectif commun. Cela correspond sur le plan du système d'information à une collaboration entre les différentes applications et objets des partenaires.

Les diagrammes de séquence montrent comment des objets communiquent pour réaliser certaines fonctionnalités. C'est donc un diagramme approprié pour la présentation des interactions[5] entre applications des partenaires. Ainsi une « *task* » particulière dans le processus BPMN correspond à une « *ligne de vie* » qui désigne l'objet qui permet de la supporter techniquement.

De plus, ce diagramme explique clairement l'invocation à partir d'un acteur ou d'un objet d'opérations ou traitements hébergés chez un autre acteur ou un autre objet. Un message contient, en effet, le nom de la méthode à invoquer et une définition des variables nécessaires pour l'exécution. Ces mécanismes sont utilisés dans un processus métier dans le cas ou un partenaire de la collaboration invoque une « *task* » d'un autre partenaire. Nous pouvons alors déduire cette règle de traduction.

[5] Le diagramme de séquence est appellé aussi diagramme d'interaction

Règle 8 (R8) :

$(x, y) \in T, z \in Mf \wedge x.z.y \xrightarrow{cor} (l1, l2) \in Lv \wedge \exists m \in M \wedge l1.m.l2$, un échange de message entre deux « *task* » correspond à deux « *ligne de vie* » qui représentent les ressources qui supportent les « *task* », échangeant des « *message* » entre elles.

Justification :
- Lien (2) de passage entre vues de modélisation d'entreprise (vue fonctionnelle→vue de ressources). Une tâche est forcément exécutée par une ressource du système d'information. Les messages entre les tâches sont les mêmes que ceux échangés entre les ressources.

Représentation graphique :

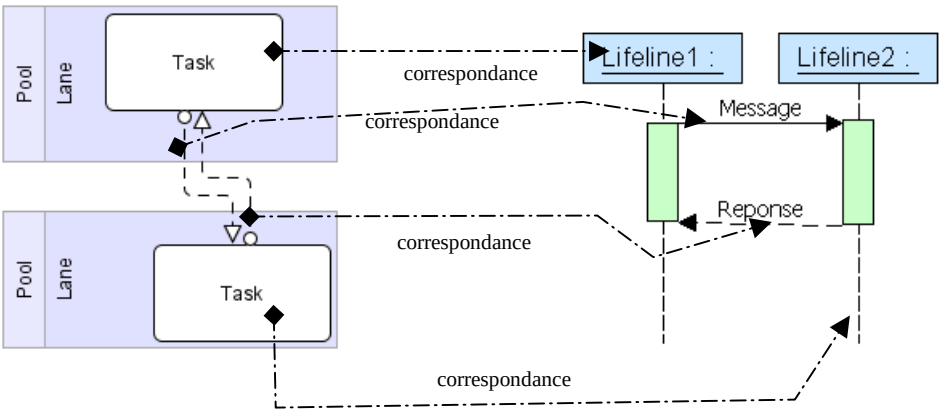


Fig IV. 1. Présentation graphique de la règle 13

1.3.3. Génération de diagramme d'activités

1.3.3.1. *Présentation de diagramme*

Le diagramme d'activités UML est le diagramme le plus proche d'un modèle BPMN : Son but est, en effet, de modéliser l'enchaînement logique des activités d'un processus. Ce diagramme, tout comme le formalisme BPMN représente une vision « événementiel » : il ne permet pas de décrire le point de vue fonctionnel[6] ou structurel du comportement du système (comme pourrait le faire par exemple un diagramme SADT ou IDEF0[7]).

Un diagramme d'activités permet de décrire le comportement d'un système sous la forme d'enchaînements ordonnés d' « *activité* » dédiées à la réalisation d'un objectif (processus). En décrivant les activités d'un processus, un diagramme d'activités veille à les classer par unité organisationnelle (qui présente l'acteur ou le rôle, selon le niveau d'abstraction) : « *partition* » chargée de l'exécution de l'activité. Un diagramme

[6] Nous entendons par « fonctionnel » la description des fonctionnalités de système. La méthode CIMOSA confond la vue fonctionnelle avec la vue de processus.
[7] Voir le deuxiéme chapitre de ce manuscrit.

d'activités peut également contenir des opérateurs de choix qui permettent de choisir parmi plusieurs branches d'activités selon la valeur d'une condition. L'aspect événementiel se limite aux deux événements de début et de fin de processus. Les deux opérateurs « *fork* » et « *join* » permettent respectivement de commencer et de finir une exécution en parallèle d'un ensemble d'activités. Dans la suite nous présentons une définition formelle de base de ce diagramme :

Définition 4.
Un diagramme d'activités D^{ac} est composé de :

- un ensemble *Ac* de « *activité*»,
- un ensemble *CF* de « *control flow* », pour illustrer les échanges entre les activités,
- un ensemble *Pa* de « *partition* »,
- un ensemble *Fr* de « *fork* » de passerelles ouvrantes,
- un ensemble *Jn* de « *join* » de passerelles fermantes,
- un seul élément *Is* de « Initial state », pour l'évènement de début de diagramme,
- un ensemble *Fs* de « final state », pour les évènements de fin de diagramme,
- un ensemble *Ch* de « choix », pour le choix parmi un ensemble d'activités en se basant sur l'évaluation de conditions,
- un ensemble *Com* de « commentaire », pour commenter les éléments de diagramme.

1.3.3.2. La traduction vers le diagramme d'activités

Le formalisme BPMN, vise à supporter la modélisation de tous les types de processus métiers de l'entreprise. BPMN se veut donc très expressif et dispose pour ce faire d'un vocabulaire riche afin d'être avantageux par rapport aux autres formalismes de modélisation de processus dont le diagramme d'activités d'UML. Nous considérons que les deux formalismes sont équivalents sémantiquement malgré les spécificités de chacun. BPMN se distingue notamment par une typologie riche de ces éléments (signes) : tâches (*Loop*, *Multiple instance*, *Adhoc* et *compensation*), événements (*timer*, *message*, *cancel* , etc.), gateways (*inclusiv*, *XOR Data-Based*, *XOR event-Based*, *complex*, etc.). UML, de son coté, et dans sa dernière version (UML2.0) se distingue par l'inclusion de capacités avancés pour la gestion des données de processus(« *data store* » pour désigner un entrepôt de données de processus) et la possibilité de créer des objets en cours de l'exécution de processus (« *create object action* » et « *destroy object action* »).

En conclusion, et d'après les travaux de [**Debauche** *et al.* **04**] et [**Morley** *et al*. **05**]**,** la grande différence entre BPMN et le diagramme d'activités UML réside dans le fait que BPMN est un formalisme de modélisation de processus métier alors que le digramme d'activités UML est un formalisme de modélisation de processus de système d'information[8]. T. Crusson, ingénieur chez Intalio[9] et auteur de plusieurs livres sur le BPM **[Crusson 03]**, va même plus loin en affirmant que : *« il n'existe pas de diagramme UML spécialisé pour la modélisation des processus métiers. UML propose uniquement un mécanisme d'extension pour la modélisation des processus métier en utilisant un profil approprié »*. Ce constat est confirmé par la fusion des deux organismes l'OMG (UML) et

[8] Nous avons exprimé la différence entre « processus métier », « processus système d'information » et « processus technique »
[9] www.intalio.com

BPMI (BPMN)[10]. Dans le futur, BPMN pourrait ainsi devenir un nouveau diagramme UML destiné à la modélisation des processus métier. Nous montrons dans la suite que pratiquement tous les éléments du langage BPMN peuvent correspondre à des éléments du diagramme d'activité UML.

1.3.3.2.a. Correspondance des éléments BPMN typés

Règle 9 (R 9) :

$\forall x \in D^{BPMN} \wedge isTyped(x) = 1 \xrightarrow{cor} y \in D^{ac} \wedge z \in Com$, le typage spécifique des éléments BPMN peut correspondre à un commentaire approprié et lié à un élément UML. La figure (Fig A.10) montre un exemple de l'utilisation de cette technique. Une tache temporisée en BPMN (modération pendant 7 jours) correspond à une activité basique UML lié à un commentaire qui montre cette temporisation.

Justification :
Le typage d'un élément BPMN est une coloration d'un élément BPMN basique. Un commentaire peut assurer une coloration équivalente d'un élément UML.

Représentation graphique :

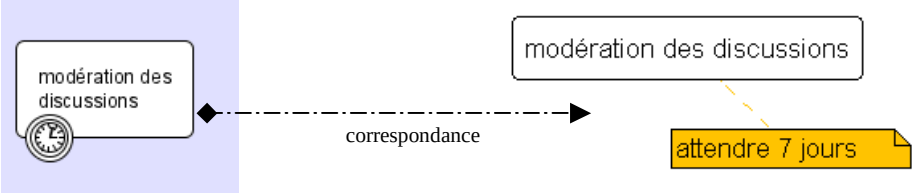


Fig A. 9. Présentation graphique de la règle 14

1.3.3.2.b. Correspondance des éléments BPMN basiques

Nous montrons dans le tableau suivant que la plupart des éléments BPMN basiques (non typés) correspondent directement à des éléments de digramme d'activité UML, selon le contexte d'utilisation de l'élément dans le processus BPMN. La justification de ces correspondances réside dans l'équivalence sémantique native entre BPMN et le diagramme d'activités UML.

[10] voir le site de l'OMG www.omg.org

Règle	*Elément BPMN*	*Elément UML*
Règle10 (R 10)	«task» *T*	«activité» *Ac*
Règle11 (R 11)	«sous-processus» T^{sp}	«activité» *Ac*
Règle12 (R 12)	«start event» E^{s}	«initial state» *Is*
Règle13 (R 13)	«intermediate event» E^{i}	«activité» *Ac*, permet d'envoyer un évènement au cours de l'exécution de processus
Règle14 (R 14)	«end event» E^{e}	«final state» *Fs*
Règle15 (R 15)	«passerelle» *G*	«join» *Jn*, «fork» *Fr*, «choix» *Ch*
Règle16 (R 16)	«sequence flow» *Sf*	«control flow» *Cf*
Règle17 (R 17)	«message flow» *Mf*	«control flow» *Cf*
Règle18 (R 18)	«pool» *P*	«partition» *Pa*
Règle19 (R 19)	«lane» *L*	«partition» *Pa*

Tab IV.2. Correspondance entre éléments BPMN basiques et diagramme d'activité

1.3.4. Synthèse

Nous avons présenté dans la section précédente des pistes quant à la traduction d'un modèle BPMN vers un ensemble de diagrammes UML qui sont primordiaux pour la conception d'un système d'information : le diagramme de cas d'utilisation, le diagramme de classe, le diagramme d'états-transitions, le diagramme de séquence et le diagramme d'activités. Nous pouvons caractériser notre travail selon un ensemble de points :

- **Formalisation et structuration des règles de transformation :**

L'intérêt de ce travail réside dans la présentation d'un ensemble de règles bien structurées et formalisées par des prédicats. Nous avons aussi assuré une préalable formalisation de fragments de méta-modèles source et cible de la traduction. Cette formalisation assure « *le maintien de la cohérence de la sémantique des concepts modélisés* » [**Baina** *et al*. **06**]. La formalisation est de nature aussi à faciliter par la suite une implémentation informatique de ces règles en utilisant un langage de transformation de modèles approprié.

- **Différence de répartition des règles de transformation :**

Nous avons remarqué que le nombre de règles identifiées entre BPMN et un diagramme UML varie d'un diagramme à un autre. En effet plus, le diagramme UML est proche sémantiquement de modèle BPMN plus il y'a des règles à identifier. Le diagramme d'activités est le diagramme qui possède plus de règles car il est proche nativement de BPMN (couverture des mêmes vues). La plupart des éléments BPMN sont traduisibles en des éléments de diagramme d'activités. Contrairement à cela, le diagramme de classe qui se focalise sur la vue informationnelle, n'offre que trois règles basées sur l'analyse des « *messages flow* », porteurs de données échangés dans la collaboration.

En conclusion, les règles de traduction sont réparties d'une manière non équilibrée sur les diagrammes UML, selon l'affinité sémantique que ces diagrammes peuvent avoir avec le formalisme BPMN. En effet, la traduction d'un élément BPMN vers un élément UML ne peut avoir lieu que s'il existe un lien sémantique fort entre les deux éléments. Ainsi et en se basant sur les travaux de [**Baina** *et al*. **06**] et si nous devons parler d'interopérabilité sémantique entre modèles BPMN et diagrammes UML, nous pouvons affirmer qu'il existe une interopérabilité sémantique partielle entre BPMN et le diagramme d'activités (une sous partie de diagramme d'activité correspond à une sous-partie de BPMN) et une

interopérabilité sémantique faible entre BPMN et les autres diagrammes UML (quelques liens existent). L'interopérabilité sémantique totale est inexistante car BPMN n'est totalement équivalent à aucun des diagrammes (pas d'isomorphisme total).

- **Incomplétude des diagrammes :**

En regardant les diagrammes obtenus, nous pouvons remarquer facilement un manque d'informations ou de détails. Les diagrammes générés sont parfois incomplets et creux. Ils comportent un ensemble d'éléments, mais pas une description détaillée de ces éléments ou d'autres éléments indispensables. Comme exemple, le digramme de classe généré décrit les classes déduites de diagramme BPMN (les objets métiers du processus BPMN). Mais aucune information sur les attributs de ces classes n'a pu être générée à partir de diagramme BPMN. La réponse à ce problème réside dans le langage BPMN lui même. Ce dernier ne peut pas véhiculer une telle connaissance car il n'est tout simplement pas fait pour. Le modèle obtenu doit donc être enrichi en se basant sur une connaissance additionnelle. Cette dernière permet de décrire les entreprises partenaires, ce qui permet de mieux comprendre leurs modes d'organisation, de fonctionnement, les objets métiers impliqués, etc. Cette connaissance est indispensable pour l'obtention de modèles de SI complets et spécifiques aux entreprises.

En conclusion, nous considérons que l'ensemble dés règles définis pour le passage d'un modèle BPMN à un modèle UML permettent de générer ce que nous pouvons appeler un « squelette » de vue logique (structurelle et comportementale) d'un système d'information. L'adjonction d'une connaissance complémentaire pour l'enrichissement des diagrammes est indispensable.

Annexe B

Le formalisme BPMN

Cette partie présente le formalisme de modélisation de processus **BPMN** en se basant sur le document de travail de **BPMI** (*Business Process Management Initiative*) : « Business Process Modeling Notation » du 25 août 2003. Nous allons pour cela articuler cette description autour de :

- Cadre d'application de BPMN
- Eléments de modélisation du formalisme
- Principe de modélisation (Exemple)

Pour donner une image et se faire une première idée, on peut très schématiquement positionner cette notation comme héritière à la fois des **Réseaux de Petri** et des **Statecharts** (même s'il ne s'agit absolument pas d'une description d'états mais bien d'un diagramme d'activité). Elle présente en effet le *Business Process* comme un ***enchaînement d'activités, processus et tâches*** (activés par des ***Token***) qu'on peut relier par le biais de ***connexions*** décrivant le flow (conditionnelles ou non) et caractérisés grâce aux ***attributs*** qui définissent chaque élément de modélisation et grâce au principe de ***décomposition*** d'une activité en sous-processus.

1. Cadre d'application de BPMN

BPMN est destiné à la modélisation de processus vus en tant que workflow. BPMN n'est pas adapté aux modélisations organisationnelles ou structurelles. Le formalisme s'adapte au niveau d'abstraction auquel on souhaite se placer et permet (principe de décomposition) de raffiner à volonté une description ou une partie d'une description de processus.

BPMN ne permet qu'une description dynamique du processus sans définir les composants de l'organisation qui seront responsables des tâches identifiées. Il s'agit d'une description « *fonctionnelle* » (identification d'activités) sans se préoccuper des entités physiques réelles qui implémenteront et réaliseront les actions identifiées.

Les diagrammes BPMN peuvent être directement reliés à du code BPEL4WS. Le document étudié (« Business Process Modeling Notation » de BPMI du 25 août 2003) présente comment traduire chaque configuration graphique issue d'un schéma BPMN en code BPEL4WS.

Enfin, il est important de noter que comme tout formalisme de modélisation, BPMN est dépendant du point de vue adopté : la modélisation d'un processus ne sera pas la même selon qu'on se place à tel ou tel point de vue.

2. Eléments de modélisation de BPMN

2.1 Présentation sommaire

BPMN décrit les activités à l'aide de *composants* reliés par des *liens* et groupés dans des *conteneurs*.

Composants de modélisation :

- Les événements (*events*)
- Les activités (*activities*)
- Les connexions (*gateways*)

Connecteurs graphiques :

- Les séquences (*sequence flows*)
- Les messages (*message flows*)
- Les associations (*associations*)

Conteneurs d'entités :

- Conteneur graphique (*pools*)
- Sous-conteneur (*lanes*)

Un diagramme BPMN est également caractérisé par ses attributs :

- **ID** (identifiant du diagramme),
- **Name** (description du diagramme),
- **Version** (numéro de version),
- **Author** (auteur du diagramme),
- **Language** (langue de conception),
- **CreationDate** (date de création),
- **Process*** (ensemble des processus modélisés : 0..n),
- **Pool+** (les conteneurs composant le diagramme : 1..n),
- **Documentation?** (éventuelle documentation complémentaire : 0..1)

2.2 Les éléments de modélisation

2.2.1 Les Composants

Les objets de modélisation (*événements*, *activités* et *passages*) disposent d'attributs communs :

- **ID** (identifiant du composant)
- **Name** (description du composant)
- **Assign*** (expressions évaluées à l'arrivée d'un token : 0..n)
- **Pool** (le conteneur dans lequel se trouve le composant)
- **Lane*** (la ou les lignes auxquelles appartient le composant)
- **Documentation?** (éventuelle documentation complémentaire : 0..1)

2.2.1.1 Evènements (Events)

Les évènements peuvent être de trois types différents : *start* (événement démarrant un processus), *intermediate* (événement en cours de processus) et *end* (événement terminant un processus).
Les règles de construction sont les suivantes :

- Il peut y avoir zéro ou plusieurs START EVENT et END EVENT.
- S'il y a (au moins) un START EVENT, il doit y avoir (au moins) un END EVENT.
- S'il y a (au moins) un END EVENT, il doit y avoir (au moins) un START EVENT.
- S'il n'y a pas de START EVENT, alors, les événement n'ayant pas de flow entrant constituent les points d'entrée du processus (ils doivent tous être « activés » lors de l'initialisation du processus).

- S'il n'y a pas de END EVENT, alors, les événement n'ayant pas de flow sortant constituent les points de sortie du processus (on doit attendre qu'ils soient tous « validés » pour terminer le processus).

Un événement est caractérisé par ses **déclencheurs** (*Trigger*) et ses **conséquences** (*Result*). Les événement START et INTERMEDIATE présente (au moins) un *Trigger* (éventuellement inconnu) et aucun *Result* alors que les END EVENT présente au moins un *Result* (éventuellement inconnu) et aucun *Trigger*.

Les symboles et attributs des événements sont les suivants :

Type d'évènement	**Symbole**	**Attributs spécifiques (en plus des attributs communs)**
START EVENT		**Trigger** (le type du déclencheur de l'événement parmi les types : *none, message, timer, rule, link…*) **Message/Timer/Rule...** (le contenu du Trigger)
INTERMEDIATE EVENT		**Trigger** (le type du déclencheur de l'événement parmi les types : *none, message, timer, rule, link …*) **Message/Timer/Rule...** (le contenu du Trigger)
END EVENT		**Result** (le Type de la conséquence de l'événement parmi les types : *none, message, exception, compensation, link…*) **Message/Exception/Compensation…** (le contenu du Trigger)

Les *Triggers* et les *Results* se répartissent (sur les évènements) et se représentent (par Type) comme suit :

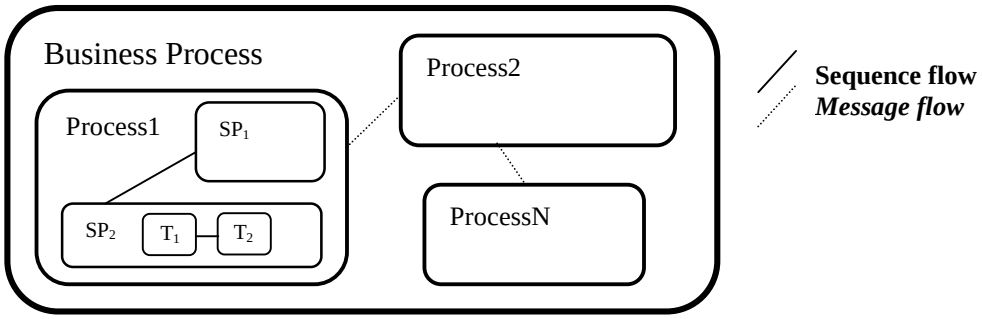
Type	**Symboles**			**Description Trigger**	**Description Result**
	Start	**Inter.**	**End**		
NONE				Le type du déclencheur est inconnu du modeleur (mais il existe)	
					Le type de la conséquence est inconnu du modeleur (mais il existe)
MESSAGE				Arrivée d'un message de la part de l'un des participants au processus	
					Un message est envoyé à l'un des acteurs en fin de processus
TIMER				Une date ou une situation temporelle particulière constitue le départ du processus	

EXCEPTION				Pour déclencher une exception (*Event* dans un flow normal) ou réagir (*Event* lié à une activité)	
					Une exception est générée qui fera réagir un *Intermediate Event* « à proximité »
CANCEL				Cet *Event*, lié à une activité de type *transaction* se produit lorsque celle-ci a été annulée	
					Cet *Event* a lieu pour une activité de type *transaction* et en indique l'annulation
COMPENSATION				Pour appeler une compensation (*Event* dans un flow normal) ou réagir (*Event* lié à une activité)	
					Une demande de compensation est générée qui fera réagir un *Intermediate Event*
RULE				Lorsqu'une assertion (ex: T°>10°C) devient vraie, l'*Event* est déclenché	
					
LINK				Relie un *End Event* (Result) d'un processus au *Start Event* (Trigger) d'un autre	
					Relie un *End Event* (Result) d'un processus au *Start Event* (Trigger) d'un autre
TERMINATE					
					Erreur fatale dans le processus, toutes les activités sont arrêtées
MULTIPLE				Causes multiples à l'*Event*, un seul de ces déclencheurs suffit à initier l'*Event*	

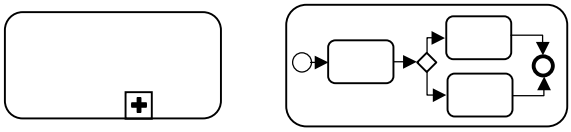
					Conséquences multiples de l'*Event*, elles se produiront toutes (cf. attributs)

2.2.1.2 Activités (Activities)

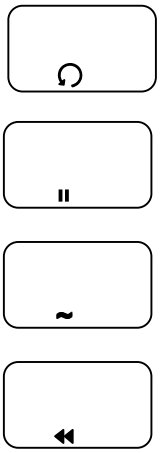
Les activités peuvent être élémentaires (*Task*), décomposables (*Sub-Process*) ou globales (*Process*). La hiérarchisation des *Business Process, Process, Sub-Process* et *Task* peut être schématisée comme suit :



Les *Process* ne disposent pas de symbole et l'on peut considérer qu'ils sont représentés graphiquement à l'aide des *Pools* (cf. § conteneurs). Les *Sub-Process* et *Task* sont représentés comme suit :

SUB-PROCESS	
TASK	

Les activités (*Task* et *Sub-Process*) peuvent présenter des caractéristiques graphiques supplémentaires traduisant certains aspects complémentaires de leur nature ou de leur comportement :



Loop : il s'agit d'une activité qui pourra être exécutée en boucle (la condition d'arrêt sera un événement lié à l'activité de type *Timer* par exemple)

Multiple Instance : l'activité peut être instanciée plusieurs fois en parallèle (par exemple, chaque chapitre d'un livre peut être rédiger selon le même processus, mais tous les chapitres peuvent être écrits en parallèle)

AdHoc : les activités décrites dans ce *sub-process* peuvent ne pas être ordonnées (par exemple, le nettoyage d'une pièce correspond au nettoyage spécifique de différente parties sans que le séquencement soit crucial)

Compensation : activité (faisant suite à une évènement de compensation) déclenchée si le *sub-process* global a été annulé afin de compenser l'une des activités spécifique du *sub-process*.

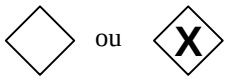
Ces caractéristiques des activités suivent les règles suivantes :

- Une activité peut comporter de 0 à 3 caractéristiques (indiquées en bas du symbole graphique)
- Une activité ne peut pas comporter simultanément les caractéristiques **Loop** et **Multiple Instance**
- Une tâche ne peut comporter la caractéristique **AdHoc** (réservé au seul *Sub-Process*)

Les activités de type *Process, Sub-Process* et *Task* présentent de nombreux attributs complémentaires précisant en particuliers les propriétés de l'activités ainsi que les éléments complémentaires liés aux caractéristiques **Loop, Multiple Instance, AdHoc** et **Compensation**.

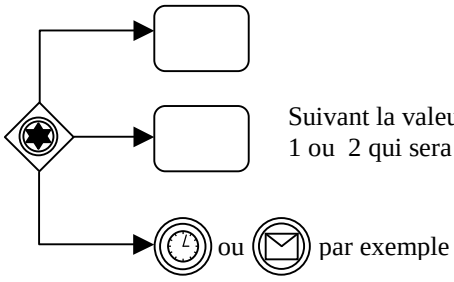
2.2.1.3 Passages (Gateways)

Les « passages » ou « aiguillages » (*Gateways*) servent à contrôler l'évolutions des flows dans le processus (convergence, divergence…). Il existe plusieurs sortes de *Gateways* à sémantiques distinctes :



XOR Data-Based : les flows sortants de ce *gateway* correspondent à diverses alternatives. Les tests conditionnant les embranchements sont évalués dans l'ordre et dès que l'un d'eux est valable, le token est transmis par le connecteur qui lui est relié. Ainsi, un seul chemin peut être pris (exclusif). Il est souvent utile de placer une option finale « Defaut » permettant de garantir le passage (BPMN ne prévoit pas le cas ou aucune alternative n'est possible en cours d'exécution).
Utilisé en tant que « merge » (point de jonction) de différents flows, ce *gateway* est une valve ouverte laissant passer successivement tous les tokens qui lui arrivent.

XOR Event-Based : Le principe de fonctionnement de ce *gateway* est identique au précédent à la différence près que les conditions déterminant quel chemin sera emprunté par le token dépendent d'un événement « attaché » au gateway :



Suivant la valeur de l'horloge (ou suivant la valeur du message reçu), c'est l'alternative 1 ou 2 qui sera choisie.

par exemple

OR : Les flows sortant de ce *gateway* sont conditionnés par des tests. Cependant, le fait qu'un chemin soit validé n'exclue pas l'évaluation des autres (comme pour le XOR). Ainsi, chaque condition vérifiée donnera lieu à un token. Comme pour le XOR, il est nécessaire de s'assurer qu'au moins l'une des option sera valable (BPMN ne prévoit pas le cas ou aucune alternative n'est validée).
Utilisé en tant que « merge » de différents flows, ce *gateway* est un synchronisateur, il fusionne les tokens qui lui arrive en simultané.

INCLUSIV : Ce *gateway* nécessite que tous les flows entrant soient actifs lorsqu'il est utilisé comme « répartiteur ». Lorsqu'il est utilisé comme « merge », ce *gateway* foutnit un token à chaque chemin sortant.

COMPLEX : Ce *gateway* sert à réaliser les situations qui n'ont pas pu être modélisées

Les *gateways* présentent de un certain nombre d'attributs précisant par exemple le type du *gateway*, le nombre de connexions qu'il exprime, leurs caractéristiques…

2.2.2 Les Conteneurs

Les conteneurs graphiques sont de deux types correspondant à une « hiérarchisation de la contenance » : les *pools* qui se décomposent éventuellement en *lanes*. Ces partitionnements présentent en particulier un intérêt pour représenter les *Business process* dans un environnement collaboratif « B2B ».

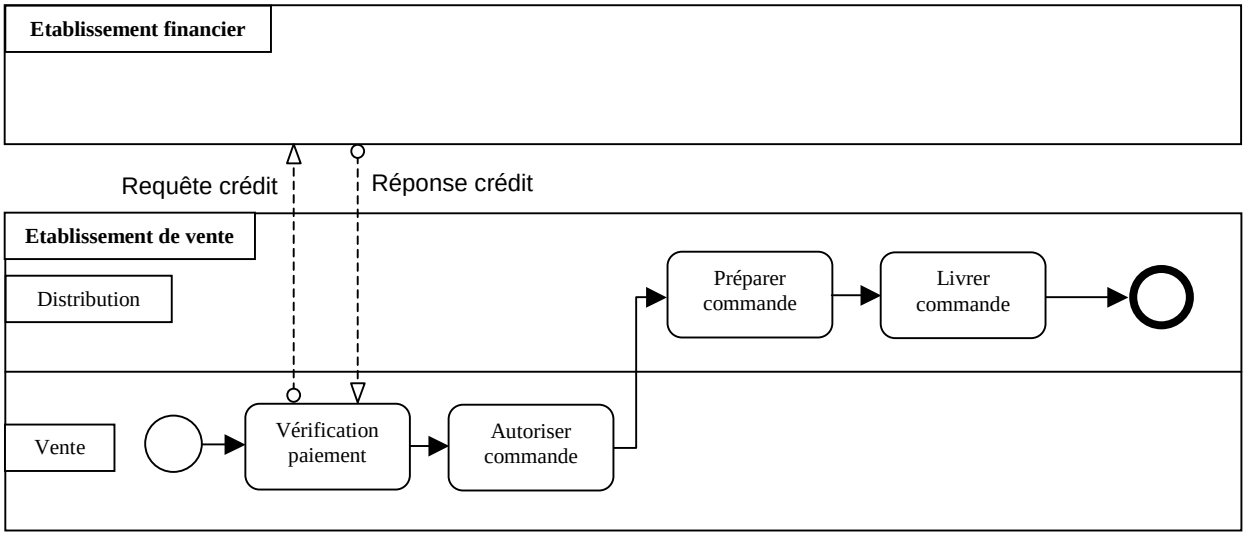
2.2.2.1 Pool

Ce conteneur permet d'isoler un ensemble de tâches du reste du modèle lorsqu'on représente une situation « Business to Business ». La *pool* peut être utilisée en tant que « boîte noire » (i.e. on ne représente pas le ou les processus qu'elle contient) ou en tant que « boîte blanche » (i.e. on détaille l'intérieur). Les seuls *flows* circulant entre les *pools* sont des *messages flows* (dans la description interne d'une *pool*, il peut bien entendu y avoir d'autres *flows* type *sequence flows*).

2.2.2.2 Lane

Il s'agit d'une sub-division de la *Pool*, destinée à organiser la *pool* (selon un critère de décomposition choisi par le modeleur). Les *lanes* peuvent elles-même être décomposées en *sublanes*.

Les *pools* et les *lanes* se représentent comme suit :



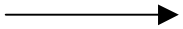
2.2.3 Les Connecteurs

Les connecteurs graphiques se classent selon qu'ils sont liés à un flow (*séquence* et *messages*) ou à une association graphique (par exemple d'un commentaire à un élément de modélisation) par le biais d'une connexion de type *association*. Usuellement, les *séquence* sont placées horizontalement et les *messages* verticalement (mais ce n'est qu'une convention de clarté).

2.2.3.1 Sequence flow

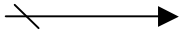
Ce connecteur est utilisé pour représenter l'ordre dans lequel les activités seront exécutées. Ce flow n'a qu'une seule origine et qu'une seule destination (toute deux de type *event*, *activity* ou *gateway*, et seulement ceux-là). Le *sequence flow* sert de passage pour le token d'un objet à l'autre (au sein d'une même *pool*). Un *sequence flow* peut franchir les limites d'une *lane* mais pas d'une *pool*.

Les différents symboles graphiques possibles pour un *sequence flow* sont les suivants :



Sequence flow basique

Sequence flow conditionné (dans la cas où ce *sequence flow* est relié à un *gateway*, le losange n'existe pas puisque le *gateway* stipulera lui-même l'aspect conditionnel)

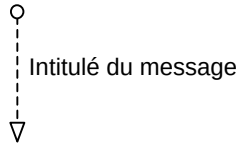


Sequence flow correspondant à une option par défaut (par exemple relié à un *gateway* de type XOR)

2.2.3.2 Message flow

Ce connecteur est utilisé pour représenter l'échange de message entre deux entités susceptibles d'envoyer ou de recevoir un message. Une règle important régit l'utilisation de ce type de *flow* : Un *message flow* doit relier deux *pools* ou deux entités situées dans deux *pools* différentes. Un *message flow* ne peut pas connecter deux objets d'une même *pool*.

La représentation graphique d'un *message flow* est la suivante :



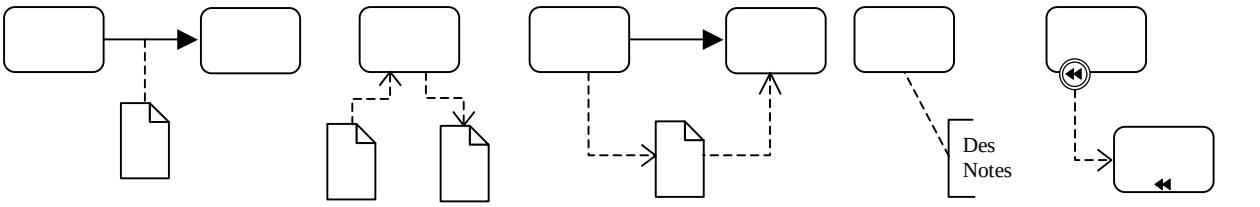
2.2.3.3 Association

Ce connecteur est utilisé dans divers cas :

- Pour associer un élément de documentation à une entité
- Pour représenter des données en tant qu'entrées ou sorties d'une activité ou comme élément du flow (ce que représenterait réellement le token).
- Pour associer une compensation à une activité

En outre, une association peut être directionnelle ou non.

Les utilisations graphiques les plus courantes pour une *association* sont les suivants :



Donnée associée à un *sequence flow*

Donnée en entrées et en sorties de l'activité

Donnée reliée au *flow* (en tant qu'élément du flow)

Commentaire sur l'activité

Association d'une tâche de compensation

2.2.3.4 Tableau récapitulatif des connexions

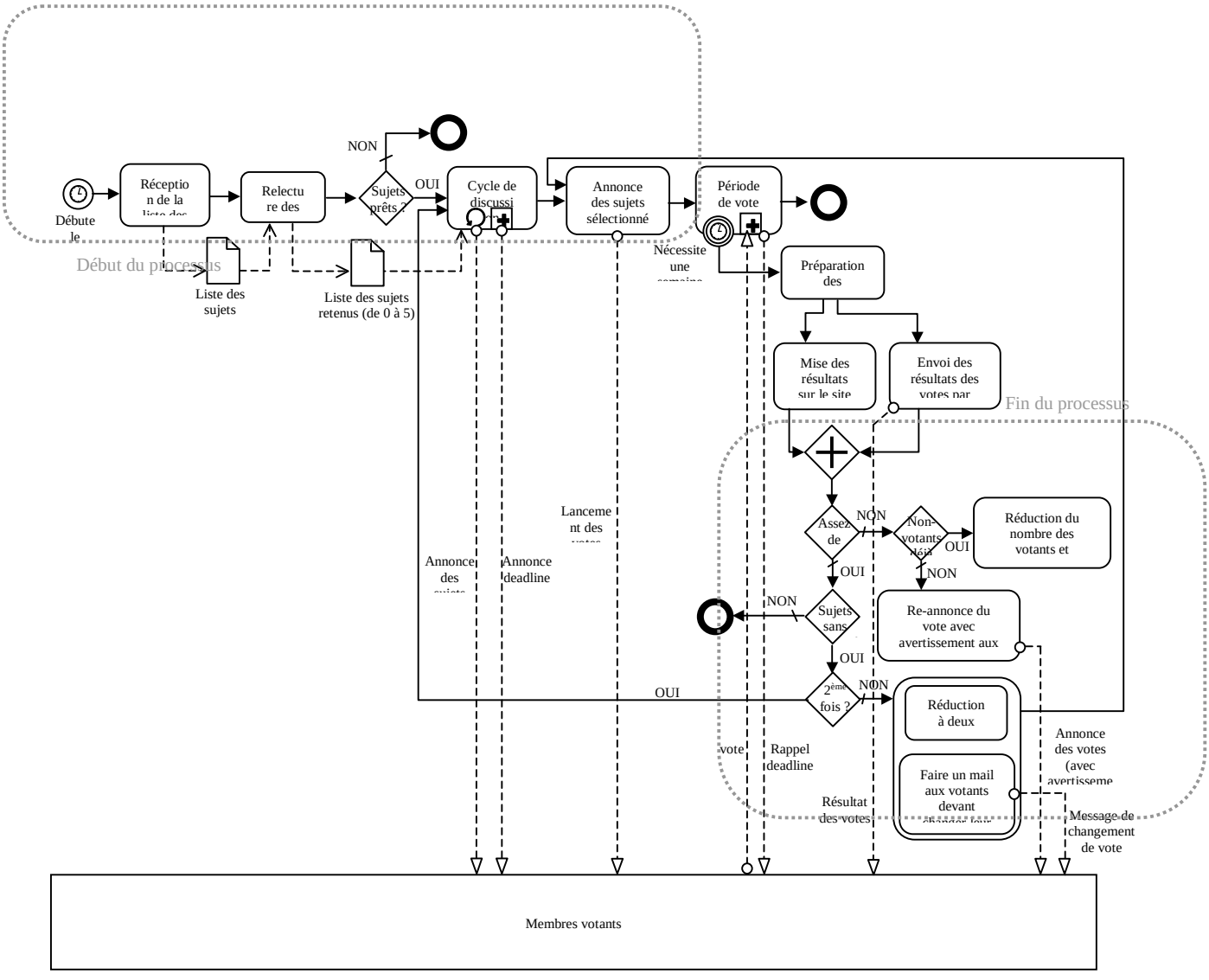
Les différentes connexions possibles entre les divers composants de modélisation (*sequence flows* et *message flows*) peuvent être représentées à l'aide du tableau récapitulatif suivant :

Dest. / Source							

Message flow
Sequence flow

3. Mise en œuvre de BPMN : Traitement d'un exemple

3.1 présentation globale



On peut expliciter cet exemple en détaillant certaines de ses composantes et en se penchant sur l'interprétation de certaines parties de ce diagramme BPMN.

3.2 Présentation détaillée

Afin d'améliorer la lecture du diagramme précédent, il peut s'avérer intéressant de s'intéresser plus en détails à l'interprétation des parties *début du processus* et *fin du processus* et également de détailler les deux sous-processus *cycle de discussion* et *période de vote*.

3.2.1 Début Du Processus

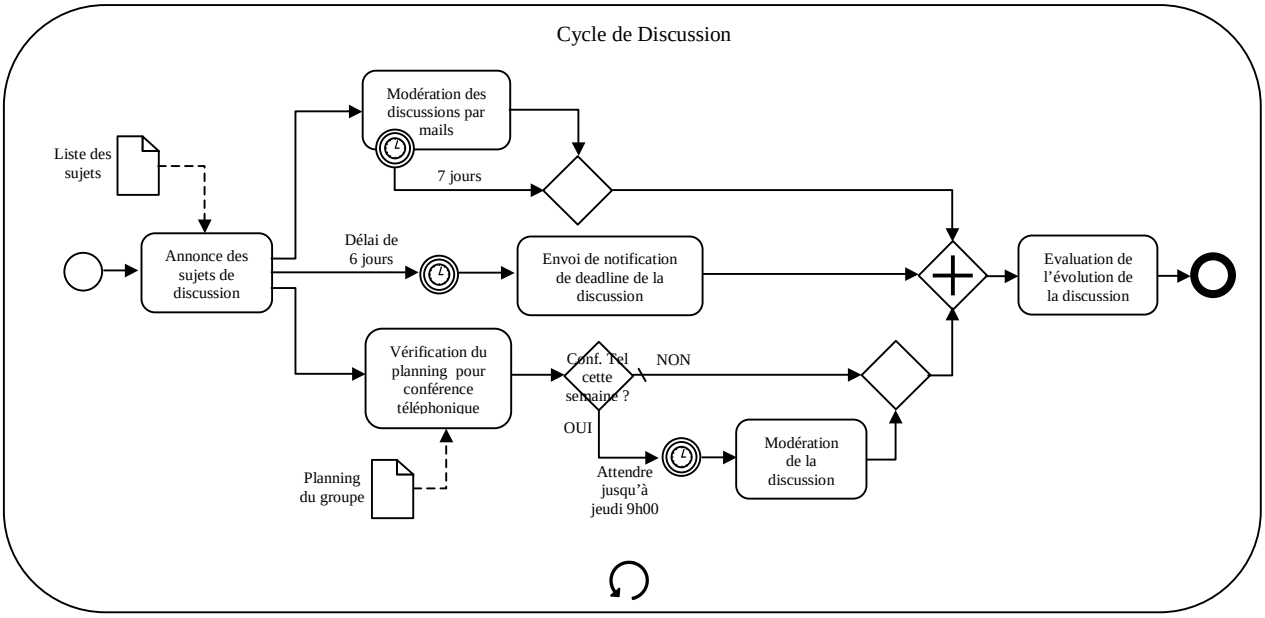
Cette partie du diagramme précédent correspond à la modélisation selon le formalisme BPMN du processus réel suivant : « Le gestionnaire de listes de sujets passe en revue la liste des sujets potentiels et choisit ceux qui sont susceptibles d'être résolus par le cycle de discussion et de vote. La décision qui en découle (soit la réponse à la question *sujets prêts ?*) engendre le déclenchement du processus de discussion et de vote ou annule le cycle pour la semaine en cours (en attendant la semaine suivante). Cette activité de discussion (représenté par le sous-processus *cycle de discussion* présent deux *flows* en entrées qui correspondent en fait aux deux façons d'initier ce sous-processus : (i) suite à la présence de sujets à traiter ou (ii) suite au choix de soumettre à nouveau au vote un sujet dont le traitement ne s'est pas avéré concluant (pas de majorité). Ce cycle de discussion sera développé dans un paragraphe suivant. »

3.2.2 Fin Du Processus

Cette partie du diagramme correspond à la modélisation selon le formalisme BPMN du processus réel suivant : « Le critère de validation de résultat du vote concerne le taux de participation. Si le quota de votants n'a pas été atteint (par exemple 2/3 des membres du groupe), les sujets de la semaine courante ne peuvent être résolus. Il faut alors se poser la question du maintien dans les listes des votants improductifs. En effet, si l'un des membres du groupe ne participe pas à un vote (sans prévenir, c'est à dire sans s'exclure du nombre des votants pour cette semaine), il est averti ; s'il ne participe pas à un second vote (dans les mêmes conditions), il est alors exclu du panel des membres votants et le taux de participation peut alors être recalculé sans les membres exclus (s'il n'est toujours pas satisfaisant, les sujets sont reconduits sur la semaine suivante).
Si le taux de participation est suffisant, il faut alors savoir si les sujets proposés ont étés traités : autrement dit, on se penche sur le résultat des votes et sur l'émergence ou non d'une majorité. Dans le cas ou la majorité existe et est satisfaisante (entre les différentes possibilités offertes), le processus est terminé. Par contre, si la majorité n'est pas atteinte, le processus prévoit de ne conserver que les deux options les plus significative et de relancer les membres dont le vote doit être changé (ceux qui n'avaient pas voté pour l'une des deux options conservées). Néanmoins, si cette voie n'est pas choisie – par exemple parce que le vote courant correspond déjà à un tel raffinement ou parce que le nombre d'options est difficilement réductible ou encore parce que les options à conserver ne sont pas suffisamment en avance sur les autres pour être significatives – on relance le sujets dans la boucle complète de décision et en particulier dans le cycle de discussion.

3.2.3 Cycle De Discussion

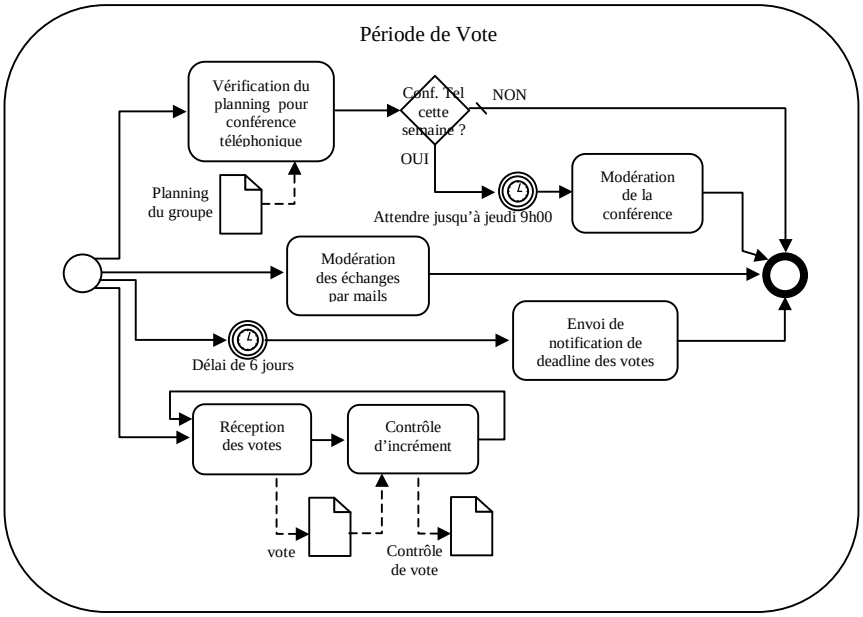
Ce sous-processus est régit par le diagramme BPMN suivant :



Il s'agit de trois activités menées en parallèle sur la semaine de discussion (la gestion des échanges par mails, l'envoi de l'avertissement concernant la deadline pour la discussion et la gestion d'une possible conférence téléphonique durant la semaine). Il est à noter que les deux *gateways XOR data-based* utilisés dans ce diagramme permettent de garantir le bon fonctionnement du *gateway Inclusiv* en « unifiant » les tokens sortant de l'activité *modération des discussions par mails* et en faisant la même opération avec ceux sortant du test sur la présence ou non d'une conférence téléphonique durant la semaine courante.

3.2.4 Période De Vote

Ce sous-processus est régit par le diagramme BPMN suivant :



Ce sous-processus fait suite à l'annonce faite par le gestionnaire de la liste des sujets que les votes peuvent commencer (ou à une relance des votes en cas de résultat non-satisfaisant). Il

s’agit de quatre branches parallèles qui concernent la gestion des conférences téléphoniques, la modération des échanges par mails, l’annonce de la deadline pour les votes et enfin, le décompte des voix. Il faut noter que l’incrémentation des votes correspond à une boucle infinie qui traduit le fait que chaque membre votant peut changer d’avis durant la semaine (d’où l’intérêt des discussions par mails) et peut donc revenir voter. Le vote est donc un « annule et remplace » qui nécessite la présence d’un contrôle permettant ce multiple vote sans fausser les résultats.

Résumé :

Dans un contexte de collaboration industrielle ou inter-organisationnelle, la qualité de l'intégration des différents partenaires dépend grandement de la capacité de leurs systèmes d'information (SI) à interagir efficacement. C'est pourquoi nous proposons dans ce manuscrit d'aborder cette problématique selon l'angle de l'interopérabilité des systèmes d'information. Dans notre approche, l'interopérabilité des SI des partenaires s'appuie conceptuellement sur deux caractéristiques : (i) la faculté de ces SI à se conformer à une orientation services (SOA pour Service-Oriented Architecture) et (ii) le positionnement au sein du réseau de partenaires d'un système d'information médiateur destiné à assurer l'intégration du « système de systèmes » ainsi créé. Ces travaux de thèse traitent précisément de la conception de ce médiateur selon une démarche MDA (Model Driven Architecture). Notre approche consiste à étudier la traduction d'un modèle des besoins situé au niveau « métier » en un modèle d'architecture spécifique situé au niveau « logique ». Pour cela, nous allons considérer que la connaissance contenue dans un modèle de processus collaboratif (formalisé en BPMN, au niveau CIM de l'approche MDA) pourrait permettre d'alimenter une démarche de modélisation de ce SI médiateur (formalisé en UML, au niveau PIM de l'approche MDA). Ce travail comporte et après avoir exposé les fondements théoriques de ces propositions, une présentation des principes de traduction de ces modèles, ainsi que les règles de transformation qui les concrétisent. Enfin, la maquette outil logiciel supportant cette démarche et permettant d'assurer cette modélisation du médiateur a été réalisée.

MOTS-CLES : **Système d'Information, Interopérabilité, Transformation de modèles, SOA, MDA, Modélisation, Processus Collaboratif, Médiation, Système de systèmes.**

Abstract :

In a collaborative context, the integration of industrial partners deeply depends of the ability of their Information Systems (IS) to interact efficiently. In this document we propose to tackle this point according to the point of view of IS interoperability. Interoperability of partners' IS is based on two main aspects: (i) the fact that partners' IS respect SOA (Service-Oriented Approach) concepts and (ii) the support of a Mediation Information System (MIS) able to put IS together in one single merging System of Systems (SoS). We propose to design such a MIS according to MDA (model-Driven Approach) principles. We aim at using business model (the needs) to design a logical model of a solution (logical architecture). The business model is a collaborative business model (in BPMN, at the CIM level), while the logical model is a MIS model (using UML, at the PIM level). This document presents the theoretical aspects of this subject, the mechanisms of transformation and the dedicated translation rules. Finally, we show the prototype of a demonstration tool embedding the transformation rules and running those principles.

KEYWORDS: **Information System, Interoperability, Model Transformation, SOA, MDA, Modelling, Collaborative Process, Mediation, System of Systems.**