Coopération interentreprises à la demande : Une approche flexible à base de services adaptables

Khouloud Boukadi

N° d'ordre : 541 I

THÈSE

présentée par

Khouloud Boukadi

Pour obtenir le grade de Docteur
de l'École Nationale Supérieure des Mines de Saint-Étienne

Spécialité : Informatique

Coopération interentreprises à la demande :Une approche flexible à base de services adaptables

Soutenue le 05 Novembre 2009 à Saint-Étienne

Membres du jury

Président :	Hervé Pingaud	Professeur, École des Mines d'Albi-Carmaux
Rapporteurs :	Corine Cauvet	Professeur, Université d'Aix Marseille 3
	Boualem Benatallah	Professeur, University of New South Wales, Australie
Examinateur :	Hanêne Ben Abdallah	Maître de Conférence, Université de Sfax (Tunisie)
	Olivier Boissier	Professeur, École des Mines de Saint Etienne
Directeurs de thèse :	Patrick Burlat	Professeur, École des Mines de Saint Etienne
	Lucien Vincent	Maître de Recherche, École des Mines de Saint Etienne

REMERCIEMENTS

On passe tous par des moments de doute, d'incertitude, de petites et grandes crises …et on ne peut les dépasser que par le soutien de tous ceux qui nous relèvent de nos petites peines. Pour cela et pour bien d'autres raisons je tiens à remercier :

Monsieur **Hervé Pingaud**, Professeur à l'École des Mines d'Albi-Carmaux, qui a accepté de présider le jury et d'examiner mon travail.

Madame **Corine Cauvet**, Professeur à l'Université d'Aix Marseille 3, et Monsieur **Boualem Benatallah**, Professeur à l'Université New South Wales (Australie), qui m'ont fait l'honneur de rapporter mon travail et pour leurs remarques enrichissantes.

Madame **Hanêne Ben Abdallah**, Maître de Conférence à l'Université de Sfax (Tunisie), et Monsieur **Olivier Boissier**, Professeur à l'École des Mines Saint-Etienne qui ont accepté de faire partie de mon jury en tant qu'examinateur.

Mon directeur de thèse, **Lucien Vincent** pour sa sympathie, son énergie positive, pour les discussions de travail, sa disponibilité et son écoute. Plus qu'un directeur je sentais tout le temps qu'il faisait attention à moi comme un père garde sa fille. Si j'écris ce manuscrit c'est parce qu'il a tout le temps cru en moi et n'a pas hésité une seconde à me remonter le moral.

Mon directeur de thèse **Patrick Burlat** de m'avoir accueillie dans son équipe, d'être tout sourire en m'offrant ses conseils méthodologiques.

Xavier Boucher qui a bien voulu être mon rapporteur interne, pour son effort qu'il a généreusement consacré à l'évaluation de mon travail de thèse et pour ses conseils et ses remarques qui ont été bénéfiques à l'aboutissement de ce document.

Chirine Ghedira, **Djamel Benslimane** et **Zakaria Maamar** pour l'intérêt qu'ils ont porté à mes questions et leurs conseils lucides. Les discussions échangées tout au long de ma thèse m'ont permis de bénéficier de leur expertise, réflexions et critiques.

Ma chère famille **Maman**, **Papa**, ma petite **Dhouha** chérie, **Malek** et **Rima** pour m'avoir fourni un cadre agréable et chaleureux qui m'a toujours aidé à me ressourcer et qui m'a permis quand mes poèmes devenaient lugubres de m'apercevoir que la vie est belle.

Mes **grands parents**, qui malgré toutes les difficultés de la vie, sont, je suis sûre, fiers de moi. Je n'oublirai jamais que c'est grâce à vous que je suis arrivée jusqu'ici.

Un merci énorme à ma meilleure amie **Natacha**, pour les moments forts qu'on a passé ensemble. Tout l'été à rédiger la thèse, tout en ayant pour rayonnement que celui de nos écrans. Mais c'est dans les moments de « combat » que naissent les plus grandes amitiés. Natacha, j'ai adoré rédiger mon manuscrit à tes cotés.

Mes sentiments de gratitude à Sonia, Evgeny, Victor ,Olga, Amin, Nicolas, Xsénia, Mohamed, Bertrand pour l'ambiance décontractée, les moments de détente et de joie passés ensemble, pour les pauses café, les restaurants organisés, etc.,

À toutes les personnes du centre G2I qui répondent toujours présentes quand nous avons besoin d'elles, et particulièrement à **Liliane Brouillet**, à **Marie Line Barneoud**, aux membres : de l'équipe informatique, du centre de documentation, des autres services techniques et à tous ceux que j'ai oublié de citer, qu'ils m'en excusent.

À tous mes amis, ceux que je vois encore et ceux que le vent a emmené vers d'autres rives.

RÉSUMÉ

Aujourd'hui la définition des nouveaux modèles d'organisation d'entreprises est guidée par l'intensification de la concurrence, la variabilité des demandes clients et la performance des communications. Dans ce nouveau contexte, les entreprises ont compris l'importance des coopérations interentreprises et mettent en place des modèles basés sur la coopération (entreprise virtuelle, réseau d'entreprises,…). Ces formes d'organisation ont un impact important sur les systèmes d'information. En particulier, la flexibilité et l'ouverture vers l'environnement deviennent des enjeux majeurs dans la conception de ces systèmes.

Nos travaux de recherche s'intéressent à la question de coopérations à la demande en considérant que le système d'information est un élément central de cette problématique. Notre étude de l'état de l'art a révélé que la coopération à la demande est un choix stratégique difficile à réaliser vu que le système d'information n'est pas adapté à ce type de fonctionnement. En effet, l'adhésion d'une entreprise à des scénarios de coopération est régie par une double préoccupation. D'une part, l'entreprise présente un manque de flexibilité au niveau du système d'information, support à son métier et d'autre part, mettre en oeuvre une coopération à la demande exige le développement d'un cadre bien défini qui permettra aux entreprises d'interconnecter leurs différents processus au sein d'un processus global.

L'architecture orientée services et la technologie service Web semblent proposer des réponses crédibles aux besoins tant au niveau interne (système d'information support à la coopération) qu'au niveau externe (services disponibles aux partenaires). Ainsi, nos travaux de recherche ont eu pour objectif principal de développer une nouvelle approche qui assure l'efficacité et l'efficience de la coopération interentreprises basée sur l'approche service. Ils se sont focalisés sur trois sous problématiques complémentaires qui sont respectivement la problématique de construction (ou de migration vers) d'une architecture orientée services au sein de l'entreprise, la problématique de construction du processus coopératif à la demande et la problématique d'adaptation de la logique métier des services au contexte permettant de doter les services d'un certain niveau de flexibilité.

Nos travaux de recherche reposent sur trois principes méthodologiques qui sont une perspective d'ingénierie, l'intégration de la flexibilité et l'ouverture. L'inscription dans une perspective d'ingénierie permet de suivre une démarche méthodologique qui guide l'identification du système attendu en utilisant un ensemble de méta-modèles et de modèles. L'intégration de la flexibilité se manifeste à travers la prise en compte du contexte et de l'adaptation d'une manière transversale tout au long du cycle de vie. Le principe d'ouverture impose de s'inscrire dans le cadre d'utilisation des standards industriels.

Mots clés : Coopération interentreprises, Architecture orientée service, Service Web, Adaptabilité, Contexte, Programmation orientée aspect.

ABSTRACT

Today, enterprises face rapidly changing market conditions, new competitive pressures, new regulatory fiats that demand compliance, and new competitive threats. To cope with these business conditions, enterprises migrate to inter-organizational relationships as a way to adapt to their new environment, increase their efficiency, and gain competitive advantages.

Suitable enterprise collaboration that fits with these large inter-organizations requires a high level and important change. It is clear that on demand enterprise collaboration drives the needs for the IT infrastructure to respond quickly, and support new business models instantaneously. Flexibility, adaptability and modularity features are the main characteristics of the target architecture. However, enterprise information systems were not designed to evolve with this new way of thinking. This adds another level of intricacy to an already complex IT infrastructure For this reason, building effective enterprise collaboration is not an easy task: it requires a new generation of IT architecture with more agility, flexibility and modularity.

A contemporary approach for addressing these critical issues is the Service Oriented architecture (SOA). Nowadays, SOA becomes the most promising paradigm for leveraging enterprise information systems. It creates opportunities for enterprises to provide value added service. Usually, this paradigm models business activities as a collection of self-contained services available across the enterprise and can be used through standard protocols internally and externally. Although the hype surrounding SOA is widespread, the concept is still in its infancy with regards to actual implementations. Several problems have arisen. Issues like the lack of precise definitions for the SOA concepts involved and the enormous demand for process guidance and proven best practices in these projects are among the most frequently discussed in both industry and academia.

Thus, our research aims at developing a new approach that ensures the effectiveness and efficiency of business cooperation based on the service concept. Our approach focuses on three complementary issues that are respectively: implementing service oriented architecture in the enterprise, building collaborative business process based on service compositions and ensuring the service adaptability to context changes.

Our research is based on three major principles that are the engineering perspective, integration of flexibility and openness. First, adopting an engineering perspective implies a comprehensive guiding method and strong design principles based on a set of models and meta-models. Second, the flexibility implies considering service adaptability in order to accommodate the changeable situations. Third, the openness principle imposes us to use and to conform to industrial standards.

Key-Words: Enterprise Collaboration, Service oriented architecture, Web Service, service adaptation, context-aware Aspect Oriented Programming.

TABLE DES MATIÈRES

LISTE DES TABLEAUX

Page

LISTE DES FIGURES

Page

INTRODUCTION GÉNÉRALE

Table des matières

1. Coopération interentreprises à la demande : motivations et contraintes

1.1 Enjeux de la coopération interentreprises

L'environnement des entreprises a fortement évolué ces dernières années, il est devenu plus complexe et imprévisible, de plus, les changements technologiques ont fortement influencé l'organisation des entreprises. L'intégration de ces changements devient indispensable pour assurer la survie de l'entreprise, non seulement au niveau d'une seule entreprise, d'un secteur, ou d'une économie donnée, mais au niveau mondial. Comme le souligne [Thisse, 2004], les entreprises essaient aujourd'hui de « rendre carré le cercle » car elles doivent d'une part

répondre aux enjeux concurrentiels en améliorant la performance industrielle en termes de coût, délais, *etc.* et d'autre part, répondre aux problématiques d'ouverture (relations entre partenaires, partage de l'information et intégration dans les systèmes d'information des partenaires, *etc.*).

L'ensemble de ces facteurs, combinés avec la mondialisation des marchés, contraignent l'entreprise à acquérir un avantage concurrentiel sur son métier d'origine. Nombreuses sont les entreprises qui ont compris l'importance des coopérations pour assurer leur survie. Ainsi, divers rapports se créent entre clients, fournisseurs, concurrents et autres partenaires. Ces rapports de coopération s'étalent sur un continum qui peut varier d'une dépendance totale ou partielle (fusions-acquisitions, prise de participation, *etc.*), à des accords ponctuels sous forme contractuelle, ou encore à la formation de formes réticulaires entre les différents intervenants.

Par ailleurs, le mouvement de réorganisation des entreprises vers des réseaux métiers a trouvé son essor grâce au développement accéléré des nouvelles technologies de l'information et de la communication (NTIC) et leur démocratisation. Ils ont amené une véritable explosion de l'information disponible aux entreprises. Les infrastructures en place permettent une circulation mondiale de l'information, via Internet en particulier, en même temps qu'elles permettent aux entreprises de communiquer avec leurs fournisseurs, leurs partenaires et leurs clients via des extranets.

Des espaces coopératifs dans lesquels les entreprises travaillent et réagissent ensemble ont émergé sous diverses formes : entreprise virtuelle, réseau d'entreprises, entreprise réseau, *etc.* La constitution et la gestion de ce type d'organisation s'appuient souvent des plateformes techniques et de partage d'information qui jouent un rôle de support et de facilitation de la coopération [Grefen et al., 2009].

Nous nous intéresserons dans le cadre de notre travail à ce type de relations, et plus particulièrement à la notion d'entreprise virtuelle ou encore à la coopération à la demande (*on demand inter-entreprises cooperation*).

Nous définissons la coopération à la demande comme un regroupement **temporaire** de partenaires distribués dans l'espace et dans le temps. Ce regroupement est formé à partir **d'alliances opportunistes** initiées par une entreprise appelée **entreprise initiatrice du projet de coopération** et qui se dissout une fois l'opportunité terminée. Un tel scénario implique la collaboration de différentes parties dans un processus coopératif composé de plusieurs processus exécutés par différents partenaires afin de répondre à un but commun ou saisir une opportunité sur le marché.

1.2 Contraintes pesant sur la coopération interentreprises

Reconnaissant l'intérêt de la coopération interentreprises à la demande dans la construction d'avantages concurrentiels, nous allons énumérer les contraintes qui entravent l'établissement de ce genre de coopération. Certes, les technologies de l'information innovent de manière croissante, mais le monde industriel reste malgré tout réticent pour adhérer à des scénarios de coopération dynamique.

Nous avons classé les contraintes en deux catégories majeures : celles qui sont relatives au système d'information, souvent considéré comme une brique de base de la coopération, et celles relatives à la nature dynamique de la coopération.

1.2.1 Contraintes associées au système d'information de l'entreprise

Souvent, l'interconnexion des processus métiers des entreprises dépend fortement de la capacité des systèmes d'information utilisés. On peut dire qu'il existe deux propriétés clés caractérisant un système d'information d'entreprise capable de participer à un scénario de coopération : la flexibilité et la possibilité d'inter opérer avec un système d'information extérieur. Néanmoins, avec leurs architectures actuelles, les systèmes d'information présentent un handicap pour les entreprises et freinent la naissance d'une coopération efficace, simple et à valeur ajoutée.

- **Manque de flexibilité**

La flexibilité constitue aujourd'hui une préoccupation majeure des entreprises qui cherchent plus d'agilité et de réactivité. La flexibilité désigne la capacité de l'entreprise à agir et l'aptitude à répondre aux changements d'une manière dynamique et efficace [McCoy and Plummer, 2006]. La question de flexibilité de l'entreprise impacte nécessairement le domaine des systèmes d'information. Le système d'information, longtemps considéré comme un outil d'intendance au service de l'entreprise, est aujourd'hui au cœur de son fonctionnement, et sa flexibilité en conditionne les performances. Fort de ce constat, la flexibilité de l'entreprise se projette directement sur son système d'information qui doit :

– S'aligner avec les nouvelles exigences métier,

– Envisager et supporter des scénarios d'évolution et des demandes de modification.

Les systèmes d'information doivent développer une capacité à agir et être extrêmement réactif afin d'aider à mettre sur le marché de nouvelles offres ou encore à répondre rapidement à des demandes d'adhésion à des scénarios de coopération. Cependant, avec leurs architectures actuelles, les systèmes d'information freinent les opportunités de coopération interentreprises. L'absence de solution architecturale efficace pour résoudre les problèmes auxquels est confronté le système d'information, a plongé ce dernier dans une situation de blocage vis-à-vis des exigences croissantes des métiers. L'enjeu consiste donc à rendre le système d'information le plus réactif possible aux évolutions du métier tout en préservant le patrimoine informationnel de l'entreprise.

- **Réticence à l'ouverture**

À l'origine, le système d'information a été conçu pour le fonctionnement interne d'une entreprise. Ni son architecture, ni ses missions n'envisageaient de prendre en considération des coopérations. En effet, le système d'information opère à l'interne de l'entreprise et l'interaction avec le monde extérieur se fait sous forme d'échanges d'information qui doivent être prévus à l'avance. Cependant, une coopération interentreprises à la demande exige une certaine ouverture du système d'information et une interopérabilité avec d'autres systèmes d'information hétérogènes.

L'hétérogénéité des systèmes d'information a motivé la recherche dans le domaine de l'interopérabilité. Malgré la multiplication des travaux sur l'interopérabilité du système d'information, dont on peut par exemple trouver des références dans [Izza, 2006], [Touzi, 2007], [Berre et al., 2007], les systèmes d'information des entreprises fonctionnent dans un environnement fermé où les frontières sont bien déterminées et l'interaction avec l'extérieur est bien planifiée.

1.2.2 Contraintes associées à la nature dynamique de la coopération

La question de coopérations à la demande est nécessairement impactée par la nature de l'écosystème économique qui tend vers une fluidification de sa structure, porteuse à la fois de nouvelles opportunités, mais également de nouvelles contraintes pour les entreprises. En effet, l'interconnexion des processus et la réalisation d'une coopération à la demande n'est pas une tâche facile à entreprendre. Suite à nos différentes études, nous avons classé l'ensemble des contraintes de la coopération en trois catégories : la confidentialité, l'autonomie et le dynamisme. Par conséquent, toute architecture d'interconnexion de processus doit respecter ces contraintes.

- **La confidentialité**

Un premier besoin lors d'une coopération est le respect de la confidentialité des entreprises participantes. Bien qu'il soit important que chaque entreprise puisse communiquer et coopérer avec les autres, ceci ne nie pas le fait qu'elles peuvent être concurrentes. En effet, c'est le dilemme de ce genre de projet qui consiste à mettre en partenariat sur une durée limitée des entreprises qui sont concurrentes le reste du temps et qui hésitent à partager des données et des activités.

De plus, la spécificité d'un processus interentreprises est que ses activités sont distribuées à travers les différentes entreprises participantes dans la coopération. Ces activités appartiennent à des processus internes qualifiés de processus privés et confidentiels. Ainsi, le respect de la confidentialité exige de les décrire avec un niveau d'abstraction suffisant tel que le savoir-faire des entreprises ne soit pas divulgué aux partenaires.

De ce fait, toute interconnexion des processus doit prendre en compte des aspects de confidentialité des processus, des données et des informations des différents partenaires.

- **L'autonomie**

Un autre besoin clairement identifié dans le cadre d'une coopération interentreprises à la demande est l'autonomie. Bien que les entreprises travaillent ensemble dans le cadre d'une coopération, elles conservent leur entière autonomie. Le souci principal est d'avoir un support pour la réalisation d'un projet commun, mais en aucun cas elles seront amenées à modifier leur propre manière d'opérer. Chaque entreprise participante souhaite garder ses méthodes et outils de travail habituels. Par conséquent, toute interconnexion des processus doit assurer la cohérence globale tout en préservant l'autonomie de chaque partenaire.

- **La coopération planifiée versus la coopération dynamique**

Lors d'une coopération et notamment dans le cadre d'une entreprise virtuelle dynamique, les entreprises peuvent avoir des compétences semblables. Cependant, elles ne résolvent pas toujours les problèmes rencontrés de la même manière. Chacune des entreprises possède ses propres manières d'agir qui les différencient des autres partenaires. C'est dans ce sens que lors de l'élaboration d'un processus coopératif on peut avoir le choix entre plusieurs partenaires. Ce choix est conditionné par un ensemble de critères stratégiques. De plus, on peut exprimer le besoin de changer l'ensemble des partenaires initiaux. Par conséquent, il s'avère difficile de décrire un scénario de coopération qui définit à l'avance l'ensemble des partenaires ainsi que leurs interactions possibles.

2. Évolutions nécessaires des systèmes d'information

Nos différents constats et analyses montrent que la coopération à la demande est un choix stratégique difficile à réaliser, étant donné que l'organisation actuelle de l'entreprise et de son système d'information ne sont pas encore matures pour ce type de fonctionnement. En effet, l'adhésion d'une entreprise à des scénarios de coopération est régie par un double problème.

Dans un premier temps, l'entreprise présente un manque de flexibilité au niveau du système d'information, support à son métier. En effet, les systèmes d'information actuels freinent les évolutions nécessaires pour répondre aux nouvelles orientations métier et sont perçus comme résistants aux changements. Dans un deuxième temps, établir une coopération à la demande d'une manière dynamique consiste à interconnecter un ensemble hétérogène de processus pour la réalisation d'un but commun. Cette interconnexion de processus doit être réalisée suivant une démarche bien déterminée tout en respectant les contraintes déjà présentées.

À l'issue des différents constats et analyses que nous avons menés jusqu'à ici, nous pouvons conclure que les entreprises doivent avoir une réflexion approfondie sur leurs systèmes d'information afin d'assurer une interconnexion efficace des processus d'entreprises. Chaque entreprise doit procéder à une remise à niveau de son système d'information de manière à le réorganiser et améliorer son efficacité. Un tel projet doit s'inscrire dans une démarche de coopération et renvoie aux deux questions de recherche suivantes :

1. Comment rendre le système d'information de l'entreprise flexible et réactif aux changements métier ?
2. Comment gérer l'hétérogénéité des systèmes d'information ?

Pour essayer de répondre aux deux questions posées précédemment, nous nous sommes intéressées à l'étude et à l'analyse des approches et des mécanismes existants dans la littérature.

2.1 Comment rendre le système d'information de l'entreprise flexible et réactif aux changements métier ?

Plusieurs auteurs et approches se sont attardés à résoudre la première question en proposant des démarches et des outils pour la conception du système d'information. Ces auteurs ont pris conscience de l'importance de la modélisation d'entreprise pour fournir des cadres de représentation adaptés à l'analyse des systèmes d'entreprise.

La conception du système d'information tout en s'inscrivant dans une problématique de réactivité et de flexibilité a atteint une maturité certaine et, on assiste aujourd'hui à la multiplication des travaux, sous des formes et des usages variés (BPM, ingénierie des processus d'entreprise, interopérabilité des entreprises et des systèmes d'information, architecture d'entreprise, *etc.*). Les travaux proposés ont mis le système d'information au centre de leurs préoccupations et le qualifient comme un support incontournable pour la mise en œuvre de la stratégie de l'entreprise.

Certains travaux comme [Indulska et al., 2009], [Henderson-Sellers et al., 2007], [Rolland, 2009], se sont attachés à comprendre et à exploiter les relations entre le système d'information à concevoir et les processus d'entreprise que ce dernier va faciliter, guider, et automatiser. Ces travaux trouvent leur origine dans le domaine de l'analyse et de la conception de systèmes d'information orientées « processus ». Dans ce courant de recherche, les auteurs partent du constat que la modélisation par les processus d'entreprise est fondamentale pour concevoir et mettre en place un système d'information qui assure la compétitivité et la réactivité de l'entreprise.

Plusieurs méthodes et outils pour l'ingénierie du système d'information ont vu le jour (approches par composants, *Model Driven Architecture*, *Model Driven Engineering*). Ces méthodes ont été proposées pour répondre à l'évolution du contexte du système d'information actuel : évolution en termes d'architecture du système d'information (hétérogène et distribué), évolution de son usage (ouvert et accessible pour différents utilisateurs).

Depuis quelques années, la notion d'**Architecture Orientée Services** (SOA : *Service Oriented Architecture*) s'est rapidement répandue et a été largement acceptée comme une architecture support du système d'information de l'entreprise.

À ses débuts, la SOA s'est imposée comme une approche pour l'architecture logicielle des systèmes. Suite à son succès grandissant, le concept de SOA s'est orienté vers des aspects très variés qui dépassent d'une certaine manière largement le domaine initial qu'est l'architecture logicielle. Aujourd'hui la SOA est considérée comme un style architectural pour le système d'information de l'entreprise grâce à son concept pivot : le service.

2.2 Comment gérer l'hétérogénéité des systèmes d'information ?

Plusieurs mécanismes et outils ont été conçus pour assurer l'ouverture des systèmes d'information et notamment l'interconnexion des processus de différentes entreprises. Les mécanismes d'envoi de messages sont parmi les outils les plus anciens qui ont permis d'établir des échanges de données entre processus interconnectés. Il s'agit de mettre d'accord l'émetteur et le récepteur sur un format de données et de messages. Cette approche implique une dépendance indéniable entre les processus interconnectés (*i.e.* le processus client doit se mettre à jour si le processus serveur modifie le format d'une de ses données).

La coordination transactionnelle entre les processus, dont le Workflow constitue l'instanciation la plus importante, permet de définir et d'exécuter les processus métiers. Bien que de nombreux systèmes de gestion de Workflow ont été développés et utilisés, ils s'attardent surtout sur l'automatisation des processus internes et ne prennent pas en compte des processus qui s'étendent au-delà des frontières de l'entreprise. Quelques travaux se sont

intéressés à faire coopérer des processus de plusieurs entreprises réparties, autonomes et hétérogènes.

S'il existe de nombreux systèmes et mécanismes de définition, de gestion et d'interconnexion des processus d'entreprises, ces derniers ne satisfont pas les besoins de la coopération. En effet, ils offrent des paradigmes de coopération qui se sont avérés mal adaptés à la coopération interentreprises à la demande [Grefen et al., 2006], [Bouzguenda, 2005], [Grefen et al., 2009].

Récemment, les **services Web** ont émergé pour proposer des solutions d'intégration des applications d'entreprises. Les services Web constituent l'instanciation la plus importante du modèle SOA dans le domaine industriel. En effet, les entreprises encapsulent les tâches automatiques comme des services logiciels pour les rendre visibles sur Internet. Les services Web sont accessibles à l'intérieur et à l'extérieur d'une entreprise. Par conséquent, ils permettent à l'entreprise d'intégrer ses applications hétérogènes ainsi que ceux des entreprises partenaires indépendamment des environnements techniques et des langages sur lesquels tournent ces applications. En d'autres termes, le concept de services Web offre aux entreprises la possibilité d'échanger des services qui peuvent être automatisés au sein de leurs processus. Cette spécificité présente un atout considérable, favorisant ainsi la mise en place des coopérations interentreprises à la demande.

Nous allons montrer dans la section suivante la pertinence de la SOA et notamment de son concept de base « le service » pour le système d'information de l'entreprise ainsi que pour la coopération interentreprises.

3. Pertinence de l'orientation service

En se basant sur les différentes constatations développées dans la section précédente, la piste qui nous paraît la plus pertinente pour mettre en œuvre des solutions que ce soit au niveau interne (ingénierie du système d'information de l'entreprise) ou encore en matière de coopération dynamique est celle basée sur l'orientation service.

Le concept de service commence à investir les milieux industriels depuis quelques années. Il est considéré comme la brique de base de l'architecture orientée services. Le principe d'une architecture orientée services consiste à structurer le système d'information d'une entreprise comme un ensemble de services qui exposent leur interface fonctionnelle et qui communiquent par messages.

Bien qu'il n'existe pas une définition consensuelle de ce qu'est un service, la définition donnée par [Papazoglou, 2003] est très largement référencée dans la littérature. D'après [Papazoglou, 2003], un service est défini comme « *un ensemble d'applications modulaires auto-contenues et auto-descriptives qui peuvent être publiées, localisées et invoquées depuis le Web. Un service peut effectuer des actions allant de simples requêtes à des processus métiers complexes. Les services permettent d'intégrer des systèmes d'information hétérogènes en utilisant des protocoles et des formats de données standardisés, autorisant ainsi un faible couplage et une grande souplesse vis-à-vis des choix technologiques effectués* ».

Comme nous l'avons déjà souligné dans la section précédente, le concept de service, ou plus précisément les services Web et l'architecture orientée services apportent des réponses

crédibles aux différentes contraintes et problèmes déjà présentés (cf. section 1.2). Plusieurs auteurs dans la littérature confirment cette vision [Vernadat, 2007], [Grefen et al., 2009],[Arsanjani, 2004].

Nous allons essayer dans la suite de cette section de souligner les avantages de l'orientation service concernant la flexibilité du système d'information et la coopération à la demande.

3.1 Avantages de l'orientation service pour le système d'information

L'approche service permet au système d'information d'acquérir une certaine flexibilité en lui offrant la possibilité de répondre aux besoins du marché et de participer à des nouvelles opportunités [Hirzalla et al., 2008]. En effet, l'architecture orientée services organise les fonctionnalités élémentaires contenues dans les applications d'entreprise en des services interopérables, standardisés qui peuvent êtres rapidement combinés et réutilisés pour répondre aux besoins métier. Cette vision permet par conséquent la construction de nouvelles applications ou encore de nouveaux processus métier informatisés en réutilisant des services déjà existants. Cette rapidité de construction des applications et des processus dote l'entreprise, et notamment son système d'information, d'une réactivité considérable, qui permet en retour d'accélérer le temps de réponse de l'entreprise vis-à-vis des changements de son environnement [Papazoglou, 2007].

En outre, l'approche service favorise l'alignement du système d'information aux divers besoins métier. Ce principe d'alignement consiste à mettre en cohérence la stratégie du système d'information avec la stratégie métier de l'entreprise [Phelizon and Rouhier, 2002]. Pour assurer ce principe, toute démarche orientée service doit partir des stratégies et des buts métier à atteindre pour identifier les services d'entreprise. Ainsi, les services identifiés seront en phase avec le monde métier, c'est-à-dire reliés à des buts et des stratégies définis par l'entreprise.

Dans la littérature relative au concept de service, plusieurs auteurs comme [Vernadat, 2007] [Cauvet and Guzelian, 2008] affirment la pertinence de l'approche service pour assurer une flexibilité considérable du système d'information de l'entreprise. Dans la suite, nous allons présenter les avantages de cette orientation pour la coopération à la demande.

3.2 Avantages de l'orientation service pour la coopération

L'émergence de l'approche service apporte une certaine simplification et facilite l'établissement de processus coopératifs reconfigurables. La coopération selon ce type d'approche est réalisée grâce au paradigme de composition des services [Grefen et al., 2006], [Grefen et al., 2009]. La composition des services a été définie dans [Casati et al., 2000a] comme étant la capacité d'offrir des services à valeur ajoutée en combinant des services existants probablement offerts par différentes entreprises.

Le paradigme de coopération par composition des services, permet aux entreprises de définir leurs services avec un certain niveau d'abstraction en présentant une interface vers les processus/sous processus qu'elles proposent. Ainsi, la définition des services répond aux besoins d'autonomie et de confidentialité des entreprises.

En somme, dans un premier temps, l'approche service permet une meilleure flexibilité du système d'information de l'entreprise. Les services sont considérés parmi les candidats capables de concrétiser le rêve d'une entreprise de réduire les délais de mise sur le marché des nouveaux produits, via la réutilisation de l'existant et l'accélération des cycles de développement. Dans un deuxième temps, l'approche service favorise la mise en place des coopérations à la demande basées sur le paradigme de composition des services.

4. Problématiques de recherche

L'importance de l'orientation service et ses avantages à l'égard du système d'information de l'entreprise et de la coopération, nous confrontent à un nouveau défi. Il s'agit, essentiellement, d'assurer une ingénierie du système d'information de l'entreprise en adoptant l'architecture orientée services comme une architecture support. Cette ingénierie doit assurer d'une part la flexibilité du système d'information et d'autre part, favoriser sa participation à des scénarios de coopérations à la demande.

Dans l'état actuel des choses, plusieurs obstacles contraignent la mise en place de la SOA au sein de l'entreprise ainsi que la formation des coopérations à base de services. Il s'agit fondamentalement du manque de méthode à mettre en œuvre pour définir l'architecture de services au sein du système d'information de l'entreprise. De plus, il existe peu de travaux sur les méthodes de construction du processus coopératif à la demande, basées sur le paradigme de composition des services.

À ces deux problèmes s'ajoute un troisième lié à la nature dynamique de la coopération. La participation à des scénarios de coopération à la demande exige des entreprises d'avoir un certain niveau de flexibilité des services qu'ils proposent sur le marché. Cette flexibilité se manifeste par l'adaptation de leurs services. En effet, il est important d'avoir des services qui adaptent leur comportement pour correspondre à l'opportunité détectée sur le marché, appelée encore « contexte d'utilisation ».

L'analyse de la problématique de coopération interentreprises à base de services nous a conduit à la décomposer en trois sous problématiques qui forment notre problématique de recherche et qui sont résumées dans la Figure 1.

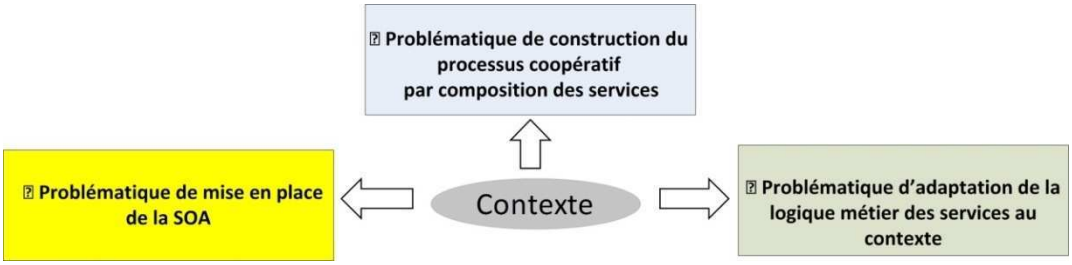


Figure 1. Sous problématiques traitées

Dans un souci de flexibilité et pour assurer une cohérence globale, nous considérons le contexte dans toutes les sous problématiques liées à la coopération à base de composition des services. Selon [Dey et al., 2001] « *le contexte est n'importe quelle information qui peut être*

utilisée pour caractériser la situation d'une entité. Une entité est une personne, un endroit ou un objet qui est considérée pertinente dans l'interaction entre un utilisateur et une application, incluant l'utilisateur et les applications elles-mêmes ».

Ces problématiques s'inscrivent dans la continuité des axes de recherches initiés déjà dans le cadre des travaux de thèse de [Izza, 2006] qui s'intéresse à la problématique d'intégration des services d'entreprises.

Notons également que les problématiques, que nous avons identifiées, sont mutuellement interdépendantes dans le sens où les notions issues d'une problématique peuvent être réutilisées dans une autre problématique. Les trois premières problématiques sont complémentaires tandis que la composante de contexte leur est commune et transversale. Le détail de ces problématiques est donné ci-après.

4.1 Problématique de mise en place de la SOA

Les architectures orientées services souffrent d'un certain nombre de limites et de lacunes surtout dans un contexte de coopération à la demande. Il s'agit fondamentalement du manque de méthodologies permettant de les construire de façon efficace au sein de l'entreprise. Certes, quelques approches de modélisation et d'identification de services ont vu le jour, mais la majorité de ces approches ne sont pas flexibles du fait qu'elles sont fortement liées aux technologies et aux outils utilisés. De plus, les démarches de développement d'applications orientées services ne proposent que des lignes directrices afin de guider les architectes vers un modèle orienté services, mais elles ne délivrent pas de méthodologie complète et précise pour atteindre leur objectif de façon efficace. Ainsi, une démarche méthodologique adaptée aux entreprises et à la coopération qui pourrait naître entre ces entreprises s'avère d'une importance capitale.

Conséquence du manque de méthodologie de construction de l'architecture orientée services, une difficulté majeure qui est généralement rencontrée est la définition de la notion de service. Qu'est ce qu'un service pour l'entreprise ? Quel niveau de granularité faut-il retenir pour définir correctement les services dédiés à la coopération interentreprises ?

En outre, la démarche doit dépasser une des limites principales du concept de service à savoir le manque de dynamisme. En effet, pour un service donné, l'entreprise qui le fournit doit décider, durant la phase de modélisation, du contrat d'utilisation de ce service. Par conséquent, la définition de service est figée et manque de flexibilité. Elle ne s'adapte pas en fonction de son contexte d'utilisation. Ce constat nous pousse à s'interroger sur la manière d'intégrer l'adaptation des services et la gestion de leurs variantes d'usage dès la phase de leur modélisation.

Ainsi, notre première sous problématique consiste à définir une méthodologie de migration du système d'information d'entreprise existant vers un système d'information basé sur le concept de service adaptable.

4.2 Problématique de construction du processus coopératif par composition des services

Dans la littérature dédiée à la coopération, la plupart des systèmes et travaux existants comme DERPA, CrossFlow et PointSynchro traitent la coopération statique et ne supportent pas la recherche de partenaires ainsi que la réalisation des coopérations de manière dynamique [Casati and Discenza, 2000], [Angelov and Grefen, 2002], [Bitcheva, 2003]. C'est la raison pour laquelle on considère que ce domaine, insuffisamment exploré, pose des problèmes spécifiques et mérite une exploration plus approfondie. Ainsi, réussir la coopération à la demande consiste à résoudre un certain nombre de problèmes se référant à la discipline de la composition des services Web et qui sont en phase avec notre orientation de recherche.

Le cycle de vie d'une application construite à base de services Web [Benatallah et al., 2003b] intègre quatre étapes fondamentales : la description, la découverte, la publication et la composition des services, qu'il s'agit ici d'inscrire dans une vision de coopération interentreprises.

– Description des services

Afin que le service puisse être visible aux partenaires, chaque entreprise doit définir l'ensemble des services qu'elle propose dans le cadre des activités qu'elle mène. Cependant, une utilisation des standards actuels comme le standard WSDL, pour la technologie service Web, propose une description fonctionnelle avec une sémantique réduite.

Afin de combler ce manque en termes de représentation de services, des propositions issues du Web sémantique ont vu le jour (telles que OWL-S [Martin *et al.*, 2004] et SAWSDL [Farrell *et al.*, 2007]), mais aucune ne propose une représentation de service assez large pour mettre en œuvre des adaptations au contexte.

– Publication des services

La publication des services repose sur l'utilisation d'un référentiel de services permettant d'exposer les services disponibles aux autres entreprises. Le registre UDDI (i.e. *Universal Description Discovery Integration*) est le registre standard pour la publication des services. De nombreux travaux proposent des registres publics sur le Web (tels que les sites XMethods[1] ou RemoteMethods[2]), mais peu offrent une représentation détaillée des services (par exemple, la publication des informations de contexte d'un service).

– Découverte des services

Quand une entreprise initiatrice du projet de coopération détecte une opportunité sur le marché, elle se fixe un objectif afin de répondre à l'opportunité en question. La réalisation de

[1] XMethods disponible sur l'adresse suivante : http://www.xmethods.net/ve2/index.po
[2] RemoteMethods, dispoinble sur l'adresse suivante : http://www.remotemethods.com/

cet objectif nécessitera la découverte d'un ou de plusieurs services qui répondent aux besoins fonctionnels et contextuels (*i.e.* préférence, localisation, temps, *etc.*) de l'entreprise.

Dans la littérature plusieurs travaux ont été proposés dans le domaine de découverte des services Web [Kritikos and Plexousakis, 2006], [Maximilien and Singh, 2004b]. La majorité de ces travaux s'intéressent particulièrement à la comparaison des descriptions textuelles entre la requête du client et la signature des services (entrées et sorties). Peu nombreux sont les travaux qui se sont focalisés sur la découverte[3] contextuelle des services. Cette dernière trouve ses origines dans les applications mobiles où l'objectif consiste à considérer la connaissance relative aux préférences de l'utilisateur et aux caractéristiques de son dispositif afin de chercher les services qui correspondent au mieux à ses besoins [Broens et al., 2004], [Dey et al., 2001].

– Composition des services

La formation de la coopération à la demande se base sur le paradigme de composition des services. Cependant, compte tenu de l'évolution de l'environnement de l'entreprise (fournisseurs et clients) il est difficile de décrire un schéma de composition qui définit à l'avance toutes les possibilités d'interactions entre les services d'entreprises. Par conséquent, on a besoin d'une composition dynamique selon les cas et les circonstances au lieu d'une composition planifiée et préétablie dans des schémas.

La composition dynamique a suscité l'intérêt de plusieurs chercheurs appartenant à différentes disciplines. À cet égard, plusieurs travaux ont été présentés [Fujii and Suda, 2006], [Thakkar et al., 2002]. Malgré ces efforts, une approche méthodologique pour découvrir et élucider les besoins des entreprises partenaires, pour identifier, composer et orchestrer les services fait défaut.

Ainsi, notre deuxième sous problématique est la construction du processus coopératif à la demande. Il s'agit essentiellement de proposer un cadre général pour la mise en œuvre de la coopération en utilisant le paradigme de composition des services.

4.3 Problématique d'adaptation de la logique métier des services au contexte

Certes, l'approche service promet une facilitation de la mise en œuvre des coopérations à la demande, mais cette approche manque de dynamisme [Chang and Kim, 2007] [Rolland and Kaabi, 2007]. Malgré la définition explicite des variantes d'usage au moment de la modélisation des services, plusieurs problèmes restent à résoudre pour assurer une adaptation efficace des services au contexte.

[3] Tout au long de notre travail de recherche, nous utilisons le concept de « découverte » pour désigner à la fois la recherche et la sélection des services.

L'adaptation au contexte dans le domaine des systèmes d'information pervasifs, s'intéresse à l'adaptation au type de terminal et au type de l'utilisateur connecté pour garantir une utilisation confortable des applications dans des nouveaux environnements. À l'image de ces travaux, l'adaptation dynamique des services consiste à modifier le comportement des services pour répondre à de nouvelles situations et à de nouveaux besoins. La réalisation de cette adaptation se base essentiellement sur un ensemble de paramètres appelés paramètres de contexte. Dans la majorité des cas, ces paramètres n'ont pas été tous pris en compte lors de développement des services. Ceci conduit généralement les informaticiens à reprendre le cycle de vie du service dès son début afin de prendre en compte ces nouveaux paramètres. De plus, les travaux d'adaptation existants se focalisent sur la création incrémentale ou sur le prototypage de systèmes (ou de services) adaptables au contexte en incorporant le code d'adaptation dans le code métier du système.

[Ketfi et al., 2002] identifient les quatre raisons principales qui guident l'adaptation des applications : l'adaptation correctionnelle, l'adaptation adaptative, l'adaptation évolutive et l'adaptation perfective. L'adaptation correctionnelle consiste à examiner l'application en cours d'exécution et à l'adapter car elle ne se comporte pas correctement ou comme le prévoyait son cahier des charges. L'adaptation adaptative considère que même si l'application s'exécute correctement, parfois son environnement d'exécution peut changer, ce qui nécessite son adaptation. Quant à l'adaptation évolutive, elle s'intéresse à l'ajout de certaines fonctionnalités qui n'ont pas été prises en compte au moment du développement de l'application. La dernière raison est l'adaptation perfective dont l'objectif est d'améliorer les performances de l'application.

En étudiant les travaux existants dans la littérature relative à l'adaptation des services Web au contexte, nous avons remarqué qu'il existe peu de travaux qui s'intéressent à l'adaptation évolutive. Nous considérons l'adaptation évolutive comme une adaptation qui vise à ajouter des fonctionnalités et des comportements au service pour répondre à son contexte d'utilisation. En revanche, les autres raisons d'adaptation ont suscité l'intérêt de plusieurs chercheurs appartenant à diverses communautés [Keidl and Kemper, 2004], [Suraci et al., 2007].

L'adaptation évolutive (appelée encore adaptation de la logique métier) des services au contexte demeure essentielle afin de mieux les exploiter dans des scénarios de coopération à la demande.

Fort de ces constats, notre troisième sous problématique est la problématique d'adaptation de la logique métier des services au contexte.

5. Principes méthodologiques

Trois principes méthodologiques caractérisent nos contributions :

- **Une perspective d'ingénierie**

Nous nous inscrivons tout au long de notre travail de recherche dans une perspective d'ingénierie. Notre proposition prend en compte plusieurs dimensions (information,

organisation et processus) tant au niveau micro (à l'interne de l'entreprise) qu'au niveau macro (à l'externe de l'entreprise).

De plus, nos contributions suivent une démarche méthodologique pour guider à la fois la conception et le développement accompagnée par un ensemble de modèles et méta-modèles avec plusieurs points de vues. Cette initiative s'inscrit dans le cadre d'une approche MDE (*Model Driven Engineering*).

Dans un premier temps, l'utilisation des méta-modèles permet de comprendre et manipuler les modèles existants, ainsi que les nouveaux modèles susceptibles d'être intégrés. Dans un deuxième temps, elle permet d'introduire un niveau d'abstraction suffisant pour proposer une sémantique commune afin d'avoir des vues cohérentes sur les concepts manipulés.

- **Prise en compte de la flexibilité**

La flexibilité est une caractéristique importante et largement recherchée par les entreprises. Dans le contexte des systèmes d'information, [Chelli, 2003] définit la notion de flexibilité comme étant la déclinaison informatique de la notion d'agilité des entreprises. Dans notre travail, la notion de flexibilité est assurée à travers la prise en compte du concept de « contexte » et ceci à plusieurs niveaux : dès la phase de mise en place de l'architecture orientée services au sein de l'entreprise jusqu'à la phase d'adaptation de la logique métier au moment de l'exécution du service.

Nous considérons que la prise en compte du contexte lors de la modélisation, la description, la publication et l'exécution des services apporte la flexibilité nécessaire à la démarche.

- **Le principe d'ouverture**

Plus qu'un principe, l'ouverture est avant tout une contrainte qui impose l'utilisation des standards industriels et des outils déjà développés. Ceci justifie les multiples choix effectués, et en particulier les choix opérés en matière de description syntaxique des services (WSDL), de description sémantique des services (OWL-S), de composition des services (BPEL) et de représentation des ontologies de contexte (OWL), *etc.*

D'autres standards ont été retenus notamment pour la modélisation (UML), pour la démarche méthodologique ainsi que pour le développement du prototype (Java).

6. Objectifs et contributions

L'objectif principal de la thèse est de développer une nouvelle approche qui assure l'efficacité et l'efficience de la coopération interentreprises. Cette coopération passe obligatoirement par une interconnexion des processus des différentes entreprises partenaires et se base sur l'usage des systèmes d'information dans un contexte d'entreprise virtuelle dynamique.

Nos contributions répondent aux limites et aux problématiques précédemment décrites en proposant (voir Figure 2) :

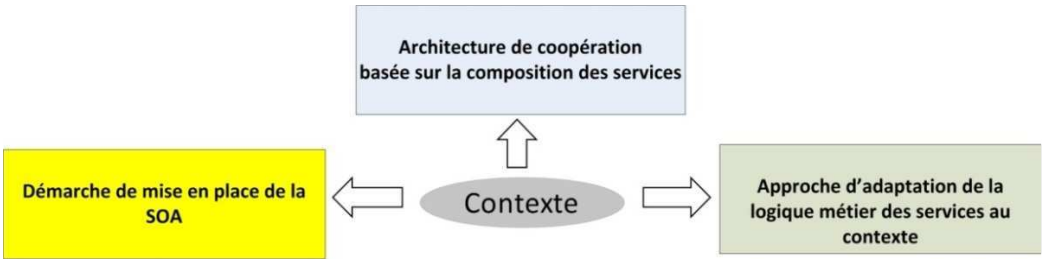


Figure 2. Contributions de la thèse

- Une démarche méthodologique dénommée CSOMA (Contextual Service Oriented Modeling and Analysis) dont l'objectif est double : assurer d'une part la flexibilité à l'interne de l'entreprise grâce à l'orientation service et favoriser d'autre part les coopérations à la demande. CSOMA est une **méthode** de construction de l'architecture de services au sein de l'entreprise qui part d'une architecture d'entreprise existante « *As-Is* » pour retrouver l'architecture cible « *To-Be* ». Elle propose une double démarche de construction de l'architecture de services : construction de la **SOA métier** et construction de la **SOA informatique** (SOA IT).

Une particularité de CSOMA repose sur le fait qu'elle tient compte de l'adaptation des services au moment de leur conception. Ainsi, CSOMA propose le concept de « **service domaine »**, qui est un service adaptable dédié à la coopération. Un service adaptable est un service qui évolue pour prendre en compte les changements qui interviennent dans son environnement. Il s'agit essentiellement des services sensibles au contexte de leur utilisation, c'est-à-dire qui changent et se présentent selon le cas d'utilisation présent dans le marché. Cette contribution est rattachée au domaine d'ingénierie du système d'information de l'entreprise.

- Une architecture de coopération basée sur le paradigme de composition des services. L'architecture proposée est accompagnée par un ensemble de **méta-modèles** détaillant les concepts manipulés. Ces concepts décrivent clairement l'organisation de la coopération à la demande, ses entités principales (les communautés, le registre de coopération, *etc.*) et les *constructs* du processus coopératif (notion de *Goal Template*, branchement, *etc.*). En outre, nous proposons une **démarche de construction du processus coopératif** qui englobe trois étapes fondamentales : la description, la publication et la découverte des services domaines. Chacune des étapes constitue une proposition.

Concernant la description des services, nous proposons un **modèle de description** qui permet (i) la prise en compte de la sémantique à travers la notion d'ontologie et (ii) l'ensemble des paramètres contextuels définis grâce à une ontologie de catégorisation du contexte.

Nous proposons également un **modèle de contexte multi-niveaux** basé sur l'ontologie OWL afin de modéliser les informations de contexte.

Quant à la publication, nous proposons **des mécanismes** (et notamment des APIs) qui se chargent d'assurer **la publication syntaxique**, **sémantique** ainsi que **des paramètres de contexte des services**.

En ce qui concerne la découverte des services, nous proposons un **processus de découverte** qui exploite les descriptions syntaxiques, sémantiques et contextuelles afin de retrouver les services qui correspondent aux exigences exprimées par le client.
Cette contribution est au carrefour de plusieurs thématiques de recherche comme le domaine des processus coopératifs, le contexte et aussi les services Web.

- Une architecture d'adaptation dynamique du service domaine à de nouveaux contextes d'utilisation. Il s'agit d'une adaptation évolutive qui vise à doter le service domaine de nouvelles fonctionnalités prenant en compte son contexte d'utilisation. Cette architecture exploite le potentiel offert par la programmation orientée aspect. Ce paradigme de programmation impliquant une séparation des préoccupations (fonctionnel/non fonctionnel), joue un rôle intéressant pour étendre et adapter le comportement d'un service. À cet égard, nous proposons le concept **d'aspect dédié à l'adaptation** (*i.e.* propose des actions d'adaptation). Ce concept enrichit le concept d'*aspect* classique à travers une extension du domaine des *join points*[4] à l'ensemble des événements reliés au contexte.

L'architecture d'adaptation est supportée par un ensemble de **modules** qui se chargent de capter les changements de contexte et d'adapter la logique métier du service afin qu'il puisse être utilisé de façon efficace dans l'opportunité de coopération. L'adaptation de la logique métier du service domaine se réalise par le biais de **tissage d'aspects**.

Cette contribution est rattachée au domaine de recherche sur l'adaptation des services au contexte.

7. Organisation du document

Ce mémoire est organisé en quatre chapitres, complétés par des annexes comportant les implémentations et les études de faisabilité réalisées :

- Le « **chapitre 1**-État de l'art » recouvre les différents aspects et axes de recherche qui s'apparentent à notre problématique. Ce chapitre est structuré en trois sections. La première, trace l'espace dans lequel évolue notre travail de recherche (la coopération à la demande) et examine les différentes vues de cette coopération (vue métier et vue technique). La deuxième section s'intéresse à l'analyse du concept de « service » au profit de la coopération interentreprises et explore les différentes approches et technologies qui lui sont relatives. Quant à la dernière section, elle s'intéresse au concept de « contexte » et de l'adaptation des services au contexte et passe en revue les différents travaux de l'adaptation des services Web au contexte.

  Les trois chapitres suivant constituent les contributions conceptuelles de la thèse :

[4] Les *join points* représentent des points définis dans l'exécution d'un programme, tels l'appel d'une méthode, la réalisation d'une exception,

- Le « **chapitre 2**- Construction de l'architecture de services au sein de l'entreprise » répond à la première problématique de mise en place de la SOA à travers la proposition de la démarche méthodologique CSOMA (**C**ontextual **S**ervice **O**riented **M**odeling and **A**nalysis).
- Le « **chapitre 3**- Construction du processus coopératif à la demande » répond à la deuxième problématique de construction du processus coopératif par composition des services à travers la proposition d'une architecture de coopération ainsi que d'une démarche de construction du processus interentreprises. Nous nous intéressons plus précisément dans la démarche de construction du processus coopératif à intégrer les paramètres de contexte lors de la description, publication et découverte des services des partenaires.
- Le « **chapitre 4**- Adaptation des services domaines au contexte à base de tissage d'aspects » répond à la troisième problématique d'adaptation de la logique métier des services au contexte à travers la proposition d'une approche d'adaptation des services à base de tissage d'aspects.

La conclusion et les perspectives clôturent ce manuscrit de thèse en présentant un bilan du travail effectué et un ensemble de perspectives liées notamment à la poursuite de ce travail ainsi qu'aux nouveaux thèmes de recherche qui nous paraissent les plus pertinents.
Enfin, une annexe est associée à chacun des chapitres de contribution conceptuelle (chapitre 2, chapitre 3 et chapitre 4). Ces annexes décrivent les implémentations techniques et les expérimentations réalisées pour valider les propositions de cette thèse. Elles ont pour but de démontrer la faisabilité technique des différentes contributions et d'illustrer leur intérêt et leur pertinence.
La figure suivante résume les différents chapitres présentés ainsi que leurs articulations.

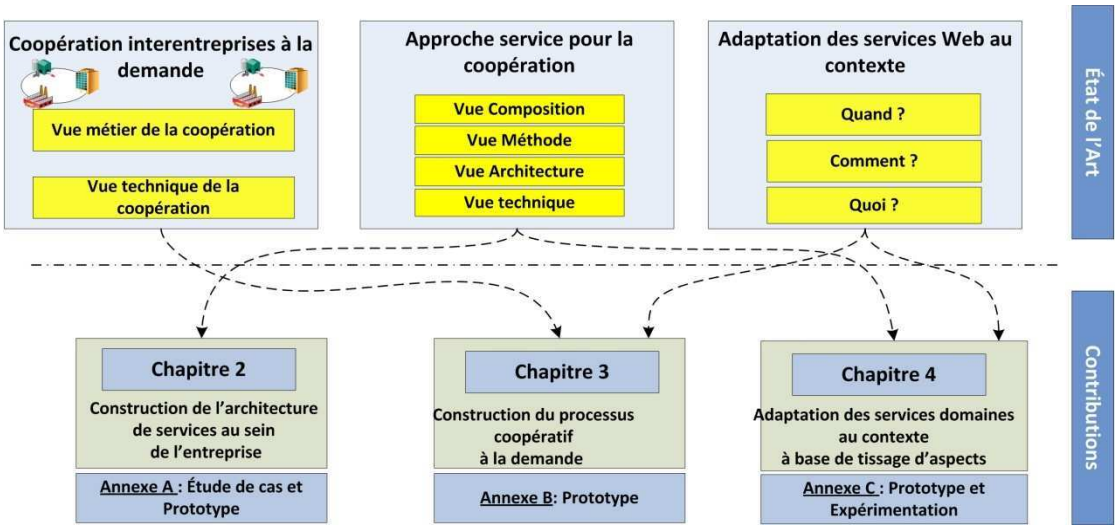


Figure 3. Organisation du manuscrit de thèse

CHAPITRE 1

État de l'art

Table des matières

La compréhension de la coopération interentreprises (entre les entreprises) à la demande et le développement d'approches qui facilitent cette forme de coopération est un domaine qui a attiré l'attention des milieux académiques et industriels. Ceci a donné naissance particulièrement à la notion d'entreprise virtuelle, de clusters d'entreprises ou de communautés virtuelles d'entreprises. La gestion efficace de ce type d'organisation a été accompagnée par l'introduction de plusieurs approches aussi bien au niveau métier qu'au niveau technique (technologique). Dans le but de mieux comprendre l'évolution de cette

pensée, nous présentons dans ce chapitre une exploration et une analyse détaillée de l'état de l'art autour de la problématique de coopération. Dans **la section 1**, nous nous attardons sur les formes émergentes de la coopération. Par la suite, nous discuterons les concepts métier et techniques relatifs à la coopération à la demande. Une vue globale des solutions, formalismes, et approches qui facilitent l'établissement de la coopération montre que l'approche orientée services semble être une solution prometteuse pour la coopération interentreprises. De ce fait, un état de l'art est proposé, dans **la section 2**, selon un cadre de référence qui permet de ressortir les différents aspects du concept de service et de positionner les approches les unes par rapport aux autres. Outre les considérations au niveau architectural de cette approche qui sont illustrées à travers plusieurs exemples, les considérations techniques et les méthodes de mise en place ainsi que la composition des services sont identifiées.

La notion clef de flexibilité rend critique le développement de services adaptables pour mettre en oeuvre la coopération interentreprises à la demande. C'est la raison pour laquelle, que **la section 3** de ce chapitre propose une revue de la littérature concernant la notion de « contexte » et d'adaptation des services Web au contexte.

1.1 Coopération interentreprises à la demande

Depuis le milieu des années 70, la coopération interentreprises est devenue une préoccupation émergente dans la littérature, et ce dans plusieurs domaines (économie, gestion, informatique, *etc.*). Les phénomènes de regroupement d'entreprises et des accords interentreprises donnent naissance à un vaste champ d'études et d'analyse des relations coopératives. Nous tenterons dans cette section d'éclaircir les notions de base de la coopération à la demande. Nous proposons une revue de la littérature et une catégorisation des formes les plus usitées afin d'éclairer la différence entre coopération statique et coopération à la demande. En plus, nous nous intéressons également aux concepts, standards et approches qui ont facilité la coopération interentreprises aux niveaux métier et technique (voir Figure 1.1).

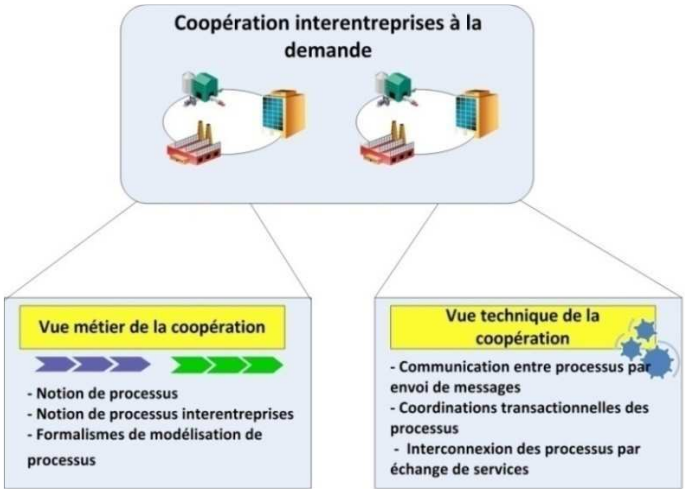


Figure 1.1. Cadre de référence pour l'analyse de la coopération interentreprises à la demande

1.1.1 Coopération interentreprises à la demande : vers une relation dynamique

La coopération interentreprises est une nouvelle forme d'organisation qui a émergé ces dernières années. Nous allons nous appuyer sur certains constats, proposés par [Touzi, 2007], afin de comprendre l'essence de ce type de coopération influencé par la nature de l'écosystème économique : environnement industriel qui tend vers une fluidité des relations et abandonne les relations figées et préétablies dans des scénarios. [Touzi, 2007] illustre l'évolution vers les coopérations à la demande par un adage simpliste : « **plus loin, plus vite, plus fort** ». En effet, cette tendance évolutive des relations de coopération se caractérise selon les trois vues suivantes :

- **Rayonnement (plus loin) :** les développements technologiques (Internet, moyen de communication, *etc.*) ont offert la possibilité d'échanger des informations entre les clients, les partenaires ou fournisseurs qui peuvent être éloignés géographiquement. En effet, les partenaires bénéficient du faible coût et de la vitesse des communications via Internet pour minimiser les déplacements, réduire les coûts et raccourcir les délais de réalisation des projets.
- **Réactivité (plus vite) :** bien qu'il soit fréquent de construire des relations de coopération durable, il est également intéressant de construire ponctuellement des relations opportunistes. Ce changement s'accompagne également d'une variation dans la prise en compte des horizons stratégiques : d'une dynamique à horizons plutôt lointains, caractéristique principale d'un milieu stable, vers des horizons plus rapprochés, significatifs d'une relative instabilité.
- **Intensité (plus fort) :** les avancées, aussi bien dans le domaine du génie industriel que dans le domaine du génie informatique, amènent nécessairement les coopérations à être plus efficaces dans leur définition et dans leur déroulement.

Les différentes réflexions et constats déjà présentés nous ont principalement permis de mettre en évidence l'importance grandissante et le rôle changeant de la notion de coopération vers des structures dynamiques. Cependant, il nous semble important de définir d'une manière précise ce concept en plein essor. Nous proposerons dans la section suivante des essais de définition de ce concept.

1.1.2 Coopération interentreprises à la demande : essais de définition

En passant en revue les travaux sur l'entreprise virtuelle, ou encore sur les coopérations à la demande, on constate qu'il n'existe pas à ce jour de définition reconnue par la communauté scientifique. L'absence de définition de ce qu'est l'entreprise virtuelle (appelée encore coopération à la demande) laisse planer un flou sur ce concept. Pour en éclaircir la compréhension, il nous semble primordial avant tout, de présenter les différentes propositions de définitions qui ont été élaborées dans la littérature.

La coopération interentreprises à la demande, comme son nom l'indique d'ailleurs, constitue une forme particulière de la coopération qui fait recours aux nouvelles possibilités des technologies de l'information et de la communication. Avant de définir le concept de la

coopération à la demande, commençons tout d'abord par définir le concept général à savoir la coopération interentreprises.

L'origine étymologique du terme coopérer est l'association de la racine « operare » et du suffixe « co » qui signifie « travailler ensemble ». Nous retrouvons cette notion de travail conjoint dans le dictionnaire Larousse où coopérer est défini comme « agir conjointement ». L'encyclopédie Universalis considère la coopération comme « le fait, pour une personne, de s'adonner consciemment à une activité complémentaire de celles d'autres personnes dans le cadre d'une finalité commune, dans un groupe donné ». Une autre définition semblable est celle proposée par [Délfard, 2000] qui considère la coopération comme « des liens que construisent entre eux des agents en vue de réaliser, volontairement, une œuvre commune ».

Selon [Luik, 1996], les coopérations à la demande héritent des coopérations interentreprises classiques et se distinguent essentiellement par quatre caractéristiques fondamentales à savoir :
- Les coopérations à la demande ou entreprises virtuelles sont dynamiques et elles peuvent réagir rapidement face à l'évolution croissante du marché,
- Les coopérations à la demande possèdent un caractère éphémère, bien que certaines aient duré jusqu'à huit ou neuf ans, bon nombre n'existent que pendant quelques mois,
- Les entreprises à la demande ont une hiérarchie vaguement définie. En effet, dans les coopérations à la demande, c'est la nature des relations, plutôt que les liens hiérarchiques officiels qui détermine le succès de la coopération,
- Les compétences de base dans une coopération à la demande sont reparties entre tout le groupe ou le consortium de groupes.

Nous retenons la définition, que nous avons déjà présenté (cf. introduction générale, section 1.1), pour désigner une coopération à la demande comme un regroupement **temporaire** de partenaires distribués dans l'espace et dans le temps. Ce regroupement est formé à partir **d'alliances opportunistes** initiées par une entreprise appelée **entreprise initiatrice du projet de coopération**. Cette alliance opportuniste peut se dissoudre une fois l'opportunité terminée.

1.1.3 Formes de coopération à la demande

Entreprise virtuelle, coopération à la volée, coopération à la demande, « *virtual breeding environnement* », *etc*. sont toutes des expressions que l'on ne cesse de retrouver dans le vocabulaire de l'entreprise moderne traduisant la survenue de nouvelles formes d'entreprises basées sur des relations de partenariat. Parmi ces appellations, nous distinguons trois concepts qui constituent les modèles les plus adoptés : « l'entreprise virtuelle », « *virtual breeding environnement* » et « la communauté virtuelle d'entreprises ». Ces trois formes de coopération à la demande sont étroitement liées et nombreux sont ceux qui tendent à les confondre. Nous proposons d'éclaircir ces concepts qui renvoient tous à la notion clef de coopération à la demande.

1.1.3.1 Entreprise virtuelle

Le concept d'entreprise virtuelle a été introduit initialement par Jan Hopland, cadre supérieur de DEC (*Digital Equipement Corporation*) vers 1987, puis diffusé en 1993 par [Goldman and

Nagel, 1993]. Selon Hopland, la notion de « virtuelle » fait référence au concept de la mémoire virtuelle d'un ordinateur. En effet, cette mémoire est conçue dans le but de permettre à des programmes de s'exécuter dans un environnement matériel possédant moins de mémoire centrale que nécessaire ou encore faire tourner plus de programmes que la mémoire centrale ne peut en contenir. En s'inspirant de cette métaphore, Hopland explique qu'une entreprise virtuelle est une entreprise qui peut maîtriser des situations données issues du changement du marché mieux que celles provenant de la coopération classique entre entreprises.

Ce concept se retrouve dans la littérature à travers les travaux de différents auteurs. Nous avons essayé de recenser dans le Tableau 1.1, dans un ordre chronologique, les définitions que nous considérons les plus pertinentes et représentatives. Ce tableau ne vise pas une exposition exhaustive de l'ensemble des définitions liées au concept d'entreprise virtuelle.

Auteur(s)	Définition proposée
[Upton and Affee, 1996]	« une communauté de douzaines voire de centaines d'entreprises, chacune concentrée sur ce qu'elle sait faire le mieux, toutes reliées par un réseau électronique qui leur permet d'opérer de façon flexible et non onéreuse, sans se soucier de leur emplacement respectif »
[Favier and Coat, 1997]	« un concept, mais pas un lieu. C'est une activité mais pas un bâtiment (….). La réalité physique et globale de l'entreprise disparaît au profit de coopérations entre personnes pour réaliser un produit, ou proposer un service, sur la base d'objectifs communs. Le cœur de ce type d'entreprise repose sur les équipes virtuelles »
[Gebauer, 1996]	« (…) au moins deux organisations indépendantes ou unités organisationnelles, formant une relation coopérative afin d'atteindre un but commun. »
[Travica, 1997]	« une entreprise virtuelle se réfère à une collection permanente ou temporaire d'un groupe d'individus géographiquement distants qui n'appartiennent pas à une même entreprise et dépendent des liaisons électroniques afin d'achever le processus de production »
[Sieber and Griese, 1998]	« une entreprise virtuelle est toute forme institutionnalisée qui a la capacité de fournir des produits et des services indépendamment du temps et du lieu de ses concurrents »
[Burn, 1998]	« les entreprises virtuelles sont des entreprises en réseau électronique dépassant les frontières organisationnelles »

Tableau 1.1. Définitions de l'entreprise virtuelle

En analysant les différentes définitions présentées dans le Tableau 1.1, nous pouvons constater que l'objectif principal d'une entreprise virtuelle est de lier des entreprises distinctes pour travailler ensemble d'une façon réactive et en collaboration. Cette caractérisation pourrait placer l'entreprise virtuelle dans le cas des autres structures (entreprise étendue, entreprise réseau ou réseau d'entreprises), mais sa particularité est que les structures des entreprises virtuelles sont fondées sur un système d'information [Martinez et al., 2001]. Chaque entité impliquée dans l'entreprise virtuelle devient un nœud d'une architecture informatique permettant, pour chaque projet, que tout se passe comme si tout était réalisé en un seul site grâce à l'utilisation des technologies de l'information.

L'entreprise virtuelle met en œuvre des processus de travail sans que le lieu et le temps n'exercent un caractère discriminant. En effet, les partenaires se retrouvent rarement en face-à-face [Hofstede et al., 1997], [Favier and Coat, 1997]. Ils travaillent ensemble à distance essentiellement via les nouvelles technologies de l'information et de la communication. Certains auteurs comme [Davidow and Malone, 1992], [Favier and F, 1999] considèrent que les activités réalisées au sein d'une entreprise virtuelle présentent un caractère ubiquitaire.

Les débats et la recherche sont loin d'être terminés en ce qui concerne l'entreprise virtuelle. De nombreuses pistes de recherche s'ouvrent et laissent entrevoir d'intéressantes perspectives surtout dans le domaine des Sciences de Gestion. Le concept ne représente donc certainement pas un aboutissement en termes de mode d'organisation.

1.1.3.2 Virtual Breeding Environnement

Dans une économie mondiale de plus en plus concurrentielle, la notion de « *Virtual Breeding Environment* », connue aussi sous le nom de cluster, suscite un intérêt fort de la part des entreprises industrielles. C'est au début des années 90, que Michael Porter, Professeur à la *Harvard Business School*, a popularisé le concept de cluster et la description des phénomènes de regroupement d'entreprises.

Selon [Sánchez et al., 2005], la notion de cluster représente l'association de plusieurs entreprises qui ont à la fois le potentiel et la volonté de coopérer les unes avec les autres sur de longues périodes. Quand une opportunité d'affaire est identifiée par un membre (agissant comme un courtier), un sous-ensemble de ces entreprises peut être sélectionné et forme donc une entreprise virtuelle. Cette définition souligne la relation qui puisse exister entre le concept de cluster d'entreprises et celui d'entreprise virtuelle.

Une autre définition semblable est proposée par [Camarinha-Matos and Afsarmanesh, 2006] qui définissent la notion de cluster comme une association d'organisations qui adhèrent à des coopérations à long terme, et adoptent des principes de fonctionnement et des infrastructures communes, avec l'objectif principal d'accroître leurs chances pour participer à des entreprises virtuelles. Dans le même article, les auteurs identifient les différentes raisons qui poussent les entreprises vers ce genre de pratique et qui sont essentiellement (voir Figure 1.2) : des raisons de marché (mieux répondre aux exigences du marché), des raisons organisationnelles et des raisons de préparation pour participer à des coopérations à la demande (*preparedness*).

Figure 1.2. Raisons liées à la formation des clusters d'entreprises
[Camarinha-Matos and Afsarmanesh, 2006]

Nous considérons que la définition proposée par [Camarinha-Matos and Afsarmanesh, 2006] est la plus adéquate pour définir le concept de cluster d'entreprises, d'ailleurs c'est la définition la plus citée dans la littérature relative aux coopérations à la demande.

Une multitude de clusters d'entreprises ont vu le jour parmi lesquels nous pouvons citer : les industries pharmaceutiques aux États-Unis, le cluster de biotechnologie au Canada, *etc.* [Sánchez et al., 2005]. Malgré le nombre croissant de clusters, plusieurs auteurs affirment que le succès des clusters est fortement lié à la mobilisation active des entreprises. Les avantages générés par le cluster sont fonction de mutualisation des connaissances, meilleures pratiques, et technologies des entreprises qui le composent.

1.1.3.3 Communauté virtuelle d'entreprises

Les « communautés virtuelles d'entreprises » constituent aujourd'hui une nouvelle forme de la coopération à la demande. Les définitions proposées pour cette forme de coopération conduisent à privilégier l'idée d'un groupe d'individus qui acceptent volontairement d'interagir au sein d'un espace de communication virtuel (« une communauté virtuelle existe lorsqu'il est possible à un groupe d'individus de se rencontrer et d'interagir dans le cyberespace et lorsque ces individus choisissent volontairement de participer à ces rencontres et à ces interactions ») [Steinmueller, 2002]. Cette définition implique qu'une communauté est constituée par un processus d'interaction spécifique basé sur l'adhésion d'agents dotés d'une autonomie personnelle. Ce processus peut être, par la suite, à l'origine de la constitution d'un capital cognitif commun [Amin and Cohendet, 2003].

La question primordiale qui doit être examinée quand on considère des communautés virtuelles d'entreprises consiste à vérifier que la définition générale d'une communauté virtuelle proposée par [Steinmueller, 2002] continue à s'appliquer lorsqu'elle concerne des entreprises. En effet, lorsque des entreprises participent à des communautés en relation avec les marchés de type « *Business to Business* », les individus qui sont concrètement impliqués dans les échanges correspondants ne constituent qu'une partie réduite de l'ensemble des employés ou des cadres des entreprises concernées. Ce qui signifie que, seuls, certains

individus participent à la vie de la communauté tout en impliquant à travers leurs décisions et actions d'autres services ou divisions fonctionnelles des entreprises dont ils font partie. Ainsi, les communautés virtuelles auxquelles on se réfère dans cet état de l'art demeurent concrètement des communautés d'individus conformément à la définition de [Steinmueller, 2002]. Cependant, certains auteurs comme [Arena, 2003] considère que la vraie difficulté renvoie à la notion de choix autonome contenue dans la définition énoncée par [Steinmueller, 2002]. En effet, le membre de l'entreprise participant à des échanges communautaires est généralement un cadre dirigeant qui bénéficie d'une certaine autonomie de moyens et de décisions liés à l'activité qu'il doit conduire [Arena, 2003]. Ainsi, il n'existe pas de véritable contradiction entre la définition de [Steinmueller, 2002] et l'émergence de communautés d'entreprises.

1.1.3.4 Synthèse

Notre étude de la littérature concernant les formes des entreprises à la demande révèle quelques constats :

- L'entreprise virtuelle se détache de l'approche territoriale de la notion de réseau et prend la forme des processus d'information qui composent son activité en dépassant les contraintes spatio-temporelles. Ceci ne nie pas le fait que ce concept semble hériter d'une grosse partie des fondements de base des réseaux d'entreprises. En effet, le concept d'entreprise virtuelle fait référence à des pratiques managériales rodées au fil de l'histoire [Meissonier, 2000]. Les stratégies de partenariat, de coopération, d'externalisation, *etc.*, sont des pratiques que les nouvelles technologies d'information ont permis d'intensifier. En outre, les principes fondamentaux de l'entreprise virtuelle (réseau d'acteurs, travail à distance via les systèmes de communication) se retrouvent donc dans des formes d'entreprises préexistantes dont l'évolution de la réticulation a suivi celle des systèmes de communication.
- L'organisation d'entreprises sous forme de clusters est de nature structurelle, même si la forme que prend le cluster peut se réaliser d'une manière dynamique et ne cesse d'évoluer. En effet, afin de faire face à un marché concurrentiel, les entreprises expriment le besoin de s'inscrire dans des clusters. Grâce aux relations que l'entreprise peut entreprendre avec les différentes entreprises du même cluster, elle pourra développer sa compétitivité, synonyme d'une inscription durable sur le marché.
- Le domaine de la recherche sur les communautés virtuelles d'entreprises est relativement récent et peu de publications ou études scientifiques ont été proposées. Nous avons essayé d'éclaircir les spécificités qui permettent d'étendre la portée de la notion de communauté à une population d'entreprises. Nous nous référons à l'étude réalisée par [Arena, 2003], qui souligne l'importance des trois critères essentiels qui participent dans la définition d'une communauté virtuelle. Le premier est lié à l'existence d'une volonté autonome des agents et débouche sur le rôle joué par les formes de communication dans l'établissement des relations au sein de la communauté. Le deuxième critère a trait à la nature de l'engagement collectif qui caractérise les communautés. Le dernier critère concerne la

place centrale occupée par la notion de confiance organisationnelle dans le mode de gouvernance et de régulation des communautés.

Afin d'étudier les similitudes et les différences entre les trois formes de coopération à la demande nous nous sommes basés sur les deux familles de caractéristiques de coopération proposées par [Rajsiri et al., 2007]. Les caractéristiques que nous retenons sont : l'objectif et la dynamique du réseau.

- L'objectif du réseau : regroupe essentiellement les deux critères suivants :
  - La nature de relation : détermine le type de relations entre deux partenaires. Il existe trois types de relation à savoir :
    - La *relation de concurrence* (*ou horizontale*) concerne la relation entre entreprises qui sont dans le même secteur ou la même industrie, les relations horizontales s'intéressent au domaine du management stratégique.
    - La *relation de sous-traitance, client/fournisseur* (*ou verticale*) concerne la relation entre une entreprise et ses partenaires lui fournissant un service complémentaire vis-à-vis de ses actions.
    - La relation *groupe d'intérêt* (ou *transversale*) concerne les entreprises qui ont les mêmes intérêts, par exemple, pour développer des technologies partageables.
  - Le métier : permet de déterminer l'activité principale du réseau. Les métiers sont par exemple, industrie, commerce, service, *etc.*
- La dynamique du réseau : comporte deux critères à savoir :
  - La fréquence d'interactions : il s'agit de l'aspect temporel qui explique la dynamique de la relation et qui permet de définir si le réseau relève d'une relation opportuniste, occasionnelle ou permanente [Bénaben et al., 2006].
  - La stabilité : il s'agit de préciser si un réseau est fixe ou évolutif. Un réseau est fixe quand les mêmes partenaires forment le réseau tout au long de son cycle de vie. Il est évolutif quand des partenaires peuvent quitter le réseau.

Le Tableau 1.2 synthétise une comparaison entre les trois formes de coopération à la demande.

Formes de coopération / **Critères**	**L'entreprise virtuelle**	**Les clusters d'entreprises (*Virtual Breeding Environment*)**	**Les communautés virtuelles d'entreprises**
Nature de relation	- Relation de concurrence - Relation de sous-traitance - Relation groupe d'intérêt	- Relation de concurrence - Relation groupe d'intérêt	- Relation groupe d'intérêt
Métier	Plusieurs métiers	Plusieurs métiers	Plusieurs métiers
Fréquence d'interactions	Opportuniste	Opportuniste, ou Permanente	Permanente
Stabilité	Évolutif	Évolutif	Évolutif

Tableau 1.2. Comparaison entre les trois formes de coopération à la demande

L'analyse du Tableau 1.2 montre un ensemble de constatations :

- La distinction majeure entre une communauté d'entreprises virtuelles d'une entreprise virtuelle ou d'un cluster d'entreprises est le développement des biens informationnels. En effet, la valeur additionnelle générée par les communautés c'est l'aspect culture et connaissance. Elle peut aussi générer exclusivement des informations. Pour que ces informations se transforment en connaissances, il suffit qu'elles aient un intérêt objectif pour ceux qui les partagent. Certains auteurs comme [Gensollen, 2003] introduisent le concept de communauté d'expérience qui permet une meilleure commercialisation des biens d'expérience en fournissant des procédures efficaces.
- La fréquence d'interaction entre les entreprises peut varier en fonction des formes de coopération. Pour l'entreprise virtuelle, il s'agit généralement de relation opportuniste qui se dissout à la fin d'un projet. Tandis que les communautés virtuelles d'entreprises, elles entretiennent des relations qui peuvent se créer d'une manière dynamique et pour un horizon qui se définit sur le long terme.
- Le cluster d'entreprises est une forme qui se positionne entre l'entreprise virtuelle et les communautés virtuelle d'entreprises. Il hérite de l'entreprise virtuelle son caractère opportuniste et des communautés virtuelles d'entreprises l'aspect permanent des relations entre les partenaires.

Bien que les concepts de clusters d'entreprises et de communauté virtuelle présentent une valeur ajoutée au niveau de la coopération, il est important de souligner que notre travail de recherche se focalise particulièrement sur le concept d'entreprise virtuelle. Néanmoins, le concept de communauté est un concept intéressant et qui peut être considéré selon plusieurs niveaux d'abstraction (niveau métier et niveau technique). Nous allons illustrer dans le chapitre 3 de ce manuscrit de thèse l'utilisation du même concept mais selon une perspective plus technique que celle présentée précédemment.

La complexité de la mise en place de la coopération à la demande nécessite certains changements au niveau organisationnel et technique. Dans ce qui suit, nous attardons sur les concepts métiers et techniques qui supportent les mécanismes de coopération.

1.1.4 Concepts relatifs à la vue métier de la coopération

Dans cette section, nous nous intéressons au niveau métier de la coopération indépendamment de toute considération technique. Ce niveau décrit l'organisation des acteurs, des interactions et des échanges au sein d'un « processus coopératif ». Nous allons nous intéresser surtout au concept de processus métier, au processus interentreprises (ou processus coopératif) ainsi qu'aux formalismes les plus fréquemment utilisés pour les modéliser.

1.1.4.1 Notion de processus

Plusieurs définitions du terme processus sont présentées dans la littérature. L'étude de ces définitions permet d'éclaircir ce concept.

D'un point de vue général, le dictionnaire Larousse définit un processus comme un enchaînement ordonné de faits ou de phénomènes, répondant à un certain schéma et aboutissant à un résultat.

Du point de vue de l'automatique et de la productique, [Vernadat, 1999] considère un processus métier (en anglais, *business process*) comme « *une succession de tâches qui contribue à la réalisation des objectifs de l'entreprise. De manière générale, un processus peut être défini comme un enchaînement d'activités à exécuter pour atteindre un but donné* ».

Du point de vue organisationnel, [Harrington, 1991] fixe un processus comme « *n'importe quelle activité ou groupe d'activités qui reçoit une entrée, lui ajoute de la valeur et fournit une sortie à un client interne ou externe. Les processus utilisent les ressources de l'organisation pour fournir des résultats définitifs* ».

[Mili et al., 2004] avancent qu'un processus métier est un ensemble d'activités exécutées par des acteurs jouant des rôles particuliers, consommant quelques ressources et en produisant d'autres. Les activités peuvent être déclenchées par des événements et peuvent à leur tour produire des événements. Les activités d'un processus peuvent être liées par des dépendances de ressources (dépendances de producteur-consommateur) ou des dépendances de commande (une activité déclenchant une autre).

À l'instar de ces définitions, nous considérons un processus métier comme un enchaînement d'activités réalisées par un ensemble d'acteurs de l'entreprise et produisant une valeur ajoutée pour celle-ci. En effet, un processus reçoit des objets en entrée et leur ajoute de la valeur, par le moyen de ressources, tout en fournissant des objets de sortie (produits/services) remplissant les besoins et les exigences d'un client (atteindre les objectifs).

1.1.4.2 Notion de processus interentreprises

La coopération interentreprises se traduit par la formation de processus interentreprises construit à partir de l'interconnexion de divers processus. [Morley et al., 2005] définissent un processus interentreprises comme un processus où « *les partenaires ont une maîtrise partielle sur ce dernier* » due à une synergie ou d'un partage des activités de processus entre les partenaires. Ces derniers sont responsables uniquement de l'exécution de leurs activités. Chaque activité dans un processus coopératif fait référence à un processus interne d'un partenaire et présente une interface dédiée à la coopération, sans que le savoir-faire de l'entreprise ne soit visible pour les autres partenaires.

[Bitcheva, 2003] définit un processus coopératif par rapport à l'objectif qu'il poursuit à savoir : la coordination des activités des partenaires. En effet, un processus coopératif doit gérer l'échange des résultats entre les partenaires et contrôler le respect des conventions prédéfinies. La particularité d'un processus interentreprises est que ses activités sont distribuées à travers les différentes entreprises participantes dans la coopération (voir Figure 1.3).

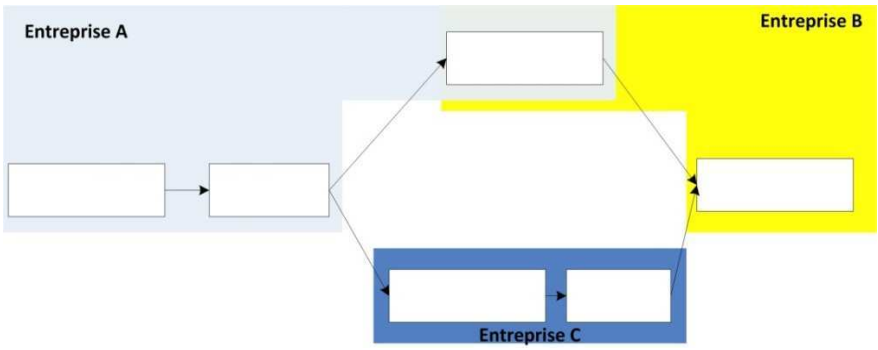


Figure 1.3. Processus coopératif dans le cadre de l'entreprise virtuelle [Bitcheva, 2003]

[Akkermans et al., 2004] précisent qu'un processus coopératif regroupe des acteurs qui travaillent ensemble de façon transparente et confiante en pilotant des flux de travail communs grâce à des technologies de l'information adaptées. Ils avancent également que cette pratique mène à une plus grande confiance et transparence entre les partenaires ainsi qu'une amélioration de la performance globale de l'entreprise.

Dans un travail récent, [Grefen et al., 2009] définissent le processus coopératif dans un contexte d'entreprise virtuelle comme la composition dynamique des processus locaux (processus intra-entreprises) au sein d'un processus global appartenant à l'entreprise virtuelle à la demande (appelée encore entreprise virtuelle instantanée).

Ce qui émerge des définitions déjà présentées est ce qui suit. Un processus coopératif (ou interentreprises) est un processus résultat de l'interconnexion de plusieurs processus appartenant à diverses entreprises. À l'opposé des processus traditionnels (intra-entreprise), où les activités font partie de la même entreprise, le processus interentreprises est le résultat de l'enchaînement de plusieurs activités issues de plusieurs entreprises. Ces différentes activités échangent des informations et des services entre elles.

Afin de capitaliser le bénéfice escompté d'un processus métier en général et d'un processus interentreprises en particulier, ces derniers doivent être modélisés. Nous allons présenter dans la prochaine section les différents formalismes de modélisation des processus.

1.1.4.3 Formalismes de modélisation de processus

La modélisation est définie 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 » [LeMoigne, 1990]. Un modèle possède une syntaxe définie dont chaque élément véhicule une sémantique particulière. Nous revenons en détail sur la définition du concept « modèle » dans la suite de ce chapitre (cf. section 1.2.4.1).

Initialement, les formalismes de modélisation ont été développés dans l'objectif de représenter les flux d'information et de matières au sein d'un processus. Ensuite, ils ont été utilisés pour aider à la conception de processus efficaces grâce à l'introduction des outils de simulation. Nous allons présenter dans ce qui suit, une revue de la littérature concernant les formalismes de modélisation existants. Ces formalismes peuvent être groupés en deux catégories, qui ne possèdent pas le même niveau de maturité [Boukadi et al., 2009a] :

- Formalismes dédiés à la représentation graphique des processus : ce sont les formalismes qui proposent essentiellement un langage graphique de représentation des processus,

- Formalismes dédiés à la fois à la représentation et la simulation des processus : ce sont les formalismes qui proposent à la fois des outils graphiques ainsi qu'une aide à l'analyse et/ou à l'étude du comportement dynamique par le biais de la simulation.

- **Formalismes dédiés à la représentation graphique des processus**

Dans cette catégorie de formalismes on trouve par exemple :

– **Diagramme de flux (*flowcharts*)**

Les diagrammes de flux sont parmi les premiers formalismes proposés pour la modélisation des processus. Ils se basent sur l'utilisation des symboles simples, de lignes et de flèches afin de représenter graphiquement une séquence logique d'activités. Un des points forts de ces diagrammes, c'est qu'ils sont simples et ils permettent d'avoir une vue globale et rapide des processus.

– **SADT (*Structured Analysis and Design Technique*)**

SADT proposée par Ross [Ross, 1977] à la fin des années 1970 pour permettre une analyse structurée des systèmes. SADT a ouvert la voie à la modélisation par représentation graphique des activités et des chaînes d'activités. La méthode SADT introduit le principe de décomposition fonctionnelle et formalise le concept d'activité. Elle se présente comme un langage graphique avec un ensemble limité de primitives, des « boîtes » et des « flèches », pour la représentation des composants des systèmes et des interfaces.

– **Chaîne de processus événementielle**

La chaîne de processus événementielle ou « *EPC-Event driven Process Chain* » en anglais, représente une notation graphique des processus métier introduite dans les années 90 par [Keller et al., 1992]. Cette représentation est rapidement devenue un standard de modélisation des processus dans l'industrie allemande. Plusieurs solutions informatiques intègrent cette notation comme SAP ou ARIS d'IDS- Scheer.

Chaque processus métier est modélisé grâce à un diagramme EPC. Ce dernier comporte des éléments visuels signifiant une logique particulière. Les événements déclenchent les fonctions qui elles-mêmes produisent des événements. La notion de fonction utilisée dans un diagramme EPC correspond à une tâche. Le diagramme EPC représente alors un processus métier comme une succession de fonctions et d'événements.

Outre les notions de fonctions et d'événements, un diagramme EPC comporte un ensemble d'opérateurs logiques. Ces derniers régulent le cheminement du flux d'activités. Il en existe trois principaux à savoir : le « et » logique (AND), le « ou » (OR) et, le « ou exclusif » (XOR).

Malgré sa simplicité apparente, la notation EPC est bien adaptée à la modélisation des processus métiers.

– **UML (*Unified Modelling Language*)**

UML [OMG, 2007] qui est un standard de l'OMG, est à l'origine un langage de modélisation de systèmes d'information à base d'objets distribués, visant à unifier les concepts majeurs introduits par les méthodologies Booch [Booch, 1992], OMT (*Object Modeling Technique*) [Rumbaugh et al., 1991] et OOSE [Jacobson, 1993]. Il permet de modéliser les multiples aspects d'un système. UML offre bien, au travers des diagrammes d'activité une possibilité de modéliser des processus métier. Ainsi, UML dans ses versions UML (1.1, 1.2, 1.3, 1.4) présente l'avantage d'offrir à la fois un méta-modèle ainsi qu'une notation pour la modélisation des processus. Cependant, sa portée reste limitée à la conception objet et son méta-modèle comporte certaines faiblesses sémantiques qui réduisent par conséquent son opérationnalisation pour la modélisation des processus. Certains auteurs comme [Morley, 2000] affirme cette assertion en ajoutant que le langage UML n'intègre pas, nativement, le concept de processus, vu qu'il n'est pas destiné, à l'origine, à ce type de modélisation. Cette faiblesse a été reconnue par l'OMG qui a profondément revu ce modèle dans la version 2 d'UML.

Le nouveau diagramme d'activité d'UML 2.0 offre une base robuste pour la modélisation des processus. De plus, la proposition du concept de profil UML permet de spécialiser le langage UML au contexte de modélisation des processus. D'une façon générale, un profil UML est un « package » stéréotypé contenant des éléments adaptés à un domaine spécifique [Heckel et al., 2003]. Nous revenons en détail sur le concept de profil UML (cf. section 1.2.4.1).

Comme exemple de profil UML nous pouvons noter le profil EDOC (*Entreprise Distributed Object Computing Systems*) [OMG, 2002], qui permet en partie de décrire un ensemble d'extensions du langage UML pour formaliser les processus métiers. Il propose des concepts pour composer les activités, sélectionner des critères sur les entités qui supportent ces activités et également coordonner les processus (évènement, communication, *etc.*).

- **Formalismes dédiés à la fois à la représentation et la simulation des processus**

Parmi les formalismes de modélisation dans cette catégorie on note :

– **Réseau de Pétri**

Le Réseau de Pétri (RdP) est une spécification mathématique qui se base sur des outils graphiques permettant de modéliser et d'analyser les systèmes discrets, plus particulièrement les systèmes concurrents et parallèles [Petri, 1962]. Dans le milieu industriel, le Réseau de Pétri a connu un succès grandissant pour la modélisation des processus métiers. Ce succès est dû à plusieurs facteurs. En effet, grâce à son rôle d'outil graphique, il est possible de produire une compréhension facile du processus modélisé. Il permet également de simuler les activités dynamiques et concurrentes. De plus, son rôle d'outil mathématique permet d'analyser le processus grâce aux modèles de graphes et aux propriétés algébriques (borné, vivacité du réseau, absence de blocage, *etc.*).

Les notations graphiques proposées par le Réseau de Pétri sont : les places (nœuds), les arcs (flèches) et les transitions (contrôles). Les places représentent les 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.

– **BPMN (*Business Process Modeling Notation*)**

La spécification BPMN décrit une notation standard de modélisation des processus métiers. Initialement proposée par le BPMI (*Business Process Management Initiative*), la spécification est actuellement maintenue par l'OMG (*Object Management Group*).

Le standard BPMN vise à offrir une notation explicite, visuelle et accessible pour tous les acteurs de l'entreprise. D'une façon plus ambitieuse, BPMN souhaite être un standard pour la conception des systèmes d'information. Bien que la notation BPMN soit destinée aux analystes métier, elle tient compte du paradoxe existant entre la nécessité de modéliser un processus métier et son exécution dans un système de gestion de processus. En effet, BPMN propose des passerelles pour l'exécution automatique des processus (sous format BPEL : *Business Process Execution Language*) à l'aide de moteurs BPMS (*Business Process Management System*). C'est la raison pour laquelle nous le considérons comme un langage qui offre des possibilités de simulation (ou d'execution).
BPMN fournit quatre types d'éléments pour la représentation graphique à savoir :

- Les conteneurs représentés par les couloirs (*pool*) et les bandes (*lane*). Un couloir représente un participant dans un processus qui peut être une entité spécifique (par exemple, une entreprise) ou encore un rôle (par exemple, un fournisseur, un client, un distributeur, *etc.*),
- Les objets de flux (*flow objects*) : comportent trois types d'objets principaux les activités (*activities*), les évènements (*events*) et les branchements conditionnels (*gateways*),
- Les objets de relation (*connecting objects*) : les objets de flux peuvent être connectés entre eux par trois types d'objets de connexion (les flux séquence, les flux message et les associations),
- Les objets symboliques (*artifacts*) : les objets symboliques apportent des informations et des précisions supplémentaires. Parmi les objets symboliques, nous pouvons citer, l'objet de données (*data object*) qui permet de représenter les documents, les données et les entités pouvant être utilisés dans un processus.

En somme, la spécification BPMN fournit une panoplie d'avantages. Tout d'abord, elle propose une notation à la fois riche en concepts et facilement maîtrisée par les acteurs de l'entreprise. Ensuite, les éléments de modélisation suggérés sont rigoureusement définis et offrent un socle robuste pour l'outillage de la modélisation des processus métiers complexes. Enfin, une modélisation basée sur BPMN peut être traduite en un processus exécutable BPEL.

- **Synthèse**

En examinant les différents formalismes présentés, nous constatons que six éléments de modélisation sont pris en compte dans la majorité des formalismes de modélisation [Boukadi et al., 2009b]. Ces éléments sont (voir Figure 1.4) : les activités, les événements, les flux de contrôle, les objets de relation, les objets symboliques et les couloirs.

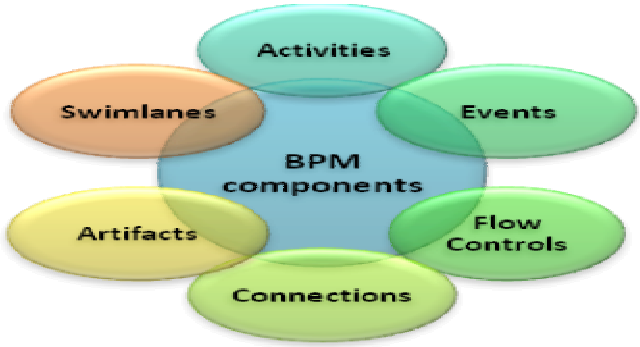


Figure 1.4. Éléments de modélisation communs entre les formalismes [Boukadi et al., 2009b]

En outre, l'étude que nous avons mené a montré l'évolution de la présentation graphique et des techniques de contrôle de flux d'activités des formalismes. Certes, il existe d'autres formalismes que nous n'avons pas abordés dans notre étude comme PSL (*Process specification Language*) [Schlenoff et al., 1999] et POP* (Product, Organization, Process and Systems) [ATHENA, 2005].

Notre étude a révélé également l'existence de catégories de formalismes : ceux dédiés à la représentation et ceux dédiées à la fois à la représentation et à la simulation des processus. Certains formalismes, comme BPMN, permettent de couvrir plusieurs aspects du processus en incluant ses acteurs et ses informations. Cependant, plusieurs auteurs considèrent que cette couverture reste restreinte vu que dans la majorité des cas c'est l'aspect fonctionnel du processus qui est prépondérant dans les formalismes [Morley et al., 2005].

Certains auteurs ont poussé encore les recherches [Rosemann and Recker, 2006] [Nurcan, 2008] afin d'inclure les informations contextuelles lors de la modélisation des processus métiers. C'est un courant de recherche assez nouveau et donc peu de travaux ont vu le jour. Dans un article proposé par [Rosemann and Recker, 2006], les auteurs soulignent l'importance de l'intégration des éléments de contexte (par exemple, le contexte de l'utilisateur impliqué dans une activité, son âge, ses capacités mentales, *etc.*) pour assurer une modélisation flexible. D'autres auteurs ont continué sur cette voie de recherche comme [Saidani and Nurcan, 2007]. Dans leur travail, les auteurs partent du constat qu'une des principales limites des approches de modélisation de processus est qu'elles présument que les comportements des utilisateurs et des rôles sont définis à priori. Ainsi, les auteurs considèrent que la connaissance relative au contexte est importante et doit être prise en compte dans l'ingénierie des processus métiers. En réalité, les capacités et le comportement des processus métiers peuvent changer selon le contexte dans lequel l'entreprise et les utilisateurs eux-mêmes se trouvent. Par exemple, dans une situation d'urgence, un utilisateur peut assumer une responsabilité, qui, dans un contexte normal, nécessite les compétences d'un utilisateur plus qualifié.

Nous considérons que cette orientation est importante pour assurer la flexibilité des processus métiers et bien évidement des services issus de ces processus. Nous allons proposer dans le chapitre 2 de ce manuscrit de thèse notre approche de modélisation des processus orientée contexte.

Afin de comparer les différents formalismes de modélisation déjà présentés, nous nous basons sur un ensemble de critères issus de différentes études réalisées dans la littérature [Morley et al., 2005], [Nurcan, 2008], [Boukadi et al., 2009b]. Les critères sectionnés sont les suivants :

- **Description de l'enchaînement d'activités :** un processus métier doit être décrit en termes d'enchaînement d'activités,
- **Expression de la granularité :** un processus peut se décomposer en sous processus,
- **Traitement de conditions :** capacité à intégrer les conditions d'exécution d'une activité en fonction des variables internes au processus,
- **Gestion des évènements :** intégration des évènements externes et des exceptions dans le processus.
- **Contrôle avancé de flux de processus :** capacité à inclure des symboles de contrôle avancés tels les branchements de décision orientés événements, *etc.*
- **Passerelle vers une définition exécutable :** présence de passerelle vers une définition exécutable de processus tels que BPEL (*Business Process Execution Language*) ou encore BPML (*Business Process Modelling Language*), *etc.*,
- **Possibilité d'extension :** capacité du formalisme à supporter des ajouts pour des raisons d'évolutivité.

Le Tableau 1.3 illustre une comparaison des différents formalismes basés sur les critères sélectionnés.

Formalisme / **Critère**	**Diagramme de flux**	**SADT**	**EPC**	**UML**	**RdP**	**BPMN**
Description de l'enchaînement d'activités	Oui	Oui	Oui	Oui	Oui, surtout les Réseaux de Pétri hiérarchiques	Oui
Expression de la granularité	Non	Oui	Non	Oui	Oui, surtout les Réseaux de Pétri hiérarchiques	Oui
Traitement de conditions	Non	Non	Oui	Oui	Oui	Oui
Gestion des évènements	Non	Non	Oui	Oui	Oui, surtout les Réseaux de Pétri colorés	Oui
Contrôle avancé de flux de processus	Non	Non	Non	Non	Non	Non
Passerelle vers une définition exécutable	Non	Non	Oui	Non	Non	Oui
Possibilité d'extension	Non	Non	Non	Oui, grâce au concept	Non	Oui, grâce au concept d'objet symbolique

				de profil		

Tableau 1.3. Comparaison des formalismes de modélisation des processus

L'analyse du Tableau 1.3 montre que les diagrammes de flux est le formalisme qui répond le moins aux critères. UML et BPMN semblent répondre à la majorité des critères. Néanmoins, BPMN a été introduit spécifiquement pour la modélisation des processus métiers (à l'encontre d'UML).

La plupart des formalismes de modélisation ne privilégient pas la transposition automatique des modèles de processus métiers dans un langage d'exécution. Seul le standard BPMN possède des passerelles vers le langage d'exécution BPEL. Ce dernier point est capital, dans le sens où BPMN permet de réconcilier les analystes métiers responsables de la modélisation des processus et les analystes IT (*i.e.* informatiques) chargés de l'exécution des processus modélisés.

Fort de ces constats, BPMN semble être le standard adéquat pour la modélisation des processus métiers que ce soit à l'interne ou à l'externe de l'entreprise. Outre ses capacités de modélisation, BPMN possède l'avantage d'être extensible et ceci grâce aux concepts d'objets symboliques. Nous nous tirerons profit de cette spécificité dans le cadre de notre travail de recherche afin de modéliser les informations contextuelles. Le chapitre 2 décrit en détail cette particularité (cf. chapitre 2, section 2.4.2).

1.1.5 Concepts et approches relatifs à la vue technique de la coopération

Dans le cadre de nos travaux, nous nous intéressons à la coopération interentreprises selon deux points de vue : métier et technique (voir Figure 1.1). Cette section décrit les technologies et standards qui ont facilité la coopération interentreprises au niveau technique. Nous proposons une classification en trois grandes catégories selon la nature de la communication entre les processus :

- Communication entre processus par envoi de messages
- Coordination transactionnelle des processus
- Interconnexion des processus par échange de services

Dans ce qui suit, nous allons détailler chacune de ces techniques.

1.1.5.1 Communication entre processus par envoi de messages

Profitant des standards d'échanges de données comme le langage XML, le mécanisme d'envoi de messages a été beaucoup présenté comme l'une des meilleures solutions assurant l'interconnexion des processus d'entreprises. En effet, ce type de mécanismes facilite la coopération à travers des paradigmes tels que l'abonnement et la notification d'événements. Cette section a pour objet de présenter quelques exemples de communication entre processus par envoi de messages.

- **Intergiciels orientés messages**

Le mécanisme d'envoi de message se base essentiellement sur des plateformes appelées des intergiciels (*middlewares*) orientés messages (*Message Oriented Middleware* ou MOM). Ces derniers permettent l'échange de messages entre applications en utilisant une communication asynchrone.

L'intérêt de ce mode de communication (asynchronisme) réside dans le fait que les applications qui échangent des messages ne sont pas liées l'une à l'autre [Hohpe and Woolf, 2004], elles peuvent donc avoir des caractéristiques de disponibilité complètement différentes. Les principaux modèles de middlewares orientés messages sont le modèle de queue de message (*message queuing*), le modèle par abonnement (ou modèle *Publish/Subscribe*) et le modèle événementiel. Le modèle par abonnement est le modèle le plus utilisé. Il propose deux types d'applications : des applications clientes et des applications fournisseurs. Les applications clientes (*consumers*) s'abonnent à des types d'événements tandis que les applications fournisseurs (*providers*) produisent et publient des événements. Les différents événements sont alors envoyés au gestionnaire de communications qui se charge de les répartir aux applications qui se sont abonnées à ces événements.

Lors d'une interconnexion des processus basée sur le déploiement d'un intergiciel MOM, le message représente le contrat entre le serveur gérant le processus d'une entreprise et le client invoquant le processus. Par conséquent, les interfaces des processus de l'entreprise sont remplacées par ces messages. De plus, l'application cliente ne communique qu'avec le MOM et n'a pas besoin de posséder beaucoup d'informations sur le serveur (par exemple, localisation, configurations, interface, *etc.*). Cela transforme le problème de l'interconnexion des processus en un problème purement technique : la communication par message entre le serveur du processus et le MOM d'une part et d'autre part entre le MOM et le client du processus [Casati and Discenza, 2000].

- **ebXML (*Electronic Business XML*)**

ebXML est un standard, sous l'égide de l'ONU UN/CEFACT (*United Nations Center For Trade Facilitation and Electronic Business*) et OASIS (*Organization for the Advancement of Structural Information Standards*) ayant pour objectif de définir une infrastructure globale de commerce électronique basée sur XML et Internet. Le standard ebXML se focalise sur les processus métiers et la gestion des transactions. Il définit une architecture qui décrit les formats de message et les composants génériques utilisés dans ces processus. Un des avantages d'ebXML est sa généralité, ce qui suppose qu'il peut être utilisé dans différents secteurs industriels.
Les principales spécifications ebXML sont [Chauvet, 2002] :

- *Core Component* : caractérise l'ensemble des schémas XML. Ces schémas comportent les formats de données métier, comme les dates, les montants de taxation, le contrat d'échange, *etc.* Un *Core Component* capture les informations concernant un concept métier qui peut être instancié dans plusieurs types d'échanges d'informations utilisés pour définir les processus métiers et les modèles d'information. Ce composant permet de faciliter l'intégration entre différents systèmes,
- *Business Process* : permet de décrire les interfaces du processus métier,

- *Trading Partner Profile* : permet de définir deux types de contrats le CPP (*Collaboration Protocol Profile*) et le CPA (*Collaboration Protocol Agreement*). Le CPP représente les capacités d'affaire d'une entreprise dans un format standard et portable (profile général, protocoles supportés, *etc.*). Quant au CPA, il décrit les éléments et les mécanismes de transactions conduits entre deux entreprises,
- *ebXML Registry & Repository* : permet de définir les mécanismes de sauvegarde et de recherche dans les registres et référentiels ebXML,
- *Transport & Routing* : permet de définir une méthode pour échanger des messages électroniques.

1.1.5.2 Coordinations transactionnelles des processus

Les « Workflows » sont considérés comme des transactions de longue durée qui accèdent à un ensemble de ressources distribuées [Baina et al., 2003]. En effet, un Workflow se définit comme l'automatisation du tout ou d'une partie d'un processus d'entreprise. Il définit un ensemble de tâches composantes, leur coordination, ainsi que l'information et les acteurs impliqués dans chaque tâche [WFMC, 1999]. Un système de gestion de Workflow (SGWF) est un système qui définit, implémente et gère l'exécution d'un ou de plusieurs Workflow à l'aide d'un environnement logiciel. Cet environnement doit fonctionner avec un ou plusieurs moteurs de Workflow et doit être capable d'interpréter la définition d'un processus, de gérer la coordination des participants et d'appeler des applications externes [WFMC, 1999].

Le Workflow a subi, lui aussi l'influence des nouvelles formes de travail et d'organisation guidées par la nouvelle économie. Il s'est avéré nécessaire de mettre en commun et de faire coopérer les processus de plusieurs entreprises réparties, autonomes et hétérogènes. Ces coopérations se déroulent dans le cadre des projets à long terme, aussi bien qu'à l'occasion de courtes transactions telles que celles du commerce électronique. C'est dans ce contexte que s'est développé le domaine du Workflow interentreprises qui constitue une extension du Workflow traditionnel.

[Bouzguenda, 2005] définit un Workflow interentreprises comme une collection de Workflow locaux (ou composants) *distribués*, éventuellement *hétérogènes*, ayant chacun une activité propre plus ou moins *autonome*, et travaillant de manière *coopérative* pour répondre à un objectif commun. L'objectif ultime d'un Workflow interentreprises est de composer/coordonner les différents Workflows locaux avec une contrainte forte imposée par chaque partenaire qui veut maintenir le contrôle sur son Workflow local.

Cependant les systèmes de Workflows interentreprises existants comme par exemple les systèmes DERPA [Angelov and Grefen, 2002] et PointSynchro [Bitcheva, 2003], traitent des coopérations structurelles dans lesquelles les partenaires sont engagés dans une coopération à long terme et proposent d'interconnecter d'une manière serrée leurs Workflows respectifs. Cette limite a poussé certains auteurs comme [Bouzguenda, 2005] à s'intéresser au domaine de la coordination dans le Workflow InterOrganisationnel (WIO) lâche, qui correspond à une coopération occasionnelle. Dans son travail de recherche, l'auteur propose une architecture à base d'agents qui implante le Service de Déploiement de Workflow (SDW). Cette architecture étend l'architecture de référence proposée par la WorkFlow Management Coalition (WFMC)

pour traiter la recherche de partenaires et la négociation entre partenaires [Andonoff and Bouzguenda, 2005].

L'originalité de ce travail réside dans la proposition d'une solution orientée Workflow dans un contexte de coopération dynamique. Ce contexte (coopération à la demande) rejoint nos travaux de recherche et constitue encore une voie de recherche prometteuse où peu de travaux ont vu le jour.

1.1.5.3 Interconnexion des processus par échange de services

- **Services Web**

Les services Web constituent l'instanciation la plus importante de l'architecture orientée services dans le domaine industriel. Le consortium W3C définit le concept de service Web comme un système logiciel identifié par un URI, dont les interfaces publiques sont définies et décrites en XML. Sa définition peut être découverte sur le Web par d'autres systèmes logiciels qui peuvent alors interagir dans une manière prescrite en utilisant les messages basés sur XML transmis par les protocoles Internet.

Les services Web permettent à l'entreprise d'intégrer ses applications hétérogènes ainsi que celles des entreprises partenaires indépendamment des environnements techniques et des langages sur lesquels tournent ces applications. En effet, un service peut être utilisé pour exporter des fonctionnalités d'une application d'entreprise et les rendre accessibles via des protocoles standards. Tel qu'il est illustré dans la Figure 1.5, le service Web sert alors d'interface d'accès et de dialogue avec l'application.

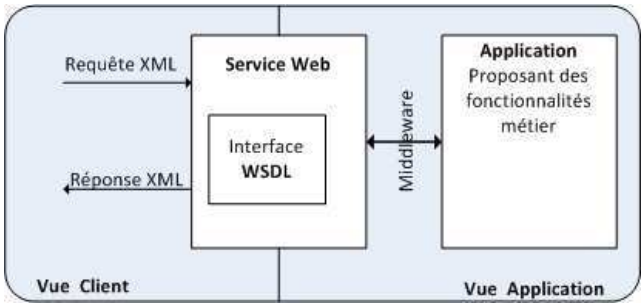


Figure 1.5. Exposition des applications industrielles en des services[Zhao and Cheng, 2005]

De plus, le concept de services Web offre aux entreprises la possibilité d'échanger les services qui peuvent être automatisés au sein de leurs processus. Actuellement, les services Web présentent une force considérable pour résoudre les problèmes d'intégration des applications en général et des processus interentreprises en particulier. Aujourd'hui, ils semblent être la solution la plus adaptée pour assurer l'interopérabilité sur Internet [Hirzalla et al., 2008], [Kellert and Toumani, 2006].

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. Par conséquent, il est alors possible de définir rapidement de nouveaux processus métiers, par assemblage de services

existants. Ceci permet à l'entreprise de réagir rapidement aux évolutions de son marché [Arsanjani, 2004].

- **Bus de services d'entreprise**

Les services Web ont ouvert la voie à d'autres solutions comme par exemple l'ESB (*Enterprise Service Bus*) ou Bus de Services d'Entreprise [Chappell, 2005]. L'ESB est un concept qui a été défini en 2003 par le *Gartner Group* et est considéré comme la convergence des outils EAI (*Enterprise Application Integration*) et des services Web.

L'ESB est une solution « packagée » qui permet de mettre en œuvre l'approche SOA grâce à son concept de base : le bus. Ce dernier permet d'implémenter une solution d'intégration distribuée dépassant l'aspect monolithique des EAI. En effet, le bus permet aux différents services métier (encapsulant des applications d'entreprise) de communiquer et de coopérer. Le bus s'appuie sur un ensemble de services de base [Lublinsky and Tyomkin, 2003] :

- Le moteur d'orchestration qui joue le rôle de service d'orchestration permettant d'exécuter des processus,
- Le service de localisation qui permet de trouver de façon transparente les services,
- Les services utilitaires qui sont des services techniques sollicités par les services métier (par exemple, les services permettant le routage, la transformation, *etc.*),
- Les services d'infrastructure qui sont des services permettant de fournir un support d'infrastructure aux services métier (par exemple, services liés à la sécurité et au monitoring, *etc.*).

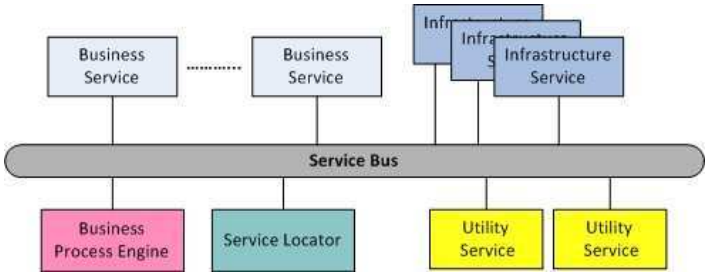


Figure 1.6. Architecture d'un ESB [Lublinsky and Tyomkin, 2003]

Les ESB s'appuyant sur l'utilisation des standards de services Web (SOAP, WSDL, UDDI) et les normes WS-*. Cette particularité facilite en retour l'interopérabilité et l'interconnexion des systèmes d'information des entreprises partenaires en se basant sur un échange des messages transitant d'un bus à un autre (voir Figure 1.7). Cependant, il s'agit des architectures techniques dédiées à des coopérations à long terme.

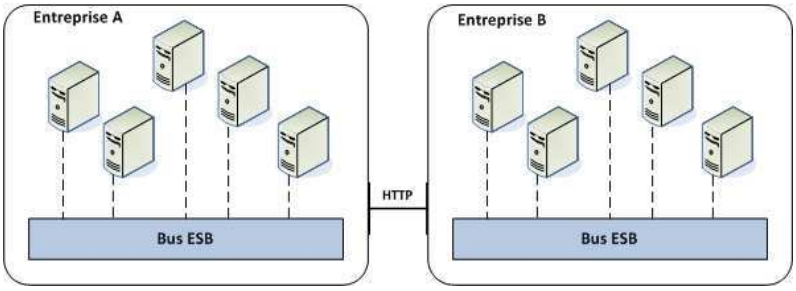


Figure 1.7. Exemple d'entreprises reliées par leurs bus ESB

1.1.5.4 Synthèse et positionnement

Nous avons essayé précédemment d'établir une analyse de la littérature concernant les mécanismes supports de l'interconnexion des processus d'entreprises. Notre revue révèle qu'il existe plusieurs outils et mécanismes qui supportent la coopération. Dans ce qui suit, nous allons comparer et analyser les différentes techniques déjà présentées en se basant sur les cinq critères suivants :

- **Autonomie :** rappelons que l'autonomie est l'une des contraintes exigées par les coopérations interentreprises. Bien que les entreprises travaillent ensemble, elles souhaitent conserver leur entière autonomie. En d'autres termes, l'autonomie permet de définir le degré de couplage entre les systèmes des entreprises partenaires,
- **Dynamisme :** permet de définir le degré d'aptitude de la technique à prendre ou à intégrer des processus des entreprises d'une manière dynamique. En effet, il s'avère difficile de décrire un scénario de coopération qui définisse à l'avance l'ensemble des partenaires ainsi que leurs interactions possibles,
- **Hétérogénéité :** désigne le degré de disparité entre les entreprises (de point de vue messages, sémantique, *etc*.),
- **Ouverture :** permet de montrer le degré de standardisation permis par la technique,
- **Flexibilité :** permet de représenter la capacité de la technique à faire face à de nouveaux changements ou à des scénarios d'évolution.

Le tableau suivant résume les différentes techniques d'interconnexion comparées suivant les cinq critères décrit précédemment.

Critère / **Technique**	**Autonomie**	**Dynamisme**	**Hétérogénéité**	**Ouverture**	**Flexibilité**
Intergiciels MOM	Oui	Non	Oui	Non	N'est pas garanti
ebXML	Non	Non	Non	Oui	N'est pas garanti
Workflows interentreprises	Non	Non, à part les Workflows	Non	Oui	N'est pas garanti

		dans un contexte lâche			
Services Web	Oui	Oui	Oui	Oui	Est garanti
Bus de services d'entreprise (ESB)	Oui	Non	Oui	Oui	N'est pas garanti

Tableau 1.4. Comparatif des techniques de coopération interentreprises

Plusieurs constats émergent de la comparaison des techniques de coopération :

- Les mécanismes d'envoi de messages profitent du langage XML qui fournit une structuration des informations pouvant être échangés entre processus d'entreprises. Cependant, ces mécanismes ne garantissent pas la flexibilité souhaitée. En effet, cette approche est appropriée quand le nombre de partenaires est réduit. Néanmoins, une limite du nombre de partenaires ne peut pas être maîtrisée dans le cas de coopération à la demande. De plus, la mise en place de plusieurs intergiciels orientés messages distribués est souvent considérée comme une tâche coûteuse et compliquée.
- Les mécanismes de coordination transactionnelle, plus particulièrement les Workflows ont démontré un fort potentiel pour la coordination des processus internes d'une entreprise. Cependant, ils se sont avérés mal adaptés à la coopération interentreprises (hétérogénéité, coopération planifiée).
- Les technologies (services Web et ESB) sont basées sur les apports du langage XML (structuration et échange des données) et des Workflows (gestion de processus). Les services Web présentent un atout considérable pour résoudre les problèmes d'intégration des applications en général et des processus interentreprises en particulier [Papazoglou, 2007], [Grefen et al., 2009],[Arsanjani, 2004]. Cette constatation partagée entre plusieurs auteurs nous conforte dans notre orientation de recherche : coopération interentreprises basée sur la composition des services. Cependant, à l'état actuel, les services sont souvent considérés comme une solution qui n'est pas totalement mûre en particulier en ce qui concerne le manque de dynamisme des services. En effet, une entreprise qui fournit un service, décide durant la phase de son développement, du contrat d'utilisation de ce service. Ceci limite considérablement la capacité d'une autre entreprise utilisatrice de ce service à modifier ou à négocier son contrat. De la même manière, si un fournisseur veut fournir un contrat plus restrictif ou plus riche, il faut qu'il publie un nouveau service. Nous tentons dans le chapitre 4 de ce manuscrit de thèse de répondre à cette limite.

1.1.6 Conclusion

Cette première partie de l'état de l'art a répondu à deux objectifs de base : le premier consiste à tracer l'espace dans lequel évolue notre travail de recherche (la coopération à la demande) et le deuxième consiste à examiner les différentes vues de cette coopération (vue métier et vue technique). Cette dichotomie entre le métier et la technique permet de mieux cerner le concept de coopération interentreprises à la demande.

Les travaux de recherche dans ce domaine ont atteint un certain niveau de maturité qui laisse penser que plusieurs entreprises vont essayer de migrer vers ce type d'organisation dans le futur. Bien que le choix de la forme ou de la méthode de coopération dépend du contexte d'activité de l'entreprise, il est primordial de développer certains aspects critiques pour assurer une participation efficace à des scénarios de coopération. D'une part, nous avons la modélisation des processus métiers qui permettra de bien comprendre le fonctionnement de l'entreprise, l'optimiser et faciliter l'interconnexion entre entreprises. D'autre part, le choix des éléments technologiques qui vont assurer le niveau opérationnel de cette interconnexion de façon fiable.

La vue métier a exploré les concepts de processus coopératif et des formalismes de modélisation. Ces formalismes sont différents et chacun d'entre eux présente un certain nombre d'avantages. Cependant, le formalisme BPMN semble s'imposer comme outil de modélisation pertinent tant au niveau micro (processus intra entreprise) qu'au niveau macro (processus interentreprises). Outre ses capacités de modélisation, BPMN possède l'avantage d'être extensible et transposable vers des langages d'exécution comme BPEL.

Quant à la vue technique, elle a présenté l'ensemble des technologies et des standards qui ont facilité la coopération interentreprises. La comparaison entre les différents mécanismes a révélé que la technologie service Web présente une piste prometteuse pour l'interconnexion des processus interentreprises. Cette constatation affirme notre direction de recherche : coopération interentreprises basée sur la composition des services.

La deuxième section de l'état de l'art explore en détail le concept de service et ses avantages pour la coopération à la demande.

1.2 Approche service pour la coopération

La première partie de l'état de l'art a souligné l'importance capitale de la notion de service comme réponse crédible aux besoins de la coopération interentreprises à la demande. Ce concept a le vent en poupe et ne cesse d'attirer les entreprises industrielles. Ces dernières commencent à parler d'une économie de services dans laquelle les entreprises manufacturières offrent leurs produits sous forme de services [Brown, 2008], [Spohrer et al., 2007], [Bitner and Brown, 2008].

Le concept de service a trouvé son essor grâce à la technologie service Web. Cette deuxième partie de l'état de l'art est consacrée à l'analyse du service au profit de la coopération interentreprises. À cet égard, nous proposons un cadre quadridimensionnel afin d'analyser les différentes perspectives (ou vues) relatives au service. Chacune des vues explore des

caractéristiques d'un service et présente les différentes approches et technologies qui lui sont relatives.

1.2.1 Cadre de référence pour les services

Dans le domaine des technologies de l'information, un « buzzword » (mot à la mode) fait référence à un néologisme et aux nouveaux concepts associés [BearingPoint et al., 2003], dans le palmarès des « buzzword », le service est le plus utilisé. Face à cette utilisation accrue du concept, plusieurs auteurs ont tenté de le définir. Nous avons essayé de catégoriser ces définitions suivant trois visions : la vision métier, la vision informatique et la vision interactionnelle.

Une définition métier du concept service est énoncée par [Han et al., 2009] qui considère un service comme une abstraction du niveau métier implémentant les processus métiers de l'entreprise. [Cauvet and Guzelian, 2008] définissent le service métier comme une unité réutilisable qui encapsule un ou plusieurs fragments d'un processus métier et qui vise à satisfaire des buts métiers. Une autre définition métier est celle proposée par [Pallos, 2001], qui définit un service comme un groupement logique de composants requis pour satisfaire une demande métier.

En ce qui concerne la vision informatique, [OASIS, 2006] définit un service comme un mécanisme qui permet l'accès à un ou plusieurs applications et dont l'accès est décrit grâce à une interface. Dans la même direction, un service est défini dans [OGSA, 2005] comme un composant logiciel participant à une architecture orientée services qui fournit des fonctionnalités et / ou participe à la réalisation d'une ou de plusieurs capacités.

D'un point de vue interactif, un service est caractérisé par [Chevrin, 2006] comme une collection de tâches utilisateur. Ces tâches sont regroupées de manière à permettre l'exécution de l'activité requise par un utilisateur.

Certains auteurs comme [Rolland and Kaabi, 2007] présentent une nouvelle vision pour définir un service à savoir la vision intentionnelle. En effet, les auteurs qualifient un service par un service intentionnel s'il permet d'atteindre l'intention ou le but d'un acteur dans un contexte donné.

La multitude des définitions proposées dans la littérature montre d'une part que le concept de service est un concept en plein essor et d'autre part, que sa définition dépend fortement de la vision de celui qui l'énonce. Dans notre travail de recherche nous adoptons à la fois la vision métier et la vision informatique du concept de service. Nous détaillons notre vision de service dans le chapitre 2.

Afin de caractériser la notion de service et de présenter par la suite les différents travaux et propositions qui lui sont relatifs, il nous a paru intéressent de proposer un cadre de référence dont l'objectif est d'analyser le concept de service selon plusieurs perspectives. Chaque vue ou perspective se focalise sur un aspect particulier et répond à des préoccupations qui lui appartiennent. En considérant le concept de service, on se rend compte qu'il se base sur une architecture de référence, il s'appuie sur une pile de standards et s'inscrit dans une démarche d'ingénierie (méthode) qui assure sa mise en place au sein de l'entreprise. De plus, le service

présente la capacité d'être composé avec d'autres services donnant naissance à un service appelé service composite. Ces considérations nous poussent à s'intéresser à quatre vues complémentaires à savoir (voir Figure 1.8) : (i) la vue architecture, (ii) la vue technologique (ii) la vue méthode et (iiii) la vue composition. Nous détaillons ces différentes vues dans ce qui suit.

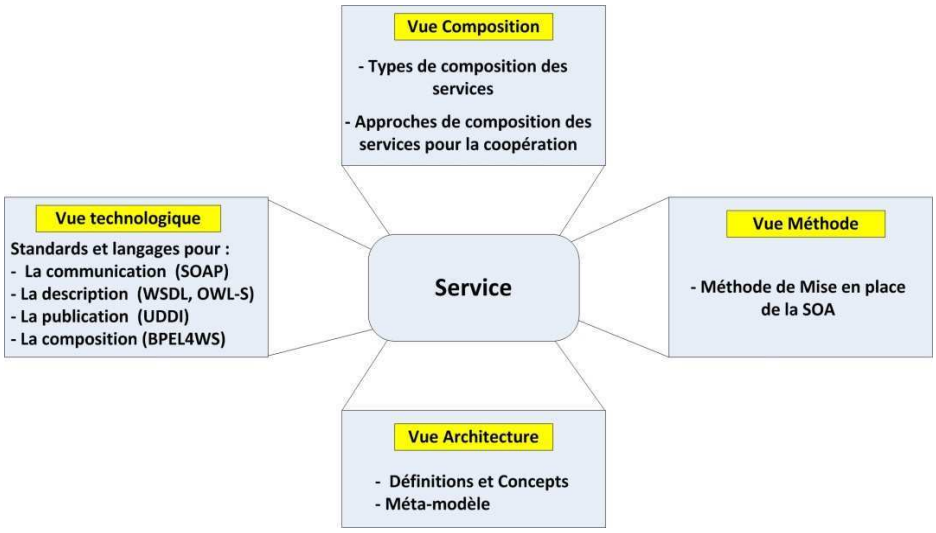


Figure 1.8. Cadre quadridimensionnel pour l'analyse bibliographique liée au service

1.2.2 Vue architecture

Le concept de service se base sur l'architecture orientée services. Il existe plusieurs façons de percevoir et de définir une architecture orientée services. La plupart de ces définitions se concentrent essentiellement sur l'aspect technique de la SOA. Seules quelques définitions intègrent des considérations métier. Nous allons essayer dans ce qui suit de recenser les définitions proposées dans la littérature qu'elle soit industrielle ou scientifique. Ces définitions sont intéressantes puisqu'elles illustrent plusieurs points de vue sur la SOA.

Définitions	Concepts introduits
SOA est un style architectural dont l'objectif est de réaliser le couplage lâche entre un ensemble d'agents logiciels. Un service est une unité de travail effectuée par un prestataire de services, qui vise à atteindre un ensemble de buts [He, 2003]	Style architectural
	Prestataire de services
SOA est un paradigme pour l'organisation et l'utilisation de services distribués. SOA permet d'offrir, de découvrir, d'interagir avec les services [OASIS, 2006]	Utilisation des services
	Découverte des services
	Interaction avec les services
SOA est un ensemble de composants qui peuvent être invoqués, et dont les descriptions d'interface peuvent être publiées et découvertes [W3C, 2004b]	Invocation à distance
	Publication des services
	Découverte des services
SOA permet de définir des politiques, des pratiques, des cadres qui guident l'encapsulation des fonctionnalités proposées par les applications en un ensemble de services. Les services identifiés doivent avoir un certain niveau de granularité qui favorise leur	Service encapsule une fonctionnalité
	Niveau de granularité pertinent

publication et leur découverte [Sprott and Wilkes, 2004]	Publication et découverte de service
SOA est une architecture logicielle basée sur les concepts suivants : service, registre de service et bus de services. Un service se compose d'un contrat, d'une ou de plusieurs interfaces, et d'une implémentation [Krafzig et al., 2004]	Registre de service
	Bus de services
	Interface de services
SOA est un style architectural qui se base sur les concepts de processus métiers et soutient l'orchestration d'un ensemble de services métier. SOA est aussi un ensemble de principes et des modèles qui favorisent des caractéristiques comme la modularité, l'encapsulation, le couplage faible, la séparation des préoccupations, la réutilisation, *etc.* [Lublinsky and Tyomkin, 2003]	Orchestration des services métier
	Modularité
	Encapsulation
	Couplage faible
SOA assure la flexibilité des processus métiers en adoptant une vision service. Les services au sein de la SOA peuvent être réutilisés et combinés pour répondre aux besoins métier [Bieberstein et al., 2005]	Flexibilité
	Réutilisation des services
	Composition des services
SOA est une approche prometteuse pour la conception architecturale où les capacités logicielles sont exposées sous formes d'un ensemble de services. SOA favorise l'alignement métier-IT, l'amélioration de l'agilité des entreprises et la réduction des coûts d'intégration [Arsanjani, 2004]	Alignement métier-IT
	Améliorer l'agilité
	Réduction des coûts d'intégration
SOA est une méthodologie de conception visant à maximiser la réutilisation des services et à accroître l'adaptabilité et l'efficacité du système d'information [BEA, 2006]	Maximiser la réutilisation
	Adaptabilité et efficacité du système d'information
SOA vise à fournir un ensemble de lignes directrices, des principes et des techniques qui guident la réorganisation des processus métiers, des informations et des entités de l'entreprise. L'objectif de la SOA est de soutenir les plans stratégiques et les niveaux de productivité exigés par l'environnement concurrentiel des entreprises [Papazoglou and Georgakopoulos, 2003]	Ensemble de lignes directrices, de principes et de techniques
	Réorganisation des processus métiers, des informations et des entités de l'entreprise

Tableau 1.5. Différentes définitions de l'architecture orientée services

En considérant ces définitions, nous pouvons conclure que tous les auteurs affirment que l'architecture orientée services est un style architectural pour la réorganisation et le redéploiement du système d'information. En effet, l'architecture orientée services permet d'encapsuler les fonctionnalités contenues dans un système d'information en un ensemble de services faiblement couplés. Les services, munis d'un contrat d'utilisation et d'une interface de description, seront publiés pour qu'ils puissent être invoqués par des clients distants.

Plusieurs fonctionnalités supportées par l'architecture orientée services ont été introduites dans les définitions comme les fonctionnalités de publication de services, de découverte et aussi d'interaction avec les services. En outre, trois rôles clés sont distingués : (i) le fournisseur de services, (ii) le client du service et (iii) le registre de service. L'interaction entre ces trois rôles est décrite dans la Figure 1.9. Le fournisseur de services crée le service, puis publie sa description dans un registre de service. Cette description précise à la fois les opérations disponibles et leur mode d'invocation. Le client accède au registre pour effectuer

des recherches afin de trouver les services désirés. Ensuite, une liaison s'établit entre le client et le fournisseur de service afin d'assurer l'invocation du service en question.

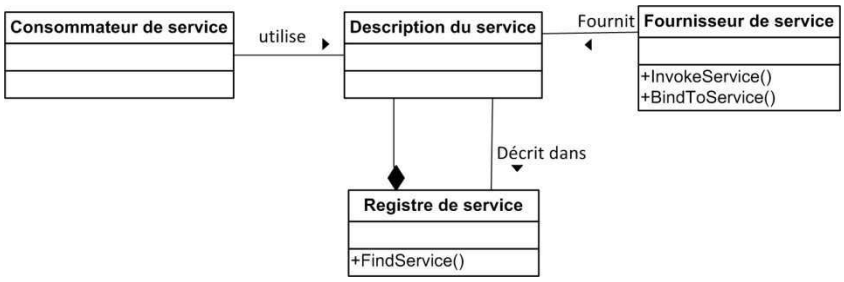


Figure 1.9. Méta-modèle de l'architecture orientée services

1.2.3 Vue technologique

La vue technologique décrit la pile de standards et langages sur lesquels s'appuient les services (voir Figure 1.10). Comme nous l'avons déjà mentionné, l'instanciation la plus importante de l'architecture orientée services dans le monde industriel est la technologie services Web. Dans la suite, nous allons décrire l'ensemble des standards et langages relatifs à cette technologie et qui permettent :

- La communication entre les services (SOAP),
- La description des services (WSDL, OWL-S),
- La publication des services (UDDI),
- La composition des services (BPEL4WS).

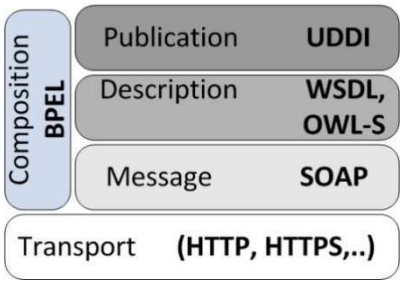


Figure 1.10. Pile de standards et langages des services Web

1.2.3.1 Communication entre les services

SOAP (*Simple Object Access Protocol*) est un standard du consortium W3C définissant un protocole pour la communication et le dialogue avec les services Web. Le standard SOAP est une surcouche de la couche application du modèle OSI des réseaux. Le protocole applicatif le plus utilisé pour transmettre les messages SOAP est HTTP, mais il est également possible d'utiliser les protocoles SMTP ou FTP : la norme n'impose pas de choix.

La structure d'un message SOAP se divise en trois parties (voir Figure 1.11).

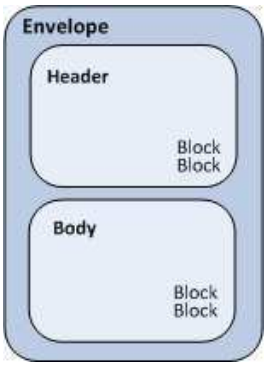


Figure 1.11. Structure d'un message SOAP [Newcomer, 2002]

- L'enveloppe SOAP (*envelope*) : marque le début et la fin du message et elle est subdivisée en deux sous parties : la partie en-tête et la partie corps du message.

- L'en-tête (*SOAP Header*) : permet d'ajouter à un message SOAP des attributs spécifiques qui ne sont pas acceptés par toutes les parties et qui peuvent donc être « négociés » au cas par cas.

- Le corps (SOAP *Body*) : permet de décrire les données spécifiques à l'application. Il s'agit essentiellement du nom de la procédure et des valeurs des paramètres d'appel dans le cas d'une demande de service ou des valeurs des paramètres de retour après l'exécution du service.

1.2.3.2 Description des services

- **WSDL : le standard de description des services Web**

La description de l'interface publique d'un service Web est effectuée par le *Web Service Definition Language* (WSDL)[5]. Proposé initialement par IBM et Microsoft, WSDL représente une grammaire commune (en format XML) pour la description des services. La description d'un service doit inclure la définition des composants nécessaires au protocole de communication et à l'interaction avec un client ou un autre service Web. WSDL distingue deux niveaux d'abstraction nommés abstrait et concret. Le *niveau abstrait* rassemble les informations pouvant être réutilisées, tandis que le niveau *concret* regroupe la description des protocoles d'accès au service Web.

— Le niveau abstrait : décrit les informations propres aux méthodes proposées par le service, ainsi que les informations traitant des messages et des données échangés lors de l'invocation du service. Si deux services proposent les mêmes méthodes, le niveau

[5] W3C (World Wide Web Consortium), Web services description language (WSDL), http: //www.w3.org/TR/wsdl, 2003.

abstrait de description WSDL peut être réutilisé. Ce niveau est composé des informations suivantes :

- Les types de données : le document WSDL permet de décrire les types de données échangées grâce à des schémas XML,
- Les messages : un message décrit les données échangées en fonction des méthodes invoquées (paramètres d'une invocation, valeur de retour, *etc.*),
- Les opérations : une opération représente une abstraction décrivant une action implémentée par le service Web. Chaque opération est identifiée par son nom.

— Le niveau concret décrit la manière dont le client accède au service. Ce niveau est composé des informations suivantes :

- Le protocole de communication : permet de préciser le protocole à utiliser pour l'appel des méthodes du service.
- Les ports d'accès au service : l'accès au service est défini par une collection de ports d'accès. Chaque port représente la localisation du service (*i.e.* son URL).

Les informations contenues dans WSDL constituent la description fonctionnelle du service. Avec WSDL, le client peut savoir ce que le service sait faire, sa localisation et les méthodes de son invocation. Cependant, une définition de service en WSDL comporte très peu de sémantique. Les informations ainsi décrites (par exemple, format des messages, types des données et protocoles de communication) ne sont pas suffisant pour permettre aux services Web d'interagir de manière claire et non ambiguë. Par exemple, deux descriptions XML identiques peuvent avoir des significations différentes selon le contexte [Paolucci et al., 2002]. L'absence d'une sémantique explicite limite les possibilités d'automatisation des services Web [Fensel et al., 2002] [Benatallah et al., 2003a]. Pour pallier cette limitation, des travaux, tels que [Sycara et al., 2003], [Roman et al., 2005], proposent de représenter de manière sémantique des services Web (autrement dit proposent de décrire des services Web sémantiques). Malgré l'abondance de ce type de travaux, aucun ne s'est imposé comme une solution de description de services Web sémantiques. Nous avons choisi d'aborder dans la suite le langage OWL-S, vu qu'il présente le plus de maturité, de généricité et de standardisation par rapport aux autres approches [Martin et al., 2004].

- **OWL-S : description sémantique des services**

OWL-S, anciennement DAML-S est une ontologie pour la description sémantique des services Web qui a été développée dans le cadre du projet DAML. OWL-S se base sur le langage standard OWL [W3C, 2004a] qui permet de représenter des ontologies de façon standardisées sur le Web. L'objectif d'OWL-S est de formaliser de façon non ambiguë les services Web de manière à ce qu'un agent logiciel puisse exploiter automatiquement les informations concernant ces services. Un des avantages de l'utilisation de l'ontologie de services consiste sans doute à l'amélioration de la découverte des services et également leur composition. OWL-S distingue trois éléments principaux :

— *ServiceProfile* : exprime ce que le service propose ainsi que les pré requis que son emploi peut imposer. Cet élément contient l'information la plus utile pour la découverte

et la composition des services du fait qu'elle est utilisée à la fois par les fournisseurs de services pour la publication et par les clients pour l'interrogation,

- *ServiceModel* : exprime le fonctionnement du service,
- *ServiceGrounding* : exprime comment le service peut être utilisé.

Certains travaux comme ceux présentés dans [Izza, 2006] permettent d'enrichir la sémantique générique de OWL-S pour décrire les spécificités des services d'entreprise appelés aussi services fondamentaux. Ces services permettent d'exposer les composants du système d'information de l'entreprise.

L'extension (OWL-S+) proposé par [Izza et al., 2005] concerne le profil de service fondamental qui permet de publier la sémantique spécifique des services fondamentaux. Il n'est autre qu'une spécialisation du concept *Profile* de l'ontologie générique *ServiceProfile* d'OWL-S.

Dans leur approche, [Izza et al., 2005] utilisent toutes les propriétés de *Profile* et notamment les propriétés *servicesClassification*, *serviceCategory* et *serviceParameter*. La propriété *servicesClassification* est spécialisée afin de décrire la classification des services fondamentaux (service métier et service IT). La propriété *serviceCategory* est utilisée pour annoter les services par une taxonomie standard telle que *NAICS*. La dernière propriété est spécialisée afin de décrire les caractéristiques de qualité des services et aussi des caractéristiques comme la vue d'entreprise afin de lier le service avec le composant du système d'information qu'il expose.

- **Synthèse**

Le standard WSDL permet la description de l'interface du service, des détails techniques, du protocole d'accès et des points d'entrée. Une des critiques adressées au standard WSDL est le fait que la description qu'il propose comporte très peu de sémantique. Afin de combler ce manque en termes de représentation de services Web, des propositions issues du Web sémantique ont vu le jour (telles que OWL-S). Cependant, ni le standard WSDL ni le langage OWL-S ne propose de description assez large pour représenter des services adaptés au contexte. Cependant, il est possible de les étendre afin de prendre en compte les notions nécessaires à l'adaptation par exemple, [Qiu et al., 2006] intègrent une représentation de l'utilisateur à OWL-S. Nous revenons en détail sur cet aspect dans la section 3 de ce chapitre.

1.2.3.3 Publication des services (UDDI)

UDDI (*Universal Description, Discovery and Integration*) est un standard né à l'initiative d'un regroupement d'un ensemble d'industriels (Ariba[6], IBM[7], Microsoft[8]). Il constitue le

[6] http://www.ariba.com/
[7] http://www.ibm.com/

registre standard de la technologie des services Web. Les services référencés dans l'UDDI sont accessibles par l'intermédiaire du protocole de communication SOAP. Les entreprises publient les descriptions de leurs services Web dans le registre UDDI sous forme de fichiers WSDL. Les clients peuvent ainsi rechercher plus facilement les services Web dont ils ont besoin en interrogeant le registre UDDI.

Les données enregistrées au sein d'un UDDI sont organisées autour de cinq structures de données principales qui sont (voir Figure 1.12) :

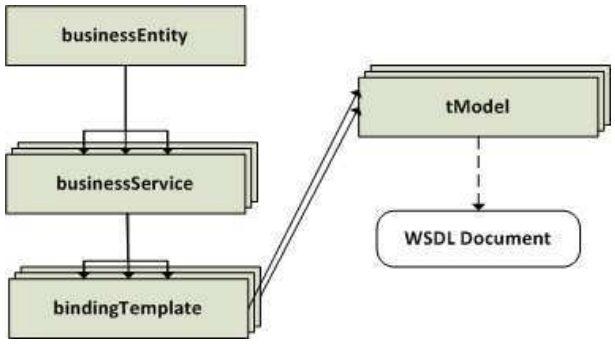


Figure 1.12. Structure de données au sein de l'UDDI

- Le fournisseur (*Business Entity*) : présente les informations concernant le fournisseur de services. Ce composant peut comporter un ou plusieurs *Business Services*,
- Le service (*Business Service*) : représente les services proposés par l'entreprise. La description des services contenue dans l'entité *Business Service* est de haut niveau (aucune information technique n'est décrite ici). Les informations à propos du nom du service et de son objectif sont représentées dans ce composant. Le fournisseur peut rassembler dans cette entité un ensemble de services répondant aux mêmes objectifs dans une même catégorie. Par exemple, une catégorie tourisme peut contenir un service hôtel et un service localisant les sites touristiques,
- Les accès au service (*Binding Template*) : décrivent les points d'accès aux services Web (URL) et le moyen d'y accéder (les différents protocoles à utiliser) afin d'invoquer les services,
- Le type de service (*tModel*) : comprends des liens vers les informations techniques d'un service (notamment sa description WSDL).

- **Synthèse**

En 2006, le nombre de services Web publiés a atteint le nombre de 50 000. Malgré ce nombre, UDDI n'a jamais atteint son objectif de base : devenir le registre de fait pour les services Web. En outre, d'après [Dovey et al., 2005], les spécifications UDDI souffrent de certaines limitations. D'une part, les API de recherche de services proposées par l'UDDI sont inadéquates pour développer efficacement des recherches de services. En effet, le modèle de recherche énoncé par l'UDDI est pauvre. D'autre part, il s'agit d'une recherche qui porte essentiellement sur l'identifiant, le nom du service ou sur des éléments du document WSDL.

[8] http://research.microsoft.com

Néanmoins, le standard UDDI présente des structures de données intéressantes, et notamment la structure *tModel*. Nous allons utiliser cette structure pour référencer les informations contextuelles d'un service (cf. chapitre 3).

1.2.3.4 Composition des services (BPEL)

Les services Web présentent la possibilité d'être composés donnant ainsi naissance à des services de valeurs ajoutées. Une pléthore de langages de composition, proposés par des industriels, et basés sur XML ont été proposés. Chacun de ces langages répond plus ou moins à un certain nombre de besoins.
Parmi ces langages, nous pouvons citer :
- BPML (*Business Process Modeling Language*) [Intalio and BPMI, 2002], développé en partie par le groupe *Business Process Management Initiative* [BPMI, 2002],
- WSCL (*Web Service Conversation Language*), qui est un langage modélisant les conversations supportées par un service,
- WSFL (*Web Services Flow Language*) [Leymann, 2001], développé par IBM dans le but de rendre possible la description de la composition d'un ensemble de services Web en se basant sur la composition des flots de manière hiérarchique,
- XLANG [Thatte, 2001], langage développé par Microsoft en 2001 pour les besoins de sa plateforme de gestion de processus BizTalk,
- BPEL4WS (*Business Process Execution Language For Web Services*) [Andrews et al., 2003a], développé par un ensemble d'acteurs tel qu'IBM, BEA et Microsoft. BPEL4WS que nous appellerons par la suite BPEL combine la représentation orientée graphe des processus de WSFL et la représentation par structures de contrôle algorithmique des processus de XLANG [Juric, 2006].
Parmi tous ces langages, BPEL est utilisé d'une façon majoritaire par différents acteurs industriels. Issu de la fusion des deux langages : WSFL et XLANG, BPEL constitue aujourd'hui le standard de fait pour la composition des services Web.
Le langage BPEL distingue les processus abstraits des processus exécutables [Juric, 2006] :

- **Le processus abstrait :** ce type de processus spécifie les messages échangés entre les différentes parties sans indiquer le comportement de chacune d'elles.
- **Le processus exécutable :** ce type de processus permet de spécifier l'ordre d'exécution des activités, les partenaires concernés, les messages échangés entre ces partenaires, et les mécanismes d'erreurs et d'exceptions.

Un document BPEL utilise XML pour décrire les multiples aspects d'un processus exécutable et qui sont : les partenaires (*Partners*), les transactions (*Exception handling and transactions*) et les espaces de stockage (*CorrelationSet and Containers*) (voir Figure 1.13).

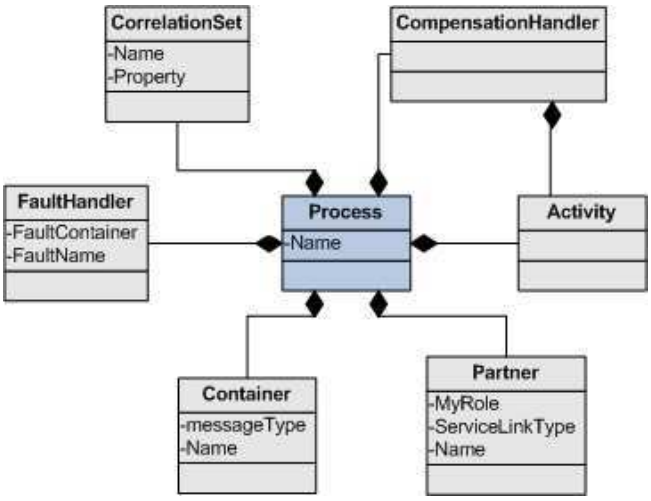


Figure 1.13. Extrait du méta-modèle BPEL4WS

- Les partenaires : sont les différents services Web invoqués dans le processus. Ils ont chacun un rôle spécifique dans un processus donné. Chaque partenaire est décrit par son nom, son rôle (en tant que service indépendant), et son rôle dans le processus,
- Les transactions : sont utilisées dans BPEL afin de gérer les erreurs et les appels à d'autres services si le service appelé est indisponible ou défaillant. Il s'agit essentiellement des *faultHandlers* qui décrivent des routines d'exception et des *compensationHandler* qui proposent des routines de compensation,
- Les espaces de stockage : décrivent les conteneurs de données utilisés par le processus et qui fournissent les définitions des données en termes de types de messages WSDL (*containers*). Un message peut être soit un message d'appel (*invoke*), de réponse (*reply*) ou d'attente (*receive*). Quant au *correlationSets*, ils représentent les dépendances qui peuvent exister entre les invocations de services Web.

Dans la littérature relative à la composition des services Web on parle souvent d'orchestration des services. L'orchestration des services permet de décrire l'enchaînement des services selon un canevas prédéfini, et de les exécuter [Waqar and Racca, 2004]. Généralement on parle d'orchestration quand il y a un processus principal (l'orchestrateur) qui prend le contrôle du déroulement de la composition et coordonne les différentes opérations des différents services. Le standard BPEL est le standard le plus important pour l'orchestration des services Web. En effet, BPEL est le langage le plus accepté par la communauté services Web. Microsoft l'utilise dans sa plateforme de gestion des processus métier appelée BizTalk Server 2009, Oracle l'utilise comme langage d'exécution des processus métiers au sein de son serveur Oracle BPEL, *etc.* Certains auteurs dans la littérature attestent que l'avantage principal de BPEL par rapport à ses prédécesseurs tient au fait qu'il présente une grande expressivité dans la définition du processus exécutable [Wohed et al., 2003]. Cependant, le langage BPEL hérite la vue statique des Workflows et ne permet pas les adaptations du comportement du processus exécutable. Nous revenons sur cette limite dans le chapitre 4 de ce manuscrit de thèse.

1.2.4 Vue méthode de mise en place de la SOA

Le dictionnaire Larousse définit le mot méthode comme « une manière de mener, selon une démarche raisonnée, une action, un travail, une activité, technique ». La vue méthode pour l'approche service s'intéresse essentiellement à la mise en place d'une SOA au sein de l'entreprise. Ce champ d'études correspond à notre première sous problématique à savoir la migration vers une architecture orientée services.

1.2.4.1 Pré requis pour la construction d'une SOA

L'objectif global poursuivi par les méthodes de mise en place de la SOA est de construire un système d'information en relation directe avec les besoins de l'entreprise et répondant par là même aux exigences du marché et de la clientèle. La mise en place de tel projet est généralement réalisée dans le même esprit que les démarches d'ingénierie d'entreprise ou de son système d'information.

L'ingénierie du système d'information de l'entreprise a pour objet la construction de modèles d'une partie déterminée d'une entreprise pour en expliquer la structure et le fonctionnement ou pour analyser le comportement [Vernadat and Hamaidi, 1998]. Dans une démarche d'ingénierie de l'entreprise et de son système d'information, une méthode de mise en place de la SOA propose tout d'abord de représenter les processus de l'entreprise, les analyser, les améliorer et capitaliser le métier ainsi que le patrimoine applicatif de l'entreprise.

La notion de modèle est une composante principale dans une méthode de mise en place d'une SOA au sein de l'entreprise. Avant d'aborder les méthodes proposées dans la littérature, il nous semble important de définir tout d'abord le concept de modèle ainsi que les langages de modélisation les plus appropriés pour une conception orientée services du système d'information de l'entreprise.

1.2.4.1.1 Notion de modèle et de méta-modèle

Selon [Muller and Gaertner, 2004], un modèle « est une abstraction d'un système physique qui distingue ce qui est pertinent de ce qui ne l'est pas dans le but de simplifier la réalité. Un modèle contient tous les éléments nécessaires à la représentation d'un système réel ». Une autre définition est proposée par [OMG, 2003b] qui stipule qu'un modèle est « une représentation d'une partie de la fonctionnalité, de la structure et/ou du comportement d'un système ».

Une définition qui favorise l'automatisation des modèles est présentée par [Kleppe et al., 2003] qui considèrent un modèle comme « une description d'un (ou d'une partie d'un) système dans un langage bien défini, c'est-à-dire respectant un format précis (une syntaxe) et une signification (une sémantique). Cette description doit être convenable pour une interprétation automatisée par un ordinateur ».

Des définitions qui soulignent le concept d'abstraction sont proposées par [Frankel, 2003] : un modèle est « une abstraction d'un système ». Une abstraction est « une description de quelque chose qui omet certains détails non pertinents pour l'objectif de l'abstraction » [OMG, 2003b].

Dans la littérature, il existe ainsi différentes définitions du concept « modèle ». Cependant, elles présentent la même vision, représentation du réel, simplification, abstraction, et description d'un système. Nous remarquons aussi qu'un modèle est créé avec un but spécifique et dans un contexte particulier. Ainsi, le but et le contexte sont les éléments qui vont diriger les choix faits pendant la création d'un modèle. Un modèle n'est pas une description complète d'un système, mais il doit permettre l'élaboration de raisonnements sur le système étudié dans un contexte donné [Cook, 2004].

La création de modèles est réalisée en utilisant un langage bien défini avec une syntaxe et une sémantique spécifiées pour réglementer la création des éléments et leurs relations. Ainsi, un modèle peut être par exemple conçu à travers un langage mathématique ou un langage graphique. Un langage de modélisation est une spécification formelle bien définie qui contient les éléments de base pour construire des modèles. Le langage conçu pour créer des modèles est souvent défini comme un méta-modèle. Un méta-modèle a pour but de permettre à l'ensemble des utilisateurs d'un modèle de se mettre d'accord sur la compréhension et l'utilisation des mêmes termes. Certains auteurs comme [Favre, 2004] définissent un méta-modèle comme « un modèle d'un langage de modèles ». Un méta-modèle utilise d'autres langages connus comme langages de méta-modélisation. Chaque langage de méta-modélisation correspond à un méta-méta-modèle. Un méta-méta-modèle « est un modèle qui définit le langage pour exprimer un méta-modèle. La relation entre un méta-méta-modèle et un méta-modèle est analogue à la relation entre un méta-modèle et un modèle » [OMG, 2003b].

Nous considérons que la réussite de la mise en place de la SOA au sein de l'entreprise repose sur l'élaboration des modèles qui serviront comme support pour toute la démarche SOA. Ces modèles permettent de maîtriser la complexité croissante du système d'information de l'entreprise et de faciliter la conduite vers un système d'information basé sur le concept de service.

1.2.4.1.2 Modèle, langage et démarche de modélisation

Étant donnée l'importance de la modélisation dans le contexte de la SOA, nous présenterons dans ce qui suit différents modèles qui nous serviront dans le cadre de notre démarche de construction de l'architecture de services qui sera présentée dans le chapitre 2 de ce manuscrit de thèse. Ces modèles sont : le modèle de motivation métier [OMG, 2005], le langage UML [OMG, 2007] et la démarche MDA [OMG, 2003b].

- **Modèle de motivations métier (BMM : *Business Motivation Model*)**

Depuis quelques années, de nombreux chercheurs s'intéressent à l'alignement des technologies de l'information à la stratégie et aux buts métier de l'entreprise. Plusieurs raisons expliquent cet intérêt croissant. D'une part, nombreux sont les projets qui échouent parce que le système n'est pas conforme aux besoins des utilisateurs [META, 2003], d'autre part, un mauvais alignement entre le système et la stratégie métier de l'entreprise engendre une baisse de la performance de l'organisation [Sabherwal and Chan, 2001]. Les travaux sur l'alignement sont très variés. Ils portent sur l'alignement entre différentes entités qui peuvent être les stratégies d'entreprise, les stratégies relatives aux technologies de l'information,

l'architecture, les processus d'entreprise, *etc.* Dans le cadre d'une mise en place d'une SOA, il s'agit essentiellement d'un alignement entre les exigences, les besoins métier et l'architecture orientée services du système d'information.

Certes, la préoccupation d'alignement constitue un point important pour garantir la réussite d'une démarche SOA, mais nous n'allons pas proposer une exploration approfondie sur les outils, méthodes et approches d'alignement. Nous allons nous contenter essentiellement de présenter un des modèles les plus utilisés, appelé modèle de motivation métier, pour la spécification des besoins métier. Ces besoins métier une fois capitalisés pourront servir pour identifier et mettre en place les services d'entreprise.

Le modèle de motivations métier appelé en anglais *Business Motivation Model* (*BMM*) fournit une structure organisée pour la spécification, la communication et la gestion des plans métier. Il permet notamment d'identifier les facteurs qui motivent la mise en place des plans métier et retrouver les éléments clés de ces plans. En outre, il précise les relations qui peuvent exister entre les facteurs et les éléments des plans métier.
Il existe trois briques de base pour le modèle de motivation métier (Figure 1.14) : les finalités (*Ends*), les moyens (*Means*) et les facteurs d'impact (*Influencers*).

- Les finalités représentent ce que l'entreprise souhaite atteindre comme (i) une vision, qui correspond à une image globale de ce que l'entreprise veut être ou devenir ; (ii) un but, c'est-à-dire un état de l'entreprise qui correspond à la vision ; (iii) un objectif, c'est-à-dire une cible mesurable que l'entreprise cherche à atteindre pour satisfaire un but.
- Les moyens représentent ce que l'entreprise utilise pour atteindre les finalités comme (i) une mission qui est l'activité réalisée pour atteindre une vision ; (ii) une stratégie, c'est-à-dire un composant de la mission qui permet d'atteindre un but. Une stratégie s'associe au but auquel elle s'applique. Elle a pour objectif principal d'extérioriser la façon d'atteindre ce but puisqu'elle permet de distinguer le but et la façon de le réaliser ; (iii) une tactique, qui aide à implémenter une stratégie et permet de réaliser un objectif.
- Les facteurs d'impact s'inspirent fortement de la méthode SWOT [Martinet 1990]. En effet, ils consistent à l'analyse des forces et des faiblesses de l'entreprise ainsi que des opportunités et des menaces qui peuvent se présenter.

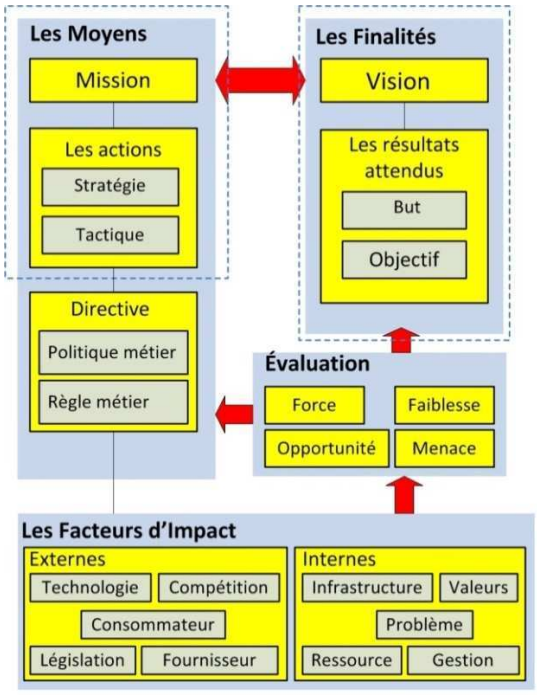


Figure 1.14. Aperçu du modèle de motivation métier (BMM) [OMG, 2005]

- **Langage UML**

Le langage UML, dont certains aspects ont été déjà présentés (cf.section 1.1.4.3), peut présenter un atout de modélisation pour la conception des architectures de services. Plusieurs auteurs affirment cette assertion et considèrent UML comme une boite à outils auxquels il faut cependant ajouter une démarche guidant la construction du système à chaque étape [Zhang, 2008], [Amir and Zeid, 2004]. Ces raisons nous ont poussé à détailler le langage de modélisation unifié UML et son utilisation dans les méthodes de mise en place d'une SOA.

UML propose un ensemble de diagrammes ayant chacun une fonction précise et correspond à un point de vue particulier de modélisation. Chaque type de diagramme UML possède une structure (formalisme) et véhicule une sémantique précise. UML distingue des diagrammes qui reflètent les caractéristiques statiques de ceux qui reflètent les caractéristiques dynamiques d'un système.

UML est un langage général qui a été conçu pour prendre en charge une grande variété de contextes. Cependant, même avec cette intention d'être général, UML ne peut pas couvrir tous les contextes et offre ainsi un mécanisme d'extensibilité basé sur les profils. Un profil permet la personnalisation d'UML pour prendre en charge des domaines spécifiques. Le profil est constitué de stéréotypes, de contraintes et de valeurs marquées (*taggedvalue*). Les contraintes peuvent être spécifiées en utilisant le langage naturel, mais l'utilisation d'OCL (*Object Constraint Language*) [OMG, 2006b] est préférable pour créer les contraintes de façon formelle et standardisée. Les propriétés supplémentaires sont définies par des valeurs marquées. L'OMG a standardisé certains profils comme le profil EAI (*Enterprise Application Integration*) ou encore le profil *Data Distribution*, *etc.*

L'utilisation des diagrammes UML reste indispensable pour accompagner la mise en place de la SOA. D'ailleurs, plusieurs acteurs appartenant à diverses entreprises de conseil française se sont associés en vue d'élaborer une méthode publique, nommée *Praxeme* [PRAXEME, 2008]. Cette méthode résulte de cette mutualisation des investissements et se base essentiellement sur le langage UML pour accompagner la démarche SOA au sein de l'entreprise. La société IBM Rational a proposé un profil UML pour la SOA. Ce travail a été relayé par l'OMG, dans le cadre de son projet UPMS (*UML Profile and Metamodel for SOA*). D'autres travaux académiques ont tenté de proposer des profils UML pour les services. Ces travaux s'intéressent à la modélisation des services informatiques comme [Heckel et al., 2003], [Zhang, 2008], [Amir and Zeid, 2004].

Nous avons choisi le langage UML comme un langage support pour notre démarche de mise en place de la SOA qui sera présentée dans le chapitre 2. Les raisons principales de ce choix tiennent au fait que d'une part, le langage UML a montré sa pertinence pour la modélisation des applications basés sur les services et d'autre part, il est bien outillé non seulement pour produire des diagrammes mais également pour étendre la notation (particulièrement grâce au concept de profil).

- **Architecture dirigée par les modèles (MDA)**

Cette section étudie la contribution de l'approche dirigée par les modèles ou approche MDA (*Model-Driven Architecture*) à l'architecture orientée services. Dans la suite, nous allons tout d'abord, définir l'approche MDA. Ensuite, nous discuterons les mécanismes de transformation et de correspondance de modèles qui relèvent du *Model Driven Engineering* (MDE). En effet, l'intérêt pour le MDE a été fortement appuyé, lorsque l'OMG a rendu publique son initiative MDA (qui vise à la définition d'un cadre normatif pour l'IDM). Enfin, nous présenterons l'intérêt de l'approche MDA pour l'architecture orientée services.

– Définition de l'approche MDA

L'approche MDA est une approche proposée par l'OMG en 2000 comme une réponse à l'hétérogénéité des technologies utilisées dans le cadre du génie logiciel. Face aux multitudes des technologies existantes et à venir il devient impératif de conserver dans l'entreprise des compétences diverses et variées pour des raisons de maintenance et d'intégration des applications [Izza, 2006]. La solution que l'approche MDA propose pour surmonter ces problèmes consiste à utiliser des modèles dans toutes les étapes du processus de développement et d'interaction des applications.

[OMG, 2003b] définit MDA comme « une évolution de l'*Object Management Architecture* qui assure l'intégration et l'interopérabilité pour couvrir le cycle de vie d'un système depuis le modèle métier (*business modelling*) et sa conception, jusqu'à la construction de composants, l'assemblage, l'intégration, le déploiement, la gestion et l'évolution ».

Précisément, l'approche MDA propose de se concentrer sur l'élaboration de modèles pour le développement de systèmes. Elle est dirigée par les modèles « parce qu'elle fournit une approche de l'utilisation de modèles pour diriger la compréhension, la conception, la construction, le déploiement, l'opération, la maintenance et la modification des systèmes » [OMG, 2003b]. Ainsi, elle propose de séparer les spécifications fonctionnelles d'un système

des spécifications de son implémentation sur une plateforme donnée, permettant ainsi de définir une architecture indépendante pour pouvoir ensuite la projeter vers différents modèles de plateformes d'exécution.

À ce jour, il n'y a pas une spécification MDA mais un ensemble de standards (UML, XMI, MOF) et des documents qui proposent un support pour l'utilisation de la démarche MDA [OMG, 2006a], [OMG, 2007].

– Principe de transformation de modèle en MDA

L'approche MDA définit quatre types de modèles qui sont donc les CIMs, les PIMs, les PDMs (ou PMs) et les PSMs [Miller and Mukerji, 2003] [Bézivin and Blanc, 2002] :

- CIM (*Computation Independent Model*) : appelé aussi modèle de domaine ou modèle métier. Son objectif est double : aider à la compréhension du problème et montrer le système dans son environnement organisationnel. Les exigences exprimées dans le CIM doivent être traçables dans le PIM et le PSM. Comme exemple de modèle au niveau CIM on peut noter un modèle de processus.
- PIM (*Platform Independent Model*) : comme son nom l'indique, ce modèle ne possède pas de dépendance avec les plateformes techniques. Il représente la logique métier de l'application et peut être réalisé par un architecte spécialisé dans le domaine de l'application. Comme exemple de PIM on peut noter un modèle d'architecture logique.
- PDM (*Platform Description Model*) ou PM (*Platform Model*) : décrit la plateforme sur laquelle le système va être exécuté (modèles de composants à différents niveaux d'abstraction : C#, EJB, EDOC, *etc.*). Dans une démarche MDA, on se base sur les PDMs pour générer les PSMs à partir des PIMs.
- PSM (*Platform Specific Model*) : le PSM sert essentiellement de base à la génération de code exécutable vers les plateformes techniques décrites dans le PDM. Comme exemple de PSM on peut noter un modèle de service Web.

MDA propose un processus qui gère la transformation entre les modèles déjà décrits. Ce processus permet de transformer un modèle en un autre modèle du même système, mais à un niveau différent. La Figure 1.15 résume le principe du processus MDA.

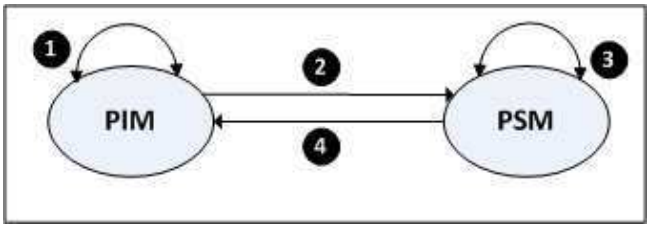


Figure 1.15. Transformations de modèles dans le processus MDA

Précisément, ce processus comporte quatre types de transformations [Miller and Mukerji, 2003] à savoir :

- PIM vers PIM : ces transformations enrichissent les modèles par un ensemble d'informations. De telles transformations ne sont pas toujours automatiques, et

demandent l'intervention du développeur. L'enrichissement de PIM vers PIM en étapes successives vise à le rendre plus complet et plus raffiné.

- PIM vers PSM : ces transformations s'effectuent lorsque les PIMs sont suffisamment enrichis et raffinés pour pouvoir être projetés sur une plateforme technologique. Cette projection est basée sur les caractéristiques de cette plateforme. Un exemple de ce genre de transformation est la transformation d'un modèle métier en UML vers un modèle utilisant une plateforme comme les services Web.
- PSM vers PIM : ces transformations doivent permettre d'obtenir un modèle indépendant à partir d'une implantation existante sur une plateforme spécifique. Plus précisément, ce genre de transformation est utilisé dans les processus de l'ingénierie inversée (*reverse engineering*). Ce sont généralement les transformations les plus difficiles à automatiser.
- PSM vers PSM : ces transformations s'appliquent sur un modèle spécifique et proposent un autre modèle spécifique à la même plateforme. Il s'agit de raffinement de plates-formes technologiques, de déploiement et d'optimisation.

– Concept de correspondance dans MDA

Dans la littérature relative à l'approche MDA on distingue deux concepts de base qui sont souvent confondus et utilisés de manière interchangeable à savoir : le concept de correspondance (*mapping*) et le concept de transformation.

Le *mapping* est définit par [OMG, 2003b] comme « un ensemble de règles et techniques utilisées pour modifier un modèle afin d'obtenir un autre modèle. Les *mappings* sont utilisés pour la transformation de PIM vers PIM, de PIM vers PSM, de PSM vers PSM et de PSM vers PIM ».

Selon [OMG, 2003b], un *mapping* « est spécifié en utilisant un langage permettant de décrire la transformation d'un modèle en un autre modèle. La description peut être en langage naturel, en langage algorithmique, en langage d'action, ou en langage de transformation de modèle ».

Le concept de transformation est introduit pour faire référence à l'action de transformer un modèle en un autre. Ainsi, le terme *mapping* se réfère aux inter-relations entre les éléments d'un modèle (ou méta-modèle) et ceux d'un autre modèle (ou méta-modèle). Alors que la transformation utilise le *mapping* pour créer un modèle cible à partir d'un modèle source. Plusieurs langages de transformation ont été proposés et ne cessent de démontrer la viabilité des concepts autour de MDA, comme ATL (*Atlas Transformation Language*) [Bézivin et al., 2003] UMLx [Willink, 2003] et MOLA (*MOdel transformation LAnguage*) [Kalnins et al., 2004].

– MDA et l'architecture orientée services

Plusieurs travaux ont démontré l'application de l'approche MDA à la technologie service Web [Baresi et al., 2003], [Lopez-Sanza et al., 2008], [Zhang, 2008]. Certains auteurs comme [Lopes, 2005] considère l'approche MDA et les services Web comme la solution indispensable dans l'avenir pour le développement des applications orientées Internet. [Frankel and Parodi, 2002] sont parmi les premiers qui ont discuté l'approche MDA pour

développer des applications ayant les services Web comme plateforme cible. Dans leur travail, les auteurs proposent un ensemble de modèles formels pour supporter l'approche MDA. Associé avec une démarche, l'approche MDA met en correspondance un modèle métier présenté en utilisant le langage UML et les services Web.

D'autres travaux ont été lancés pour étudier la proposition de méta-modèles pour les services Web. [Bordbar and Staikopoulos, 2004a] proposent un méta-modèle pour les services Web, à partir de schémas WSDL ou d'autres spécifications existantes. Dans un autre travail, [Bordbar and Staikopoulos, 2004b] proposent un méta-modèle pour BPEL dans lequel ils spécifient la correspondance entre le diagramme d'activité d'UML et le langage BPEL et définissent des règles de transformation en OCL (*Object Constraint Language*).

[Gronmo et al., 2004] proposent une méthodologie pour développer les services Web et illustrent une correspondance entre un modèle UML et WSDL. [Skogan et al., 2004] proposent les diagrammes d'activités d'UML pour réaliser la composition des services. Le service composé est ensuite exporté comme un document WorkSCo ou BPEL.

Dans les travaux de thèse de [Touzi, 2007], l'auteur part du constat que la nature et la complexité de l'interopérabilité des systèmes d'information hétérogènes obligent à suivre une démarche rigoureuse et structurée. Pour ce faire, l'auteur adopte l'approche orientée services afin de faciliter l'intégration des systèmes d'information des partenaires. Cette intégration se base sur la proposition d'une démarche qui s'inscrit dans l'approche de l'interopérabilité dirigée par les modèles (MDI). La démarche proposée (voir Figure 1.16) 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.

Il s'agit plus précisément d'une descente en abstraction depuis la couche *métier (processus collaboratif)* jusqu'à la couche *logique* concrétisée par la traduction d'un modèle de processus collaboratif exprimé en BPMN en un modèle de système d'information collaboratif exprimé en utilisant le langage UML.

L'identification des règles de correspondance entre les différents éléments des deux méta-modèles constitue également une étape importante dans l'approche proposée. À travers les règles de correspondance, il est possible de passer du niveau métier (processus collaboratif) au niveau logique (système d'information basé sur les services). L'auteur justifie cette orientation d'une part par le rôle central qu'occupent les processus dans la modélisation métier de la collaboration et d'autre part, par l'importance de la conception de système d'information basée sur les services.

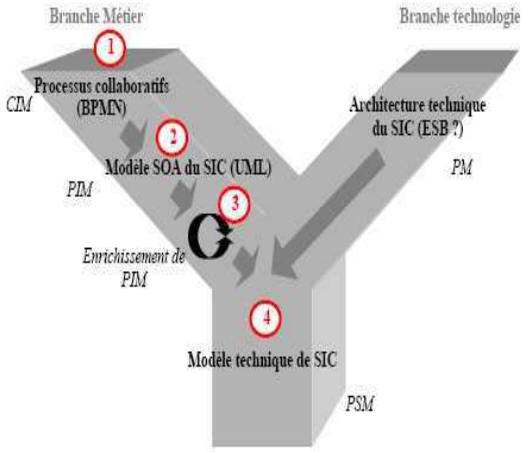


Figure 1.16. Conception du système d'information collaboratif [Touzi, 2007]

L'intérêt grandissant pour l'approche MDA dans le domaine des services Web ou des services d'une façon générale se justifie par le fait que les niveaux proposés par l'approche MDA s'apparentent à toute démarche convenable d'une mise en place d'une SOA. En effet, au niveau métier, il s'agit de déterminer une modélisation qui permet de couvrir les spécificités du métier de l'entreprise (et notamment des modèles de processus métiers). Le niveau logique consiste à spécifier les fonctionnalités (services) mises en oeuvre pour répondre à la stratégie et aux besoins métier. Enfin, le niveau technique résulte d'une projection du niveau logique sur une solution technologique particulière (EAI, ESB, service Web, *etc.*), afin de rendre exécutable l'architecture proposée. Cependant à ce jour il n'existe pas une démarche unifiée qui guide l'utilisation de l'approche MDA lors de la construction de la SOA. Nous nous intéressons dans nos travaux de recherche à inscrire la méthode de construction de services dans une démarche dirigée par les modèles. Ces aspects sont décrits en détail dans le chapitre 2 de ce manuscrit de thèse.

1.2.4.2 Méthodes de mise en place d'une SOA

Quelques méthodes de mise en place d'une SOA ont été présentées. Nous allons exposer dans ce qui suit une revue de la littérature des travaux que nous avons jugés pertinents.

SOMA d'IBM **[Arsanjani, 2004]** est une méthode pour l'identification, la modélisation et la spécification des services. Elle consiste en trois phases :

- **La phase d'identification des services** se base sur trois démarches : (i) une démarche descendante (*Top-Down*) pour la décomposition des domaines, (ii) une démarche ascendante (*Bottom-Up*) pour l'analyse du patrimoine existant et (iii) une démarche hybride (*Middle-out*) pour une modélisation du service selon le but. La démarche descendante offre une cartographie des cas d'utilisation. Tandis que la démarche ascendante, elle analyse les systèmes existants afin d'identifier les services de bas niveau. La démarche hybride consiste en une modélisation de service basé sur les buts et vise à déterminer les services qui n'ont pas été identifiés dans les deux autres démarches,

- **La phase de spécification** commence par filtrer l'ensemble des services candidats en se basant sur des règles bien déterminées à savoir : (i) l'alignement entre fonction et métier (ii) l'élimination des redondances, (iii) la réutilisation des services par les différents processus métiers, et (iv) la facilité de l'implémentation. Dans cette phase, les auteurs proposent d'attribuer une note pour les services sur une échelle de 1 à 5. Les services qui auront les scores les plus élevés seront considérés comme des candidats pour la réalisation,
- **La phase de réalisation** se charge d'implémenter l'ensemble des services.

En outre, [Arsanjani, 2004] propose une architecture comportant cinq couches comme support à la mise en place d'une SOA au sein de l'entreprise. La première couche inclut les systèmes *legacy*, les ERP et les applications Web de l'entreprise. La deuxième propose d'étudier les composants métier responsable de la réalisation des services. Quant à la troisième couche qui est la couche de services, elle montre l'agrégation des composants métiers afin de former les services. La quatrième couche assure la chorégraphie des différents services sous forme de Workflow applicatif afin de réaliser les processus métiers. Finalement, la couche présentation, elle offre la possibilité d'invoquer des services à partir des portails de l'entreprise.

Outre les couches horizontales déjà mentionnées, deux couches verticales font partie de l'architecture proposée. La première couche est la couche d'intégration, qui offre les moyens pour identifier et retrouver les services et les composants et proposer les protocoles essentiels à leurs fonctionnements. La deuxième est la couche qualité de service qui se charge de superviser les différents attributs de la qualité de service.

L'approche SOMA [Arsanjani, 2004] est une approche à la fois descendante et ascendante qui tient compte des besoins métier et de l'existant de l'entreprise. Cependant SOMA ne propose pas de description ouverte de sa méthode ce qui rend difficile l'analyse de ses capacités réelles.

[Papazoglou and Heuvel, 2006] proposent une méthode de développement des services Web. La méthode englobe six phases et se base sur le *Rational Unified Process* (RUP), le développement orienté composant et la modélisation des processus (*BPM* : *Buisness Process Modelling*). Cette méthode propose des directives pour la spécification des interfaces de services Web ainsi que des modèles de flux de services, de manière à maximiser la cohésion et minimiser le couplage. Les six phases identifiées par [Papazoglou and Heuvel, 2006] sont :

- **La phase de planification :** détermine la faisabilité, la nature et la portée de la solution service dans le contexte de l'entreprise. Elle comprend l'analyse des besoins métier, l'étude des technologies actuelles et de leurs capacités. Cette phase comprend aussi une analyse financière et une estimation des coûts et des bénéfices d'un projet de développement d'une SOA.
- **La phase d'analyse et de spécification :** comporte deux étapes

  – Durant la phase d'analyse, les analystes métier préparent les modèles de processus métiers « *as-is* » (ceux qui existent actuellement) et « *to-be* » (ceux qui doivent exister). Cette étape examine l'ensemble des services existants dans le but de savoir quels sont les

services qui sont déjà en place et quels sont ceux qui doivent être développés. Il s'agit principalement de : (i) l'identification des processus, l'identification de la portée des processus, (iii) l'analyse des services existants et des services à obtenir, et (iv) l'analyse de la réalisation des processus.

- L'étape de spécification de service, comme son nom l'indique, se préoccupe de spécifier les services ainsi que les processus métiers (composition des services). La spécification des services se focalise sur la présentation de la structure du service, de son comportement et de ses politiques. En revanche, la spécification des processus métiers comprend la description de la structure des processus ainsi que l'identification des paramètres non fonctionnels de ces processus.

- **La phase de construction et de test :** se concentre sur l'implémentation des services et la définition de leurs interfaces. Par la suite, il s'agit de vérifier la compatibilité des services obtenus par rapport aux objectifs fixés durant la phase d'analyse.
- **La phase de provisionning :** s'intéresse à la gouvernance, la certification, l'enregistrement, l'audit des services afin de contrôler le comportement d'un service durant son utilisation.
- **La phase de déploiement :** assure la publication des interfaces des services.
- **La phase d'exécution et de supervision :** durant cette phase un consommateur de service peut découvrir la définition et invoquer toutes les opérations définies. Il s'agit par la suite d'évaluer le service en considérant ses performances d'exécution.

La méthode proposée par [Papazoglou and Heuvel, 2006] présente un cycle complet pour le développement des services. Néanmoins, peu de détail concernant la méthode a été présenté ce qui peut poser des questions sur sa faisabilité. En outre, les auteurs ne présentent pas une architecture support à leur méthode et la relation entre service et processus métier n'est pas assez détaillée.

SOAD (*Service-Oriented Analysis and Design*) est l'approche proposée par **[Zimmermann et al., 2004]**. Dans cette approche, les auteurs combinent plusieurs techniques à savoir la conception et l'analyse orientée objet (*Object-Oriented Analysis and Design*), l'architecture d'entreprise (*Enterprise Architecture*) et la modélisation des processus d'entreprise (*Business Process Modelling*). Plus précisément, les auteurs soulignent la nécessité de bénéficier des travaux déjà réalisés dans le cadre de la modélisation d'entreprise lors d'une démarche SOA. De la même manière, les auteurs considèrent que les approches de modélisation des processus et la conception orientée objet sont des approches pertinentes qui permettent d'identifier des services suivant deux niveaux d'abstraction (métier et technique).

La Figure 1.17 illustre la cohabitation des différentes techniques au sein de la SOAD.

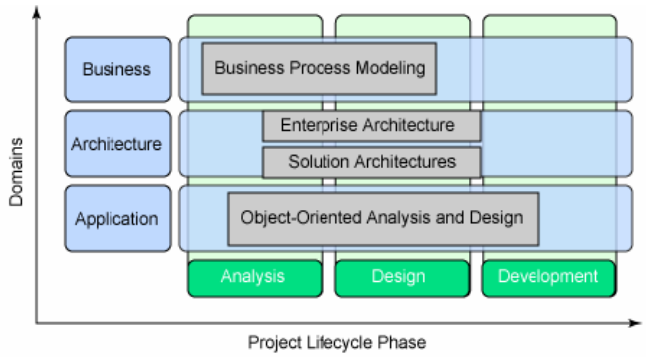


Figure 1.17. Fondements de base de la SOAD [Zimmermann et al., 2004]

En outre, la SOAD englobe un ensemble de concepts qui lui sont spécifiques à savoir : une approche hybride d'identification des services, une typologie de services ainsi que leurs politiques, la médiation sémantique entre les services et la gestion des connaissances autour des services.

Les auteurs proposent d'utiliser une approche hybride pour l'identification des services qui combine à la fois l'analyse des processus métiers afin de définir les services nécessaires à leurs réalisations et l'analyse de l'existant pour déterminer les fonctions existantes du système d'information. Deux types de services d'entreprise sont définis à savoir : les services métier (*business service*) et les services logiciels (*software service*). Ces derniers peuvent être des services atomiques ou des services composites. Par la suite, les services sont décrits grâce à des politiques qui enrichissent la description de leurs interfaces avec des spécifications de qualité (*QoS*).

En quête de flexibilité et d'agilité des relations interentreprises, les auteurs soulignent la nécessité d'enrichir sémantiquement les services d'entreprises afin qu'ils puissent se découvrir dynamiquement et former des processus coopératifs à la volée. Ceci justifie l'utilisation des médiateurs sémantiques qui se chargent de résoudre les conflits sémantiques entre les services appartenant à des entreprises coopérantes.

Un autre élément de base de l'approche SOAD consiste à la gestion des connaissances autour des services d'entreprise. Il s'agit de construire une culture de service qui permet par la suite de juger la pertinence d'un service. En effet, un service qui n'est pas réutilisable est un service non pertinent et ne devra pas exister dans le catalogue de services de l'entreprise.

En résumé, l'approche proposée par [Zimmermann et al., 2004] se focalise sur l'analyse et la conception des services au sein de l'entreprise. L'originalité de leur approche réside dans l'utilisation des techniques déjà existantes comme la conception orientée objet, l'architecture d'entreprise et la modélisation des processus d'entreprise afin de bâtir une approche SOA au sein de l'entreprise.

Un reproche qui peut être adressé à ce travail tient au fait que la SOAD ne propose pas de démarche méthodologique pour l'identification des services. En effet, elle propose un ensemble d'éléments et de concepts qui ressemblent plutôt à des bonnes pratiques qu'à des directives d'aide pour la mise en place d'une SOA.

Emig et al. dans leur approche **[Emig et al., 2006]** partent de la description des processus métiers à l'aide du BPMN. Par la suite, en se basant sur des règles de transformation, le processus métier est automatiquement converti sous forme exécutable et sera décrit avec un langage d'exécution qui est BPEL. Cette approche présente beaucoup de défaillances. En effet, les auteurs ne présentent pas une démarche claire de définition des services. Le fait de passer directement d'une description de processus métier vers un langage exécutable de processus pose un certain nombre de questions sur la nature des services définis, leurs granularités et leurs réutilisations. Cette approche est aussi très rigide étant donné que n'importe quel changement au niveau du processus métier impliquera une régénération du code du processus exécutable. En plus, la mise en œuvre de l'approche proposée dans le cadre d'une entreprise est critique. En effet, il s'agit d'une approche strictement descendante qui ne tient pas compte du patrimoine applicatif existant dans l'entreprise.

Dans leur travail **[Chang and Kim, 2007]** présentent une approche d'analyse et de spécification orientée service pour le développement des services adaptables. Leur approche est composée de six phases. La première est la phase de définition des services dans laquelle les auteurs procèdent à une analyse des services existants et à une identification des processus métiers cibles. La deuxième phase s'intéresse à la définition des services unitaires. Cette définition se fait en trois étapes : (i) identifier les services unitaire à partir des processus métiers définis dans la phase précédente en considérant des caractéristiques comme la cohésion et la réutilisation, (ii) définir les interfaces de chaque service unitaire et (iii) spécifier le modèle structurel et comportemental du service.

Afin d'assurer l'adaptation des services, les auteurs définissent les variabilités de services sous forme de différents comportements ainsi que les situations qui les déclenchent. Quant à la troisième phase, elle définit une typologie de services. Trois types de services ont été identifiés : (i) les services extraits à partir des applications existantes (systèmes *legacy*) de l'entreprise, (ii) les services qui doivent être développés et (iii) les services qui doivent être découverts à partir des registres de service externes. La quatrième phase est la phase d'acquisition (*i.e.* développement des services) qui se base sur la classification faite dans la phase précédente. Pour les services qui vont être découverts, les auteurs identifient plusieurs contraintes comme les contraintes fonctionnelles et non fonctionnelles des services. Pour les services qui doivent être extraits à partir du patrimoine applicatif, les auteurs proposent d'ajouter des façades ou des médiateurs pour encapsuler ces systèmes et obtenir les services. Finalement, la cinquième phase se charge de développer les nouveaux services selon des techniques existant comme la conception orientée objet (*OOAD : Object Oriented Analysis and Design*) ou le développement basé sur les composants (*CBD* : *Component Based Development*). La dernière phase est la phase de composition des services unitaires.

Le point fort de ce travail c'est qu'il considère l'adaptation des services au moment de leur définition. Ceci constitue un point commun avec notre approche qui sera présentée dans le chapitre 2 de ce manuscrit de thèse. Cependant, les auteurs ne montrent pas la façon d'incorporer explicitement cette adaptabilité dans les services. Un autre reproche du travail présenté dans [Chang and Kim, 2007] tient au fait que les services définis sont de très fines granularités. La notion de service métier n'a pas été traitée et uniquement une couche de services techniques est disponible. Ce choix peut être critiqué vu que l'entreprise ne pourra

pas mettre ses services pour une utilisation externe puisque ces derniers sont de très fines granularités et ne possèdent pas une forte valeur métier. En outre, le fait d'avoir des services de ce type rend la composition très complexe puisqu'on sera amené à faire plusieurs compositions de services pour obtenir un processus métier.

Dans son livre intitulé « *Service-oriented Architecture: Concepts, Technology, And Design* », **[Erl, 2005]** présente une méthode de définition des services qui couvre les deux premières étapes du cycle de vie de la construction d'une SOA à savoir : l'identification et la conception des services.

Avant d'aborder l'approche d'identification des services, il est important de faire le point sur la typologie des services proposée par [Erl, 2005]. Trois types de services ont été identifiés à savoir : les services métier, les services applicatifs et les services hybrides. Le service métier encapsule la logique métier de l'entreprise et il est issue de l'analyse de ses différents processus. Deux types de services métier ont été présentés : les services métier orientés tâches (*Task-centric business service*) et les services métier orientés entités (*Entity-centric business service*). Un service métier orienté tâche encapsule la logique métier spécifique à une tâche ou à une activité (ou plusieurs activités) tandis que le service métier orienté entité encapsule la logique de fonctionnement spécifique à une entité bien particulière (telle qu'une facture). Quant au service applicatif, il expose des fonctionnalités de fines granularités, qui sont sollicitées par les services métier. L'auteur propose trois types de services applicatifs : les services d'intégration (pour les besoins d'intégration), les services d'infrastructure et les services *Wrapper* exposant la logique qui réside dans les applications de l'entreprise.

La particularité de l'approche proposée par [Erl, 2005] consiste à la proposition des services hybrides qui encapsulent à la fois la logique métier et la logique applicative. Par contre, l'auteur ne propose aucune démarche permettant de les spécifier.

La Figure 1.18 illustre le processus d'identification des services (applicatifs et métier) proposé par [Erl, 2005]. Il s'agit d'une approche descendante comportant douze phases à réaliser dans l'ordre. Dans ce qui suit, nous allons décrire en détail ces différentes phases.

La première phase consiste à décomposer les processus métiers déjà cartographiés en un ensemble d'étapes (*steps*). Une fois identifiées, il s'agit par la suite d'éliminer les étapes qui ne peuvent pas être encapsulées par des services telles que les étapes réalisées manuellement. Par la suite, la maîtrise d'ouvrage se charge d'identifier les services potentiels en regroupant les étapes qui semblent appartenir à un même contexte. Chaque étape pourra correspondre à une opération au sein d'un service. Par la suite, [Erl, 2005] propose d'identifier les logiques métier issues de l'étude des processus. L'objectif d'une telle identification est de sélectionner les étapes éligibles au rang d'opérations de services.

La phase suivante du processus d'identification propose d'appliquer la logique de la SOA aux services potentiels préalablement définis. L'auteur insiste tout particulièrement sur l'importance d'appliquer deux principes de base de la SOA aux services métier à savoir : la réutilisation et l'autonomie. L'auteur prévoit que l'analyste métier pourra ajouter des opérations à un service particulier afin d'améliorer sa réutilisation future.

La sixième phase du processus d'identification consiste à définir les services composites. Ceci se réalise à travers la construction des différents processus métiers par composition des services. Ainsi, il sera plus facile de détecter le besoin de développer un nouveau service ou encore le besoin d'ajouter une opération à un service. Quant à la septième phase, elle se charge d'étudier les besoins fonctionnels de chaque opération d'un service. Ces besoins seront pris en considération dans la huitième phase qui permet d'identifier les opérations des services applicatifs à partir des besoins d'implémentation exprimés dans la phase précédente. Par la suite, il s'agit de regrouper les opérations appartenant à un même contexte afin d'identifier les services applicatifs. Ces derniers vont être vérifiés dans la phase suivante afin de s'assurer qu'ils sont réutilisables et autonomes.

Les deux dernières phases du processus proposé par [Erl, 2005] consistent à revoir les services déjà identifiés ainsi que leurs opérations.

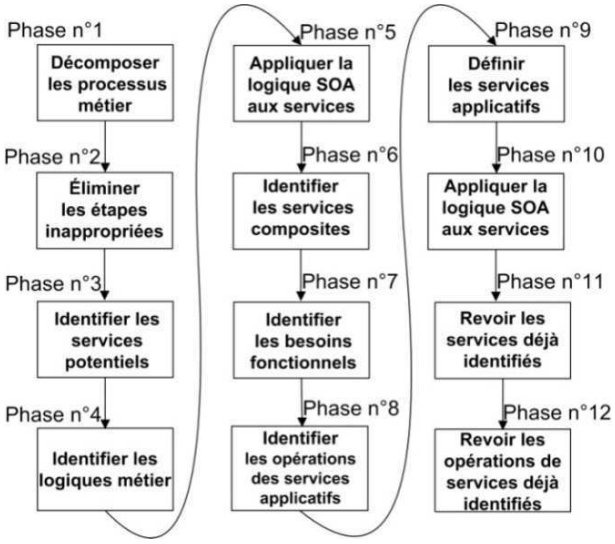


Figure 1.18. Processus d'identification des services (*step-by-step Process*) [Erl, 2005]

Le travail proposé dans [Erl, 2005] traite de l'identification des services d'entreprise. L'approche proposée est une approche assez claire et comporte des exemples qui montrent chaque phase du processus d'identification. D'ailleurs ce point constitue l'un des points fort de l'approche. Cependant, le travail a été testé sur un exemple assez simple qui est un peu loin de la réalité des entreprises. De plus, la typologie de services n'est pas supportée par un méta-modèle qui récapitule l'ensemble des concepts manipulés ainsi que les relations entre eux. Ce qui laisse régner un certain flou concernant la signification de certains types de services (tels que par exemple les services hybrides). En outre, l'approche proposée est une approche descendante qui signifie une refonte de tout ou partie du système d'information de l'entreprise, jugée bien souvent trop coûteuse et trop risquée.

- **Comparaison entre les méthodes de mise en place d'une SOA**

En se basant sur un ensemble de bonnes pratiques pour la mise en place d'une SOA telles que celles présentées dans [Arsanjani, 2004], [Ramollari et al., 2007], [Boukadi et al., 2009d], nous avons synthétisé les caractéristiques des différentes méthodes à l'aide de six critères :

- **Démarche de développement :** il existe trois démarches de développement d'une approche SOA. La première est l'approche descendante (D) dans laquelle la logique métier des processus existants est examinée afin d'identifier les services. À l'inverse, l'approche ascendante (A) débute par l'analyse du patrimoine applicatif afin de déterminer les fonctions existantes du système d'information et tracer ainsi une cartographie applicative. À partir de cette cartographie, il est possible d'identifier les fonctions qui sont éligibles au rang de service. Enfin, l'approche hybride (H) préconise de mener en parallèle une approche descendante et une approche ascendante.
- **Étapes du cycle de vie :** le cycle de vie pour le développement d'une SOA comprend les étapes suivantes : la planification, l'analyse, la spécification, la construction, le test, le déploiement et la gouvernance [Papazoglou and Heuvel, 2006]. Les approches traitées se focalisent soit sur la totalité du cycle de vie soit sur une étape particulière.
- **Degré de clarté :** ce critère permet de tester la qualité de la méthode proposée en termes de clarté de la démarche et d'exemples illustratifs des différentes phases de la méthode. Nous avons choisi de les classer en attribuant une note allant de 1 à 5 afin d'évaluer la clarté de la méthode.
- **Techniques et modèles :** ce critère permet de renseigner si la méthode propose de réutiliser des techniques déjà existantes et/ou elle présente un ensemble de modèles ou de méta-modèles supports. Des exemples de ces techniques et modèles, nous pouvons noter la conception orientée objet (*OOAD : Object Oriented Analysis and Design*) ou le développement basé sur les composants (*CBD* : *Component Based Development*) et les modèles de processus (*BPM : Business Process Modelling*).
- **Prise en compte de la flexibilité :** la flexibilité est une caractéristique importante et largement recherchée par les entreprises. Dans cette perspective, la flexibilité se manifeste à travers la prise en compte de la variabilité (*i.e.*, les variantes d'usage des services) au moment de leur modélisation. Nous avons choisi d'examiner les méthodes proposées par rapport à cette caractéristique.
- **Prise en compte de la coopération interentreprises :** ce critère permet d'indiquer si les méthodes proposées considèrent les spécificités de la coopération interentreprises.

Nous avons voulu résumer, dans le Tableau 1.6, les principales méthodes de mise en place d'une SOA au sein de l'entreprise. Ces méthodes sont synthétisées à l'aide des six critères déjà présentés.

Approche / Critère	SOMA [Arsanjani, 2004]	[Papazoglou and Heuvel, 2006]	SOAD [Zimmermann et al., 2004]	[Emig et al., 2006]	[Chang and Kim, 2007]	[Erl, 2005]
Démarche de développement	H	H	H	D	H	D
Étapes du cycle de vie	Analyse et spécification	Cycle de vie complet	Analyse et spécification	Analyse et spécification	Analyse et spécification	Analyse et spécification
Degré de clarté	2	2	1	1	2	4
Techniques et modèles	?	CBD, BPM	OOAD, BPM	BPM	?	BPM
Flexibilité	Non	Non	Non	Non	Oui Point de variabilité	Non
coopération interentreprises	Oui	Oui	Non	Non	Non	Non

Tableau 1.6. Comparaison des méthodes de mise en place de la SOA

- **Synthèse et positionnement**

L'analyse du Tableau 1.6 permet de dégager un ensemble de constats :

– Peu nombreux sont les travaux de recherche qui se sont intéressés à la problématique de mise en place d'une SOA. Généralement, ce sont les sociétés de conseil qui proposent de définir des lignes directrices et d'énumérer les phases qui permettent de diriger les architectes vers un modèle orienté services. De plus, la plupart des approches présentées n'offrent pas une démarche complète, claire et précise permettant d'atteindre le but d'une manière efficace.

– La majorité des approches de mise en place d'une SOA sont des approches hybrides. Nous considérons que dans le cadre d'une entreprise, l'approche hybride est l'approche la plus réaliste. En effet, il est intéressant de mener en parallèle à la fois une approche descendante et une approche ascendante. La première permet de définir des services de haut niveau nécessaires à la réalisation des processus métiers tandis que la deuxième permet de cartographier l'existant applicatif de l'entreprise afin d'identifier les services de bas niveau (services informatiques).

– À part le travail de [Chang and Kim, 2007], les méthodes de mise en place d'une SOA ne se sont pas intéressées à l'étude de l'adaptation des services au moment de leur modélisation.

La démarche que nous allons proposer dans le chapitre 2 de ce manuscrit de thèse essaye de répondre aux limites déjà énoncées.

1.2.5 Vue de composition des services

Dans cette section, nous allons présenter la dernière vue du concept de service à savoir : la vue de composition (voir Figure 1.8).

Les services ou plus particulièrement les services Web ont l'avantage d'être facilement composables. La composition des services Web a été définie par [Casati and Shan, 2002] comme étant la capacité d'offrir des services à valeur ajoutée en combinant des services existants probablement offerts par différentes organisations. Techniquement parlant, la composition permet d'assembler des services Web afin d'atteindre un objectif particulier, par l'intermédiaire de primitives de contrôles (boucles, test, traitement d'exception, *etc.*) et d'échange (envoi et réception de message) [Kellert and Toumani, 2006]. Il s'agit en d'autre terme de spécifier quels services seront invoqués, dans quel ordre et comment gérer les conditions d'exceptions.

La composition des services Web a plusieurs points communs avec la technologie de Workflow. Les deux visent à spécifier le processus métier par la composition des entités autonomes. Leur différence réside dans la nature de l'entité. Dans le cas de Workflow, les entités sont des applications conventionnelles, dans celui des services Web, les entités sont des services.

La définition la plus référencée dans la littérature est celle énoncée par [Benatallah et al., 2005a]. Les auteurs considèrent la composition des services Web comme étant un moyen efficace pour créer, exécuter, et maintenir des services qui dépendent d'autres services.

Dans la suite, nous allons examiner les différents types de composition existant. Ensuite, nous allons présenter les approches de composition des services Web proposées pour des fins de coopération interentreprises.

1.2.5.1 Types de composition des services

Le problème de composition des services est un sujet d'actualité qui suscite l'intérêt de plusieurs chercheurs et entreprise informatiques. Plusieurs préoccupations ont motivé des chercheurs appartenant à différentes disciplines. Avant de passer en revue les approches de composition qui s'apparentent à notre problématique de recherche, nous allons examiner tout d'abord les types de composition possible.

En considérant le degré d'automatisation de la composition certains auteurs comme [Foster et al., 2003] classent la composition en deux types :

- **La composition manuelle**

La composition manuelle suppose que l'utilisateur génère la composition à la main via un éditeur de texte et sans l'aide d'outils dédiés. Ainsi, l'utilisateur se charge de définir son besoin en termes de composition des services en consultant un registre de service et en se basant essentiellement sur sa connaissance sur le domaine. La composition manuelle est considérée par certains auteurs comme l'un des verrous le plus important au développement des architectures orientées services [Milanovic and Malek, 2004].

- **La composition automatique**

La composition automatique prend en charge tout le processus de composition et le réalise automatiquement, sans qu'aucune intervention de l'utilisateur ne soit requise. Ces dernières années, de nombreux travaux ont porté sur l'automatisation de la composition des services [Milanovic and Malek, 2004], [Singh, 2004], [Bourdon, 2007]. Ils trouvent leur justification dans l'évolution constante de l'offre de services en ligne ainsi que de leurs propriétés non fonctionnelles, ce qui rend une description de la composition difficile à maintenir.

D'autres auteurs comme [Casati and Shan, 2001] considèrent que la composition des services peut être qualifiée de statique et de dynamique :

- **La composition statique**

La composition statique passe par une phase de spécification durant laquelle les services sont identifiés, interconnectés, compilés et déployés pour être utilisés.

- **La composition dynamique**

La composition dynamique prend en compte les services disponibles, leurs fonctionnalités et le but à atteindre que ce soit avant ou pendant l'exécution des services Web. [Benatallah et al., 2003a] considèrent la composition dynamique comme l'agrégation de services Web permettant de résoudre un objectif précis soumis par un utilisateur en prenant en compte ses préférences. Étant donné une spécification de haut niveau des objectifs d'une tâche particulière, la composition dynamique implique la capacité de sélectionner, de composer et de faire interopérer des services existants. La sélection (découverte) dynamique consiste à la possibilité de localiser automatiquement un service Web qui répond à des besoins particuliers. Différentes approches ont été proposées dans la littérature pour réaliser la découverte dynamique de services ([Chakraborty et al., 2001], [Benatallah et al., 2003a], [Paolucci et al., 2002], entre autres). Toutes ces approches implémentent en fait une découverte approximative car il n'est pas réaliste d'imaginer qu'il y a toujours un service qui correspond exactement aux besoins spécifiés [Kellert and Toumani, 2006]. Ces approches diffèrent par le langage de description de services utilisé (OWL-S par exemple) et/ou par l'algorithme de découverte mis en oeuvre. Plusieurs travaux de composition dynamique ont vu le jour parmi lesquels nous pouvons citer ceux qui se basent sur les règles métier comme le travail de [Medjahed et al., 2003] et le travail de [Thakkar et al., 2004] et ceux qui se basent sur la sémantique et la notion d'ontologie comme [Fujii and Suda, 2006].

1.2.5.2 Approches de composition des services pour la coopération

Dans cette section, nous étudions les travaux issus du monde académique qui proposent des approches de composition des services. Les travaux de composition que nous allons présenter appartiennent à trois communautés différentes. Les premiers sont issus de l'ingénierie dirigée par les modèles et considèrent la composition de services comme une préoccupation de modélisation et une projection de cette modélisation sur des plateformes de composition. Les deuxièmes sont issus du domaine de Workflow, un domaine qui s'intéresse fortement à la coopération interentreprises. Quant aux troisièmes, ils s'inspirent du domaine de l'intelligence artificielle et notamment des règles appelées règle métier pour proposer des compositions dynamiques.

Ces différentes approches s'apparentent à la problématique de composition des services Web auxquelles nous nous intéressons (mise en œuvre d'une composition dynamique (ou semi-automatique) dédiée pour la coopération).

- **Approches de composition dirigées par les modèles**

Le travail proposé dans **[Castro et al., 2006]**, **[Castro et al., 2007]** fait partie **du cadre MIDAS** qui est un cadre méthodologique pour l'ingénierie de système d'information de l'entreprise en adoptant l'architecture SOA. Dans leur papier, les auteurs proposent une méthode de modélisation de la composition des services en définissant de nouveaux concepts et de nouveaux modèles. Les différents concepts et modèles associés s'inscrivent au niveau PIM de la démarche MDA.

La méthode proposée englobe quatre modèles à savoir : « *user services model* », « e*xtended use cases model* », « *service process model* » et le « *service composition model* ». Le premier modèle à savoir le « *user service model* » est une extension du diagramme de cas d'utilisation UML et se préoccupe de représenter les services manipulés par les utilisateurs. « l'*extended use cases model* » est aussi une extension du diagramme de cas d'utilisation UML et s'intéresse à la modélisation des services élémentaires et des services composites. Le « *service process model* » constitue une extension du diagramme d'activité UML dont l'objectif est de décrire en détail le flux d'exécution des opérations au sein d'un service. Quant au « *service composition model* », il représente par le biais d'une extension du diagramme d'activité UML, le processus interne d'un service composite. Par rapport au modèle « *service process model* », ce modèle ajoute les concepts « *business collaborator* » et « *activity operation* ».

Chacun de ces modèles propose les concepts qui lui sont relatifs ainsi que les relations qui pourraient exister entre ces concepts. Outre les modèles, la méthode de modélisation de la composition présente un processus de transformation (voir Figure 1.19).

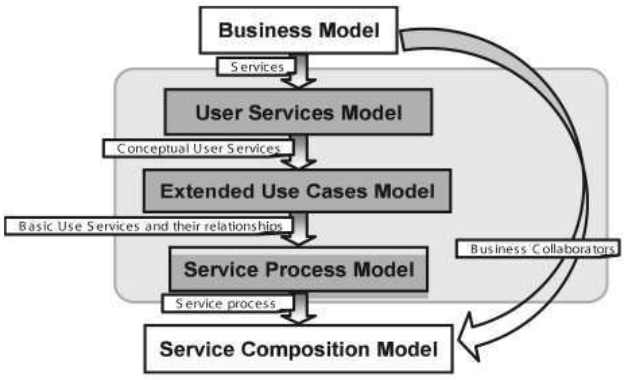


Figure 1.19. Processus de modélisation de la composition [Castro et al., 2006]

Le point d'entrée du processus de transformation est le modèle métier (processus métier, acteurs impliqués, *etc.*) qui servira de base pour l'identification des services requis par les utilisateurs. Une fois identifiés, les services sont modélisés en utilisant les différents modèles jusqu'à obtenir le modèle de composition « *Service Composition Model* ». Le passage d'un modèle à un autre est accompagné par un ensemble de règles de transformation de trois types :

manuelle, automatique et semi-automatique. Ces transformations visent à enrichir, filtrer ou spécialiser le modèle de composition des services sans utiliser des informations dépendantes d'une plateforme.

[Baina et al., 2004] proposent une démarche pour le développement de services Web dirigée par les modèles. Dans leur travail, les auteurs présentent un framework qui génère automatiquement des modèles de composition BPEL à partir des spécifications de protocoles basées sur la description de service (une variation de diagramme d'état-transition). La Figure 1.20 illustre la méthodologie proposée par [Baina et al., 2004] qui s'articule autour de trois phases de transformation. Tout d'abord, chaque transition est transformée dans un squelette de processus. Ensuite, chaque état est transformé dans un squelette d'état en ajoutant la transition sous-jacente à l'état. Enfin, le tout est lié dans un squelette de processus général selon la topologie d'un diagramme d'état transition.

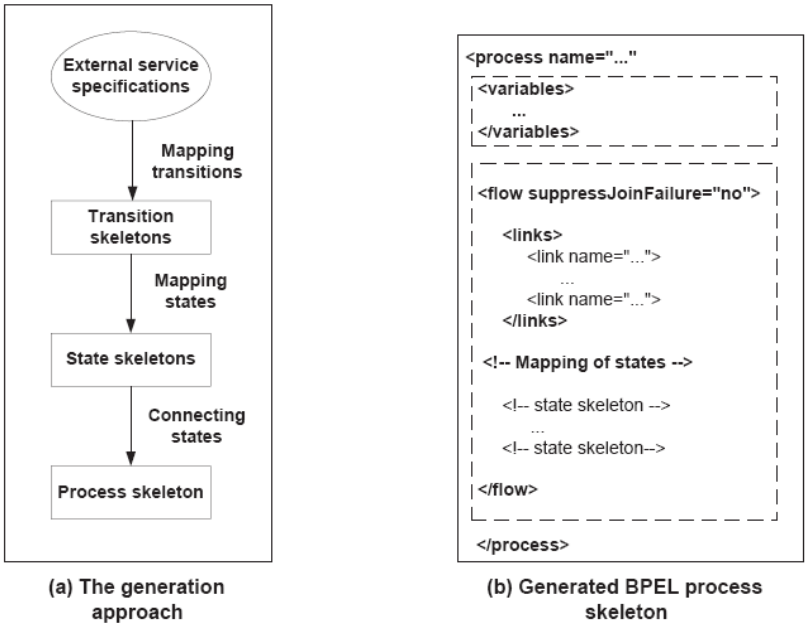


Figure 1.20. Approche de génération de modèle de composition de services Web

[Baina et al., 2004]

Afin de réaliser cette transformation, les auteurs proposent un ensemble de règles de génération du modèle de composition en BPEL à partir de la spécification du protocole. Pour ce faire, les auteurs distinguent les opérations en quatre catégories : *internal*, *singleton*, *serial* et *parallel*. Pour chacune de ces opérations, ils définissent les éléments de BPEL correspondant.

L'approche proposée par **[Touzi, 2007]**, déjà présentée précédemment (cf. section 1.2.4.1.2) traite de la problématique d'intégration des systèmes d'information des partenaires. Il s'agit essentiellement d'une étude de la traduction d'un modèle des besoins situé au niveau « métier » en un modèle d'architecture spécifique situé au niveau « logique ». Bien que le centre d'intérêt de ce travail de recherche n'est pas la composition des services en soi, l'approche

proposée nous semble intéressante. La composition des services se manifeste au niveau de la vue processus qui permet de décrire les appels des différents services appartenant aux différents partenaires.

- **Approches orientées Workflow**

Les approches proposées dans cette section partent du constat que d'une certaine manière, un service composite est similaire à un Workflow [Benatallah et al., 2003b] [Grefen et al., 2000]. Un service composite comporte un ensemble de services atomiques ainsi que les contrôles et les échanges de données entre ces services. De la même façon, un Workflow est composé d'un ensemble d'activités élémentaires structurées, ainsi que l'ordre d'exécution entre elles. Dans ce domaine, la recherche et le développement ont été particulièrement riches. Nous allons exposer dans ce qui suit quelques approches que nous avons jugés pertinentes.

Le projet Self-Serv (compoSing wEb accessibLe inFormation and buSiness sERVices) de **[Benatallah et al., 2003b]** consiste en une plateforme de composition dynamique des services Web interentreprises. La plateforme proposée permet une composition plus aisée des services dans un environnement pair à pair (*Peer to Peer*).

L'architecture proposée par Self-Serv est illustrée en Figure 1.21, elle montre les deux parties fondamentales de l'architecture à savoir : le gestionnaire de services (*Service Manager*) et le *Pool de Services*. Le gestionnaire de services permet de stocker les services, les déployer et aussi les découvrir. Il effectue les fonctionnalités d'un registre dans l'architecture classique des services Web et il est connecté à l'UDDI. Quant au *pool de services*, il se charge de gérer la composition des services.

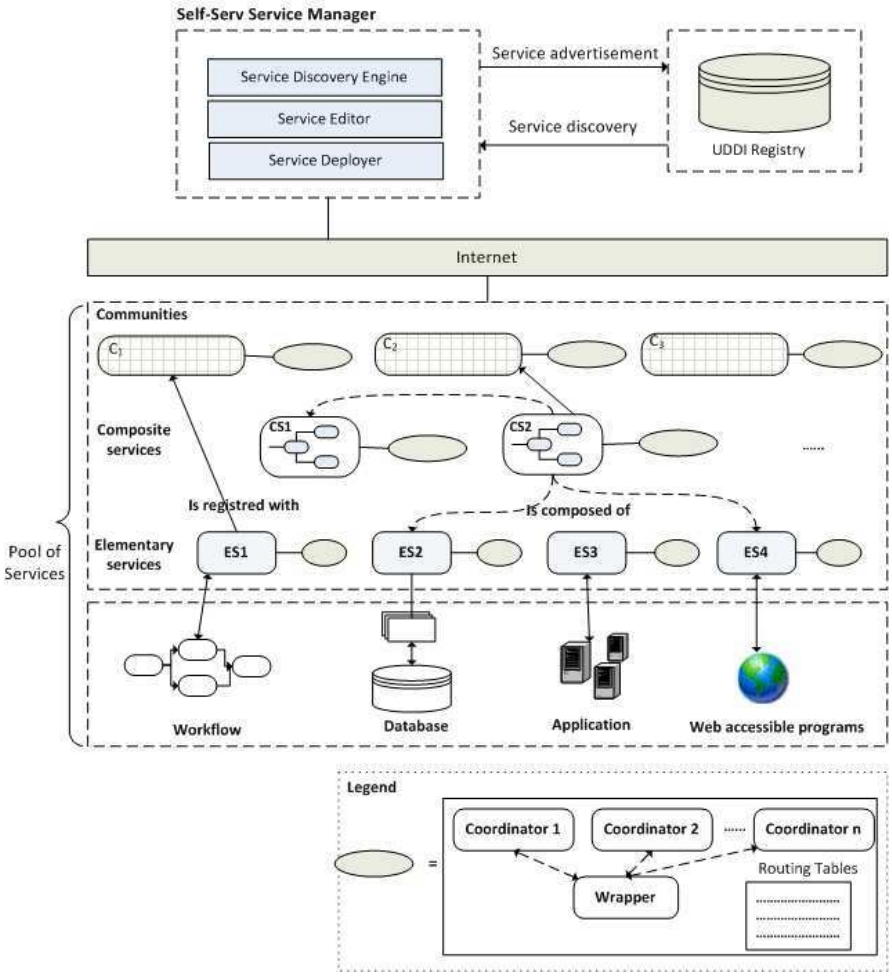


Figure 1.21. Architecture de Self-Serv [Sheng et al., 2002]

Self-Serv distingue trois types de services (voir Figure 1.21) : les services élémentaires, les services composés et les communautés de services. Ces dernières représentent l'agrégation d'un ensemble de services substituables.

Les compositions des services au sein de la plateforme Self-Serv sont rédigées avec un langage déclaratif basé sur les diagrammes à état. Afin de les mettre en œuvre, Self-Serv propose des agents logiciels appelés coordinateurs. En effet, la coordination de l'exécution d'un service composé est distribuée entre les composants appelés coordinateurs. Un coordinateur est généré pour chacun des services composés ainsi que pour les composants. Ils sont hébergés par le prestataire du service correspondant. Les coordinateurs sont des planificateurs légers qui reçoivent les notifications de terminaison des autres coordinateurs et invoquent le service correspondant. Une fois le service accompli, le coordinateur transmet les résultats obtenus à celui qui a invoqué le service et envoie une notification de terminaison aux coordinateurs des états suivants.

eFlow **[Casati and Shan, 2002]**, **[Casati et al., 2000b]** est une plateforme qui permet la spécification, l'exécution et la gestion des services composites. Un service composite est

décrit comme un schéma de processus qui combine des services composites et des services élémentaires. Un processus est modélisé sous la forme d'un graphe qui définit l'ordre d'exécution des nœuds. Les nœuds peuvent être de trois types différents : (i) service, (ii) décision ou (iii) événement. Le nœud service représente l'invocation d'un service basique ou composite. Le service correspondant est choisi au moment de l'exécution. eFlow propose un module de choix des services qui peut être remplacé par un module spécifique en fonction du domaine de l'utilisation. Le nœud décision spécifie les alternatives et les règles qui contrôlent le flux d'exécution. Le nœud événement permet au service d'envoyer ou de recevoir plusieurs types d'événements : données temporelles, événements produits par le processus ou notifications spécifiques des applications externes.

Pour faciliter la mise en place des processus, eFlow propose un registre qui contient des processus complets et des patrons de processus dont certaines parties sont spécifiées comme des services abstraits, pas encore définis. Une instance de processus correspond à l'exécution d'un schéma de processus. Pour supporter l'hétérogénéité des services, eFlow définit des adaptateurs de service qui supportent plusieurs protocoles d'interaction B2B comme par exemple *OBI* ou *RosettaNet*.

- **Approches issues de l'intelligence artificielle (IA)**

La composition des services Web par des techniques issues de l'intelligence artificielle, et plus particulièrement par des techniques de règle, est la voie qui semble prometteuse pour certains auteurs comme [Orriëns et al., 2004] et [Medjahed et al., 2003].

Dans leur travail **[Orriëns et al., 2003]**, **[Orriëns et al., 2004]**, partent du constat qu'un processus métier peut être construit dynamiquement par composition des services, s'il est gouverné par un ensemble de règles métier (*business rules*) ou encore règles de composition des services (*service composition rules*). D'une façon générale, les règles sont des indications logiques qui guident le fonctionnement d'un système. Quand elles traitent du métier de l'entreprise, ces règles deviennent des règles métier. Ces dernières peuvent représenter des situations métier tel qu' « envoyer un document à une personne responsable », ou encore représenter la gestion des exceptions telle que par exemple « s'assurer du délai de réalisation d'une tâche qui ne doit pas dépasser les trente minutes comme prévu dans le contrat avec le client ».

La particularité du travail de [Orriëns et al., 2003], [Orriëns et al., 2004] tient au fait qu'il propose un méta modèle en UML qui sert de base pour la représentation des différentes compositions possibles des services Web. Les auteurs utilisent également le langage OCL (*Object Constraint Language*) afin d'exprimer les règles de composition des services. Ces dernières interviennent lors de la sélection de services et permettent ainsi de structurer la composition. De plus, les auteurs proposent une classification des règles de composition en cinq catégories à savoir : les règles de structure, les règles d'assignation de rôles, les règles d'assignation de messages, les règles de gouvernance des événements imprévus et les règles d'assignation de contraintes.

L'architecture de composition basée sur les règles proposées dans [Orriens et al. 2003] est illustrée dans la Figure 1.22.

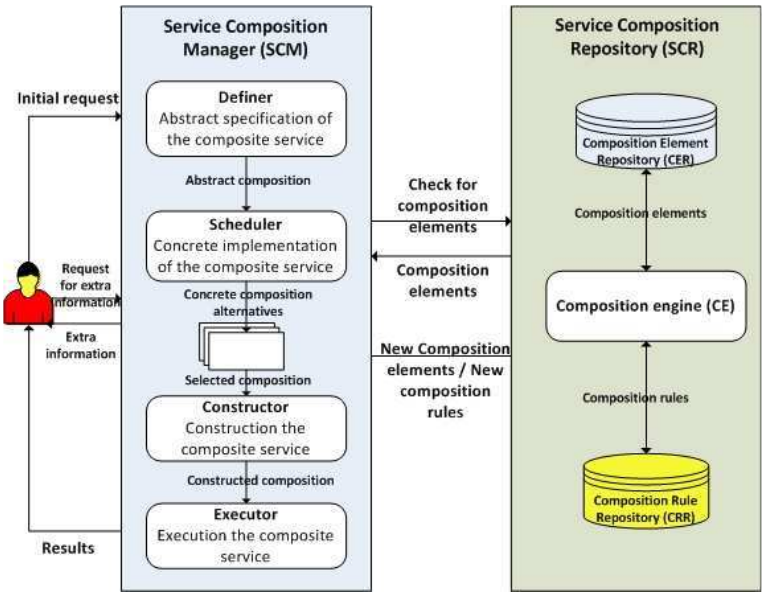


Figure 1.22. Architecture de composition basée sur les règles métier
[Orriëns et al., 2004]

Deux modules peuvent être distingués à savoir : le *Service Composition Manager* (**SCM**) et le *Service Composition Repository* (**SCR**). Le **SCM** (*Service Composition Manager*) est le responsable du développement des services, de leurs exécutions et de la gestion de la composition des services. Il comporte quatre composants : le *Definer*, le *Scheduler*, le *Constructor* et l'*Executor*. Quant au **SCR** (*Service Composition Repository*) est le composant clé de l'architecture proposée puisqu'il est le responsable de la gestion des règles métier. Il comporte trois éléments fondamentaux : le *Composition Engine* (**CE**), le *Composition Element Repository* (**CER**) et le *Composition Rule Repository* (**CRR**). Le **CE** facilite le stockage et la récupération des règles métier et des éléments de la composition contenus respectivement dans le **CER** et le **CRR**.

Le scénario proposé par [Orriëns et al., 2004] se déroule comme suit : un utilisateur émet une requête au **SCM** en spécifiant l'activité qu'il souhaite réaliser (un plan de voyage). Le **SCM** transmet cette requête au *Definer*. Ce dernier, interroge le **CE** afin de savoir s'il existe des informations concernant l'activité en question. En recevant cette demande, le **CE** interroge le **CRR** pour vérifier s'il y aurait des règles de composition se rapportant à l'activité proposée par l'utilisateur. Si c'est le cas, le **CE** emploie la règle afin de rechercher dans le **CER** les activités qui la satisfassent. Par la suite, le *Definer* propose à l'utilisateur l'ensemble des activités trouvées pour qu'il choisisse celles qu'il veut inclure dans la composition. L'utilisateur pourra également définir et ajouter d'autres activités à la composition. Dans ce cas, le *Definer* rajoute ces activités, les stocke également dans le **CER** et met à jour les règles relatives aux activités de planification de voyage stockées dans le **CRR**. Dans le cas où l'utilisateur ne voudrait pas utiliser les activités trouvées, il devra par la suite déterminer les activités qui doivent être incluses dans le processus de composition d'un plan de voyage. De la même manière le *Definer* valide ces activités et met à jour le **CER** et le **CRR**.

La validation des activités donne naissance à une définition abstraite d'un processus de composition. Par la suite, le *Scheduler*, se charge d'instancier le processus abstrait déjà

spécifié par le *Definer* en se basant sur les services publiés dans l'UDDI. Le *Scheduler* opère de la même manière qu'un *Definer*. Il consulte à chaque fois le **CE** afin de savoir s'il existe ou pas des règles pour la sélection des services. Si c'est le cas, il envoie à l'utilisateur l'ensemble de services trouvés ainsi que leurs fournisseurs afin qu'il choisisse ceux qui lui semble les plus adéquats. Le *Scheduler* présente à l'utilisateur plusieurs instanciations du même processus abstrait et c'est à l'utilisateur de choisir l'instance qu'il considère pertinente. L'instance choisie sera transmise au *Constructor* afin de planifier le processus d'exécution. Par la suite, l'*Executor* se charge d'exécuter et de superviser le service composite.

Un autre travail qui s'inscrit dans la même voie de recherche est celui de **[Medjahed et al., 2003]**, **[Medjahed and Bouguettaya, 2005]** qui propose un framework pour la composition dynamique des services à base de règles. Ces dernières sont des règles de *composabilité* qui déterminent dans quelle mesure deux services sont composables en se basant sur la comparaison de leurs caractéristiques syntaxiques et sémantiques. Les caractéristiques syntaxiques dépendent (i) du protocole d'interaction supporté par chaque service et (ii) du *binding* protocole. Quant aux caractéristiques sémantiques elles sont liées (i) à la comparaison du nombre de paramètres contenus dans un message, leurs types de données, *etc.*,(ii) de la sémantique des opérations offertes par le service, (iii) des propriétés qualitatives et (iv) de la valeur ajoutée de la composition.
L'approche proposée par [Medjahed et al., 2003] se déroule en quatre phases :

- La phase de spécification : les auteurs décrivent la requête utilisateur en se basant sur le langage CSSL (*Composite Service Specification Language*). Cette requête inclut l'ordre des opérations désirées par le client. Ce dernier peut aussi spécifier le flux de contrôle qui pourrait exister entre les différentes opérations. Par contre, il ne pourra pas imposer quel(s) service(s) composite(s) pourra exécuter les opérations sélectionnées.
- La phase de correspondance : se base sur les règles de *composabilité* afin de générer des plans conformes aux spécifications de l'utilisateur. En effet, pour chaque opération choisie, il lui est attribué une ou plusieurs opérations appartenant à des services élémentaires. Cette attribution se fait en considérant les caractéristiques syntaxiques et sémantiques des services.
- La phase de sélection : si plusieurs plans ont été générés alors la sélection est réalisée en se basant sur les paramètres de qualité de la composition.
- La phase de génération : permet de générer automatiquement une description détaillée du service composite qui sera fourni à l'utilisateur.

- **Synthèse et discussion**

Les travaux que nous avons étudiés montrent que la composition des services est au centre d'une intense activité de recherche. Plusieurs approches ont été proposées, chacune appartient à un champ d'études bien définie (approches dirigées par les modèles, approches orientées Workflow et approches issues de l'intelligence artificielle). Nous avons voulu résumer, dans le Tableau 1.7, les principales caractéristiques des approches de composition des services présentées précédemment. Les caractéristiques de ces approches sont synthétisées à l'aide de cinq critères qui sont, à notre sens, les plus judicieux :

- Le critère *point de vue* qui permet de préciser le niveau d'abstraction (conceptuel, technique) associé à la démarche considérée,
- Le critère *type de composition* qui définit le type de la composition (manuelle, automatique, dynamique, semi-automatique),
- Le critère *description de la méthode* qui précise les notations ou les modèles proposés pour soutenir l'approche de composition. Trois types de notations existent pour décrire une approche de composition : la notation informelle comme par exemple le langage naturel [Papazoglou and Heuvel, 2006], la notation semi-formelle comme les règles [Orriëns et al., 2003], les diagrammes UML et les notations formelles telles que la logique des situations (Pi-Calcul),
- Le critère *flexibilité* qui permet de représenter la capacité de l'approche à faire face à de nouveaux changements ou à des scénarios d'évolution,
- Le critère *niveau de maturité* désigne en fait le degré de maturité de l'approche vis-à-vis de la problématique de coopération interentreprises.

Critère	**Approches dirigées par les modèles**		**Approches orientées Workflow**		**Approches issues de l'IA**	
	[Baina et al., 2004]	[Castro et al., 2006]	[Benatallah et al., 2003a]	[Casati and Shan, 2002]	[Orriëns et al., 2003]	[Medjahed and Bouguettaya, 2005]
Point de vue	Conceptuel Technique	Conceptuel	Technique	Technique	Conceptuel Technique	Technique
Type de composition	--	--	Automatique	Dynamique	Automatique	Dynamique
Description de la méthode	Semi-formelle	Semi-formelle	Semi-formelle	Semi-formelle	Semi-formelle	Semi-formelle
Niveau de maturité l'approche	Forte	Très faible	Moyenne	Forte	Moyenne	Moyenne
Flexibilité	Non	Non	Oui	Oui	Oui	Oui

Tableau 1.7. Comparaison des approches de composition des services

L'analyse du Tableau 1.7 permet de retenir un certain nombre de points importants qui sont :

- Une partie des approches de composition adoptent une vision technique et se préoccupent essentiellement de définir des mécanismes pour l'exécution des services et non pour la définition conceptuelle de la composition [Benatallah et al., 2003a], [Casati and Shan, 2002], [Medjahed and Bouguettaya, 2005]. Le travail proposé par [Castro et al., 2006], adopte une vision fortement conceptuelle et ignore la technologie support à la démarche conceptuelle. Quoique, dans leurs perspectives de travail, les auteurs envisagent de proposer des transformations du modèle de composition dans une plateforme spécifique de composition (le langage BPEL).

  Nous considérons que la composition des services est loin d'être un problème purement technique et surtout dans un contexte de coopération interentreprises. Certes, les mécanismes techniques pour la composition sont fondamentaux, mais la composition doit être accompagnée par une démarche conceptuelle.

- Les types de composition supportés par les approches de composition sont largement variés. [Benatallah et al., 2003a] et [Orriëns et al., 2003] adoptent l'approche automatique de la composition. Partant d'un objectif à atteindre, leurs approches respectives permettent de déterminer le schéma de composition ainsi que les services qui le composent. La volonté d'automatiser à outrance n'est certainement pas une voie réaliste. Certains travaux de recherche semblent faire abstraction de la complexité du contexte de l'automatisation par des hypothèses simplificatrices qu'ils imposent dans leurs solutions. En effet, dans le cadre d'une coopération interentreprises de nombreuses tâches doivent rester à la charge des concepteurs. Il est, par exemple, illusoire de vouloir automatiser complètement la gestion d'une chaîne logistique [Kellert and Toumani, 2006]. Nous confirmons cette assertion dans nos travaux de recherche et nous considérons qu'il est important que le client participe dans la création du processus coopératif construit à partir de composition des services.

  [Casati and Shan, 2002] et [Medjahed and Bouguettaya, 2005] s'intéressent à une approche dynamique de la composition. La requête client inclut l'ordre d'opérations désirées ainsi que le flux de contrôle qui pourra exister entre les différentes opérations. Par la suite, il s'agit de sélectionner dynamiquement les services répondant à cette requête.

  L'orientation dynamique de la composition est généralement centrée utilisateur dans le sens où ce dernier supervise la phase de spécification ainsi que la phase de découverte des services. Ce point est important dans le cadre d'une coopération interentreprises où l'entreprise initiatrice de projet commence par définir une spécification du schéma de coopération qui sera satisfait en cherchant l'ensemble des services adéquats.

  Fort de ces constats, nous allons favoriser dans notre travail de recherche une solution de composition semi-automatique. Le client (*i.e.* l'entreprise initiatrice du projet de coopération) maintiendra un certain contrôle sur le processus envisagé. Néanmoins, il n'aura pas besoin de connaissances de programmation lors de la définition du processus de coopération et ceci en se basant sur des outils de modélisation. Par la suite, la découverte des services participants au processus coopératif se réalisera d'une manière dynamique. Dans le chapitre 3 nous exposerons notre démarche de construction du processus coopératif à base de composition des services.

- La description de la composition est semi-formelle dans la majorité des approches présentées. Par exemple, dans Self-Serv la composition des services est modélisée par le biais de diagrammes d'états [Harel, 1987], tandis que dans eFlow le formalisme choisi est celui des réseaux de Pétri. L'approche de composition présentée dans [Castro et al., 2006], est structurée en un ensemble de modèles supportés par des méta-modèles. La proposition des modèles et des méta-modèles est capitale pour comprendre les concepts manipulés ainsi que leur sémantique. Cette affirmation rejoint la nécessité d'avoir une vision conceptuelle pour la composition évoquée précédemment.
- Concernant le niveau de maturité vis-à-vis de la problématique de coopération interentreprises, certaines approches ont été proposés pour répondre aux spécificités de la coopération comme les travaux de [Touzi, 2007] et [Casati and Shan, 2002]. D'autres

travaux comme [Medjahed and Bouguettaya, 2005] et [Orriëns et al., 2003] sont des approches appartenant au domaine de la composition des services pour répondre à des requêtes complexes des clients. Ainsi, ils ne traitent que d'une manière implicite la coopération interentreprises.

- Malgré son importance dans le milieu industriel, la composante de flexibilité ne constitue pas une préoccupation centrale des approches de composition présentées. La majorité des propositions ne supportent pas les scénarios d'évolution ou d'adaptation d'un schéma de composition.

1.2.6 Conclusion

Cette partie de l'état de l'art a mis en évidence le concept de service et sa pertinence pour la coopération à la demande. Elle a présenté à travers un cadre de référence les différents aspects relatifs au service : la vue architecture, la vue technique, la vue méthode et la vue composition.

La vue architecture a présenté l'architecture orientée services dont le principe consiste à structurer le système d'information de l'entreprise comme un ensemble de services qui exposent leur interface fonctionnelle et qui communiquent par messages. L'instanciation la plus connue de cette architecture est la technologie service Web. Cette dernière a fait l'objet de la vue technique

La vue technique a exposé les différents standards et langages sur lesquels se basent les services : SOAP pour la communication et le dialogue avec les services Web, le standard WSDL ou le langage OWL-S pour la description sémantique des services, le registre UDDI pour la publication des services et le langage BPEL pour la composition des services.

Notre étude de l'ensemble des standards et langages a montré l'insuffisance de certains langages comme WSDL, OWL-S ou encore le registre UDDI et le langage BPEL par rapport à l'intégration des informations de contexte du service. En effet, les langages de description des services (comme WSDL et OWL-S) ne proposent pas une description assez large pour représenter des services adaptés au contexte. En outre, le registre UDDI ne prévoit pas de mécanisme pour intégrer les informations de contexte. De plus, le langage BPEL hérite la vue statique des Workflows et ne permet pas les adaptations du comportement du processus exécutable. Ces différentes constatations s'intéressent essentiellement à la sous problématique d'adaptation des services lors de la construction du processus coopératif (cf. introduction générale, section 4). Nos travaux présenteront dans les chapitres 3 et 4 des réponses à ces limites partagées entre les standards et langages des services Web.

La vue méthode a souligné l'importance de la modélisation, de la méta-modélisation pour accompagner la méthode de mise en place de la SOA. En outre, elle a montré l'intérêt grandissant pour l'approche MDA dans le domaine des services Web ou des services d'une façon générale. L'étude des travaux dans ce domaine (*i.e.*, la démarche MDA avec les services) a révélé l'absence d'une démarche unifiée qui guide l'utilisation de l'approche MDA lors de la construction de la SOA. Cette constatation, nous pousse à nous intéresser à cette orientation lors de nos travaux de recherche (cf. chapitre 2). La vue méthode a également

exposé quelques méthodes de mise en place d'une SOA présentées dans la littérature. Cette revue de la littérature a exhibé deux constats majeurs : le manque des travaux scientifiques qui traitent de la mise en place de la SOA et le manque de prise en compte de l'adaptation des services au moment de leur modélisation. L'adaptation des services est une caractéristique importante qui peut favoriser considérablement la flexibilité des services issus de la démarche.

Nous allons essayer de répondre à ces limites dans le chapitre 2 de ce manuscrit de thèse.

La vue de composition des services a présenté tout d'abord les différents types de composition existants (manuelle /automatique, statique/dynamique). Ensuite, elle a exposé les approches de composition des services Web proposées pour des fins de coopération interentreprises.

L'analyse de ces travaux a permis de retenir qu'il existe plusieurs travaux de composition avec des degrés de maturité différents vis-à-vis de la coopération. La majorité des approches de composition adoptent une vision technique et s'intéressent à la proposition des mécanismes pour l'exécution des services et non à un cadre pour la définition conceptuelle de la composition. Cette affirmation soutient en particulier l'importance de la conception d'une architecture de composition pour la coopération, à contrario d'une projection technique qui bien que loin d'être triviale, semble tout de même moins problématique.

Une limite partagée entre les travaux tient au fait qu'ils ne prennent pas en compte la préoccupation de l'adaptation lors de la composition des services. Cette particularité permet d'assurer la flexibilité de la solution. Nous allons essayer dans le chapitre 4 de répondre à ces limites énumérées.

Le concept d'adaptation est un concept assez important dans le cadre de nos travaux de recherche. La dernière section de cet état de l'art proposera une revue de la littérature sur l'adaptation des services au contexte d'utilisation.

1.3 Adaptation des services Web au contexte

Nous avons étudié dans la partie précédente de cet état de l'art les travaux qui portent sur l'approche service pour la coopération interentreprises. Nous avons essentiellement présenté les approches de mise en place de la SOA au sein de l'entreprise ainsi que les approches de composition des services pour des fins de coopération. Notre étude a révélé que, le concept de service est un concept qui a le vent en poupe et qui est bien approprié à la construction des processus coopératifs à la demande, grâce à sa caractéristique de composition. Cependant une des limites majeures du service, c'est le manque de dynamisme.

Nous proposons de pallier à cette limite en étudiant l'adaptation des services au contexte. Doter le service d'un ensemble de mécanismes qui lui permettent de changer son comportement en fonction des changements du contexte, devient un moyen efficace pour assurer son dynamisme.

Plusieurs auteurs dans la littérature se sont penchés sur la technologie des services Web et l'adaptation au contexte. Des auteurs comme [Benslimane et al., 2006], [Sheng et al., 2009], [Maamar et al., 2007a] considèrent qu'une nouvelle préoccupation qui met la notion de

contexte au centre des travaux sur les services Web émerge. Ceci est dû en fait, à la nature dynamique d'Internet en général et des services Web en particulier. D'une part, les services Web ont besoin d'adapter leurs opérations afin de répondre au mieux à la situation dans laquelle ils sont appelés. D'autre part, les services doivent pouvoir participer ou non à une composition en fonction du contexte. Cependant, les langages et les spécifications relatives aux services Web ne prennent pas en compte la notion de contexte. Plusieurs auteurs comme [Chen et al., 2004a] soulignent l'incapacité du langage WSDL ou encore le langage OWL-S, avec leurs formes actuelles, à intégrer les informations de contexte.

D'autres auteurs comme [Cuddy et al., 2005] affirment l'incapacité du registre UDDI à intégrer les informations contextuelles pour la découverte des services.

Dans cette partie de l'état de l'art, nous commençons notre étude par la définition du contexte (cf. section 1.3.1). Nous exposons la notion de sensibilité au contexte (cf. section 1.3.2) et nous présentons ensuite les éléments fonctionnels d'un système sensible au contexte (cf. section 1.3.3). Dans la section 1.3.4, nous nous intéressons à l'adaptation, qui constitue le dernier maillon de la chaîne de sensibilité des systèmes au contexte. Nous évoquons aussi ses finalités et ses mécanismes supports. Ensuite, nous passons en revue les différents travaux de l'adaptation des services Web au contexte (cf. section 1.3.5). Enfin, nous synthétisons ces travaux à l'aide d'un ensemble de critères afin de mieux positionner notre travail de recherche.

1.3.1 Notion de contexte

L'encyclopédie Larousse définit le contexte comme « un ensemble des conditions naturelles, sociales, culturelles dans lesquelles se situe un énoncé, un discours » ou encore comme un « ensemble des circonstances dans lesquelles se produit un événement, se situe une action ».

Au regard de la littérature informatique, il n'existe pas une seule définition du contexte. Notamment, divers domaines de recherche se sont attelés à définir la notion de « contexte » [Chen and Kotz, 2000], [Saha and Mukherjee, 2003]. Cette notion a trouvé son essor dans le domaine de l'informatique pervasive ou ambiante où elle a suscité l'intérêt de plusieurs chercheurs. De nombreux travaux issus de cette communauté de recherche tentent de donner une définition du contexte. Selon [Tigli and Lavirotte, 2006], quatre familles de contexte peuvent être identifiées. Chacune de ces familles apporte sa propre définition du contexte. Ces familles de contexte sont les suivantes : la notion de contexte géolocalisé, la notion de contexte environnemental, la notion de contexte généralisé et la notion de contexte unifié.

- **Notion de contexte géolocalisé**

[Schilit and Theimer, 1994] définissent principalement le contexte en se référant à la géolocalisation de l'utilisateur : « *la localisation de l'utilisateur, les identités et les états des personnes et des objets qui l'entourent* ». La vision de [Schilit and Theimer, 1994] pour le contexte met l'accent sur les questions de type « où est l'utilisateur ? », « avec qui ? », « quelles sont les ressources utilisés ? » *etc*.

- **Notion de contexte environnemental**

[Brown, 1996] relit le contexte aux éléments de l'environnement de l'utilisateur tout en introduisant de nouveau concepts comme l'heure, la saison, la température, l'identité et la localisation de l'utilisateur. Peu après [Ryan et al., 1997] présentent une définition du contexte qui introduit explicitement la notion du temps : « *C'est la localisation, l'environnement, l'identité et le temps relatif à un utilisateur* ».

La définition la plus reconnue dans la littérature est proposée par Dey [Dey et al., 1999], qui considèrent que le contexte est construit à partir de tous les éléments d'information qui peuvent être utilisés pour caractériser la situation d'une entité. Une entité correspond à toute personne, tout endroit, ou tout objet (en incluant les utilisateurs et les applications eux-mêmes) considéré (e) comme pertinent (e) pour l'interaction entre l'utilisateur et l'application. Cependant, cette définition est souvent considérée comme générique, ce qui rend impossible l'énumération des informations qui font partie du contexte de celles qui n'en font pas partie. Ce caractère généraliste a suscité les critiques de certains auteurs comme [Winograd, 2001]. Ce dernier affirme que quoique la définition de [Dey et al., 1999] englobe l'ensemble des définitions proposées dans la littérature, l'utilisation des expressions comme « toute information » ou encore « caractériser une entité » ne trace aucune limite à la notion de contexte (*i.e.* tout peut être considéré comme contexte).

- **Notion de contexte unifié**

[Jang and Who, 2003] proposent le concept de contexte unifié et fixent le contexte comme étant une situation donnée par les réponses à six questions : Qui (*Who*) ?; Quoi (*What*) ?; Où (*Where*) ?; Quand (*When*) ?; Comment (*How*) ?; Pourquoi (*Why*) ? La réponse à la question « Qui » définit le sujet de l'action qui peut être un utilisateur humain ou une entité logicielle. La réponse à la question « Quoi ? » représente l'objet qui supporte l'action. Les réponses aux questions « Où ? » et « Quand ? » renseignent sur les informations spatio-temporelles. La réponse à la question « Pourquoi ? » décrit les objectifs du sujet. Quant à la réponse à la question « Comment ? », elle détaille la façon dont l'action va être réalisée.

- **Notion de contexte généralisé**

Certains auteurs ont proposé une définition générale du contexte. [Gong, 2005] définit le contexte comme étant ce qui entoure et donne du sens à quelque chose d'autre. À l'égard de ses critiques envers la définition de [Dey et al., 1999], [Winograd, 2001] définit le contexte comme « l'ensemble d'informations structuré et partagé qui évolue et sert l'interprétation ». De plus, il appuie sa définition par le fait que « la considération d'une information comme contexte est due à la manière dont elle est utilisée et non à ses propriétés inhérentes ». Ainsi, l'ensemble des informations et son interprétation dépendent de la finalité des travaux utilisant le contexte.

Nous considérons la définition de [Winograd, 2001] comme une définition bien appropriée pour l'exploitation du contexte dans le domaine industriel. Nous avons repris les principes de cette définition et nous l'avons étendu pour s'ajuster à notre domaine de recherche à savoir la

coopération interentreprises. Nous détaillons notre vision du contexte dans le chapitre 3 de ce manuscrit.

1.3.2 Notion de sensibilité au contexte

La notion de sensibilité au contexte utilise le contexte afin de personnaliser et d'adapter les applications à différents environnements, à différentes vues sur les données, à diverses circonstances extérieures, *etc.* Cette notion est une traduction de l'expression anglaise « *Context-Awarness* ». [Schilit and Theimer, 1994] ont défini un système sensible au contexte comme tout système « qui s'adapte selon son endroit d'utilisation, la collection de personnes et objets voisins aussi bien que le changement de ces objets ». D'autres auteurs comme [Hull et al., 1997] et [Pascoe, 1997] considèrent la sensibilité au contexte comme la capacité d'un système à détecter, capturer, interpréter et réagir à un aspect changeant de l'environnement d'un utilisateur et des dispositifs qu'il utilise.

Selon [Dey et al., 2001], les définitions attribuées au système sensible au contexte n'incluent pas tous les types de systèmes sensibles au contexte. En effet, dans le cadre de ces définitions, un système qui ne fait que collecter le contexte pour le fournir à une application n'est pas considéré comme système sensible au contexte. Ainsi, ils considèrent qu'« un système est sensible au contexte s'il utilise le contexte pour fournir des informations et des services pertinents pour l'utilisateur, où la pertinence dépend de la tâche demandée par l'utilisateur » [Dey et al., 2001].

D'autres travaux comme [Xiaohang, 2003] ont constaté que les définitions des systèmes sensibles au contexte n'englobent pas la description du processus de gestion du contexte. Ainsi, l'auteur stipule que « la sensibilité au contexte d'un système logiciel est sa capacité à acquérir, gérer, interpréter et répondre aux changements du contexte afin de fournir les services appropriés ».

Dans le cadre de nos travaux, nous ferons une distinction entre les systèmes qui s'adaptent en fonction du contexte et ceux qui fournissent le contexte. Ainsi, nous appelons service (ou système) sensible au contexte tout système (ou service) qui s'adapte aux changements du contexte.

Malgré la multiplicité des travaux [Ryan et al., 1997], [Chen and Kotz, 2000] ,[Baldauf et al., 2007], le domaine de la sensibilité de contexte est loin d'être précis. [Winograd, 2001] souligne qu'il existe plusieurs points à approfondir et à étudier. De plus, l'auteur énumère trois lacunes principales de ce domaine : (i) la notion de contexte n'est pas encore bien définie, (2) la majorité des travaux manquent de modèles et (iii) peu d'outils sont disponibles pour le développement des systèmes sensibles au contexte.

Par rapport à ces trois limites, nous pouvons ajouter également que la sensibilité au contexte a été exclusivement traitée dans le domaine de l'informatique mobile ou pervasive. À notre connaissance la sensibilité au contexte est un sujet peu exploré dans le domaine des organisations.

Dans la section suivante, nous détaillons les éléments de base et les fonctions qu'un système doit accomplir pour qu'il puisse être sensible au contexte.

1.3.3 Éléments fonctionnels d'un système sensible au contexte

Dans le domaine de l'informatique mobile ou pervasive un système sensible au contexte est caractérisé par des fonctionnalités qui lui permettent d'interagir avec des capteurs, d'analyser les données collectées et d'agir en conséquence. Par analogie à ce domaine, nous considérons un service sensible au contexte comme un service qui possède des fonctionnalités lui permettant de collecter des informations de contexte, de les analyser et de s'adapter en conséquence.

Plusieurs travaux ont proposé des architectures pour les systèmes sensibles au contexte [Dey et al., 1999], [Baldauf et al., 2007]. Ces architectures diffèrent dans les éléments fonctionnels qu'ils englobent, le nom et l'emplacement de leurs couches. Cependant, la majorité des propositions se basent sur quatre couches principales : capture du contexte, interprétation du contexte, stockage du contexte et dissémination du contexte (voir Figure 1.23).

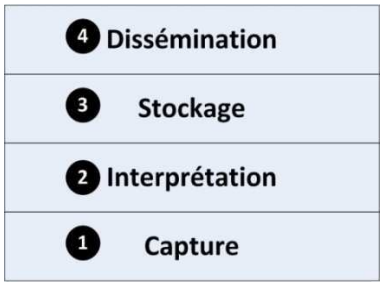


Figure 1.23. Éléments fonctionnels d'un système sensible au contexte

Nous allons essayer dans ce qui suit de décrire ces quatre couches et leurs différents éléments. Au fur et à mesure de notre étude, nous allons identifier les éléments fonctionnels qui peuvent être intégrés au sein de l'architecture d'un service sensible au contexte dans le domaine de la coopération interentreprises.

1.3.3.1 Couche de capture du contexte

La capture ou l'acquisition du contexte représente le processus de collecte des données à partir de l'environnement du système. La collecte des informations de contexte dépend de la source du contexte et du type des informations collectées. Par exemple, les informations du profil d'un client sont des informations explicitement fournies par ce dernier, elles sont caractérisées par un changement peu fréquent. En revanche, le contexte collecté à partir des capteurs est sujet à de fréquents changements. Sa collecte nécessite une interaction avec des capteurs logiciels ou matériels distribués et hétérogènes.

Étant donnée la finalité de nos travaux de recherche, c'est-à-dire l'adaptation des services dédiés à des fins de coopération, nous nous intéressons particulièrement à deux types de capture de contexte : (i) la capture explicite du contexte dans le sens où c'est à la charge du client de définir son contexte de coopération (ii) la capture implicite où des modules logiciels se chargent de mettre à jour les informations de contexte pendant l'exécution du service. Nous proposerons notre solution de capture de contexte dans le chapitre 4 de ce manuscrit de thèse.

1.3.3.2 Couche d'interprétation du contexte

L'interprétation du contexte représente le mécanisme qui déduit un contexte de haut niveau en utilisant d'autres informations de contexte. Les informations utilisées pour déduire un contexte de haut niveau peuvent être des informations interprétées, des informations capturées ou des informations de profil. Par ailleurs, les capteurs fournissent généralement des données brutes qui ne sont pas appropriées pour une utilisation immédiate par le système, plusieurs transformations sont nécessaires : l'extraction, la quantification, raisonnement, agrégation, *etc*.

Nous considérons que dans le cadre d'une coopération interentreprises, les deux premières couches de la Figure 1.23 peuvent être intégrées en une seule couche qui permet à la fois la capture et l'interprétation des informations de contexte. Ce choix est justifié par le fait que dans notre travail de recherche, les observations et les interprétations du contexte permettent de détecter les situations pertinentes dans le but d'adapter le service, contrairement aux travaux de l'informatique mobile où l'interprétation se focalise sur la résolution des conflits causés par l'utilisation de plusieurs sources de contexte.

1.3.3.3 Couche de stockage du contexte

Cette couche permet d'organiser les données capturées et de les stocker pour une utilisation future. Cependant pour stocker une information, il est indispensable de définir un modèle pour la décrire.

Dans la suite, nous classifions la modélisation du contexte en trois types d'approches : l'approche paires/triplet, l'approche orientée modèle, et l'approche orientée ontologie. Cette classification se base sur la structure de données utilisée pour modéliser le contexte, les outils utilisés à cet effet, l'expressivité du modèle et la possibilité de le réutiliser.

- **Approches paires/triplets :** ces approches présentent le contexte sous formes de paires (attribut, valeur). L'attribut caractérise le nom d'une information contextuelle. La valeur représente la valeur courante de cette information. Les premiers travaux pour la description du contexte sont présentés par [Schilit et al., 1993], dans lesquels les auteurs proposent l'utilisation d'un serveur d'environnement dynamique qui se charge d'observer un ensemble de contextes. Quant à [Schmidt et al., 1999], ils modélisent le contexte comme un triplet constitué de l'attribut, de la valeur observée du contexte, et un degré de certitude sur la cohérence des données observées. Par exemple, (*Location*, *ConferenceRoom*, 95), définit la localisation comme le lieu d'une conférence avec un degré de certitude égal à 95.

  La modélisation de type paires/triplets utilise des structures de données simples à gérer, ce qui facilite son implantation. Cependant, elle ne permet pas une description complète du contexte, ni l'expression des relations qui peuvent exister entre les informations de contexte.

- **Approches orienté modèle :** ces approches utilisent des modèles formels afin de modéliser les informations de contexte. Certains auteurs comme [Sheng and Benatallah, 2005] ont exploité la généricité du langage UML pour modéliser le contexte. Dans leur travail, les auteurs ont proposé un méta-modèle basé sur une extension d'UML qui

permet de modéliser le contexte auquel les services Web sont sensibles. Ce langage est appelé *ContextUML*. Le méta-modèle support au langage comporte un ensemble de classes qui permettent de créer des services sensibles au contexte. La classe *Context* permet de décrire un contexte observable. Ce dernier peut être un contexte de bas niveau représenté par la classe *AtomicContext*, ou encore un contexte interprété représenté par la classe *CompositeContext*. Le méta-modèle *ContextUML* permet également la description des actions d'adaptation et les situations pertinentes qui permettent de les déclencher. Un des points fort du méta-modèle *ContextUML* est sa généricité. Cependant, la description des relations de dépendance entre les informations de contexte n'a pas été considérée. D'autres auteurs comme [Pascoe, 1997] utilise le langage de balises comme le langage XML afin de décrire à la fois le contexte associé à l'utilisateur ainsi que le profil de l'application qui caractérise le contexte nécessaire pour son exécution. Le travail de l'auteur a donné naissance à un protocole baptisé *ConteXtML* [Ryan, 2006]. Bien que la modélisation de données hiérarchiques grâce aux balises XML soit efficace pour certains types d'applications, elle ne permet pas d'expliciter les contextes interprétés. De plus, aucune description des relations entre les informations de contexte n'est prévue.

Les approches orientées modèle utilisent un modèle formel pour décrire le contexte. Ce type d'approche permet de décrire le contexte de manière plus riche que les approches paires/triplets. Les modélisations basées sur un langage de balises permettent de décrire des contextes simples, sans offrir la possibilité de décrire des relations de dérivation et de dépendance entre ces informations. Les modélisations existantes basées sur UML permettent de décrire des relations entre les informations de contexte, mais ne prennent pas en compte la description des dépendances entre ces informations.

- **Approches orientées ontologie :** plusieurs modèles d'ontologies ont été proposés afin de modéliser le contexte. Ces approches permettent non seulement de modéliser le contexte, mais aussi de raisonner sur les données décrites. *CoBrA-ONT* (*Context Broker Architecture ONTology*) est une ontologie écrite en utilisant le langage OWL-DL [Chen et al., 2004a]. Cette ontologie a été créée pour décrire le contexte que les applications multi-agents de maison intelligente utilisent [Chen et al., 2004b]. *CoBrA-ONT* se compose de quatre sous ontologies qui décrivent des informations de localisation, des informations d'agents (des personnes ou des agents logiciels), la localisation de ces agents et leurs activités. *CoBrA-ONT* se focalise sur la modélisation des contextes et offre au système la possibilité de raisonner sur les informations collectées.

CONON (*CONtext ONtology*) est une ontologie de contexte extensible écrite en utilisant le langage OWL-DL. Elle permet de décrire le contexte auquel des services peuvent être sensibles dans un environnement ubiquitaire [Wang et al., 2004b]. Cette ontologie est structurée en deux niveaux. Le premier niveau permet de décrire les concepts généraux des informations de contexte. Les concepts généraux concernent les informations de localisation, les informations sur l'utilisateur, les informations sur des activités et les entités de calcul. Quant au second niveau, il permet de décrire des contextes pertinents spécifiques à un domaine [Wang et al., 2004a]. L'ontologie *CONON* se limite à la description du contexte et de certaines relations existantes entre les informations de contexte. L'idée d'avoir un niveau d'ontologie pour décrire les concepts communs et un niveau plus spécifique à l'application

nous semble être une idée intéressante à exploiter et à mettre en œuvre pour la description des services sensibles au contexte.

L'objectif de l'étude que nous avons réalisé consiste à montrer l'apport de chaque approche de modélisation de contexte dans le but d'utiliser l'une d'elles pour modéliser les informations de contexte relatives aux services. Notre choix d'une approche de modélisation comporte plusieurs critères : l'expressivité du modèle, la possibilité de sa réutilisation et de son extension et la possibilité de publier les informations de contexte afin de les exploiter dans l'adaptation.

L'approche orientée ontologie répond bien à ces cinq critères, ainsi nous rejoignons l'affirmation de [Stang and Linnhoff-Popoen, 2004] qui stipule que l'approche orientée ontologie est l'approche la plus expressive et la plus prometteuse pour la description du contexte dans un environnement sensible au contexte.

1.3.3.4 Couche de dissémination du contexte

Cette couche permet la transmission des informations contextuelles au système. Dans les applications mobiles, ces informations sont généralement distribuées sur différents lieux géographiques et proviennent de plusieurs types de dispositifs. Plusieurs auteurs comme [Bauer et al., 1998] ont étudié les mécanismes pour notifier au système les changements du contexte. Plus précisément, ces mécanismes se basent sur les fonctions de gestion des requêtes de notification. Dans ce cas-là, le système peut demander un accès direct à une information contextuelle (le mode *pull*) mais il peut également s'abonner pour recevoir tous les changements des valeurs de cette information (le mode *publish/subscribe*). Ces deux modes assurent un moyen de communication efficace et transparent pour la dissémination des valeurs de contexte au système.

Le concept de « dissémination » a été largement adopté dans le domaine des applications mobiles. Dans notre travail, la dissémination désigne la transmission des informations de contexte sans pour autant se référer à des dispositifs de captures distribués. En outre, pour la simplicité des communications, nous favorisons l'utilisation du mode d'accès direct aux informations de contexte afin de garantir la sensibilité au contexte dans un service dédié à la coopération.

1.3.4 Définition, finalité et mécanismes de l'adaptation au contexte

Nous nous intéressons dans cette section à l'adaptation, qui constitue le dernier maillon de la chaîne de sensibilité des systèmes au contexte. Tout d'abord, nous définissons la notion d'adaptation. Ensuite, nous énumérons les finalités de l'adaptation au contexte. Enfin, nous proposons une revue des techniques d'adaptation existant dans la littérature.

1.3.4.1 Définition de l'adaptation

Le dictionnaire Larousse définit le verbe *adapter* comme « rendre un dispositif apte à assurer ses fonctions dans des conditions particulières ou nouvelles ». Ainsi découle la définition de l'adaptabilité comme la qualité de ce qui peut être adapté et de ce qui peut s'adapter. En considérant la définition sémantique du terme, une adaptation correspond au processus de

modification du système, indispensable pour permettre un fonctionnement adéquat dans un contexte donné. Le terme *adéquat* implique que le système correspond parfaitement à ce que l'on attend dans ce contexte précis.

Cette définition suppose la présence d'un système (le dispositif), qui réalise un objectif particulier (ses fonctions). La façon dont le système réalise cet objectif dépend de l'environnement, du contexte dans lequel il se trouve. Lorsque le contexte est modifié, le système doit être ajusté pour qu'il soit à même de réaliser son objectif au mieux dans ces nouvelles conditions. Une adaptation est ainsi une *modification* d'un système, suite à la *décision* d'un ajustement en réponse à *un changement dans son contexte* [David, 2005].

Plusieurs auteurs comme [Behlouli et al., 2006] affirment que d'une manière générale l'adaptation d'une application suit un processus à trois étapes successives : une étape de déclenchement, une étape de décision et une dernière étape de réalisation.

- **L'étape de déclenchement** consiste à détecter et à notifier un changement. Ce déclenchement peut être effectué par différents acteurs (par exemple un utilisateur, ou une entité logicielle) et à différents moments de cycle de vie d'une application (par exemple au déploiement, ou au moment de l'exécution),

- **L'étape de décision** consiste à déterminer les modifications qui doivent être effectuées pour réagir aux changements détectés dans la première étape. Ces choix peuvent être exécutés par différents acteurs (par exemple, un programmeur, un module logiciel, *etc.*), concerner différents sujets à adapter (par exemple, un ensemble de codes Java) et suivre différentes règles ou stratégies d'adaptation,

- **L'étape de réalisation** concerne tous les moyens qui vont être mis en œuvre pour appliquer la décision prise à l'étape précédente. La réalisation peut être exécutée par différents acteurs (par exemple, un intergiciel) et selon différents mécanismes. La réalisation s'opère sur les sujets à adapter qui ont été définis dans l'étape de décision.

1.3.4.2 Finalités de l'adaptation au contexte

L'adaptation au contexte a été introduite dans plusieurs domaines de l'informatique : l'informatique ubiquitaire, l'informatique pervasive et l'informatique sensible au contexte. Dans ce qui suit, nous présentons la finalité de l'adaptation du contexte vu sous l'angle de ces différents domaines :

- **L'informatique ubiquitaire** (*ubiquitous computing*) vise à faire disparaître l'informatique traditionnelle au profit d'un espace informatisé au service des activités humaines [Weiser, 1991]. [Weiser, 1991] était le premier à définir le concept d'informatique ubiquitaire (ou ambiante) en comparant, à son époque, le passé de l'informatique (une machine *mainframe*) pour plusieurs utilisateurs, le présent de l'informatique (un ordinateur pour un utilisateur) et le futur de l'informatique (plusieurs ordinateurs distribués dans l'environnement de l'utilisateur). Aujourd'hui ce phénomène prend beaucoup d'ampleur dans notre société et le nombre de publications scientifiques en est le témoin.

La finalité de l'adaptation au contexte, dans le domaine de l'informatique ubiquitaire, consiste à doter les utilisateurs des applications adaptées à la diversité des dispositifs qui les entourent.

- **L'informatique pervasive** (*pervasive computing*) est définie comme le moyen qui tend à simplifier la vie des utilisateurs par l'intermédiaire d'environnements digitaux qui capturent, s'adaptent, et répondent aux besoins de l'utilisateur [Saha and Mukherjee, 2003]. Les systèmes pervasifs ont pour objectif de rendre l'information partout et à tout moment. Pour ce faire, ils jouissent de l'ensemble des ressources disponibles de façon transparente pour l'utilisateur.

  La finalité de l'adaptation au contexte, dans le domaine de l'informatique pervasive, est d'utiliser le plus de ressources possible localisées dans l'environnement de l'utilisateur afin de lui présenter une application adaptée d'une manière transparente.

- **L'informatique sensible au contexte** (*context-aware computing*) caractérise un paradigme de l'informatique dans lequel les applications perçoivent la situation de l'utilisateur dans son environnement et adaptent ainsi leurs comportements [Abowd et al., 1999].

  La finalité de l'adaptation au contexte, dans le domaine de l'informatique sensible au contexte, est de fournir aux utilisateurs des applications adaptées qui prennent en compte le changement de leur contexte d'utilisation.

La présente étude menée sur les finalités de l'adaptation au contexte a souligné que l'objectif poursuivi par l'adaptation dépend du domaine de son application. Nous nous positionnons dans le domaine de l'informatique sensible au contexte avec une certaine vision métier de l'adaptation à savoir l'adaptation des services pour la coopération interentreprises.

1.3.4.3 Mécanismes d'adaptation existants

Il existe plusieurs techniques qui permettent à un système d'être adapté ou de s'adapter aux variations pertinentes du contexte. Dans ce qui suit, sans prétendre à l'exhaustivité, nous décrivons les techniques les plus connues.

1.3.4.3.1 Réflexivité

Le concept de réflexivité a été introduit par Smith dans sa thèse de doctorat en 1982 [Smith, 1982], et depuis il a été largement utilisé pour l'adaptation des applications à base de composants. Selon [Smith, 1982], la réflexivité représente la capacité d'un système à s'observer et à agir sur lui-même durant son exécution [Cazzola et al., 1999]. Un système ou un composant réflexif doit présenter deux niveaux : un niveau de base et un niveau « méta ». Le niveau de base comporte le code métier du système tandis que le niveau « méta » contient une représentation abstraite du système. Ces deux niveaux sont fortement connectés, c'est-à-dire que tout changement dans l'un d'eux se répercute sur l'autre.

Les interactions entre les deux niveaux se font dans les deux sens. Ainsi, [Smith, 1982] identifie deux aspects : l'introspection et l'intercession. L'introspection est la capacité d'un

système à examiner son propre état et l'intercession, est sa capacité à modifier son propre état d'exécution ou d'altérer sa propre interprétation.

Grâce à cette séparation de préoccupations (niveau métier et niveau « méta »), la réflexivité permet de modifier le comportement d'un système en réponse aux changements de son environnement. Cependant, cette séparation engendre un surcoût inévitable en temps d'exécution.

1.3.4.3.2 Adaptation contrôlée par les politiques

Une politique est un moyen pour définir et modifier l'organisation et le comportement d'un système. Elle se compose généralement de règles, dérivant la description d'une stratégie et spécifiant au système les actions à entreprendre pour répondre à une situation donnée. Plus formellement, une politique peut être définie selon deux perspectives [Westerinen, 2001] :

– Des objectifs permettant de guider et déterminer les actions présentes et futures à exécuter au sein du système,

– Un ensemble de règles permettant d'administrer, superviser et commander l'accès aux ressources du système.

Cette définition met en évidence deux niveaux d'abstraction pour introduire une spécification d'une politique en fonction de l'utilisateur concerné. Au niveau métier, la politique simplifie le travail des analystes métier en définissant leurs besoins en tant qu'objectifs globaux. Quant au niveau technique, la politique est considérée comme un ensemble de règles concernant les détails de l'implémentation de la stratégie pour atteindre les objectifs des intervenants métier.

Afin de gérer un système à base de politiques, la première étape consiste à spécifier l'ensemble de règles qui seront appliquées dans des situations prédéfinies. Ces règles peuvent correspondre à des actions correctives qui devraient être faites dans certaines conditions ou encore ajouter des contraintes sur certains éléments du système.

L'approche d'adaptation des systèmes par les politiques a été utilisée dans différents domaines comme les bases de données actives et distribuées [Baral and Lobo, 1996] et les applications du Web sémantique [Chen et al., 2008].

1.3.4.3.3 Adaptation par tissage d'aspects

La programmation orientée aspect est un nouveau paradigme de programmation qui permet de séparer l'implémentation des considérations techniques (ou non fonctionnelles) des descriptions métier (ou fonctionnelles) d'un système. Ce paradigme est complémentaire aux langages objets et se déploie progressivement dans les entreprises [Baltus, 2001]. Son principe de base est la séparation des préoccupations (*Separation of Concerns*) dont l'objectif est de permettre la maîtrise de systèmes logiciels en adressant, de façon isolée, les diverses préoccupations qui les composent.

En se basant sur ce principe, la programmation orientée aspect permet d'introduire chaque préoccupation séparément et de définir leurs règles d'intégration afin de les combiner pour former le système final.

De nouveaux concepts ont été développés et tout un vocabulaire spécifique a été défini pour la programmation orientée aspect. Ces concepts ont été formulés par Gregor Kiczales et son équipe [Kiczales et al., 2001a] :

- **L'aspect :** regroupe un code (*advice*) que l'on souhaite greffer (ou tisser) au sein d'un code source grâce à un programme appelé tisseur (*weaver*) ainsi que des points de coupure ou d'action (*pointcuts*). Il correspond souvent aux exigences non fonctionnelles d'un système comme la performance, la sécurité *etc.*
- **Le greffon (*advice*) :** c'est le code qui sera activé à un certain point d'exécution du système, précisé par un point de jonction (*join point*). Il existe trois types *d'advice* : les « *before advices* », les « *around advices* » et les « *after advices* ». Comme leurs noms l'indiquent, les « *before advices* » s'exécutent avant que les points de jonction ne soient exécutés, les « *around advices* » permettent d'exécuter du code avant et après les points de jonction, et les « *after advices* » s'exécutent uniquement après l'exécution des points de jonction.
- **Les points de jonction (*join points*) :** correspondent à des points précis dans l'exécution d'un programme tels que l'appel d'une méthode et l'affectation d'une variable, où on peut insérer un *advice*.
- **Les points de coupure ou d'action (*pointcuts*) :** un *pointcut* définit un ensemble de points de jonction (*join points*) satisfaisant aux conditions d'activation de l'aspect. En d'autres termes, un *pointcut* définit les différents emplacements où un *advice* particulier sera inséré.
- **Le tisseur (*weaver*) :** correspond à un programme qui permet de tisser le code des aspects dans le code des méthodes des classes. Il est important de noter que selon les tisseurs, ce tissage peut avoir lieu à la compilation ou à l'exécution (appelé aussi tissage dynamique). La Figure 1.24 illustre les deux types de tisseurs ainsi que leur mode d'emploi.

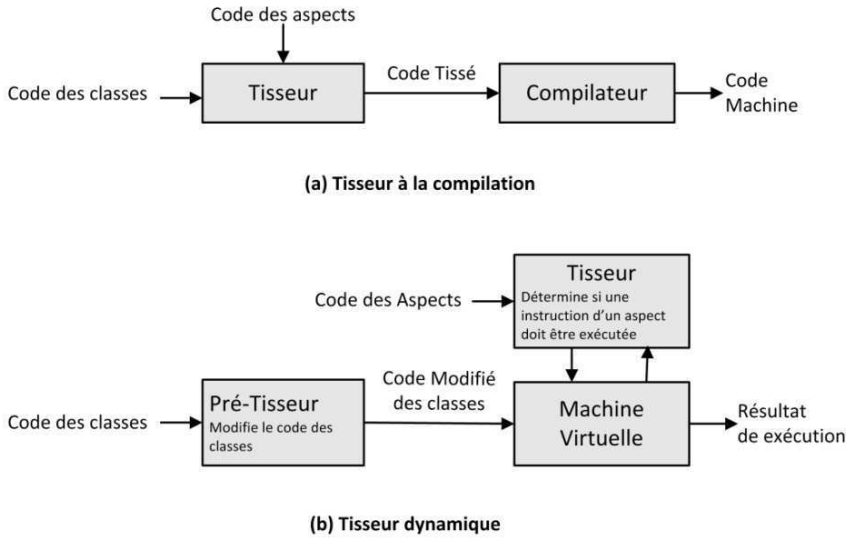


Figure 1.24. Deux types de tisseurs supportés par la programmation orientée aspect

L'implémentation d'une application orientée aspect peut se réaliser en trois étapes :

(i) L'identification des composants et des aspects d'un système. Ainsi, on identifie chaque préoccupation à part (fonctionnelles ou non fonctionnelles).

(ii) L'implémentation de chaque préoccupation. Chaque préoccupation sera codé séparément que ce soit dans un composant ou dans un aspect. Le programmeur doit définir les règles d'intégration de l'aspect avec les composants concernés.

(iii) L'intégration des composants et des aspects se fait grâce à un tisseur.

Le mécanisme de tissage d'aspects peut être utilisé comme une technique d'adaptation efficace. Le principal atout de ce type d'adaptation est le tissage dynamique des aspects grâce aux tisseurs dynamiques qui sont capables d'appliquer les aspects pendant l'exécution du programme. Ainsi, les aspects peuvent être ajoutés, supprimés ou modifiés à chaud pendant l'exécution du programme.

Nous proposons d'étudier dans la section suivante les différents travaux d'adaptation au contexte dans le domaine de service Web.

1.3.5 Approches d'adaptation au contexte pour les services Web

Compte tenu de la diversité des travaux de recherche dans le domaine de l'adaptation des services Web au contexte, nous proposons un cadre de référence permettant de les analyser. Le cadre de référence illustré en Figure 1.25 catégorise les travaux d'adaptation en faisant un lien entre les formes d'adaptation (*Quoi ?*), les techniques d'adaptation utilisées (*Comment ?*) et le moment d'adaptation (*Quand ?*).

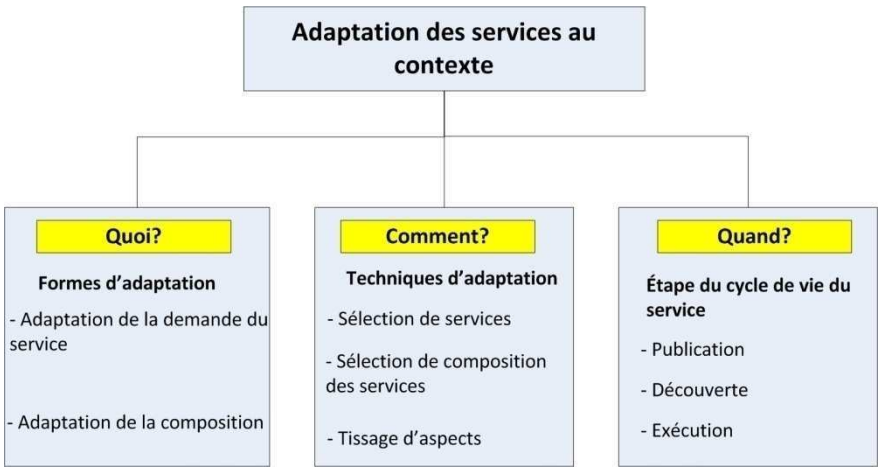


Figure 1.25. Cadre de référence pour l'adaptation des services au contexte

- **Les formes d'adaptation (*Quoi ?*)**

Dans les travaux relatifs à l'adaptation des services Web, il existe deux formes d'adaptation complémentaires : l'adaptation de la demande du service et l'adaptation de la composition.

– Adaptation de la demande du service : consiste à mettre en œuvre une application adaptée qui utilise les besoins exprimés par le client. Certains travaux qui appartiennent à ce courant de recherche proposent au client des solutions afin de trouver les services

Web qui conviennent le mieux à sa demande selon un contexte donné et ceci en utilisant la technique de sélection de services Web.

- Adaptation de la composition : établie par l'intermédiaire de deux techniques d'adaptation (sélection de composition et ajout de nouvelles préoccupations). La sélection de composition consiste à sélectionner parmi un ensemble de compositions prédéfinies, la composition qui répond au mieux au contexte du client. Quant à l'ajout de nouvelles préoccupations, il s'agit d'adapter le service par l'ajout de nouvelles fonctionnalités pour répondre au changement de contexte.

- **Les techniques d'adaptation utilisées (*Comment ?*)**

Dans le domaine des services Web, plusieurs auteurs comme [Agarwal et al., 2005], [Maamar et al., 2007a], [Bettini et al., 2007], [Sheng et al., 2009] soulignent que les techniques d'adaptation les plus utilisées sont les techniques de sélection de services ou encore de sélection de composition des services. La technique de tissage d'aspects (ou notamment tissage de nouvelles préoccupations), commence aussi à intéresser la communauté de recherche dans ce domaine [Courbis and Finkelstein, 2005], [Charfi and Mezini, 2007].

- **Le moment d'adaptation (*Quand ?*)**

Le moment d'adaptation inclut essentiellement les étapes du cycle de vie du service. En examinant le cycle de vie d'un service, nous avons pu déterminer les étapes où l'adaptation peut être mise en œuvre. En effet, l'adaptation peut être intégrée dans les trois étapes suivantes :

- L'étape de publication des services : pendant la publication des services (au moment de la conception), il est nécessaire d'améliorer leurs descriptions classiques, faite à l'aide de WSDL ou à l'aide d'un autre langage de description, afin d'y intégrer le (ou les) paramètres de contexte.
- L'étape de découverte des services : pendant l'étape de découverte, il est possible d'intégrer une forme d'adaptation en filtrant dans le registre de services Web, ceux correspondant au mieux, tant aux besoins fonctionnels du client, qu'aux besoins en termes d'adaptation au contexte d'utilisation (contexte du client, *etc.*).
- L'étape d'exécution des services : pendant l'étape d'exécution, il s'agit de capter les informations de contexte courant afin d'adapter convenablement les services.

Nous présentons dans la prochaine section les travaux mettant en œuvre l'adaptation des services Web au contexte en nous situant dans le cadre de référence proposé (*Quoi*, *Comment*, et *Quand*).

1.3.5.1 Approches d'adaptation de la demande de services

Dans le travail de [Suraci et al., 2007], les auteurs présentent une approche orientée contexte pour la découverte de services Web. Leur travail s'inscrit dans le cadre du projet DAIDALOS II (*Designing Advanced network Interfaces for the Delivery and Administration of Location independent, Optimised personal Services*), issu d'un programme de recherche européen. Les auteurs considèrent la découverte contextuelle des services comme la capacité d'utiliser

l'information du contexte pour découvrir les services les plus pertinents pour l'utilisateur. La Figure 1.26 illustre l'architecture de ce travail.

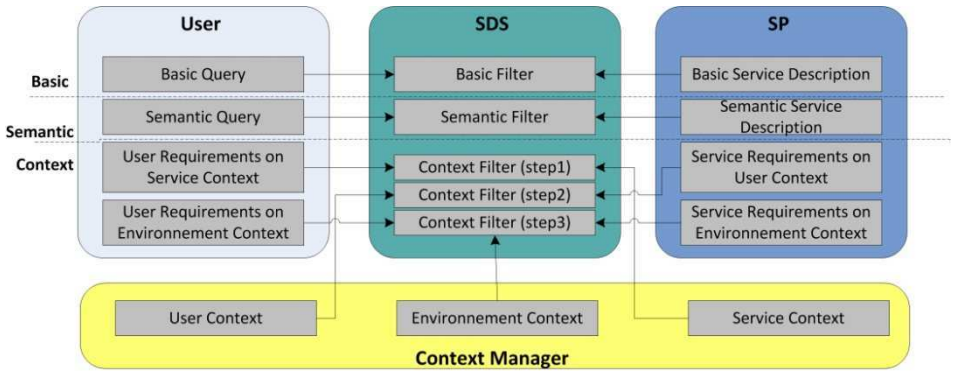


Figure 1.26. Architecture de découverte de services sensibles au contexte [Suraci et al., 2007]

- **Fondements de base**

Le processus d'adaptation proposé se déroule comme suit : le fournisseur publie ses services sur un serveur auquel l'utilisateur envoie sa requête de service. Suivant cette architecture, le fournisseur doit publier dans le gestionnaire de contexte (*Context Manager*) le contexte auquel le service peut répondre. Ce contexte inclut les conditions de l'utilisation du service en fonction de la description de l'utilisateur et de son environnement. La description du service Web publié peut se faire à deux niveaux. La première description est une description basique (*Basic Service Description*), exprimée dans un langage de bas niveau tel que XML ou WSDL. La seconde description est une description sémantique (*Semantic Service Description*) exprimée en OWL-S. Les deux descriptions du service (basique et sémantique), les conditions d'utilisation du service (*Service Requirements on User Context* et *Service Requirements on Environment Context*), et la référence du contexte du service sont publiés dans un module du serveur nommé *Service Provider* (*SP)*.

Étant donnée la finalité de leur travail, l'adaptation dans le domaine des applications mobiles, les auteurs proposent deux catégories de contexte à savoir : le contexte de l'utilisateur et le contexte de son environnement. La définition du contexte englobe toutes les informations pour lesquelles l'utilisateur souhaite que le service puisse lui répondre de manière adaptée. Par exemple, le contexte de l'utilisateur peut contenir sa position géographique, des informations concernant son dispositif, comme la charge de la batterie, ou la taille de l'écran. Quant aux informations relatives à l'environnement, les paramètres de contexte incluent la température, le taux de pollution, *etc.*

En somme, la définition du contexte dans ce travail rassemble tous les éléments qui peuvent être détectés de manière automatique par un capteur physique.

Avant de proposer une requête, l'utilisateur doit enregistrer son contexte (*User Context*) dans le gestionnaire de contexte (*Context Manager*). La requête de l'utilisateur est composée de deux parties : la requête basique (*Basic Query*) et la requête sémantique (*Semantic Query*). La requête basique est exprimée en langage de recherche de bas niveau et la requête sémantique est exprimée en langage de requête sémantique de haut niveau. Ensuite, l'utilisateur doit

envoyer au serveur les requêtes et le pointeur sur son contexte. Ces informations sont stockées dans un module du serveur nommé *User*.

- **Mise en œuvre**

Après la réception de la requête utilisateur, le serveur de recherche de services (*SDS*) invoque le moteur de filtrage de services qui effectue la recherche de services en trois phases : le filtre de base, le filtre sémantique et le filtre de contexte (Figure 1.26).

- Le filtre de base (*Basic Filter*) : utilise la requête basique de l'utilisateur et la description basique des services. Chaque description est comparée à la requête par l'intermédiaire du protocole *Service Location Protocol* (SLP),
- Le filtre sémantique (*Semantic Filter*) : compare la requête sémantique de l'utilisateur et la description sémantique des services sélectionnés lors de la première étape. Le résultat de ce filtre est une liste de services qui répondent exactement aux besoins sémantiques désirés par l'utilisateur,
- Le filtre de contexte (*Context Filter*) : inclut trois sous étapes : (i) le serveur de recherche de service (SDS) compare le contexte du service et les attentes de l'utilisateur en termes de contexte de service. (ii) Le SDS compare le contexte de l'utilisateur avec les conditions d'utilisation du service. (iii) Le SDS compare le contexte de l'environnement avec les conditions d'utilisation du service et les attentes de l'utilisateur.

- **Évaluation**

L'originalité de ce travail réside dans la représentation du contexte auquel peut répondre le service grâce à l'extension de *OWL-S*. Le fait d'intégrer le contexte dans la définition du service est, à notre sens, une étape fondamentale lors de la mise en oeuvre de l'adaptation dans le domaine des services Web.

Le travail de **[Keidl and Kemper, 2004]** consiste à adapter l'exécution des services Web selon le contexte intégré dans le message SOAP. En effet, les informations de contexte sont échangées dans l'entête des messages SOAP (voir Figure 1.27).

```
<env:Envelope xmlns:env="http://schemas.xmlsoap.org/soap/envelope/">
  <env:Header>
    <Context xmlns="http://sg.fmi.uni-passau.de/context">
      <Location>
        <address useType="Office">
          <addressLine keyName="Street" keyValue="60">Innstrasse 33</addressLine>
          <addressLine keyName="City" keyValue="40">D-94032 Passau</addressLine>
        </address>
      </Location>
      <Client>
        <DeviceDefaults>http://example.com/context/device/PDA</DeviceDefaults>
        <Hardware>
          <ScreenSize>320x200</ScreenSize>
          <IsColorCapable>Yes</IsColorCapable>
        </Hardware>
      </Client>
    </Context>
  </env:Header>
  <env:Body>
    <!-- serialized object data -->
  </env:Body>
</env:Envelope>
```

Figure 1.27. Structure du message SOAP avec des informations de contexte [Keidl and Kemper, 2004]

- **Fondements de base**

Les catégories de contexte définies par [Keidl and Kemper, 2004] regroupent toutes les informations concernant le client qui sont nécessaires au processus d'adaptation des services Web. Cinq catégories d'informations de contexte sont proposées : la localisation, le dispositif, la présentation du résultat, l'utilisateur et les préférences de connexion. La localisation permet de situer géographiquement le client par l'intermédiaire d'une description présentée explicitement par ce dernier telle qu' « au travail » ou « à la maison ». Le dispositif décrit le terminal utilisé en termes de caractéristiques matérielles (comme le type de processeur, la mémoire libre, *etc.*) et les caractéristiques logicielles (comme le système d'exploitation, le type de navigateur, *etc.*). Concernant la présentation du résultat, elle décrit le format à utiliser pour présenter le résultat du service Web au client et respecter par la suite ses préférences et ses restrictions d'affichage. Quant à l'utilisateur, cette catégorie de contexte regroupe les informations spécifiques à l'utilisateur (comme son nom, son adresse électronique, *etc.*).

La dernière catégorie de contexte prévue par les auteurs concerne les préférences de connexion exprimées par l'utilisateur. Ces dernières spécifient les paramètres de connexions aux services Web (comme l'encodage du résultat, le besoin d'encryptage, *etc.*).

- **Mise en œuvre**

Pour mettre en œuvre l'adaptation des services à ces catégories de contexte [Keidl and Kemper, 2004] proposent un processus d'adaptation réalisé grâce à un module appelé le gestionnaire de contexte (*Context Manager*). Ce processus s'effectue selon quatre étapes (voir Figure 1.28) :

- Pré-traitement de la requête : la requête SOAP est pré-traitée par le gestionnaire de contexte (*Context Manager*) plus précisément par le module *Pre-Process Request* afin de récupérer le contexte inclus dans la requête,
- Sélection du service adéquat : le gestionnaire de contexte appelle le module *Invocation Manager*, dont le rôle est de sélectionner un service Web en adéquation avec le contexte défini dans la requête,
- Transmission de la réponse du service : la réponse du service sera transmise au gestionnaire de contexte pour s'assurer de son adéquation avec le contexte du client,
- Envoi du résultat au client : le résultat du service est envoyé au module de traitement SOAP (*SOAP Request Processing*) afin d'être traduit dans ce langage et être renvoyé au client.

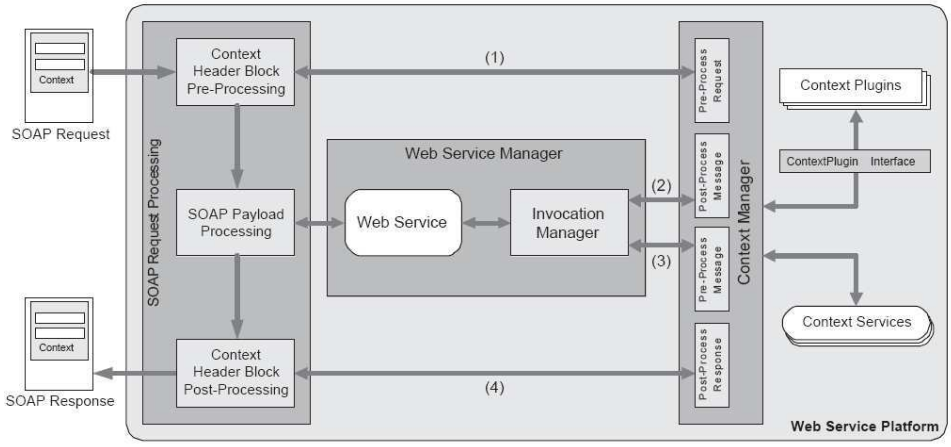


Figure 1.28. Modules dédiés à l'adaptation des services Web au contexte [Keidl and Kemper, 2004]

- **Évaluation**

L'originalité du travail proposé par [Keidl and Kemper, 2004] consiste à utiliser les messages SOAP pour représenter les informations de contexte. Ces informations sont par la suite traitées par le gestionnaire de contexte pour sélectionner le service Web qui correspond au mieux au contexte du client. Nous considérons que l'utilisation des standards de services Web (comme SOAP) dans le processus d'adaptation est particulièrement intéressant et permet de capitaliser sur des approches et des outils déjà développés. Cependant, la proposition des auteurs ne présente pas en détail le processus de sélection du service Web. Ce qui pose des questions concernant la mise en œuvre de ce processus, l'une d'entre elles consiste à s'interroger sur la manière de décrire les services pour pouvoir les sélectionner par la suite. S'agit-il d'une description classique ou d'une description annotée par les informations de contexte ? De la même façon on pourrait se poser la question sur le processus de récupération du contexte client à partir de la requête. Autant de questions qui méritent des réponses.

1.3.5.2 Approches d'adaptation de la composition des services

Dans la suite de cette section, nous allons présenter les approches d'adaptation de la composition des services Web et ceci suivant les deux techniques d'adaptation utilisées (sélection de composition et ajout de nouvelles préoccupations).

1.3.5.2.1 Adaptation par sélection de composition

[Qiu et al., 2006], **[Qiu et al., 2007]** proposent une méthode de composition des services Web sensible au contexte. La particularité de ce travail est qu'il se base sur la planification. Ainsi,

il permet une planification globale[9] (*Global Planning*) selon la requête de l'utilisateur, aussi qu'une optimisation locale selon les changements du contexte.
Dans leur travail, les auteurs proposent une extension d'OWL-S, nommée OWL-SC (*OWL-S Context*) permettant d'intégrer le contexte à la description des services Web. De plus, ils présentent une méthode de planification basée sur le contexte pour la composition des services.

- **Fondements de base**

Dans [Qiu *et al.*, 2006], les auteurs distinguent trois catégories de contexte : le contexte de l'utilisateur (*U-Context*), le contexte du service Web (*W-Context*) et le contexte de l'environnement (*E-Context*). Chaque catégorie de contexte est représentée par une ontologie et est intégrée aux ontologies existantes d'OWL-S.

- U-Context : cette catégorie de contexte spécifie les informations concernant l'utilisateur. Les auteurs identifient deux types d'informations : le contexte statique de l'utilisateur (son profil, ses intérêts, ses préférences) et le contexte dynamique de l'utilisateur (sa localisation, son activité courante et la tâche qu'il est en train de réaliser).
- W-Context : cette catégorie de contexte regroupe les informations non fonctionnelles d'un service (son prix, la durée de son exécution, son degré de confiance).
- E-Context : cette catégorie de contexte comprend des informations concernant l'environnement de l'utilisateur (le temps, la date, *etc.*).

- **Mise en œuvre**

L'architecture d'adaptation proposée par [Qiu et al., 2007] est illustrée dans la Figure 1.29.

[9] La planification (*Automated planning*) est une discipline de l'intelligence artificielle qui vise le développement d'algorithmes pour produire des plans, qui vont être exécutés par des agents.

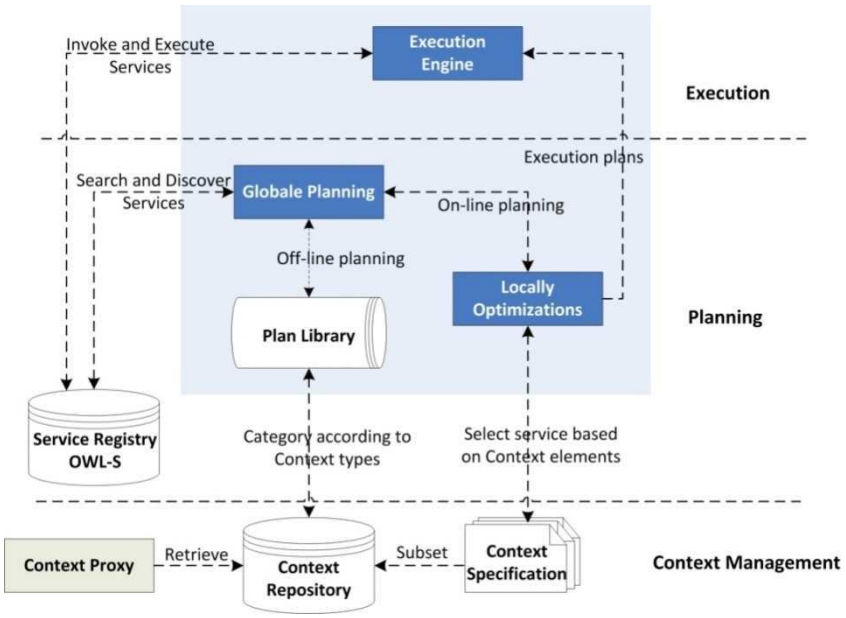


Figure 1.29. Architecture de composition des services adaptables [Qiu et al., 2007]

Cette architecture comporte trois couches de bases :

- La gestion du contexte : c'est la première couche, elle comporte trois modules qui permettent de gérer le contexte : le *Context Proxy*, le *Context Repository* et le *Context Specification.* Le *Context Proxy* génère les conditions du contexte qui doivent être utilisées lors de la requête de la composition. Le *Context Repository* enregistre les représentations des contextes (*U-Context*, *W-Context* et *E-Context*) formalisées sous le format OWL-SC (voir Figure 1.30). Quant au *Context Specification*, il représente des règles prédéfinies associées aux conditions de contexte pour découvrir les services.

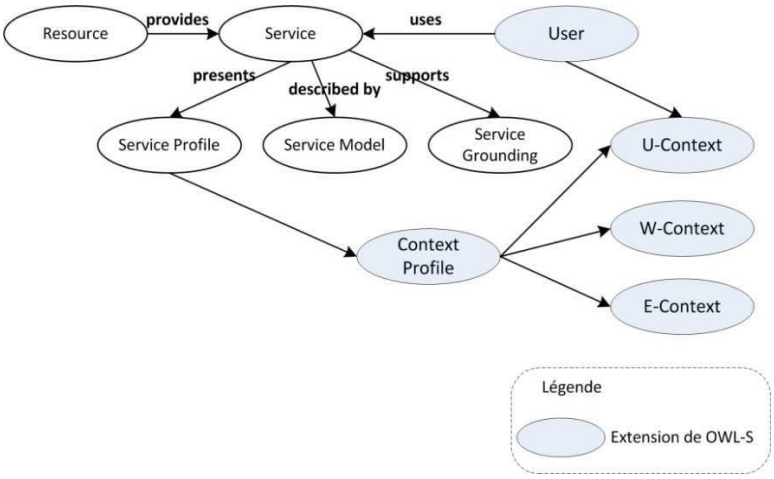


Figure 1.30. OWL-SC pour la description des services Web [Qiu et al., 2007]

- La planification : cette couche permet de réceptionner la requête de l'utilisateur ainsi que les informations du contexte, afin de définir une séquence d'actions à exécuter.

Pour ce faire, le module *Globale Planning* consulte la base de données de plans (*Plan Library*) afin de sélectionner un *plan template*. La base de données de plans stocke un ensemble de schémas de plans, représentés par des diagrammes d'états UML, définis au moment de la conception. Le module *Globale Planning* se charge par la suite de générer pour le *plan template* sélectionné le graphe dirigé acyclique (*DAG : Directed Acyclic Graph*) en utilisant un algorithme d'optimisation proposé par les auteurs dans [Qiu et al., 2006]. Ensuite, il s'agit d'identifier les services Web candidats et de générer les plans exécution. S'il existe plusieurs plans exécution, le module *Locally Optimizations* sélectionne le plan adéquat selon le contexte.

- L'exécution : une fois le plan choisi est instancié par les services Web, il sera exécuté par le moteur d'exécution (*Execution Engine*).

- **Évaluation**

Le travail introduit par [Qiu et al., 2006], [Qiu et al., 2007] propose une méthode de planification pour composer des services Web adaptés au contexte. L'extension de l'ontologie OWL-S pour modéliser le contexte du service, du client et de l'environnement est une approche intéressante. Cette orientation permet d'une part de prendre en compte une large définition du contexte et d'autre part, de raisonner sur les éléments de contexte. De plus, la génération des plans au moment de la conception et d'en proposer par la suite des optimisations est une solution pertinente. Nous allons utiliser certaines propositions de ces travaux dans le cadre du chapitre 4 de notre proposition.

1.3.5.2.2 Adaptation par tissage d'aspects

Les travaux de **[Charfi and Mezini, 2007]**, **[Charfi et al., 2007a]**, **[Charfi, 2008]** s'intéressent à l'introduction de la programmation orientée aspect dans les processus métiers. Leur analyse se fonde sur le constat de deux lacunes du langage BPEL : d'une part les compositions manquent de modularité, notamment en ce qui concerne les préoccupations « éparpillées », et d'autre part les compositions demeurent figées du fait que BPEL spécifie des informations statiques.

- **Fondements de base**

Pour apporter des solutions à ces limitations, une extension de BPEL est réalisée en introduisant des mécanismes orientés aspect. Le résultat est l'élaboration du langage AO4BPEL (*Aspect Oriented for Business Process Execution Language*) offrant plus de modularités pour la spécification des compositions des services Web et un support d'adaptation dynamique des compositions.

Un moteur BPEL spécifique a été également développé pour la gestion des aspects, ainsi que pour l'intégration des services middlewares (sécurité, authentification, transaction). Le langage AO4BPEL possède un modèle de *join point*, un langage de *pointcut* et un langage d'*advice*. Il a la particularité d'être entièrement rédigé en XML. Au sein du langage AO4BPEL, chaque activité est un *join point* potentiel et il existe également des *join points* liés aux événements d'envoi et de réception de messages SOAP. Étant donné que le langage BPEL est spécifié en XML, le langage de *pointcut* est basé sur XPath qui permet de faire des requêtes sur des documents XML. Un *pointcut* consiste en une collection de *join points*, c'est-

à-dire d'expressions XPath qui référencent des activités BPEL et pouvant être composées à l'aide des opérateurs d'union ou d'intersection. Pour exprimer les *advices*, les auteurs utilisent des instructions BPEL enrichies par des mots-clés (*Proceed*) et un mécanisme de réflexion (*ThisJPInVariable*). Les auteurs argumentent ce choix par le fait qu'ils ne souhaitent pas corrompre la portabilité du BPEL. Le contenu d'un aspect AO4BPEL consiste ainsi en un *pointcut* et une activité BPEL devant être appelée à l'endroit du *pointcut*.

Pour palier au manque d'expressivité du langage BPEL pour l'implémentation des *advices*, ils évoquent la possibilité de faire des appels vers un « service Web d'infrastructure » qui contiendra la logique de l'*advice*. Les auteurs ont aussi travaillé sur la possibilité de faire intervenir un aspect de sécurité dans les compostions de services Web [Charfi et al., 2006] en ayant également recours à l'appel à des services Web d'infrastructure. L'idée de ce travail est de rajouter des appels vers un service qui s'occupe de gérer la sécurité et d'autres paramètres de qualité de services.

- **Mise en œuvre**

AO4BPEL s'appuie sur le moteur BPWS4J d'IBM. Le moteur AO4BPEL étend le moteur BPWS4J avec un composant de déploiement d'aspects et un autre composant d'exécution d'aspects. Le composant de déploiement offre la possibilité à un administrateur de manipuler les aspects et de visualiser les interactions de ces derniers avec le processus, tandis que le composant d'exécution construit autour de l'interprète BPEL du moteur original, vérifie l'exécution potentielle d'aspects.

- **Évaluation**

Ce travail souligne que le concept de programmation par aspects est pertinent pour l'adaptation des services composites. Le langage AO4BPEL s'inspire du langage AspectJ [Kiczales et al., 2001a] (pour la spécification du type d'*advice*) et du langage XPath (pour le *pointcut*). Il étend le modèle de *join point* pour couvrir des événements de type envoi et réception de messages SOAP. Cependant, ce travail requiert un moteur BPEL spécifique, ce qui rend la proposition inexploitable dans d'autres environnements d'exécution. De plus, l'utilisation d'aspects modularisant du code BPEL n'apporte pas une forte valeur ajoutée à la solution, étant donné que BPEL est un langage dédié[10]. Ainsi, il n'est pas possible de rajouter de nouvelles préoccupations sauf en développant de nouveaux services Web d'infrastructure (*Web Service Infrastructure*) qui intègreront ces préoccupations. Ainsi, les auteurs ajoutent une couche supplémentaire qui alourdit l'architecture proposée.

[10]Un langage dédié, ou *domain-specific language* (DSL), est un langage qui a été conçu pour répondre à un type donné de problème, ou à un domaine d'application spécifique, contrairement aux langages généralistes comme Java par exemple.

Ce choix d'implémentation n'est pas simple à réaliser et impose un couplage fort entre le moteur d'exécution et les services Web contenant la logique des préoccupations non fonctionnelles. Enfin, AO4BPEL n'intègre pas la gestion des paramètres de contexte et ne s'intéresse pas aux actions d'adaptation vis-à-vis des changements de ces paramètres.

Dans ses travaux de thèse **[Hmida et al., 2005] [Hmida et al., 2007]** a traité les problématiques de l'adaptation dynamique du service ainsi que l'interfacage correcte entre le client et le service. Les auteurs ont proposé une approche basée sur la programmation orientée aspect qui permet de changer le comportement d'un service Web au moment de l'exécution [Hmida et al., 2005],[Tomaz et al., 2006], [Hmida et al., 2007].

- **Principes de base**

Le travail proposé consiste en une nouvelle approche nommée *Aspect Service Weaver* (ASW). ASW s'interpose entre le client (l'appelant) et le service (l'appelé) et utilise le tissage d'aspects (avant ou après l'invocation du service). Ce tissage permet d'injecter ou de ne pas injecter de nouveaux comportements pour le service Web d'origine au moment de son exécution sans affecter les autres applications qui partagent ce même service. [Hmida et al., 2005] définissent les *join points* au sein de l'interface du service Web (WSDL) en utilisant XPath comme langage de *pointcut*. Les *advices* sont représentées par des services appelés *services conseils*. Ces derniers sont des services Web qui implémentent les nouveaux comportements des services. L'ASW joue le rôle d'intermédiaire entre le client et le service Web.

- **Mise en œuvre**

L'approche proposée par [Hmida et al., 2005],[Tomaz et al., 2006], [Hmida et al., 2007], agit dans deux cas d'utilisation différents. Le premier cas consiste à modifier dynamiquement un service Web élémentaire tandis que le deuxième se préoccupe de modifier un processus BPEL contenant un ensemble de services Web. Dans le premier cas, l'ASW intercepte les messages SOAP avant qu'ils arrivent au service et vérifie pendant l'exécution s'il y a une règle d'invocation d'aspects (*aspect service*) qui s'applique sur la méthode invoquée. S'il en trouve, l'ASW dirige le message capturé vers le tisseur d'aspect en sélectionnant le *service conseil* adéquat.

Dans le cas d'un service Web composite (processus BPEL), l'ASW examine l'exécution du processus BPEL. Il vérifie avant l'exécution de chaque activité BPEL s'il existe des règles d'invocations d'aspects (*aspect service*) définies pour l'activité en cours. S'il trouve, l'ASW invoque le ou les *service(s) conseil(s)* associé(s). Les *join points*, dans ce cas, sont définis au sein du processus BPEL en utilisant XPath. L'ASW est placé du côté serveur au niveau du moteur BPEL.

- **Évaluation**

La particularité du travail de [Hmida et al., 2005] [Hmida et al., 2007] tient au fait qu'il intègre dans le processus d'adaptation, à base d'aspects, l'interfaçage correct avec le client afin de correspondre à l'adaptation réalisée. Cependant, l'une des critiques qui peut être adressée à ce travail consiste au choix de présenter les *join points* dans le WSDL des services

Web. Cette approche peut affecter la flexibilité et surtout la disponibilité des services Web à adapter. C'est-à-dire à chaque ajout de nouveaux *join points*, le fournisseur des services est obligé d'arrêter le service, de modifier le fichier de description WSDL et de le redéployer.

- **Synthèse et discussion**

L'objectif de cette section est d'effectuer une évaluation des propositions existantes pour l'adaptation des services au contexte. Afin d'analyser ces travaux de manière pertinente, nous avons identifié des critères d'évaluation propres au domaine étudié :

– **Catégories de contexte prises en compte :** comme il a été présenté précédemment (cf. section 1.3.1), la notion de contexte est un vaste domaine qui fait intervenir plusieurs dimensions (contexte unifié, contexte généralisé, *etc*.). Les travaux existants adressent diverses catégories d'information de contexte. Notamment, certaines contributions se sont spécialisées et offrent des outils pour une prise en charge exhaustive des aspects liés à la mobilité du client tandis que d'autres, plus généralistes mais moins avancées en termes de réalisation, adressent un spectre plus large de préoccupations.

  Ainsi, ce critère permet d'analyser la couverture et l'étendu des catégories de contexte qu'adressent les différentes contributions.

– **Expressivité du modèle de contexte :** les approches étudiées permettent de spécifier les informations de contexte ainsi que les traitements qu'ils proposent pour mettre en œuvre les adaptations. Pour cela, elles présentent des modèles plus ou moins riches permettant de formaliser les informations de contexte. Un modèle trop pauvre (par exemple sous la forme de simples fichiers de configuration) peut-être restrictif car l'adaptation peut ne pas répondre à l'ensemble des exigences du client, tandis qu'un modèle trop riche peut nécessiter une expertise trop complexe à maîtriser.

  Les approches paires/triplets sont caractérisées par une pauvreté d'expressivité et la simplicité des données qu'elles représentent. Les approches orientées modèle sont des approches prometteuses car elles utilisent non seulement un modèle formel pour décrire le contexte, mais offrent aussi un méta-modèle de description qui peut être réutilisé par plusieurs applications. Les approches orientées ontologie sont des approches formelles qui tirent parti des caractéristiques des ontologies pour modéliser le contexte. En effet, les caractéristiques de partage et de distribution des données ont été exploitées afin de définir des méta-modèles de description du contexte. De plus, les moteurs d'inférence fournis par les ontologies ont été utilisés pour déduire des contextes de haut niveau à partir des données collectées.

  Il nous semble donc important d'évaluer quel est le degré d'expressivité offert par les modèles de contexte présentés par les différentes approches.

– **Représentation enrichie grâce aux informations de contexte :** ce critère permet de définir si les informations de contexte enrichissent la requête du client, la représentation du service ou les deux au même temps. L'enrichissement de la requête, comme son nom l'indique d'ailleurs, permet d'intégrer le contexte dans la requête et offre la possibilité de connaître le contexte client qui mérite une adaptation. Quant à l'enrichissement du service, il est généralement utilisé pour informer le client du contexte pour lequel le

service est adapté. Enfin, la représentation enrichie de la requête et du service permet de réaliser une mise en correspondance entre la requête et les services disponibles.

Nous pensons qu'il est intéressant d'examiner la représentation des informations de contexte afin d'évaluer les processus d'adaptation proposés.

– **Technique d'adaptation** : ce critère permet de renseigner sur la technique utilisée pour assurer l'adaptation au contexte. Comme il a été déjà souligné, trois techniques peuvent intéresser la communauté de recherche dans le domaine des services Web. Ces techniques sont : la sélection de service, la sélection de la composition des services et le tissage d'aspects (et notamment l'ajout de nouvelles préoccupations).

– **Capacité d'adaptation :** nous adaptons la catégorisation des capacités d'adaptation proposée par [Villanova-Oliver, 2002] pour étudier les approches présentées précédemment :

  - L'adaptation minimale : cette capacité définit si les systèmes permettent de choisir un service Web parmi un ensemble de services Web défini au préalable.
  - L'adaptabilité : cette capacité étudie l'aptitude d'un service à réagir à des demandes implicites d'adaptation de la part de l'utilisateur ou de son environnement.
  - L'adaptativité : cette capacité définit l'aptitude du service à observer le comportement de l'utilisateur, et à procéder en conséquence à des adaptations.

Approche	**Adaptation de la demande de services**		**Adaptation de la composition des services**		
Critère	**[Suraci et al., 2007]**	**[Keidl and Kemper, 2004]**	**[Qiu et al., 2007]**	**[Charfi et al., 2007a]**	**[Hmida et al., 2007]**
Catégories de contexte prises en compte	Utilisateur, Environnement	Utilisateur, Environnement	Utilisateur, Environnement	--	--
Expressivité du modèle de contexte	Ontologies spécifiques	Présentation XML	Extension d'OWL-S	--	--
Représentation enrichie grâce aux informations de contexte	Requête Service	Requête	Requête Service	--	--
Technique d'adaptation	Sélection de service, Sélection de composition de services	Sélection de service	Sélection de service, Sélection de composition de services	Ajout de nouvelles préoccupations	Ajout de nouvelles préoccupations
Capacité d'adaptation	Adaptation minimale	Adaptation minimale	Adaptabilité	Adaptabilité	Adaptabilité

Tableau 1.8. Comparaison des travaux d'adaptation des services Web

D'après l'étude du tableau comparatif, nous mettons en évidence les points suivants :

– Les catégories de contexte que proposent la majorité des approches déjà décrites, illustrent parfaitement le domaine d'application de ces dernières à savoir : l'informatique pervasive. En effet, une des problématiques dans ce domaine est l'utilisation des services dans un contexte de mobilité de l'utilisateur et des ressources. À l'exception des travaux

de [Charfi and Mezini, 2007] et [Hmida et al., 2007], qui s'intéressent à l'adaptation technique (de bas niveau), comme l'ajout des paramètres de sécurité ou d'authentification. Les travaux comme [Suraci et al., 2007], [Keidl and Kemper, 2004] et [Qiu et al., 2007] intègrent les informations relatives à l'utilisateur, à sa mobilité et à son environnement.

Aucun des travaux présentés ne s'intéresse à un contexte orienté métier du service ou du client, comme par exemple les règles métier sensibles au contexte.

- Concernant la modélisation du contexte, elle repose, soit sur les langages semi structurés, tels que XML [Keidl and Kemper, 2004], soit sur des spécifications orientées ontologies. Certains approches ont étendu des ontologies existantes [Qiu et al., 2007], d'autres ont choisi de développer des ontologies spécifiques [Suraci et al., 2007].
- Concernant l'enrichissement des représentations, celles-ci portent soit uniquement sur la requête client [Keidl and Kemper, 2004] soit sur à la fois la requête client et la description des services [Suraci et al., 2007], [Qiu et al., 2007].
- Les travaux de [Suraci et al., 2007] et [Keidl and Kemper, 2004] traitent de l'adaptation en utilisant les techniques de sélection de service ou de composition des services qui répondent au mieux au contexte du client et présentent par la suite une adaptation minimale. Quant aux travaux de [Hmida et al., 2007] et [Charfi and Mezini, 2007], ils se focalisent essentiellement sur l'ajout de nouvelles préoccupations afin d'adapter des compositions existantes et présentent, par conséquent, une adaptation de type adaptabilité. Le travail [Charfi and Mezini, 2007] se préoccupe essentiellement de l'ajout des informations de sécurité et d'authentification dans les processus BPEL existants tandis que le travail de [Hmida et al., 2007] se focalise sur la génération automatique du client qui prend en compte les changements dynamiques du service. Plus récemment, Anis Charfi et Mira Mezini ont travaillé sur la possibilité de rajouter des appels vers un service qui s'occupe de gérer certaines règles métier et d'autres paramètres de qualité de service [Charfi et al., 2007b].

En considérant le cadre de référence déjà présenté (voir Figure 1.25), nous pouvons souligner que nous nous intéressons aux différentes étapes du cycle de vie du service (la publication, la découverte, et l'exécution des services). À la différence des travaux déjà présentés qui portent sur une étape particulière du cycle de vie du service, nous souhaiterons prendre en compte dans la même démarche, l'ensemble des étapes dans lesquelles l'adaptation peut s'insérer. Cette transversalité permet ainsi d'offrir une solution d'adaptation homogène qui couvre la totalité du cycle de vie d'un service dédié à la coopération.

La transversalité de l'adaptation des services s'exprime ainsi dans les différents chapitres de notre contribution (voir Figure 1.31).

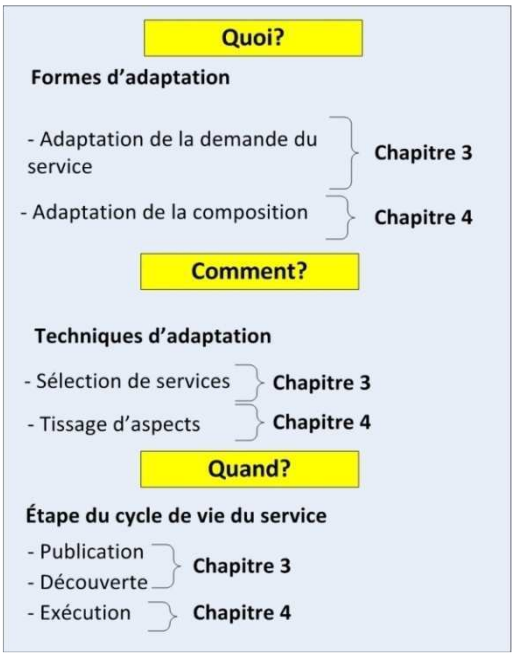


Figure 1.31. Contribution par rapport au cadre de référence

1.3.6 Conclusion

Cette partie de l'état de l'art nous a permis de faire un tour d'horizon sur le contexte (définition, système sensible au contexte, éléments fonctionnels d'un système sensible au contexte) ainsi que l'adaptation au contexte (les finalités et les mécanismes supports).

Notre étude, concernant le contexte et les finalités d'adaptation au contexte, a révélé qu'il existe plusieurs manières de caractériser le contexte (contexte généralisé, géolocalisé, environnemental et unifié). Chaque travail s'intéressant à l'adaptation, débute en définissant le contexte pour lequel il apporte une adaptation. Il présente ainsi sa propre définition et formalisation du contexte.

Dans le domaine d'adaptation des services Web au contexte, notre exploration du domaine a donné naissance à un cadre de référence qui met en évidence : les formes d'adaptation (adaptation de la demande du service et adaptation de la composition), les techniques d'adaptation (sélection de services ou encore sélection de composition des services et la technique de tissage d'aspects), le moment d'adaptation (étapes de cycle de vie du service).

L'examen des travaux d'adaptation au contexte a montré la diversité des travaux et des techniques utilisés. En outre, il a détecté certaines limites : d'une part, aucun des travaux étudiés ne s'intéresse à la transversalité de l'adaptation partant de la description, la publication jusqu'à l'exécution du service et d'autre part, à notre connaissance, aucun travail n'a étudié l'adaptation de la logique métier du service en fonction des changements de contexte. Nous considérons que ce dernier point est prépondérant dans le cadre des coopérations interentreprises.

1.4 Conclusion

La mise en place de la coopération interentreprises à la demande constitue un enjeu pour les futures pratiques en matière de gestion d'entreprise. La revue de la littérature a montré clairement que dans la plupart des cas, une dissociation entre les aspects métiers et les aspects techniques laisse certaines failles au niveau de la modélisation de la coopération. De ce fait, une approche combinant à la fois ces éléments pourrait améliorer clairement le processus de coopération.

Une vue détaillée des solutions, formalismes et approches qui facilitent la mise en place de la coopération à la demande a souligné l'importance de l'approche service. L'approche service propose de contourner les faiblesses des systèmes d'information à travers une réponse dynamique et non planifiée, ce qui rend la coopération plus facile à gérer à travers une composition dynamique des services des partenaires. Cependant, l'approche service nécessite une mise en place convenable au sein du système d'information. Cette mise en place passe par une ingénierie du système d'information qui assure d'une part sa flexibilité et d'autre part, favorise sa participation à des scénarios de coopérations à la demande.

L'ensemble des études menées dans ce chapitre de l'état de l'art a révélé également que le concept de service en particulier à travers sa description, présente une certaine stabilité dans le temps. Ainsi, la mesure de la réactivité du processus de coopération, construit par composition des services, reste faible. Une amélioration significative de cette réactivité passe par la prise en compte de la notion d'adaptation au contexte tout au long du cycle de vie de la coopération. Donc, une approche de coopération interentreprises basée sur l'adaptation de services pourrait améliorer clairement la réactivité de ce processus.

CHAPITRE 2

Construction de l'architecture de services au sein de l'entreprise

Table des matières

L'introduction générale de cette thèse a permis d'étudier les trois sous problématiques de base de notre travail de recherche. La première question posée consistait à savoir comment migrer vers une architecture de services, c'est-à-dire, comment modéliser et mettre en place une architecture orientée services appropriée au cadre d'une coopération interentreprises. Comme nous l'avons souligné, les architectures orientées services souffrent d'un certain nombre de limites et de lacunes. Il s'agit fondamentalement du manque de méthodologies permettant de les construire de façon efficace au sein de l'entreprise. Certes, quelques approches de modélisation et d'identification de services ont vu le jour, mais la majorité de ces approches ne sont pas flexibles du fait qu'elles sont fortement liées aux technologies et aux outils utilisés. De plus, les démarches de développement d'applications orientées services ne proposent que des lignes directrices afin de guider les architectes vers un modèle orienté services, mais elles ne délivrent pas de méthodologie complète et précise pour atteindre leur objectif de façon efficiente. Ainsi, une démarche méthodologique adaptée aux entreprises et à la coopération qui peut naître entre ces entreprises s'avère d'une importance capitale.

C’est dans le but de répondre à une telle question que nous proposons dans ce chapitre de présenter notre démarche méthodologique dénommée CSOMA (**C**ontextual **S**ervice **O**riented **M**odelling and **A**nalysis). Cette démarche de construction de l’architecture de services au sein de l’entreprise permet de répondre à la première sous problématique de notre travail de recherche (cf. introduction générale, Figure 1).

Dans ce qui suit, nous allons exposer dans la section 2.1 une présentation générale de la démarche CSOMA. À l’issue de cette présentation, nous allons dans la section 2.2 proposer un méta-modèle support à la démarche, puis dans la section 2.3 présenter une typologie des services ainsi qu’un ensemble de méta-modèles qui les décrivent. La section 2.4 décrira en détail les différentes phases de la démarche CSOMA. Enfin, la démarche CSOMA a été appliquée à une étude de cas industriel dont certains aspects sont synthétisés dans la section 2.5.

2.1 Présentation générale de CSOMA

La construction d’une architecture orientée services au sein de l’entreprise est un processus qui pose un certain nombre de défis et de risques. Si les principes d’une SOA sont connus, il n’y a pas encore d’approche méthodologique unifiée permettant de la construire d’une façon efficace. Déployer une architecture orientée services demande plus qu’un simple regroupement des entités logicielles existant dans l’entreprise. En effet, un tel déploiement nécessite la mise en œuvre d’une méthode efficace et robuste afin d’assurer l’identification, la modélisation et la spécification des services en vue de permettre, par la suite, leur utilisation que ce soit à l’interne ou à l’externe de l’entreprise.

Autour de la problématique de migration vers une architecture de services s’articulent plusieurs approches et théories, chacune sous tendant des positions souvent trop dogmatiques. Parmi ces approches on trouve l’approche descendante (*Top-Down*) et l’approche ascendante (*Bottom-Up*). CSOMA (Contextual Service Oriented Modelling and Analysis) que nous proposons, est une approche hybride dans le sens où elle est à la fois ascendante et descendante. La Figure 2.1 illustre un aperçu général de la démarche méthodologique CSOMA qui intègre plusieurs dimensions.

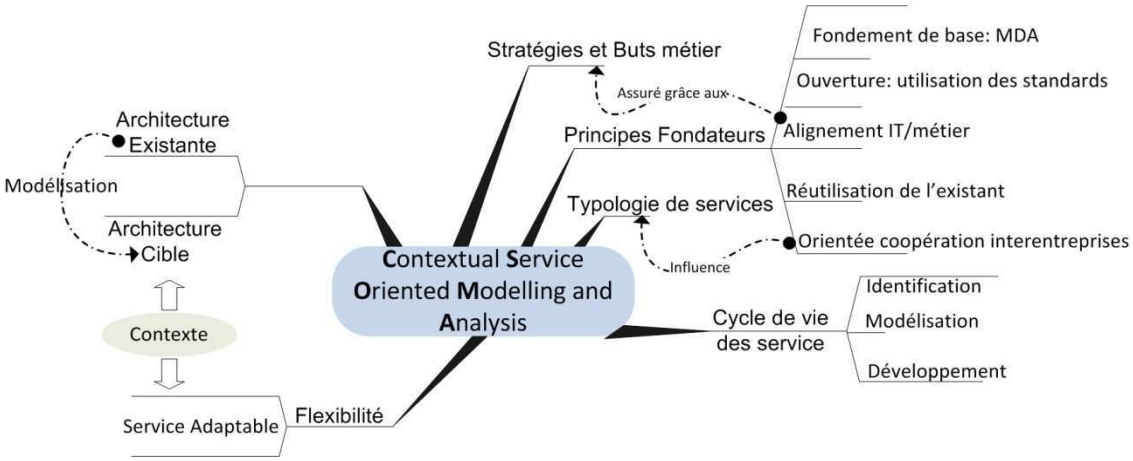


Figure 2.1. Aperçu général de la démarche CSOMA

Comme illustré dans la Figure 2.1, CSOMA est une démarche de construction de l'architecture SOA au sein de l'entreprise qui couvre les trois phases de cycle de vie des services : phase d'identification, de modélisation et de développement des services. CSOMA part d'une architecture d'entreprise existante « *As-Is* » pour retrouver l'architecture cible « *To-Be* » à travers la réalisation d'un ensemble de cartographies (métier et applicatives). Ces dernières proposent une vision globale de l'entreprise et de son système d'information dans son intégralité.

Une caractéristique principale de CSOMA est l'intégration de la flexibilité. Cette dernière est une caractéristique importante et largement recherchée par les entreprises. D'une façon générale, la flexibilité peut être définie comme l'aptitude d'un système à changer facilement pour pouvoir s'adapter aux circonstances. La notion de flexibilité est assurée à travers l'identification des services adaptables. Un service adaptable est un service qui évolue pour prendre en compte les changements intervenus dans son environnement. Il s'agit essentiellement des services sensibles au contexte de leur utilisation, c'est-à-dire qui se présentent et changent selon l'opportunité de la coopération. La prise en compte des variantes d'usage des services dès la phase de modélisation, apporte la flexibilité nécessaire à la démarche CSOMA.

En outre, CSOMA repose sur un ensemble de principes fondateurs :

- **Principe de compatibilité avec l'approche d'ingénierie dirigée par les modèles MDA (*Model Driven Architecture*) :** à l'encontre des méthodes existantes qui sont fortement reliées à la technologie service Web, CSOMA est une démarche indépendante des technologies de réalisation. En effet, la démarche méthodologique CSOMA s'inscrit dans un cadre d'architecture dirigée par les modèles (MDA). Comme nous l'avons déjà mentionné (cf. chapitre 1, section 1.2.4.1.2), MDA est une approche d'ingénierie mettant le modèle au cœur du problème [OMG, 2003a]. Tout d'abord, les fonctionnalités et le comportement d'un système sont modélisés à partir d'un ensemble de modèles métier appelés CIM (*Computation Independent Model*) et ceci sans tenir compte des caractéristiques technologiques. Les modèles obtenus sont les PIM (*Platform Independent Model*). 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. Ces derniers sont les modèles de niveau PSM (*Platform Specific Model*). En se basant sur les bénéfices attendus de cette approche d'ingénierie (cf. chapitre 1, section 1.2.4.1.2), nous considérons que son utilisation dans notre travail de recherche est justifiée. La possibilité de disposer d'un modèle de services d'entreprise décrit indépendamment des choix technologiques prend alors toute son importance.

  La démarche CSOMA proposée dans ce chapitre est compatible avec l'approche MDA (Figure 2.2).

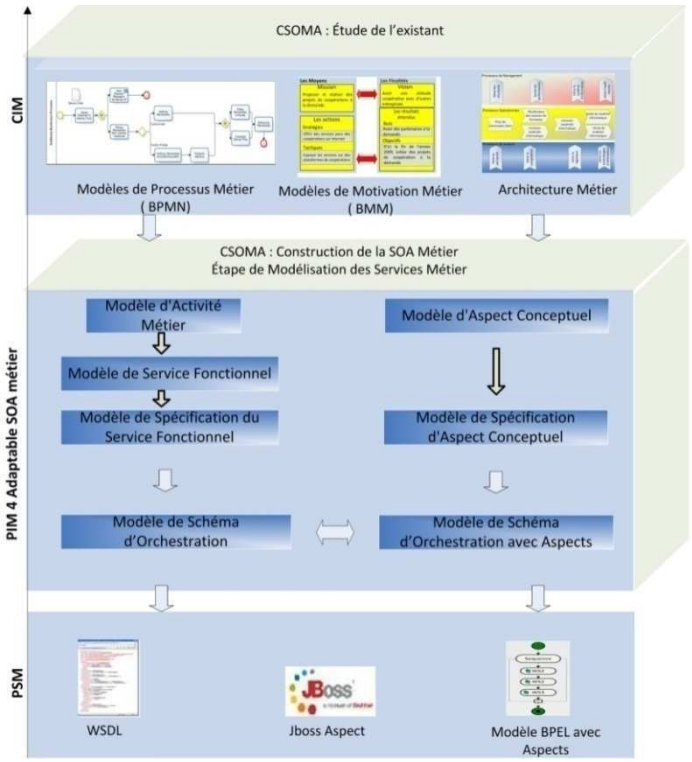


Figure 2.2. Démarche CSOMA par rapport au cadre MDA

CSOMA permet de partir d'une spécification métier (ensemble de modèles métier) pour retrouver par la suite le modèle de système d'information orienté services. Il est important de signaler que notre démarche méthodologique ne traite pas de la transformation du CIM (*Computation Independent Model*) vers le PIM (*Platform Independent Model*). Cette dernière relève plutôt de la compétence des analystes métier dans l'interprétation et dans l'analyse des modèles de processus pour identifier par la suite l'ensemble des services métier. Néanmoins, CSOMA propose un ensemble de lignes directrices qui guident les architectes vers un modèle orienté services. Cet ensemble de lignes directrices est décrit en détail dans la suite de ce chapitre.

Le niveau PIM de la démarche CSOMA met en évidence la construction de la SOA métier adaptable, essentielle vis-à-vis de notre problématique de coopération interentreprises basée sur les services. Quant au niveau PSM (c'est-à-dire le développement des services adaptables), il a été intégralement réalisé dans nos travaux de recherche. Le chapitre 4 de cette thèse exposera en détail la mise en oeuvre des services adaptables et ceci en utilisant la programmation orientée aspect.

En revanche, nous ne nous intéressons pas à la transformation du niveau PIM vers le niveau PSM (*Platform Specific Model*). Plus précisément, nous n'allons pas nous focaliser sur la proposition des mécanismes de transformation de modèles qui réalisent la transition entre le niveau logique et technique et qui relève plutôt de la discipline du génie logiciel. De plus, dans la littérature plusieurs travaux ont été déjà proposés pour essayer de résoudre cette problématique de transformation [Zhang, 2008], [Rahmani et al., 2006], [Brown et al., 2005].

- **Principe d'ouverture :** il impose de s'inscrire dans le cadre d'utilisation des standards industriels. En effet, CSOMA s'adresse aux petites et moyennes entreprises pour lesquelles, il est indispensable de proposer une solution basée sur des standards. L'utilisation des standards industriels constitue une boîte à outils à laquelle s'ajoute une démarche guidant la construction de l'architecture de services à chaque étape.
- **Principe d'alignement IT/ métier :** il consiste à mettre en cohérence la stratégie du système d'information avec la stratégie métier de l'entreprise [Phelizon and Rouhier, 2002]. Afin d'assurer ce principe, CSOMA se base sur les stratégies et les buts métier à atteindre pour identifier les services d'entreprise. Ainsi, les services identifiés seront en phase avec le monde métier, c'est-à-dire reliés à des buts et des stratégies définis par l'entreprise. Une telle identification de services permet de réduire l'écart entre les stratégies métier et la définition des services.

  Dans la même optique, CSOMA propose une double démarche de construction de l'architecture de services : construction de la SOA métier et construction de la SOA IT. Ces deux niveaux sont fondamentaux pour pouvoir distinguer entre les services plus abstraits (métiers) et les services plus techniques (informatiques), séparant ainsi les préoccupations d'ordre métier des préoccupations d'ordre informatique. Cette double préoccupation permet d'assurer en retour un meilleur alignement IT/métier.
- **Réutilisation de l'existant :** le patrimoine applicatif de l'entreprise, résulte de l'empilement de générations successives d'applications, de la stratification des logiciels, comportant des redondances et manquant de cohérence. La complexité croissante de ce patrimoine existant génère des difficultés de plus en plus grandes pour faire évoluer le système d'information en adéquation avec les attentes des différents métiers de l'entreprise. Néanmoins, il n'est pas économiquement envisageable de le refondre complètement. Par conséquent, CSOMA supporte la vision de la réutilisation ; il faut au maximum ré-exploiter l'existant quand c'est possible.
- **Principe de prise en compte de la coopération interentreprises :** CSOMA est une démarche adaptée à la coopération interentreprises. Pour ce faire, nous proposons une typologie de service qui prend en compte cette spécificité. Au cœur de cette typologie se trouve le service domaine, qui est un service dédié à la coopération. Dans cette optique, le processus coopératif se construit par composition de plusieurs services domaines. En effet, un service domaine fourni par une entreprise représente un processus ou un sous-processus métier englobant les compétences et le savoir-faire de cette entreprise.

2.2 Méta-modèle de CSOMA

Avant d'aborder l'approche mise en avant par CSOMA, il nous paraît tout d'abord nécessaire de présenter le méta-modèle qui résume les concepts de base et qui facilite en retour la compréhension de la démarche de conception proposée par CSOMA.

CSOMA considère les services comme un concept central résultant d'une double démarche de construction de l'architecture de services : construction de la SOA métier et construction de la SOA IT. En effet, CSOMA permet de conjuguer la réutilisation du patrimoine applicatif

existant (SOA IT) avec l'émergence d'une SOA naturellement orientée vers le métier et les processus de l'entreprise (SOA métier). Pour identifier les services métier, il est fondamental de « penser service » dès le premier niveau de décomposition du système d'information d'une entreprise. En effet, en quête d'agilité et de flexibilité, il n'est plus concevable de reléguer cette identification à un problème purement applicatif. En outre, il n'est pas économiquement envisageable de refondre complètement le patrimoine applicatif. L'enjeu consiste donc à rendre le système d'information le plus réactif possible tout en préservant le patrimoine informationnel de l'entreprise. Ces différentes caractéristiques sont bel et bien respectées au niveau du méta-modèle de CSOMA présenté en Figure 2.3 .

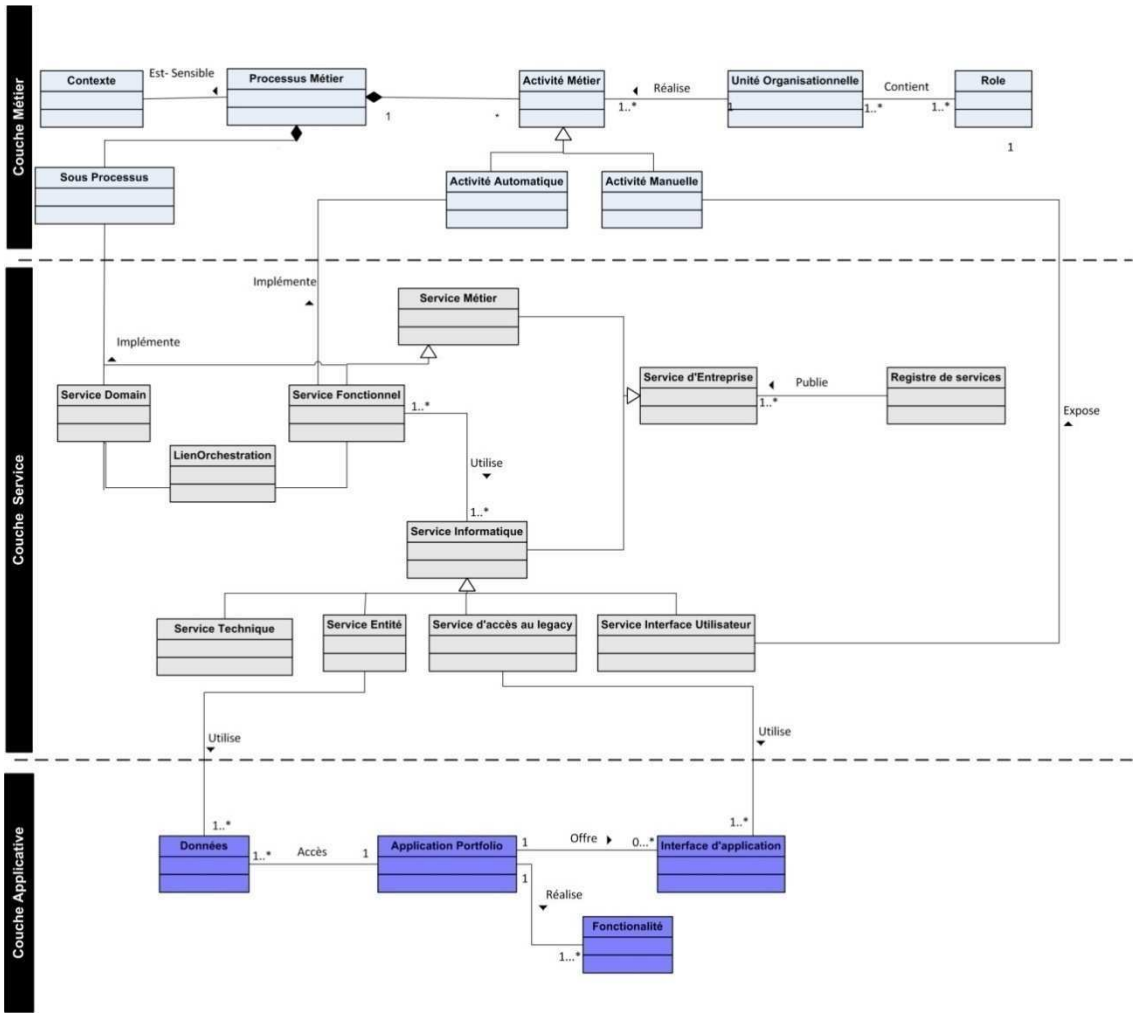


Figure 2.3. Méta-modèle de CSOMA

Ce méta-modèle met en exergue les différents concepts de CSOMA tout en tenant compte de la double préoccupation IT /métier. Nous allons détailler ces différents concepts en mettant en évidence les relations qui existent entre eux.

- **La couche métier :** concerne les processus métiers et les flux d'information entre eux. Elle permet également de recenser les orientations stratégiques de l'entreprise ainsi que les activités qui doivent être supportées par le système d'information. Il s'agit essentiellement des éléments relatifs au métier de l'entreprise. Les processus métiers sont

au cœur de la couche métier. Rappelons que nous définissons un processus métier comme un enchaînement d'activités réalisées par un ensemble d'acteurs de l'entreprise et produisant une valeur ajoutée pour celle-ci (cf. chapitre 1, section1.1.4.1). Les activités métier peuvent être manuelles ou automatiques. Chaque processus est en communication avec d'autres et peut être décomposé en sous-processus. Les processus métiers et précisément les activités métier sont supervisées par des unités organisationnelles de l'entreprise. Ces dernières sont les entités qui organisent les acteurs dans l'organigramme de l'entreprise. Les acteurs et les unités organisationnelles sont liés avec les processus métiers au travers des rôles qu'ils remplissent.

On distingue deux types de processus métier : les processus intra entreprise et les processus interentreprises. Les processus intra entreprise correspondent aux processus privés de l'entreprise qui ne doivent pas être dévoilés aux autres. Quant aux processus interentreprises, ils sont formés par composition des services exposés par les différents partenaires.

En quête de flexibilité, nous considérons qu'il est important dans le cadre de CSOMA de prendre en compte la connaissance relative au contexte lors de l'ingénierie des processus métiers. Rappelons que l'une des principales limites des approches de modélisation des processus est qu'elles présument que les comportements des acteurs et des rôles sont définis a priori. Cependant, les capacités et le comportement de ces derniers peuvent être altérés en fonction du contexte dans lequel l'entreprise et ses acteurs se trouvent. Ainsi, nous considérons qu'une modélisation orientée contexte des processus métiers constitue une base importante pour l'identification des services métier adaptables. Cette spécificité sera décrite en détail dans la suite de ce chapitre.

- **La couche service :** constitue la couche pivot du méta-modèle de CSOMA. Un service d'entreprise a pour objectif de répondre à des préoccupations en réalisant certaines actions [Izza, 2006]. Il est clairement identifiable, expose un contrat d'utilisation, documenté, autonome, localisable et activable à distance. Un service d'entreprise est raffiné en deux concepts fils qui sont les services métier et les services informatiques. Les services métier sont des services qui font sens pour les analystes métier de l'entreprise et qui sont issus de l'analyse et la modélisation des processus métiers. Une particularité des services métier de CSOMA est qu'ils sont des services adaptables et présentent des comportements différents en fonction du contexte de leur utilisation. Nous distinguons deux types de services métier : les services fonctionnels et les services domaines. Quant aux services informatiques, ils regroupent des fonctionnalités offertes par la partie informatisée du système d'information et ils sont utilisés par les services métier décrits précédemment. On distingue quatre types de services informatiques qui sont les services

d'accès au système legacy[11], les services d'interface utilisateur, les services entités et les services techniques.

Une description plus détaillée des services d'entreprise est proposée dans la section suivante de ce chapitre.

- **La couche applicative :** décrit l'ensemble des applications mises en œuvre pour implémenter les fonctions de l'entreprise. Ces applications font partie du patrimoine applicatif de l'entreprise et sont développées par différents intervenants au cours de nombreuses années. Ceci inclut les progiciels de l'entreprise, les bases de données et les applications propriétaires existantes. En passant en mode services, le patrimoine applicatif de l'entreprise sera encapsulé par un ensemble de services informatiques et plus précisément par les services d'accès au système *legacy* ainsi que les services entités (les services permettant l'accès aux données). Outre le patrimoine applicatif, le niveau applicatif englobe les technologies utilisées et l'infrastructure technique sur laquelle reposent les applications de l'entreprise. Ces informations sont fondamentales pour pouvoir identifier les services techniques qui permettront de gérer l'infrastructure du système d'information comme par exemple les services liés à la gestion du réseau et de la sécurité.

2.3 Typologie de services de CSOMA

Une typologie de services permet de structurer la démarche de conception et de réalisation d'une architecture de services au sein de l'entreprise. Une approche rigoureuse passe par la définition d'une catégorisation des services : une nomenclature des services et des liens qui peuvent exister entre eux. Chaque catégorie de service se distingue par sa nature (degré d'abstraction) ainsi que sa granularité (degré de composition). Sans cela, le risque est important de conduire à un système où des services de grain différent se recouvrent, créant une prolifération de services pour laquelle aucune possibilité de réutilisation et de rationalisation ne peut prévaloir.

2.3.1 Services métier

Les services métier correspondent à des fonctionnalités métier. On peut distinguer les services fonctionnels et les services domaines. Les services fonctionnels sont des services de moyenne granularité comme par exemple des services offrant des fonctionnalités liées à la gestion d'une commande client, ou encore au calcul du montant d'une commande. Ce sont des

[11] Nous empruntons l'anglicisme « *legacy* » pour désigner le patrimoine applicatif d'une entreprise.

services qui exposent la notion de fonction du système d'information, et qui permettent d'implémenter la notion d'activité métier.

Les services domaines sont des services composites qui encapsulent des processus ou des sous-processus métier d'un domaine comme par exemple le processus de planification d'une tâche de formation de personnel d'une entreprise. Techniquement parlant, ces services encapsulent la logique du Workflow du processus ou du sous-processus métier qu'ils exposent. Typiquement, les services domaines orchestrent des services fonctionnels qui utilisent à leur tour des services de plus bas niveau, les services informatiques. Les services domaines peuvent participer à plusieurs processus à l'interne d'une entreprise. Une fois que la maîtrise d'ouvrage décide de les publier, ces derniers pourront être invoqués par les partenaires de l'entreprise et participer par la suite à des processus coopératifs.

Quelque soit sa catégorie (fonctionnel ou domaine), les services métier sont identifiés et définis suivant une logique métier, indépendamment de la technologie.

Le service métier possède l'avantage d'être adaptable en fonction de son contexte d'utilisation. En effet, il propose un comportement qui suit la situation dans laquelle il évolue. Cette capacité introduit la flexibilité dans la manière de satisfaire l'objectif métier qu'il poursuit. Elle permet également de configurer le service en fonction du contexte du consommateur, du service lui-même et aussi de l'opportunité de coopération dans le cas où le service participerait à un processus coopératif.

La Figure 2.4 met en évidence la description du service métier et ceci en trois parties correspondant à son interface, son contrat et à l'expression de son comportement adaptable.

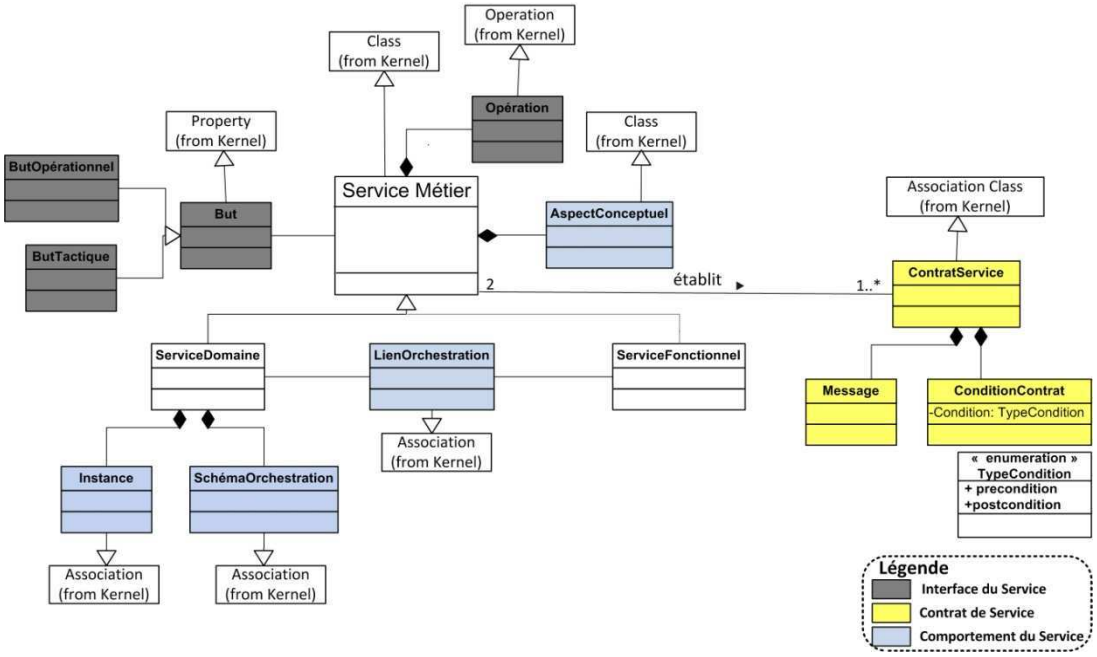


Figure 2.4. Méta-Modèle du service métier

La Figure 2.5 montre les différents stéréotypes proposés pour représenter les concepts introduits dans le méta-modèle du service métier. L'ensemble de ces stéréotypes représente le profil UML proposé pour le concept de service métier.

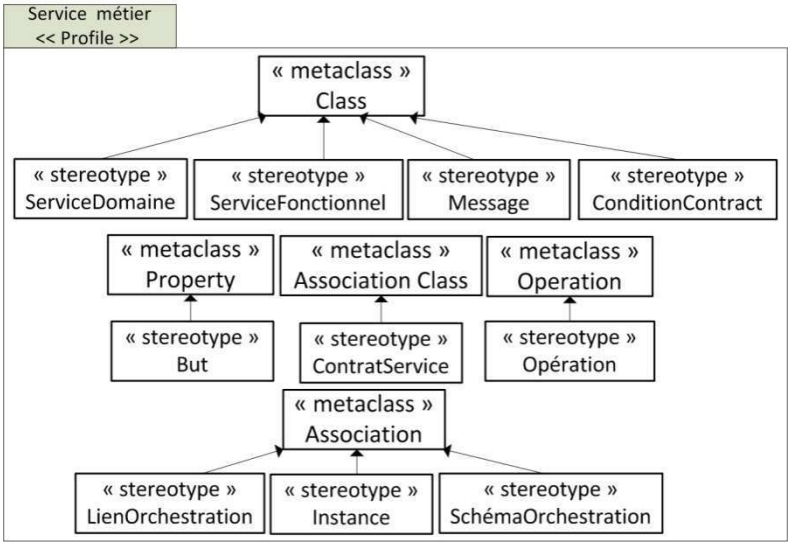


Figure 2.5. Profile UML du service métier

Dans ce qui suit, nous allons détailler les différents concepts appartenant au méta-modèle de la Figure 2.4 et ceci en trois parties : l'interface, le contrat et le comportement du service.

2.3.1.1 Interface du service métier

Tout comme l'interface d'un service logiciel classique, l'interface d'un service métier permet de caractériser le service d'un point de vue externe, celui de consommateur de service. L'interface d'un service métier a pour objectif de permettre au service d'être visible au client. Dans le cadre de notre travail, l'interface du service métier a la particularité d'attribuer à chaque service métier un but qui lui est propre et qu'il cherche à atteindre. L'interface décrit le but et l'ensemble des traitements appelés opérations. Ces dernières constituent la manière avec laquelle le service réalise son but. On peut définir une ou plusieurs opérations pour un service métier. Cette approche n'impose pas le choix d'une implémentation technique.

2.3.1.2 Contrat du service métier

Dans une approche SOA, les services se communiquent et interagissent entre eux par le biais de contrats. D'une manière similaire dans une SOA métier, la communication entre deux services métier est décrite grâce au contrat de service. Les contrats établis entre les services reflètent les services qui sont impliqués dans le contrat, les rôles joués, ainsi que d'autres propriétés non fonctionnelles telles que la qualité du service, le type de communication (synchrone ou asynchrone), *etc.*

Le contrat de service s'exprime sous forme de pré et de post conditions que les services impliqués doivent remplir afin d'interagir entre eux. Les pré et les post conditions correspondent respectivement à des conditions sur les états des objets manipulés par les paramètres d'entrée et les paramètres de sortie d'un service.

2.3.1.3 Comportement du service métier

Le comportement du service métier est décrit par deux caractéristiques majeures. La première est relative à l'orchestration qu'il peut réaliser. En effet, par définition un service domaine est un service composite qui orchestre un ensemble de services fonctionnels. Ceci est schématisé dans le méta-modèle grâce à l'association « *LienOrchestration* ». La deuxième caractéristique du service métier tient au fait qu'il est un service sensible au contexte d'utilisation. En effet, il adapte son comportement en fonction du contexte dans lequel il évolue. Ces deux caractéristiques sont décrites en détail dans les sections suivantes.

- **Orchestration du service métier**

L'orchestration des services fonctionnels au sein du service domaine donne naissance à une séquence d'orchestration appelée schéma d'orchestration. La Figure suivante complète le méta-modèle proposé dans la Figure 2.4 et ceci en distinguant les trois types d'orchestration qui relient le service domaine avec les différents services fonctionnels : l'orchestration séquentielle, l'orchestration parallèle et l'orchestration itérative.

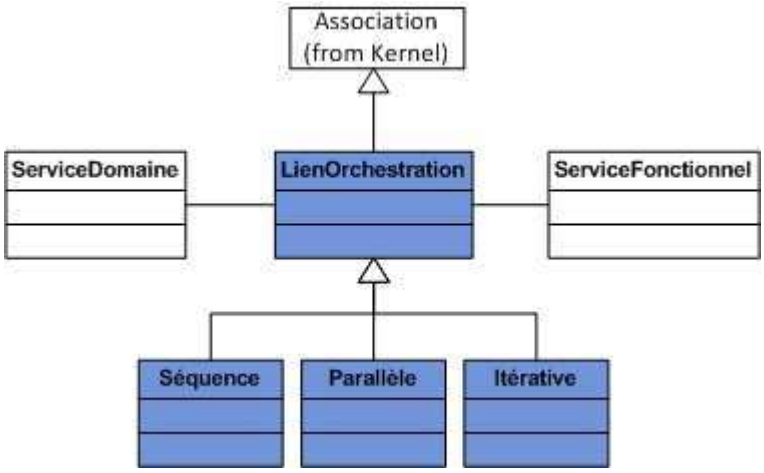


Figure 2.6. Méta-Modèle d'Orchestration

Nous détaillons dans ce qui suit les types d'orchestration des services fonctionnels.

– Orchestration séquentielle

C'est le cas où le service domaine requiert la livraison séquentielle des services fonctionnels. Conformément au profile UML présenté en Figure 2.6, nous associons au lien d'orchestration séquentielle l'association <<Séquence>>.

o Notation graphique

L'association <<Séquence>> relie les différents services fonctionnels au sein du service domaine. La Figure 2.7 illustre un exemple d'un service domaine appelé « Service Livraison » qui orchestre séquentiellement trois services fonctionnels à savoir : le « Service Camion », le « Service Calcul Itinéraire » et le « Service Calcul Coût ».

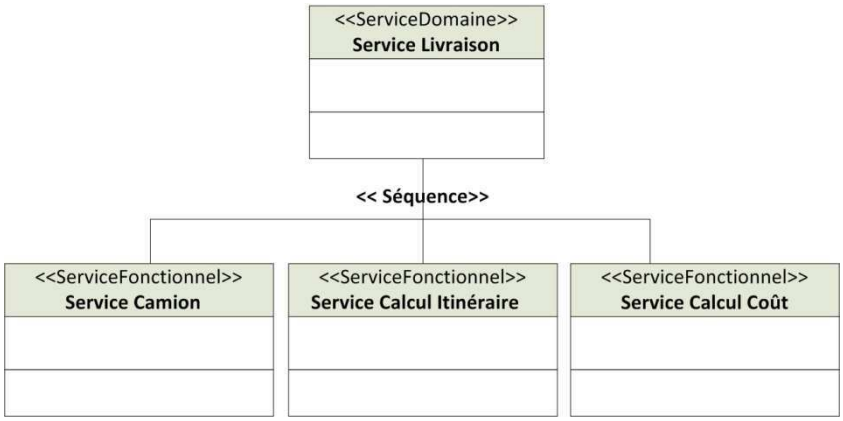


Figure 2.7. Représentation graphique d'une orchestration séquentielle

– Orchestration parallèle

À l'encontre de l'orchestration séquentielle, l'orchestration parallèle n'exige pas un ordre de séquence des différents services fonctionnels au sein d'un service domaine. La Figure 2.8 illustre l'association << Parallèle>> que nous avons retenue pour l'orchestration parallèle. Il s'agit d'une association (au sens UML) reliant les services fonctionnels qui peuvent être exécutés dans n'importe quel ordre.

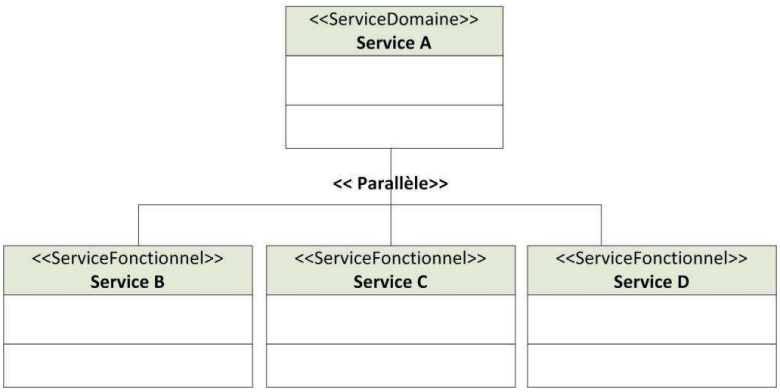


Figure 2.8. Représentation graphique d'une orchestration parallèle

– Orchestration itérative

L'orchestration itérative se manifeste par le fait qu'un service domaine peut nécessiter l'exécution itérative d'un ou de plusieurs service(s) fonctionnel(s). La Figure 2.9 illustre l'association <<Itérative>> appliquée au service fonctionnel qui s'exécutera en mode itéré.

Il est possible d'ajouter une contrainte en utilisant le langage OCL [OMG, 2006b] pour exprimer le nombre d'itérations possibles.

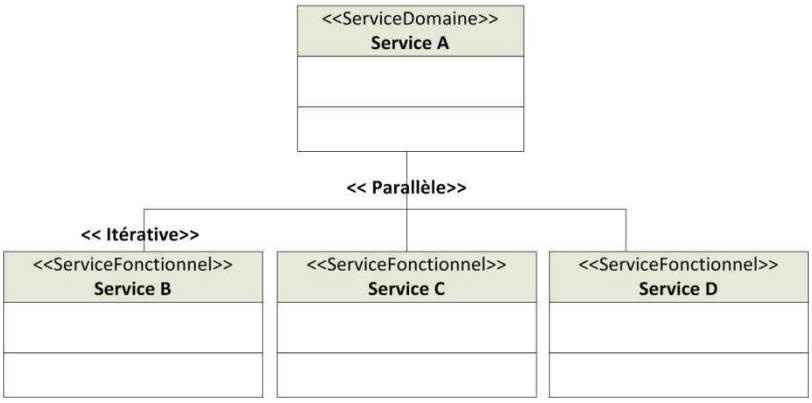


Figure 2.9. Représentation graphique d'une orchestration itérative

- **Adaptation du service métier**

Comme il a été déjà souligné, un service métier est un service adaptable qui change de comportement en fonction du contexte qui se présente. Cette adaptation dépend de la nature du service métier lui-même. En effet, un service fonctionnel possède des adaptations qui concernent essentiellement sa logique métier alors qu'un service domaine peut avoir des adaptations qui se référant à sa logique métier ainsi qu'à sa logique d'orchestration. D'une façon générale, nous considérons que plus la modélisation du service est paramétrée, plus son adaptation face à de nouveaux contextes d'utilisation est facilitée. Nous reviendrons en détail dans le chapitre 4 de cette thèse sur les mécanismes logiciels proposés pour l'adaptation de la logique métier du service.

Dans ce qui suit, nous allons présenter la capacité du service métier à présenter des adaptations que ce soit dans sa logique métier ou encore dans sa logique d'orchestration. Nous commençons tout d'abord par traiter l'adaptation au niveau de la logique métier.

– Adaptation au niveau de la logique métier

Dans le cadre de la méthodologie CSOMA, on s'inspire du concept de base de la programmation orientée aspect pour modéliser certaines préoccupations du service qui influencent son comportement. Comme nous l'avons déjà souligné (cf. chapitre 1, section 1.3.4.3.3) la programmation orientée aspect est un nouveau paradigme de programmation qui permet de séparer les considérations techniques des descriptions métier en un ensemble de modules appelés aspects [Kiczales et al., 1997]. D'une manière similaire, un service métier possède un ensemble d'aspects qu'on appelle les Aspects Conceptuels.

Un Aspect Conceptuel est défini comme étant une préoccupation particulière pour un service métier qui peut inclure : une exigence non fonctionnelle (telle que la sécurité), une règle métier, ou encore une capacité d'adaptation en réponse à certains changement de contexte.

Compte tenu de cette définition, trois types d'Aspect Conceptuels peuvent être identifiés : l'Aspect Non Fonctionnel, l'Aspect Règle Métier et l'Aspect Prise en Compte du Contexte.

- L'Aspect Non Fonctionnel (*Non Fonctionnel Aspect*) : exprime des préoccupations non fonctionnelles d'un service telles que la sécurité, le coût, la disponibilité. Ces préoccupations affectent le comportement du service. Un exemple d'Aspect Non Fonctionnel peut être un aspect de sécurité qui applique un contrôle d'accès au

service. Ce contrôle se manifeste à travers un mécanisme d'authentification ou d'autorisation qui attribut des droits d'accès au consommateur de service. Ainsi, il permet de définir qui peut utiliser telle ou telle opération d'un service.

- L'Aspect Règle Métier (*Business Rule Aspect*) : définit une déclaration de haut niveau structurée, qui permet de contraindre, contrôler et influencer une logique métier d'un service. Un Aspect Règle Métier peut encapsuler plusieurs type de règles : une règle qui exprime une contrainte, une règle qui effectue une action spécifique ou encore une règle de calcul ou d'inférence [BRG, 2002].
- L'Aspect Prise en Compte du Contexte (*Context Enabler Aspect*) : spécifie une adaptation du service en fonction de certains changements contextuels. Il permet notamment de personnaliser le service en tenant compte des préférences de l'utilisateur, de son contexte ou encore du contexte du service lui-même. Les actions d'adaptation proposées par l'Aspect Prise en Compte du Contexte peuvent modifier une opération d'un service, ajouter une autre opération afin de répondre aux informations de contexte.

Comme illustré par le méta-modèle de la Figure 2.10, un Aspect Conceptuel étend un service métier grâce à l'association « crosscut»[12]. Quelque soit son type, un Aspect Conceptuel comporte un ensemble de fonctionnalités appelées *advices*. Ces *advices* peuvent être exécutées avant, après ou en remplacement des fonctionnalités du service métier en question.

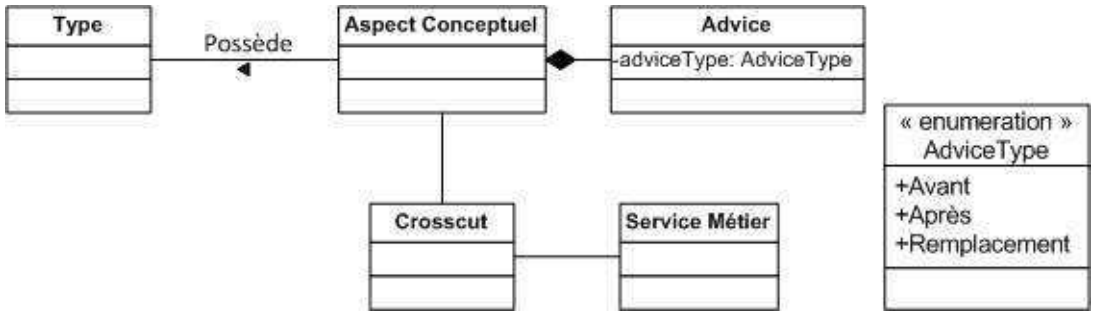


Figure 2.10. Méta modèle d'un Aspect Conceptuel [Boukadi et al., 2009e]

La Figure 2.11 montre les différents stéréotypes proposés pour représenter les concepts introduits dans le méta-modèle de l'Aspect Conceptuel.

[12] Nous empruntons l'anglicisme « crosscut » pour désigner la relation entre un Aspect Conceptuel et un service métier. Cette relation signifie que le service métier est enrichi par de nouvelles fonctionnalités lui permettant de garantir un certain niveau de flexibilité.

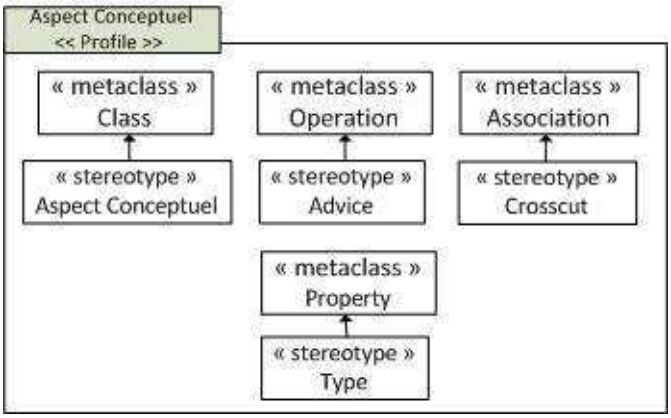


Figure 2.11. Profile UML de l'Aspect Conceptuel

– Adaptation au niveau de la logique d'orchestration

Ce type d'adaptation est consacré au service domaine. En effet, le service domaine propose plusieurs alternatives d'orchestration des services fonctionnels. Comme illustré dans la Figure 2.4, le service domaine propose deux types d'adaptation au niveau de sa logique d'orchestration à savoir : le choix alternatif entre les instances de services fonctionnels et le choix de schéma d'orchestration à appliquer. Le premier type d'adaptation correspond à une adaptation de type « OU exclusif » (XOR) entre services et le deuxième correspond à un choix au niveau du schéma d'orchestration de services. Ces deux types d'adaptation sont détaillés dans ce qui suit.

- Service domaine avec choix d'instance

Cette adaptation correspond au fait que le service domaine peut exprimer un choix alternatif entre les différents services fonctionnels qu'il orchestre. Chaque service fonctionnel propose une manière pour répondre au but métier du service domaine. Ainsi, le service domaine regroupe plusieurs services fonctionnels qui peuvent être mutuellement exclusifs et au moment de l'exécution un seul service sera choisi parmi l'ensemble d'alternatives proposées.

Conformément au profil UML proposé dans la Figure 2.5, nous représentons le choix alternatif entre plusieurs services fonctionnels au sein d'un service domaine grâce à l'association <<Instance>>.

- Notation graphique

Le service domaine avec un choix alternatif est représenté en Figure 2.12. En effet, le service domaine A orchestre séquentiellement deux services fonctionnels B et C. Comme nous pouvons le remarquer, le service domaine peut avoir au moment de l'exécution un choix alternatif entre deux instances différentes du service fonctionnel B (noté comme B'et B''). Le choix entre les deux instances constitue l'adaptation au niveau du service domaine.

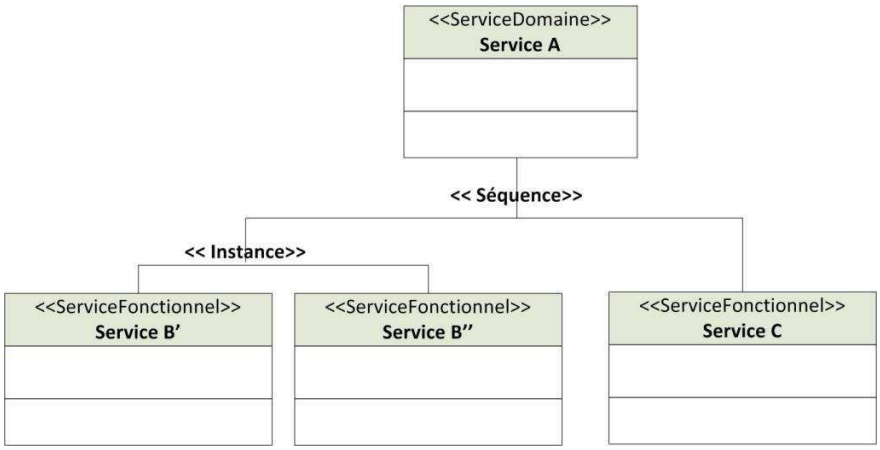


Figure 2.12. Service domaine avec choix d'instance

- Service domaine avec choix de schéma

À la différence du cas d'adaptation précédent où l'on est dans une logique de choix alternatif entre deux services fonctionnels, le choix du schéma introduit une adaptation qui porte sur des enchainements alternatifs d'un ensemble de services fonctionnels. Ainsi, chaque service domaine peut avoir un ensemble de schémas d'orchestrations alternatifs et exclusifs. Au moment de l'exécution, le service domaine sélectionnera le schéma qui correspondra au mieux à son contexte d'utilisation.

Conformément au profil UML proposé dans la Figure 2.5, nous adoptons l'association <<*SchémaOrchestration*>> afin de désigner les différents schémas d'orchestration qu'un service domaine peut avoir.

- Notation graphique

La Figure 2.13 illustre la représentation graphique d'un service domaine A avec un choix entre deux schémas d'orchestration. Ces derniers sont reliés entre eux grâce à l'association <<SchémaOrchestration>> représentant ainsi l'adaptation du service domaine.

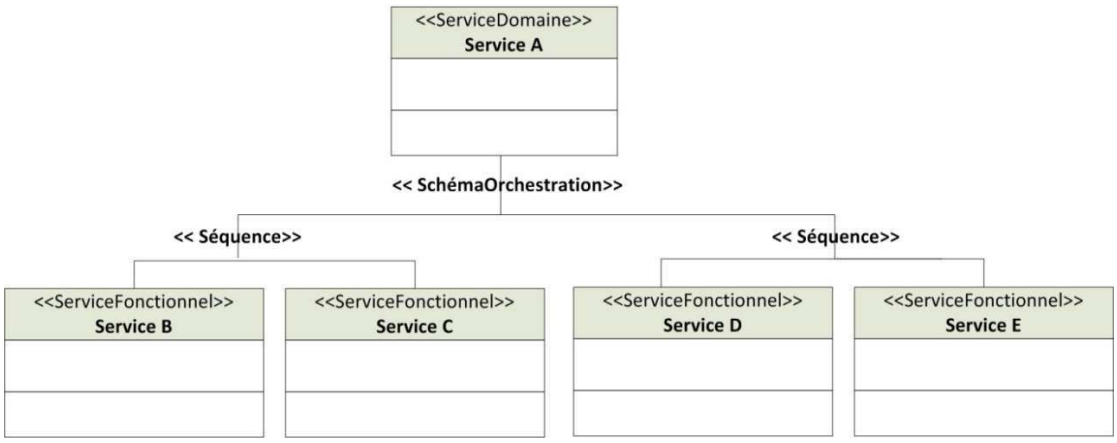


Figure 2.13. Service domaine avec choix de schéma d'orchestration

2.3.2 Services informatiques (ou services IT)

Les services informatiques (ou *services IT*) regroupent les fonctionnalités offertes par la partie informatisée du système d'information et sont utilisés par les services fonctionnels décrits précédemment. On distingue quatre catégories de services informatiques qui sont : les services d'accès au système *legacy*, les services entités, les services d'interface utilisateur et les services techniques.

Une autre catégorie de service peut être envisagée. Il s'agit des services applicatifs qui permettent de développer certaines fonctionnalités non existantes dans le patrimoine applicatif de l'entreprise. Les services applicatifs sont généralement proposés pour répondre à de nouveaux besoins métier.

D'une façon générale, les services informatiques ne sont pas associés à une couche de service particulière. Ils peuvent être en relation avec différents services indépendamment de leur type (service métier ou service informatique).

La Figure 2.14 permet de récapituler les principales catégories de services informatiques que nous avons retenues. Quelque soit sa catégorie, le service IT est identifié par un nom et possède une interface présentée en gris clair dans la Figure 2.14. L'interface du service informatique regroupe l'ensemble des opérations qu'il propose et lui permet d'établir des communications avec des services fonctionnels. En effet, le service fonctionnel utilise les services informatiques et ceci conformément à des contrats. Ces contrats, présentés en gris foncé dans la Figure 2.14, comportent un ensemble de messages ainsi qu'un ensemble de conditions relatives à l'utilisation des services IT.

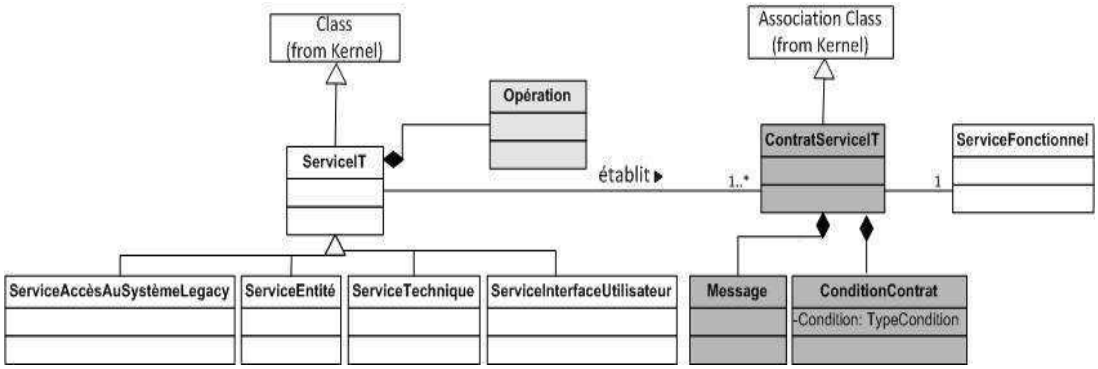


Figure 2.14. Méta-modèle du service informatique

2.3.2.1 Services d'accès au système legacy

Les services d'accès au système *legacy* permettent d'encapsuler le patrimoine applicatif (appelé encore le système *legacy*). Le système *legacy* de l'entreprise inclut les ressources applicatives mises en œuvre pour implémenter les fonctions de l'entreprise. Il est développé par différents intervenants au cours de nombreuses années afin de répondre aux besoins de ses utilisateurs et aux technologies de l'époque. L'objectif des services d'accès au système *legacy* est de mettre en place des façades homogènes au-dessus des applications informatiques existantes. À titre d'exemple, nous pouvons considérer un service qui encapsule l'application

informatique permettant de gérer la paie du personnel. En général, ce sont des services de grosse granularité.

2.3.2.2 Services d'interfaces utilisateur

Les services d'interfaces utilisateur permettent de gérer les communications et le dialogue avec les acteurs. En outre, les activités métier manuelles nécessitent d'être encapsulées par des services de type interfaces utilisateur. Ces derniers déclenchent par la suite des événements qui vont inciter les activités manuelles. Cette vision permet de répondre à l'un des objectifs de l'automatisation des processus métiers à savoir rationaliser et réduire le poids des activités manuelles lors de l'exécution des processus métiers.

2.3.2.3 Services entités

Les services entités sont les services qui se focalisent sur les objets métier clés du système d'information de l'entreprise. Il s'agit essentiellement des services qui permettent l'accès aux informations relatives aux objets métier. Typiquement, il s'agit des services qui réalisent des opérations de création, lecture, écriture et suppression (*CRUD*). Ces services correspondent à la notion de « Gestion des données de base » (*MDM* : *Master Data Management*).

2.3.2.4 Services techniques

Les services techniques regroupent les services de plus bas niveau permettant de gérer l'infrastructure du système d'information de l'entreprise. Les services techniques fournissent des services transverses, et relativement indépendants du métier de l'entreprise, comme les services liés à la gestion du réseau, à la messagerie et à l'éditique.

2.4 Phases de CSOMA

CSOMA est une démarche méthodologique à quatre phases mettant en exergue une double démarche de construction de l'architecture de services au sein de l'entreprise : la construction de la SOA métier et la construction de la SOA IT. Chaque phase de la démarche CSOMA peut comporter une ou plusieurs étapes et peut être réalisée soit par un intervenant métier, soit par un intervenant IT soit par les deux au même temps (voir Figure 2.15). Nous allons présenter en détail les différentes phases de CSOMA en soulignant pour chaque phase l'ensemble des livrables et les intervenants impliqués.

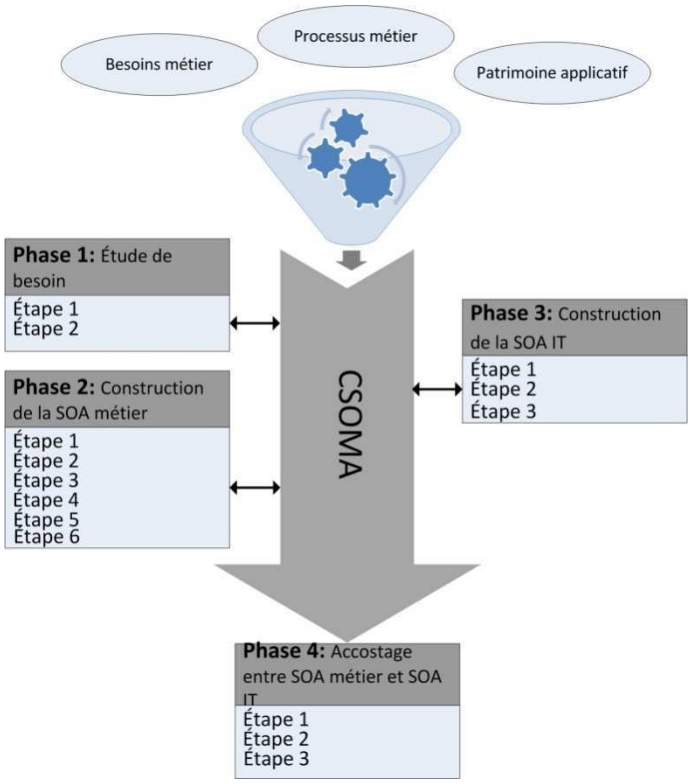


Figure 2.15. Phases proposées par la démarche CSOMA

2.4.1 Phase 1 : Étude de besoin

La première phase de CSOMA est réalisée conjointement par les intervenants métier et les intervenants IT de l'entreprise. Elle comporte deux étapes majeures : l'élaboration du ou des modèles de motivation métier et l'étude de l'architecture d'entreprise.

- **Étape 1 : Élaboration du modèle de motivation métier**

En préalable à la mise en place d'une architecture de services, il est important de définir les motivations qui conduisent à un tel effort. Les motivations métiers doivent être définies, et déterminent au plus haut niveau les attentes vis-à-vis de l'architecture de services à élaborer. Elles doivent donner naissance à un plan métier indiquant les moyens à mettre en œuvre pour les satisfaire.

Afin de déterminer les motivations métier pour une démarche d'architecture de services, nous nous sommes basés sur le standard appelé modèle de motivation métier (*Business Motivation Model*) (BMM) proposé par l'OMG [OMG, 2005]. Rappelons que le BMM (cf. chapitre 1, section 1.2.4.1.2) fournit une structure organisée pour la spécification, la communication et la gestion des plans métier. Il permet notamment d'identifier les facteurs qui motivent la mise en place des plans métier et retrouve les éléments clés de ces plans. En outre, il précise les relations qui peuvent exister entre les facteurs et les éléments des plans métier. Il existe trois briques de base pour le modèle de motivation métier : les finalités (*Ends*), les moyens (*Means*) et les *facteurs d'impact* (*Influencers*).

L'utilisation du modèle BMM dans notre démarche peut se justifier par plusieurs arguments :

- Le BMM est un standard de l'OMG ce qui respecte l'un des principes fondateurs de CSOMA qui est le principe d'ouverture.

- La construction de l'architecture de services au sein de l'entreprise doit être déduite des objectifs, des stratégies et des politiques métier. L'énumération et la définition des objectifs attendus d'une approche SOA permet de répertorier les besoins fonctionnels pour le système d'information et l'organisation de l'entreprise. Ces besoins seront exploités pour identifier les services d'entreprise. À défaut, l'architecture résultante ne respectera que partiellement l'alignement entre les objectifs métier et les fonctionnalités proposées par les services.

Dans le cadre de CSOMA, nous nous intéressons aux deux briques (moyens et finalités) lors de l'élaboration du modèle de motivation métier.

- **Étape 2 : Étude de l'architecture d'entreprise existante et cible**

Cette étape de la démarche CSOMA prend son origine de l'urbanisation. L'urbanisation est une pratique francophone dont l'objectif est d'organiser la transformation progressive du système d'information visant à le simplifier, à optimiser sa valeur ajoutée et à le rendre plus réactif et flexible vis-à-vis des évolutions stratégiques [Longépé, 2004]. Une des phases phare de l'urbanisation est l'étude de l'architecture d'entreprise. Il s'agit de dresser un ensemble de cartographies du système d'information et du métier de l'entreprise. La réalisation des cartographies permet, au travers d'une modélisation et d'un référentiel commun, d'être un support partagé qui permet la communication entre les différents acteurs du projet de mise en place d'une architecture de services. La cartographie est dans ce cas un outil facilitant la communication entre les maîtres d'œuvre et les maîtres d'ouvrage. Cependant, pour assurer cette finalité elle doit répondre à un ensemble de qualificatifs :

- Globale : Elle doit couvrir l'ensemble des activités de l'entreprise et permet d'avoir une vision synthétique de l'ensemble.

- Communicante : elle doit être compréhensible par toute l'entreprise afin de permettre de nombreuses utilisations (analyses métier, implémentations informatique, analyses des processus, *etc.*).

- Stable dans le temps : elle doit décrire les invariants qui entrent dans la composition du système d'information pour qu'il supporte les processus métiers de l'entreprise.

À partir des cartographies, les dirigeants disposent d'une connaissance formalisée et partagée des processus métiers, de l'architecture fonctionnelle et technique, des référentiels et des données, nécessaires pour analyser, construire et adapter le système d'information aux enjeux métiers de l'entreprise [Longépé, 2004]. Cet acquis est considéré comme le support permanent à la réflexion pour toute la démarche CSOMA.

2.4.2 Phase 2 : Construction de la SOA métier

La deuxième phase de CSOMA est la phase de construction de la SOA métier. Cette dernière adopte l'un des principes fondateurs de l'approche SOA à savoir l'alignement du système d'information sur le métier de l'entreprise. Ainsi, elle part de la définition (ou de la formalisation) des processus métiers pour descendre ensuite au travers des différentes strates

du système d'information pour définir les services métier nécessaires à la réalisation de ces processus. Six étapes constituent la phase de construction de la SOA métier :
Étape 1 : Modélisation contextuelle des processus métier,

Étape 2 : Identification des services métier,

Étape 3 : Raffinement des services fonctionnels,

Étape 4 : Identification des Aspects Conceptuels,

Étape 5 : Modélisation des Aspects Conceptuels,

Étape 6 : Création du ou des Modèles de Schéma d'Orchestration de Services.

Dans la suite, nous allons examiner en détail ces étapes.

- **Étape 1 : Modélisation contextuelle des processus métiers**

Cette étape de la démarche CSOMA se base sur un livrable de la première phase à savoir la cartographie des processus métiers. Une fois cartographiés, il s'agit par la suite d'identifier les processus candidats pour le passage en mode service. Certaines conditions doivent être remplies pour qu'un processus métier puisse être considéré comme un candidat potentiel. Dans le cadre de CSOMA, nous considérons un processus métier éligible à une démarche SOA s'il remplit en tout ou une partie des conditions suivantes :

- Le processus permet de répondre à un des objectifs stratégiques décrits dans le ou les modèles de motivation métier de l'entreprise. Ainsi, il présente une forte valeur ajoutée pour le métier de l'entreprise. Ceci garantit en retour un réel apport métier et un retour sur investissement notable.

- Le nombre d'activités métier le composant est significatif. En effet, on traite ici des processus volumineux dont les activités peuvent correspondre à des fonctions réutilisables, donc évidemment candidates à devenir des services métier.

- La fréquence de changement du processus est élevée. Il s'agit des processus qui évoluent rapidement dans le temps et qui nécessitent une certaine flexibilité lors de leur conception et de leur mise en œuvre. Ces caractéristiques peuvent être satisfaites grâce au passage en mode service.

- Le nombre d'activités à automatiser au sein du processus est important. Ceci traduit une volonté d'automatiser de plus en plus les activités pour des raisons de volumétrie et de fiabilité.

L'ensemble de processus métiers candidats est par la suite modélisé. Nous considérons que la modélisation des processus métiers doit prendre en compte deux caractéristiques importantes :

- La première consiste à modéliser les processus tout en prenant en considération la réalité organisationnelle et les contingences des outils informatiques. Nous considérons qu'une modélisation qui n'intègre pas les acteurs ni les systèmes informatiques peut déboucher sur une identification des services qui soit loin du contexte organisationnel de l'entreprise et reste au niveau théorique. Cette caractéristique orientera notre choix sur le langage de modélisation à utiliser.

- La deuxième caractéristique de la modélisation des processus tient au fait qu'elle doit intégrer la connaissance relative au contexte. Cette dernière associée avec l'ingénierie des processus est un domaine de recherche en émergence [Rosemann and Recker, 2006] [Saidani and Nurcan, 2007]. Nous considérons qu'une modélisation contextuelle des processus constitue la base de l'identification des services métier adaptables. Dans le cadre de CSOMA, nous définissons cette connaissance relative au contexte d'un processus métier comme l'ensemble des informations (ou paramètres) pouvant affecter la conception et l'implémentation d'un processus.

En considérant ces deux caractéristiques, une question importante se pose : quelles sont les informations contextuelles qui peuvent être considérées comme des informations pertinentes pour caractériser le contexte d'un processus métier. À cet égard, nous proposons une ontologie de catégorisation de contexte (Figure 2.16) qui illustre trois catégories de paramètres contextuels [Boukadi et al., 2009b] :

- Le contexte environnemental : permet de situer le processus par rapport à certaines informations spatiaux/temporelles. Certains processus métiers peuvent être sensibles au temps, d'autres à la localisation. Le contexte environnemental peut être renseigné par les analystes métier qui savent a priori quelles informations environnementales peuvent affecter le comportement d'un processus.
- Le contexte fonctionnel : traite des paramètres contextuels qui sont en relation directe avec le métier du processus. Cette catégorie de contexte inclut trois sous-catégories de base à savoir : le contexte du rôle, les règles métier et le but stratégique.

  - Le contexte du rôle traite de la sensibilité d'un processus métier au rôle (acteur ou ressource). Les informations concernant un acteur décrivent les caractéristiques de l'acteur comme ses compétences, son profil et ses droits d'accès. Le contexte d'une ressource peut inclure sa disponibilité, son coût, *etc*. Ces informations sont pertinentes lors du passage en mode service dans la mesure où elles vont permettre d'attribuer les contraintes que les services d'entreprise doivent respecter.
  - Les règles métier expriment les directives et les politiques métiers qui gouvernent les activités d'un processus.
  - Le but stratégique abstrait les détails des fonctionnalités proposées par un processus pour se concentrer sur son essence, c'est-à-dire le but qu'il doit atteindre. Le but stratégique d'un processus est extrait à partir du modèle de motivation métier.
- Le contexte non fonctionnel : comme son nom l'indique cette catégorie de contexte traite des paramètres qui n'ont pas de lien direct avec les fonctionnalités métiers du processus. Le contexte non fonctionnel peut inclure par exemple des paramètres de qualité et de sécurité.

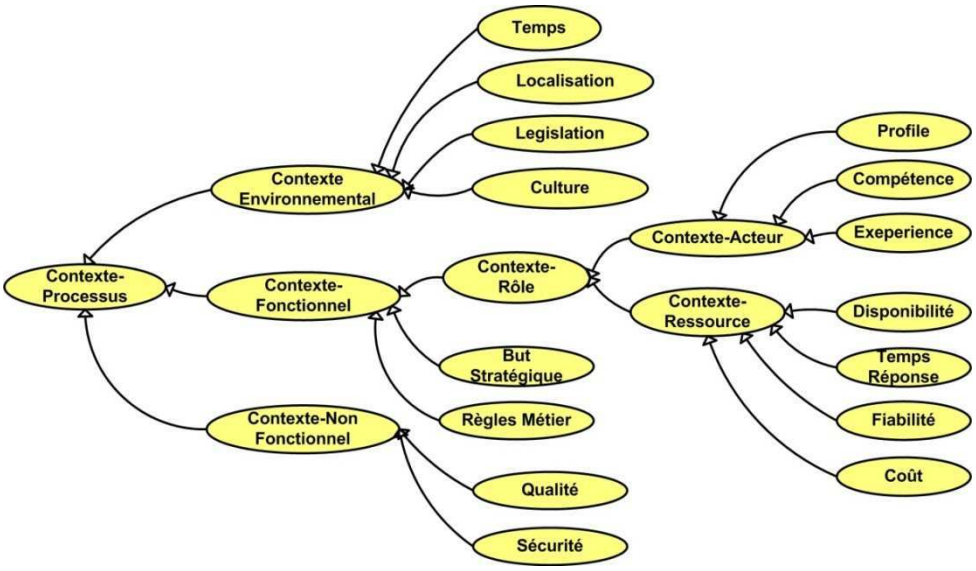


Figure 2.16. Ontologie de catégorisation de contexte d'un processus métier

En somme, ces catégories de contexte permettent de percevoir la situation d'un processus métier dans son environnement et doivent être prises en compte lors de l'identification des services métier. Ces derniers présenteront des comportements adaptables en harmonie avec les informations contextuelles des processus métiers.

Afin de modéliser les processus métiers, nous nous sommes référés à l'étude de la littérature présentée dans le chapitre 1 de ce manuscrit de thèse (cf. chapitre 1, section 1.1.4.3). Cette étude a montré l'importance du standard BPMN [BPMI, 2004] pour la modélisation des processus. Le standard BPMN fournit une notation compréhensible par tous les acteurs de l'entreprise, depuis les analystes métier jusqu'aux développeurs. Ainsi, nous le retenons pour la modélisation des processus métiers dans le cadre de CSOMA.

Pour répondre à la spécificité de la modélisation contextuelle, nous avons enrichi la modélisation BPMN avec un ensemble d'annotations contextuelles. Ces dernières sont représentées grâce au concept d'objets symboliques appartenant au formalisme BPMN.

- **Étape 2 : Identification des services métier**

Dans le cadre de la construction de la SOA métier, CSOMA adopte une démarche orientée processus pour l'identification des services métier et se base sur deux principes fondateurs :

- L'identification des services ne peut pas être déconnectée de leur usage, c'est-à-dire des processus métiers qu'ils servent. Notre position est donc d'identifier les services métier en adoptant une démarche orientée processus (voir Figure 2.17).

- L'identification des services doit minimiser la discordance conceptuelle entre la définition des services et l'énoncé des buts formulés dans le modèle de motivation métier. Ceci permet de réduire le décalage entre les expressions fonctionnelles des services et celles des buts formulés par les analystes métier. Ainsi, nous proposons une démarche d'identification des

services qui met en avant les buts que ces services permettent d'atteindre, et non plus, les fonctionnalités qui leur permettent de les atteindre.

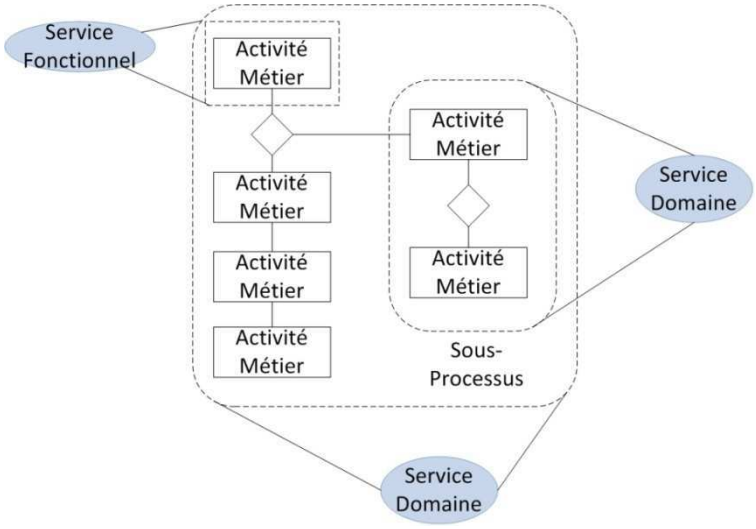


Figure 2.17. Principe d'identification des services métier à partir des processus

CSOMA se base sur la notion de but pour identifier les services métier (fonctionnel ou de domaine). Un but exprime une intention, un objectif que l'on souhaite atteindre. Selon [Jackson, 1995] et [Zave and Jackson, 1997], un but est une déclaration « *optative* » qui exprime ce que l'on veut, un état ou un résultat que l'on cherche à atteindre sans entrer dans les détails du processus d'atteinte. Un but peut être formulé à des différents niveaux : stratégique, tactique et opérationnel [Lamsweerde et al., 1995], [Ralyté, 2001]. Le concept de but a montré sa pertinence dans plusieurs domaines de recherche tels que dans l'ingénierie des besoins [Rolland et al., 1999], [Yu, 1997], [Dardenne et al., 1993], ou encore dans le BPR (*Business Process Reengineering*) [Yu and Mylopoulos, 1994], [Anton et al., 1994].

Dans le cadre de CSOMA, l'idée motrice consiste à étudier le but stratégique pour chaque processus métier. Comme nous l'avons déjà mentionné, chaque processus métier comporte dans son contexte fonctionnel un but stratégique (BS) qu'il souhaite satisfaire. Le but stratégique peut être décomposé en un ou plusieurs buts tactiques (BT). Ces derniers servent à identifier le ou les services domaines indispensables pour l'implémentation du processus métier en question. Par la suite, il s'agit de décomposer les buts tactiques (BT) en des buts opérationnels (BO). Fondamentalement, les buts opérationnels devraient être satisfaits par les activités métier. On étudie le degré de couplage entre les activités métier par rapport aux buts opérationnels que ces dernières permettent de satisfaire afin d'identifier les services fonctionnels.

La Figure 2.18 illustre le principe de mise en correspondance entre les buts d'un processus métier et les services métier.

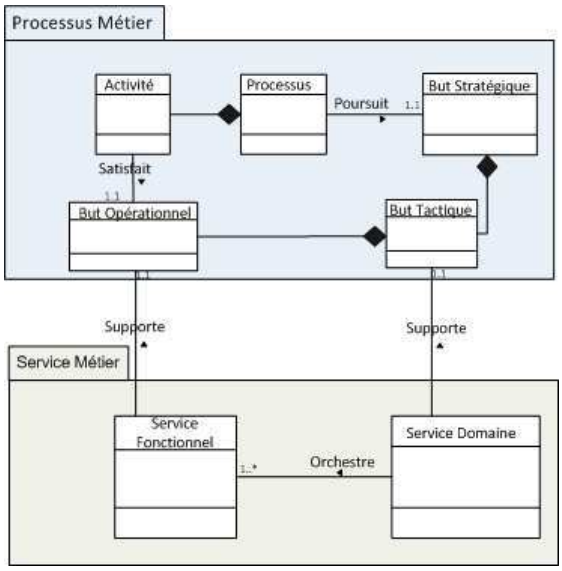


Figure 2.18. Mise en correspondance entre les buts et les services métier

Nous détaillons dans ce qui suit, les quatre sous étapes majeures de l'identification des services métier à savoir : la décomposition des buts métier, l'identification des services domaines, l'identification des services fonctionnels et la modélisation des services fonctionnels.

– Étape 2.1 : Décomposition des buts métier

Il s'agit d'étudier pour chaque processus métier le but stratégique qu'il poursuit afin de procéder par la suite à sa décomposition. Pour réaliser cette finalité, nous nous sommes basés sur le concept de graphes de décomposition de buts inspiré de la discipline de l'intelligence artificielle [Nilsson, 1971]. Une décomposition en but/sous-buts procède du but principal **A**. La réalisation de ce but principal nécessite l'accomplissement de plusieurs sous-buts. Certains de ces sous-buts nécessitant eux-mêmes d'être décomposés en des sous-buts plus élémentaires. La décomposition s'arrête lorsqu'un but ne peut plus être décomposé et qu'il est réalisé par l'accomplissement d'un ensemble d'actions. Deux types de décomposition existent à savoir : la décomposition ET et la décomposition OU. La décomposition OU d'un but père **A** en des buts fils **B** et **C** signifie que si **B** ou **C** est atteint alors **A** est lui-même atteint. La décomposition ET d'un but père **A** en des buts fils **B**, **C** signifie que si **B** et **C** sont atteints, alors **A** l'est. Ainsi, la décomposition ET permet d'affiner un but de niveau d'abstraction *i* en un but de niveau d'abstraction inférieur *i+1*. C'est le cas quand on décompose un but stratégique en des buts tactiques ou un but tactique en des buts opérationnels.

CSOMA tire profit de mécanisme de décomposition ET des buts afin de décomposer le ou les but(s) stratégique(s) d'un processus métier en des buts tactiques et les buts tactiques en des buts opérationnels (voir Figure 2.19).

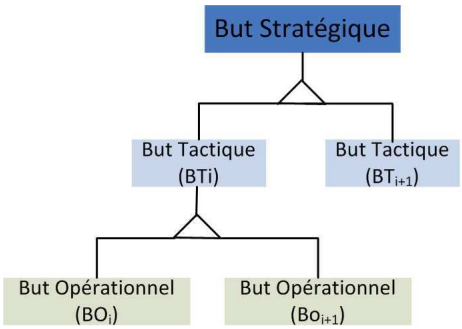


Figure 2.19. Arbre de décomposition des buts dans CSOMA

Formellement l'arbre de décomposition des buts peut être considéré comme un graphe G, noté G (X, U) avec X est l'ensemble fini des sommets qui ne sont autres que les buts et U est l'ensemble d'arcs reliant les sommets. Ce graphe sera exploité dans l'étape 2.2 de la démarche CSOMA.

– Étape 2.2 : Identification des services domaines

Cette étape permet d'identifier les services domaines en étudiant les buts tactiques que le processus cherche à atteindre. Comme il a été déjà mentionné, les services domaines sont des services composites qui encapsulent des processus ou des sous processus métier et constituent la brique de base pour la construction du processus coopératif. Ceci justifie bien leur identification basée sur les buts tactiques. En effet, les services domaines contribuent à réaliser la stratégie de l'entreprise que ce soit à l'interne ou à l'externe. À l'interne, ils permettent de satisfaire les buts tactiques des processus en orchestrant un ensemble de services fonctionnels. À l'externe, ils permettent à l'entreprise de participer à des processus coopératifs à la demande et doivent satisfaire à cet égard un but tactique que le processus ou le sous processus externe souhaite satisfaire.

L'identification des services domaines à partir des buts tactiques est à la charge de l'analyste métier de l'entreprise. Cette identification se réalise en examinant l'arbre de décomposition du but stratégique du processus. Au principe chaque but tactique peut être satisfait par un et un seul service domaine. En outre, quelques bonnes pratiques peuvent guider l'analyste métier à savoir :

- Le ou les services domaines à identifier sont fortement reliés au processus métier. À l'encontre de l'identification des services fonctionnels, l'identification des services domaines ne s'inscrit pas dans une optique de réutilisation. Il ne s'agit pas de soucier si le service domaine peut être réutilisé dans un autre processus. Par exemple, un service domaine de gestion des formations des employés est fortement relié au processus métier de gestion des formations et ne peut pas participer à un autre processus.
- L'identification d'un service domaine ne doit pas se préoccuper des traitements qu'il offre. En effet, d'après sa définition un service domaine assure la fonction d'orchestration d'un ensemble de services fonctionnels. Cette fonction d'orchestration est décrite en détail plus loin dans la démarche.

– Étape 2.3 : Identification des services fonctionnels

Chaque processus métier dont les buts ont été raffinés en des buts opérationnels, va être analysé afin d'identifier les activités automatiques et les activités manuelles. Les premières vont être représentées sous forme d'opérations de services fonctionnels alors que les secondes seront représentées sous la forme de services d'interface utilisateur. Notre proposition consiste à regrouper les activités métier fortement couplées au sein d'un même service. L'idée étant d'étudier le couplage entre les activités d'un processus métier en examinant le but opérationnel que chacune d'entre elles poursuit. L'hypothèse que nous adoptons est la suivante :

> Si une activité **A** satisfait un but opérationnel **BO** issu de la décomposition d'un but tactique **BT** et si une activité **B** satisfait le même but opérationnel **BO** alors **A** et **B** sont candidats pour appartenir à un même service.

Cette hypothèse est reprise dans l'algorithme de regroupement que nous proposons.

Afin

Algorithme de regroupement des activités au sein des services
Input
Graphe **G (X, U)** tel que Types (X) = {BS, BT, BO} // * *L'arbre de décomposition des buts comme un graphe*
AM : l'ensemble des activités métier,
BT_i : l'ensemble des sommets de X tel que le type de sommet est BT,
Output
A_i : l'ensemble d'activités résultat du regroupement
Début
Pour chaque activité a ∈ AM **Faire** //* *Pour chaque activité métier*
a ∈ $\mathbf{A_i}$ si et seulement si a Satisfait **BO** tel que ∃ u∈ U avec u = (BT_i, BO)
Fin Faire
Fin

d'appliquer l'algorithme, nous proposons de réaliser une matrice de correspondance telle qu'elle est illustrée sur le Tableau 2.1.

BO / AM	BO_j	BO_{j+1}	BO_{j+2}	…..	BO_{j+n}
AM_a	X				
AM_b		X			
AM_c					X
AM_d			X		

Tableau 2.1. Mise en correspondance entre les activités métier et les buts opérationnels

La Figure 2.20 illustre un exemple de groupement de deux activités métier dans un service fonctionnel appelé « Service Calcul Coût Livraison ».

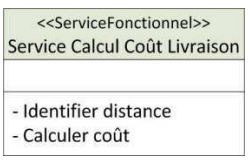


Figure 2.20. Exemple d'un service fonctionnel regroupant deux activités métier

D’autres critères peuvent appuyer la démarche de regroupement des activités métier. Des exemples de ces critères incluent : l’homogénéité du rythme de déroulement des activités, l’homogénéité du cycle de vie des informations créées par les activités, la cohérence des relations avec les autres sous-systèmes et leurs acteurs, la propriété d’échange d’information, et aussi la cohérence du périmètre de travail des activités.

Il est important de préciser que cette identification est une première identification, et à ce stade de la démarche, les activités encapsulées par un service fonctionnel ne constituent pas ses opérations. L’étape suivante de CSOMA se focalisera sur la modélisation des services fonctionnels et décrira en détail les différents traitements proposés par ces services.

- Étape 2.4 : Modélisation des services fonctionnels

La modélisation des services métier dans le cadre de CSOMA s’inscrit dans le niveau PIM de l’approche MDA et plus spécialement elle suit une transformation de PIM vers PIM. Ces transformations visent à enrichir, filtrer ou spécialiser les modèles de services sans se préoccuper des choix de plateformes (voir Figure 2.2). Les différentes transformations de PIM vers PIM sont utilisées pour le raffinement de modèle de services et présentent ainsi en détail les modèles au sein de la SOA métier. Nous avons choisi le langage UML [Booch et al., 2004] pour modéliser les différents modèles de services métier. Les modèles proposés étendent les diagrammes UML en utilisant de nouveaux stéréotypes. La figure suivante, est extraite de la Figure 2.2, elle présente les trois modèles proposés pour modéliser les services fonctionnels.

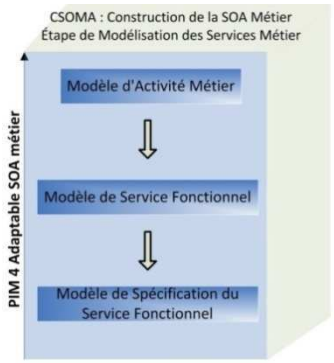


Figure 2.21. Modèles PIM vers PIM pour les services fonctionnels

Dans la suite nous allons décrire en détail les différents modèles proposés ainsi que les règles de transformation qui permettent le passage d’un modèle à un autre.

- Le Modèle d’Activité Métier

Le Modèle d’Activité Métier est une extension du diagramme de cas d’utilisation UML. Il permet de décrire d’une manière microscopique le ou les activités métier encapsulées par le service fonctionnel identifié à partir de l’étape 2.3 de la démarche CSOMA. En effet, chaque activité métier peut être décomposée en une ou plusieurs tâches en se basant à la fois sur la description du processus métier et sur le but opérationnel qu’elle doit réaliser. Certaines activités métier sont de fines granularités, dans ce cas-là, elles ne sont pas décomposables en tâches. Une activité métier est représentée par un cas d’utilisation spécial appelé *ActivMétier*.

Tandis qu'une tâche est représentée par un cas d'utilisation spécial appelé *Tâche*. Le Tableau 2.2 illustre les stéréotypes proposés pour représenter le Modèle d'Activité Métier.

ActivMetier	
Méta-classe UML	étend la méta-classe « *Use Case* » d'UML
Signification	Représente une activité métier
Notation	<< *ActivMétier* >>

Tâche	
Méta-classe UML	étend la méta-classe « *Use Case* » d'UML
Signification	Décrit en détail les différentes fonctionnalités supportées par une activité métier
Notation	<< *Tâche* >>

Tableau 2.2. Stéréotypes proposés pour représenter le Modèle d'Activité Métier

- Le Modèle de Service Fonctionnel

Le Modèle de Service Fonctionnel est aussi une extension du diagramme de cas d'utilisation UML. Il se concentre sur la description du service fonctionnel et notamment sur les opérations qu'il propose. En effet, chaque tâche identifiée dans le modèle précédent va être transformée en une opération du service fonctionnel. Cette dernière est représentée comme un cas d'utilisation spécial appelé *OperationService*. Dans ce modèle, nous identifions deux types d'opération : opération de base notée comme *OperationBase* et opération de données notée *OpérationDonnées*. Les opérations de base permettent de mettre en œuvre la logique métier préconisé par le service. Tandis que les opérations de données, comme leur nom l'indique, permettent l'accès à des référentiels de données afin de récupérer certaines informations. Le Tableau 2.3 décrit en détail les éléments du modèle de service fonctionnel.

OperationBase	
Méta-classe UML	étend la méta-classe « *Use Case* » d'UML.
Signification	Représente une opération de base supportée par un service fonctionnel
Notation	<<OB>>

OpérationDonnées	
Méta-classe UML	étend la méta-classe « *Use Case* » d'UML
Signification	Représente une opération de service fonctionnel qui nécessite l'accès à une ou à plusieurs bases de données
Notation	<<OD>>

Tableau 2.3. Stéréotypes proposés pour représenter le Modèle de Service fonctionnel

- Le Modèle de Spécification du Service Fonctionnel

Le Modèle de Spécification du Service Fonctionnel est une extension du diagramme d'activités UML. Il propose une description détaillée du flux d'exécution des différentes opérations du service fonctionnel. Pour ce faire, ce modèle présente un nouveau concept intitulé *ServiceActivitéOpération*. Nous identifions deux types d'opérations : une opération de base représentée comme une activité UML spéciale appelée *ServiceActivitéOperationBase* et une opération de données est représentée comme *ServiceActivitéOperationDonnées* (voir Tableau 2.4).

ServiceActivitéOperationBase	
Méta-classe UML	étend la méta-classe « Activity » d'UML
Signification	Représente le comportement d'une opération de base qui fait partie de la spécification d'un service fonctionnel
Notation	<<SAOB>>

ServiceActivitéOperationDonnées	
Méta-classe UML	étend la méta-classe « Activity » d'UML
Signification	Représente le comportement d'une opération de données qui fait partie de la spécification d'un service fonctionnel
Notation	<<SAOD>>

Tableau 2.4. Stéréotypes proposés pour le Modèle de Spécification du Service Fonctionnel

En somme, la modélisation du service fonctionnel suit une transformation du PIM vers PIM du cadre MDA et inclut deux modèles intermédiaires pour aboutir enfin au Modèle de Spécification du Service Fonctionnel. Le passage entre les différents modèles est réalisé grâce à un ensemble de règles de transformation qui permettent de passer d'un modèle source à un modèle cible. Le modèle source reste intact alors qu'il ya génération d'un nouveau modèle, résultat de la transformation. Les règles de transformation peuvent être décrites de plusieurs façons. Selon [Miller and Mukerji, 2003], les règles de transformation peuvent être décrites en utilisant le langage naturel, ou un algorithme écrit en utilisant un langage d'action (*action language*) ou encore à l'aide d'un modèle de *mapping* (modèle qui permet de relier les concepts des deux modèles sources et cibles) (cf. chapitre 1, section 1.2.4.1). Dans le cadre de nos travaux, nous avons décidé d'adopter le langage naturel pour décrire les règles de transformation entre les modèles. Notre choix se justifie essentiellement par le fait que les règles de transformation ne sont pas assez complexes pour qu'elles soient décrites en utilisant un algorithme ou encore en utilisant un modèle de *mapping*. Ainsi, nous nous sommes inspirés des travaux de [Miller and Mukerji, 2003] pour réaliser les règles de transformation des trois modèles (voir Figure 2.21). Ces différentes règles peuvent être complètement automatique (désignées par la lettre **A**) ou encore semi-automatique (désignées par la lettre **S**). Dans ce cas-là, elles nécessitent l'intervention de l'analyste métier.

Modèle Source	Modèle Cible	Règle de Transformation	Automatisation

Modèle d'Activité Métier	Modèle de Service Fonctionnel		
		R_1: Chaque << *ActivMétier* >> ou << *Tâche*>> contenue dans le Modèle d'Activité Métier se transforme en une opération dans le Modèle de Service Fonctionnel (*OB* ou *OD*)	**S**
Modèle de Service Fonctionnel	**Modèle de Spécification du Service Fonctionnel**	**R_2 :** Pour chaque opération de service fonctionnel <<*OB*>> ou <<*OD*>>, il va y avoir une activité (au sens UML) qui va être générée	**A**
		R_3 : Chaque relation d'inclusion <<*include*>> qui relie deux cas d'utilisation <<OB>> ou <<OD>> dans le Modèle de Service Fonctionnel, se transforme en deux activités que ce soit <<SAOB>> ou <<SOAD>> dont l'ordre de déclenchement est le suivant : L'activité qui correspond au cas d'utilisation source de la relation d'inclusion se réalise après le cas d'utilisation cible	**A**
		$R_{3.1}$: Si la relation d'inclusion contenue dans le Modèle de Service Fonctionnel comporte plusieurs cas d'utilisation cibles, c'est à la tâche de l'analyste métier de décider de l'ordre d'exécution des différentes activités <<SAOB>> ou <<SOAD>> contenues dans le Modèle de Spécification du Service Fonctionnel	**S**
		R_4 : Chaque relation d'extension <<*extend*>> se transforme en un flot de contrôle de type « *Fork* » dans le Modèle de Spécification du Service Fonctionnel. L'activité <<SOAB>> ou <<SOAD>> qui correspond au cas d'utilisation source (celui qui étend un autre cas d'utilisation) doit se déclencher après l'activité qui correspond au cas d'utilisation étendu. - Si la relation d'extension implique seulement deux cas d'utilisation, alors le flot de contrôle de type « *Fork* » permet de présenter l'activité qui correspond au cas d'utilisation source avec une alternative de type transition vide (ne comportant pas d'activité). Par la suite, le « *Fork* » est résolu par un « *Join* » afin de regrouper les deux transitions - Si la relation d'extension implique plusieurs cas d'utilisation sources et un seul cas d'utilisation cible, alors le flot de contrôle de type « *Fork* » présentera l'ensemble des alternatives correspondant aux cas d'utilisation sources avec une alternative de type transition vide. Par la suite, le « *Fork* » est résolu par	**A**

		un « *Join* » afin de regrouper les différentes transitions[13]	

Tableau 2.5. Règles de transformation des modèles de service fonctionnel

- **Étape 3 : Raffinement des services fonctionnels**

Les services fonctionnels ainsi identifiés et modélisés doivent être validés et raffinés par les analystes métier de l'entreprise. L'étape de raffinement est une étape importante dans le cadre de CSOMA. Elle vise à revoir les services candidats afin de répondre au mieux aux deux propriétés de base des services, qui sont : la propriété de réutilisation et la propriété d'autonomie. La propriété de réutilisation permet de s'assurer qu'une opération d'un service ou un service peut être réutilisé dans d'autres cas d'usage. Il s'agit essentiellement de changer le nom de l'opération (ou du service) si cette dernière n'obéit pas à la propriété de réutilisation. Par exemple, si une opération d'un service fonctionnel s'intitule « Calcul montant de formation », cette opération ne peut être utilisée que pour calculer le prix d'une formation. Pour rendre cette opération plus réutilisable il serait intéressant de changer son nom en « Calcul montant ».

Quant à la propriété d'autonomie, elle permet de s'assurer que d'une part, les opérations d'un service sont faiblement couplées entre elles et, d'autre part, que les différents services fonctionnels sont faiblement couplés entre eux. Cette propriété suppose qu'au niveau d'un service fonctionnel, si deux opérations sont fortement couplées alors elles doivent être combinées en une seule opération. Quant au couplage faible entre les services, il consiste à ce qu'un service fonctionnel acquiert la connaissance d'un autre service en restant toujours indépendant de ce service.

Une autre considération fondamentale est la détermination de la granularité adéquate pour l'exposition des services fonctionnels. Un service offre un traitement de bout en bout ayant un sens métier pour ses clients. Il ne doit pas être de grosse granularité sous peine de perturber la compréhension fonctionnelle. Il ne doit pas être également de fine granularité sous peine de ne pas offrir une valeur métier suffisante.

Nous considérons que la granularité d'un service fonctionnel est déterminée par le nombre d'opérations incluses dans le service. En étudiant la granularité des services fonctionnels, il faut trouver le bon compromis entre des services qui englobent trop d'opérations et des services avec trop peu d'opérations.

Le Tableau 2.6 recense les différentes considérations qui doivent être prises en compte lors de la conciliation entre une granularité fine et une granularité forte. Cette conciliation illustre l'impact de la granularité sur un certain nombre de facteurs (valeur métier, possibilité de

[13] La règle R_4 sera expliquée en détail à travers un exemple dans l'annexe A de ce manuscrit de thèse.

composition, alignement, *etc.*). Suivant son besoin, l'entreprise privilégiera un ou plusieurs facteurs.

Granularité	Avantages	Inconvénients
Fine	- Possibilité de composer de nouveaux services plus larges - Grande possibilité de réutilisation des services	- Difficulté de cerner la valeur métier d'un service - Difficulté de gérer un grand nombre de services -Diminution des performances d'exécution due à la multiplication d'appel aux services
Forte	- Augmentation de la valeur métier proposé par le service - Amélioration de la performance d'exécution vu la diminution d'appels de services	- Difficulté de cerner les différentes fonctionnalités proposées par le service - Difficulté d'alignement du service sur le besoin métier

Tableau 2.6. Comparaison entre fine et forte granularité

Il est important de souligner que le raffinement des services fonctionnels entraînera la mise à jour des modèles de services fonctionnels, déjà élaborés dans l'étape précédente de la démarche.

- **Étape 4 : Identification des Aspects Conceptuels**

Comme nous l'avons déjà mentionné, une particularité du service métier est qu'il adapte son comportement en fonction du contexte d'utilisation. La réutilisation d'un service pour un nouvel usage conduit souvent à l'intégration d'une variante afin de modifier son comportement d'origine. Par exemple, le service doit s'adapter par partenaire, pays, *etc.* sur des aspects fonctionnels (extension d'une règle métier) et non fonctionnels (qualité, sécurité). Cette forte pluralité des contextes d'utilisation impose de s'intéresser à l'adaptation des services dès la phase de leur modélisation. En effet, lors de la modélisation d'un service, il est déterminant de spécifier son « modèle d'adaptation », c'est-à-dire l'ensemble des paramètres et des préoccupations qui gouvernent son comportement. Comme il a été déjà souligné, on s'inspire du concept de base de la programmation orientée aspect pour modéliser les paramètres ou les préoccupations qui influencent le comportement du service. À cet égard, nous définissons pour chaque service métier un ensemble d'Aspects Conceptuels. Un Aspect Conceptuel peut être : une exigence non fonctionnelle (par exemple la sécurité), une règle métier, ou encore une capacité d'adaptation en réponse à certains changement de contexte. Lorsqu'il s'agit d'identifier les Aspects Conceptuels des services métier, l'implication des analystes métier de l'entreprise est déterminante pour obtenir le meilleur niveau de généricité possible.

Un Aspect Conceptuel d'un service métier est identifié en analysant les paramètres contextuels du processus métier dont il est issu. En effet, les annotations contextuelles d'un processus métier doivent être prises en compte lors de l'intégration de l'adaptation au niveau des services métier. Outre les paramètres contextuels du processus métier, il faut mener un effort particulier d'analyse des cas d'usage connus et probables du service pour découvrir un

maximum de paramètres. Ces différents paramètres vont être encapsulés par un ensemble d'Aspects Conceptuels.

- **Étape 5 : Modélisation des Aspects Conceptuels**

Tout comme la modélisation des services métier, la modélisation des Aspects Conceptuels se positionne au niveau PIM et plus précisément de la transformation de PIM vers PIM du cadre MDA. L'objectif de cette étape est de fournir une modélisation détaillée des Aspects Conceptuels déjà identifiés. La Figure 2.22, extraite de la Figure 2.2 montre les différents modèles proposés. Ces modèles étendent les diagrammes UML en utilisant de nouveaux stéréotypes.

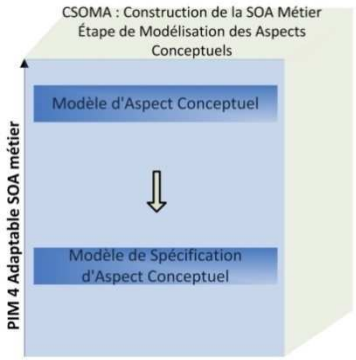


Figure 2.22. Modèles PIM vers PIM pour les Aspects Conceptuels

Dans la suite nous allons décrire en détail les différents modèles proposés ainsi que les règles de transformation qui permettent le passage d'un modèle à un autre.

- Le Modèle d'Aspect Conceptuel

Le Modèle d'Aspect Conceptuel, comme son nom l'indique, décrit en détail un Aspect Conceptuel. Ce modèle est une extension du diagramme de cas d'utilisation UML dans lequel chaque fonctionnalité proposée par un aspect est représentée grâce à un cas d'utilisation spécial nommé *AspectAdvice*. Chaque *AspectAdvice* peut être décomposé en un ensemble d'actions élémentaires. Chaque action élémentaire est représentée par un cas d'utilisation spécial appelé *AdviceAction*. Les différents stéréotypes déjà énumérés sont décrits dans le Tableau 2.7.

AspectAdvice	
Méta-classe UML	étend la méta-classe « Use Case » d'UML
Signification	Décrit une fonctionnalité proposée par un Aspect Conceptuel
Notation	<<AspectAdv>>
Contrainte	- le seul acteur supporté par le Modèle d'Aspect Conceptuel est le système - la seule relation supportée par le Modèle d'Aspect Conceptuel est la relation d'inclusion <<*include*>>

AdviceAction	
Méta-classe UML	étend la méta-classe « Use Case » d'UML
Signification	Décrit une action proposée par un AspectAdvice
Notation	<<AdviceAct>>
Contrainte	- le seul acteur supporté par le Modèle d'Aspect Conceptuel est le système - la seule relation supportée par le Modèle d' Aspect Conceptuel est la relation d'inclusion <<*include*>>

Tableau 2.7. Stéréotypes proposés pour le Modèle d'Aspect Conceptuel

- Le Modèle de Spécification d'Aspect Conceptuel

Le Modèle de Spécification d'Aspect Conceptuel complète la description de l'Aspect Conceptuel, dans le sens où il ajoute le flux d'exécution des différentes actions appartenant à l'aspect. Ce modèle est une extension du diagramme d'activités UML. Une activité dans le modèle de spécification d'Aspect Conceptuel est représentée comme un nœud spécial appelé « *AspectActivité* » (voir tableau suivant).

AspectActivité	
Méta-classe UML	étend la méta-classe « Activity » d'UML.
Signification	Décrit une action proposée par un AspectAdvice.
Notation	<<AspectActiv>>

Tableau 2.8. Stéréotype proposé pour le Modèle de Spécification d'Aspect Conceptuel

Le passage entre les différents modèles est réalisé grâce à un ensemble de règles de transformation. Le tableau suivant illustre les règles proposées. Ces dernières suivent la même logique que celles proposées dans le Tableau 2.5.

Modèle Source	Modèle Cible	Règle de Transformation	Automatisation
Modèle d'Aspect Conceptuel	**Modèle de Spécification d'Aspect Conceptuel**	**R_1 :** Pour chaque cas d'utilisation du Modèle d'Aspect Conceptuel (*AspectAdv* ou *AdviceAct*), il va y avoir une activité (au sens UML) qui va être générée	**A**
		R_2 : Chaque relation d'inclusion <<*include*>> qui relie le cas d'utilisation <<*AspectAdv*>> à un cas d'utilisation <<*AdviceAct*>> dans le Modèle d'Aspect Conceptuel, se transforme en deux activités <<AspectActiv>> dont l'ordre de déclenchement est le suivant : L'activité qui correspond au cas d'utilisation source de la relation d'inclusion se déclenche après le cas d'utilisation cible	**A**

		$R_{2.1}$: Si la relation d'inclusion contenue dans le Modèle d'Aspect Conceptuel comporte plusieurs cas d'utilisation cibles, c'est à la tâche de l'analyste (ou du concepteur) de décider de l'ordre d'exécution des différentes activités dans le Modèle de Spécification d'Aspect Conceptuel	**S**

Tableau 2.9. Règles de transformation des modèles d'Aspect Conceptuel

Le Modèle de Spécification d'Aspect Conceptuel peut être par la suite enrichi à l'aide d'un ensemble d'informations supplémentaires. Ceci nécessite une intervention humaine (et notamment des développeurs) afin de compléter efficacement le modèle de Spécification d'Aspect Conceptuel et de le projeter par la suite sur un support technologique. L'objectif est de créer un ensemble d'aspects (au sens de programmation orientée aspect) conforme à une technologie spécifique (langage de programmation orientée aspect tel qu'*AspectJ* [Kiczales et al., 2001b] ou framework tel que *Jboss AOP* [JBoss, 2006]). Comme nous l'avons déjà précisé, dans le cadre de nos travaux, nous ne nous intéressons pas aux règles de passage du niveau PIM au niveau PSM. Néanmoins, le développement des aspects pour la gestion de l'adaptation des services est proposé dans le chapitre 4 de ce manuscrit en utilisant une plateforme spécifique.

- **Étape 6 : Création du ou des Modèle de Schéma d'Orchestration de Services**

Le Modèle de Schéma d'Orchestration de Services suit le méta-modèle déjà proposé (voir Figure 2.6). Ainsi, l'objectif de cette étape est d'identifier les différents schémas d'orchestration du service domaine afin d'implémenter le processus métier associé. Il s'agit d'énumérer les enchaînements possibles de services fonctionnels. Ceci permet, d'une part, de définir les relations potentielles entre le service domaine et les différents services fonctionnels et d'autre part, d'illustrer le principe d'adaptation du service domaine déjà discuté (cf. section 2.3.1.3).

Nous rappelons que deux types d'adaptation ont été distingués à savoir : l'adaptation de la logique métier et l'adaptation de la logique d'orchestration. L'adaptation de la logique métier a déjà été traitée dans l'étape 4 de la démarche à travers l'attribution des différentes Aspects Conceptuels aux services fonctionnels. Quant à l'adaptation de la logique d'orchestration, les analystes métier identifient les services fonctionnels qui peuvent être mutuellement exclusifs ou encore les différents schémas d'orchestration pour chaque service domaine.

2.4.3 Phase 3 : Construction de la SOA IT

La troisième phase de CSOMA est la phase de construction de la SOA IT. L'enjeu majeur de la SOA IT est la réutilisation des composants informatiques (composants applicatifs, composants techniques, *etc.*). De nos jours, la construction de la SOA IT est moins problématique que celle de la SOA métier. Dans le cadre de CSOMA, cette phase est à la charge des responsables IT (informaticiens, développeurs, *etc.*). Elle débute par une analyse du portefeuille applicatif de l'entreprise afin d'examiner les fonctions développées. Ainsi

cartographiées, il est possible d'identifier les fonctions qui sont éligibles au rang de services informatiques.
Trois étapes fondamentales sont à considérer à savoir :

Étape 1 : Analyse du portefeuille applicatif de l'entreprise,

Étape 2 : Identification des services d'accès au système *legacy*,

Étape 3 : Identification des services entités.

Ces étapes sont décrites en détail dans ce qui suit.

- **Étape 1 : Analyse du portefeuille applicatif de l'entreprise**

L'entreprise a accumulé au fil du temps un patrimoine applicatif considérable. Applications sur mesure, logiciels standard, voir solutions de type Web plus récentes. Cet acquis applicatif contribue à divers degrés au capital stratégique et tactique de l'entreprise. Compte tenu du cadre de notre étude à savoir les PME, l'analyse du portefeuille applicatif s'avère une tâche faisable qui nécessite l'intervention des développeurs et des informaticiens afin de cartographier les applications développées ainsi que les technologies utilisées.

Cette étape reprend la cartographie applicative réalisée dans la phase de l'étude de besoin et permet d'identifier les ressources applicatives qui vont être exploitées pour implémenter les services d'entreprise. En outre, elle permet de déterminer les verrous présents sur l'existant applicatif et rendant l'évolution du système d'information difficile.

- **Étape 2 : Identification des services d'accès au système legacy**

En étudiant la cartographie applicative, nous pouvons constater que certaines ressources doivent être conservées et ne nécessitent pas une restructuration en mode services. Ceci peut être expliqué soit par le fait qu'il n'est pas économiquement envisageable de les refondre complètement, soit par le fait qu'elles donnent globalement satisfaction à leurs utilisateurs en termes de fonctionnalités. Par conséquent, il s'agit de construire des façades homogènes au-dessus de ces ressources. Ces façades ne sont autres que les services d'accès au système *legacy*. Dans la littérature, il existe une multitude de méthodes et d'outils relativement matures qui permettent de définir des services d'accès au système *legacy*. Des exemples de ces méthodes et outils qui peuvent être utilisés dans le cadre de l'identification des services d'accès au système *legacy* sont : SAP NetWeaver [SAP, 2006], Apache Beehive [Apache, 2005], IBM WebSphere [IBM, 2007a], *etc.*

- **Étape 3 : Identification des services entités**

Comme il a été déjà souligné, les services entités sont des services qui se focalisent sur les objets métier clés du système d'information de l'entreprise. Il s'agit essentiellement des services qui permettent l'accès aux informations relatives aux objets métier. D'après leur définition, les services entités sont fortement reliés au modèle d'objets métier de l'entreprise. Afin de retrouver le modèle d'objets métier, nous proposons de modéliser le code des applications déjà cartographiées. Cette étape relève plutôt du génie logiciel et notamment de l'ingénierie inversée (*reverse engineering*) et profite des outils développés et des avancées

réalisées dans le cadre de la réutilisation du code ou encore de l'optimisation et de la maintenance du patrimoine applicatif de l'entreprise.

Nous envisageons d'utiliser des outils qui permettent d'extraire le code des applications existantes et de le modéliser sous formes de classes UML. Des exemples de tels outils existent dans la littérature et nous pouvons citer par exemple *Rational Rose reverse engineering tool* [IBM, 2007b], Borland Together [Together, 2008], *PyUT Python UML Tool* [PyUT, 2007] , *etc.*

Le résultat de cette étape consiste à un ensemble de diagrammes de classes relatifs aux applications d'entreprise. Ces diagrammes décrivent les classes implémentées par les applications ainsi que leurs relations.

À partir des diagrammes de classes nous proposons une méthode d'identification des services entités avec le niveau de granularité adéquat. Cette méthode consiste à regrouper les classes en des services entités et respecte un ensemble de caractéristiques à savoir :

- **La caractéristique de consistance :** le service entité obtenu suite au regroupement des classes doit posséder un sens métier significatif. Certaines classes ont peu de sens utilisées seules comme, la classe *véhicule utilitaire* qui est un sous-type de la classe *véhicule*. Ainsi, le regroupement des classes doit se focaliser sur une classe centrale qui illustre le concept métier et les classes dérivées qui la décrivent.
- **La caractéristique d'autonomie :** les services entités ne doivent pas se chevaucher entre eux. Chaque classe doit appartenir exclusivement à un seul et unique service entité.
- **La caractéristique de la mono préoccupation :** chaque service entité doit contenir uniquement les classes qui décrivent son seul et unique concept métier.
- **La caractéristique de la stabilité :** chaque service entité doit avoir une durée de vie assez stable dans le temps. Il ne doit pas être dépendant d'un projet particulier.

Outre ces différentes caractéristiques, nous avons développé un ensemble de règles qui appuient les décisions des architectes IT en fournissant des directives sur la façon de regrouper les classes. Ces règles se basent sur la notion de relation entre les classes. La relation **R** entre une classe **A** et une classe **B** se définit formellement par $< \mathbf{A}, \mathbf{B} > \in \mathbf{R}$. On distingue deux types de relation : la relation statique et la relation dynamique.

- Une **relation statique** entre une classe **A** et une classe **B** peut être une relation d'héritage, de composition ou d'agrégation. Nous définissons pour chaque type de relation une règle spécifique afin de regrouper les classes qui y participent. Le Tableau 2.10 résume les relations statiques entre les classes et les règles de regroupement à appliquer.

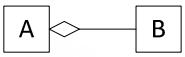
Relation	Description	Règle
	La classe A **agrège** la classe B	La classe B présente de **grande probabilité** pour rejoindre le même service entité que la classe A
	La classe A **est composé de** la classe B	La classe B présente de **grande probabilité** pour rejoindre le même service entité que la classe A
	La classe B est une **sous-classe** de la classe A	La classe B présente de **grande probabilité** pour rejoindre le même service entité que la classe A

Tableau 2.10. Les relations statiques et les règles de regroupement

- Une **relation dynamique** entre une classe **A** et une classe **B** se traduit par une association (au sens UML). Une association entre une classe **A** et une classe **B** signifie que les deux classes sont dépendantes (c'est-à-dire il existe un appel de méthode entre les classes). Afin d'évaluer le degré de dépendance entre les classes, nous classons les types d'appels de méthode comme suit :

- L'objet $\mathbf{A_a}$ de la classe **A** appelle une méthode qui crée l'objet $\mathbf{B_b}$ de la classe **B**,

- L'objet $\mathbf{A_a}$ de la classe **A** appelle une méthode qui supprime l'objet $\mathbf{B_b}$ de la classe **B**,

- L'objet $\mathbf{A_a}$ de la classe **A** appelle une méthode qui modifie l'objet $\mathbf{B_b}$ de la classe **B**,

- L'objet $\mathbf{A_a}$ de la classe **A** appelle une méthode qui lit des informations sur l'objet $\mathbf{B_b}$ de la classe **B**. En considérant ces types d'appels de méthode, nous définissons un ensemble de règles à appliquer à savoir :

- Règle 1 : les classes qui participent à une association dont le type d'appel de méthode est soit la création ou la suppression, doivent appartenir à un même service entité car il existe une forte dépendance entre elles.

- Règle 2 : les classes qui participent à une association dont le type d'appel de méthode est la lecture, peuvent être membres de différents services entités, car elles maintiennent de faible dépendance entre elles.

- Règle 3 : les classes qui participent à une association dont le type d'appel de méthode est la modification, peuvent appartenir à un même service entité.

Il est important de signaler qu'une fois les règles de regroupement appliquées, l'architecte IT doit valider le regroupement en fonction de son expérience dans le domaine ainsi que de son point de vue vis-à-vis de l'application.

2.4.4 Phase 4 : Accostage entre la SOA métier et la SOA IT

La dernière phase de CSOMA constitue la phase d'accostage, qui est une phase primordiale dans la démarche. Elle peut être considérée comme une phase de consolidation. Le dictionnaire de la langue française définit le verbe « consolider » comme le fait de « rendre plus ferme, plus stable et donner plus de solidité ». En effet, la phase d'accostage permet de

confronter les résultats des deux phases (phase métier et phase informatique) afin de valider les services qui ont été déjà identifiés.

De cette façon, la phase d'accostage soutient l'alignement stratégique du système informatique sur le métier de l'entreprise. En effet, les changements qui peuvent toucher les processus métiers suite à de nouvelles orientations stratégiques de l'entreprise seront répercutés sur le système informatique et plus précisément sur les services informatiques.

La phase d'accostage inclut trois étapes : la consolidation de services, la consolidation de données et la consolidation des modèles de schémas d'orchestration.

- **Étape 1 : Consolidation de services**

La consolidation de services permet de s'assurer que les services métier qui ont été identifiés dans la SOA métier peuvent être implémentés à l'aide des services informatiques retrouvés dans la SOA IT. Il s'agit essentiellement d'identifier les services informatiques à supprimer du système informatique de l'entreprise et/ou les services informatiques qu'il faut développer pour répondre à de nouveaux besoins métier.

- **Étape 2 : Consolidation des données**

La consolidation des données permet de valider les paramètres d'entrée et de sortie des services métier et ceci en considérant les données manipulées par les services entités. Nous considérons qu'il est important que les paramètres d'entrée et de sortie des services métier soient conformes à ceux des paramètres manipulés par les services entités retrouvés à partir de patrimoine applicatif de l'entreprise. L'objectif ultime de cette étape est de déboucher sur un modèle de données unifié.

La Figure 2.23 illustre un exemple de consolidation des données. Il s'agit de valider le paramètre d'entrée (*adresse*) d'une opération intitulée « Envoyer facture au client » et ceci en considérant les informations manipulées par le service entité intitulé « Facture ». En confrontant le paramètre d'entrée adresse issue de l'opération « Envoyer facture au client » avec le même paramètre issue du service « Facture » nous pouvons remarquer que chacun d'entre eux couvre un ensemble d'attributs. Ainsi, la consolidation des données permet d'obtenir le même modèle de donnée pour le paramètre *adresse*.

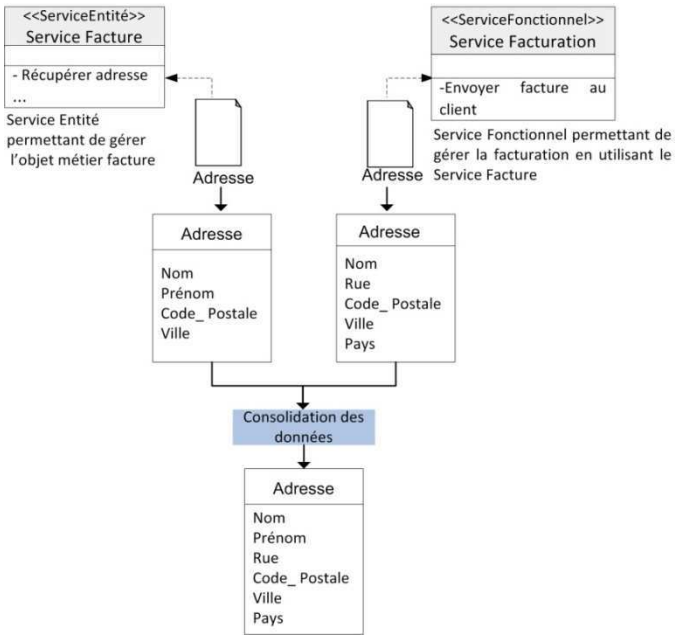


Figure 2.23. Exemple de consolidation des données

- **Étape 3 : Consolidation des Modèles de Schémas d'Orchestration**

La consolidation des Modèles de Schémas d'Orchestration consiste à revoir les schémas d'orchestration en tenant compte des deux consolidations précédentes. Chaque service fonctionnel appartenant au schéma d'orchestration du service domaine va être examiné afin de prendre en compte les modifications proposées par l'étape de la consolidation des services. De plus les paramètres d'entrée et de sortie doivent être mis à jour afin qu'ils soient conformes aux modifications proposées par la consolidation de données.

Ce type de consolidation vise à mettre à jour les services fonctionnels contenus dans le schéma d'orchestration sans pour autant changer la logique de l'orchestration.

2.5 Étude de cas et validation

Afin de tester la démarche CSOMA, nous avons considéré le cas d'étude de l'entreprise *Enterprise Training Solutions* (ETS). ETS est une PME spécialisée dans la planification et l'exécution des sessions de formation pour le personnel de ses clients appartenant au secteur public (organismes gouvernementaux, organismes publics) ainsi qu'au secteur privé. L'entreprise ETS est considérée comme un acteur majeur dans le domaine de formation en informatique. Elle propose des formations individuelles sur site ou par groupe qui peut être composé de 30 à 40 personnes.

Les activités de l'entreprise ont été marquées par une croissance majeure durant les deux dernières années. En effet, le nombre de clients ne cesse de croître accompagné par une augmentation au niveau du personnel de formation et du personnel de l'entreprise. Cependant, le système d'information de l'ETS se trouve incapable de suivre cette évolution et la majorité du travail s'effectue à travers des appels téléphoniques et des feuilles Excel. Afin d'y

remédier, la direction métier et la direction du système d'information proposent d'améliorer le système d'information de l'ETS tout en prenant en compte les considérations suivantes :

- **Des considérations techniques :** il s'agit d'éviter une refonte complète du système d'information et ceci en utilisant au maximum le patrimoine existant. Ce dernier englobe quelques applications utilisant des technologies diverses ainsi qu'une gestion de l'entrepôt de données et des flux qui sont relativement complexes.
- **Des considérations fonctionnelles :** face à d'importants enjeux de croissance et d'amélioration de la productivité, l'ETS souhaite que ses processus métiers soient plus flexibles afin de supporter les évolutions stratégiques de l'entreprise telle que la participation à des scénarios de coopération avec d'autres entreprises du domaine. De plus, l'ETS a pris conscience qu'Internet peut lui permettre de trouver les meilleurs fournisseurs et les clients les plus intéressants. Ainsi, l'entreprise souhaite avoir un site Web qui lui permet d'exposer les sessions de formation qu'elle propose et de permettre aussi à ses clients de commander en ligne des sessions de formation.

L'objectif de l'ETS est de mettre en place un nouveau système flexible répondant à une architecture conforme aux standards informatiques. La mission consiste donc à appliquer la démarche CSOMA afin de rénover le système d'information et de proposer une architecture flexible qui résout les problèmes de l'ETS. L'application de la démarche CSOMA pour cette étude de cas permet :

- de mieux appréhender son fonctionnement en pratique,
- une meilleure compréhension des différentes phases,
- de montrer l'implication des différents acteurs (*i.e.* analyste métier, analyste IT, développeur) dans la mise en place de la démarche.

Grâce aux données recueillies et aux cartographies élaborées (*i.e.*, cartographie métier et cartographie applicative), nous nous sommes efforcés de proposer un ensemble d'axes d'amélioration. L'usage de ces cartographies et l'identification des axes d'amélioration (architecture cible) constituent les bases de la démarche CSOMA. En effet, grâce aux cartographies métier, CSOMA permet à l'entreprise ETS d'identifier ses différents services métier à partir des différents processus métiers. Cette identification se base sur les buts métier et respecte les considérations fonctionnelles déjà décrites.

De la même manière, l'application de la démarche CSOMA, notamment la phase de construction de la SOA IT, permet de retrouver les différents services informatiques à partir des applications existantes.

Outre l'identification des services métiers, CSOMA offre également à l'entreprise ETS une modélisation détaillée des différents services fonctionnels qui peuvent être par la suite projetés dans une plateforme technologique. Elle permet également une étude approfondie des variantes d'usage et d'adaptabilité de ces services à travers la notion d'Aspects Conceptuels et des schémas d'orchestration attribués aux services domaines. Cette étude se base essentiellement sur les besoins exprimés par les différents analystes métier de l'entreprise.

Les différents services métier et informatiques retrouvés sont orchestrés par un ensemble de services domaines. Ces derniers respectent les diverses considérations énoncées par

l'entreprise ETS. Ils répondent également aux problèmes soulignés par la direction métier de l'entreprise et satisfont les différents objectifs et les plans stratégiques de l'entreprise.

Une description détaillée de l'application de la démarche CSOMA au sein de l'ETS est présentée en annexe A de ce manuscrit de thèse. Ce dernier comprend également quelques détails de réalisation de l'atelier de génie logiciel support à CSOMA.

2.6 Conclusion

La mise en place d'une SOA au sein de l'entreprise est un projet de longue haleine. Plusieurs entreprises industrielles ont déjà compris le besoin de migrer vers un système d'information basé sur les services. Cependant, une migration non accompagnée par une démarche rigoureuse entraînera la perte de la valeur ajoutée apporté par la SOA. Ce premier chapitre a apporté des réponses à cette problématique à travers la proposition d'une démarche de mise en place de la SOA appelée CSOMA (*Contextual Service Oriented Analysis and Design*).

Nous avons présenté dans ce chapitre notre approche CSOMA, une démarche hybride, qui répond au manque de méthodes académiques pour la construction de la SOA. La démarche CSOMA a été décrite grâce à un ensemble de méta-modèles dont l'objectif est de présenter ses concepts clés. Une attention particulière a été adressée aux deux concepts pivots : service métier et service IT. Ensuite, nous avons présenté la démarche à quatre phases qui guide les analystes métier et IT dans le processus d'identification et de modélisation des services. La première phase s'est focalisé sur l'élaboration du ou des modèles de motivation métier lié (s) à la mise en place de la SOA ainsi qu'à l'étude de l'architecture d'entreprise. Quant à la deuxième phase, elle a permis de modéliser et de raffiner les services métier retrouvés à partir d'une modélisation contextuelle des processus de l'entreprise. Au sein de cette phase, les services métier ont été reliés à un ensemble d'Aspect Conceptuels qui encapsulent les paramètres et les préoccupations qui gouvernent les comportements adaptables des services. La troisième phase a permis d'identifier certains services informatiques, pertinent pour notre travail, à partir du patrimoine applicatif de l'entreprise. Enfin, la quatrième phase a présenté une confrontation des résultats des deux précédentes phases afin de valider les services déjà identifiés. En procédant de la sorte, cette phase soutient l'alignement stratégique du système informatique sur le métier de l'entreprise.

Par rapport aux méthodes présentées dans l'état de l'art, CSOMA est une démarche indépendante des technologies de réalisation (et notamment de la technologie service Web) et s'inscrit dans un cadre d'architecture dirigée par les modèles (MDA). CSOMA a donné naissance à un service dédié à la coopération appelé service domaine. Ce service présente l'avantage d'être adaptable. En effet, CSOMA a intégré l'adaptation des services lors de leur modélisation à travers les Aspects Conceptuels ainsi qu'à travers la proposition de l'ensemble des schémas d'orchestration relatifs aux services domaines. Cette caractéristique est importante et permet d'assurer la flexibilité de la méthode.

Un atelier de génie logiciel a été implémenté pour supporter la démarche CSOMA. Cet atelier a servi de base pour le test de la démarche CSOMA dans le cadre d'une entreprise de service. Le cas industriel et l'application de la démarche sont décrits en détail dans l'annexe A.

CHAPITRE 3

Construction du processus coopératif à la demande

Table des matières

_Toc247269502

Le précédent chapitre de cette thèse a permis de répondre à la première sous problématique de construction de l'architecture de services au sein de l'entreprise. Il a permis notamment de décrire la démarche CSOMA (Context-aware Service Oriented Modelling and Analysis) dont l'objectif est double : assurer d'une part la flexibilité du système d'information de l'entreprise grâce à l'orientation service et favoriser d'autre part les coopérations à la demande. En effet, CSOMA donne naissance à un ensemble de services domaines qui peuvent être invoqués par des partenaires externes et contribuent par la suite à la réalisation de la coopération à base de composition des services. Nous allons nous intéresser dans ce chapitre à la problématique de construction du processus coopératif à la demande.

L'objectif de ce chapitre consiste à répondre à certaines questions et limites énoncées dans le chapitre de l'état de l'art (cf. section 1.1.5). Par exemple, les mécanismes de coordination transactionnelle (Workflows) traitent de la coopération statique et ne supportent pas la recherche de partenaires ainsi que la réalisation des coopérations de manière dynamique [Casati and Discenza, 2000], [Angelov and Grefen, 2002], [Bitcheva, 2003]. C'est la raison

pour laquelle nous considérons que ce domaine, insuffisamment exploré, pose des problèmes spécifiques et mérite une exploration plus approfondie.

Nous proposons dans ce chapitre un cadre pour la mise en œuvre de la coopération en utilisant le paradigme de composition des services. Comme le domaine de composition des services est assez vaste, nous nous limitons dans notre étude à trois aspects fondamentaux qui constituent les principales contributions de ce chapitre : description, découverte et composition des services.

Ce chapitre est organisé de la manière suivante. La section 3.1 exposera les principes fondateurs permettant de guider la construction du processus coopératif à la demande. La section 3.2 présentera un ensemble de méta-modèles supports à la coopération. Ensuite, la section 3.3 proposera une vue globale de l'architecture de coopération à la demande. Enfin, la section 3.4 présentera la démarche de construction du processus coopératif.

3.1 Principes fondateurs

De même que pour la construction de l'architecture de services au sein de l'entreprise (cf. chapitre 2, section 2.1), la construction du processus coopératif à la demande doit répondre à plusieurs principes qui portent sur la spécificité de ce type de coopération.

Étant donnée la nature de la coopération à la demande, le premier principe qui s'impose, concerne le dynamisme et la durée de vie de la coopération. Le dynamisme vise à répondre rapidement à une opportunité d'affaire. Comme nous l'avons déjà souligné auparavant, la coopération à la demande est une coopération occasionnelle, sans contraintes structurelles, dans laquelle les partenaires impliqués et leur nombre ne sont pas pré-définis. En ce qui concerne la durée de vie, l'objectif est de réduire les temps de mise en place du processus coopératif par composition des services tout en améliorant la qualité du service composite obtenu. Ces caractéristiques (dynamisme et durée de vie) exigent des mécanismes pour prendre connaissance de l'existence de partenaires, les sélectionner, les localiser et faciliter la construction du processus coopératif.

Le deuxième principe à considérer pour la coopération à la demande est le principe d'ouverture. En effet, l'ouverture, notamment dans un contexte Internet, a pour conséquence immédiate une multiplication du nombre de partenaires potentiels. Par conséquent, le besoin est fort d'une entité qui gère d'une part, les différents services proposés par les partenaires et d'autre part, qui permet d'instaurer le niveau de sécurité requis par les entreprises. Bien qu'il soit important que chaque entreprise puisse communiquer et déclarer ce qu'elle peut offrir en matière de services, ceci ne nie pas le fait qu'elles peuvent être concurrentes. Par la suite, la présence d'une entité tierce qui assure ces fonctionnalités est d'une importance capitale.

Le dernier principe qui régit la coopération à la demande, telle que nous la considérons, est le besoin accru en termes de flexibilité. Comme nous l'avons déjà souligné, la flexibilité dans le cadre de notre travail peut être assurée grâce à l'adaptation au contexte. Dans le cadre de ce chapitre nous nous focalisons principalement sur la prise en compte de l'adaptation au contexte lors des étapes de description, de publication et de découverte des services. En effet, dans un univers ouvert, où le nombre de partenaires est important, leur découverte exige une

attention particulière qui ne doit pas se limiter aux descriptions fonctionnelles des services qu'ils proposent. La découverte peut intégrer une forme d'adaptation en filtrant dans la collection de services domaines disponibles, ceux correspondant au mieux, tant aux besoins fonctionnels du client, qu'aux besoins en termes d'adaptation au contexte d'utilisation (profil de client, domaine de la coopération, temps et lieu de la coopération, *etc.*). Ceci implique d'associer à la requête classique du client le besoin d'adaptation.

Ces différents principes déjà énumérés constituent une base fondamentale pour la construction de l'architecture de la coopération à la demande.

3.2 Méta-modèles de la coopération interentreprises à la demande

Dans un environnement ouvert et multipartenaire, un méta-modèle permet d'une part, de comprendre les concepts existants dans une coopération dynamique et d'autre part, de proposer une sémantique commune afin que tous les partenaires puissent avoir une vue cohérente sur les concepts manipulés. Le méta-modèle de la coopération à la demande, présenté dans la Figure 3.1, se base sur l'hypothèse préalable que pour coopérer, les partenaires définissent et rendent publique la description de leurs services domaines. On distingue dans le méta-modèle trois sous méta-modèles de base : le méta-modèle organisationnel, le méta-modèle du processus coopératif et le méta-modèle de la communauté de services. Ces dimensions permettent d'appréhender la mise en place de la coopération à la demande comme la composition des différents services domaines au sein de ce que l'on nomme un processus coopératif.

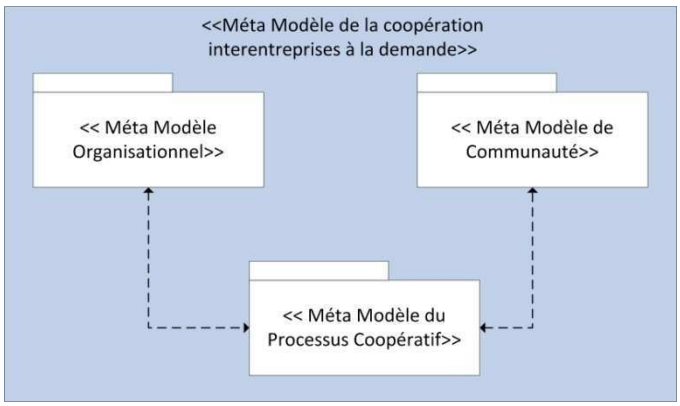


Figure 3.1. Méta-modèle de la coopération à la demande

Nous présentons dans les sections suivantes une description détaillée de ces méta-modèles indispensables à la mise en place d'une coopération interentreprises à la demande.

3.2.1 Méta modèle organisationnel de la coopération à la demande

Le méta-modèle organisationnel de la coopération à la demande est indépendant des modèles des entreprises partenaires dans une opportunité de coopération. En effet, les partenaires peuvent changer leurs organisations internes (suite à un regroupement ou à une séparation d'entreprises par exemple), mais ceci ne devra en aucun cas affecter leurs engagements

contractuels vis-à-vis du projet de coopération. Ce choix implique que chaque partenaire devra être apte à établir une correspondance entre ses structures internes et les rôles qu'il effectue au sein des processus coopératifs dans lesquels il est impliqué. Cette correspondance permet notamment de ne pas limiter l'évolution des modèles internes des partenaires, et ceci indépendamment à leurs participations à des projets de coopération. Techniquement parlant, une entreprise partenaire peut être amenée à modifier ses processus internes et/ou ses méthodes de travail sans pour autant affecter le fonctionnement global des différents processus coopératifs auxquels elle participe.

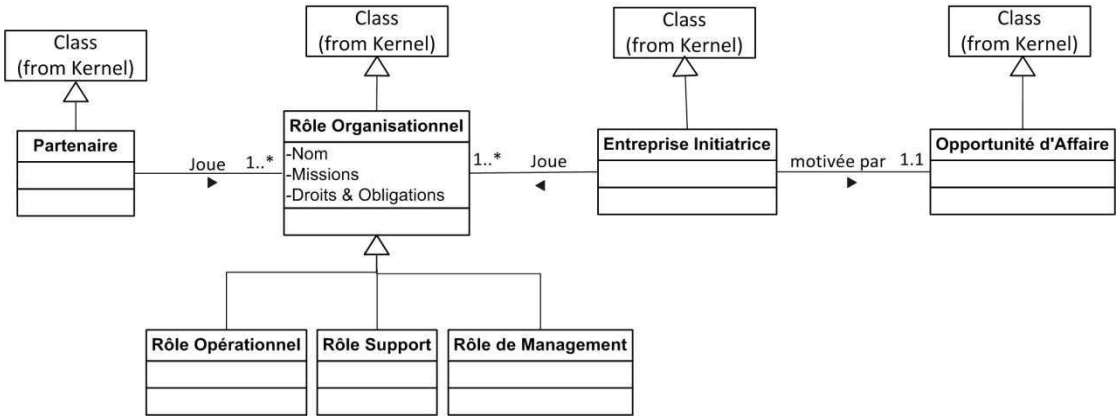


Figure 3.2. Méta-modèle organisationnel de la coopération à la demande

La Figure 3.2. montre le méta-modèle organisationnel que nous proposons. Le concept clé de ce méta-modèle est celui de l'entreprise initiatrice du projet de coopération. Cette dernière détecte une opportunité d'affaire sur le marché et initialise, comme son nom l'indique, un projet de coopération regroupant un ensemble de partenaires. L'entreprise initiatrice du projet de coopération incarne un rôle organisationnel. Dans la littérature, un rôle organisationnel est défini comme étant une abstraction du comportement d'une unité organisationnelle ou d'une entité [Ferraiolo and Kuhn, 1992], [Bitcheva, 2003], [Zaidat, 2005]. Il décrit les macro-compétences et les caractéristiques que doit acquérir une unité organisationnelle ou une entité afin de réaliser un ensemble de missions liées à un contexte donné.

Nous nous sommes basés sur cette définition afin de décrire le concept du rôle organisationnel. Nous attribuons au concept de rôle un ensemble d'attributs à savoir : nom, missions, droits et obligations. Le *nom* précise le nom du rôle. Les missions révèlent les activités à réaliser au cours de l'accomplissement du rôle. Les droits et obligations caractérisent les droits et les obligations associées à l'entreprise initiatrice du projet de coopération lors de la réalisation des activités reliées à son rôle.

Les travaux menés dans le cadre des coopérations interentreprises statiques ont mis l'emphase sur l'importance du concept de rôle organisationnel. La plupart des projets étudiés dans [Katzy and LÄoh, 2003] ont identifié des rôles de management nécessaires à la création et à la gestion des coopérations interentreprises. Plusieurs auteurs ont même insisté sur la nécessité d'étudier et de formaliser de tels rôles [Katzy and LÄoh, 2003] et [Lupu et al., 1999].

En nous inscrivant dans cette optique, nous allons dans la suite proposer une typologie de rôles organisationnels qui tient compte des activités supportées par une coopération interentreprises à la demande. Ainsi, une entreprise initiatrice du projet de coopération peut réaliser trois types d'activités :

- Les activités qui contribuent d'une manière directe à la réalisation de l'opportunité d'affaire (activités liées à l'expression du besoin de coopération, aux choix des services domaines, *etc.*).
- Les activités support au fonctionnement de la coopération (supervision de l'exécution des services domaines appartenant au processus coopératif, *etc.*) et
- Les activités de management (activités de gestion et de décision).

Cette distinction entre les trois types d'activité a induit à une spécialisation du concept de rôle organisationnel en trois types (voir Figure 3.1) :

- **Rôle Opérationnel :** décrit les macro-compétences et caractéristiques qui sont nécessaires à la réalisation d'un ensemble de missions qui participent d'une manière directe dans la réalisation de l'opportunité d'affaire.
- **Rôle Support :** décrit les macro-compétences et caractéristiques qui sont nécessaires à la réalisation d'un ensemble de mission de support.
- **Rôle de Management :** décrit les macro-compétences et caractéristiques qui sont nécessaires à la réalisation d'un ensemble de missions de gestion de la coopération. Des exemples de missions incluent la création, le suivi, et la dissolution de la coopération.

Dans la suite de ce document, nous utilisons le terme client pour désigner l'entreprise initiatrice du projet de coopération.

3.2.2 Méta-modèle de la communauté de services domaines

Un autre concept aussi important que les concepts relatifs à l'organisation de la coopération est celui de la communauté. La notion de communauté, telle qu'elle sera présentée dans cette section, souligne une vue technique du concept de communauté d'entreprises déjà présenté dans l'état de l'art relatif à la coopération (cf. chapitre 1, section 1.1.3.3).

Le dictionnaire Larousse définit le concept de communauté comme un groupe de personnes qui vivent ensemble et/ou sont unies par des intérêts communs, une religion, une nationalité, etc. Appliquée aux domaines des services Web, [Benatallah et al., 2003b] définissent une communauté comme un ensemble de services Web qui possèdent une même fonctionnalité même si ces services ont des propriétés non fonctionnelles distinctes.

Quant à [Medjahed and Bouguettaya, 2005], ils considèrent une communauté comme un moyen d'offrir une ontologie des services Web ayant le même domaine d'intérêt. Enfin, [Maamar et al., 2007b] définissent une communauté par la fonctionnalité d'un service Web représentatif de cette communauté (*i.e.*, sans faire référence explicitement à quelconque service Web concret dans la communauté).

D'autres auteurs comme [Duarte-Amaya, 2007] considèrent la communauté comme un cadre à la recherche et à la sélection dynamique de services Web et qui permet ainsi de remédier aux déficiences des annuaires UDDI [Benatallah et al., 2005b], [Duarte-Amaya, 2007]. C'est aussi un moyen pour supporter la composition dynamique de services [Fauvet et al., 2005] ou encore de permettre la substitution (à la conception ou à l'exécution) de services [Taher et al., 2006].

Dans le cadre de notre travail, nous considérons une communauté comme un espace regroupant un ensemble de services domaines qui proposent les mêmes fonctionnalités métier. Ces services se distinguent par leurs paramètres contextuels ainsi que les sous processus métiers qu'ils encapsulent et qui désignent les savoirs faire des entreprises qui les offrent. Par exemple, une communauté de transport peut rassembler les différents services domaines proposés par les différentes entreprises de transport. Le méta-modèle que nous proposons pour la communauté des services domaines est illustré dans la Figure 3.3.

Une communauté est initiée par un **fournisseur de communauté** qui peut être par exemple un consortium particulier, un groupe d'organisations qui opère sur une place de marché ou tout simplement un administrateur. Une communauté est identifiée par un ensemble d'attributs : *Identifiant*, *Catégorie* et *ListeMembres.*

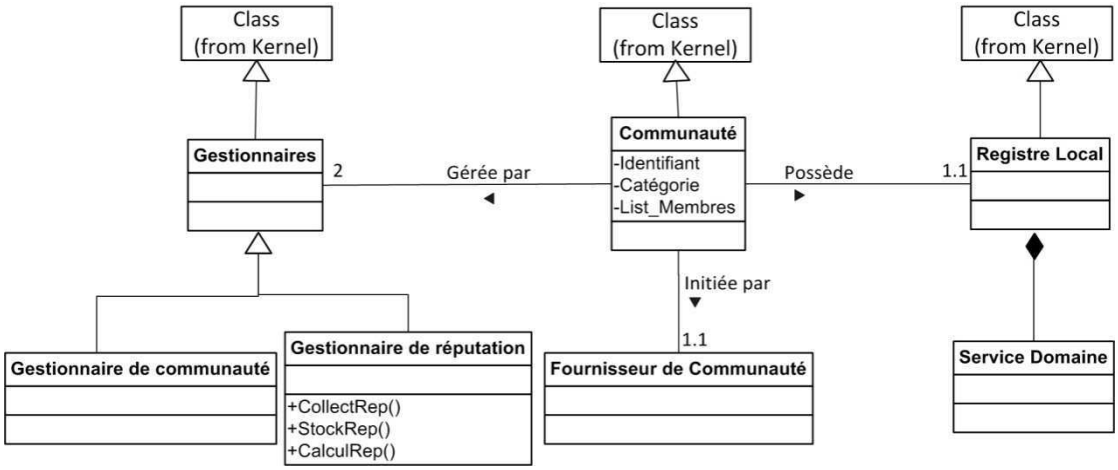


Figure 3.3. Méta-Modèle de la communauté des services domaines

Initialement, le fournisseur définit la catégorie de la communauté (par exemple, « communauté de transport »). Ensuite, il déploie le **gestionnaire de la communauté** qui dirigera la communauté, ce qui veut dire entre autres, inviter des services domaines à s'inscrire dans sa communauté. Chaque communauté détient un registre de services qui stocke l'ensemble des services inscrits. Le désassemblage d'une communauté est mené par le fournisseur et se produit à la demande du gestionnaire de la communauté. Ce dernier surveille tous les événements qui se déroulent dans sa communauté, comme par exemple l'arrivée de nouveaux services domaines et le départ de certains services.

Attirer de nouveaux services domaines et retenir ceux déjà existants dans la communauté relèvent des responsabilités du gestionnaire de la communauté. Ce dernier consulte régulièrement le registre de services de la coopération à la recherche de nouveaux services domaines. Lorsqu'un service domaine candidat est identifié par sa fonctionnalité, le **gestionnaire de la communauté** interagit avec son fournisseur. Par ailleurs, retenir les services domaines dans la communauté pour une longue période de temps est un bon indicateur. En effet, d'une part, les services domaines affichent une attitude coopérative bien qu'ils soient en compétition. D'autre part, les fournisseurs de services domaines sont dans une certaine mesure satisfaits de leur taux de participation dans les processus coopératifs.

Le gestionnaire de la communauté travaille en concert avec un **gestionnaire de réputation** qui se charge de maintenir la réputation des différents services domaines appartenant à la communauté. D'une façon générale, la réputation est définie comme étant l'opinion (plus techniquement, une évaluation sociale) publique envers une personne, un groupe de personnes, ou une organisation. C'est un facteur important surtout dans le cadre d'une coopération à la demande où les entreprises ont besoin d'un certain niveau de confiance afin d'interagir avec les services des partenaires. Dans ce cas de figure, la communauté joue le rôle d'un garant pour ses clients. En effet, la réputation de service sera utilisée par la suite pour répondre aux besoins des clients. Ces derniers peuvent exprimer certaines exigences en termes de réputation du service avec lequel ils vont interagir.

Le gestionnaire de réputation calcule et met à jour la réputation des différents services domaines appartenant à la communauté. Pour ce faire, il se charge de réaliser trois fonctionnalités : la collecte des réputations, leur stockage ainsi que le calcul des scores de réputation. Ces trois fonctionnalités sont modélisées sous formes d'opération au sens UML dans le méta-modèle de la Figure 3.3 (*CollectRep, StockRep, CalculScore*). Nous détaillons ces trois fonctionnalités dans ce qui suit.

- **La collecte des réputations**

Nous supposons que les clients attribuent une note (ou un avis) indiquant le niveau de satisfaction vis-à-vis du service et ceci après chaque interaction avec ce dernier. Cette note est un entier allant de 1 à 10, où « 10 » signifie une extrême satisfaction, « 5 » désigne une satisfaction moyenne et « 1 » signifie l'extrême mécontentement.

La note proposée par les clients exige que ces derniers effectuent une évaluation objective. Nous ne nous sommes pas intéressés dans le cadre de notre travail de recherche à l'étude de l'objectivité de l'évaluation des clients. Nous considérons que les notes attribuées sont à la fois valides et disponibles.

- **Le stockage des réputations**

Le gestionnaire de réputation se base sur le système de stockage appelé *SuperstringRep*. Ce système est un protocole proposé par [Wishart et al., 2005]. Les différentes notes attribuées aux services sont stockées dans une base de données locale maintenue par le gestionnaire de réputation. Cette base contient le classement des différents services. Chaque classement regroupe l'identifiant du service (Service ID), l'identifiant du client (Client ID), la note attribuée (Note) et un *timestamp*. Ce dernier est une séquence de caractère qui contient le

temps en termes de date et d'heure, pour situer l'évaluation du client dans le temps. L'identifiant du service est celui qu'on lui attribue dans le registre de service alors que l'identifiant du client désigne l'adresse IP du client. Le tableau suivant illustre un exemple de classement des services. Comme nous le pouvons remarquer, il existe deux occurrences du même service ID « 2226cb6e-e8c9-4fe3-9ea8-3c97b1kk8bf4 ». Chacune de ces occurrences comporte une note attribuée par un client particulier. Le *timestamp* peut être utilisé afin de déterminer laquelle des notes est la plus récente.

Service ID	Client ID	Note	Timestamp
2226cb6e-e8c9-4fe3-9ea8-3c97b1kk8bf4	25.16. 7.20	7	2009-06-01 09:15 :10
8841cb6e-e8c9-4fe3-9ea8-3c99b1fa8bf4	27.17. 19.27	6	2009-06-06 19:20 :01
9961cb6e-e8c9-4fe3-9ea8-3c99b1fa8bf4	7.22. 33.111	5	2009-06-08 12:15 :02
2226cb6e-e8c9-4fe3-9ea8-3c97b1kk8bf4	34.0. 9.27	6	2009-07-01 09:25 :10
7421ab5e-j2j7-4fe3-9zk8-4c93d1ja8bf4	112.28. 1.2	8	2009-07-04 13:20 :10

Tableau 3.1. Exemple de classement des services basé sur *SuperstringRep* [Wishart et al., 2005]

- **Le calcul des scores de réputation**

Le calcul du score de réputation que nous proposons étend les travaux de [Majithia et al., 2004] et [Wishart et al., 2005]. Dans le travail de [Majithia et al., 2004], les auteurs présentent une méthode pour calculer le score de réputation comme la somme pondérée des notes pour un service. Quant à [Wishart et al., 2005], ils proposent une fonction appelée *aging factor* qui applique un facteur à chacune des notes proposées pour un service. Dans notre travail, le score de réputation (R) est calculé comme la moyenne pondérée de toutes les notes proposées par plusieurs clients pour un service donné, avec un poids attaché à chacune des notes. Une faible valeur du poids signifie que seules les dernières notes sont incluses. Tandis qu'une grande valeur du poids implique qu'un plus grand nombre de notes sont incluses dans le calcul du score de réputation d'un service.

Formellement, le score de réputation (R) est décrit comme suit :

$$R = \frac{\sum_{i=1}^{N} N_i \gamma_i}{\sum_{i=1}^{N} \gamma_i}$$

$$\gamma_i = \lambda^{d_i}$$

Où R est le score de réputation,

N_i est la ième note attribuée à un service,

γ_i est le facteur de vieillissement (*aging factor*) de la ième note attribuée à un service,

λ est le poids (*weight*) avec $0<\lambda<1$,

d_i est le nombre de jours séparant deux temps t_c et t_i (le temps courant et le temps

correspondant à la ième note attribuée au service)

Il est important de souligner que les différents concepts déjà décrits dans le méta-modèle relatif à la communauté vont être réutilisés dans la démarche de construction du processus coopératif.

3.2.3 Méta-modèles du processus coopératif à la demande

La troisième composante de la coopération à la demande est celle du processus coopératif. Nous définissons le méta-modèle du processus coopératif en se basant sur le cycle de vie de ce dernier. Comme nous l'avons déjà souligné, le processus coopératif à la demande est perçu comme une composition des services domaines. Le concept de **service domaine** a déjà été étudié en détail dans le précédent chapitre et constitue ainsi la brique de base du processus coopératif (voir figure suivante).

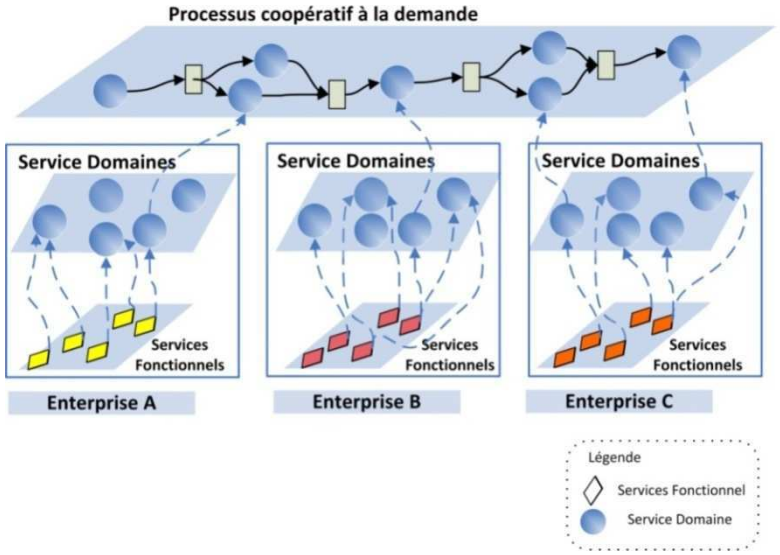


Figure 3.4. Essence du service domaine

Ainsi, par analogie avec le cycle de vie de la composition des services Web proposé dans [Benatallah et al., 2005a], nous considérons le cycle de vie d'un processus coopératif comme la réalisation de trois activités :

- **Configuration du schéma abstrait de la composition**

Cette activité permet de spécifier l'ensemble des Goals Templates ainsi que les flux qui les relient. En effet, les Goals Templates permettent d'exprimer l'objectif poursuivi par le projet de coopération. Il s'agit d'un premier niveau de la composition des services d'une manière abstraite (*i.e.*, sans faire référence à des services concrets). Cette activité est déclenchée par une requête d'une entreprise initiatrice du projet de coopération. Nous revenons en détail dans la suite de cette section sur ces différents concepts.

- **Découverte des services**

Cette activité consiste à découvrir les services domaines qui répondent aux différents Goals Templates. Nous allons présenter dans la suite de ce chapitre le processus de découverte des services que nous proposons. Ce processus propose des mécanismes pour la découverte des services qui correspondent aux besoins fonctionnels ainsi que contextuels de l'entreprise initiatrice du projet de coopération.

- **Composition semi-automatique des services**

Cette activité consiste à l'exécution des spécifications de la composition précédemment définies. C'est à la charge de la plateforme de la coopération d'exécuter le processus coopératif et ceci en utilisant un moteur d'orchestration.

En suivant ce cycle de vie, nous proposons de décrire le processus coopératif grâce à deux méta-modèles distincts qui s'étalent sur deux niveaux d'abstraction différents. Le premier, appelé méta-modèle du processus coopératif abstrait, concerne la phase de création du processus coopératif. Un processus coopératif abstrait représente un processus dont les flux de données et de contrôle sont définis, mais les services concrets qui supporteront le processus ne sont pas encore fixés. Le deuxième, appelé méta-modèle du processus coopératif exécutable, s'intéresse à l'exécution du processus coopératif. Chacun de ces méta-modèles propose ses propres concepts et relations.

3.2.3.1 Méta-modèle du processus coopératif abstrait

Le méta-modèle du processus coopératif abstrait que nous proposons est illustré en Figure 3.5. Il décrit les éléments de modélisation qui constituent un modèle de processus coopératif lors de la phase de création. Cette phase correspond à l'étape où l'entreprise initiatrice de projet de coopération commence par configurer manuellement le **schéma abstrait** du processus. Le schéma abstrait comportera un ensemble de **Goals Templates** indispensables pour l'expression du besoin de coopération ainsi qu'un ensemble de **flux de séquence** et **des branchements** qui les connectent.

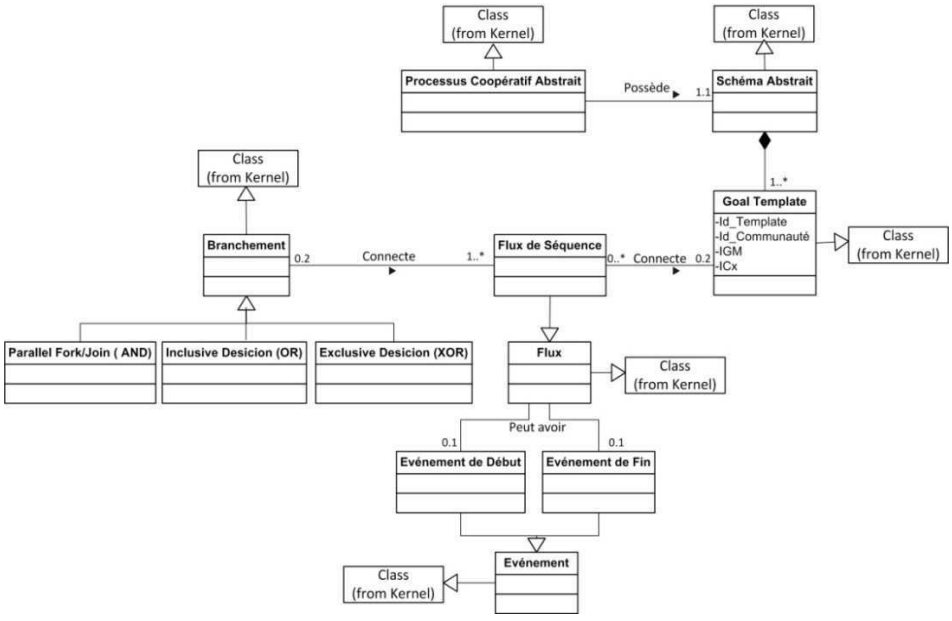


Figure 3.5. Méta-modèle du processus coopératif abstrait

Nous allons présenter ces différents concepts dans la suite de cette section.

- **Le concept de schéma abstrait de processus coopératif**

La création du processus coopératif se base essentiellement sur la sélection d'un ou de plusieurs Goal Templates et de préciser ensuite les flux de séquence qui les relient [Boukadi et al., 2008a], [Chaari et al., 2008]. La composition des différents Goal Templates constitue un schéma abstrait. Nous avons choisi d'exprimer le schéma abstrait dans un langage orienté utilisateur et ceci en proposant une notation visuelle. L'aspect visuel permet de faciliter la tâche pour l'entreprise initiatrice du projet dans le sens où elle ne sera pas amenée à connaître les détails techniques des différents services des partenaires, ni à se soucier de leurs interconnexions.

En se basant sur le schéma abstrait, les services d'entreprises vont être sélectionnés et le schéma d'orchestration exécutable va être généré. Formellement, un schéma abstrait (SA) est défini de la manière suivante :
SA = < GTs, B, L > où :

- *GTs* est l'ensemble des Goal Templates,
- B est l'ensemble de branchements,
- $L \in (GTs \cup B) \times (GTs \cup B)$ est l'ensemble des flux de séquence qui décrivent les relations entre les Goal Templates,

- **Le concept de Goal Template**

Les Goals Templates correspondent aux tâches qui doivent être accomplies par les entreprises partenaires dans le but de répondre aux objectifs du processus coopératif. Une entreprise initiatrice du projet de coopération à la recherche d'un service particulier exprime les conditions que le service à découvrir doit remplir. Nous appelons l'ensemble de ces conditions un Goal Template. Définir un Goal Template revient à exprimer une requête pour récupérer les services enregistrés dans le registre de la coopération. Un Goal Template se présente donc de façon similaire à une capabilité, spécifiant ce que l'on souhaite réaliser en sortie et ce que l'on peut fournir en entrée.

Ils jouent le rôle d'un patron qui va être utilisé pour identifier les services candidats et qui vont être par la suite impliqués dans le processus coopératif. Un Goal Template représente les exigences du client vis-à-vis du service à sélectionner. Formellement, un Goal Template est défini comme :
GT= <*Id_Template, Id_Communauté, IGM, ICx*> où :

- *Id_Template* est l'identifiant du Goal Template,
- *Id_Communauté* est l'identifiant de la communauté sélectionnée par le client,
- *IGM* (*Instanciated Generic Method*) représentent les méthodes génériques instanciées par le client. Ces dernières révèlent les caractéristiques fonctionnelles représentées au niveau du service grâce aux opérations,

- ICx (*Instanciated Context*) caractérise l'ensemble de paramètres de contexte qui définissent le Goal Template. Ils correspondent aux paramètres contextuels du service à chercher.

- **Le concept de flux de séquence**

Le concept flux de séquence s'inspire fortement de la définition formelle proposée pour ce concept dans le cadre du langage de modélisation BPMN (*sequence flow*). Le flux de séquence est utilisé pour représenter l'ordre des Goals Templates dans le schéma abstrait du processus coopératif et par suite l'ordre des services dans le processus exécutable. Ce flux n'a qu'une seule origine et qu'une seule destination. Il peut connecter au maximum deux Goals Templates. Cette restriction se traduit dans le méta-modèle (voir Figure 3.5.) par l'association qui les lie avec une cardinalité (0.2).

- **Le concept de branchements**

Les branchements servent à contrôler l'évolution des flux dans le processus coopératif (convergence, divergence, *etc.*). Il existe plusieurs sortes de branchements à sémantiques distinctes. Nous nous sommes inspirés du langage BPMN afin de proposer les branchements dans le cadre d'un processus coopératif. Nous avons classé ces branchements en trois catégories :

– *Exclusive Decision*/ *Merge* (*XOR*) : les flux sortants de ce branchement correspondent à diverses alternatives. Les tests conditionnant les branchements sont évalués dans l'ordre et dès que l'un d'eux est valable, un seul service concret sélectionné par le Goal Template et répondant à la condition peut être exécuté (exclusif).

– *Inclusive Decision*/ *Merge* (*OR*) : les flux sortant de ce branchement sont conditionnés par des tests. Cependant, le fait qu'un chemin soit validé n'exclue pas l'évaluation des autres.

– *Parallel Fork*/*Join* (*And*) : permet de créer et de synchroniser des flux parallèles reliant des Goal Templates.

Les symboles graphiques associés à un flux de séquence, des événements ainsi qu'aux branchements sont illustrés dans la figure suivante.

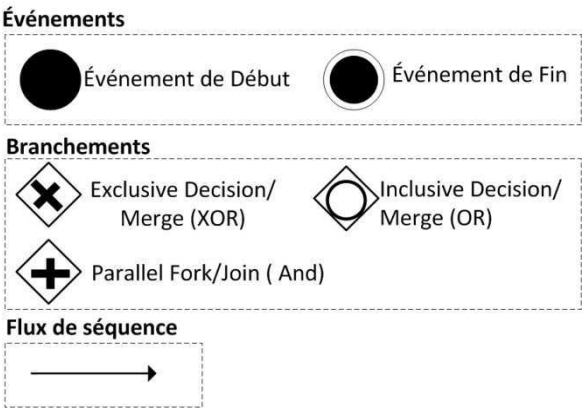


Figure 3.6. Flux de séquence, événements et branchements dans le cadre d'un processus coopératif abstrait

La Figure 3.7 présente un exemple de schéma abstrait construit à partir de trois Goal Templates reliés par un ensemble de flux de séquence et de deux branchements de type *Fork* et *Join*.

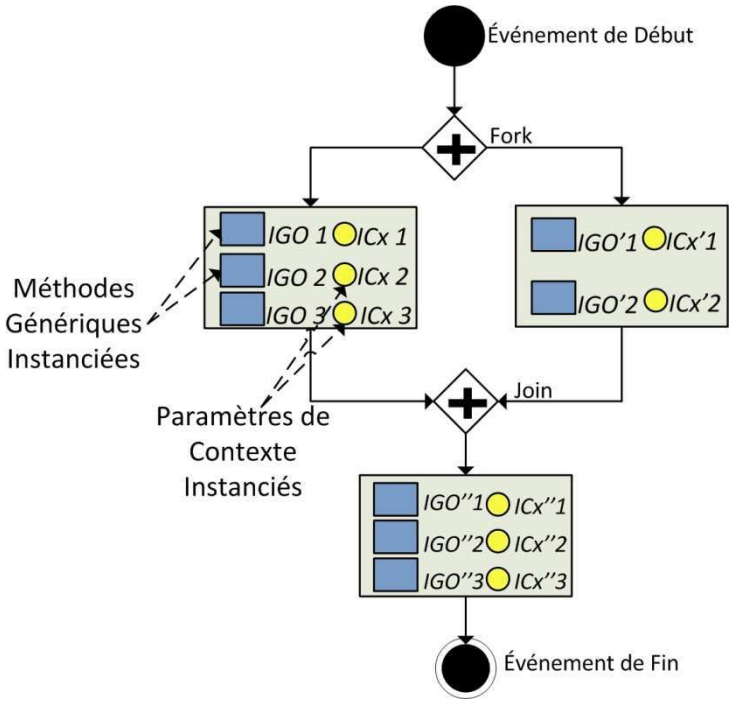


Figure 3.7. Exemple de schéma abstrait [Boukadi et al., 2008a]

3.2.3.2 Méta-Modèle du processus coopératif exécutable

Le processus exécutable constitue une instanciation d'un schéma abstrait construit par une entreprise initiatrice de projet de coopération. La transformation du processus coopératif abstrait vers un processus coopératif exécutable est analogue à une transformation du langage BPMN vers le langage BPEL. Ce type de transformation a été traité dans plusieurs travaux dans la littérature [Ouyang et al., 2006], [Indulska et al., 2007]. C'est la raison pour laquelle nous ne l'avons pas intégré dans nos travaux de recherche.

Le méta-modèle relatif au processus coopératif exécutable est montré dans la figure suivante et ceci en utilisant un jeu de deux couleurs (gris clair et gris foncé).

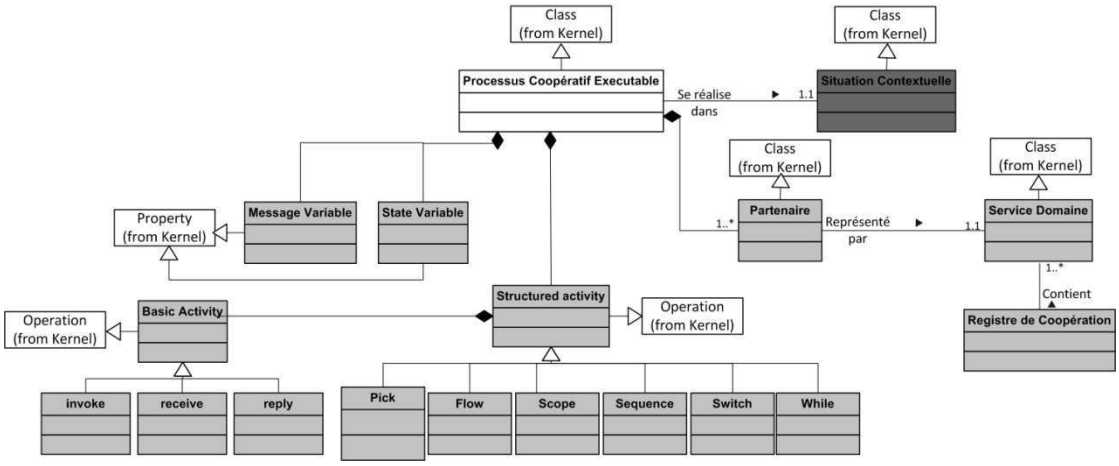


Figure 3.8. Méta-Modèle du processus coopératif exécutable

Nous allons revenir en détail sur les différents concepts retenus dans le cadre du méta-modèle de la Figure 3.8. à savoir : la situation contextuelle du processus coopératif (présenté en gris foncé) et la composition des services (présentée en gris clair).

3.2.3.2.1 Situation contextuelle du processus coopératif exécutable

Une particularité du processus coopératif exécutable est sa capacité à intégrer le contexte du client et des fournisseurs de services ainsi que celui de la coopération afin de changer et/ou d'altérer son comportement. Selon le méta-modèle de la Figure 3.8, cette spécificité est exprimée grâce au concept de situation contextuelle (représentée en gris foncé au niveau de la Figure 3.8.). Afin de bien cerner ce concept, il est nécessaire d'établir les fondations de la notion de contexte.

- **Notre vision du contexte**

Dans le domaine de la sensibilité au contexte, les auteurs n'ont pas encore proposé une définition à la fois générique et pragmatique de la notion de contexte. Nous avons montré dans le chapitre de l'état de l'art (cf. section 1.3.1) que la majorité des propositions de définition sont soit très abstraites soit très spécifiques à un domaine particulier [Brown, 1996], [Schilit and Theimer, 1994]. Malgré la variété des définitions, la formalisation de la notion de contexte reste difficile. La définition la plus répandue dans la littérature et la plus acceptée par les chercheurs est celle proposée par [Dey et al., 2001]. Cette dernière définit le contexte comme « toute information qui caractérise la situation d'une entité. Une entité étant une personne, un lieu ou un objet considéré comme pertinent relativement à une interaction entre un utilisateur et une application, incluant l'utilisateur et l'application eux-mêmes ». Cependant, cette définition trace un espace infini et illimité de ce qui fait partie du contexte. Le contexte comme nous l'avons déjà signalé a surtout été étudié dans le domaine des applications mobiles et de l'informatique diffuse.

Dans le cadre du processus coopératif plusieurs différences apparaissent pour la notion de contexte. En effet, dans le domaine des applications mobiles, le contexte assure le bon fonctionnement des services appartenant à ces applications. En revanche, dans le cadre d'une coopération interentreprises, le contexte est considéré comme un élément central pour la flexibilité des services et ne constitue pas un élément primordial pour leurs fonctionnements.

En outre, dans le domaine de l'informatique diffuse et notamment dans le domaine de l'interaction homme-machine, les paramètres de contexte sont transparents pour le client. Cependant, dans la coopération interentreprises, le contexte ne doit pas altérer ou changer complètement les fonctionnalités attendues par le client. Ce dernier doit être averti à l'avance des adaptations prévues.

En considérant les exigences imposées par l'utilisation du contexte dans la coopération interentreprises à la demande, nous proposons ci-après notre définition du contexte :

« Le contexte est l'ensemble des paramètres qui peuvent appartenir à l'environnement du processus coopératif (service, client et/ou coopération) et qui influencent son comportement en définissant de nouvelles vues sur les fonctionnalités proposées par les services participants. Ces paramètres peuvent être statiques ou dynamiques (c'est-à-dire qui changent durant l'exécution du processus) ».

Cette définition complète celle proposée par [Dey et al., 2001] en apportant plus de précision. En plus, elle appuie la définition de [Winograd, 2001] qui stipule que « la considération d'une information comme contexte est due à la manière dont elle est utilisée et non à ses propriétés inhérentes ».

- **Les catégories de contexte prises en compte dans le processus coopératif**

Étant donnée la diversité des informations composant le contexte, il est utile d'essayer de les classer par catégories pour faciliter leur utilisation dans le processus coopératif. Notre étude de la littérature (cf. chapitre 1 section 1.3.1) a révélé que plusieurs catégories de contexte ont été proposées et étudiées surtout dans le domaine de l'informatique pervasive ou ambiante où la notion de contexte prend beaucoup d'ampleur [Schilit and Theimer, 1994], [Winograd, 2001].

En se basant sur les différentes tentatives de catégorisation de contexte, nous proposons une ontologie illustrant les informations contextuelles qui peuvent être prises en compte dans le processus coopératif. L'ontologie de catégorisation de contexte propose un dictionnaire détaillé des paramètres contextuels pouvant influencer le comportement du processus coopératif. La figure suivante montre l'ontologie de catégorisation de la notion de « contexte » présentée dans [Boukadi et al., 2008b]. Cette dernière est une ontologie dynamique dans le sens où de nouvelles catégories de contexte peuvent être ajoutées.

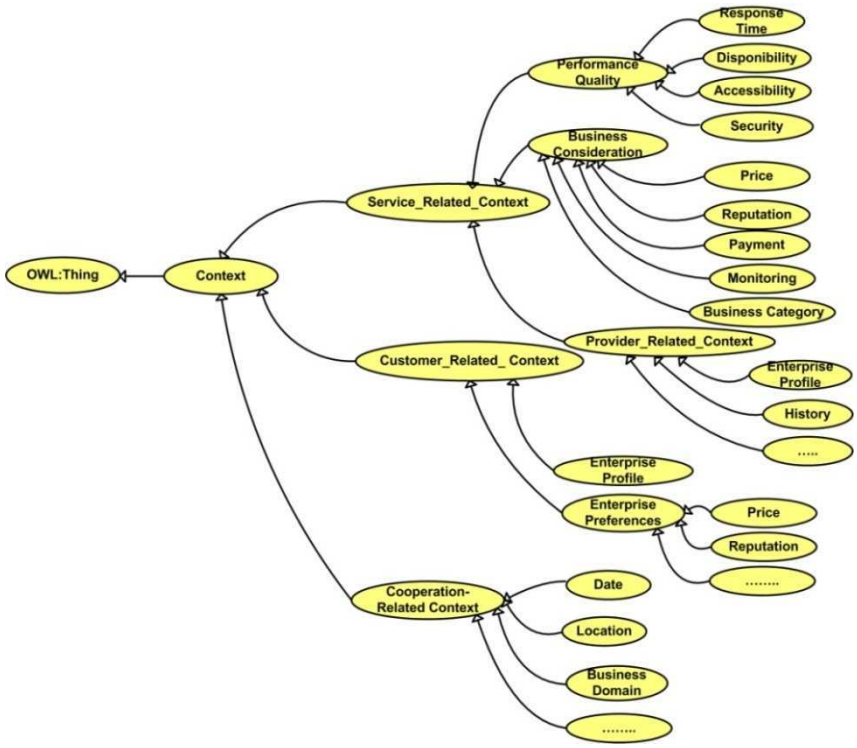


Figure 3.9. Ontologie de catégorisation de contexte [Boukadi et al., 2008b]

En considérant les entités participantes dans un processus coopératif, on identifie trois catégories de contexte : le contexte du service (et notamment de son fournisseur), le contexte du client du service et aussi le contexte de la coopération. Ces différentes catégories sont décrites en détail dans ce qui suit.

- **Contexte du service domaine (*Service_Related_Context*) :**

Cette catégorie de contexte porte sur les conditions selon lesquelles les fournisseurs peuvent offrir leurs services domaines afin de participer à des processus coopératifs. Le contexte du service domaine comporte trois sous-catégories :

- Les informations concernant l'exécution du service : présentées dans l'ontologie comme (*Performance_Quality)*, elles incluent un ensemble de paramètres caractérisant la performance des services tels que :

Le temps de réponse (*Response Time*) : caractérise le temps écoulé depuis la soumission de la requête jusqu'à la réception de la réponse,

La disponibilité (*Disponibility*) : représente le pourcentage de temps pendant lequel un service est opérationnel. Les valeurs les plus grandes montrent une disponibilité élevée tandis que les petites valeurs impliquent une disponibilité basse,

Le taux d'accessibilité (*Accessibility*) : représente le ratio du nombre de réponses rendues par le service par rapport au nombre global d'appels.

La sécurité (*Security*) : illustre la capacité d'un service à proposer des mécanismes de sécurité. Des exemples de ces mécanismes incluent le cryptage (cryptage des messages en entrée et en sortie), et aussi l'authentification (identification du client de service).

- Les informations concernant les considérations métier du service : présentées dans l'ontologie comme (*Business Consideration*), ces informations englobent les paramètres relatifs à la consommation du service tels que :

Le prix (*Price*) : représente le montant monétaire échangé contre la consommation du service. D'une manière générale, il illustre le prix d'invocation des opérations du service.

La réputation (*Reputation*) : désigne la réputation du service suite au retour d'expériences des clients. Cette réputation est calculée est mise à jour par le gestionnaire de réputation de la communauté à laquelle appartient le service.

Les modalités de payement (*Payment*) : expose les moyens de payement acceptés par le fournisseur de services comme par exemple le transfert bancaire, la carte VISA, *etc.*

La supervision (*Monitoring*) : désigne les mécanismes offerts par le fournisseur de service pour contrôler l'exécution du service et notamment en cas d'échec. Par exemple, la proposition d'un portail de contrôle, des échanges de mail à chaque étape d'exécution du service, *etc.*

La catégorie métier du service (*Business Category*) : porte sur le secteur d'activité représenté par le service. Par exemple, un service domaine de livraison appartient au secteur du transport.

Les informations concernant l'exécution et aussi les considérations métier d'un service permettent de différencier un service domaine par rapport à un autre. Ces différents éléments interviennent lors du processus de découverte et de sélection d'un service domaine pour participer à un processus coopératif.

- Les informations concernant le fournisseur du service : présentées dans l'ontologie comme (*Provider_Related_Context*), elles encapsulent un ensemble de paramètres qui caractérisent le fournisseur du service tels que son identité, sa localisation, ses historiques d'interaction.

- **Contexte du client (*Customer_Related_Context*) :**

Le contexte du client regroupe l'ensemble des informations et des métadonnées utilisées par les fournisseurs de services afin d'adapter leurs services. Des exemples du contexte du client incluent :
- Le profil : désigne des paramètres caractérisant le client tels que son identité, sa localisation, ses activités, ses performances industrielles, *etc.*,
- Les préférences : regroupent les préférences déjà enregistrées pour le client comme par exemple ses préférences en termes de coût de service, de réputation, de préférence concernant le temps d'exécution du service *etc.* Ces préférences sont exprimées lors de l'instanciation des Goals Templates.

- **Contexte de la coopération (*Cooperation_Related_Context*) :**

Le contexte de coopération représente les paramètres métier caractérisant une opportunité de coopération. Nous pouvons identifier par exemple : le lieu, la date et le domaine d'activité. Le lieu et la date représentent la localisation géographique et la période de temps durant laquelle se réalise l'opportunité de coopération. Le domaine renseigne sur le secteur d'activité de la coopération.

Nous modélisons les informations du contexte relatives au service domaine, au client et à la coopération sous la forme d'un ensemble de contraintes appelées *Context Constraints*. Nous présenterons dans la section 3.4.1 de ce chapitre de thèse les mécanismes et le langage proposés afin de représenter les différents *Context Constraints*.

- **Concept de situation contextuelle**

Nous nous sommes inspirés des travaux réalisés pour la sensibilité au contexte dans le domaine des applications mobiles [Dey et al., 2001] afin de caractériser la notion de situation contextuelle dans le cadre d'une coopération interentreprises. Une situation contextuelle correspond à une instance des paramètres d'un contexte donné. Dans le cadre du service domaine, cette situation ne modifie pas le but satisfait par le service, mais peut conduire à altérer ou à modifier son comportement initial. Par exemple une situation contextuelle d'un client peut être caractérisée par les paramètres suivants : {client= « entreprise production de jouets », préférence de livraison= « transport routier », localisation= « Russie »}. Le service domaine doit être adapté à cette situation selon les préférences exprimées (utilisation d'un service de transport terrestre au lieu d'un transport aérien) et selon la localisation (par exemple, en favorisant les moyens de transport adapté au climat russe), *etc.* Pour définir formellement une situation contextuelle, nous considérons un espace multidimensionnel où chaque dimension représente une catégorie de contexte déjà présentée dans l'ontologie de catégorisation de contexte définie dans le paragraphe précédent. Nous définissons une situation contextuelle dans le cadre d'un processus coopératif comme un vecteur à trois dimensions :

Situation-Contextuelle= (*ContexteService, ContexteClient, ContexteCoopération*)

Ces différentes dimensions affectent l'exécution d'un service domaine appartenant au processus coopératif.

À un instant donné, une situation contextuelle est définie par les valeurs associées aux paramètres de contexte. Certaines valeurs sont statiques et ne changement pas durant l'interaction avec le client, d'autres sont dynamiques. La modification d'une de ces valeurs correspond à une transition vers une autre situation contextuelle. Cette transition est le résultat de nouveaux événements. Ces derniers déclenchent la modification de la logique métier du service domaine en définissant de nouvelles vues sur les opérations appartenant aux différents services fonctionnels qu'il orchestre. Le prochain chapitre de ce manuscrit se préoccupera de détailler l'adaptation de la logique métier du service domaine grâce à l'injection des aspects.

3.2.3.2.2 Composition des services domaines au sein d'un processus coopératif

La composition des services domaines au sein d'un processus coopératif est montrée en gris clair au niveau de la Figure 3.8. Cette composition décrit l'enchaînement et la coordination des services domaines en termes d'interactions et des flux de contrôle qui les relient.

La définition du concept de composition au sein d'un processus coopératif s'inspire des normes BPMN et BPEL [BPMI, 2002], [Andrews et al., 2003b]. Ainsi les classes qui héritent de ces normes sont les suivantes :
- La classe « *partenaire* » correspond à un partenaire de processus qui n'est autre qu'un service domaine proposé par un partenaire,

- 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 illustre les classes de base dans un processus qui peuvent être :

- La classe « *receive* » désigne l'attente de l'arrivée d'un message,
- La classe « *invoke* » désigne l'appel à un service,
- 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 des parties de processus :

- La classe « *while* » permet l'exécution d'une partie de processus en boucle jusqu'à la satisfaction d'une condition,
- La classe « *pick* » permet d'attendre la production d'un événement (arrivée d'un message, *etc.*) pour continuer l'exécution du processus,
- La classe « *scope* » permet de regrouper un ensemble d'activités. Ce regroupement peut avoir ses propres variables et ses propres gestionnaires d'évènements,
- 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,
- La classe « *sequence* » permet de lier directement deux parties dans le processus. L'exécution de la deuxième partie suit l'exécution de la première,
- La classe « *flow* » permet l'exécution en parallèle de plusieurs activités de processus.

3.3 Présentation générale de l'architecture de coopération à la demande

Le paradigme de coopération par composition des services, permet aux entreprises partenaires de définir leurs services, de sélectionner des services et finalement de former le processus coopératif afin de répondre à l'objectif de la coopération. Suivant cette approche, la formation de la coopération à la demande s'appuie sur des concepts importants à savoir : la description, la publication, la découverte et la composition des services.

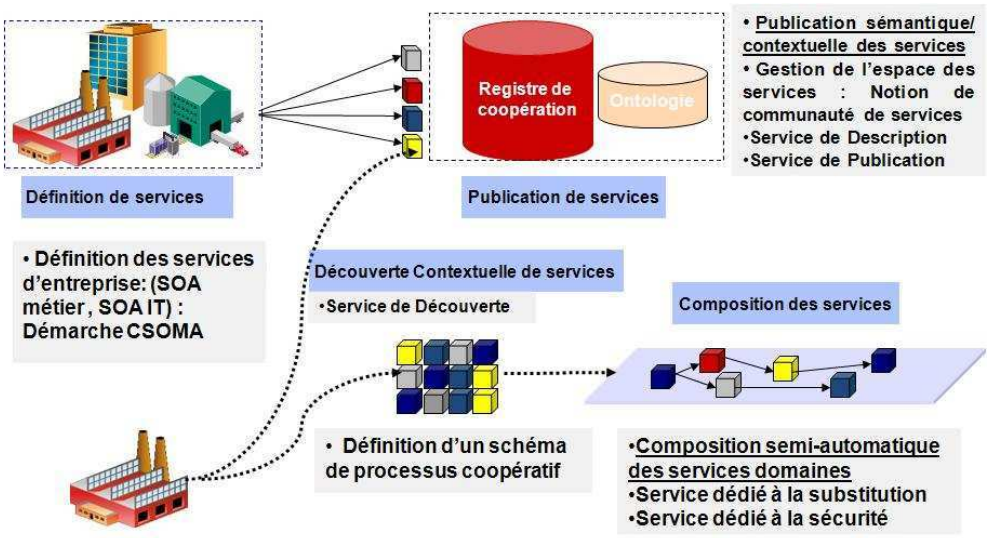


Figure 3.10. Architecture de coopération à la demande

La Figure 3.10 illustre l'architecture de coopération proposée qui supporte toute la démarche de cette thèse. La littérature concernant la gestion de la coopération à la demande est très limitée. Dans les coopérations interentreprises statiques, le CMI (*Collaborative Management Infrastructure*) reste en effet l'étude la plus spécifique qui introduit une gestion sophistiquée des rôles ainsi que des activités collaboratives [Georgakopoulos et al., 2000]. Cependant, les concepts développés restent limités et ne répondent pas aux besoins de la coopération à la demande.

Dans le cadre du processus coopératif à la demande, il nous semble intéressant afin de permettre le bon déroulement de la coopération, de définir un ensemble de services appelés services de coopération. L'exercice consiste donc à définir en fonction des principes fondateurs de la coopération à la demande (cf. section 3.1), l'ensemble des services qui puissent les satisfaire. Nous avons ainsi défini cinq types de services relatifs à la gestion du processus coopératif (voir Figure 3.10) :

- Service de description permettant la description sémantique et contextuelle des services. Afin que les services domaines puissent être utilisés (et réutilisés), le service de description, comme son nom l'indique, se charge de décrire la sémantique et les paramètres de contexte des services à l'aide de la notion d'ontologie,
- Service de publication permettant la publication sémantique et contextuelle des services domaines,
- Service de découverte permettant la découverte sémantique et contextuelle des services domaines,
- Service dédié à la substitution des services permettant de substituer des services au sein du processus coopératif,
- Service dédié aux aspects de sécurité assurant la sécurité d'accès aux différents services domaines.

Nous traitons brièvement dans la suite de cette section les deux derniers services de coopération qui constituent des services supports et pour lesquels un certain nombre de travaux a été déjà réalisé [Dumas et al., 2008], [Benatallah et al., 2006]. Les autres services seront présentés en détail dans la suite de ce chapitre.

– Le service dédié à la substitution des services

Dans un processus coopératif, il arrive parfois qu'un service domaine ne soit plus disponible (*i.e.*, tombe en panne), ce qui nécessite de le substituer par un autre service domaine similaire. Plusieurs raisons peuvent être à l'origine du déclenchement d'un processus de substitution de services. Plusieurs auteurs dans la littérature relative à la substitution des services ont essayé d'établir une revue portant sur les raisons à l'origine de la substitution. Ces raisons peuvent être : (1) l'arrêt d'un service, (2) la modification de l'interface/protocole du service, et enfin (3) la dégradation de la qualité de service [Dumas et al., 2008], [Bordeaux et al., 2004], [Benatallah et al., 2006]. En considérant ces raisons, nous attribuons au service dédié à la substitution les tâches suivantes :

- La substitution des services pour des raisons d'arrêt : la finalité de cette substitution consiste à assurer l'interopérabilité au sein du processus coopératif construit par composition des services domaines. En effet, l'arrêt d'un service participant risque de mettre en péril le fonctionnement du processus coopératif. Pour remédier à ce problème, le service dédié à la substitution se charge de remplacer le service par un autre disponible.
- La substitution correctionnelle : l'objectif de cette substitution est de corriger le problème d'incohérence qui pourrait se produire entre les services suite à la modification d'interface/protocole.
- La substitution pour l'amélioration de la qualité : le but de cette substitution est de maintenir le niveau de qualité de service exigé par le client. Au bout d'un certain temps, un service donné peut subir une certaine dégradation au niveau de sa qualité. Ainsi, le service de substitution se charge de changer le service par un autre dont le niveau de qualité correspond à celui demandé par le client.

Vu l'abondance des travaux de recherche portant sur cette thématique, nous nous sommes contentés dans le cadre de cette thèse de réutiliser les avancées déjà réalisées, en particulier les travaux de [Dumas et al., 2008], [Benatallah et al., 2006].

– Le service dédié aux aspects de sécurité

La visibilité d'un service domaine désigne le droit d'utiliser les points d'accès aux ressources du processus/sous processus encapsulé par le service. Dans les systèmes de gestion des processus (tels que les systèmes de gestion de Workflow), seulement deux alternatives sont possibles concernant les droits d'accès à un processus : *boîte noire* (*Black Box* ou *BB*) et *boîte blanche* (*White Box* ou *WB*). Concernant la *boîte noire*, ni les appels de méthodes de processus ni les événements de son exécution ne sont accessibles pour une entreprise externe. Quant à la *boîte blanche*, elle permet à l'interface du processus d'être complètement accessible (méthodes et événements). Si une entreprise estime que la deuxième alternative est très permissive (à l'égard de la confidentialité, de la sécurité par exemple), elle présente son processus comme étant une *boîte noire*. La sécurité de cette dernière est évidente, elle n'est, cependant, pas viable du point de vue de coopération interentreprises.

Dans le cadre de service domaine, nous avons adopté l'approche appelée « boite grise ». Cette dernière offre la possibilité aux entreprises partenaires de visualiser l'avancement des processus externalisés et même d'avoir des possibilités d'exercer des activités de *monitoring*. Nous reviendrons sur cette approche « boite grise » dans le cadre du prochain chapitre de cette thèse.

Le service dédié aux aspects de sécurité se charge de contrôler l'accès aux différents processus internes encapsulés par les services domaines participant dans le cadre d'un processus coopératif. De plus, il se préoccupe de la vérification de l'identité des entreprises coopératives et au respect des différents contrats de coopérations élaborés.

3.4 Services de coopération pour la composition des services

Après avoir présenté une vision globale de l'architecture de coopération, nous allons détailler les principales étapes indispensables pour la construction du processus coopératif à base de composition des services.

3.4.1 Description des services domaines

La recherche de flexibilité nous a poussé à considérer deux aspects fondamentaux lors de la description des services qui sont : (i) la prise en compte de la sémantique à travers la notion d'ontologie et (ii) l'enrichissement de la description des services par l'ensemble des paramètres contextuels. La sémantique permet d'offrir le meilleur moyen afin de rendre la découverte ainsi que la composition des services plus flexibles et plus intelligibles à travers l'utilisation des technologies Web sémantique. Quant à l'enrichissement de la description des services domaines en utilisant les paramètres contextuels, il part du constat que si les fournisseurs de services domaines mettent à la disposition des clients une description du contexte d'utilisation pour lequel le service est adapté, la découverte peut être facilitée.

Le service de description appartenant à l'architecture de coopération se charge de récupérer les descriptions des services domaines fournis par les entreprises partenaires et de les enrichir sémantiquement et contextuellement et ceci en deux étapes :

- **Étape 1. Description sémantique des services domaines**

L'objectif de cette étape consiste à décrire sémantiquement les services domaines proposés par les fournisseurs. Cette description se base en particulier sur l'enrichissement de l'ontologie OWL-S qui constitue à notre sens l'initiative la plus mature pour la mise en œuvre des services sémantiques. Cette préoccupation s'inscrit dans la continuité des problématiques déjà traitées dans les travaux de [Izza, 2006]. L'ontologie de description sémantique du service domaine permet d'enrichir la sémantique générique d'OWL-S par une sémantique spécifique (voir Figure 3.11). Cette ontologie a été décrite dans l'état de l'art (cf. section 1.2.3.2).

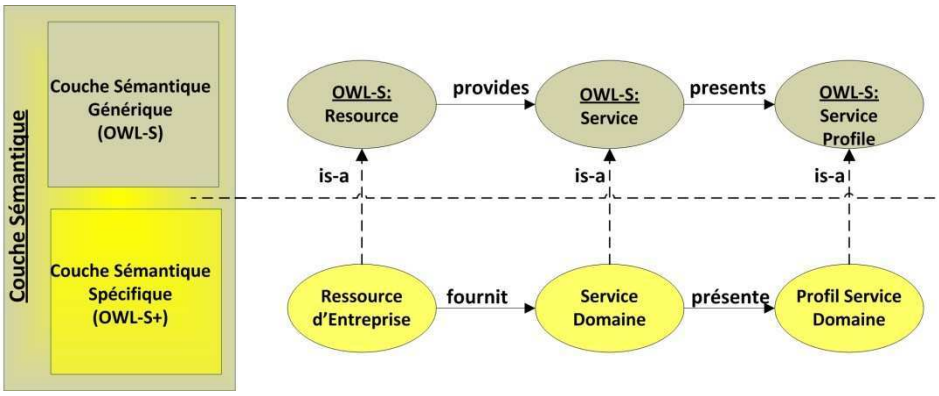


Figure 3.11. Extrait de l'ontologie OWL-S+ [Izza et al., 2005]

- **Étape 2. Description contextuelle des services domaines**

La description contextuelle appelée aussi modélisation du contexte des services permet de représenter les informations relatives au contexte du service, présentées au niveau de l'ontologie de catégorisation de la Figure 3.9. Cette description vise à permettre la mise en œuvre de différents processus d'adaptation que ce soit lors de la découverte ou encore lors de l'exécution des services.

Avant de présenter notre approche de description contextuelle des services domaines nous allons présenter ci-après un rappel des travaux permettant la modélisation des informations contextuelles.

– **Approches classiques de modélisation du contexte**

La modélisation du contexte constitue une étape fondamentale dans le processus de construction des applications sensibles au contexte.

Rappelons que notre étude de la littérature a mis en évidence trois types d'approches pour la modélisation du contexte : l'approche paires/triplet, l'approche orientée modèle, et l'approche orientée ontologie (cf. chapitre 1, section 1.3.3.3). Les approches paires/triplets sont caractérisées par un faible niveau d'expressivité et la simplicité des données qu'elles représentent. Les approches orientées modèle sont des approches prometteuses car elles utilisent non seulement un modèle formel pour décrire le contexte, mais offrent aussi un méta-modèle de description qui peut être réutilisé par plusieurs applications. Les approches orientées ontologie sont des approches formelles qui tirent parti des caractéristiques des ontologies pour modéliser le contexte.

En comparant ces trois approches, il s'est avéré que l'approche orientée ontologie est l'approche la plus expressive et la plus prometteuse pour la description du contexte dans un environnement sensible au contexte (cf. chapitre 1, section 1.3.3.3).

– **Modèle multi-niveaux pour les informations contextuelles**

Afin de modéliser les informations contextuelles des services domaines nous proposons un modèle de contexte multi-niveaux (extrait de [Boukadi et al., 2008a]) basé sur l'ontologie OWL (voir Figure 3.12). Le concept de base de ce modèle est le concept de *Context*

Constraint qui est défini à son tour grâce à deux autres concepts à savoir : *Context Data* et *Context Assertion*. Nous allons détailler ces différents concepts dans ce qui suit.

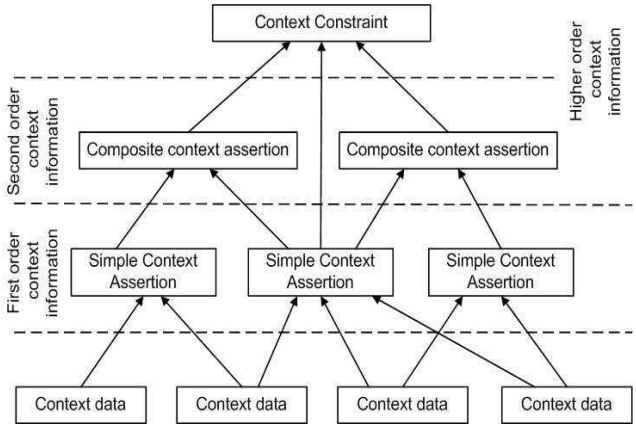


Figure 3.12. Modèle de contexte multi-niveaux

- ***Context Data (CD)*** sont définis comme les valeurs des données pour les informations contextuelles. Ils représentent les données à un faible niveau d'abstraction correspondant à des valeurs uniques issues de la capture du contexte. *Context Data* correspond aux différentes catégories de contexte présentées dans l'ontologie de catégorisation de contexte (voir Figure 3.9). Comme exemple de *Context Data* nous pouvons noter localisation, temps de réponse, *etc.*
- ***Simple Context Assertion (SCA)*** est un prédicat qui vérifie la validité d'une condition et se compose d'un opérateur et deux opérandes. La première est de type *Context Data*, tandis que la seconde peut être soit *Context Data* soit une valeur prédéfinie. Exemple de *Simple Context Assertion* peut être (Coût= 10 €) ou (TempsRéponse= 5 secondes).
- ***Composite Context Assertion (CCA)*** est défini par conjonction de plusieurs *Simple Context Assertion*. Grâce à ce concept des informations contextuelles de haut niveau peuvent être obtenues.
- ***Context Constraint (CCx***) regroupe un ou plusieurs Composite *Context Assertion*. Comme exemple de *Context Constraint* nous pouvons considérer CCx = (Coût= 10 € ^TempsRéponse= 5 secondes) ^(Localisation=France ^ Payement=VISA).

Afin de spécifier les *Context Constraints* nous avons proposé une ontologie de haut niveau appelée *Context Constraint Upper Ontology*. Cette ontologie permet de décrire les concepts indispensables afin de représenter les *Context Constraints*. La figure suivante illustre les concepts inclus dans le *Context Constraint Upper Ontology*.

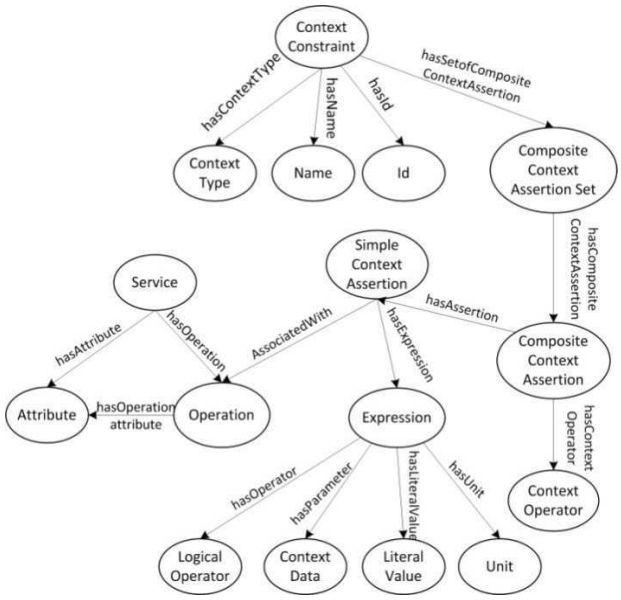


Figure 3.13. Context Constraint Upper Ontology [Boukadi et al., 2008a]

Comme on peut le constater sur la Figure 3.13, l'ontologie décrit un ensemble de concepts :

- *Context Constraint* : représente la racine de l'ontologie. Elle se réfère à une catégorie de contexte et possède un nom et un identifiant,
- *Simple Context Assertion* et *Composite Context Assertion* ont été déjà définies dans la section précédente,
- *Context Operator* est défini pour le *Composite Context Assertion Set* afin de préciser le niveau de combinaison entre les différents *Composite Context Assertion.* On définit trois types de *Context Operator* à savoir : « *All* », « *ExactlyOne* » et « *OneOrMore* ». L' opérateur « *All* » indique que toutes les *Composite Context Assertion* au sein de l'ensemble (*Composite Context Assertion Set*) vont être considérées. L'opérateur « *ExactlyOne* » précise qu'à un moment donné une seule *Composite Context Assertion* va être appliquée. Enfin l'opérateur « *OneOrMore* » indique qu'au moins une *Composite Context Assertion* sera appliquée,
- *Expression* assure un traitement automatique d'une *Context Assertion* et se présente comme une expression comportant un *Context Data*, un *Logical Operator*, un *Literal Value* et un *Unit* qui caractérise l'unité de mesure utilisé pour le paramètre en question,
- *Logical Operator* représente une relation entre le *Context Data* et sa valeur. Exemples de *Logical Operateur* peuvent être {=, ≠, ≤, ≥, >, <, min, max, set}.

Comme nous l'avons déjà souligné auparavant, l'idée d'avoir un niveau d'ontologie pour décrire les concepts communs et un niveau plus spécifique au domaine nous semble être une idée intéressante à exploiter (cf. chapitre 1, section 1.3.3.3). C'est la raison pour laquelle nous proposons un modèle d'ontologie hiérarchique se composant de deux niveaux distincts (voir Figure 3.14). Le premier niveau comporte deux ontologies indépendantes du domaine qui sont : le *Context Constraint Upper Ontology* (Ontologie haut niveau du *Context Constraint*) et l'ontologie de catégorisation de contexte (voir Figure 3.9). Tandis que le deuxième niveau

comporte un ensemble d'ontologies de domaine qui se chargent de spécialiser les concepts présentés dans les deux ontologies du premier niveau. Par définition les ontologies de domaines consistent en une représentation formelle des concepts du domaine étudié ainsi que des différentes relations qui les lient. Des exemples d'ontologies de domaines peuvent inclure l'ontologie des unités, l'ontologie des opérateurs, l'ontologie géographique, *etc*.

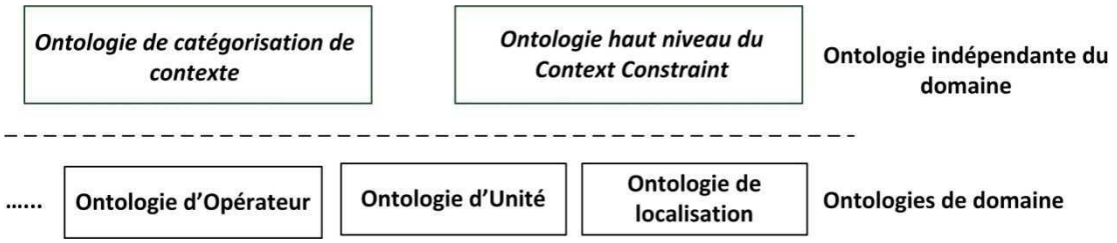


Figure 3.14. Modèle d'ontologie

Introduire un modèle d'ontologie hiérarchique peut être justifié par deux arguments. D'une part, les paramètres de contexte d'un service peuvent dépendre d'un domaine particulier en fonction du service à décrire. Ainsi, les ontologies de domaines complètent la description proposée au niveau de l'ontologie de catégorisation du contexte. D'autre part, durant le processus de découverte de services qui participeront dans un processus coopératif, les ontologies de domaines peuvent être utilisées afin de compléter les concepts décrits par le *Context Constraint Upper Ontology*. Par exemple, une ontologie d'unité peut déduire que 1 seconde est équivalente à 1000 millisecondes.

L'exemple introduit dans la Figure 3.15 illustre un *Context Constraint* relatif à un service domaine de livraison. Comme nous le pouvons remarquer sur la figure, il s'agit de paramètre de contexte relatif au coût d'exécution du service. Les lignes (01-04) contiennent les espaces de noms : *ccuo*, *cxco* et *dsuo* qui correspondent respectivement aux URLs du *Context Constraint Upper Ontology*, de l'ontologie de catégorisation de contexte et l'ontologie d'unités. Les lignes (21-27) indiquent que le coût du service est égal à 10 €.

```
(01) <xmlns:
        ccuo= " ontology/ContextConstraintUpperOnto"
(03)    cxco= " ontology/ContextCategorizationOnto"
        dsuo= " ontology/DomainSpecificUnitOnto" >

(05)    <cxco : deliveryCx  rdf: ID = "DeliverParcel"
       <ccuo: hasCompositeContextAssertion  rdf:resource:"  CompositeContextAssertion"/>
(07)   < ccuo: hasContextOperator  rdf:resource:" ccuo:ExactlyOne" />
       < ccuo: hasAssertion>
(09)       <ccuo:SimpleContextAssertion   rdf:resource:" # ContextAssertion1"/>
           <ccuo:SimpleContextAssertion   rdf:resource:" # ContextAssertion2"/>
       < ccuo: hasAssertion>
(13)   <ccuo: ContextAssertion   rdf:ID="ContextAssertion1">
           <ccuo: hasExpression>
(15)       <ccuo: Expression   rdf:ID="ExpressionContextAssertion1">
           <ccuo: hasParameter>
(17)       <ccuo: Attribute  rdf:resource= " cxco: Cost"/>
           </ccuo: hasParameter>
(19)       <ccuo: hasExpressionOperator rdf:resource=" IsEqualTo"/>
           <ccuo: hasLiteralValue>
(21)       <ccuo: LiteralValue  rdf:ID="literalValue1">
           <ccuo: hasValue> 10 </ccuo: hasValue>
(23)       <ccuo: hasUnit> dsuo: euro </ccuo: hasUnit>
           </ccuo: LiteralValue>
(25)       </ccuo: hasLiteralValue>
           </ccuo: Expression>
(27)       </ccuo: hasExpression>
       </cxco : ContextAssertion>
```

Figure 3.15. Exemple de Context Constraint

3.4.2 Publication des services domaines

3.4.2.1 Approches classiques de publication

À l'issue de la description syntaxique, sémantique et contextuelle des services domaines, ces derniers doivent être publiés afin qu'ils puissent être éventuellement découverts et utilisés par des processus coopératifs. Traditionnellement W3C [W3C, 2009] identifie deux principales approches de publication qui sont :

- l'approche basée sur un référentiel unique qui permet de centraliser l'ensemble des descriptions au sein d'un même référentiel,

- l'approche *peer-to-peer* qui se base sur l'utilisation d'un ensemble de référentiels permettant ainsi de distribuer les descriptions sur un ensemble de nœuds.

3.4.2.2 Notre approche de publication des services domaines

Notre approche est considérée comme une combinaison de ces deux approches dans le sens où nous proposons une approche hiérarchique de publication, qui se base sur la dualité registre de la coopération/communauté. Une description complète du service en question doit être contenue dans le registre de la coopération. Tandis que dans le registre local de la communauté, il doit y avoir une référence de cette description.

La publication des services domaines se base sur les descriptions syntaxiques, sémantiques et contextuelles déjà définies (cf. section 3.4.1). Notre approche de publication se base sur une extension sémantique/contextuelle des référentiels UDDI. Comme nous l'avons souligné dans

l'état de l'art (cf. section 1.2.3.3), le registre UDDI est le registre le plus utilisé pour le stockage et la découverte des services Web appartenant à diverses entreprises. Ceci peut être confirmé par le nombre de registre UDDI qui ne cesse d'augmenter sur le Web. Cependant, la spécification actuelle de l'UDDI n'est pas encore assez mature pour tenir compte des informations sémantiques et contextuelles. Par conséquent, nous proposons une approche de publication qui consiste en une extension sémantique et contextuelle du registre UDDI.

L'approche de publication que nous proposons est stratifiée et organisée en couches afin d'apporter le maximum de flexibilité. Les services domaines sont décrits et publiés syntaxiquement dans le registre de coopération avant d'être éventuellement décrits puis publiés sémantiquement dans un registre sémantique lié au registre de coopération. Par la suite, les informations contextuelles de ces services sont également publiées dans le registre de coopération afin qu'elles puissent être utilisées dans le processus de découverte décrit dans la prochaine section de ce chapitre de thèse. Cette technique qui est plus réaliste apporte plus de flexibilité en permettant ainsi à certains services domaines d'exister sans description sémantique. Dans la même direction que les travaux réalisés dans [Izza et al., 2005] nous adoptons le principe qui recommande l'utilisation à la fois des descriptions syntaxiques et/ou sémantiques appelé le principe de cohabitation syntaxique-sémantique. Ce dernier est utile si l'on considère que le processus de description sémantique des services est long et nécessite un cycle incrémental et itératif durant lequel des services peuvent exister avec différents degrés de maturité sans nécessairement disposer de description sémantique (degré 1 : service syntaxique, degré 2 : service sémantique).

Le processus de publication est pris en charge par le service de publication. Il s'active dès la réception d'une requête de type :

Request (# Publish, PublicationType, Publication)

Où *PublicationType* désigne le type de publication (publication syntaxique de service, publication sémantique de service, publication des informations contextuelles), et où *Publication* désigne les caractéristiques (sous forme de fichier XML ou OWL) à publier.

Le service de publication étend la publication traditionnelle en permettant la publication de descriptions sémantiques et des informations contextuelles grâce aux deux fonctionnalités (API) que nous avons développées et qui s'intitulent *PublishSemanticService* et *PublishContextuelInformation*. Celles-ci permettent respectivement de publier la description sémantique d'un service et les informations contextuelles.

Concernant la première fonctionnalité, nous adoptons la même API que celle développée dans [Izza, 2006]. Pour la deuxième fonctionnalité, la publication des informations contextuelles, nous avons choisi de les ajouter dans le champ *tModel* [W3C, 2006] figurant dans la spécification UDDI. En effet, le *tModel* est une structure de donnée appartenant à l'UDDI permettant d'enregistrer la spécification technique d'un service. Chaque *tModel* sera référencé dans le *business service entry* d'un service publié (cf. chapitre 1, section 1.2.3.3). Ainsi, chaque service domaine possède un ensemble de *tModels* qui stockent les informations contextuelles qu'il présente (modélisées sous la forme de *Context Constraints*). Nous avons

choisi pour des raisons de simplicité de représenter chaque paramètre de contexte dans un fichier *tModel* séparé.
La structure *tModel* est définie formellement de la manière suivante :

tModel=< {ID-CCx, URL-CCx, ID_SD, Cgx} >

où *ID-CCx* est l'identifiant du *Context Constraint* du service, *URL-CCx* représente l'URL du fichier OWL relatif à la *Context Constraint*, *ID_SD* est l'identifiant du service domaine, et *Cgx* désigne la catégorie du paramètre de contexte présenté par le *Context Constraint*.

La Figure 3.16 illustre un exemple d'utilisation de la structure *tModel* afin de publier l'information contextuelle relative au coût du service de livraison déjà présenté dans la section précédente. L'attribut *tModelKey* (ligne 01) dans la balise tModel correspond à un identifiant unique généré par l'UDDI pour désigner le tModel. Le premier *keyedReference* (ligne 7-10) définit l'URL du fichier OWL qui représente le *Context Constraint*. Le deuxième *keyedReference* (ligne 11-14) correspond à l'identifiant du service domaine à qui le *Context Constraint* sera attaché. Le troisième *keyedReference* (ligne 15-19) représente la catégorie du paramètre de contexte (Coût du service).

```
(01) <tmodel tModelKey=" uuid:04cfa...."
          <name>...</name>
(03)      <description xml:lang="EN">
           Context Constraint for a delivery service domain
(05)         </description>
           <CategoryBag>
(07)       < keyedReference
              KeyName=" Context Constraint Identifier"
(09)          KeyValue=" identifier"
              tModelKey="uuid:fa1d77cd-edf0-3a84-a99a-5972e43e9993"/>

(11)       < keyedReference
             KeyName=" Context Constraint for Delivery Service"
(13)         KeyValue=" http://www.example.com/myservice/ContextConstraint"
              tModelKey="uuid:kb2d77cd-edf0-3a84-a99a-5972e43e7773"/>

(15)          < keyedReference
              KeyName=" Categorization"
(17)          KeyValue=" Price"
              tModelKey="uuid:kb2d88ej-edf0-3a84-a99a-5972e43ebb43"/>
(19)          </CategoryBag>
            </tmodel>
```

Figure 3.16. Publication d'un paramètre de contexte dans la structure *tModel*

Pour conclure, la publication d'un service domaine est effectuée comme suit : au début, les descriptions syntaxiques WSDL sont publiées dans le registre de coopération en utilisant les API standard *PublishService* puis il s'agit de compléter les descriptions syntaxiques par des informations contextuelles grâce à l'API *PublishContextuelInformation*. Enfin, il s'agit de publier les descriptions sémantiques (OWL-S+) qui constituent des descriptions sémantiques complétées par des références à l'UDDI et qui sont publiées dans un registre d'ontologies en utilisant l'API *PublishSemanticService*. La Figure 3.17 récapitule le principe de fonctionnement du service de publication.

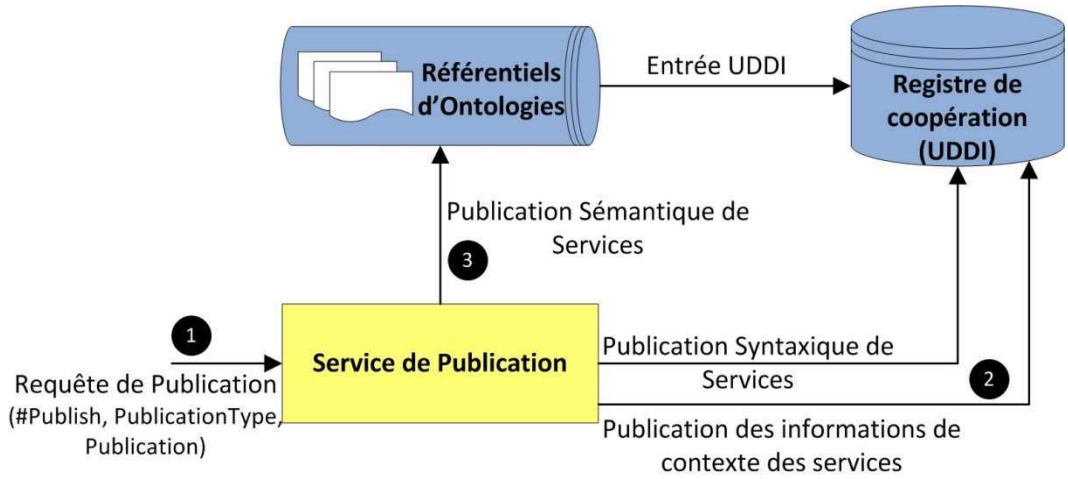


Figure 3.17. Principe de fonctionnement du service de publication

3.4.3 Découverte des services

Une fois que le schéma abstrait du processus coopératif a été défini, il s'agit par la suite de découvrir les services qui correspondent aux exigences exprimées par le client. Comme nous l'avons déjà souligné, le client exprime des conditions que le service doit remplir. Ces conditions concernent à la fois les exigences fonctionnelles (*i.e.*, le ou les fonctionnalités supportées par le service) ainsi que les exigences contextuelles (*i.e.*, les contraintes contextuelles appelées aussi *Context Constraints*).

3.4.3.1 Approches classiques de découverte de services

Dans la littérature, plusieurs travaux traitent le domaine de découverte des services Web [Kritikos and Plexousakis, 2006], [Maximilien and Singh, 2004a]. La majorité de ces travaux s'intéressent particulièrement à la comparaison des descriptions textuelles entre la requête du client et la signature des services (*inputs* et *outputs*). La découverte sémantique des services a été introduite puisqu'une description strictement syntaxique des services Web ne peut permettre une recherche et une sélection efficaces de ces services. L'objectif visé par la notion de services Web sémantiques est de créer un Web sémantique de services dont les propriétés, les capacités, les interfaces et les effets sont décrits de manière non ambiguë et exploitable par des machines. Ces travaux reposent sur des langages et des concepts issus du domaine de l'intelligence artificielle et mettent en œuvre des mécanismes facilitant les étapes de recherche et de sélection pour les clients.

D'autres travaux se sont penchés sur la prise en considération des paramètres de qualité lors de la découverte des services [Al-Masri and Mahmoud, 2007], [Diamadopoulou et al., 2008], [Garcia et al., 2006]. Les paramètres de qualité de services (QoS) regroupent les paramètres qui caractérisent le comportement des services dans leurs fonctionnalités. Des exemples de ces paramètres incluent la sécurité, l'authentification, *etc.* Les travaux qui adoptent cette piste partent du constat que vu le nombre important des services similaires de point de vue fonctionnel, il est intéressent de se doter de descriptions qui permettent de les différencier et

par la suite de les classer. C'est le rôle que peuvent jouer les paramètres de qualité appelés encore les paramètres non fonctionnels.

D'autres travaux se sont orientés vers la découverte contextuelle des services. Cette dernière trouve ses origines dans les applications mobiles où l'objectif consiste à considérer la connaissance relative aux préférences de l'utilisateur et les caractéristiques contextuelles afin de chercher les services qui correspondent au mieux au besoin du client [Broens et al., 2004], [Dey et al., 2001]. En effet, dans ce genre d'application, l'utilisateur s'attend à un service qui respecte ses propres caractéristiques (telles que ses données personnelles : son nom, ses préférences, *etc.*) ainsi que celles de son dispositif d'accès (telle que la taille de l'écran, la limite de sa bande passante, *etc.*). En passant en revue ces travaux (cf. chapitre 1, section 1.3.5), nous avons constaté que la découverte contextuelle des services n'a pas été exploitée dans le cadre de la coopération interentreprises à la demande.

Nous considérons que la prise en compte du contexte permet de répondre à l'une des problématiques de recherche : l'adaptation. En réalité, lors de la construction du processus coopératif, l'intégration des éléments de contexte afin de découvrir et de sélectionner les services constitue une forme d'adaptation. En effet, l'étape de découverte consiste à filtrer dans la collection de services domaines disponibles sur le registre de services, ceux correspondant au mieux, tant aux besoins fonctionnels du client, qu'aux besoins en termes d'adaptation au contexte d'utilisation (profil du client, temps et lieu de la coopération, *etc.*).

3.4.3.2 Approche de découverte proposée

Dans le cadre de notre processus de découverte et de sélection de services nous proposons d'utiliser la notion de situation contextuelle courante afin de sélectionner les services correspondant à la fois aux exigences fonctionnelles ainsi qu'aux contraintes de contexte. Ces dernières sont exprimées au sein des Goals Templates. Il est important de souligner que la découverte des services domaines est à la charge du service de découverte (cf. section 3.3).

Notre processus de découverte des services est présenté dans la Figure 3.18. En effet, le service de découverte utilise un module de *matching*, un évaluateur de contexte, le registre de services ainsi que les différentes communautés présentes [Boukadi et al., 2008].

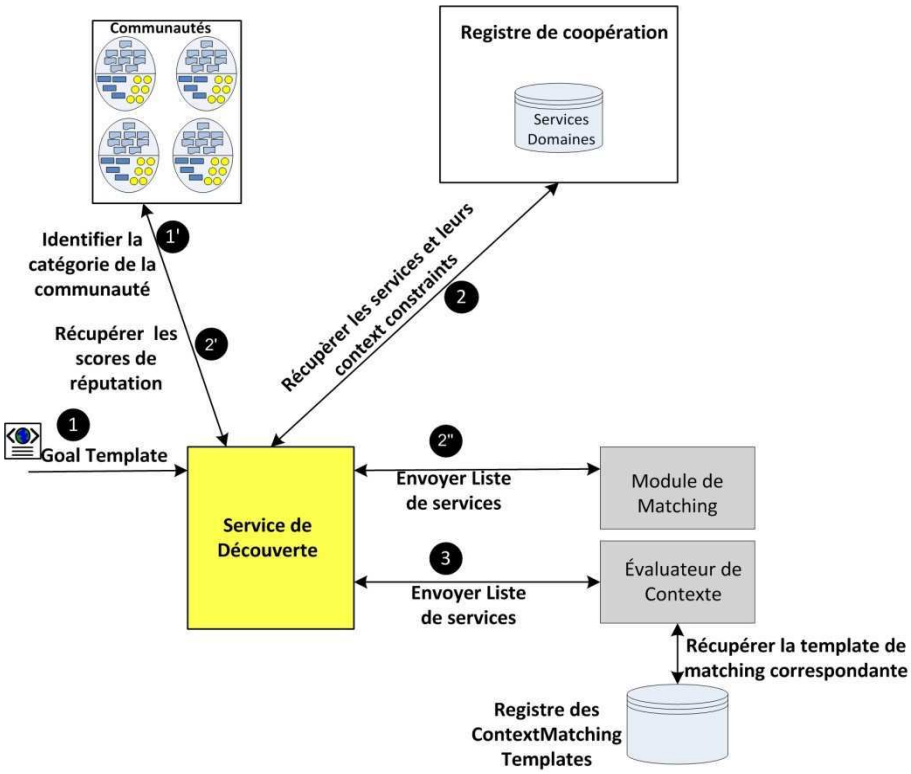


Figure 3.18. Processus de découverte des services domaine [Boukadi et al., 2008a]

Le processus de découverte des services domaines qui participeront au processus coopératif est décrit en trois étapes :

- **Étape 1. Identification de la communauté en question**

Quand le service de découverte récupère les informations définies dans un Goal Template, il commence tout d'abord par déterminer la catégorie de la communauté sélectionnée par l'entreprise initiatrice du projet de coopération.

- **Étape 2. Identification des services domaines qui correspondent aux méthodes génériques instanciées**

Le service de découverte récupère par la suite la liste des services domaines membres de la communauté ainsi que la liste de leurs *Context Constraints* à partir du registre de coopération. Il interroge ensuite le gestionnaire de réputation de la communauté en question pour récupérer les scores de réputation des services domaines.

Une fois ces informations collectées, le service de découverte envoie une requête au module de *matching*, qui se chargera d'identifier à partir de la liste des services domaines ceux dont les fonctionnalités couvrent l'ensemble des méthodes instanciées dans le Goal Template. Le module de *matching* se base sur la fonction appelée *SemanticMatch*. Cette dernière prend en paramètres deux arguments à savoir : l'ensemble des méthodes d'un service domaine appartenant à la communauté et l'ensemble des méthodes génériques instanciées par le client. La fonction *SemanticMatch*, comme son nom l'indique, réalise un *matching* sémantique entre les méthodes d'un service domaine et les méthodes souhaitées par le client et ceci en se basant sur les descriptions sémantiques des services domaines (déjà décrites dans la section 3.4.1).

SemanticMatch renvoie une valeur booléenne qui renseigne sur le succès ou l'échec du *matching* (voir Figure 3.21). Le résultat de cette étape est un ensemble de services domaines qui vont être transférés par la suite au service de découverte.

- **Étape 3. Identification des services domaines qui satisfont les paramètres de contexte instanciés (*Context Constraints*)**

Le service de découverte envoie une requête à l'évaluateur de contexte en lui transmettant les paramètres de contexte instanciés et les paramètres de contexte des services candidats. L'évaluateur de contexte se chargera à son tour de comparer les paramètres de contexte instanciés avec les paramètres de contexte de chaque service domaine. Il procède également à comparer le score de réputation voulu par le client et celui affiché par le service. Pour ce faire, il se base sur le *Context Matching Template* (CMT), qui est une structure utilisée pour comparer les paramètres de contexte des services. En effet, l'évaluateur de contexte détient un ensemble de *Context Matching Templates* qui sont stockés dans une base de données appelé *CMT base*. Le CMT propose des règles qui permettent de comparer deux *Context Constraints* et de déterminer par la suite s'ils sont équivalents ou pas. Il existe autant de CMT que de catégories de contexte. Par exemple, nous pouvons identifier un CMT qui permet de comparer si le temps de réponse exigé par le client est inférieur ou égal au temps de réponse proposé par un service domaine. Si c'est le cas, le CMT retourne vrai. La structure d'une CMT est présenté dans la Figure suivante.

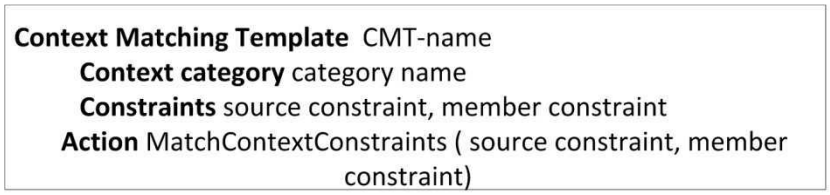


Figure 3.19. Structure d'un *Context Matching Template* (CMT)

Le CMT contient un nom ainsi qu'une catégorie. Cette dernière désigne le paramètre de contexte en question. Par exemple, on peut avoir le CMT dont la catégorie est « Prix ». Les différentes catégories des CMT font référence à l'ontologie de catégorisation de contexte (voir Figure 3.9). Comme nous l'avons déjà souligné, le CMT propose des règles pour comparer les différentes *Context Constraints* et ceci grâce à la fonction de *matching* appelée *matchContextConstraints*. Cette fonction est dépendante du paramètre de contexte en question. En effet, chaque catégorie de contexte peut avoir son propre traitement. Par exemple, dans certains cas cette fonction peut utiliser une comparaison syntaxique entre deux valeurs, alors que dans d'autres cas, plus complexes, elle nécessite certaines transformations à l'aide des règles.

Le processus de découverte des services que nous venons de présenter se base sur l'algorithme suivant qui résume les différentes étapes déjà décrites.

```
Algo Service Domaine Discovery
Input:  (Goal Template)// a Goal Template description
Output: {S1, S2, S3, S4, ..} // set of service domaines that meet the Goal Template
description

Funct

DetectCommunityIdentifier (Goal Template)
  Return CommunityIdentifier
GetInstanciatedGenericMethod (Goal Template)
   Return InstanciatedGenericMethodList  // igmList
GetInstanciatedGenericContext (Goal Template)
   Return instanciatedGenericContext // IgxConstraintList
GetContextConstraint ( ServiceID, CooperationRegistry)


Begin
For each Sk  in C  do // C is the community having as the identifier
CommunityIdentifier
  if ∀ IGMethodi ∈  igmList, ∃Methodj  ∈ Sk  such as
        SemanticMatch(Methodj, IGMethodi )== True
  then  PrimaryList.add ( Sk )
  End if
   if PrimaryList.length==0
   then NoServiceException ()
   else
      RepScores= GetReputationScores(PrimaryList)
      For each IgCxi ∈  IgxConstraintList  do
        if  Category (IgCxi)== Reputation
        then  CustRep= IgCxi
        Break
        End if
        For each Sk  in PrimaryList do
         If RepScore[Sk] ≥ CustRep
         then  SecondaryList.add ( Sk )
         End if
         if SecondaryList.length==0
         then NoServiceException ()
         else
               For each Sk  in SecondaryList do
               CxConstraintList[]= GetContextConstraint(Sk-ID,CooperationRegistry)
               if ∀IgCxi ∈ IgxConstraintList/CustRep,∃ CxCj  ∈  CxConstraintList such as
                  MatchContextConstraints(IgCxi, CxCj)== true
                 then FinalList.add ( Sk )
               End if
                   if FinalList.length==0
                      then NoServiceException ()
                      Else
                        Return  FinalList
                 End if
    End
```

Figure 3.20. Algorithme de découverte des services domaines

```
Funct SemanticMatch ( Methodj ,IGMethodi)
Input: Methodj a method belonging to a service domaine Sk
       IGMethodi a generic instancied method
Output: Match -Boolean
Begin
 match ← False
 If Subsume (Methodj ,IGMethodi) then
 match ←True
 Return match
End
```

Figure 3.21. Fonction de *matching* sémantique

3.5 Conclusion

Nous avons présenté dans ce chapitre une architecture de coopération à la demande et ceci en utilisant le paradigme de composition de services domaines.

L'architecture de coopération que nous avons présentée a été décrite grâce à un ensemble de méta-modèles dont l'objectif est de représenter une description uniforme des concepts manipulés (cf. section 3.2). Ces concepts décrivent clairement l'organisation de la coopération à la demande, ses entités principales (les communautés, le registre de coopération, *etc.*) et les *constructs* du processus coopératif (notion de Goal Template, branchement, *etc.*).

Notre architecture de coopération comprend cinq principaux types de services de coopération : le service de description, le service de publication, le service de découverte, le service dédié à la substitution et le service dédié aux aspects de sécurité (cf. section 3.3). Le service de description permet de décrire sémantiquement et contextuellement les services domaines (cf. section 3.4.1). Le service de publication permet d'effectuer la publication sémantique, contextuelle (et syntaxique) des services domaines (cf. section 3.4.2). Le service de découverte permet d'effectuer des recherches sémantiques ainsi que contextuelles des services (cf. section 3.4.3). Et enfin le service dédié à la substitution et le service dédié aux aspects de sécurité permettent de gérer respectivement la substitution et la sécurité au sein du processus coopératif.

En outre, le chapitre a permis de présenter une démarche de construction du processus coopératif en se basant sur une composition semi-automatique des services domaines. Cette démarche utilise essentiellement la notion de Goal Template qui remplace la requête classique du client, en intégrant à la fois les besoins fonctionnels et les besoins contextuels du client.

La démarche proposée comble certaines limites et répond à certains questionnements concernant les technologies des services Web. Précisément, notre démarche a proposé un enrichissement du mécanisme de description, publication et de découverte de services en exploitant les informations contextuelles dans un cadre de coopération dynamique. Enfin, notre démarche intègre la flexibilité dans la mise en œuvre du processus coopératif à la demande du fait qu'elle repose sur le principe d'adaptation au contexte du client ainsi que de la coopération lors de la découverte des services. Ce principe sera détaillé sous un autre angle (adaptation contextuelle de la logique métier des services) dans le prochain chapitre de cette thèse.

CHAPITRE 4

Adaptation des services domaines au contexte à base de tissage d'aspects

Table des matières

Les deux chapitres précédents (chapitres 2 et 3) ont permis de répondre respectivement à la problématique de construction de l'architecture de services au sein de l'entreprise (cf. introduction générale, section 4.1) et à la problématique de construction du processus coopératif à base de composition des services (cf. introduction générale, section 4.2) permettant d'utiliser les services d'entreprise dans des scénarios de coopération à la demande.

Pour garantir la flexibilité des services participant à des coopérations, nous allons nous intéresser à présent à la problématique d'adaptation des services domaines qui a pour but de répondre à la question d'adaptation dynamique des services au contexte.

Rappelons que l'adaptation des services se base sur la modélisation des variantes d'usage des services (Aspects Conceptuels) définies au niveau du chapitre 2 ainsi que des particularités relatives au processus coopératif (ontologie de contexte, exposée au chapitre 3).

L'intérêt principal de ce chapitre est qu'il permet de proposer une approche d'adaptation dynamique des services domaines. L'approche d'adaptation exploite les avantages proposés par la programmation orientée aspect et se base essentiellement sur le paradigme de tissage d'aspects.

Aussi dans ce chapitre, nous décrirons tout d'abord les objectifs que nous envisageons pour notre approche d'adaptation du service domaine. La section 4.2 présentera une architecture générale du service domaine en mettant l'accent sur les différents modules qui le composent. La section 4.3 exposera une analyse fine de l'adaptation de la logique métier ainsi que les motivations liées à l'utilisation de la programmation orienté aspect. La section 4.4 décrira l'approche d'adaptation à base de tissage d'aspects qui sera ensuite mise en œuvre dans la section 4.5. Un scénario d'adaptation et une validation feront l'objet de la dernière section de ce chapitre.

4.1 Adaptation du service domaine aux changements de contexte : constats

L'adaptation des services domaines au contexte demeure essentielle afin de mieux les exploiter dans des scénarios de coopération à la demande. Autrement dit, garantir à ces services un niveau d'adaptabilité important, pour rendre les services toujours exploitables même dans des environnements en perpétuel changement. Cette adaptation devient de plus en plus exigeante en termes de flexibilité, d'automatisme et de rapidité d'autant plus que l'adaptation manuelle des services reste peu efficace et entraîne beaucoup de problèmes surtout en termes de disponibilité du service à adapter. En effet, la pratique montre qu'à chaque modification de contexte, le fournisseur du service est contraint d'arrêter le service, de le modifier et de le redéployer. Pendant ce temps, le service reste indisponible pour ses utilisateurs. Par ailleurs, vu que le service est partagé entre plusieurs applications clientes, si un changement affecte la description du service (par exemple son document WSDL : *Web Service Description Language*), les clients ne pourront plus interagir avec celui-ci et auront des erreurs d'exécution. Il est par conséquent primordial d'avoir une approche permettant une adaptation dynamique des services sans affecter leurs disponibilités et leurs fonctionnalités.

L'objectif de notre travail consiste à proposer une architecture d'adaptation dynamique du service domaine à de nouveaux contextes d'utilisation. Suite à l'apparition de nouveaux événements : une situation contextuelle SC_i à un instant t peut basculer vers une autre situation contextuelle SC_j à un instant t+1. Par conséquent, le service domaine doit s'adapter à ce changement. Nous définissons l'adaptation du service domaine à une situation contextuelle par les mécanismes et les transformations indispensables pour modifier son comportement afin qu'il puisse être utilisé de façon efficace dans l'opportunité de coopération décrite par la situation contextuelle. L'adaptation visée dans le cadre de notre travail de recherche est définie par deux qualificatifs :

- **Adaptation adaptative**

Même si le service domaine s'exécute correctement parfois son environnement (son contexte, le contexte du client ou même de la coopération) change. Dans ce cas le service doit être adapté dynamiquement en réponse aux changements affectant son environnement [Ketfi et al., 2002].

- **Adaptation évolutive**

Au moment du développement du service domaine, certaines fonctionnalités sensibles au contexte ne sont pas prises en compte. Avec l'évolution des besoins des clients ou encore du fournisseur du service, le service doit être étendu avec de nouvelles fonctionnalités. Cette extension doit être réalisée sans mettre en péril le fonctionnement du service.

Ces deux qualificatifs assurent d'une part une délimitation du champ d'études étant donné que le domaine d'adaptation des services est un domaine assez vaste, et d'autre part, elles contraignent les techniques d'adaptation à mettre en œuvre au sein du service domaine.

4.2 Architecture générale du service domaine

Les architectures présentées dans les travaux existants pour assurer la sensibilité au contexte accordent une très grande importance à la gestion du contexte sans toutefois présenter comment modifier le comportement d'une application pour qu'elle s'adapte au contexte. Dans notre approche d'adaptation, nous accordons une importance capitale à la fois à la gestion du contexte et à l'adaptation. La gestion de contexte permet la capture, le stockage et la dissémination du contexte. Tandis que l'adaptation, comme son nom l'indique, se préoccupe d'assurer l'adaptation du service domaine aux diverses situations contextuelles. Comme nous l'avons signalé auparavant dans le chapitre 2, le service domaine a été conçu principalement pour être utilisé dans des scénarios de coopération à la demande. Il orchestre un ensemble de services fonctionnels et présente un comportement adaptable en fonction de son contexte d'utilisation. Pour ce faire, nous définissons une architecture particulière qui le distingue d'un service composite ordinaire (voir Figure 4.1). Cette architecture repose sur un ensemble de modules qui concourent à réaliser son objectif et à satisfaire ses caractéristiques et notamment celle de l'adaptation en fonction de son contexte d'utilisation. Ces modules sont *Entry Module*, *Service Orchestration Module*, *Aspect Activator Module*, *Context Manager Module*. Ces derniers sont décrits dans ce qui suit.

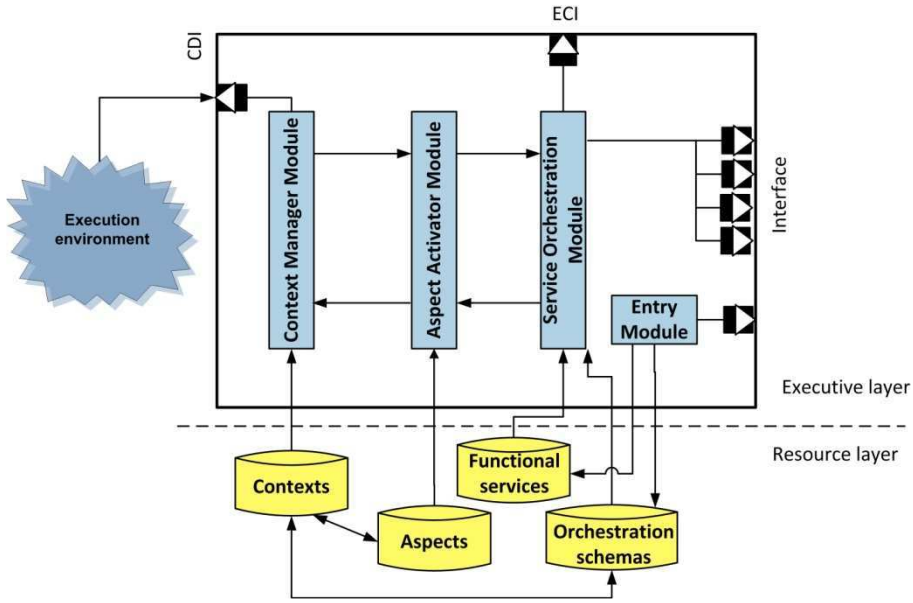


Figure 4.1. Architecture générale du service domaine [Boukadi et al., 2008b]

4.2.1 Gestionnaire de contexte (CMM : *Context Manager Module*)

À l'encontre des gestionnaires de contexte dans les applications mobiles où ils sont définis comme des composants logiciels liés à un ou plusieurs dispositifs ou capteurs, le gestionnaire de contexte que nous proposons est un module au sein du service domaine qui permet de collecter les paramètres de contexte au moment de la sélection du service (c'est-à-dire quand le service domaine est sélectionné par le client). Les paramètres du contexte collectés (relatifs au client et à l'opportunité de coopération) sont stockés dans un registre appelé registre de contexte. Afin d'enregistrer les paramètres de contexte, nous adoptons le modèle de représentation (modèle multi-niveaux) proposé dans le chapitre précédent (cf. section 3.4.1).

En outre, le gestionnaire de contexte détecte les changements dynamiques des paramètres de contexte qui surviennent au moment de l'exécution du service. Par exemple, l'identité de l'entreprise cliente est statique durant l'interaction avec le service. Par contre, les préférences concernant le mode de paiement peuvent varier au cours de l'interaction avec le service.

La détection du changement de contexte est réalisée grâce à l'interface appelée *Context Detection Interface* (CDI). Par la suite, il enregistre les nouvelles valeurs de paramètres de contexte dans le registre de contexte. Les nouvelles valeurs stockées sont utilisées dans le processus d'adaptation du service domaine. Ce processus est détaillé dans la suite de ce chapitre.

Outre les fonctions de capture du contexte, le gestionnaire de contexte assure la communication avec les modules d'adaptation (décrits dans la suite de cette section). Il les informe des informations contextuelles et des changements éventuels du contexte.

4.2.2 Module d'orchestration de services (*SOM : Service Orchestration Module*)

Le SOM est responsable de l'exécution des schémas d'orchestration au sein d'un service domaine. Pour ce faire, il se base sur le langage BPEL en tant que langage de composition des services Web [Andrews et al., 2003a]. Le SOM présente une vue externe aux consommateurs du service. Cette vue est appelée ECI (*Execution Control Interface*). En effet, à l'encontre des services classiques qui suivent une approche de type boite noire, le service domaine permet aux clients de suivre son état d'exécution. Cette dernière caractéristique est importante dans le cadre d'une coopération interentreprises. Elle permet aux clients non seulement d'avoir une vision globale sur le service domaine et notamment sur les services fonctionnels qu'il orchestre mais aussi de contrôler leurs exécutions si nécessaire. Le contrôle agit au niveau du processus et/ou sous processus encapsulé par le service domaine en question. Le client peut influencer l'exécution du service domaine et ceci via des commandes de type « *AnnulerProcessus* » ou encore « *PauseProcessus* » et « *ContinuerProcessus* ». Par exemple, une entreprise qui sélectionne un service de livraison peut à un instant *t*, connaître l'état d'avancement des marchandises à livrer. De plus, elle peut aussi arrêter l'exécution ou encore faire une pause et reprendre l'exécution du processus encapsulé par le service de livraison. Il est important de souligner que la vue proposée par l'ECI reste une vue restreinte, avec un niveau d'abstraction suffisant pour que le savoir-faire des entreprises ne soit pas divulgué aux partenaires.

4.2.3 Modules d'adaptation du service domaine

Les modules d'adaptation permettent la mise à jour et la réactualisation automatique du service domaine à de nouveaux contextes d'utilisation. Cette adaptation touche la logique d'orchestration ainsi que la logique métier du service. Nous dédions un module d'adaptation à chaque logique d'adaptation. Les deux modules d'adaptation du service domaine se basent sur le gestionnaire de contexte pour réaliser les adaptations du service. Nous détaillons dans ce qui suit ces deux modules.

- **Module d'adaptation de la logique d'orchestration (*Entry Module*)**

Le module d'adaptation de la logique d'orchestration (*Entry Module)* permet de gérer les différents schémas d'orchestration et les instances des services fonctionnels appartenant à un service domaine. Ces schémas d'orchestration ont été déjà étudiés et analysés dans le cadre du chapitre 2 de ce manuscrit.

Ce module s'appuie sur le gestionnaire de contexte afin de récupérer les paramètres de contexte du client et de la coopération et de sélectionner par la suite le schéma d'orchestration qui répond au mieux à la situation contextuelle. Dans la littérature plusieurs travaux présentent des solutions pertinentes pour l'adaptation de la logique d'orchestration d'un service [Agarwal et al., 2005], [Chafle et al., 2007], [Qiu et al., 2007]. Étant donnée sa pertinence, nous nous sommes basés dans le cadre de cette thèse sur l'approche développée par [Qiu et al., 2007] afin d'assurer l'adaptation de la logique d'orchestration du service domaine. Ce travail a été déjà décrit en détail dans le chapitre de l'état de l'art (cf. section 1.3.5.3).

- **Module d'adaptation de la logique métier du service domaine (*Aspect Activator Module*)**

Comme nous l'avons déjà mentionné auparavant, nous employons le standard BPEL pour l'orchestration des différents services fonctionnels au sein d'un service domaine. Ainsi, chaque service domaine est implémenté grâce à un processus BPEL. Afin d'assurer l'adaptation du service domaine aux différents contextes d'utilisation, nous utilisons la programmation orientée aspect. Rappelons que la programmation orientée aspect est un nouveau paradigme de programmation qui permet de séparer l'implémentation des considérations techniques (ou non fonctionnelles) des descriptions métier (ou fonctionnelles) d'un système. Ce paradigme est complémentaire aux langages objets et se déploie progressivement dans les entreprises. Son principe de base étant de définir chaque préoccupation séparément et de définir leurs règles d'intégration afin de les combiner pour former le système final (cf. chapitre 1, section 1.3.4.3.3). Nous tirons profit de ce type de programmation afin d'étendre et d'adapter le comportement du service domaine en fonction des changements contextuels, ceci sans modifier son code d'origine. Nous présenterons dans la suite de ce chapitre les motivations qui nous ont poussé à considérer ce paradigme de programmation comme une solution pour l'adaptation des services.

Afin d'exploiter la programmation orientée aspect pour gérer l'adaptation de la logique métier du service, certaines considérations doivent être prises en compte :

- La sélection des aspects en fonction des changements contextuels. En effet, en fonction de la situation contextuelle détectée, le ou les aspects adéquats doivent être déclenchés.
- L'activation dynamique des différents aspects au moment de l'exécution du service domaine. En effet, les Aspects Conceptuels déjà identifiés (cf. chapitre 2, section 2.4.2) vont être implémentés sous formes d'aspects au sens de la programmation orientée aspect. Leur activation doit être transparente au client.

L'adaptation de la logique métier est assurée grâce au module d'activation d'aspect appelé en anglais *Aspect Activator Module* qui travaille en concert avec le gestionnaire de contexte. En effet, le module d'activation d'aspect est responsable de la sélection de l'aspect à partir du registre d'aspect. L'aspect sélectionné doit contenir l'action d'adaptation correspondant au changement de contexte. Par la suite, cet aspect est tissé au sein du service à adapter. Nous revenons sur les détails techniques de ce module dans la section 4.5.2 de ce chapitre.

4.3 Analyse de l'adaptation de la logique métier des services

Au cours de l'exécution du service, le contexte du client, de la coopération ou encore du service lui-même peuvent être altérés. Le besoin d'adapter le service à ce genre de changement devient crucial pour garantir sa flexibilité.

Le standard BPEL a été retenu pour implémenter les services domaines. Ainsi, assurer l'adaptation d'un service domaine revient à examiner la capacité du langage BPEL à intégrer et à gérer les comportements sensibles au contexte. Dans la suite de cette section, nous allons examiner tout d'abord le langage BPEL et sa capacité à gérer les changements de contexte. Ensuite, nous allons argumenter notre choix pour la programmation orientée aspect afin

d'assurer l'adaptation dynamique des services. Enfin, nous allons rappeler les deux travaux pertinents qui utilisent la technologie aspect pour un meilleur positionnement de notre contribution.

4.3.1 BPEL et les changements de contexte

Comme nous l'avons déjà cité dans l'état de l'art, le standard BPEL est le standard de fait pour l'orchestration des services Web. BPEL est un langage de spécification basé sur XML dont l'objectif est la modélisation du comportement des services dans les interactions au sein d'un processus intra ou interentreprises.

Le langage BPEL présente plusieurs avantages par rapport à ses prédécesseurs et s'adapte partiellement à un contexte de coopération interentreprises. Tout d'abord, la spécification, ayant été travaillée par plusieurs acteurs industriels ne présente pas d'ambiguïté, contrairement aux langages dit de première génération tels que WSFL (*Web Services Flow Language*) et XLANG (*XML Business Process Language*). Ainsi, les objectifs sont donc fixés de manière claire et précise et chaque élément appartenant au langage est bien détaillé. En outre, il possède un bon pouvoir d'expression (des notions d'agrégation, de branchement, de concurrence, d'itération, de compensation et de contraintes de temps). Ces avantages ne nient pas le fait que ce langage présente aussi des limites. En particulier, deux limites majeures de BPEL par rapport à notre problématique d'adaptation des services, celles-ci sont d'ailleurs partagées par les différents langages de description de la composition des services Web tels que : WSCL (*Service Choreography Language*) et BPML *(Business Process Modelling Language*).

La première limite, c'est qu'il ne prend pas en compte la séparation des préoccupations au sein d'un processus. Ainsi, certaines préoccupations non fonctionnelles recoupent l'implémentation des préoccupations relatives à la logique métier. Par conséquent, un processus BPEL va contenir des éléments de préoccupations de nature différente. Un exemple de préoccupation non fonctionnelle peut être une activité de sécurité qui va être insérée au sein du code métier présenté par le processus.

La deuxième limite tient au fait que BPEL s'attarde sur le niveau descriptif de l'orchestration sans offrir des mécanismes supports à l'adaptation dynamique du service aux changements de contexte. En effet, le standard BPEL trouve ses origines dans les travaux relatifs aux systèmes de gestion de Workflows et hérite par conséquent de la vue statique supportée par cette approche. Tout comme les systèmes de gestion de Workflow, le standard BPEL présente une description figée d'un processus, ce qui ne lui permet pas de supporter des évolutions en temps réel sans arrêter le processus en cours. Cette limite affecte à la fois les fournisseurs et les clients de services.

Ces deux limites sont fortement reliées l'une à l'autre. L'incapacité de BPEL à soutenir la séparation des préoccupations engendre une difficulté pour définir, modéliser et gérer des comportements sensibles au contexte. Généralement, ces comportements sont des comportements susceptibles d'évoluer et ne méritent pas d'être encapsulés avec la logique métier du processus, qualifiée de stable pour une certaine période de temps.

4.3.2 Motivations liées à l'utilisation de la programmation orientée aspect

Dans le cadre de notre travail, nous tirons profit des outils et des mécanismes proposés par la programmation orientée aspect pour étendre et adapter le comportement du service domaine en fonction des changements contextuels.

Rappelons que les concepts de base de la programmation orientée aspect sont : l'aspect, le greffon (*advice*), les points de coupure ou d'action (*pointcuts*), les points de jonction (*join points*) et le tisseur (*weaver*). Ces concepts ont été déjà détaillés (cf. chapitre 1, section 1.3.4.6).

L'utilisation de la programmation orientée aspect pour l'adaptation dynamique des services peut être justifiée par plusieurs arguments. En effet, le principe de base de la programmation orientée aspect étant de séparer l'implémentation des exigences fonctionnelles des exigences non fonctionnelles d'une application, il implique donc de coder chaque problématique séparément et de définir leurs règles d'intégration. Dès lors, nous considérons que le principe de séparation des préoccupations est important pour la gestion des informations contextuelles et des actions d'adaptation qui lui sont relatives. En effet, la séparation entre les considérations métier contenues dans le processus BPEL et les considérations contextuelles reste fondamentale pour assurer la flexibilité et l'évolutivité des services. Par conséquent, la modification des actions d'adaptation ou des considérations techniques ne doit pas engendrer la mise en cause de la logique métier du service. Le principe de réutilisation est assuré grâce à la modularité supportée par la programmation orientée aspect. Ainsi, les différents aspects identifiés et implémentés pour un service peuvent être utilisés dans d'autres services domaines. Par ailleurs, grâce au principe de tissage dynamique (ou tissage à l'exécution) les aspects peuvent être activés ou désactivés au moment de l'exécution du service. Ainsi, l'adaptation des services est assurée en tissant des aspects au sein des différents services fonctionnels.

En adoptant la programmation orientée aspect pour gérer les actions d'adaptation et les considérations techniques ou de sécurité, on garantit la flexibilité requise par les services et ceci par le biais d'ajustements rapides et faciles à exécuter. Néanmoins, dans la littérature la programmation orientée aspect, se concentre essentiellement sur des extensions de bas niveau tel que l'ajout des préoccupations de sécurité ou d'authentification à des programmes déjà existants.

4.3.3 Rappel des travaux précédents et positionnement

L'état de l'art relatif à l'adaptation des services Web au contexte (cf. chapitre 1, section 1.3.5) a montré que l'adaptation des services en utilisant la programmation orientée aspect n'a pas été prise en compte dans son intégralité dans la littérature. Cette orientation de recherche constitue un domaine qui demande une exploration plus approfondie. Quelques travaux dans cette direction ont été présentés par [Charfi and Mezini, 2007], [Hmida et al., 2007].

Le travail de [Charfi and Mezini, 2007] démontre que le concept de programmation par aspect est non seulement applicable aux processus métiers, mais qu'il est également approprié pour les rendre facilement adaptables. Cependant, le travail proposé requiert un moteur BPEL

spécifique, ce qui en fait une solution inexploitable par les autres plateformes. Par ailleurs l'utilisation d'aspects modularisant du code BPEL est limitée du seul fait que le langage BPEL n'est pas un langage généraliste. Par conséquent, il n'est pas possible de rajouter de nouvelles préoccupations excepté en développant de nouveaux services Web d'infrastructure (*Web Service Infrastructure*) qui intègreront ces préoccupations.

[Hmida et al., 2007] ont traité les problématiques de l'adaptation dynamique du service ainsi que l'interfaçage entre le client et le service. Le travail de [Hmida et al., 2007] intègre les *join points* sous la forme de Xpath dans le WSDL des services Web. Cette approche peut affecter la flexibilité et surtout la disponibilité des services Web à adapter. C'est-à-dire à chaque modification ou ajout de *join points*, le fournisseur des services est obligé d'arrêter le service, de modifier le fichier de description WSDL et de le redéployer.

Néamoins, une limite partagée entre ces deux travaux tient au fait que les adaptations visées sont qualifiées de bas niveau. Le travail de [Charfi and Mezini, 2007] se préoccupe essentiellement de l'ajout des informations de sécurité et d'authentification dans les processus BPEL existants tandis que le travail de [Hmida et al., 2007] se focalise sur la génération automatique du client qui prend en compte les changements dynamiques du service. Nous allons essayer dans notre travail d'adaptation de dépasser ces différentes limites.

4.4 Adaptation du service domaine à base de tissage d'aspects

Le service domaine propose un ensemble de services fonctionnels orchestrés. Certains services ou opérations de ces services doivent être adaptés pour répondre à une situation contextuelle ou encore à tout changement de contexte. Dans la suite, nous allons présenter le modèle d'orchestration adaptable que nous proposons. Ensuite, nous allons proposer une anatomie d'un aspect dédié à l'adaptation de la logique métier d'un service.

4.4.1 Modèle d'orchestration adaptable des services

Afin de résoudre la problématique d'adaptation des services domaines aux changements de contexte nous présentons dans la suite de cette section le modèle formel que nous proposons pour l'orchestration des services. Ce modèle décrit dans [Boukadi et al., 2008b], propose des mécanismes indispensables pour modifier le comportement du service domaine afin qu'il puisse être utilisé de façon efficace durant l'opportunité de coopération décrite par la situation contextuelle courante. En outre, le modèle d'orchestration répond parfaitement aux qualificatifs de l'adaptation telle qu'ils ont été décrits dans la première section de ce chapitre à savoir : adaptative et évolutive.

Le cœur de notre approche d'adaptation de la logique métier du service domaine consiste à l'injection dans le processus BPEL des aspects et ceci en fonction des changements contextuels. Notre contribution consiste à l'encapsulation des actions d'adaptation en un ensemble d'aspects. Comme exemples d'outils les plus utilisés pour la programmation orientée aspect nous pouvons citer : *AspectJ* [Kiczales et al., 2001a], *Aspect Werkz* [AspectWerkz, 2008], *Jboss AOP* [JbossAOP, 2008]. Nous nous attardons sur les détails techniques de l'implémentation des aspects dans la section 6 de ce chapitre.

Le modèle d'orchestration adaptable utilise deux modules déjà décrits à savoir : le gestionnaire de contexte (*Context Manager Module*) et le module d'activation d'aspect (l'*Aspect Activator Module*). Ces deux modules se préoccupent respectivement de la gestion du contexte et de l'activation ainsi que de l'injection de l'aspect adéquat qui correspond au changement de contexte.

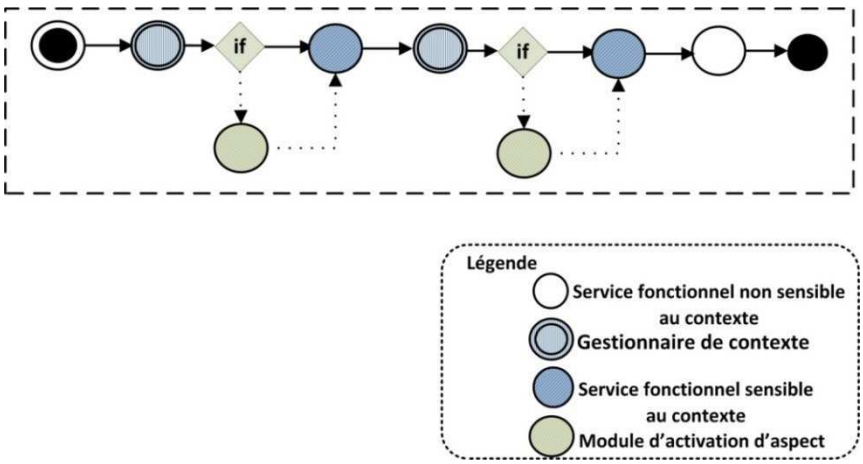


Figure 4.2. Modèle d'orchestration adaptable

Nous proposons dans ce qui suit un ensemble de définitions formelles afin de décrire le modèle d'orchestration adaptable que nous proposons.

Formellement, nous définissons le modèle d'orchestration implémentant un service domaine comme suit :
Soit un graphe **G (V,E)** , un graphe dirigé acyclique. Chaque vi ∈ V est un service Web et E désigne l'ensemble des arêtes.
Au sein du graphe G on définit trois types de services : les services sensibles au contexte, les services non sensibles au contexte et les sommets relatifs au gestionnaire de contexte et au module d'activation d'aspect (voir Figure 4.2).

Un service fonctionnel sensible au contexte (**C**ontext-aware **F**unctional **S**ervice) est défini formellement comme suit :
CFS={<ID-CFS, Ctx, Asp> } avec ID-CFS est l'identifiant du service fonctionnel, Ctx est le nom du paramètre de contexte, et Asp est l'identifiant de l'aspect relatif à ce paramètre de contexte. Ainsi un service fonctionnel sensible au contexte peut avoir plusieurs comportements possibles en fonction des situations contextuelles. Ces comportements sont encapsulés par un ensemble d'aspects qui lui sont attribués dès sa phase de modélisation (démarche CSOMA).

Un aspect dédié à l'adaptation est défini formellement comme suit :
Asp=< ID-Asp, pointcut, advice> avec ID-Asp est l'identifiant de l'aspect, le *pointcut* qui caractérise un ensemble de *join points* annotés dans le code du service (que ce soit du service domaine ou des services fonctionnels) et l'*advice* qui se réfère au code à exécuter à l'endroit du *pointcut*.

4.4.2 Anatomie d'un aspect dédié à l'adaptation

Dans notre travail, les aspects permettent d'étendre le comportement d'un service en fonction des informations contextuelles et ceci sans modifier son code d'origine. La figure suivante illustre les mécanismes de base de la programmation orientée aspect appliquée à la technologie service Web.

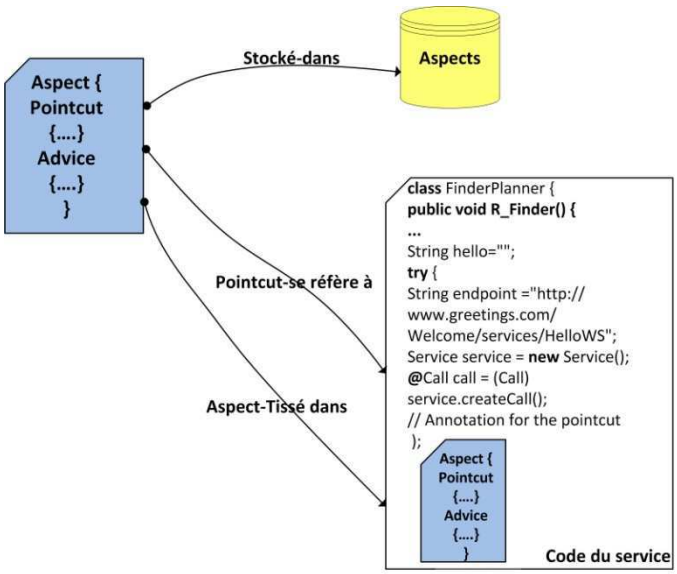


Figure 4.3. Mécanismes de la programmation orientée aspect appliqués au service Web

Dans la suite, nous présentons une description détaillée des concepts relatifs à un aspect dédié à l'adaptation d'un service domaine à savoir : les *pointcuts*, les *advices* et le tissage (*weaving*).

- **Les *pointcuts***

Afin de déterminer les couples (*pointcut*, *advice*), notre réflexion porte sur le processus d'adaptation. Ce dernier est dynamique dans le sens où chaque changement significatif (*i.e.* un changement des paramètres de contexte) qui se produit dans l'environnement doit être géré par une décision d'adaptation suivie par la réalisation de cette décision. Cette vision possède un point de ressemblance avec l'approche ECA (Événement-Condition-Action) issu du domaine des bases de données actives [Collet, 1996]. Ainsi, les *join points* de la coupe peuvent être considérés comme des événements relatifs à l'exécution du service (plus spécialement l'invocation d'opération de service) (voir Figure 4.3). Dans cette approche, le rôle du tisseur est de coordonner l'exécution du code de base et des *advices* proposées par les aspects par rapport à l'occurrence de ces événements. Ces événements liés à l'exécution du service lui-même ne sont pas aptes pour déclencher l'adaptation du service domaine telle que nous l'envisageons.

Un service domaine adaptable doit en effet être capable de réagir aux évolutions du contexte du client ou de la coopération comme par exemple le changement de la date de la demande de livraison souhaitée par le client, *etc*. Quoique les événements qui correspondent aux changements des informations de contexte (ou les situations relatives au contexte courant) ont une origine différente par rapport aux événements reliés au service, nous pensons que ces événements peuvent aussi être considérés comme des *join points*, puisqu'ils déclenchent

l'exécution des actions d'adaptation. Notre contribution pour les *pointcuts* de l'aspect d'adaptation est donc l'extension du domaine des *join points* à l'ensemble des paramètres de contexte auquel le service domaine est sensible. Les événements liés à l'exécution du service correspondent ainsi aux *join points* ordinaires, alors que les événements liés au contexte du service, du client et de la coopération constituent un nouveau type de *join points*. Ce nouveau type de *join point* offre au service la possibilité de s'adapter aux évolutions du contexte, enrichissant par là même le pouvoir d'expression des aspects.

- **Les *advices***

Les *advices* sont déclenchées par la détection d'un *join point*, elles vont correspondre aux modifications à mettre en œuvre pour adapter le comportement du service domaine pour répondre à une situation contextuelle donnée. Une *advice* définit la ou les actions d'adaptation à effectuer sur le service en question. Une *advice* peut être exécutée avant, après ou en remplacement des fonctionnalités du service en question. Elle se compose généralement de la forme si l'expression <condition > est vraie, alors l'adaptation <action> est exécutée, adaptant ainsi le service domaine à la nouvelle situation contextuelle issue de l'occurrence de l'événement.

Nous modélisons une *advice* par une paire *advice* = (expressions, actions). La partie « expressions » décrit une situation contextuelle. Elle est défini par un ensemble d'expressions logiques basées sur des opérateurs logiques simples comme « *equal* », « *superior* », « *inferior* », « *exist* ». Ces expressions concernent les valeurs de paramètres de contexte auxquelles le service est sensible. Elles sont représentées suivant le modèle de contexte présenté dans le chapitre précédent (*Context Constraint*) (cf. chapitre 3, section 3.4.1). La partie « *actions* » décrit une liste d'actions à effectuer lorsque l'évaluation des expressions est vraie. Chaque action correspond à une adaptation particulière.

- **Le *weaving***

L'intérêt que nous portons pour l'adaptation à la volée, nous amène à penser que le tissage des aspects d'adaptation doit pouvoir être dynamique. Si le code de base des services est tissé statiquement avec les aspects d'adaptation, la logique métier devient inexploitable dans certains contextes qui n'ont pas été prévus. Nous allons donc choisir le tissage dynamique permettant un découplage entre le métier proposé par le service et les actions d'adaptation relatives aux changements de contexte. Ce qui nous imposera par la suite certains choix techniques concernant le serveur d'aspect ainsi que son tisseur.

4.5 Mise en œuvre de l'adaptation du service domaine à base de tissage d'aspects

Dans cette section, nous allons présenter en premier lieu une vue générale du processus d'adaptation. Par la suite, nous allons nous intéresser à l'architecture technique que nous proposons.

4.5.1 Vue générale du processus d'adaptation

L'adaptation du service domaine suit un processus à trois étapes (voir Figure 4.4) :

1. La détection en temps réel des informations contextuelles : le gestionnaire de contexte se charge de collecter pour chaque service fonctionnel sensible au contexte, les informations contextuelles qui le concernent. De plus, il détecte tout changement éventuel de contexte.
2. En fonction des informations contextuelles détectées par le gestionnaire de contexte, le module d'activation d'aspect se charge de sélectionner l'aspect à tisser. Ce dernier sera injecté au sein du service à adapter.
3. Récupération de l'exécution du processus BPEL en tenant compte des nouveaux changements.

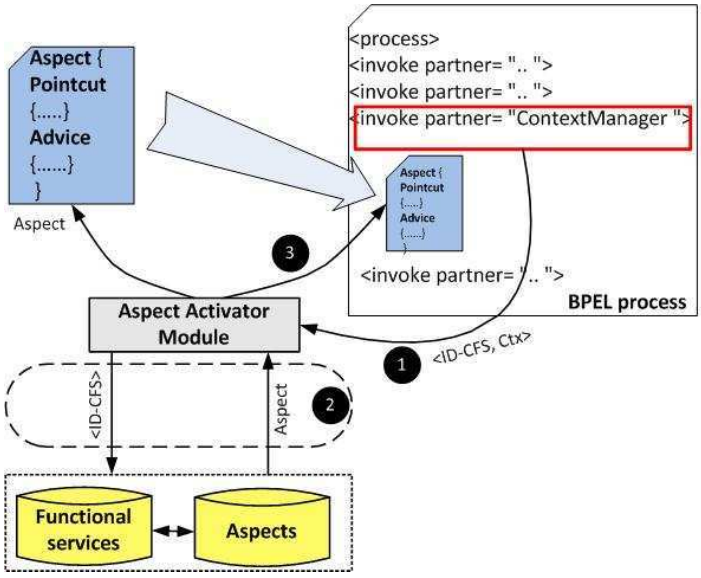


Figure 4.4. Vue générale du processus d'adaptation [Boukadi et al., 2008b]

Le processus d'adaptation sera décrit en détail dans la section suivante.

4.5.2 Architecture technique de l'adaptation

Afin de mettre en œuvre notre approche d'adaptation de la logique métier du service domaine à base de tissage d'aspects nous présentons dans cette section l'architecture technique que nous proposons. Cette dernière permet d'implémenter l'architecture du service domaine déjà exposée (cf. section 4.2).

L'architecture technique proposée est répartie en trois couches distinctes à savoir (voir Figure 4.5) : une couche applicative, une couche d'adaptation et une couche de ressources. Ces différentes couches sont décrites dans ce qui suit.

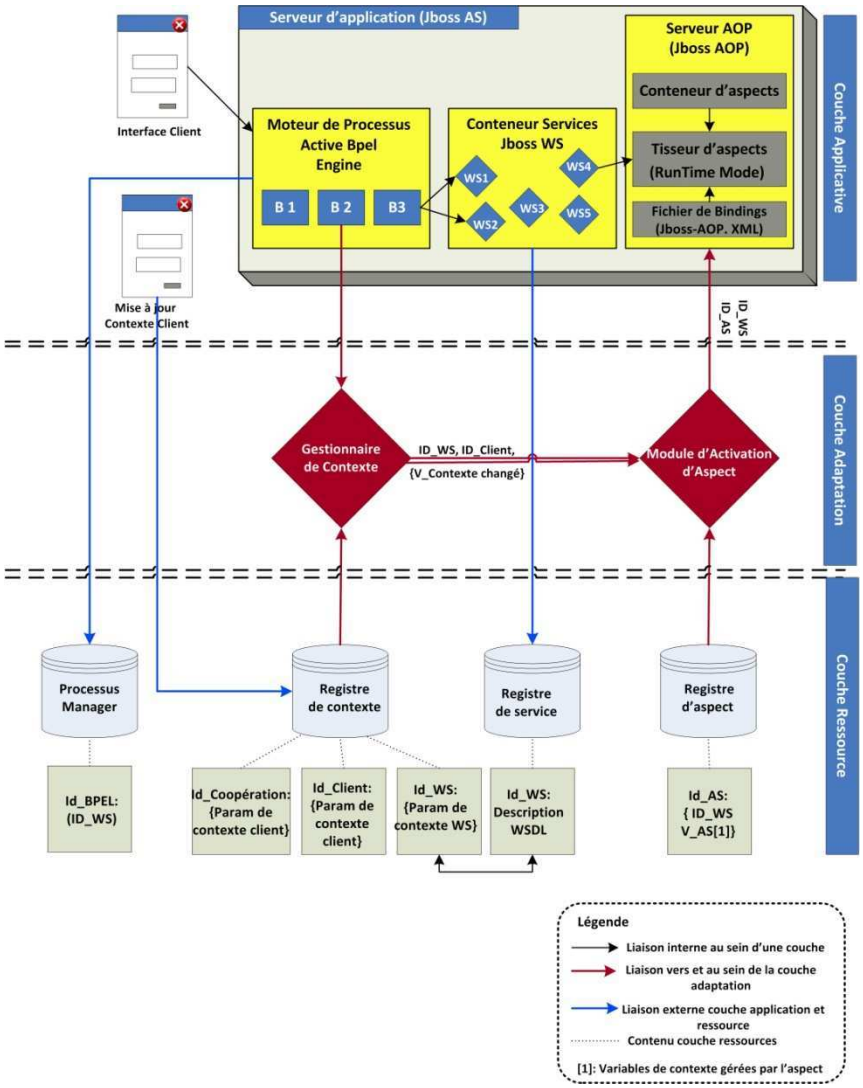


Figure 4.5. Architecture technique de l'adaptation du service domaine

- **La couche applicative**

C'est la couche supérieure de l'architecture technique incluant la plateforme applicative sur laquelle nous implémentons notre approche d'adaptation. L'élément central de cette couche est le serveur d'application. Généralement, un serveur d'applications est un serveur sur lequel sont installées les applications utilisées par les clients. Ces applications sont chargées sur le serveur d'applications, les clients y accèdent à distance. Pour notre travail d'adaptation par tissage d'aspects, le serveur d'applications doit permettre le déploiement des différents aspects relatifs aux services.

Afin de choisir le serveur d'applications adéquat nous avons étudié divers travaux dans la littérature tels que [Kersten, 2004] et [Kiczales et al., 2001a]. En se basant sur ces travaux, nous avons pu comparer les quatre serveurs d'applications qui supportent la programmation orientée aspect (*AspectJ*, *AspectWerkez*, *Jboss AOP* et *Spring AOP*) (cf. annexe C). Notre étude comparative a révélé que les serveurs d'applications fournissent des fonctionnalités semblables mais avec des techniques et des outils spécifiques à chacun d'entre eux. À l'issue

de cette étude comparative, la version 5 du serveur d'applications *Jboss* Application Server (*Jboss AS*) a été retenue. *Jboss AS* est un serveur d'applications libre entièrement écrit en Java, publié sous licence GNU LGPL[14]. Grâce à l'utilisation du langage Java, *Jboss AS* peut être utilisé sur tout système d'exploitation disposant d'une machine virtuelle Java (JVM). Cette version a été retenue pour trois raisons principales. La première tient au fait que *Jboss AS* dans sa version 5 permet le déploiement, l'exécution et aussi le tissage des aspects lors de l'exécution des services Web. En effet, grâce à son conteneur d'aspects (*Jboss AOP*) et à son tisseur, le serveur *Jboss AS* permet d'injecter les aspects dans le corps des services. Ce qui permet par conséquent de répondre à notre objectif d'adaptation à savoir l'adaptation dynamique des services. La deuxième raison porte sur le déploiement à chaud (déployer des aspects sans redémarrer le serveur). Cette caractéristique est très intéressante pour garantir la flexibilité ainsi que la facilité de la solution d'adaptation. La dernière raison concerne la stabilité de *Jboss AS* ainsi que son utilisation accrue dans le milieu industriel.

Outre le serveur d'applications, la couche applicative doit inclure le conteneur des différents services Web. En effet, afin de déployer et d'exécuter les services Web, toute architecture orientée service doit avoir dans sa partie applicative un conteneur (moteur) de services Web. Ce dernier assure le bon déploiement et l'exécution des services Web. *Axis2* est connu comme le moteur de services Web le plus utilisé par les plateformes orientées services et peut être ajouté à notre serveur d'applications choisi (*Jboss AS* Version 5). Cependant, afin d'éviter tout problème de compatibilité entre le moteur de service Web et le serveur d'applications, nous avons choisi le moteur fourni avec le serveur d'application *Jboss AS* nommé *Jboss SW*.

Outre les serveurs déjà cités et afin de garantir l'utilisation de la technologie BPEL, toute plateforme de services Web doit avoir un moteur (*engine*) de processus BPEL. C'est à la charge de ce dernier de déployer et d'exécuter les processus BPEL implémentant les services domaines. *ActiveBpel Engine*, sous sa version 5.0.2, est considéré comme le moteur BPEL le plus fiable et compatible avec notre serveur d'application *Jboss AS* déjà choisi.

Pour choisir ces serveurs déjà cités, nous nous sommes basés sur des études réalisées dans la littérature telles que [Umberto, 2008], [Kersten, 2005], [Baltus, 2001]. En outre, nous nous sommes également intéressés à travers un ensemble de tests à l'étude de compatibilité entre les serveurs choisis. Cette compatibilité est primordiale pour le bon acheminement de l'opération d'adaptation vu que tous ces serveurs vont y intervenir. Une description détaillée de ces études comparatives est fourni en annexe C de ce manuscrit de thèse.

[14] La Licence publique générale GNU, ou GNU General Public License est une licence qui fixe les conditions légales de distribution des logiciels libres du projet GNU.

- **La couche d'adaptation**

C'est la deuxième couche de notre architecture. Placée entre les couches applicative et ressource, elle contient les composants qui vont assurer le traitement des informations de contexte ainsi que toutes les autres opérations nécessaires pour réaliser l'adaptation des services invoqués par le processus BPEL. Essentiellement, cette couche contient les deux modules : le gestionnaire de contexte et le module d'activation d'aspect que nous avons déjà décrits auparavant (cf. section 4.2).

Afin de proposer une approche technique assez générique et afin de respecter le modèle d'orchestration adaptable que nous avons présenté (cf. section 4.4.1), nous avons choisi d'implémenter les deux modules déjà cités sous la forme de services Web. Ce choix permet d'une part d'avoir la possibilité de réutiliser ces deux composants dans d'autres processus BPEL, d'autre part, de les invoquer dans des activités BPEL comme tout autre service Web et par conséquent de réaliser des tests sur ces invocations.

– Le Gestionnaire de Context (*Context Manager Module*)

Comme nous l'avons déjà mentionné, le gestionnaire de contexte est chargé de collecter les informations de contexte et aussi de détecter les changements éventuels de ces informations. Afin de respecter le modèle d'orchestration proposé, le gestionnaire de contexte sera invoqué par le processus BPEL comme tout autre service Web. En effet, nous le placerons avant l'invocation de tout service sensible au contexte. Pour chaque service fonctionnel, sensible au contexte, orchestré au sein du service domaine, son invocation est précédée par l'appel du gestionnaire de contexte.

Lors de l'exécution du processus BPEL, chaque invocation du gestionnaire de contexte lance les trois opérations dans l'ordre suivant :

- Opération 1 : effectue une lecture des paramètres de contexte courant et prédécesseur à partir du registre de contexte. Ce dernier contient les valeurs des paramètres de contexte au moment de l'invocation du service et aussi les valeurs courantes.

- Opération 2 : détecte les changements de contexte par une comparaison des valeurs de paramètres de contexte (prédécesseur /courant). De la même manière, il identifie les informations de contexte auxquelles le service en question est sensible.

- Opération 3 : établir la liste des paramètres modifiés relatifs au contexte client ou encore au contexte de l'opportunité de coopération. Dans le cas où le contexte reste inchangé, le gestionnaire de contexte renseigne le paramètre de contexte auquel le service est sensible (*i.e.*, il s'agit du cas où le service en question est sensible à cette information de contexte). Dans le cas inverse (*i.e.*, changement de contexte), le gestionnaire de contexte retourne l'identifiant du service au module d'activation d'aspect ainsi que l'ensemble des paramètres de contexte qui ont changé et leurs nouvelles valeurs (voir Tableau 4.1).

ID-S	**Nom_paramètre_modifié**	**Valeur_paramètre_modifié**	**…..**

Tableau 4.1. Structure du tableau envoyé par le gestionnaire de contexte

─ Le module d'activation d'aspect (*Aspect Activator Module*)

Le deuxième module de notre couche d'adaptation est le module d'activation d'aspect. Tout comme le gestionnaire de contexte, le module d'activation d'aspect est implémenté sous la forme d'un service Web afin de mieux interagir avec les autres éléments de notre architecture. Pour respecter le modèle d'orchestration adaptable déjà présenté (cf. section 4.1), le module d'activation d'aspect est toujours placé après le gestionnaire de contexte dans le processus BPEL et son invocation passe par un test qui doit être effectué sur le résultat retourné par le gestionnaire de contexte (voir Figure 4.2). Ce test d'invocation est fait au niveau du processus BPEL sous la forme d'une activité IF. Techniquement, le test porte sur le contenu de la réponse du gestionnaire de contexte. À partir de ce contenu, le module d'activation d'aspect pourra donc déterminer s'il y a eu un changement de contexte (null/not null). De la même manière, il pourra vérifier s'il existe un aspect qui correspond au paramètre du contexte en question (dans le cas où il ne s'agit pas d'un changement de contexte mais d'une adaptation de la logique métier du service).

En outre, le module d'activation d'aspect se charge de sélectionner et de créer les ressources nécessaires afin de permettre au serveur *Jboss AS* de tisser le ou les aspects au sein du code de service à adapter. L'outil de tissage d'aspect que fournit le serveur *Jboss AOP* peut être configuré de façon à choisir le type de tissage souhaité. C'est-à-dire que nous pouvons choisir le moment de tissage à savoir pendant la compilation du service ou même lors de son exécution (*loadTimeWeaving*). Afin de satisfaire le besoin d'adaptation dynamique du service domaine, nous adoptons le deuxième type de tissage à savoir le tissage à l'exécution. Ainsi, nous configurons le serveur *Jboss AOP* selon ce type de tissage. Appliquer les aspects dynamiquement lors de l'exécution du service, présente un avantage très important pour assurer une adaptation dynamique, rapide et efficace. En outre, une autre caractéristique intéressante offerte par ce mode de tissage consiste à la capacité d'ajouter, de supprimer ou de modifier des aspects relatifs à un service à chaud pendant son exécution (sans redémarrer le serveur Jboss).

Le module d'activation d'aspect se charge de réaliser les deux opérations suivantes :
- Opération 1 : sélectionner l'aspect indispensable pour l'adaptation du service en question. La sélection de l'aspect à partir du registre d'aspect se base sur le résultat retourné par le gestionnaire de contexte, notamment l'identifiant du service et les paramètres et valeurs de contexte. Pour ce faire, le module d'activation d'aspect parcourt l'ensemble des aspects contenus dans le registre d'aspect en utilisant l'outil XML JDOM[15]. Le ou les aspects sélectionnés sont ceux dont les conditions satisfont les valeurs des paramètres de contexte.

[15] JDOM est une API du langage Java qui permet de manipuler des donnés XML plus simplement qu'avec les API classiques.

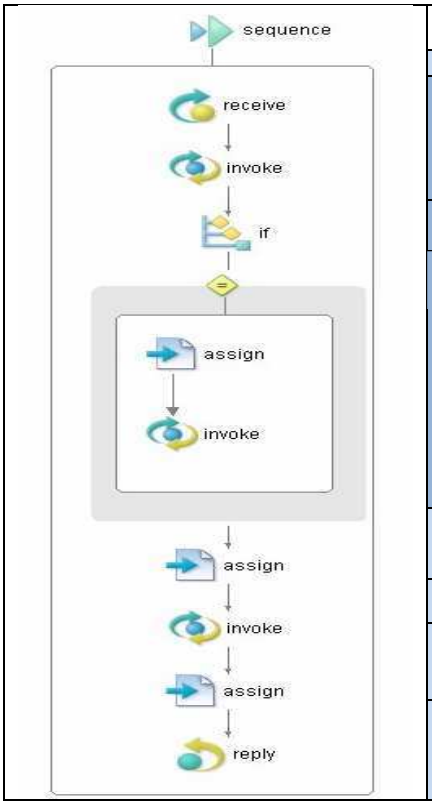
- Opération 2 : créer le fichier de *Binding*. En effet, afin de tisser un aspect dans un service, le module *Jboss AOP*, intégré au sein du serveur *Jboss AS*, a besoin d'un fichier de *Binding* nommé généralement avec la terminaison -aop.xml (voir Listing 1).

Le module d'activation d'aspect crée le fichier (-aop.xml) en indiquant le nom du service, le *pointcut* et le nom de l'aspect à tisser. Ces informations sont récupérées à partir du registre d'aspect. Le fichier de *Binding* généré par le module d'activation d'aspect est indispensable pour que le tisseur d'aspects puisse savoir quel aspect tisser et à quel niveau (*join point*) du service. La structure du fichier de *Binding* est montrée en Listing 1.

```
(01)  <?xml version="1.0" encoding="UTF-8" standalone="yes"?>
      <aop>
(03)   <aspect class="aop.jboss.MyJbossAOPAspect"/>
       <bind pointcut="execution( void *->@ annotation(..))">
(05)    <(around/ before/ after) aspect="" />
       </bind>
(07)  </aop>
```

Listing 1. Structure d'un fichier jboss-aop.xml

Comme illustré dans le Listing 1, ce fichier contient les *pointcuts* (Listing 1, ligne 04), le mode d'injection (*around*, *before*, *after*…) et le nom de l'aspect (Listing 1, ligne 05).
La figure suivante résume la structure du processus BPEL implémentant un service domaine adaptable.



Rôle de l'activité	
Recevoir la requête du Client BPEL	
Invoquer le gestionnaire de contexte (ws) **Input :** ID-Ws **Output:** ObjectTab [contient ID-Client, ID-Ws, {valeurs des paramètres de contexte changées}]. **Ressources utiles :** registre de contexte	
Vérifier si l'ensemble des variables retournées par le gestionnaire de contexte est null/not null	
NOT NULL	***Assigner le tableau Résultat à l'input du module d'activation d'aspect***
	Invoquer le module d'activation d'aspect(ws) **Input :** ObjectTab [] **Output:** Boolean(True/False) True=création du fichier jboss-aop.xml bien effectuée. False= problème de création du fichier jboss-aop.xml. **Ressources utiles :** registre d'aspect
Assigner les variables d'entrées au service Web concerné par l'adaptation.	
Invoquer le service Web concerné	
Assigner le résultat du Service Web à l'output du processus BPEL	
Retourner le résultat du processus BPEL au Client BPEL	

Figure 4.6. Structure du processus BPEL pour l'organisation de la procédure d'adaptation d'un service domaine

- **La couche ressource**

Elle représente la troisième couche de notre architecture technique. La couche ressource englobe l'ensemble des ressources indispensables pour les deux autres couches. Elle interagit essentiellement avec la couche adaptation au niveau des deux modules d'adaptation : le gestionnaire de contexte et le module d'activation d'aspect.

La couche ressource comporte les éléments suivants :

– *Répertoire de processus* : il répertorie l'ensemble des processus déployés dans le moteur BPEL. Ce répertoire est exigé pour le bon fonctionnement du moteur d'exécution des processus BPEL (*ActiveBpel engine*).

– *Registre de service* : représente dans notre cas le registre des services fonctionnels implémentés sous la forme de services Web. Ces différents services sont déployés au niveau du serveur *JbossWS*.

– *Registre d'aspect* : contient l'ensemble des aspects représentés sous la forme d'une arborescence XML. Chaque aspect constitue un nœud de l'arborescence (voir Figure 4.7) caractérisé par : un identifiant, une condition qui représente une valeur d'un paramètre de contexte, le nom de l'aspect ainsi que le *join point*, le mode d'exécution de l'*advice* (avant, après ou remplacement) et l'identifiant du service Web où il peut être tissé.

– *Registre de contexte :* alimenté et mis à jour par le gestionnaire de contexte, ce registre stocke les paramètres de contexte relatifs au client ainsi qu'à l'opportunité de coopération. Il stocke les informations courantes ainsi que les informations collectées lors de l'activation du service.

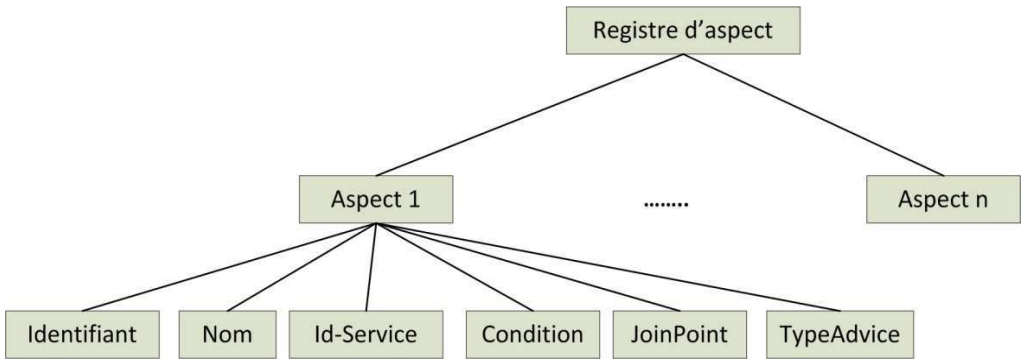


Figure 4.7. Structure du registre d'aspect

4.6 Scénario d'adaptation et validation

Pour mieux illustrer notre proposition nous présentons dans cette section un exemple d'utilisation concret. Nous allons traiter l'exemple d'un service domaine de livraison de marchandises offert par une entreprise de livraison. Cet exemple a été déjà exposé dans [Boukadi et al., 2008b].

Dans cet exemple, le client est l'entreprise qui souhaite livrer ses marchandises à partir de son entrepôt jusqu'à une destination finale. Tandis que le fournisseur, c'est celui qui offre le service domaine de livraison.

La Figure 4.8 illustre la modélisation du processus BPEL relatif au service domaine de livraison. Cette modélisation est conforme au modèle d'orchestration adaptable proposé (cf. section 4.4.1), sauf que pour des raisons de clarté nous avons omis la représentation du module d'activation d'aspect (*l'Aspect Activator Module*).

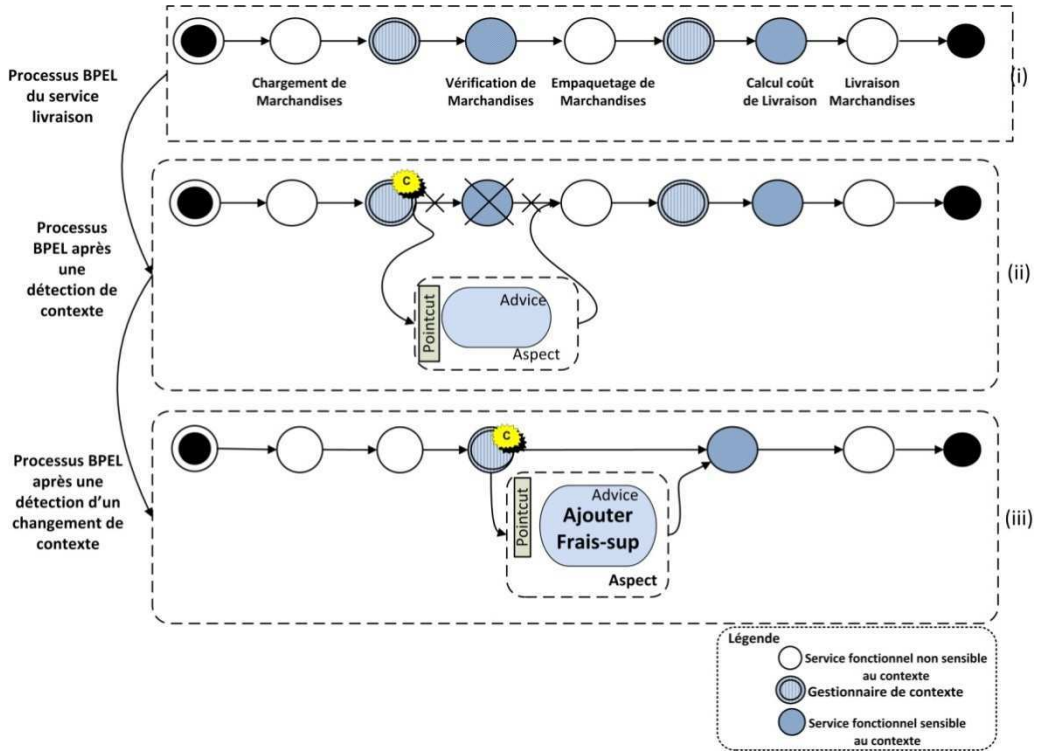


Figure 4.8. Modélisation du processus BPEL relative au service livraison

Le sous processus encapsulé par le service domaine comporte cinq services fonctionnels : le service de chargement de marchandises, le service vérification de marchandises, le service d'empaquetage de marchandises, le service calcul coût de livraison et enfin le service livraison de marchandises. Ces services sont décrits dans ce qui suit :

- Le service de chargement de marchandises : permet de récupérer les marchandises de l'entrepôt du client,
- Le service vérification de marchandises : effectue un contrôle des marchandises à transporter,
- Le service d'empaquetage de marchandises : selon le volume des marchandises il effectue une mise en paquets adéquats,
- Le service calcul coût de livraison : compte tenu de certains paramètres comme le trajet parcouru, la période de livraison (haute saison, basse saison), il calcule le coût total de la livraison,
- Le service livraison de marchandises : après l'acheminement des marchandises à la localisation souhaitée, ce service se charge d'informer le client.

Il est important de souligner que ces services, sont des services fonctionnels qui déclenchent par la suite le flux physique. Par exemple, le service de chargement de marchandises déclenche l'affectation d'un acteur humain qui se chargera de récupérer les marchandises et de les charger dans le moyen de transport utilisé.

Afin d'illustrer notre approche d'adaptation, nous considérons que le service de vérification des marchandises et le service calcul coût de livraison sont sensibles au contexte. Le service de vérification de marchandises est sensible au contexte du client. En effet, ce service contient une règle métier sensible au contexte qui stipule que les marchandises ne doivent pas être vérifiées pour les clients avec qui l'entreprise de livraison a déjà eu des transactions antérieures. Quant au service de calcul de coût de livraison, il est sensible à toute modification de lieu ou encore de la période de livraison. Ainsi, il ajoute des frais supplémentaires si le client change le contexte de la coopération (date et lieu de la livraison).

Le code BPEL du service domaine de livraison est illustré en Listing 2.

```
(01) <process name = "DeliveryPackage" .../>
       <sequence>
(03)     <receive partner="client" operation="getDeliveryPackage"
          variable="request"  createInstance="yes" .../>
(05)     <invoke partner="PickingMerchandises" operation="getOkResponse"
          outputVariable="PickingResponse"/>
(07)     <invoke partner="ContextManager" operation="getContextElement"
          iutputVariable= "ObjectTab"   outputVariable="ContextResponse"/>
(09)     <switch>
         <case condition=  "getVariableData ( ContextResponse ) != 'Null'" >
(11)     <invoke partner="AspectActivator"   inputVariable="ObjectTab" operation="SelectAspect" />
         </case>
(13)     </switch>
         <invoke partner="VerifyingMerchandises" operation="VerifyMerchandises"
(15)      outputVariable="VerifyingResponse />
         <invoke partner="PuttingMerchandisesInParcels"
(17)     operation="getOkResponse"  outputVariable="PuttingResponse"/>
         <invoke partner="ContextManager" operation="getContextElement"
(19)      iutputVariable= "ObjectTab"  outputVariable="ContextResponse"/>
         <switch>
(21)     <case condition=  "getVariableData ( ContextResponse ) != 'Null'" >
         <invoke partner="AspectActivator"  inputVariable="ObjectTab"  operation="SelectAspect" />
(23)     </case>
         </switch>
(25)     <invoke partner="ComputingDeliveryPrice"
          operation="ComputePrice" outputVariable="PriceResponse"/>
(27)     <invoke partner="DeliveryMerchandises"
          operation="getOkResponse" outputVariable="DeliveryResponse"/>
(29)   </sequence>
         <assign>...</assign>
(31)     <reply partner="client" operation=" getDeliveryPackage " variable="proposition" .../>
     </sequence>
(33) </process>
```

Listing 2. Code BPEL relatif au service domaine de livraison

Lors de l'exécution du processus BPEL, le premier service fonctionnel à invoquer est le service de chargement des marchandises (Listing 2 ligne 05). Le gestionnaire de contexte est invoqué pour collecter les informations de contexte auxquelles le service de vérification des marchandises est sensible (historique d'interaction) (Listing 2 ligne 07). Supposons que le client est une entreprise partenaire (historique d'interaction= Vrai). Dans ce cas de figure, le comportement du service de livraison doit être adapté pour répondre à cette information contextuelle. Pour assurer cette adaptation, le module d'activation d'aspect est invoqué pour sélectionner l'aspect adéquat. Puis, il crée le fichier Jboss-aop.xml indispensable pour l'exécution de l'aspect.

L'aspect sélectionné est montré en Listing 3. Cet aspect permet d'empêcher l'exécution du service de vérification des marchandises et de passer par la suite au service d'empaquetage des marchandises (Listing 3 ligne 06).

```
(01) import org.jboss.aop.joinpoint.ConstructorInvocation;
(02) import org.jboss.aop.joinpoint.FieldWriteInvocation;
(03) import org.jboss.aop.joinpoint.MethodInvocation;

(04) public class CheckMerchandiseAspect {
(05)    public Object CheckMerchandise (ConstructorInvocation invocation) throws Throwable
(06)     return invocation.invokeNext(); }
```

Listing 3. Code source de l'aspect sélectionné

Une fois que le service d'empaquetage des marchandises est exécuté, le gestionnaire de contexte est invoqué afin de collecter les informations de contexte dont le service de calcul de coût de livraison est sensible. Nous supposons que le client intervient au cours de l'exécution du processus et change le contexte de la coopération à travers une interface dédiée. Le changement de contexte correspond à un changement du lieu de livraison. Le nouveau lieu de livraison devient « Saint-Étienne » alors qu'au départ le client a sélectionné des entrepôts situés à « Lyon ». Par conséquent, le service domaine doit être adapté. Afin de détecter ce changement de contexte, le gestionnaire de contexte compare les informations contextuelles enregistrées au moment de l'invocation du service et les informations contextuelles enregistrées après l'exécution et ceci en consultant le registre de contexte. Pour notre exemple, l'exécution du gestionnaire de contexte donne naissance au Tableau 4.2.

20	**Lieu-Coopération**	**Saint-Étienne**

Tableau 4.2. Tableau résultant du traitement du gestionnaire de contexte

Étant donné ce changement de contexte, le module d'activation d'aspect est invoqué. C'est à la charge de ce module d'adaptation de désigner l'aspect qui répond à ce changement contextuel. Pour ce faire, le module d'activation d'aspect consulte le registre d'aspect afin d'identifier l'aspect à activer.

```
(01) <? Xml v='1.0' encoding='UTF-8'?>
     <AspectRepository>
(03) <aspect>
     <aspect-identifier> Asp12</aspect-identifier>
(05) <aspect-name> ExtraFees </aspect-name>
     <condition> Delivering Place!= Lyon</condition>
(07) <joinpoint>ComputePrice(....) <jointpoint>
     <advicetype> around  </advicetype>
(09) <id-ws> 30 </id-ws>
     </aspect>
     .....
 (n) </AspectRepository>
```

Listing 4. Aspect sélectionné à partir du registre d'aspect

En parcourant l'ensemble des aspects et en utilisant la condition issue de la nouvelle valeur du paramètre de contexte, le module d'activation d'aspect sélectionne l'aspect (ExtraFees) ayant la condition d'invocation convenable (Listing 4 ligne 06). Une fois que l'aspect est sélectionné, le module d'activation d'aspect crée le fichier Jboss-aop.xml. Ce fichier contient des informations indispensables pour le tissage de l'aspect au sein du service à savoir : le nom de l'aspect, le type de tissage (*around*) ainsi que le *join point* (la méthode compute (..)) et l'action d'adaptation (AddExtraFees). Le fichier Jboss-aop.xml résultat crée est illustré dans la Figure 4.9.

```
(01) <?xml version="1.0" encoding="UTF-8" standalone="yes"?>
     <aop>

(03) <!-- Aspect Declaration-- >
      <aspect class="ExtraFees"/>

(05) <!-- Pointcut Declaration-- >

      <bind pointcut="execution( public float $ instanceof    (ComputePrice)-> Compute (....)) ">
(07)  <around      aspect="AddExtraFees" />
      </bind>
(09) </aop>
```

Figure 4.9. Fichier Jboss-aop.xml créé

C'est au moment de l'invocation de la méthode calcul du service calcul coût livraison que le serveur *Jboss AOP* exécute l'aspect nommé « ExtraFees ». L'exécution de cet aspect attribut des frais supplémentaires au coût total de livraison. Par la suite, le processus BPEL continue son exécution par l'appel du service livraison de marchandises.

4.7 Conclusion

Nous avons présenté dans ce chapitre notre approche d'adaptation dynamique de la logique métier d'un service domaine. L'approche proposée répond à certaines limites concernant les services en général et celles relatives à leur adaptabilité dynamique aux changements de la logique métier.

Précisément, notre approche d'adaptation comble l'une des limites principales des services à savoir le manque de dynamisme. Ainsi, la définition d'un service n'est pas figée et sa logique métier peut évoluer en fonction des besoins et des changements affectant son contexte d'utilisation.

Notre approche d'adaptation a exploité le potentiel offert par la programmation orientée aspect, essentiellement par son concept de base à savoir l'aspect. Ce paradigme de programmation impliquant une séparation des préoccupations (métier/ actions d'adaptation vis à vis des changements de contexte), a joué un rôle prépondérant pour étendre et adapter le comportement d'un service. En outre, l'approche proposée a donné naissance à un modèle d'orchestration adaptable qui met en œuvre deux modules dédiés à l'adaptation : le gestionnaire de contexte et le module d'activation d'aspect (cf. section 4.2). Ce modèle a mis en évidence l'importance du tissage d'aspects pour assurer une adaptation dynamique de la logique métier des services. Notre approche d'adaptation a également contribué à enrichir le concept d'aspect à travers une extension du domaine des *join points* d'un aspect à l'ensemble des événements reliés au contexte (cf. section 4.4.2).

Afin de mettre en pratique l'ensemble des éléments définis dans l'approche d'adaptation, nous avons présenté une architecture technique répartie sur trois couches : application, adaptation et ressources. Cette architecture a démontré la faisabilité et la flexibilité de l'approche d'adaptation des services par tissage d'aspects.

L'annexe C de ce manuscrit de thèse présente un ensemble de détails techniques supplémentaires de notre contribution.

CONCLUSION GÉNÉRALE ET PERSPECTIVES

1. Rappel du cadre et des objectifs de la thèse

Les travaux de recherche effectués dans cette thèse portent sur la problématique de coopération interentreprises à la demande. Ils s'inscrivent dans la continuité des travaux initiés par [Izza, 2006] qui visent à répondre à la problématique d'intégration flexible des applications industrielles et par [Zaidat, 2005] qui ont défini un cadre générique de modélisation pour les réseaux d'entreprises.

Nos travaux de recherche s'intéressent à la question de coopérations à la demande en considérant que le système d'information est un élément central de cette problématique. L'examen des systèmes d'information des entreprises ainsi que leurs aptitudes à interopérer et à s'ouvrir à d'autres systèmes d'une manière dynamique, a constitué notre point de départ. Nous avons considéré que la réponse à cette interrogation constitue une clé pertinente et légitime pour résoudre la problématique de base : la coopération interentreprises à la demande.

Ainsi, nous avons porté l'essentiel de nos efforts sur l'étude de la capacité du système d'information à supporter des coopérations non planifiées qui lient sur une période de temps limitée un ensemble de partenaires.

Notre étude a révélé que la coopération à la demande est un choix stratégique difficile à réaliser vu que le système d'information n'est pas adapté à ce type de fonctionnement. En effet, l'adhésion d'une entreprise à des scénarios de coopération est régie par une double préoccupation. D'une part, l'entreprise présente un manque de flexibilité au niveau du système d'information, support à son métier et d'autre part, mettre en œuvre une coopération à la demande exige le développement d'un cadre bien défini qui permettra aux entreprises d'interconnecter leurs différents processus au sein d'un processus global.

Pour essayer de répondre à cette double préoccupation, nous avons étudié et analysé les approches et les mécanismes existants dans la littérature.

L'architecture orientée services et la technologie service Web semblent proposer des réponses crédibles aux besoins tant au niveau interne (système d'information support à la coopération) qu'au niveau externe (services disponibles aux partenaires). Ainsi, nos travaux de recherche ont eu pour objectif principal de développer une nouvelle approche qui assure l'efficacité et l'efficience de la coopération interentreprises. Ils se sont focalisés sur trois sous problématiques complémentaires qui sont respectivement la problématique de construction (ou de migration vers) d'une architecture orientée services au sein de l'entreprise, la

problématique de construction du processus coopératif à la demande et la problématique d'adaptation de la logique métier des services au contexte permettant de doter les services d'un certain niveau de flexibilité.

2. Principales contributions

Dans le cadre de cette thèse, nos contributions reposent sur trois principes méthodologiques qui sont une perspective d'ingénierie, l'intégration de la flexibilité et l'ouverture. L'inscription dans une perspective d'ingénierie permet de suivre une démarche méthodologique qui guide l'identification du système attendu (le système cible) en utilisant un ensemble de méta-modèles et des modèles. L'intégration de la flexibilité se manifeste à travers la prise en compte du contexte et de l'adaptation d'une manière transversale tout au long du cycle de vie. Le principe d'ouverture impose de s'inscrire dans le cadre d'utilisation des standards industriels. Chacun de ces principes est fondamental dans le sens où il intervient dans la réalisation de chacune de nos trois sous problématiques.

En tenant compte de ces principes méthodologiques, nous avons alors proposé une approche à trois niveaux. La construction de chaque niveau constitue une contribution principale de notre travail et répond à une des sous problématiques précédemment décrites. Ces trois contributions principales sont :

- La construction de l'architecture orientée services au sein de l'entreprise, qui permet de modéliser et de mettre en place une architecture orientée services appropriée au cadre d'une coopération interentreprises. Les principales contributions que nous avons apportés lors de la construction de cette architecture portent sur la dichotomie SOA métier et SOA IT, et aussi la proposition d'une démarche de construction. La dichotomie SOA métier et SOA IT est motivée par le fait qu'elle permet de traiter à la fois les deux préoccupations complémentaires qui sont celle de la maîtrise d'œuvre et celle de la maîtrise d'ouvrage. La démarche de construction, appelée CSOMA, permet de définir un ensemble de phases qui comportent des étapes nécessaires pour la construction de notre architecture de services. Il s'agit essentiellement d'une démarche qui vise à définir des services dédiés à la coopération nommés « services domaines ». Trois points pertinents distinguent notre démarche. Le premier consiste à proposer des méta-modèles qui définissent les concepts manipulés et la typologie des services prises en compte dans la démarche. Le deuxième point consiste à l'inscrire dans un cadre d'architecture dirigée par les modèles (MDA). Ainsi, à l'encontre des méthodes existantes qui sont fortement reliées à la technologie service Web, CSOMA est une démarche indépendante des technologies de réalisation qui part d'une spécification métier (ensemble de modèles métier) pour retrouver par la suite le modèle de système d'information orienté services. Quant au dernier point, il s'agit de la considération des variantes d'usage des services (étude de l'adaptation des services) dès la phase de modélisation. Cette particularité apporte la flexibilité nécessaire à la démarche CSOMA.

- La construction du processus coopératif à la demande à base de composition des services domaines. Les principales contributions que nous avons apporté lors de la construction du processus coopératif portent sur la proposition d'une architecture de coopération favorisant les scénarios de coopération à la demande et la proposition d'une démarche de construction du processus basé sur la composition des différents services

domaines proposés par les partenaires. L'architecture proposée est accompagnée par un ensemble de méta-modèles qui décrivent la sémantique des concepts manipulés comme l'organisation de la coopération à la demande, ses entités principales (les communautés, le registre de coopération, *etc.*) et les *constructs* du processus coopératif (notion de *Goal Template*, branchement, *etc.*).

Quant à la démarche de construction du processus coopératif, elle définit trois étapes fondamentales : la description, la publication et la découverte des services. Deux considérations distinguent notre démarche de construction. La première consiste à impliquer l'entreprise initiatrice du projet de coopération lors de la définition d'un schéma de processus coopératif et proposer par la suite une découverte des services qui satisfont les besoins du client (*i.e.* l'entreprise porteuse de projet de coopération). La deuxième consiste à la mise en oeuvre de l'adaptation au contexte tout au long des étapes de description, de publication et de découverte des services. Ainsi nous contribuons à améliorer les étapes de description, de publication et de découverte des services. Cette amélioration repose à la fois sur l'ontologie de catégorisation du contexte et le modèle multi-niveaux basé sur l'ontologie OWL qui propose de modéliser les informations de contexte.

- L'adaptation des services domaines au contexte à base de tissage d'aspects qui permet de définir les mécanismes permettant aux services domaines de s'adapter au contexte courant et d'évoluer pour correspondre à de nouveaux contextes d'utilisation. Les principales contributions que nous avons apporté lors de l'adaptation des services domaines sont la proposition d'une architecture d'adaptation et l'exploitation de la programmation orientée aspect pour étendre et adapter le comportement d'un service. L'architecture d'adaptation est supportée par un ensemble de modules qui se chargent de capter les changements de contexte et d'adapter la logique métier du service afin qu'il puisse être utilisé de façon efficace dans l'opportunité de coopération. L'adaptation de la logique métier se base sur l'injection et le tissage d'aspects lors de l'exécution du service. Ainsi, nous avons proposé le concept d'aspect dédié à l'adaptation qui enrichit le concept d'aspect classique à travers une extension du domaine des *join points* à l'ensemble des événements reliés au contexte.

L'adaptation des services domaines à base de tissage d'aspect a été testée sur une plateforme technologique qui regroupe la technologie service Web (WSDL, BPEL, *etc.*) et le langage de programmation orientée aspect (*Jboss AOP*).

Nos contributions présentent trois originalités qui les distinguent des travaux existants :

- La première originalité réside dans la proposition d'une démarche méthodologique pour la mise en place de l'architecture orientée services dédiée à la coopération. En effet, ce domaine est peu exploité dans les travaux académiques et semble être réservé aux entreprises de conseil, qui multiplient les démarches et les visions vis-à-vis de l'architecture de services.
- La deuxième originalité est la prise en compte du contexte depuis la modélisation des services (présentée dans la démarche CSOMA) jusqu'à leur exécution. Les travaux qui ont développé une approche transverse de l'adaptation au contexte sont peu nombreux.

- La troisième originalité consiste à utiliser la programmation orientée aspect pour assurer l'adaptation de la logique métier des services au contexte. Dans la littérature, la programmation orientée aspect a été employée pour réaliser des préoccupations de bas niveau comme la sécurité ou l'authentification. De plus, la possibilité de réaliser le tissage dynamique des aspects et l'ajout des aspects d'adaptation à chaud sans affecter les services constituent un autre point pertinent de la solution.

3. Limites de ces travaux de thèse

Nos travaux de recherche ne prétendent pas apporter une réponse parfaite et indiscutable à une problématique complexe qui est celle de la coopération interentreprises à la demande. Nous avons essentiellement essayé de proposer une démarche cohérente en adoptant une approche orientée services adaptables afin de réaliser des processus coopératifs à la demande basés sur une composition des services des partenaires.

- **Limites liées à la construction de l'architecture orientée services au sein de l'entreprise**

Notre approche d'interconnexion des processus d'entreprises repose sur une vision orientée services des systèmes d'information des partenaires. Ainsi, les partenaires doivent être en mesure de migrer vers une architecture orientée services et présenter leur système d'information selon cette même optique.

Cette contrainte préalable d'une mise en conformité avec les principes de l'architecture orientée services constitue une limite de nos travaux. Le fait d'exiger des partenaires une réflexion de fond sur leurs systèmes d'information constitue une hypothèse forte même s'ils affichent leur volonté de s'inscrire dans des scénarios de coopération à la demande. Néanmoins, des solutions de conformité technologiques ont été déjà longuement adopté par les entreprises et ce malgré les coûts de mise en place assez élevés (comme par exemple la technologie EDI). Par rapport à ce genre de conformité l'architecture orientée services promet plus d'interopérabilité, de flexibilité et d'agilité des systèmes. À l'heure actuelle, l'architecture orientée services connaît un développement incontestable manifesté par les concepts qui lui sont reliés comme les « *mash-up* » (combinaison de services applicatifs en ligne), le « *cloud computing* » (ressources informatiques en ligne) ou le « *Saas* » (logiciels fournis en ligne sous forme de services).

Au sein de la démarche CSOMA, nous avons proposé un ensemble de lignes directrices qui guident les architectes vers un modèle orienté services. Cependant, nous n'avons pas proposé une transformation automatique du niveau CIM (*Computation Independent Model*) vers le niveau PIM (*Platform Independent Model*). Dans cette phase, l'expérience de l'analyste et son savoir-faire sont déterminants pour la réussite du processus.

- **Limites liées à la construction du processus coopératif**

Dans cette partie de nos contributions, nous pouvons mentionner trois limites principales :

- La complexité de l'algorithme de découverte des services. En effet, l'algorithme que nous avons développé est relativement complexe du fait qu'il a pour ambition d'employer trois filtres (syntaxique, sémantique et contextuel) afin de découvrir les services domaines. Cette

complexité doit, à notre sens, être masquée par des mécanismes de relaxation qui vont proposer au client des services qui ne couvrirait qu'une partie de la requête (syntaxique, sémantique ou contextuelle).

- L'expérimentation de l'algorithme de découverte des services domaines. En effet, nous n'avons pas pu tester l'algorithme sur une collection de services afin d'évaluer ses performances d'exécution comme le temps de réponse en millisecondes ou le nombre de services retournés suite à une requête client. Cette restriction découle en grande partie du nombre limité des services domaines développés.

Néanmoins ces restrictions sur la découverte des services, si elles sont importantes dans un contexte général (Internet), sont à relativiser dans notre problématique de coopération à la demande. En effet, la construction du processus coopératif est un processus de négociation dans lequel les acteurs se connaissent et par conséquent connaissent leurs besoins en termes de services pour le processus coopératif.

- La sélection des schémas d'orchestration au sein des services domaines n'a pas fait l'objet de contribution supplémentaire plus approfondie par rapport à celles existant dans l'état de l'art.

- **Limites liées à l'adaptation des services domaines à base de tissage d'aspects**

La principale question que nous pouvons formuler concernant notre approche d'adaptation de la logique métier des services porte principalement sur le fait que les actions d'adaptation ne sont pas clairement visibles au client et sont par la suite appliquées à son insu à chaque changement de contexte. Dans une optique de coopération interentreprises, l'adaptation de la logique métier du service au moment de son exécution doit être appréhendée par les clients. Une solution pertinente qui pourrait être envisagée consiste à l'utilisation conjointe de la programmation orientée aspect et des politiques descriptives (par exemple le standard *WS-Policy*) afin de décrire les actions d'adaptation dans un langage compréhensible par le client. Nous avons commencé cette initiative dans un travail récent [Boukadi et al., 2009c].

Les hypothèses qui sous-tendent les mécanismes d'adaptation de la logique métier du service sont définies de manières restrictives. Par exemple, dans l'état actuel du prototype même si un service est sensible à plusieurs paramètres de contexte, à un instant *t* l'aspect correspondant à un et un seul paramètre de contexte sera injecté.

- **Limite liée aux prototypes développés**

Comme nous l'avons déjà souligné, chacune de nos propositions a été mise en œuvre grâce à un prototype support (cf. Annexe A, Annexe B et Annexe C). Ces prototypes sont reliés entre eux et proposent des passerelles afin de naviguer d'un prototype à un autre. La figure suivante illustre les trois prototypes développés ainsi que leurs articulations. Néanmoins, certains modules ne sont pas encore mis en œuvre (parties représentées en gris clair) et nous envisageons de les compléter dans nos travaux futurs.

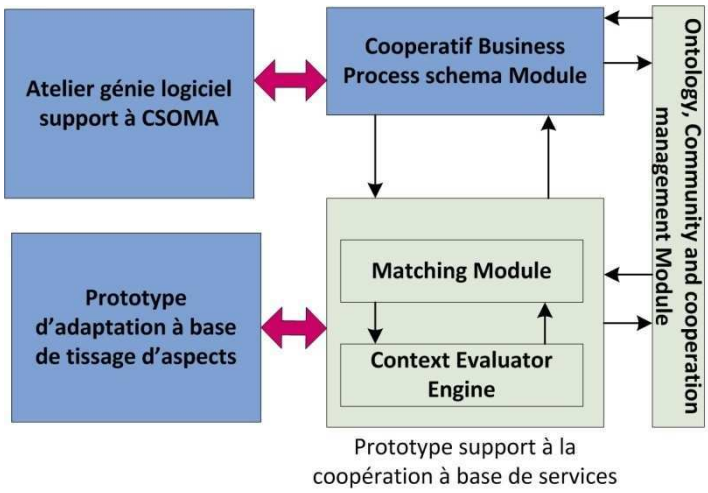


Figure 4. Bilan des prototypes développés ou à développer

4. Perspectives

À partir des limites évoquées et soucieux de la continuité des travaux futurs qui pourraient être développés sur cette thématique, nous avons dégagé un ensemble de perspectives. Ces perspectives nous les avons classées en deux catégories : les perspectives liées à la suite de nos contributions et les perspectives qui s'inscrivent dans une optique de travail connexe mené actuellement dans notre laboratoire de recherche.

- **Perspectives liées à la continuité de nos contributions**

Une première perspective consiste à intégrer les connaissances métier dans la démarche CSOMA. Il est important de prendre en considération les connaissances métier formalisées dans le processus d'identification des services de l'entreprise. Cette perspective peut hériter des avancées réalisées dans le cadre de la gestion des connaissances métier. En effet, la dimension cognitive relative au savoir faire, aux concepts métier a souvent suscité l'intérêt de plusieurs travaux de recherche qui ont proposé des ontologies de capitalisation de connaissances et des mémoires d'entreprise. Après une phase d'évaluation des outils de capitalisation des connaissances métier (ontologie, mémoire d'entreprise *etc.*), CSOMA pourra mettre en œuvre un processus d'identification des services d'entreprise plus sophistiqué et réaliser ainsi une transformation plus automatisée du niveau CIM vers le niveau PIM du cadre MDA.

La deuxième perspective consiste à proposer des règles de transformation qui visent à réaliser une transformation semi-automatique du niveau PIM (modèles de services fonctionnels, modèles d'orchestration des services domaines et modèles d'Aspects Conceptuels) vers le niveau PSM ayant comme plateforme cible la technologie services Web et le langage *JBoss*

AOP pour l'implémentation des aspects. En l'état actuel, cette transformation est réalisée d'une manière manuelle.

Une troisième perspective consiste à intégrer des bonnes pratiques collaboratives au sein de la démarche CSOMA. Ainsi, les services domaines identifiés pourront être en phase avec ces bonnes pratiques. Cette perspective rejoint les travaux réalisés dans le cadre du projet régional Copilotes 2[16] auquel nous participons. En effet, une des activités, dans le cadre de ce projet, consiste à construire une ontologie qui récapitule les concepts fondamentaux du modèle SCOR qui est un modèle définissant les bonnes pratiques pour la chaîne logistique. Nous envisageons d'intégrer cette ontologie afin de mieux définir la spécification du service domaine.

Une dernière perspective consiste à achever le développement des modules manquants afin d'assurer une plateforme de coopération complète et la tester sur Internet.

- **Les perspectives qui s'inscrivent dans une optique de travail connexe**

Ces perspectives s'inscrivent dans le cadre du projet SP-OS (Système de Production Orientés Services)[17]. Sur la base de nos travaux de thèse, le travail futur a pour objectif d'étendre et d'approfondir la notion d'adaptation au contexte, en la traitant non plus pour des processus informatisables, mais en la traitant directement au niveau de **la réponse organisationnelle** apportée aux besoins d'un client-usager. Il ne s'agit plus de travailler sur un système informatique, mais de travailler sur la configuration du système de production global qui sera destiné à délivrer le produit service. Ce système de production sera constitué par « maillage » (ou encore « interconnexion ») entre un ensemble de services métier délivrés par une ou plusieurs entreprises, qui sont impliquées dans la réponse au client.

L'enjeu consiste donc à contribuer à l'élaboration d'un modèle d'ingénierie d'entreprise qui permette de traiter la configuration agile de chaînes productives, directement au niveau « métier ». Dans l'élaboration de ce modèle d'ingénierie d'entreprise, une réflexion nouvelle et approfondie doit être faite sur l'intégration de la notion de contexte, et sur la composition adaptable des services métier.

Ce travail recouvre plusieurs objectifs spécifiques :

- Étude, structuration et formalisation des différents facteurs d'incertitude ou de contextualisation à prendre en compte dans la configuration de chaînes productives délivrant des systèmes produits/services. Selon le degré d'incertitude à prendre en compte

[16] Le projet COPILOTES est un projet qui a reçu le soutien de la Région Rhône-Alpes et ayant pour objectifs d'aider les industriels dans la mise en place de pratiques collaboratives au sein de leur chaîne logistique.
[17] Le projet SP-OS (Système de Production Orientés Services) soutenu par la région Rhônes-Alpes, coordonné par le centre G2I de l'EMSE en collaboration avec les laboratoires G-SCOP de l'université Joseph Fourrier, INP Grenoble et CO-ACTIS de l'université de Lyon II.

dans la réponse à un besoin, la chaîne productive à configurer ne sera pas la même, et reposera sur des compétences et des capabilités technologiques différentes.

- Spécifier les éléments du méta-modèle d'architecture d'entreprise qui permettent d'intégrer la notion d'adaptation au contexte à différents niveaux : dans l'expression du besoin client où le contexte doit être recueilli ; au sein des scénarios d'usage du système produit/service conçus pour répondre au besoin (scénarios d'usage intégrant un certain degré d'incertitude) ; au niveau des chaînes productives configurées par composition de services métier et fournissant une réponse opérationnelle aux scénarios d'usage identifiés.
- Développer des mécanismes de compositions des services, directement applicables au niveau de services métier intégrant les exigences d'adaptation au contexte. Ces mécanismes de composition de services métier devront permettre d'évaluer par anticipation la performance des chaînes productives configurées pour répondre à des besoins du client.

En conclusion nos travaux s'inscrivent au sein d'un domaine, le service, pour lequel beaucoup de potentialités restent à découvrir et à exploiter. Nous avons proposé une démarche permettant de développer et d'utiliser des services adaptables pour améliorer la flexibilité des systèmes d'information et par là, la réactivité des entreprises. Nous avons ouvert un certain nombre de perspectives dont le développement nécessitera un travail de longue haleine et dont l'aboutissement nécessitera sans doute plusieurs années. Cependant, nous demeurons convaincus, compte tenu de son intérêt, que ce travail doit être soutenu et poursuivi.

Références Bibliographiques

Abowd, G. D., Dey, A. K., Brown, P. J., Davies, N., Smith, M. and Steggles, P., 1999, Towards a Better Understanding of Context and Context-Awareness, In *the 1st international symposium on Handheld and Ubiquitous Computing*Springer-Verlag Karlsruche, Germany pp. 304 - 307.

Agarwal, V., Chafle, G., Dasgupta, K., Karnik, N., Kumar, A., Mittal, S. and Srivastava, B., 2005, Synthy: A system for end to end composition of web services, *Web Semantics: Science, Services and Agents on the World Wide Web*, **3**(4), pp. 311–339.

Akkermans, H. A., Bogerd, P. and P, P. V. D., 2004, Travail, Transparency and Trust: A case study of computer-supported collaborative supply chain planning in high-tech electronics, *European Journal of Operational Research*, **153**(2), pp. 445-456.

Al-Masri, E. and Mahmoud, Q. H., 2007, QoS-based Discovery and Ranking of Web Services, In *IEEE 16th International Conference on Computer Communications and Networks*, pp. 529-534

Amin, A. and Cohendet, P., 2003 *Knowledge practices, communities and competences in firms*, Oxford Universiry Press

Amir, R. and Zeid, A., 2004, An UML Profile for Service Oriented Architectures, In *Companion to the 19th Annual ACM SIGPLAN Conference on Object-Oriented Programming, Systems, Languages, and Applications, OOPSLA 2004*, pp. 192-193.

Andonoff, E. and Bouzguenda, L., 2005, An Agent-Based Negotiation between Partners in Loose Inter-Organizational Workflow, In *the International Conference On Intelligent Agent, (IAT'05), IEEE Computer Society*Université de Compiègne-France, 19-22 Septembre, 2005., pp. 619-625.

Andrews, T., Curbera, F., Dholakia, H., Goland, Y., Klein, J., Leymann, F., Liu, K., Roller, D., Smith, D., Thatte, S., Trickovic, I. and Weerawarana, S., 2003a, Business Process Execution Language for Web Services , available at: http://www.ibm.com/developerworks/library/specification/ws-bpel/.

Andrews, T., Curbera, F., Dholakia, H., Goland, Y., Klein, J., Leymann, F., Liu, K., Roller, D., Smith, D., Thatte, S., Trickovic, I. and Weerawarana, S., 2003b, Business Process Execution Language for Web Services Version 1.1 available at: http://www-128.ibm.com/developerworks/library/specification/ws-bpel/.

Angelov, S. and Grefen, P., 2002, Support for B2B E-contracting- The Process Perspective, In *the 5th IFIP International Conference on Information Technology for Balanced Automation Systems In Manufacturing and Services (BASYS' 02)*Cancun, Mexico, pp. 87-96.

Anton, A., McCracken, W.-M. and Potts, C., 1994, Goal decomposition and scenario analysis in Business Process Reengineering, In *6th International Conference on Advanced Information Systems Engineering (CAiSE '94)*, Utrecht, Pays-Bas.

Apache, 2005, Apache beehive, available at: http://beehive.apache.org/.

Arena, R., 2003, Relation inter-entreprises et communautés médiatées : Une analyse preliminaire Latapses-Demos, Université de Nice-Sophia Antipolis/CNRS.

Arsanjani, A., 2004, Service-oriented modeling and architecture available at: http://www.ibm.com/developerworks/library/ws-soa-design1/.

AspectWerkz, 2008, http://aspectwerkz.codehaus.org/.

ATHENA, 2005, Report on methodology description and guidelines definition Version 1.0, ATHENA Integrated Project, deliverable D.A1.3.1, March 2005. .

Baina, K., Benali, K. and Godart, C., 2003, Dynamic Interconnection of Heterogeneous Workflow Processes through Services, In *CoopIS/DOA/ODBASE*, pp. 444-461.

Baina, K., Benatallah, B., Casati, F. and Toumani, F., 2004, Model-Driven Web Service Development, In *16th International Conference on Advanced Information Systems*

*Engineering - CAiSE 2004*Lecture Notes in Computer Science, Riga, Latvia, June 7-11.

Baldauf, M., Dustdar, S. and Rosenberg, F., 2007, A survey on context-aware systems, *International Journal of Ad Hoc and Ubiquitous Computing,* **2**(4), pp. 263-277

Baltus, J., 2001, La Programmation Orientée Aspect et AspectJ : Présentation et Application dans un Système Distribué, In *Mini-Workshop: Systèmes Coopératifs*Institut d'informatique, Namur.

Baral, C. and Lobo, J., 1996, Formal Characterization of Active Databases, In *the International Workshop on Logic in Databases*(Eds, Pedreschi, D. and Zaniolo, C.) Lecture Notes In Computer Science, pp. 175-195.

Baresi, L., Heckel, R., Thone, S. and Varro, D., 2003, Modeling and validation of service-oriented architectures: Application vs. style., In *the 9th European Software Engineering Conference (ESEC/FSE 2003)*Helsinki, Finland, September 1-5.

Bauer, M., Heiber, T., Kortuem, G. and Segall, Z., 1998, A collaborative wearable system with remote sensing, In *The 2nd International Symposium on Wearable Computers (ISWC98)*(Ed, IEEE, C.) Los Alamitos, pp. 10-17.

BEA, 2006, Service-oriented Architecture, available at: http://dev2dev.bea.com/soa/.

BearingPoint, SUN and SAP, 2003, Les Services Web, pourquoi ? Dix questions essentielles, disponible sur : http://www.bearingpoint.fr/media/Library/20030108_CH5Whitepaper_Services_Web.pdf.

Behlouli, N. B., Taconet, C. and Bernard, G., 2006, An architecture for supporting Development and Execution of Context-Aware Component applications, In *IEEE International Conference on Pervasive Services* Lyon, France, pp. 26-29.

Bénaben, F., Touzi, J., Rajsiri, V. and Pingaud, H., 2006, Collaborative Information System Design, In *11th International Conference of the Association Information and Management (AIM), Luxembourg June 8-9, 2006*(Eds, Feltz, F., Otjacques, B., Oberweis, A. and Poussing, N.).

Benatallah, B., Casati, F. and Toumani, F., 2006, Representing, analysing and managing web service protocols, *Data Knowl. Eng.*, **58**(3), pp. 327-357.

Benatallah, B., Dijkman, R. and Dumas, M. (2005a) In *Service- Oriented Software Engineering: Challenges and Practices*(Eds, Stojanovic, Z. and Dahanayake, A.) Idea Group Inc (IGI), pp. 48-66.

Benatallah, B., Dumas, M. and Sheng, Q., 2005b, Facilitating the Rapid Development and Scalable Orchestration of Composite Web Services, *Distributed and Parallel Database*, **17**(1), pp. 5-37.

Benatallah, B., Hacid, M.-S., Rey, C. and Toumani, F., 2003a, Semantic Reasoning for Web Services Discovery, In *the 2nd International Semantic Web Conference (ISWC 2003)*, pp. 242-257.

Benatallah, B., Sheng, Q. Z. and Dumas, M., 2003b, The Self-Serv environment for Web services composition, *IEEE Internet Computing*, **7**(1), pp. 40-84.

Benslimane, D., Arara, A., Falquet, G., Maamar, Z., Thiran, P. and Gargouri, F., 2006, Contextual Ontologies: Motivations, Challenges, and Solutions, In *Fourth Biennial International Conference on Advances in Information Systems*(Ed, Springer) Izmir, Turkey, pp. 168 - 176.

Berre, A.-J., Elveaster, B., Figay, N., Guglielmina, C., Johnsen, S., Karlsen, D., Knothe, T. and Lippe, S. (Eds.) (2007) *The ATHENA Interoperability Framework, Enterprise interoperability: New challenges and approaches II*, Springer.

Bettini, C., Maggiorini, D. and Riboni, D., 2007, Distributed Context Monitoring for the Adaptation of Continuous Services, *World Wide Web*, **10**(4), pp. 503-528.

Bézivin, J. and Blanc, X., 2002, MDA : standards et travaux, In *Développeur Référence*.

Bézivin, J., Dupé, G., Jouault, F., Pitette, G. and Rougui, J. E., 2003, First Experiments with the ATL Model Transformation Language: Transforming XSLT into XQuery, In *the 2nd OOPSLA Workhop on Generative Techniques in the context of Model Driven Architecture,*.

Bieberstein, N., Bose, S., Fiammante, M., Jones, K. and Shah, R., 2005, *Service-Oriented Architecture Compass: Business Value, Planning, and Enterprise Roadmap*, IBM Press.

Bitcheva, J., 2003, PointSynchro : un service pour la coordination de procédés interentreprises coopératifs, Thèse de l'Université Henri Poincaré, Nancy, pp. 160.

Bitner, M. J. and Brown, S. W., 2008, The service imperative, *Business Horizons*, **51**(1), pp. 39-46.

Booch, G., 1992, *Conception orientée objets et applications*, Editions Addison-Wesley.

Booch, G., Jacobson, I. and Rumbaugh, J., 2004, *UML 2.0 Guide de référence*, Paris, CampusPress,.

Bordbar, B. and Staikopoulos, A., 2004a, Automated Generation of Metamodels for Web Service Languages, In *Second European Workshop on Model Driven Architecture (MDA) with an emphasis on Methodologies and Transformations (EWMDA), September 2004*.

Bordbar, B. and Staikopoulos, A., 2004b, On Behavioural Model Transformation in Web Services, In *the 23rd International Conference on Conceptual Modeling (ER2004) and eCOMO*

Bordeaux, L., Salaün, G., Berardi, D. and Mecella, M., 2004, When are Two Web Services Compatible? , In *the 5th International Workshop on Technologies for E-Services (TES)*, pp. 15-28.

Boukadi, K., 2009, How to run ActiveBpel 5.0.2 on Jboss 5.0, available at: http://www.emse.fr/~boukadi/.

Boukadi, K., Chaabane, A. and Vincent, L., 2009a, Context-Aware Business Processes Modelling: Concepts, Issues and Framework, In *the 13th IFAC Symposium on INFORMATION CONTROL PROBLEMS IN MANUFACTURING (INCOM' 09)*Moscow.

Boukadi, K., Chaabane, A. and Vincent, L., 2009b, A Framework for Context-Aware Business Processes Modelling, In *In the International Conference on Industrial Engineering and Systems Management, IESM 2009, May 13 - 15*, MONTREAL - CANADA. .

Boukadi, K., Ghedira, C., Chaari, S., Vincent, L. and Bataineh, E., 2008a, CWSC4EC:How to Employ Context, Web Service, and Community in Enterprise Collaboration, In *In the 8th International Conference on New Technologies of Distributed Systems (NOTERE 2008)*Lyon, France.

Boukadi, k., Ghedira, C., Maamar, Z., Benslimane, D. and Vincent, L., 2009c, Context-Aware Data and IT Services Collaboration in E-Business (à paraître), *Transactions on Large Scale Data and Knowledge Centered Systems*, pp. 91-115.

Boukadi, K., Ghedira, C. and Vincent, L., 2008b, An Aspect Oriented Approach for Context-Aware Service Domain Adapted to E-Business, In *The 20th International Conference on Advanced Information Systems Engineering*,Montpellier-France

Boukadi, K., Vincent, L. and Burlat, P., 2009d, Contextual Service Oriented and Analysis: the IT concerns, In *research report available at: http://www.emse.fr/~boukadi/.*

Boukadi, K., Vincent, L. and Burlat, P., 2009e, Modelling Adaptable Business Service for Enterprise Collaboration In *10th IFIP Working Conference on Virtual Enterprises*, Hessaloniki, Greece, 7-9 October 2009.

Bourdon, J., 2007, Multi-agent systems for the automatic composition of semantic web services in dynamic environments, In *Rapport de master, École des Mines de Saint Etienne - G2I & Université Joseph Fourier*.

Bouzguenda, L., 2005, How to design a Loose Inter-Organizational Workflow: An illustrative case study, In *the 16th International Conference on Database and Expert Systems Applications (DEXA'05)*, K.V. Andersen, J.K. Debenham, R. Wagner (Eds), LNCS 3588, Copenhagen-Denmark, 22-26 August, 2005., pp. 1-13.

BPMI, 2002, Business Process Management Initiative. http ://www.bpmi.org.

BPMI, 2004, Business Process Modeling Notation (BPMN), Version 1.0, May 2004.

BRG, 2002, The Business Rule Group, Defining Business Rules: What Are They Really? , available at : http://www.businessrulesgroup.org/bra.shtml.

Broens, T., Pokraev, S., Sinderen, M. v., Koolwaaij, J. and Costa, P. D. (Eds.) (2004) *Context-aware, ontology-based, service discovery*, Springer.

Brown, A. W., Delbaere, M., Eeles, P., Johnston, S. and Weaver, R., 2005, Realizing Service-oriented solutions with the IBM Rational Software Development Platform, *IBM System Journal*, **44** (4), pp. 727-752

Brown, M. J. B. a. S. W., 2008, The service imperative, *Business Horizons*, **51**(1), pp. 39-46.

Brown, P. J., 1996, The stick-e document: a framework for creating context-aware applications, In *Proceedings of EP'96* pp. 259-272.

Burn, J. M., 1998, Aligning the On-Line Organisation - with What, How and Why?, In *the 8th Business Information Technology Conference*Manchester, 4-5th November.

Camarinha-Matos, L. M. and Afsarmanesh, H., 2006, Creation of virtual organizations in a breeding environment, In *12ème Symposium Triennal de l'IFAC INCOM06 (Information Control Problems in Manufacturing)* Saint-Etienne, France.

Casati, F. and Discenza, A., 2000, Supporting Workflow Cooperation Within and Across Organisations, In *15th ACM Symposium on Applied Computing (SAC'00)*Como, Italy, pp. 9-21.

Casati, F., Ilnicki, S., Jin, L.-J., Krishnamoorthy, V. and Shan, M.-C., 2000a, eFlow : a platform for developing and managing composite e-services, HPL-2000-36. Software Technology Laboratory, HP Laboratories.

Casati, F., Ilnicki, S., Jin, L.-J., Krishnamoorthy, V. and Shan, M.-C., 2000b, eFlow : an Open, Flexible, and Configurable Approach to Service Composition, In *2nd International Workshop on Advance Issues of ECommerce and Web-Based Information Systems (WECWIS'00)*Milpitas, California, pp. 125-132,.

Casati, F. and Shan, M.-C., 2001, Dynamic and adaptive composition of e-services, *Information Systems*, **26**(3), pp. 143-163.

Casati, F. and Shan, M.-C., 2002, Event-Based Interaction Management for Composite E-Services in eFlow, *Information Systems Frontiers*, **4**(1), pp. 19-31.

Castro, V. d., Marcos, E. and Sanz, M. L., 2006, A model driven method for service composition modelling: a case study, *International Journal of Web Engineering and Technology*, **2**(4).

Castro, V. d., Vara, J. M. and Marcos, E., 2007, Model Transformation for Service-Oriented Web Applications Development, In *the 3th International Workshop on Model-Driven Web Engineering (MDWE'07)*Como, Italy.

Cauvet, C. and Guzelian, G., 2008, Business Process Modeling: a Service-Oriented Approach, In *the 41st Hawaii International Conference on System Sciences*Hawaii

Cazzola, W., Savigni, A., Sosio, A. and Tisato, F., 1999, Architectural Reflection: Concepts, Design, and Evaluation, In *Technical Report RI-DSI*Universitá di Milano-Bicocca, Milan, Italy.

Chaari, S., Boukadi, K., Benamar, C., Biennier, F. and Favrel, J., 2008, Developing Service Oriented Enterprise by Composing Web Services Based on Context and Policy, *International Journal of Computer Science and Security*, **8**(6), pp. 79-92.

Chafle, G., Das, G., Dasgupta, K., Kumar, A. and Mittal, S., 2007, An Integrated Development Environment for Web Service Composition, In *The IEEE International Conference on Web Services*Salt Lake City, USA.

Chakraborty, D., Perich, F., Avancha, S. and Joshi, A., 2001, DReggie: Semantic Service Discovery for M-Commerce Applications, In *Workshop on Reliable and Secure Applications in Mobile Environment, In Conjunction with 20th Symposium on Reliable Distributed Systems (SRDS), October*

Chang, S. H. and Kim, S. D., 2007, A Service-Oriented Analysis and Design Approach to Developing Adaptable Services, In *the IEEE International Conference on Services Computing (SCC 2007)*, pp. 204-211

Chappell, D., 2005, *Enterprise Service Bus*, O'Reilly media Inc.

Charfi, A., 2008, Aspect-Oriented Workflow Management. Concepts, Languages, Applications, (Ed, Mueller, V. V. D.).
Charfi, A., Berbner, R., Mezini, M. and Steinmetz, R., 2007a, Management Requirements of Web Service Compositions, In *the 2nd ECOWS07 Workshop on Emerging Web Services Technology, WEWST 2007*Germany.
Charfi, A., Khalaf, R. and Mukhi, N., 2007b, QoS-Aware Web Service Compositions Using Non-intrusive Policy Attachment to BPEL, In *the 5th International Conference on Service-Oriented Computing (ICSOC), industry track*.
Charfi, A. and Mezini, M., 2007, AO4BPEL: An Aspect-oriented Extension to BPEL, *World Wide Web*, **10**(3), pp. 309-344.
Charfi, A., Schmeling, B., Heizenreder, A. and Mezini, M., 2006, Reliable, Secure, and Transacted Web Service Compositions with AO4BPEL, In *the 4th IEEE European Conference on Web Services (ECOWS), December 2006* Zürich, Switzerland.
Chauvet, J. M., 2002, *De XML aux services Web pour les entreprises,Techniques de l'ingénieur*.
Chelli, H. (Ed.) (2003) *Urbaniser l'entreprise et son système d'Information. Guide des entreprises agiles*.
Chen, G. and Kotz, D. A., 2000, Survey of Context-Aware Mobile Computing Research, In *Technical Report TR2000-381, Dartmouth*, pp. 16.
Chen, G., Ren, W., Zhang, J. B., Yang, Z., Low, C. P., Sun, C. and Chen, D., 2008, Dynamic Virtual Enterprise Integration via Business Rule Enhanced Semantic Service Composition Framework, In *the 3rd IEEE Conference on Industrial Electronics and Applications, Singapore*.
Chen, H., Finin, T. and Joshi, A., 2004a, An Ontology for Context-Aware Pervasive Computing Environments, *Special Issue on Ontologies for Distributed Systems, Knowledge Engineering Review*, **18**(3), pp. 197-207.
Chen, H., Perich, F., Chakraborty, D., Finin, T. and Joshi, A., 2004b, Intelligent Agents Meet Semantic Web in a Smart Meeting Room, In *the Third International Joint Conference on Autonomous Agents and Multi Agent Systems (AAMAS 2004)*New York City, NY.
Chevrin, V., 2006, L'Interaction Usagers/Services, multimodale et multicanale : une première proposition appliquée au domaine du e-Commerce, Université de Lille1.
Collet, C., 1996, Bases de Données actives : des systèmes relationnels aux systèmes à objets, In *Mémoire pour l'obtention du diplôme d'Habilitation à diriger des recherches*LSR-IMAG, Grenoble, France.
Cook, S., 2004, Domain-Specific Modeling and Model Driven Architecture, *MDA Journal*, pp. 1-10.
Courbis, C. and Finkelstein, A., 2005, Towards Aspect Weaving Applications, In *the 27th International Conference on Software Engineering*ACM Press, pp. 66-77.
Cuddy, S., Katchabaw, M. and Lutfiyya, H., 2005, Context-Aware Service Selection Based on Dynamic and Static Service Attributes, In *the IEEE International Conference on Wireless and Mobile Computing, Networking and Communications*, Montréal, Canada, pp. 148-159.
Dardenne, A., Lamsweerde, A. v. and Fickas, S., 1993, Goal-directed requirements acquisition, *Science of Computer Programming*, **20**(1-2), pp. 3-50.
David, P.-C., 2005, Développement de composants Fractal adaptatifs: un langage dédié à l'aspect d'adaptation, École des mines de Nantes,.
Davidow, W. and Malone, M., 1992, *The Virtual Corporation*.
Délfard, H., 2000, *Une explication non réductionniste de la coopération inter-firmes*, Economica.
Dey, A. K., Abowd, G. D. and Salber, D., 2001, A Conceptual Framework and a Toolkit for Supporting the Rapid Prototyping of Context-Aware Applications, *Human-Computer Interaction*, **16**(2), pp. 97-166.
Dey, A. K., Salber, D., Futakawa, M. and Abowd, G. D., 1999, An Architecture To Support Context-Aware Applications, In *GVU Technical Report GIT-GVU-99-23. Submitted*

to the 12th Annual ACM Symposium on User Interface Software and Technology (UIST '99), June.

Diamadopoulou, V., Makrisa, C., Panagis, Y. and Sakkopoulos, E., 2008, Techniques to support Web Service selection and consumption with QoS characteristics, *Journal of Network and Computer Applications*, **31**(2), pp. 108-130.

Dovey, M., Kostadinov, I., Giddy, J., Green, P., Berry, D., Chohan, D. and Wang, X., 2005, UK Engineering Tasks Force Evaluation of UDDI for UK e-Science, Disponible sur : http://www.nesc.ac.uk/technical_papers/UKeS-2005-04.pdf.

Duarte-Amaya, H., 2007, Canevas pour la composition de services web avec propriétés transactionnelles, Université Joseph Fourrier-Grenoble 1, pp. 159.

Dumas, M., Benatallah, B. and Nezhad, H. R. M., 2008, Web Service Protocols: Compatibility and Adaptation, *IEEE Data Eng. Bull*, **31**(4), pp. 40-44.

Emig, C., Weisser, J. and Abeck, S., 2006, Development of SOA-Based Software Systems – an Evolutionary Programming Approach, In *the Advanced International Conference on Telecommunications and International Conference on Internet and Web Applications and Services (AICT/ICIW 2006)*, pp. 182-187.

Erl, T., 2005, *Service-Oriented Architecture (SOA): Concepts, Technology, and Design* Prentice Hall

Fauvet, M.-C., Duarte, H., Dumas, M. and Benatallah, B., 2005, Handling Transactional Properties in Web Service Composition, In *the 6th International Conference on Web Information Systems Engineering (WISE'05), New York City, New York*, pp. 273-289.

Favier, M. and Coat, F., 1997, Comment gérer une équipe virtuelle ?, In *Actes du 3ème colloque de l'AIM*.

Favier, M. and F, F. C., 1999, Le futur des systèmes d'information, *Revue Française de Gestion*, pp. 19-28.

Favre, J. M., 2004, Towards a Basic Theory to Model Driven Engineering, In *the 3rd Workshop in Software Model Engineering, WiSME 2004*.

Fensel, D., Bussler, C. and Maedche, A., 2002, Semantic Web Enabled Web Services, In *the 1st International Semantic Web Conference (ISWC 2002), Sardinia, Italy, June 2002*LNCS Springer.

Ferraiolo, D. and Kuhn, R., 1992, Role-based access control, In *15th NIST-NCSC National Computer Security Conference*.

Foster, H., Uchitel, S., Magee, J. and Kramer, J., 2003, Model-based Verification of Web Service Compositions In *the18th IEEE International Conference on Automated Software Engineering (ASE 2003)*Montreal, Canada, pp. 152-161.

Frankel, D. and Parodi, J., 2002, Using Model-Driven ArchitectureTM to DevelopWeb Services, Research Report in IONA Technologies PLC.

Frankel, D. S., 2003, *Model Driven Architecture: Applying MDA to Enterprise Computing*, Wiley and OMG Press.

Fujii, K. and Suda, T., 2006, Semantics-based Dynamic Web Service Composition, *the International Journal of Cooperative Information Systems (IJCIS), special issue on Service-Oriented Computing*, **15**(3), pp. 293-324.

Garcia, D. Z. G., Toledo, M. B. F. d. and 2006, Semantics-enriched QoS policies for web service interactions, In *the 12th ACM Brazilian symposium on Multimedia and the web*

Gebauer, J., 1996, Virtual organization from an economic perspective, In *the 4th European Conference on Information Systems*Lisbon, 2-4 July.

Gensollen, M., 2003, Biens informationnels et communautés médiatées, disponible en ligne sur : http://gdrtics.u-paris10.fr/pdf/ecoles/sept2003/01-01_gensollen.pdf.

Georgakopoulos, D., Schuster, H., Baker, D. and Cichocki, A., 2000, Managing escalation of collaboration processes in crisis mitigation situations, In *16th International Conference on Data Engineering (ICDE 2000), San Diego California, USA*, pp. 45-56.

Goldman, S. I. and Nagel, R. N., 1993, *Management, Technology and Agility: the Emergence of New Era in Manufacturing*, **8**(1-2), pp. 18-38.

Gong, L., 2005, Contextual modeling and applications, In *2005 IEEE International Conference on Systems, Man and Cybernetics*, Vol. 1, pp. 381- 386.

Grefen, P., Aberer, K., Hoffner, Y. and Ludwig, H., 2000, CrossFlow: Cross-Organizational Workflow Management in Dynamic Virtual Enterprises, *International Journal of Computer Systems Science & Engineering*, **15**(5), pp. 277-290.

Grefen, P., Ludwi, H., Dan, A. and Angelov, S., 2006, An analysis of web services support for dynamic business process outsourcing, *Information and Software Technology*, **48**(11), pp. 1115-1134.

Grefen, P., Mehandjiev, N., Kouvas, G., Weichhart, G. and Eshuis, R., 2009, Dynamic business network process management in instant virtual enterprises, *Computers in Industry*, **60**(2), pp. 86–103.

Gronmo, R., Skogan, D., Solheim, I. and Oldevik, J., 2004, Model-Driven Web Services Development, In *IEEE International Conference on e-Technology, e-Commerce and e-Service (EEE'04)* pp. 42-45.

Han, Y., Wang, J. and Zhang, P., 2009, Business-oriented service modeling: A case study, *Simulation Modelling Practice and Theory*, (Article in press).

Harel, D., 1987, Statecharts : A visual Formalism for Complex Systems, *Science of Computer Programming*, **8**(3), pp. 231-274.

Harrington, J., 1991, *Business Process Improvement: The Breakthrough Strategy for Total Quality, Productivity, and Competitiveness* McGraw-Hill.

He, H., 2003, What is Service-Oriented Architecture?, available at: http://webservices.xml.com/pub/a/ws/2003/09/30/soa.html.

Heckel, R., Kuster, J., Thoneand, S. and Voigt, H., 2003, Towards a UML Profile for Service-Oriented Architectures., In *Workshop on Model Driven Architecture: Foundations and Applications (MDAFA '03), University of Twente, Enschede, June 2003*.

Henderson-Sellers, B., Gonzalez-Perez, C. and Ralyté, J., 2007, Situational method engineering: chunks or fragments?, In *the CAiSE Forum*Trondheim, pp. 89-92.

Hirzalla, M., Cleland-Huang, J. and Arsanjani, A., 2008, A Metrics Suite for Evaluating Flexibility and Complexity in Service Oriented Architectures, In *the 6th International Conference on Service Oriented Computing, ICSOC Workshops 2008*, pp. 41-52.

Hmida, M. B., Boutrous-Saab, C., Haddad, S., Monfort, V. and Tomaz, R. F., 2007, Towards the Dynamic Adaptability of SOA, In *the Ninth International Conference on Enterprise Information Systems (ICEIS)*Funchal, Madeira, Portugal, pp. 474-479.

Hmida, M. B., Tomaz, R. F. and ValerieMonfort, 2005, Applying AOP Concepts to Increase Web Services Flexibility, In *the proceedings of the IEEE International Conference on Next Generation Web Services Practices (NWeSP 05)*Seoul, Korea, pp. 169-174.

Hofstede, G. J., Vermunt, A., Smits, M. and Noorderhaven, N., 1997, Wired international teams: experiments in strategic decision-making by multi-cultural virtual teams, In *the 5th European Conference on Information System*, pp. 321-336.

Hohpe, G. and Woolf, B., 2004, *Enterprise Integration Patterns: Designing, Building and Deploying Messaging Solutions*, Peason Education.

Hull, R., Neaves, P. and Bedford-Roberts, J., 1997, Towards Situated Computing, In *the First International Symposium on Wearable Computing (ISWC), October 1997*, pp. 146-153.

IBM, 2007a, Getting Started: WebSphere Process Server and WebSphere Enterprise Service Bus V6, available at:chttp://www.redbooks.ibm.com/abstracts/sg247378.html?Open.

IBM, 2007b, IBM Rational Developer for Visual Studio, available at: www-01.ibm.com/awdtools/developer/rose.

Indulska, M., Green, P., Recker, J. and Rosemann, M., 2009, Business Process Modeling: Perceived Benefits, In *the 28th International Conference on Conceptual Modeling*Springer, Gramado, Brazil.

Indulska, M., Recker, J., Green, P. and Rosemann, M., 2007, Are We There Yet? Seamless Mapping of BPMN to BPEL4WS, In *the 13th Americas Conference on Information Systems. Keystone, Colorado 2007*.

Intalio and BPMI, 2002, Bussiness process modeling language, available at http://www.bpmi.org/bpmi-downloads/BPML-SPEC-1.0.zip, 2002.

Izza, S., 2006, Intégration des systèmes d'information industriels: Une approche flexible basée sur les services sémantiques, Vol. Thèse de Doctorat L'École Nationale Supérieure des Mines de Saint-Étienne et de l'Université Jean Monnet, Saint-Étienne, pp. 375.

Izza, S., Vincent, L. and Burlat, P., 2005, Framework for Semantic Enterprise Integration, In *The INTEROP-ESA'05*Geneva, Switzerland, pp. 78-89.

Jackson, M., 1995, *Software Requirements and Specifications – A Lexicon of Practice, Principles and Pejudices*, Addison-Wesley.

Jacobson, I., 1993, *Le génie logiciel orienté objet*, Addison Wesley.

Jang, S. and Who, W., 2003, Ubi-UCAM: A Unified Context-Aware Application Model., In *the 4th International and Interdisciplinary Conference on Modeling and Using Context* Springer, Stanford, USA, pp. 178-189.

JBoss, 2006, JBoss AOP Web site availble at : http://www.jboss.org/developers/projects/jboss/aop.

JbossAOP, 2008, http://www.jboss.org/developers/projects/jboss/aop.

Juric, M. B., 2006 *Business Process Execution Language for Web Services Second Edition*, Packt Publishing Ltd.

Kalnins, A., Barzdins, J. and Celms, E., 2004, Basics of Model Transformation Language MOLA, In *Workshop on Model Driven Development (WMDD 2004) at ECOOP 2004*.

Katzy, B. and LÄoh, H., 2003, Virtual Enterprise Research State of the Art and Ways Forward, available at: http ://www.cetim.org/.

Keidl, M. and Kemper, A., 2004, A Framework for Context-Aware Adaptable Web Services (Demonstration), In *International Conference on Extending Database Technology (EDBT), Heraklion, Crete*.

Keller, G., Nuttgens, M. and Scheer, A. W., 1992, Semantische Prozessmodellierung auf der Grundlage, Ereignisgesteuerter Prozessketten (EPK), Wirtschaftsinformatik, Germany.

Kellert, P. and Toumani, F., 2006, Les services web sémantiques, *Revue I3 (Information-Interaction-Intelligence)*.

Kersten, M., 2004, Comparison of the leading aop tools In *University of British Columbia*.

Kersten, M., 2005, AOP Tools Comparison, available at: http://www.ibm.com/developerworks/library/j-aopwork1/#author1.

Ketfi, A., Belkhatir, N. and Cunin, P.-Y., 2002, Adaptation dynamique, concepts et expérimentations In *International Conference Software & Systems Engineering and their Applications, ICSSEA*Paris, France.

Kiczales, G., Hilsdale, E., Hugunin, J., Kersten, M., Palm, J. and Griswold, W., 2001a, Overview of AspectJ., In *the European Conference on Object Oriented Programming (ECOOP 01)*Budapest, Hungary.

Kiczales, G., Hilsdale, E., Hugunin, J., Kersten, M., Palm, J. and Griswold, W. G., 2001b, An Overview of AspectJ, In *The European Conference on Object-Oriented Programming (ECOOP)*.

Kiczales, G., Lamping, J., Mendhekar, A., Maeda, C., Lopez, C. and Irwin, J. M., 1997, Aspect-Oriented Programming, In *The European Conference on Object-Oriented Programming (ECOOP)*(Ed, 1241, S.-V. L.) Finland.

Kleppe, A., Warmer, J. and Bast, W., 2003, *MDA Explained: The Model Driven Architecture: Practice and Promise*, Addison-Wesley.

Krafzig, D., Banke, K. and Slama, D., 2004, *Enterprise SOA: Service-Oriented Architecture Best Practices* Prentice Hall.

Kritikos, K. and Plexousakis, D., 2006, Semantic QoS Metric Matching In *European Conference on Web Services (ECOWS'06)*, pp. 265-274.

Lamsweerde, A. V., Dairmont, R. and Massonet, P., 1995, Goal Directed Elaboration of Requirements for a Meeting Scheduler: Problems and Lessons Learnt, In *Proceedings of the Second IEEE International Symposium on Requirements Engineering*IEEE, pp 194-204, New York.

LeMoigne, J.-L., 1990, *La modélisation des systèmes complexes*, Afcet-systèmes, Dunod.
Leymann, F., 2001, Web Services Flow Language (WSFL 1.0), available at http://xml.coverpages.org/WSFL-Guide-200110.pdf.
Longépé, C., 2004, *Le projet d'urbanisation du SI. Démarche pratique avec un cas concret*, DUNOD, Paris.
Lopes, D. C. P., 2005, Étude et applications de l'approche MDA pour des plates-formes de Services Web
Vol. Thèse de Doctorat l'Université de Nantes, pp. 286.
Lopez-Sanza, M., Acuna, C. J., Cuesta, C. E. and Marcos, E., 2008, Modelling of Service-Oriented Architectures with UML, *Electronic Notes in Theoretical Computer Science*, **194**(4), pp. 23-37.
Lublinsky, B. and Tyomkin, D., 2003, Dissecting service oriented architectures, *Business Integration Journal* pp. 52-58.
Luik, J., 1996, L'entreprise virtuelle, In *Conférancier invité (Forum organisé par le ministre du développement économique, du commerce et du tourisme en Ontario), disponible sur : www.2ontario.com/archives/chfall96*
Lupu, E., Milosevic, Z. and M, M. S., 1999, Use of Roles and Policies for Specifying and Managing a Virtual Enterprise, In *the 9th International Workshop on Research Issues on Data Engineering : Information Technology for Virtual Enterprises*, pp. 273-286.
Maamar, Z., Benslimane, D., Thiran, P., Ghedira, C., Dustdar, S. and Sattanathan, S., 2007a, Towards a context-based multi-type policy approach for Web services composition, *Data & Knowledge Engineering, Elsevier Science Publisher*, **62**(2).
Maamar, Z., Lahkim, M., Benslimane, D., Thiran, P. and S, S., 2007b, Web Services Communities Concepts & Operations, In *The WEBIST'2007, Barcelona, Spain*.
Majithia, S., Shaikhali, A., Rana, O. and Walker, D. W., 2004, Reputation-based Semantic Service Discovery, In *Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises (WET ICE'04)*.
Martin, D., Paolucci, M., McIlraith, S., Burstein, M., McDermott, D., McGuinness, D., Parsia, B., Payne, T., Sabou, M., Solanki, M., Srinivasan, N. and Sycara, K., 2004, Bringing Semantics to Web Services: The OWL-S Approach, In *SWSWPC*(Eds, Cardoso, J. and Sheth, A.).
Martinez, M. T., Fouletier, P., Park, K. H. and Favrel, J., 2001, Virtual enterprise -- organisation, evolution and control, *International Journal of Production Economics (IJPE)*, **74**(1-3), pp. 225-238.
Maximilien, E. M. and Singh, M. P., 2004a, A Framework and Ontology for Dynamic Web Services Selection, *IEEE Internet Computing*, **8**(5), pp. 84 - 93
Maximilien, E. M. and Singh, M. P., 2004b, Toward Autonomic Web Services Trust and Selection, In *International Conference On Service Oriented Computing* pp. 212-221
McCoy, D. W. and Plummer, D. C., 2006, Defining, Cultivating and Measuring Enterprise Agility, *Gartner Research*.
Medjahed, B. and Bouguettaya, A., 2005, A Dynamic Foundational Architecture for Semantic Web Services, *Distributed and Parallel Databases*, **17**(2), pp. 179–206.
Medjahed, B., Bouguettaya, A. and Elmagarmid, A. K., 2003, Composing Web services on the Semantic Web, *Very Large Data Base*, **12**(4), pp. 333-351.
Meissonier, R., 2000, Organisation virtuelle : conceptualisation, ingénierie et pratiques, enquête auprès des PME de la région des pays de la Loire, Institut d'Administration des Entreprises Aix-en-Provence, pp. 368.
META, 2003, META Group Research on Requirements Realization and Relevance, report, 2003.
Milanovic, N. and Malek, M., 2004, Current Solutions for Web Service Composition, *IEEE Internet Computing*, **8**(6), pp. 51-59.
Mili, H., Jaoude, G. B., Lefebvre, É., Tremblay, G. and Petrenko, A., 2004, Business process modeling languages : Sorting through the alphabet soup, Research Report, Département d'Informatique, UQAM.

Miller, J. and Mukerji, J., 2003, MDA Guide Version 1.0.1, available at: http://www.omg.com/mda/03-06-01.pdf.
Moitraa, D. and Ganeshb, J., 2005, Web services and flexible business processes: towards the adaptive enterprise, *Information and Management*, **42**(pp. 921-933.
Morley, C., 2000, Changement organisationnel et Modélisation des processus, In *5éme congrés de l'Association Information et Management(AIM)*.
Morley, C., Hugues, J., Leblanc, B. and Hugues, O., 2005, *Processus métiers et S.I : évaluation, modélisation, mise en oeuvre*, DUNOD.
Muller, P.-A. and Gaertner, N., 2004, *Modélisation Objet avec UML*.
Newcomer, E., 2002, *Understanding Web service XML, WSDL, SOAP et UDDI*, Addison-Wesley Professional.
Nilsson, N. J., 1971, *Problem Solving Methods in Artificial Intelligence*, McGraw Hill.
Nurcan, S., 2008, A survey on the flexibility requirements related to business processes and modeling artifacts, In *the 41st Annual Hawaii International Conference on System Sciences*Big Island, Hawaii, USA, 7-10 January.
OASIS, 2006, Reference Model for Service Oriented Architecture 1.0, available at: http://www.oasis-open.org/committees/download.php/19679/soa-rm-cs.pdf.
OGSA, 2005, Glossary of Terms, available at: http://www.ggf.org/documents/GWD-I-E/GFD-I.044.pdf.
OMG, 2002, Request for Proposal: MOF 2.0 Query/Views/Transformations RFP, Disponible sur http://www.omg.org/docs/ad/02-04-10.pdf.
OMG, 2003a, MDA guide version 1.0.1, In *Object Management Group*.
OMG, 2003b, MDA guide version 1.0.1, proposed by the Object Management Group.
OMG, 2005, Business Motivation Model - Business governance in a volatile world, available at : http://www.businessrulesgroup.org/second_paper/BRG-BMM.pdf
OMG, 2006a, OMG, Meta Object Facility Core Specification, version 2.0. January, 2006.
OMG, 2006b, OMG. Object Constraint Language (OCL) Specification, v 2.0, May 2006.
OMG, 2007, Object Management Group, UML Version 2.1.1 Infrastructure. February.
Orriëns, B., Yang, J. and Papazoglou, M. P., 2003, ServiceCom: A Tool for Service Composition Reuse and Specialization, In *4th IEEE International Conference onWeb Information Systems Engineering (WISE2003)*, pp. 355–358.
Orriëns, B., Yang, J. and Papazoglou, M. P., 2004, Service Component: a mechanism for web service composition reuse and specialization, *Journal of Integrated Design and Process Science*, **8**(2), pp. 13-28.
Ouyang, C., Dumas, M., Hofstede, A. H. M. t. and Aalst, W. M. P. v. d., 2006, From BPMN Process Models to BPEL Web Services. ICWS 2006: 285-292
In *the IEEE International Conference on Web Services*Chicago, USA.
Pallos, M., 2001, Service Oriented Architecture: A Primer, *EAI Journal, December 2001*.
Paolucci, M., Kawamura, T., Payne, T. R. and Sycara, K. P., 2002, Semantic matching of Web services capabilities, In *the First International Semantic Web Conference on The Semantic Web* Springer-Verlag, pp. 333-347
Papazoglou, M. P., 2003, Service oriented computing: concepts, characteristics and directions, In *the 4th IEEE International Conference on Web Information Systems Engineering*, Italy, pp. 3-12.
Papazoglou, M. P., 2007, What's in a Service? , In *First European Conference*(Ed, Oquendo, F.) Lecture Notes in Computer Science , Springer Aranjuez, Spain, pp. 11-28.
Papazoglou, M. P. and Georgakopoulos, D., 2003, Service oriented computing, *Communications of the ACM* **46**(10), pp. 24-28.
Papazoglou, M. P. and Heuvel, W.-J. v. d., 2006, Service-oriented design and development methodology, *International Journal of Web Engineering and Technology (IJWET)*, **2**(4), pp. 412-442.
Pascoe, J., 1997, The stick-e note architecture : extending the interface beyond the user, In *the 2nd international conference on Intelligent user interfaces*ACM Press, New York, NY, USA, pp. 261-264.
Petri, C., 1962, Kommunikation mit Automaten, University of Bonn, Germany.

Phelizon, R. and Rouhier, S., 2002, Alignement stratégique du système d'information, available at : http://cigref.typepad.fr/cigref_publications/RapportsContainer/Parus2002/2002_-_Alignement_strategique_du_systeme_d_information_web.pdf, In *Rapport Cigref*.

PRAXEME, 2008, Praxeme méthode d'entreprise, disponible sur: http://www.praxeme.org/.

PyUT, 2007, Python UML Tool, available at: http://pyut.sourceforge.net/index.php

Qiu, L., Shi, Z. and Lin, F., 2006, Context Optimization of AI planning for Services Composition, In *the IEEE International Conference on e-Business Engineering (ICEBE 2006), October 2006, Shangai, China*.

Qiu, L., Shi, Z., Lin, F. and Shi, Z., 2007, Context Optimization of AI Planning for Semantic Web Services Composition, *Service Oriented Computing and Applications,* **1**(2), pp. 117-128.

Rahmani, A. T., Rafe, V., Sedighian, S. and Abbaspour, A. (2006) In *Computational Science – ICCS 2006*(Ed, Science, L. N. i. C.), pp. 578-585.

Rajsiri, V., Lorre, J.-P., Benaben, F. and Pingaud, H. (2007) In *Enterprise Interoperability II - New challenges and approaches*(Eds, Goncalves, R. J., Müller, J. P., Mertins, K. and Zelm, M.) Springer, pp. 257-260.

Ralyté, J., 2001, Ingénierie des méthodes à base de composants, Thèse de doctorat en informatique, Université Paris 1, France

Ramollari, E., Dranidis, D. and Simons, A. J. H., 2007, A Survey of Service Oriented Development Methodologies, In *The 2nd European Young Researchers Workshop on Service Oriented Computing*University of Leicester, UK

Rolland, C., 2009, Method engineering: towards methods as services, *Software Process: Improvement and Practice,* **14**(3), pp. 143-164.

Rolland, C. and Kaabi, R. S., 2007, An Intentional Perspective to Service Modeling and Discovery, *Computer Software and Applications Conference,COMPSAC 2007,* **2**(24-27), pp. 455-460.

Rolland, C., Prakash, N. and Benjamen, A., 1999, A multi-Model View of Process Modelling, *Requirements Engineering,* **4**(4), pp. 169-187.

Roman, D., Keller, U., Lausen, H., Bruijn, J. d., Lara, R., Stollberg, M., Polleres, A., Feier, C., Bussler, C. and Fensel, D., 2005, Web Service Modeling Ontology, *1,* **1**(pp. 77-106.

Rosemann, M. and Recker, J. C., 2006, Context-aware Process Design: Exploring the Extrinsic Drivers for Process Flexibility, In *18th International Conference on Advanced Information Systems Enginnering. Proceedings of Workshops and Doctoral Consortium*, Luxembourg, 5-9 Juin.

Ross, D., 1977, Structured Analysis (SA): A Language for communicating ideas, *IEEE Transactions on Software Engineering,* **3**(1), pp. 16-34.

Rumbaugh, J., Blaha, M., Premerlani, W., Eddy, S. and Lorensen, W., 1991, *Object-Oriented Modeling and Design*, Englewood Cliffs NJ Prentice Hall.

Ryan, N., 2006, ConteXtML : Exchanging contextual information between a mobile client and the fieldnote server.

Ryan, N., Pascoe, J. and Morse, D., 1997, *Enhanced Reality Fieldwork : the Context-Aware Archeological Assistant*.

Sabherwal, R. and Chan, Y. E., 2001, Alignment Between Business and IS Strategies: A Study of Prospectors, Analyzers, and Defenders, *Information Systems Research,* **12**(1), pp. 11-33.

Saha, D. and Mukherjee, A., 2003, A Pervasive Computing: a Paradigm for the 21st Century, *Computer,* **36**(3), pp. 25-31.

Saidani, O. and Nurcan, S., 2007, Towards Context Aware Business Process Modelling, In *Workshop on Business Process Modelling, Development, and Support (BPMDS), Held in conjunction with 19th International Conference on Advanced Information Systems Engineering (CAiSE'07)*Trondheim, Norway, June 2007.

Sánchez, N. G., Zubiaga, D. A. G., Atahualpa, J., González, I. and Molina, A. (2005) In *Information Control Problems in Manufacturing* Vol. 12 (Eds, Dolgui, A., Morel, G. and Pereira, C.).

SAP, 2006, SAP NetWeaver: solution overview, available at: http://www.sap.com/platform/netweaver/pdf/BWP_OV_SAP_NetWeaver.pdf.

Schilit, B., Theimer, M. and Welch, B., 1993, Customizing Mobile Application In *the USENIX Symposium on Mobile and Location-independent Computing*Cambridge, US, pp. 129-138.

Schilit, B. N. and Theimer, M. M., 1994, Disseminating active map information to mobile hosts, *Network, IEEE*, **8** (5), pp. 22 - 32.

Schlenoff, C., Gruninger, M., Tissot, F., L.Valois and Lubell, J., 1999, The Process Specification Language (PSL): Overview and Version 1.0 Specification,, In *NIST Internal Report (NISTIR) 6459, National Institute of Standards and Technology, Gaithersburg, MD, USA*, .

Schmidt, A., Aidoo, K. A., Takaluoma, A., Tuomela, U., Laerhoven, K. V. and Velde, W. V. d., 1999, Advanced Interaction in Context, In *the 1st international symposium on Handheld and Ubiquitous Computing*(Ed, Springer) London, UK, pp. 89-101.

Sheng, Q. Z. and Benatallah, B., 2005, ContextUML : A UML-Based Modeling Language for Model-Driven Development of Context-Aware Web Services, In *the 4th International Conference on Mobile Business (ICMB'05), IEEE Computer Society*Sydney, Australia, pp. 11-13.

Sheng, Q. Z., Benatallah, B., Dumas, M. and Mak, E. O.-Y., 2002 SELF-SERV: A Platform for Rapid Composition of Web Services in a Peer-to-Peer Environment, In *the 28th International Conference on Very Large Data Bases (VLDB 2002), August 2002*(Ed, Kaufmann, M.) Hong Kong, China, pp. 1051-1054.

Sheng, Q. Z., Benatallah, B., Maamar, Z. and I, A. H. H. N., 2009, Configurable Composition and Adaptive Provisioning of Web Services, *IEEE Transactions on Services Computing (TSC)*, **2**(1), pp. 34-49.

Sieber, P. and Griese, J., 1998, Virtual organizing an a strategy for the "Big Six" to stay competitive in a global market, *Journal of Strategic Information Systems*, **7**(3), pp. 167-181

Singh, E. M. M. a. M. P., 2004, A framework and ontology for dynamic Web services selection, *IEEE Internet Computing*, **8**(5), pp. 84-93, 2004.

Skogan, D., Gronmo, R. and Solheim, I., 2004, Web Service Composition in UML, In *the 8th IEEE International Enterprise Distributed Object Computing Conference (EDOC 2004)*, pp. 47-57.

Smith, B., 1982, Procedural reflection in programming languages, PhD thesis, Massachussetts Institute of Technimogy, USA.

Sneed, H. M., 2006, Integrating legacy Software into a Service oriented Architecture, In *Conference on Software Maintenance and Reengineering (CSMR'06)*.

Spohrer, J., Maglio, P. P., Bailey, J. and Gruhl, D., 2007, Steps Toward a Science of Service Systems, *IEEE Computer Society*, **40**(1), pp. 71-77.

Sprott, D. and Wilkes, L., 2004, Understanding Service-Oriented Architecture, available at: http://msdn2.microsoft.com/en-us/library/Aa480021.aspx.

Stang, T. and Linnhoff-Popoen, C., 2004, A Context Modeling Survey, In *In Workshop on Advanced Context Modelling, Reasoning and Management as part of UbiComp 2004 - The Sixth International Conference on Ubiquitous Computing*Nottingham/England.

Steinmueller, E., 2002, *Virtual communities and the New Economy*.

Suraci, V., Mignanti, S. and Aiuto, A., 2007, Context-aware Semantic Service Discovery, In *the 16th IST Mobile and Wireless Communications Summit*Budapest, Hungary.

Sycara, K. P., Paolucci, M., Ankolekar, A. and Srinivasan, N., 2003, Automated discovery, interaction and composition of semantic web services, Web Semantics: Science, Services and Agents, In *World Wide Web*, pp. 27-46.

Taher, Y., Benslimane, D., Fauvet, M.-C., Maamar., Z. and 2006, Towards an Approach for Web Services Substitution, In *the 10th IEEE International Database Engineering and Applications Symposium*India, pp. 166-173.

Thakkar, S., Ambite, J. L. and Knoblock, C. A., 2004, A Data Integration Approach to Automatically Composing and Optimizing Web Services, In *International Workshop on Planning and Scheduling for Web and Grid Services*.

Thakkar, S., Ambite, J. L., Knoblock, C. A. and Shahabi, C., 2002, Dynamically Composing Web Services from On-line Sources, In *AAAI Workshop on Intelligent Service Integration*1-7.

Thatte, S., 2001, XLANG: Web services for business process design, available at: http://www.gotdotnet.com/team/xml_wsspecs/xlang-c/default.htm, 2001.

Thisse, D., 2004, Simplifier la supply chain : une obligation incontournable devenue un vrai enjeu stratégique, In *Conférence Carrefours Logistique, 12-14 Octobre 2004, Paris*, pp. 11-12.

Tigli, J.-Y. and Lavirotte, S., 2006, Adaptation dynamique à l'environnement d'exécution : un enjeu pur l'informatique mobile et ambiante, In *Séminaire, Grenoble, 2006*

Together, 2008, Borland Together, available at: http://www.borland.com/products/downloads/download_together.html.

Tomaz, R. F., Hmida, M. B. and Monfort, V., 2006, Concrete Solutions for Web Services Adaptability Using Policies and Aspects, *International Journal of Cooperative Information Systems (IJCIS)*, **15**(3), pp. 415-438.

Touzi, J., 2007, Aide à la conception de Système d'Information Collaboratif support de l'interopérabilité des entreprises, In *Centre de Génie Industriel*Ecole des Mines d'Albi Carmaux, pp. 248.

Travica, B., 1997, The Design of the Virtual Organization: A research Model, In *the Association for Information Systems, Americas conference* Indianapolis, 15-17 August.

Umberto, 2008, How to run ActiveBPEL 4.1 on Jboss 4.2, available at: http://umberto.tiani-spirit.com/blog/2007/10/howto-run-activebpel-41-on-jboss-42.html.

Upton, D. and Affee, A. M., 1996, The real virtual factory, *Harvard Business Review*, **74**(4), pp. 123-133.

Vernadat, F., 1999, *Techniques de modelisation en entreprise: Application aux processus opérationnels*, Economica.

Vernadat, F. (2007) In *Service Enterprise Integration, Springer US*Springer, pp. 77-101.

Vernadat, F. and Hamaidi, L., 1998, *La modélisation en entreprise : Méthodes descriptives des processus opérationnels*.

Villanova-Oliver, M., 2002, Adaptabilité dans les systèmes d'information sur le Web : Modélisation et mise en oeuvre de l'accès progressif, thèse de doctorat, Institut National Polytechnique de Grenoble, pp. 238.

W3C, 2004a, OWL Web Ontology Language Overview Published online at http://www.w3.org/TR/owl-features/.

W3C, 2004b, Web Services Glossary, available at: http://www.w3.org/TR/ws-gloss.

W3C, 2006, Web Services Policy Attachment, available at http://www.w3.org/Submission/WS-PolicyAttachment.

W3C, 2009, The World Wide Web Consortium, available at: http://www.w3.org/.

Wang, X. H., Zhang, D. Q., Gu, T. and Pung, H. K., 2004a, an Ontology Based Context Model in intelligent Environment, In *Communication networks and Distributed System modeling and simulation Conference (CNDS 2004)*San Diego, Clalifornia, USA, pp. 270-275.

Wang, X. H., Zhang, D. Q., Gu, T. and Pung, H. K., 2004b, Ontology Based Context Modeling and Reasoning using OWL, In *In PerCom Workshops*, pp. 18-22.

Waqar, S. and Racca, F., 2004, *Business services orchestration: The hypertier of information technology*, Cambridge University Press.

Weiser, W., 1991, The computer of the 21st century, *Scientific American*, **3**(265).

Westerinen, A., 2001, Rfc 3198 : Terminology for policy-based management, available at: http://www.codes-sources.com/rfc.aspx?rfc=3198.

WFMC, 1999, Workflow Management Coalition Interface 1: Process Definition Interchange Process Model, available at: http://www.wfmc.org/standards/docs/TC-1016-P_v11_IF1 Process_definition_Interchange.pdf.

Willink, E. D., 2003, UMLX - A Graphical Transformation Language for MDA. Workshop on Model, In *Driven Architecture Foundations and Applications*, pp. 13-24.

Winograd, T., 2001, Architectures for Context, *Human Computer Interaction Journal*, **6**(2&3), pp. 401-419.

Wishart, R., Robinson, R., Indulska, J. and Josang, A., 2005, SuperstringRep:Reputation-enhanced Service Discovery In *the 28 Australasian conference on Computer Science*, Vol. 38, pp. 49-57.

Wohed, P., Aalst, W. M. P. v. d., Dumas, M. and Hofstede, A. H. M. t. (2003) In *Lecture Notes in Computer Science*, Vol. 2813/2003 (Ed, Heidelberg, S. B.).

Xiaohang, W., 2003, The context gateway : a parvesive computing infrastructures for context aware services, In *Technical report, National university of singapore*.

Yu, E., 1997, Towards Modelling and Reasoning Support for Early-Phase Requirements Engineering

In *the 3rd IEEE International Symposium On Requiorements Engineering (RE'97)* , Washington D.C., USA.

Yu, E. and Mylopoulos, J., 1994, Using goals, rules and methods to support reasoning in Business Process Reengineering, In *27th Hawaii International Conference on System Sciences*, Maui, Hawaii.

Zaidat, A., 2005, Spécification d'un cadre d'ingénierie pour les réseaux d'organisations, Thèse de Doctorat à l'École Nationale Supérieure des Mines de Saint-Étienne et de l'Université Jean Monnet, pp. 258.

Zave, P. and Jackson, M., 1997, Four Dark Corners of Requirements Engineering, *ACM Transaction on Software Engineering and Methodology (TOSEM)*, **6**(1), pp. 1-30.

Zhang, X. G., 2008, Model Driven Data Service Development In *IEEE International Conference on Networking, Sensing and Control, ICNSC 2008*China, pp. 1668-1673.

Zhao, J. L. and Cheng, H. K., 2005, Web services and process management: a union of convenience or a new area of research? , *Decision Support Systems*, **40**(1).

Zimmermann, O., Krogdahl, P. and Gee, C., 2004, Elements of Service-Oriented Analysis and Design available at: http://www.ibm.com/developerworks/library/ws-soad1/.

Annexe A

Étude de cas et Validation

Cette partie décrit l'étude de cas réalisée ainsi que le prototype développé.

1. Étude de cas

Cette section décrit une étude de cas qui permet d'appréhender, par la pratique, les différentes phases de la démarche CSOMA. La première partie de cette section est consacrée à la présentation du contexte dans lequel évolue l'entreprise ETS (*Enterprise Training Solutions*). Quant à la deuxième, elle mettra en évidence l'application de la démarche CSOMA au système d'information de l'ETS. Ce qui permet en retour de tester l'applicabilité et la pertinence de la démarche scientifique dans le domaine industriel.

2. Présentation de l'étude de cas

L'ETS (*Enterprise Training Solutions*) est une PME spécialisée dans la planification et l'exécution des sessions de formation du personnel de ses clients. Les clients sont notamment les gouvernements, les organismes publics et les entreprises.

L'entreprise ETS est considérée comme un acteur majeur du secteur de formation informatique. Elle propose des formations sur site individualisées ou pour grand groupe (30 à 40 personnes). Les formations sont organisées par sessions. Ces dernières couvrent des cours sur les logiciels de la gamme Microsoft, les logiciels de gestion, et la création de sites Web. Le niveau va de débutant à utilisateur avancé.

Outre les formations informatiques, ETS propose aux entreprises la vente de matériels et logiciels informatiques. Ainsi, elle se charge de l'acquisition du matériel et de la livraison jusqu'aux entrepôts de ses clients.

Les deux dernières années ont marqué une vraie expansion des activités de l'ETS. En effet, le nombre de clients ne cesse de croître accompagné par une augmentation du nombre de formateurs et de personnels de l'entreprise. Cependant, le système d'information de l'ETS se trouve incapable de suivre cette évolution et la majorité du travail s'effectue à travers des appels téléphoniques et des feuilles Excel. Ainsi plusieurs problèmes émergent tels que : les

problèmes de chevauchement de sessions, les problèmes d'affectation de formateurs, ou l'augmentation du temps de réponse à une demande client et aussi la perte de certains clients. Ces défaillances ont été préjudiciables à l'image de l'entreprise, en particulier vis-à-vis de ses clients et de ses partenaires. Afin d'y remédier, les directions métier et la direction du système d'information proposent d'améliorer le système d'information de l'ETS, et ce en apportant des réponses crédibles aux problèmes et aux besoins en pleine explosion. Bien évidement, ces acteurs s'attendent à une avancée de l'entreprise afin d'effectuer un bond dans les performances. Pour ce faire, ils ont défini un programme qui s'étale sur quelques mois et prend en compte :

- **Des considérations techniques :** il s'agit d'éviter l'effet de refonte globale du système d'information et aussi d'utiliser au maximum le patrimoine existant. Ce dernier englobe quelques applications dans des technologies diverses et avec une gestion de données et des flux complexes. L'application la plus importante est celle qui effectue la planification des formations adressées aux personnels de l'ETS afin d'actualiser leurs compétences en termes de certains logiciels ou outils informatiques. La direction du système d'information de l'ETS estime que cette application est importante et que certaines fonctionnalités peuvent être réutilisées pour résoudre les problèmes actuels.
- **Des considérations fonctionnelles :** face à d'importants enjeux de croissance, d'amélioration de la productivité, l'ETS souhaite que ses processus métiers soient plus flexibles afin de supporter les évolutions stratégiques de l'entreprise telle que la participation à des scénarios de coopération avec d'autres entreprises du domaine. En effet, ETS juge qu'il est parfois intéressant de coopérer avec d'autres entreprises pour répondre à un besoin d'un client auquel elle est incapable de répondre toute seule. De plus, l'ETS a pris conscience qu'Internet peut lui permettre de trouver les meilleurs fournisseurs et les clients les plus intéressants. Ainsi, l'entreprise souhaite avoir un site Web qui lui permet d'exposer les sessions de formation qu'elle propose et de permettre aussi à ses clients de commander en ligne des sessions de formation.

  L'objectif est donc de mettre en place un nouveau système flexible répondant à une architecture conforme aux standards informatiques. La direction du système d'information est bien consciente de l'importance de l'architecture orientée services pour son système d'information.

  La mission consiste donc à appliquer la démarche CSOMA afin de rénover le système d'information et de proposer une architecture flexible qui résout les problèmes de l'ETS.

3. Application de la démarche CSOMA

3.1 Phase 1 : Étude de l'existant

- **Étape 1 : Élaboration du ou des modèles de motivation métier de l'ETS**

La première étape consiste à élaborer le modèle de motivation métier (*BMM : Business Motivation Model*) selon le standard OMG. Rappelons que les concepts fondateurs du BMM sont : les finalités et les moyens. Les premiers englobent la vision, les buts et les objectifs et les seconds rassemblent la mission, les stratégies et les tactiques. Rappelons aussi qu'une vision est l'image de l'entreprise dans le futur sans se préoccuper de « comment l'atteindre ».

Un but exprime un état pour satisfaire la vision et un objectif constitue une manière mesurable, atteignable avec une certaine référence temporelle afin de satisfaire un but. Quant à la mission, elle décrit les activités opérationnelles de l'entreprise qui permettent à la vision de devenir une réalité. La mission est projetée au moyen des stratégies qui représentent les actions nécessaires pour réaliser les buts. Les stratégies sont mises en œuvre grâce aux tactiques.

En examinant les considérations fonctionnelles et en interviewant les directeurs métiers de l'ETS, les responsables métiers peuvent construire le ou les modèles de motivation métier. Deux modèles de motivation métier sont élaborés (Figure A.1). Chacun de ces modèles correspond à une vision que l'entreprise souhaite réaliser à long terme.

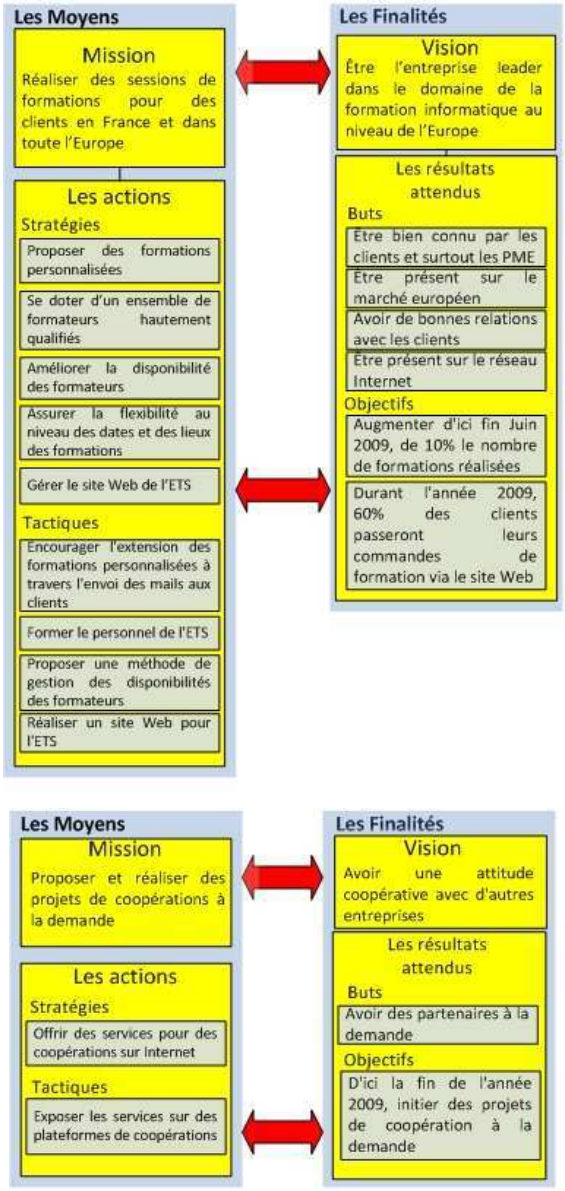


Figure A.1. Modèles de motivation métier de l'ETS

- **Étape 2 : Étude de l'architecture d'entreprise existante et cible**

L'architecture d'entreprise (existante ou cible) constitue le socle fondamental supportant l'ensemble de la démarche CSOMA. Afin de cerner cette architecture, il est nécessaire de procéder à l'élaboration d'un ensemble de cartographies dont la réalisation nécessite une attention particulière. La réalisation des cartographies relatives à l'architecture d'entreprise se décline en deux parties complémentaires qui représentent les deux axes de modélisation du système d'information à savoir : la modélisation des processus et la modélisation du système informatique. La principale raison de distinguer ces deux approches est qu'elles font intervenir des acteurs et des méthodes de modélisation différentes.
Lors de la réalisation de la cartographie des processus, nous avons constaté que l'entreprise ne possédait pas de culture de gestion de processus et ne possédait que peu de formalisations de processus. Par conséquent, nous avons eu recours à la technique d'interviews des responsables métiers impliqués dans les processus. La conduite des interviews est une condition indispensable pour assurer la qualité de la cartographie. Certains processus sont extraits à partir des outils informatiques déjà mis en place dans l'entreprise. D'autres sont directement formalisés à partir des interviews des acteurs impliqués. La cartographie des processus métiers de l'ETS est présentée dans la Figure A.2. L'objectif de cette cartographie est l'énumération de l'ensemble des processus, que ce soit les processus de management, les processus opérationnels, ou encore les processus de support.

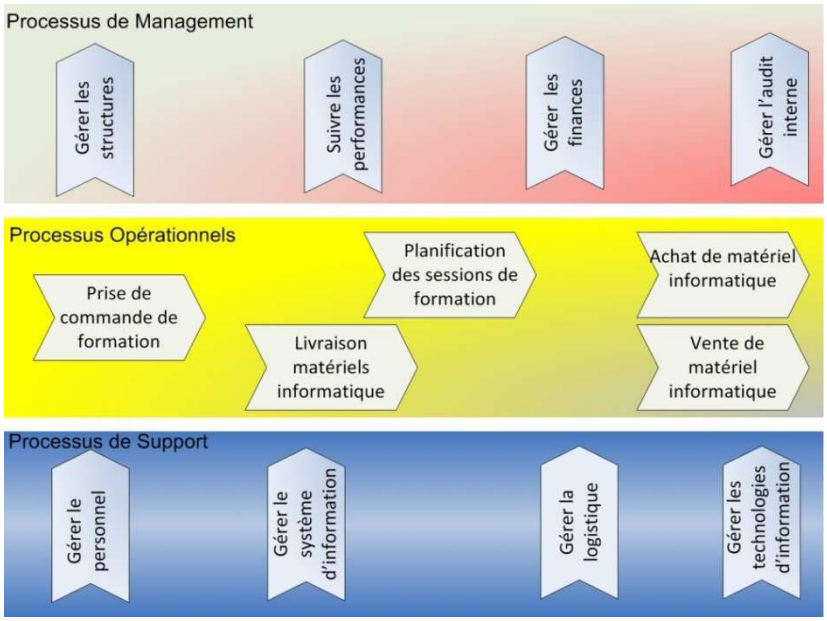


Figure A.2. Cartographie des processus métiers de l'entreprise ETS

Pour des raisons de simplicité nous allons nous concentrer sur les deux processus phares : le processus de prise de commande client et le processus de planification des sessions de formation. À l'état actuel des choses, le processus de prise de commande est un processus manuel qui s'effectue à travers des appels téléphoniques échangés entre le service relation client et le client lui-même. La même constatation est valable pour le processus de

planification des sessions, sauf que l'ETS utilise un logiciel de planning pour récupérer la disponibilité des formateurs et accéder à leurs informations à partir d'une base de données.
La cartographie applicative de l'ETS est représentée par la Figure A.3. Elle permet de recenser les applications existantes dans le système informatique ainsi que l'ensemble des flux d'information échangés entre ces applications. L'identification des applications est réalisée grâce aux documents formalisés lors de la conception et par les équipes assurant le maintien de ces applications. Comme nous pouvons le remarquer, à part l'application de gestion de formations du personnel et l'application de gestion d'approvisionnement en matériel informatique, les autres applications existantes sont des applications supports telles que : les applications de gestion de paie, de facturation, de comptabilité des achats, *etc*.

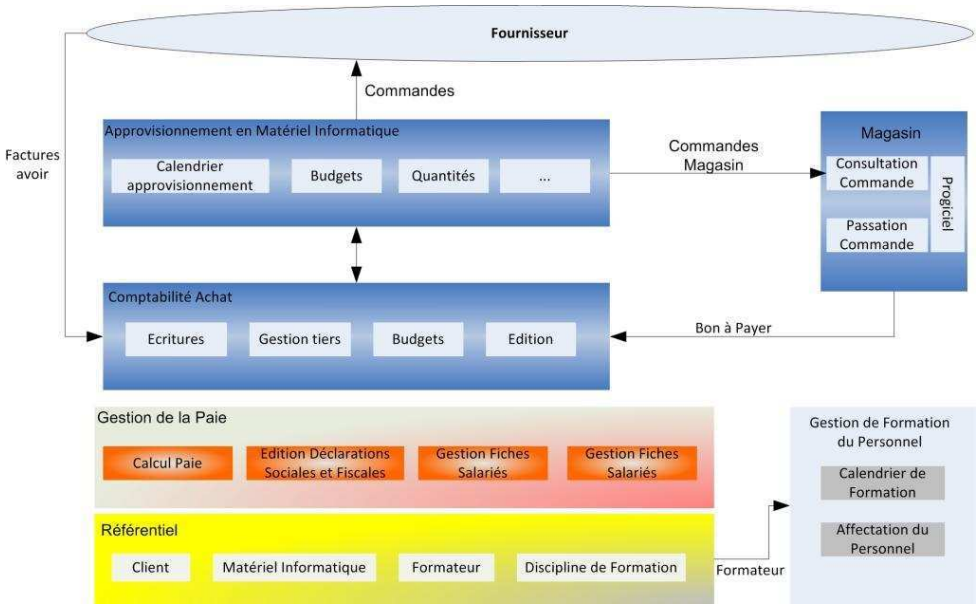


Figure A.3. Cartographie applicative de l'entreprise ETS

Pour soutenir les objectifs et les motivations métier, il est nécessaire de s'efforcer de dégager l'ensemble des axes d'améliorations possibles pour l'architecture métier et l'architecture applicative. Notre étude se consacre essentiellement sur les deux processus métiers déjà mentionnés. Les axes d'amélioration de l'architecture métier sont :
- L'automatisation du processus de prise de commande et l'intégration de la demande de formation à travers le site Web de l'ETS.
- L'exposition des deux processus en des services afin de favoriser la participation à des scénarios de coopération à la demande.

Concernant l'architecture applicative, la direction du système d'information de l'ETS a décidé de garder son patrimoine applicatif et de procéder uniquement à la construction des façades homogènes au-dessus de ces applicatifs. De plus, elle a jugé important de conserver l'application de gestion des formations continue de son personnel.
En somme, les données que nous avons pu recueillir nous ont permis de mettre en œuvre deux types de cartographies. L'analyse de ces cartographies par rapport aux motivations métier a souligné un ensemble d'axes d'amélioration. L'usage de ces cartographies et des axes d'amélioration (architecture cible) seront étudiés dans la suite de la démarche CSOMA.

3.2 Phase 2 : Construction de la SOA métier

Nous allons appliquer dans ce qui suit la phase de construction de la SOA métier sur le cas d'étude de l'entreprise ETS. Nous rappelons que la construction de la SOA métier doit partir de la définition (ou de la formalisation) des processus métiers afin de définir les services nécessaires à la réalisation de ces processus.

- **Étape 1 : Modélisation contextuelle des processus métiers**

Après avoir sélectionné les deux processus phares pour cette étude, nous allons maintenant procéder à leur modélisation. Cette modélisation tient compte de la réalité organisationnelle de l'ETS et intègre la connaissance relative au contexte dans lequel évoluent les processus. Dans ce qui suit, nous allons tout d'abord présenter une description détaillée de chaque processus métier. Ensuite, nous allons identifier les informations contextuelles relatives à chaque processus et enfin, nous allons procéder à leur modélisation en utilisant le standard BPMN (*Business Process Modeling Notation*).

─ Description du processus de prise de commande client

Le processus cible de prise de commande débute par une réception d'une commande client effectuée à partir du site Web de l'ETS. Elle comporte l'identité du client, la date de début, la date de fin de la formation, le lieu de la formation, le nombre de personnes à former, le nombre de formateurs désirés et la discipline de la formation. Tout d'abord il s'agit de vérifier les informations de la commande. Une fois vérifiée, la commande est transmise au service financier qui se charge de vérifier la solvabilité du client. Par la suite, le service du personnel cherche des formateurs pour la formation en question. S'il existe des formateurs pour la discipline choisie, le service du personnel vérifie la disponibilité des formateurs. Si les heures de disponibilité des formateurs sont supérieures ou égales au nombre d'heures demandées par le client, alors la commande client est acceptée et sera envoyée au service de planification des sessions. Dans le cas inverse, un message sera envoyé au responsable de formations afin de le mettre au courant de la décision de refus d'une commande client suite à l'indisponibilité des formateurs.

─ Modélisation contextuelle du processus de prise de commande client

Avant de procéder à la modélisation du processus, les intervenants métiers de l'ETS se chargent de définir les informations contextuelles qui peuvent influencer le comportement du processus. Ces informations sont importantes lors de l'identification des services métier adaptables.
Conformément à l'ontologie de catégorisation de contexte (cf. chapitre 2, section 2.4.2, étape 1), le processus de prise de commande client possède :
- Un contexte environnemental : le processus de prise de commande se déclenche à chaque réception d'une commande client et ceci pendant les horaires de travail de l'entreprise. Au-delà de ces horaires, le processus ne doit pas être déclenché.
- Un contexte fonctionnel : cette catégorie de contexte traite des paramètres qui sont en relation directe avec le métier du processus (la validation de la commande client). Les intervenants métiers de l'ETS renseignent trois sous catégories de base à savoir : le contexte du rôle, les règles métier et le but stratégique du processus. Concernant les rôles, seul le

responsable financier de l'ETS peut vérifier la solvabilité du client et peut par la suite juger s'il est possible ou pas d'accepter la commande client. Les intervenants métiers proposent de plus une règle métier qui traduit une politique interne de l'entreprise et qui consiste à effectuer certaines exceptions pour l'acceptation d'une commande client. Cette règle consiste à accepter une commande client dont le nombre d'heures de formation dépasse les 100 heures, même si le client s'est avéré « non solvable ».

Quant à la troisième sous catégorie du contexte fonctionnel, qui est le but stratégique, elle est déterminée à partir du modèle de motivations métier. Rappelons que chaque processus métier créateur de valeur ajoutée doit soutenir un but stratégique. D'après sa nature et sa finalité, le processus de prise de commande poursuit le but stratégique BS_1= « Être présent sur le réseau Internet ».

- Un contexte non fonctionnel : cette catégorie de contexte inclut les paramètres de qualité et de sécurité qui peuvent influencer le comportement d'un processus. Les intervenants métiers de l'ETS expriment des contraintes de sécurité en ce qui concerne certaines activités métier. Ces contraintes exigent qu'ils doivent prévoir un mécanisme de sécurisation d'accès à des données comme les données client, ou encore les données relatives aux formateurs (seuls les utilisateurs autorisés peuvent y accéder).

Ces différentes informations contextuelles sont prises en compte lors de la modélisation du processus. La Figure A.4 montre le processus de prise de commande client modélisé en utilisant le standard BPMN enrichi par un ensemble d'annotations contextuelles.

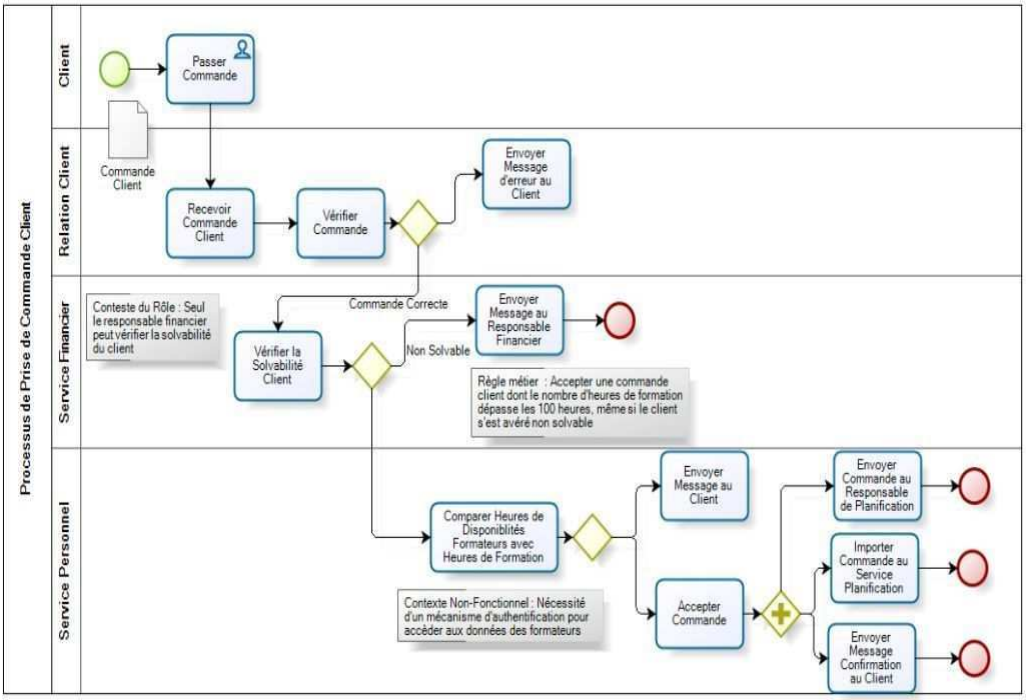


Figure A.4. Modélisation contextuelle du processus de prise de commande client

– Description du processus de planification des sessions de formation

Le processus de planification des sessions de formation permet d'affecter des formateurs à une ou plusieurs sessions de formation suite à l'acceptation d'une commande client. Il est

déclenché par la réception de la commande client, qui a été déjà validée par le service de relation client. Le processus débute par la détermination du nombre de sessions de formation indispensables pour satisfaire la commande client. Le nombre de sessions à prévoir sera calculé en considérant la date de début et la date de fin de la formation. Par la suite, le service de planification se charge de répartir les sessions de formation sur les semaines.
Étant donné que chaque session de formation donne naissance à un nouveau projet de formation, le service de gestion de projet de l'ETS crée un nouveau projet de formation qui inclut l'identité du client, la discipline de la formation, les heures de formations prévues, le lieu et les formateurs impliqués. De plus, il met à jour le profil des formateurs affectés en changeant leur statut de « disponible » à « affecté ». Parallèlement à ces activités, le service comptable calcule le coût total de la formation. Ce coût est variable en fonction de la discipline de la formation, des heures demandées et aussi des frais journaliers des intervenants. Des frais de déplacement sont aussi à prévoir, et ceci dans le cas où la formation ne se passerait pas dans les locaux de l'ETS. Ensuite, le service comptable crée un contrat de formation qui récapitule les informations concernant les sessions de formation et mentionne également l'engagement de l'ETS vis-à-vis du client. Une copie de ce contrat, accompagnée d'une facture détaillée et d'un calendrier de remboursement des frais de formation, sera envoyée au client. La facture et le calendrier de remboursement sont réalisés par le service de facturation qui les envois par la suite au système comptable de l'ETS.

– Modélisation contextuelle du processus de planification des sessions de formation

Les intervenants métiers de l'ETS proposent un ensemble d'informations contextuelles, jugées pertinentes, pour le processus de planification des sessions de formation à savoir :
- Un contexte environnemental : tout comme le processus de prise de commande, le processus de planification est déclenché suite à une acceptation d'une commande, et ceci pendant les horaires de travail de l'entreprise. Au-delà de ces horaires, le processus ne doit pas être déclenché. Certaines exceptions sont autorisées, notamment pendant les périodes où les demandes clients sont croissantes.
- Un contexte fonctionnel : les intervenants métiers renseignent deux sous catégories pour le contexte fonctionnel du processus de planification : le contexte du rôle et le but stratégique soutenu par le processus. Certaines activités métier sont sensibles au rôle. L'activité qui permet de déterminer les frais de déplacements des formateurs doit être réalisée exclusivement par le responsable comptable. Seul ce responsable possède la compétence nécessaire pour proposer les frais de déplacement en fonction de certaines considérations telles que le nombre de kilomètres à parcourir, le moyen de transport à prendre, *etc.* Concernant le but stratégique, le processus de planification poursuit le but stratégique BS_2 « Avoir de bonnes relations client » et ceci en gérant au mieux la répartition des sessions de formation et l'affectation du personnel compétent. De plus, le processus doit assurer l'affectation des formateurs et valider la commande client et ceci dans les meilleurs délais.
- Un contexte non fonctionnel : certaines activités métier telles que l'accès à des données du projet de formation, ou encore l'accès à des données des contrats de formations, nécessitent un mécanisme d'authentification. Ce mécanisme est assez important afin de s'assurer que seuls les utilisateurs autorisés peuvent avoir accès à ces données.

La Figure A.5 montre le processus de planification des sessions de formation modélisé en utilisant le standard BPMN enrichi par un ensemble d'annotations contextuelles.

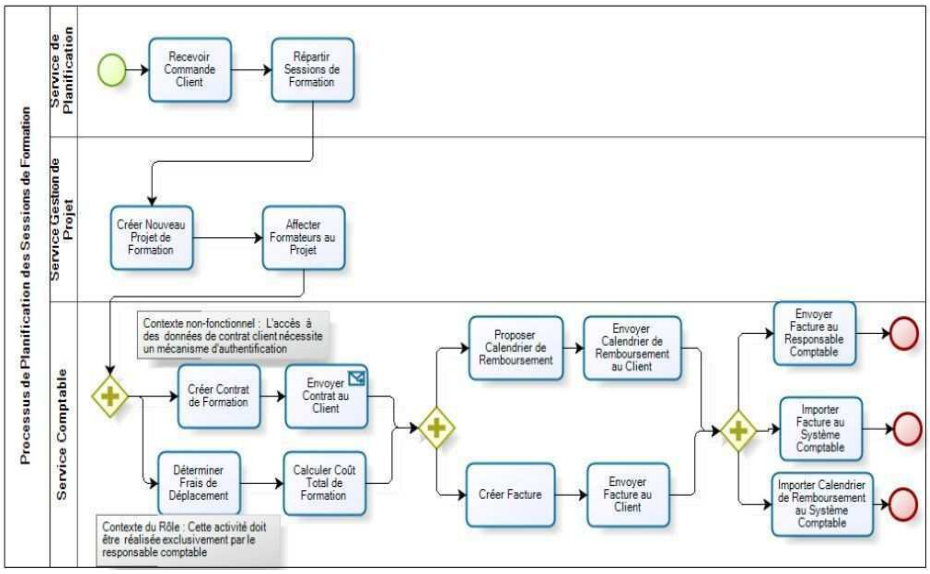


Figure A.5. Modélisation contextuelle du processus de planification des sessions de formation

- **Étape 2 : Identification des services métier**

Comme il a été déjà précisé dans le descriptif de la démarche CSOMA, l'identification des services métier débute par l'analyse des activités afin de les regrouper en un ensemble de services. Ce regroupement se base sur la notion de décomposition des buts. En effet, chaque processus métier comporte dans son contexte fonctionnel un but stratégique qu'il souhaite satisfaire. Un but stratégique peut être décomposé en un ou plusieurs buts tactiques. Les buts tactiques servent à identifier le ou les services domaines indispensables pour l'implémentation du processus métier en question. Par la suite, il s'agit de décomposer les buts tactiques afin de retrouver les buts opérationnels. Ces derniers devraient être satisfaits par les activités métier.
Dans la suite, nous allons appliquer les quatre sous étapes majeures de l'identification des services métier à savoir : la décomposition des buts, l'identification des services domaines et l'identification des services fonctionnels et la modélisation des services fonctionnels.

– Étape 2.1 : Décomposition des buts

o Décomposition des buts du processus de prise de commande client

Les intervenants métiers de l'ETS se chargent de décomposer les buts du processus conformément à l'arbre de décomposition (cf. chapitre 2, section 2.4.2, étape 2.1). Ils notent que le processus de prise de commande client doit satisfaire le but stratégique « Être présent sur le réseau Internet ». En effet, la prise de commande client à partir du site traduit une mission de l'entreprise qui consiste à conquérir de nouveaux marchés.

La décomposition du but stratégique du processus de prise de commande client est illustrée dans la Figure A.6.

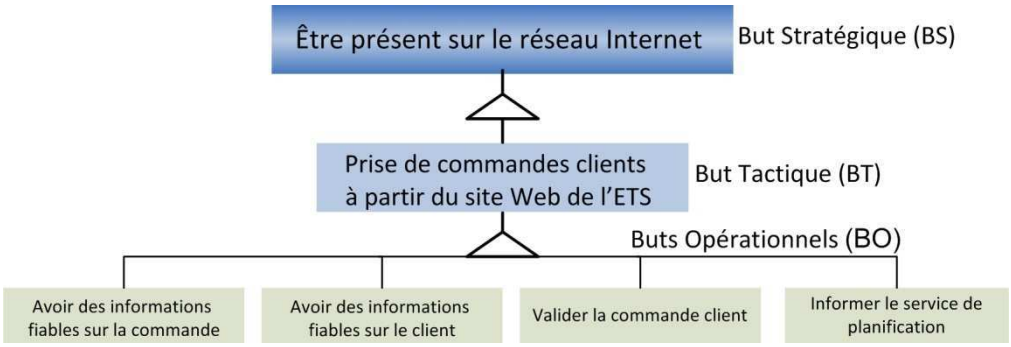


Figure A.6. Arbre de décomposition des buts du processus de prise de commande client

- Décomposition des buts du processus de planification des sessions de formation

Le but stratégique poursuivi par le processus de planification des sessions de formation est le BS_2= « Avoir de bonnes relations client ». En effet, le processus doit gérer au mieux la répartition des sessions de formation de manière à respecter les exigences du client. En outre, il doit proposer un calendrier de remboursement des frais de formation. Ainsi, le client peut payer et ceci sur plusieurs mensualités et sans frais supplémentaires. Ces différentes considérations visent à améliorer la relation avec le client. Ainsi, le but stratégique processus de planification se décompose en deux buts tactiques qui se décomposent à leur tour en des buts opérationnels (voir Figure A.7).

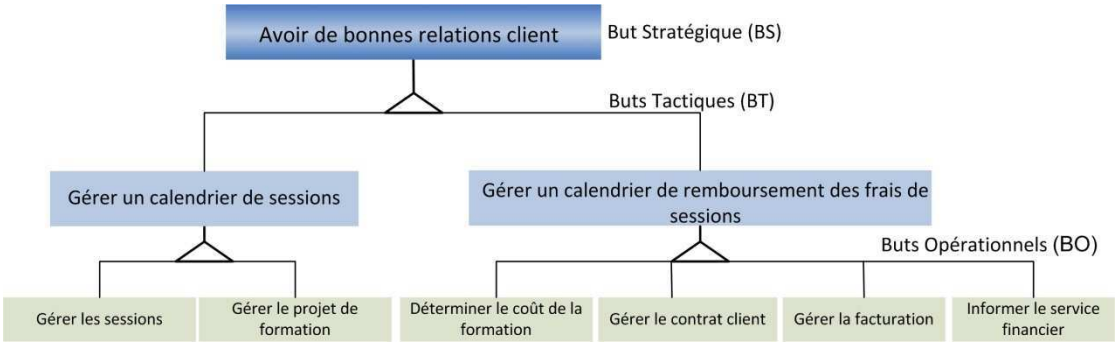


Figure A.7. Arbre de décomposition des buts du processus de planification des sessions de formation

- Étape 2.2 : Identification des services domaines
  - Identification du ou des services domaines à partir du processus de prise de commande client

Cette étape permet d'identifier le ou les services domaines en étudiant les buts tactiques qu'il (s) permet (tent) d'atteindre. Comme il a été déjà mentionné, les services domaines sont des services composites qui encapsulent des processus ou des sous-processus métier et constituent la brique de base pour la construction du processus coopératif.

Le but stratégique que le processus de prise de commande client cherche à atteindre est le BS_1 = « Être présent sur le réseau Internet » (voir Figure A.6). Ce but est décomposé en un but tactique BT_1 = « Prise de commandes clients à partir du site Web de l'ETS ». Ainsi, le service domaine responsable de l'implémentation du processus de prise de commande client doit satisfaire le but tactique BT_1. Ce service est nommé « Service de Prise de Commande en ligne ».

À ce stade de l'étude, on ne se préoccupe pas des traitements que le service domaine propose. En effet, d'après sa définition un service domaine assure la fonction d'orchestration d'un ensemble de services fonctionnels.

- Étape 2.3 : Identification des services fonctionnels
  - Identification des services fonctionnels à partir du processus de prise de commande client

L'identification des services fonctionnels se base sur l'étude du couplage entre les activités d'un processus, et ceci par rapport aux buts opérationnels que chacune d'entre elles poursuit. Tout d'abord, il est important de souligner que l'activité de passation de commande sur le site est une activité manuelle qui n'entre pas dans le cadre de l'identification des services fonctionnels.
La mise en correspondance entre les activités métier (AM) et les buts opérationnels (BO) du processus de prise de commande client est présentée dans le Tableau A.1.

BO / AM	Vérifier les informations de la commande	Vérifier les informations client	Valider la commande client	Informer le service de planification
Recevoir Commande Client	**X**			
Vérifier Commande	**X**			
Envoyer Message Erreur au Client	**X**			
Vérifier la Solvabilité Client		**X**		
Envoyer Message au Responsable Financier		**X**		
Comparer Heures Disponibilités Formateurs avec Heures de Formation			**X**	
Envoyer Message au Client			**X**	
Accepter Commande			**X**	
Envoyer Message Confirmation au Client			**X**	
Importer Commande au Service Planification				**X**
Envoyer Commande au Responsable de Planification				**X**

Tableau A.1. Mise en correspondance entre les activités métier (AM) et les buts opérationnels (BO) du processus de prise de commande client

Comme nous pouvons le remarquer, certaines activités métier satisfont les mêmes buts opérationnels. Ainsi, elles sont candidates pour appartenir au même service fonctionnel. En appliquant l'algorithme d'identification (cf. chapitre 2, section 2.4.2, étape 2.3), nous pouvons identifier quatre services fonctionnels (voir Figure A.8).

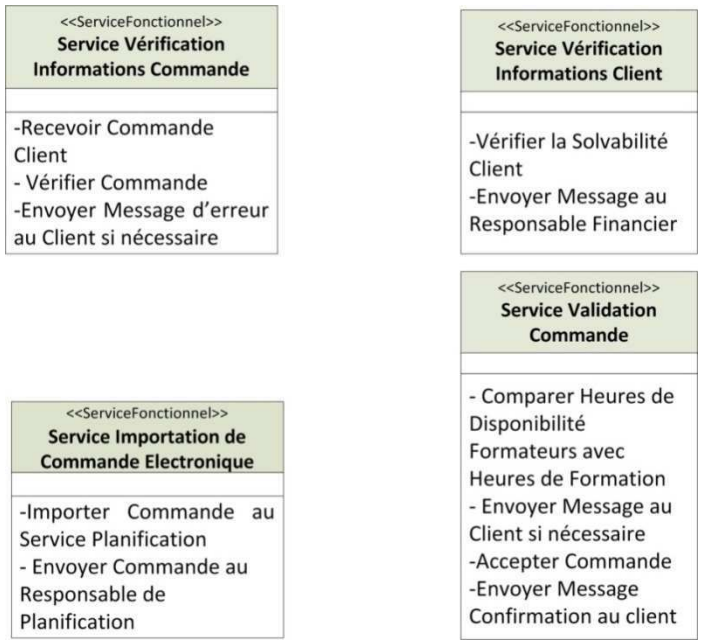


Figure A.8. Services fonctionnels identifiés à partir du processus de prise de commande client

- Identification du ou des services domaines à partir du processus de planification des sessions de formation

Le but stratégique que poursuit le processus de planification des sessions de formation est BS_2= « Avoir de bonnes relations client ». Ce but est décomposé en deux but tactique BT_2= « Gérer un calendrier de sessions » et BT_3= « Gérer un calendrier de remboursement des frais de session ». Par conséquent, on identifie deux services domaines responsables de l'implémentation du processus de planification et qui satisfont les deux buts tactiques BT_2 et BT_3. Ces deux services sont nommés respectivement « Service Planification » et « Service Remboursement Frais ». Le Tableau A.2 et Tableau A.3 étudient la mise en correspondance entre les activités métier du processus et les différents buts opérationnels issus des deux buts tactiques (BT_2 et BT_3).

BO / AM	Gérer les sessions	Gérer le projet de formation
Recevoir Commande Client	**X**	
Répartir Sessions de Formation	**X**	
Créer Nouveau Projet de Formation		**X**
Affecter Formateurs au Projet		**X**

Tableau A.2. Mise en correspondance entre les activités métier (AM) et les buts opérationnels (BO) issus du but tactique BT_2

BO / AM	Déterminer le coût de la formation	Gérer le contrat client	Gérer la facturation	Informer le service financier
Déterminer Frais de Déplacement	**X**			
Calculer Coût Total de Formation	**X**			
Créer Contrat de Formation		**X**		
Envoyer Contrat au Client		**X**		
Créer Facture			**X**	
Proposer Calendrier de Remboursement			**X**	
Envoyer Facture au Client			**X**	
Envoyer Calendrier de Remboursement au Client			**X**	
Importer Facture au Service Financier				**X**
Importer Calendrier de Remboursement au Service Financier				**X**
Envoyer Facture au Responsable Financier				**X**

Tableau A.3. Mise en correspondance entre les activités métier (AM) et les buts opérationnels (BO) issus du but tactique BT_3

En considérant les deux tableaux (Tableau A.2 et Tableau A.3), nous identifions les différents services fonctionnels orchestrés par les deux services domaines (voir Figure A.9).

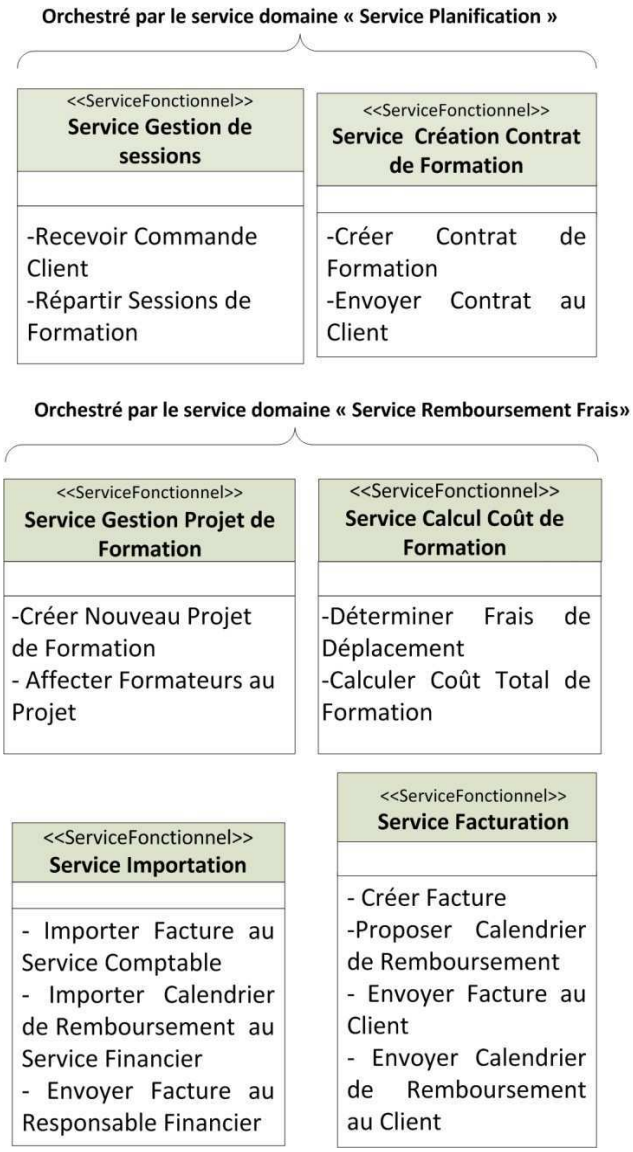


Figure A.9. Services fonctionnels identifiés à partir du processus de planification des sessions de formation

– Étape 2.4 : Modélisation des services fonctionnels

La modélisation des services fonctionnels au sein de CSOMA se situe au niveau PIM de la démarche MDA. Elle propose de mener un chantier de raffinement d'un ensemble de modèles pour obtenir à la fin des modèles de spécification des services fonctionnels. Étant suffisamment raffinés, ces modèles peuvent être projetés par la suite sur des plateformes d'exécution.

À titre d'exemple, nous allons nous concentrer sur la modélisation de deux services fonctionnels : le Service Vérification Informations Client (identifié à partir du processus de prise de commande client) et le Service Calcul Coût de Formation (identifié à partir du processus de planification des sessions de formation).

- Modélisation du Service Vérification Informations Client

Nous élaborons dans la suite l'ensemble des modèles décrivant le Service Vérification Informations Client.

Le Modèle d'Activité Métier : rappelons que le Modèle d'Activité Métier permet de décrire d'une manière microscopique la ou les activités métier encapsulée(s) par le service fonctionnel. Ce modèle est une extension du diagramme de cas d'utilisation UML et utilise les nouveaux stéréotypes déjà présentés (cf. chapitre 2, section 2.4.2, étape 2.4). Le Modèle d'Activité Métier relatif aux deux activités encapsulées par le Service Vérification Informations Client est présenté en Figure A.10. Ce modèle est réalisé par un analyste métier et ceci en se référant à la description du processus de prise de commande client et aux buts opérationnels satisfaits par les deux activités. En considérant ces informations, l'analyste décide de décomposer l'activité « Vérifier la Solvabilité Client » en une tâche appelée « Récupérer Informations Client ». Ainsi, la vérification de la solvabilité d'un client inclut la récupération des informations client.

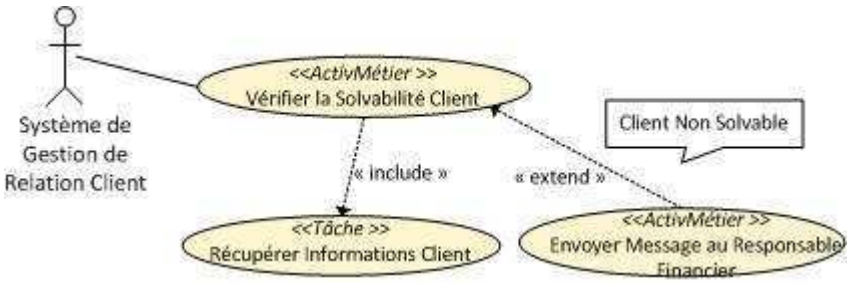


Figure A.10. Modèle d'Activité Métier relatif aux activités encapsulées par le « Service Vérification Informations Client »

Le Modèle de Service Fonctionnel : ce modèle se concentre sur la description du service fonctionnel et notamment sur les opérations qu'il propose. Il est réalisé par application des règles de transformation déjà présentées (cf. chapitre 2, section 2.4.2, étape 2.4). Cependant cette transformation n'est qu'une transformation semi-automatique et nécessite ainsi l'intervention de l'analyste métier. Ce dernier doit analyser la sémantique des opérations du service afin de distinguer les opérations de données (c'est-à-dire qui nécessitent l'accès à une base de données) des opérations de base. L'opération « Récupérer Informations Client » est une opération qui permet de récupérer les informations client à partir d'une base de données Client. Par conséquent, cette opération est étiquetée par le stéréotype <<OD>>. Tandis que l'opération « Envoyer Message au Responsable Financier » est une opération de base <<OB>>.

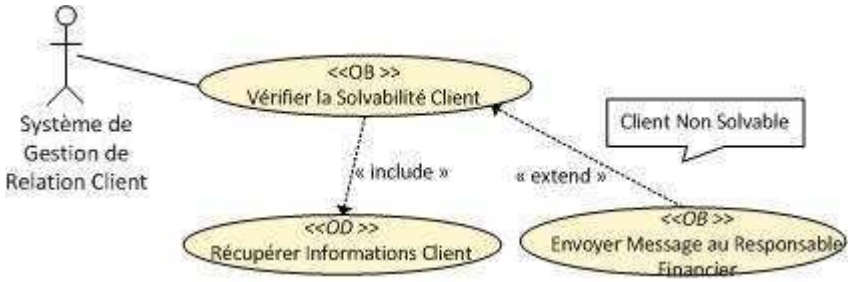


Figure A.11. Modèle de Service Fonctionnel intitulé « Service Vérification Informations Client »

Le Modèle de Spécification du Service Fonctionnel : ce modèle est une extension du diagramme d'activités UML. Il propose une description détaillée de l'enchaînement des différentes opérations du « Service Vérification Informations Client » et ceci conformément aux stéréotypes déjà présentés (cf. chapitre 2, section 2.4.2, étape 2.4).

Ce modèle est obtenu par application de l'ensemble des règles suivant (voir Figure A.12) :

- **R_2 :** les opérations du service fonctionnel se transforment en des activités que ce soit <<*SOAB*>> ou <<*SOAD*>> en fonction de leur types (*OB* ou *OD*). Cette règle de transformation est une règle automatique qui s'effectue sans l'intervention de l'analyste métier.

- **R_3 :** la relation d'utilisation <<*include*>> qui relie les deux cas d'utilisation « Vérifier Solvabilité Client » et « Récupérer Informations Client », dans le Modèle de Service Fonctionnel, se transforme en deux activités dont l'ordre de déclenchement est le suivant :

L'activité qui correspond au cas d'utilisation source (Vérifier Solvabilité Client) se réalise après le cas d'utilisation cible (Récupérer Informations Client).

- **R_4 :** la relation d'extension <<*extend*>> entre le cas d'utilisation « Vérifier la Solvabilité Client » et le cas d'utilisation « Envoyer Message au Responsable Financier » se transforme en flot de contrôle de type « *Fork* » dans le Modèle de Spécification du Service Fonctionnel. L'activité qui correspond au cas d'utilisation cible (Vérifier la Solvabilité Client) se déclenche avant l'activité qui correspond au cas d'utilisation source (Envoyer Message au Responsable Financier). Par la suite, le flot de contrôle de type « *Fork* » permet de présenter l'activité qui correspond au cas d'utilisation source (Envoyer Message au Responsable Financier) avec une alternative de type transition vide. Par la suite, le « *Fork* » est résolu par un « *Join* » afin de regrouper les deux transitions.

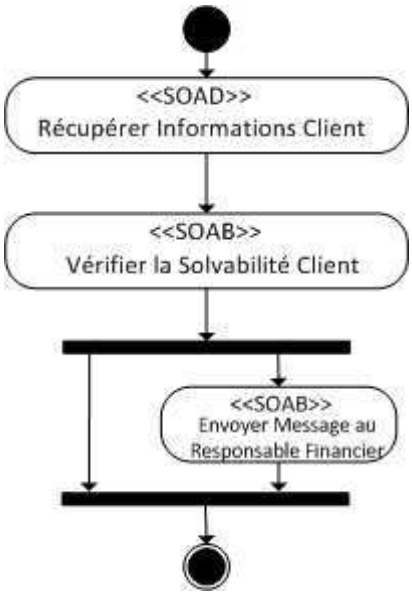


Figure A.12. Modèle de Spécification du « Service Vérification Informations Client »

- Modélisation du Service Calcul Coût de Formation

Tout comme la modélisation du Service Vérification Informations Client, la modélisation du Service Calcul Coût consiste à élaborer deux modèles intermédiaires afin d'obtenir le Modèle de Spécification du Service Fonctionnel.

Le Modèle d'Activité Métier : est présenté dans la Figure A.13. Il comporte les deux activités métier encapsulées par le service à savoir « Déterminer Frais de Déplacement » et « Identifier Coût Total de Formation ». Les deux activités sont décomposées en deux tâches intitulées « Identifier Frais Journalier des Formateurs » et « Identifier Coût de Discipline de Formation».

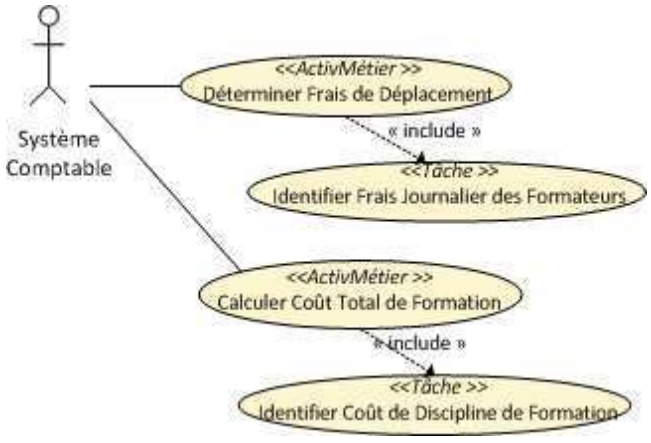


Figure A.13. Modèle d'Activité Métier relatif aux activités encapsulées par le « Service Calcul Coût de Formation »

Le Modèle de Service Fonctionnel : est présenté dans la Figure A.14. Il est obtenu par application de la règle de transformation $\mathbf{R_2}$ (cf. chapitre 2, section 2.4.2, étape 2.4). En effet, chaque activité ou tâche se transforme en une opération de service et c'est à la tâche de

l'analyste métier de décider de la sémantique des opérations (opération de base ou opération de données). Les deux activités « Déterminer Frais de Déplacement » et « Calculer Coût Total de Formation » se transforment en deux opérations de base. Alors que les deux tâches « Identifier Frais Journalier des Formateurs » et « Identifier Coût de Discipline de Formation » se transforment en deux opérations de données.

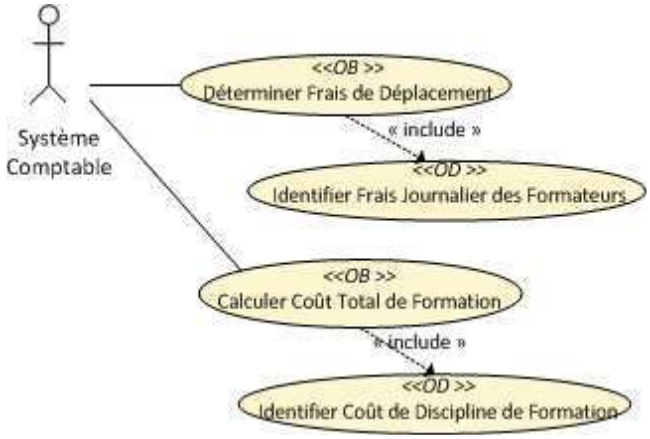


Figure A.14. Modèle de Service Fonctionnel intitulé « Service Calcul Coût de Formation »

Le Modèle de Spécification du Service Fonctionnel : est obtenu par application de la règle de transformation $\mathbf{R_3}$ (cf. chapitre 2, section 2.4.2, étape 2.4). En effet, les activités correspondant aux cas d'utilisations sources (Déterminer Frais de Déplacement et Calculer Coût Total de Formation) se réalisent après les cas d'utilisation cibles (Identifier Frais Journalier des Formateurs et Identifier Coût Discipline de Formation).

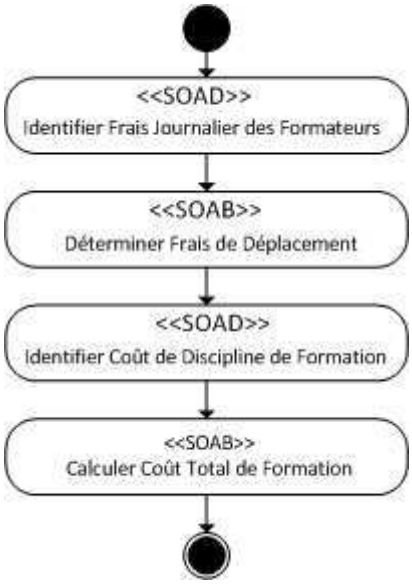


Figure A.15. Modèle de Spécification du « Service Calcul Coût de Formation »

- **Étape 3 : Raffinement des services fonctionnels**

Après avoir identifié et modélisé les services fonctionnels, il est intéressant de les raffiner afin de répondre au mieux aux propriétés de service. Les deux propriétés à vérifier sont : la propriété de réutilisation et la propriété d'autonomie.

En examinant les services fonctionnels du processus de prise de commande client, quelques axes de raffinements se présentent :

- Le « Service Vérification Informations Commande » contient une opération intitulée « Recevoir Commande Client » qui traduit la réception d'un message. Cependant, la réception d'un message fait partie de la logique des opérations d'un service et ne mérite pas d'être présenté comme une opération d'un service. Ainsi, cette opération doit être retirée de la liste des traitements proposés par le service.

- Le « Service Validation Commande » contient deux opérations intitulées « Comparer Heures Disponibilités Formateurs avec Heures de Formation » et « Envoyer message au client » qui sont fortement liées et qui méritent d'être encapsulées par un autre service fonctionnel. C'est la raison pour laquelle nous avons décidé de créer un nouveau service fonctionnel que nous avons appelé « Service Gestion de Formateurs ». De plus, le « Service Validation Commande » comporte deux autres opérations qui sont fortement liées l'une à l'autre. En effet lors de l'acceptation d'une commande client, un message d'acceptation doit être envoyé au client. Cependant, l'envoi du message doit être accompagné par l'invocation de l'opération d'acceptation. Ainsi, nous avons décidé de combiner ces deux opérations en une seule opération que nous appelons « Accepter Commande et envoyer message confirmation au client ».

- Le « Service Importation de Commande Electronique » est un service qui permet d'importer une commande dans le système de planification et de l'envoyer par la suite au responsable de planification. Afin de garantir la réutilisation du service, nous avons décidé de revoir son nom ainsi que les noms des opérations qu'il propose. Le nouveau nom du service est « Service Importation de Document Electronique ». Grâce à cette nouvelle appellation, le service peut participer à plusieurs scénarios d'usage (c'est-à-dire qu'il peut être utilisé pour importer d'autres documents que la commande).

En examinant les services fonctionnels du processus de planification des sessions de formation, quelques axes de raffinements se présentent :

- Le « Service Gestion de Sessions » comporte une opération de type réception de message intitulée « Recevoir Commande Client ». Comme nous l'avons déjà précisé, la réception d'un message fait partie de la logique des opérations d'un service et ne mérite pas d'être présentée comme une opération à part entière. Par conséquent, cette opération est retirée de la liste des traitements proposés par le service.

- Le « Service Création Contrat de Formation » contient deux opérations fortement liées. On ne peut pas invoquer l'opération d'envoi de contrat au client sans appeler l'opération « Créer Contrat de Formation ». Afin d'assurer le principe d'autonomie, ces deux opérations doivent être combinées en une seule opération. De plus, pour augmenter la réutilisation du service, nous avons décidé de réviser son nom pour devenir « Service Création de Contrat ». Ainsi, il

peut être utilisé pour créer d'autres types de contrat (tel que un contrat de livraison de matériels informatiques).

- Le « Service Importation » est remplacé par le « Service Importation de Documents Electroniques » car il permet d'assurer les mêmes fonctionnalités.

- **Étape 4 : Identification des Aspects Conceptuels**

La démarche CSOMA s'intéresse à l'adaptation des services dès la phase de modélisation. En effet, lors de la modélisation d'un service, il est déterminant de spécifier son « modèle d'adaptation », c'est-à-dire l'ensemble des paramètres et des préoccupations qui gouvernent son comportement. Il est aussi important d'étudier les actions d'adaptation qui doivent être mises en place pour gérer les changements de son contexte d'utilisation. Ces différentes considérations sont gérées par les Aspects Conceptuels. Un Aspect Conceptuel d'un service métier est identifié en analysant les paramètres contextuels du processus métier dont il est issu. Trois types d'Aspect Conceptuels sont identifiés à savoir : l'aspect non fonctionnel, l'aspect règle métier et l'aspect prise en compte du contexte.

Dans la suite, nous allons identifier les Aspects Conceptuels pour les services métier issus des deux processus de l'entreprise ETS.

En examinant le contexte fonctionnel du processus de prise de commande client, nous pouvons identifier un ensemble d'Aspects Conceptuels qui vont être attribués aux services concernés :

- Le contexte du rôle proposé dans le contexte fonctionnel du processus est mis en œuvre grâce à un aspect règle métier noté :

R_1= « seul le responsable financier de l'ETS peut vérifier la solvabilité du client ».

- La règle métier mentionnée dans le contexte fonctionnel du processus est mise en œuvre grâce à un aspect règle métier noté : R_2= « accepter de traiter la commande client dont le nombre d'heures de formation dépasse les 100 heures, même si le client s'est avéré non solvable ».

Ces deux aspects sont attribués au « Service Vérification Informations Client ».

- Les considérations de sécurité exprimées dans le contexte non fonctionnel du processus sont prises en compte par un aspect non fonctionnel noté A_1= « Autorisation Requise ». Cet aspect est attribué aux deux services : « Service Gestion des Formateurs » et « Service Vérification Informations Client ».

Concernant le processus de planification des sessions de formation, nous identifions les Aspects Conceptuels suivant :

- Le contexte du rôle énuméré dans le contexte fonctionnel du processus est pris en compte grâce à l'aspect règle métier noté R_3= « Seul le responsable comptable possède la compétence nécessaire pour proposer des frais de déplacement pour un client donné ». Cet aspect règle métier est attribué au « Service Calcul Coût de Formation ». De plus, les intervenants métiers souhaitent que le « Service Calcul Coût de Formation » soit un service sensible au contexte dynamique du client. En effet, si la commande a été validée, et si le client décide de changer

le nombre de personnes à former, le « Service Calcul Coût de Formation » doit ajouter des frais supplémentaires lors du calcul du coût de la formation. Ses frais constituent les frais d'études et de mise à jour du projet de formation. La sensibilité au contexte du client est prise en compte grâce à l'aspect de prise en compte du contexte noté C_1= « Si le client change le nombre de personnes à former, alors attribuer des frais supplémentaires lors du calcul du coût de la formation ». En outre, les intervenants métiers de l'ETS, précisent qu'ils souhaitent que le « Service Gestion de Projet de Formation » soit un service sensible au contexte de son utilisation. En effet, ils suggèrent qu'il change de comportement en fonction des interactions passées avec le client. Pour les nouveaux clients, le service doit affecter les formateurs les plus expérimentés dans la discipline de formation qu'ils ont sélectionnée. Cette action dépendante du contexte est mise en œuvre grâce à l'aspect noté C_2= « Si le client est nouveau alors affecter des formateurs expérimentés ».

- Les considérations de sécurité, exprimées dans le contexte non fonctionnel du processus, sont prises en compte par un aspect non fonctionnel noté A_2= « Authentification Requise ». Cet aspect est attribué au « Service Création de Contrat ».

- **Étape 5 : Modélisation des Aspects Conceptuels**

L'objectif de cette étape est de proposer une modélisation détaillée des Aspects Conceptuels déjà identifiés. Les modèles proposés étendent les diagrammes UML en utilisant de nouveaux stéréotypes. À titre d'exemple, nous allons présenter la modélisation uniquement de deux Aspects Conceptuels :

- Le « Service Vérification Informations Client » possède l'Aspect Conceptuel de type règle métier.

- Le « Service Calcul Coût de Formation » possède l'Aspect Conceptuel de type prise en compte du contexte.

  - Modélisation de l'Aspect Conceptuel relatif au « Service Vérification Informations Client »
    - Le Modèle d'Aspect Conceptuel de type règle métier relatif au « Service Vérification Informations Client »

Rappelons que ce modèle permet de décrire en détail un Aspect Conceptuel. Il présente une extension du diagramme de cas d'utilisation UML et comporte deux nouveaux stéréotypes appelés *AspectAdvice et AdviceAction*. L'*AspectAdvice* représente une fonctionnalité proposée par un Aspect Conceptuel. Chaque *AspectAdvice* peut être décomposé en un ensemble d'actions élémentaires. Ces actions sont représentées par les *AdviceAction*. Le Modèle d'Aspect Conceptuel de type règle métier est représenté sur la Figure A.16.

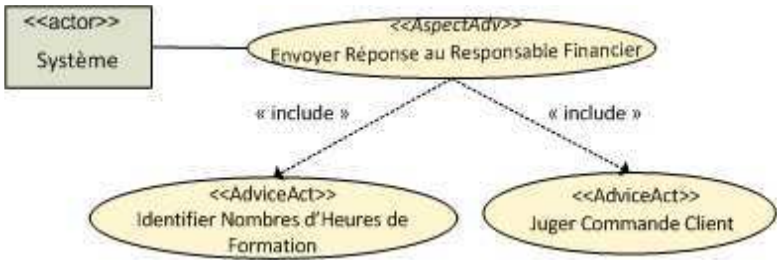


Figure A.16. Modèle d'Aspect Conceptuel de type règle métier relatif au « Service Vérification Informations Client »

- Le Modèle de Spécification d'Aspect Conceptuel de type règle métier relatif au « Service Vérification Informations Client »

Ce modèle complète la description de l'Aspect Conceptuel, dans le sens où il ajoute le flux d'exécution des différentes actions appartenant à l'aspect. La Figure A.17 présente le Modèle de Spécification d'Aspect Conceptuel de type règle métier relatif au « Service Vérification Informations Client ». Ce modèle est obtenu par application des règles de transformation $\mathbf{R_1}$ et $\mathbf{R_{2.1}}$ (cf. chapitre 2, section 2.4.2, étape 5). La première règle ($\mathbf{R_1}$) est une règle de transformation automatique alors que la seconde est semi-automatique et nécessite l'intervention de l'analyste métier. Ce dernier se charge de déterminer l'ordre de réalisation des deux cas d'utilisation cibles de la relation d'inclusion (Identifier Nombres d'Heures de Formation et Juger Commande Client).

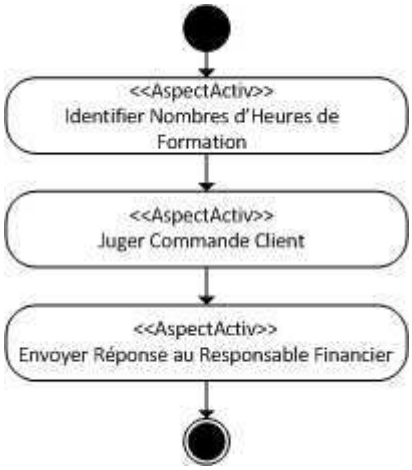


Figure A.17. Modèle de Spécification d'Aspect Conceptuel de type règle métier relatif au « Service Vérification Informations Client »

– Modélisation de l'Aspect Conceptuel relatif au « Service Calcul Coût de Formation »

- Le Modèle d'Aspect Conceptuel de type prise en compte du contexte relatif au « Service Calcul Coût de Formation »

Ce modèle présente l'Aspect Conceptuel qui permet de prendre en considération la sensibilité du « Service Calcul Coût » au contexte de son utilisation. En effet, si le client décide de changer le nombre de personnes à former, le « Service Calcul Coût de Formation » doit ajouter des frais supplémentaires lors du calcul du coût de la formation. Pour ce faire, cet

Aspect Conceptuel propose deux actions appelées « Identifier Nouveau Nombre de Personnes » et « Identifier les Frais par personne » (voir Figure A.18). Ces deux actions constituent la base pour le calcul des frais supplémentaires, présentées à l'aide des deux cas d'utilisation stéréotypés <<*AdviceAct*>>.

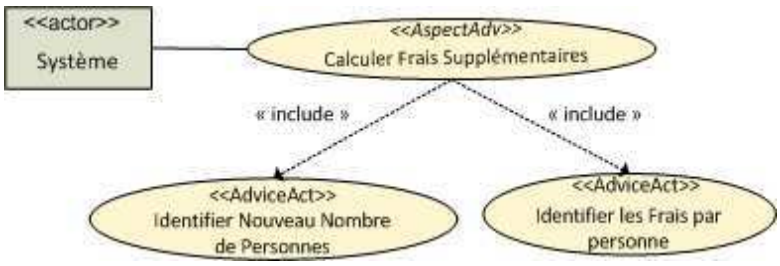


Figure A.18. Modèle d'Aspect Conceptuel de type prise en compte du contexte relatif au « Service Calcul Coût de Formation »

- Le Modèle de Spécification d'Aspect Conceptuel de type prise en compte du contexte relatif au « Service Calcul Coût de Formation »

Ce modèle montre le flux d'exécution des différentes actions appartenant à l'aspect (voir Figure A.19). Il est obtenu par application des règles de transformation $\mathbf{R_1}$ et $\mathbf{R_{2.1}}$ (cf. chapitre 2, section 2.4.2, étape 5). La règle $\mathbf{R_1}$ est une règle automatique qui permet de transformer le Modèle d'Aspect Conceptuel en un Modèle de Spécification d'Aspect Conceptuel. Alors que la règle $\mathbf{R_{2.1}}$ nécessite l'intervention de l'analyste métier afin de déterminer l'ordre de réalisation des deux cas d'utilisation cibles de la relation d'inclusion (Identifier Nouveau Nombre de Personnes et Identifier les Frais par personne).

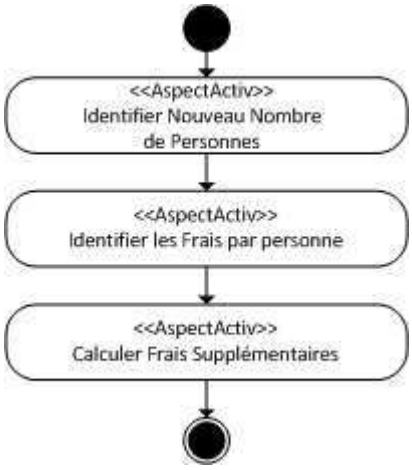


Figure A.19. Modèle de Spécification d'Aspect Conceptuel de type prise en compte du contexte relatif au « Service Calcul Coût de Formation »

- **Étape 6 : Création du ou des Modèles de Schéma d'Orchestration des Services Domaines**

La dernière étape de la phase de construction de la SOA métier est l'étape de création du ou des modèles de schéma d'orchestration des services. Son objectif est d'identifier les différents

schémas d'orchestration qui servent à implémenter le processus métier en question. Le ou les modèle (s) obtenu (s) doi(ven)t être conformes au méta-modèle déjà décrit (cf. chapitre 2, section 2.3.1.3).

Le Modèle de Schéma d'Orchestration du service domaine « Service de Prise de Commande en ligne » est présenté en Figure A.20. Il montre les cinq services fonctionnels qu'il orchestre ainsi que leur Aspects Conceptuels. Pour des raisons de lisibilité, les traitements offerts par les Aspects Conceptuels ne sont pas présentés dans la figure.

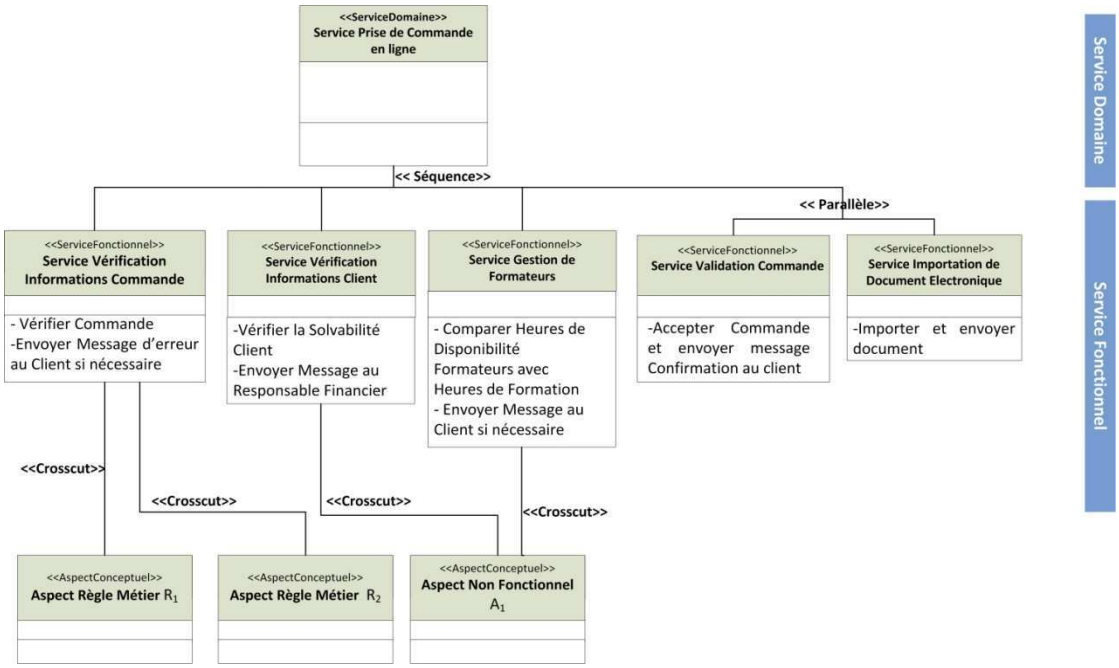
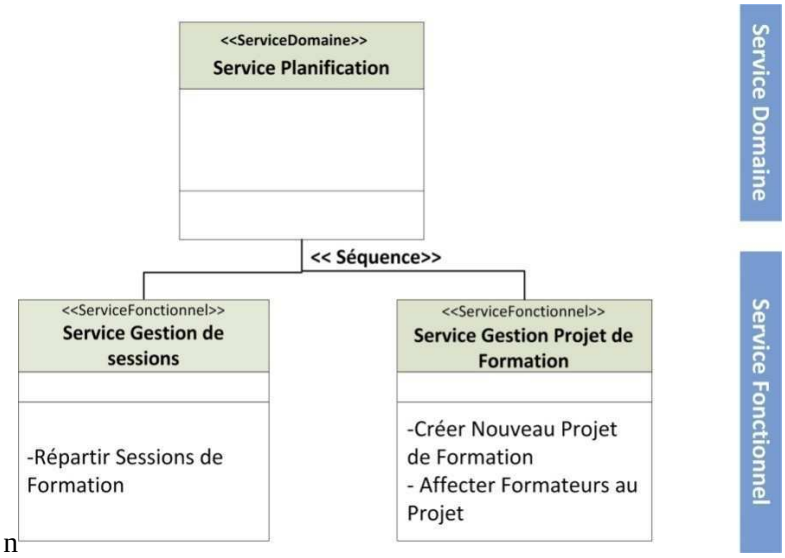


Figure A.20. Modèle de Schéma d'Orchestration du « Service de Prise de Commande en ligne » avec l'ensemble des Aspects Conceptuels

Quant au service domaine nommé « Service Planification », il orchestre deux services fonctionnels. Son Modèle de Schéma d'Orchestration est montré dans la Figure A.21. Le service domaine appelé « Service Remboursement Frais de Sessions » orchestre quatre services fonctionnels (voir Figure A.22).



n

Figure A.21. Modèle de Schéma d'Orchestration du « Service Planification »

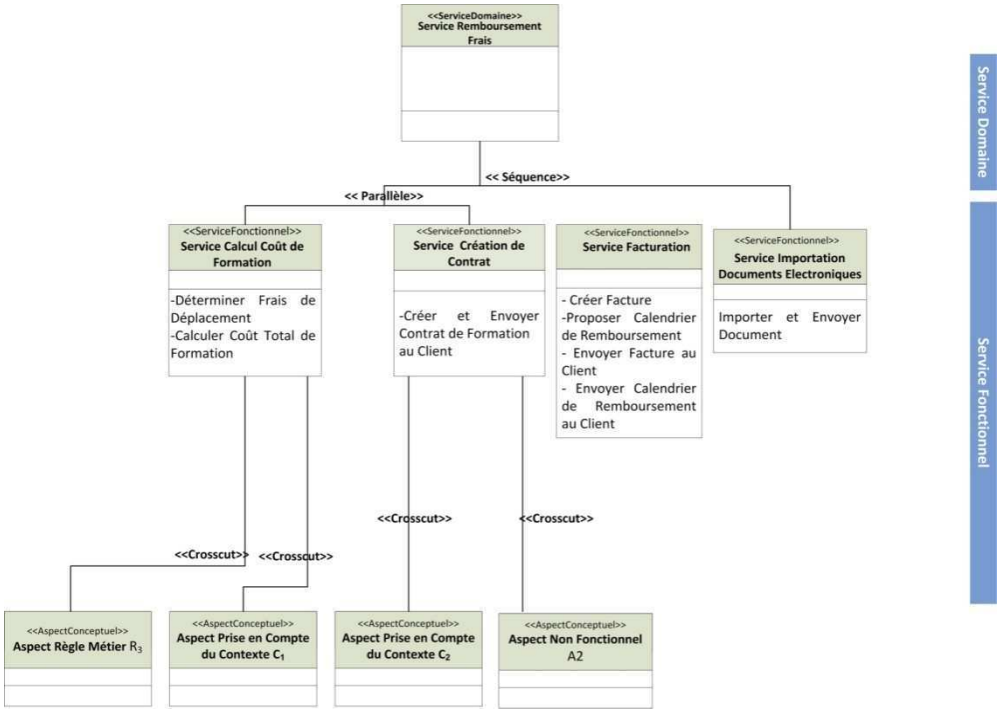


Figure A.22. Modèle de Schéma d'Orchestration du « Service Remboursement Frais de Sessions »

3.3 Phase 3 : Construction de la SOA IT

Les intervenants IT de l'ETS se chargent dans cette phase d'analyser le portefeuille applicatif de l'entreprise. Dans le cadre de notre étude, seule l'application de formation de personnel de l'ETS et les référentiels de données sont à examiner. L'objectif majeur étant d'analyser les fonctionnalités mises en place pour identifier celles qui sont éligibles au rang de services informatiques. Trois étapes fondamentales sont à considérer :

- **Étape 1 : Analyse du portefeuille applicatif de l'entreprise**

L'analyse du portefeuille applicatif de l'ETS se focalise essentiellement sur l'application de gestion des formations continues du personnel de l'ETS et sur l'étude des référentiels de données. L'application de gestion des formations est une application récente (développée en 2006), dont l'objectif est de proposer des sessions de formation interne pour le personnel de l'ETS. Les sessions proposées permettent aux personnels de l'ETS de suivre les évolutions technologiques dans le monde informatique, souvent considéré comme un monde en perpétuelle évolution. Cette application gère trois modules de base à savoir :

- Module d'identification des disciplines de formation à envisager : ce module est utilisé par les responsables de formation qui se préoccupent de l'étude des évolutions technologiques telles que les évolutions d'outils, de démarches et des technologies informatiques. Par la suite, ils déterminent les disciplines de formation qu'il faut envisager pour satisfaire les demandes client.

- Module d'affectation du personnel à des projets de formation : ce module permet d'identifier le personnel à former et de l'affecter par la suite à un ou plusieurs projets de formation.

- Module de répartition des sessions de formation en fonction des disponibilités du personnel : ce module se charge d'identifier le nombre de sessions à effectuer et de les répartir sur des semaines en fonction de la disponibilité du personnel.

Quant aux référentiels de données, ils regroupent les bases de données contenant les « références » du système d'information. Ces référentiels sont les données dont les applications ont besoin pour fonctionner. Des exemples de référentiels incluent les bases de données client, matériels informatiques, fournisseurs, *etc.*

- **Étape 2 : Identification des services d'accès au système legacy**

L'étape précédente a permis d'analyser l'application de gestion des formations de personnel et les référentiels de données que l'entreprise souhaite conserver. Dans la présente étape, nous nous focalisons sur l'identification des services d'accès au système *legacy*. Les trois modules implémentés par l'application seront encapsulés par trois services d'accès au système *legacy* appelés :

- Service Identification Discipline de Formation
- Service d'Affectation du Personnel
- Service Répartition des Sessions

Pour des raisons de simplicité, l'encapsulation des trois modules de l'application en des services et la construction des façades homogènes au-dessus de ces modules ne sont pas prise en compte dans le cadre de notre étude.

- **Étape 3 : Identification des services entités**

Cette étape s'intéresse à l'identification des services entités à partir du modèle d'objets métier de l'entreprise. Ce modèle est constitué d'un ou de plusieurs diagrammes de type diagramme de classes UML montrant ces objets métiers et leurs relations. L'entreprise ETS possède un diagramme de classe présenté en Figure A.23.

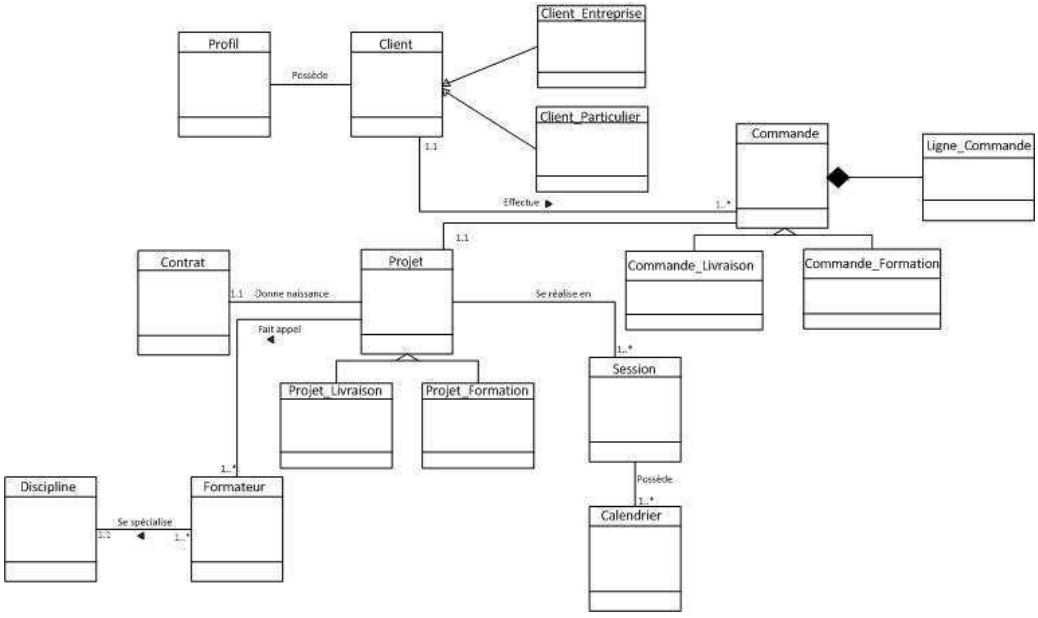


Figure A.23. Modèle des objets métier de l'ETS

Par la suite, il s'agit de regrouper les classes en des services entités et ceci en obéissant à l'ensemble des caractéristiques et des règles de regroupement déjà présentées en détail (cf. chapitre 2, section 2.4.3, étape 3). L'application des règles de regroupement donne naissance à un ensemble de services entités. Chaque service encapsule une ou plusieurs classes. La Figure A.24. illustre les services entités identifiés.

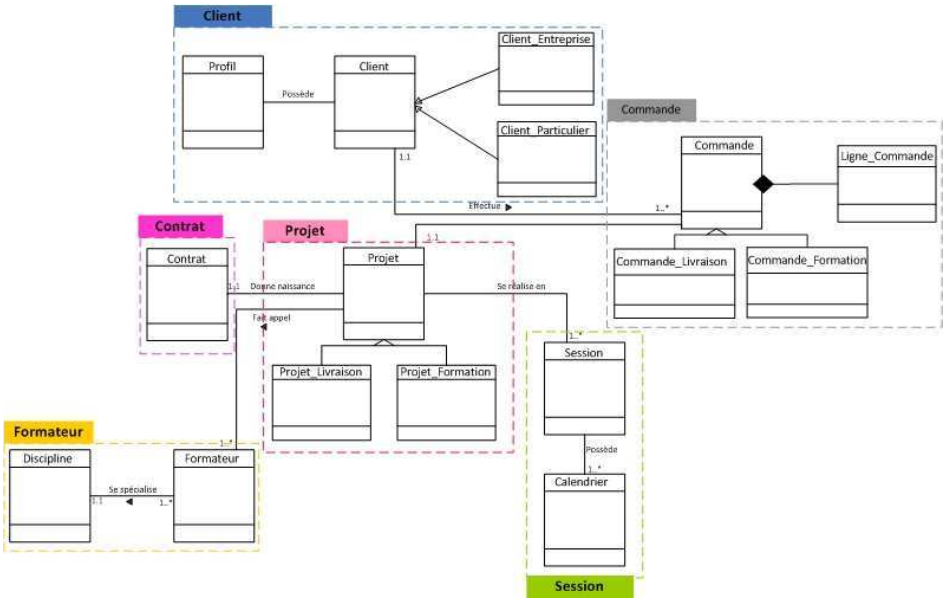


Figure A.24. Services entités identifiés

Il est à souligner que l'identification des services entités est en partie empirique. Par exemple, le regroupement des concepts de discipline et de formateur pourrait être discuté.

3.4 Phase 4 : Accostage entre la SOA métier et la SOA IT

Cette phase est souvent la plus difficile à réaliser car elle est au centre d'intérêts, souvent divergents, des différents responsables applicatifs et métiers de l'entreprise. Rappelons que cette phase permet de confronter les résultats des deux phases (phase métier et phase informatique) afin de valider les services qui ont été déjà identifiés. De plus, elle permet de retrouver les services informatiques à supprimer du système informatique de l'entreprise et/ou les services informatiques qu'il faut développer pour répondre à de nouveaux besoins métier.
Pour simplifier les choses, nous allons nous concentrer sur les deux étapes 1 et 3 respectivement : la consolidation de services et la consolidation des schémas d'orchestration.

- **Étape 1 : Consolidation des services**

Rappelons que la consolidation de services permet de s'assurer que les services métier qui ont été identifiés dans la SOA métier peuvent être implémentés à l'aide des services informatiques retrouvés dans la SOA IT.
En confrontant les services fonctionnels identifiés à partir du processus de prise de commande client et les services informatiques retrouvés en appliquant la SOA IT, plusieurs constatations peuvent être notées :
- Le « Service Vérification Informations Commande » et le « Service Validation Commande » doivent être développés car il n'existe aucun service IT permettant de vérifier et de valider la commande client,

- Le « Service Vérification Informations Client » doit utiliser le service entité appelé « Service Client » qui permettra de récupérer les informations relatives à un client,

- Le « Service Gestion de Formateurs » doit utiliser le service entité appelé « Service Formateur » qui permettra de récupérer la liste des formateurs disponibles.

De la même manière en étudiant les services fonctionnels identifiés à partir du processus de planification des sessions de formation et les différents services informatiques nous pouvons noter que :

- Le « Service Gestion de Sessions » peut être implémenté par le service d'accès au système *legacy* appelé « Service Répartition des Sessions »,

Le « Service Calcul de coût de Formation » doit utiliser le service entité appelé « Service Formateur ». Ce dernier englobe les opérations d'identification du coût de la discipline de formation, l'identification des frais de formateur, et de leur frais de déplacement,

- Le « Service Facturation » doit être développé car il n'existe aucun service IT qui permet de gérer la facturation. Le même constat est valable pour le « Service Création de Contrat » et le « Service Importation de Document Electronique »,

- Le « Service Gestion Projet de Formation » doit utiliser deux services entités qui sont : le « Service Projet » et le « Service Formateur ». Le Service « Projet » permettra de créer un nouveau projet de formation alors que le « Service Formateur » permettra d'identifier les formateurs impliqués et de mettre à jour leurs profils.

- **Étape 3 : Consolidation du ou des Modèles de Schéma d'Orchestration des Services Domaines**

Ce type de consolidation consiste à revoir le ou les Modèles de Schéma d'Orchestration en considérant la consolidation des services. Le service domaine responsable de l'implémentation du processus de prise de commande client orchestre cinq services fonctionnels. Deux parmi eux utilisent à leur tour deux services entités. La Figure A.25 montre le nouveau Modèle de Schéma d'Orchestration du service domaine « Service de Prise de Commande en ligne ». De même, le service domaine « Service Planification » se charge d'orchestrer les deux services fonctionnels dont deux d'entre eux utilisent, à leur tour, des services entités (voir Figure A.26). Quant au nouveau Modèle de Schéma d'Orchestration du service domaine « Service Remboursement Frais de Sessions », il est montré dans la Figure A.27.

Pour des raisons de lisibilité des deux figures, nous n'allons pas montrer les Aspects Conceptuels relatifs aux différents services fonctionnels.

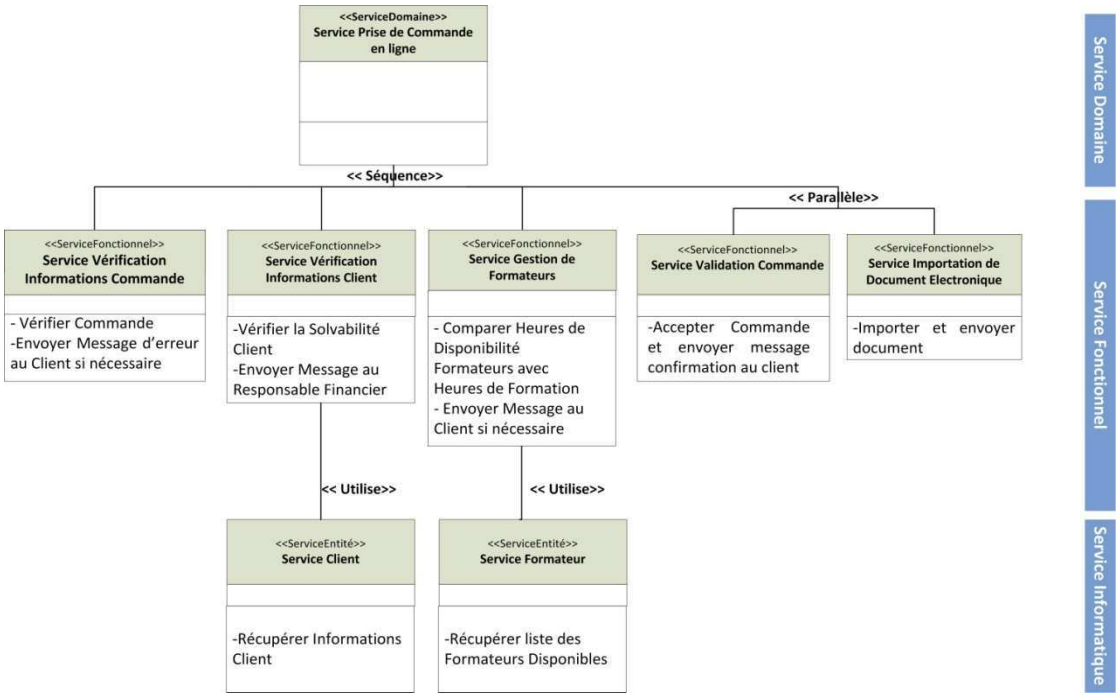


Figure A.25. Modèle de Schéma d'Orchestration du « Service de Prise de Commande en ligne » après l'étape de consolidation

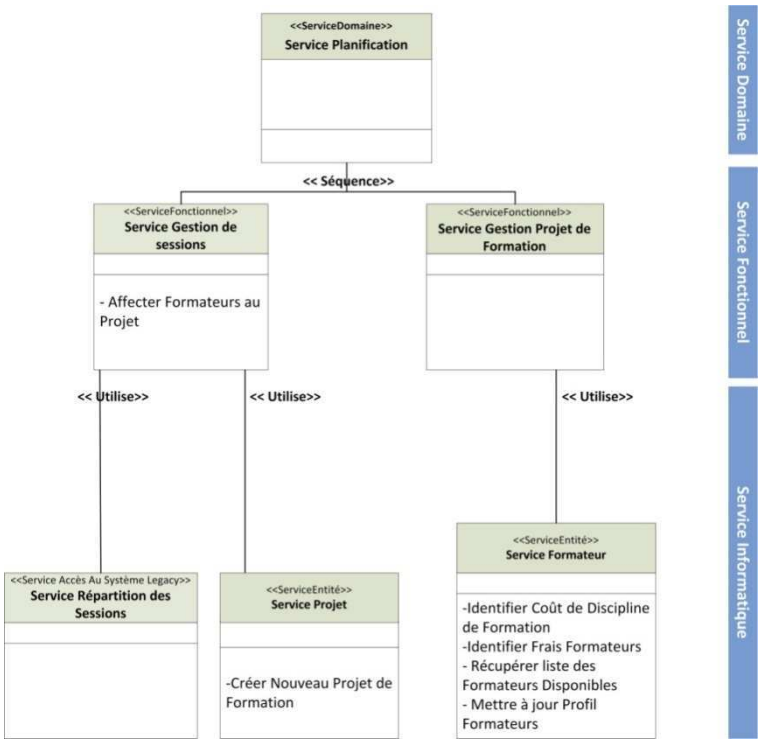


Figure A.26. Modèle de Schéma d'Orchestration du « Service Planification » après l'étape de consolidation.

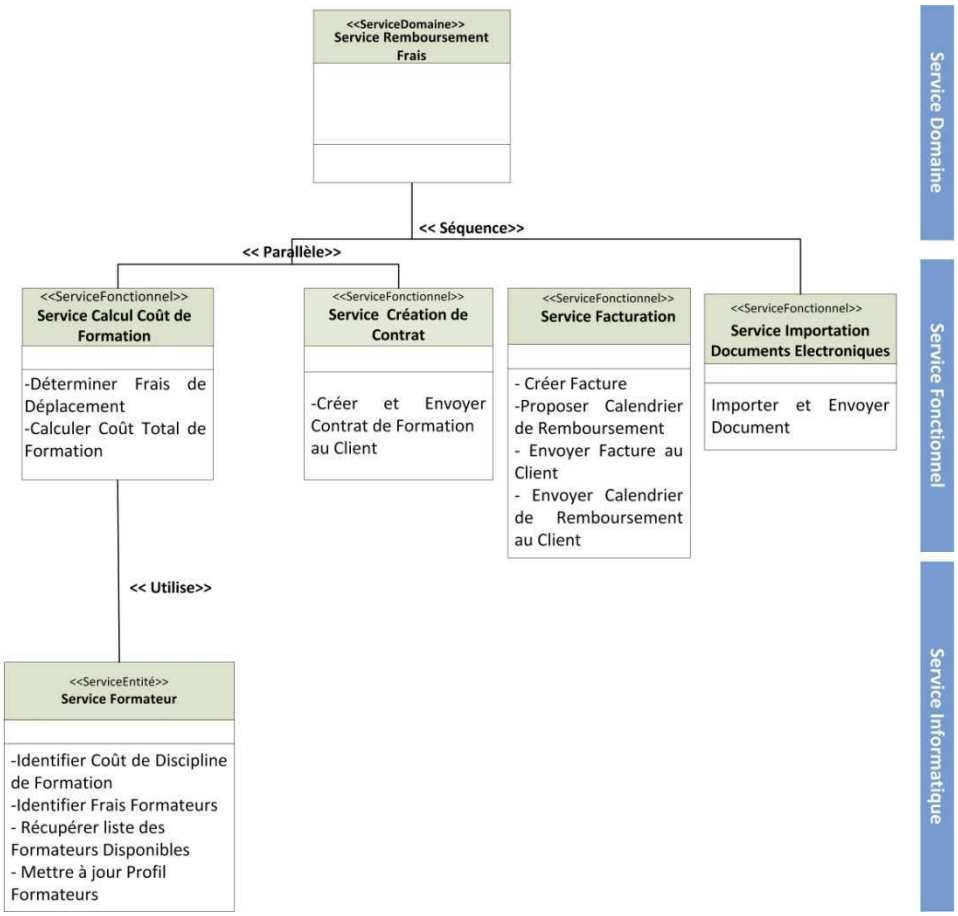


Figure A.27. Modèle de Schéma d'Orchestration du « Service Remboursement Frais de Sessions » après l'étape de consolidation

4. Prototype

Dans cette section, nous présentons quelques détails de réalisation de l'atelier de génie logiciel support à CSOMA. Tout d'abord, nous commençons par détailler l'environnement logiciel choisi. Par la suite, nous exposons quelques interfaces homme machine réalisées. Nous clôturons cette section par une campagne de test et expérimentation du prototype.

4.1 Environnement logiciel

L'atelier de génie logiciel support à CSOMA se base sur un ensemble d'outils et de langages :

- **Eclipse IDE** est un environnement de développement libre permettant potentiellement de créer des projets de développement mettant en œuvre n'importe quel langage de programmation (C++, PHP...). Eclipse IDE est principalement écrit en Java.
- **JAVA** est un langage de programmation orienté objet, développé par Sun Microsystems. Il permet de créer des logiciels compatibles avec de nombreux systèmes d'exploitation (Windows, Linux, Macintosh, Solaris). Java permet également de développer des programmes pour téléphones portables et assistants

personnels. Ce langage peut être utilisé sur Internet pour des applications intégrées à une page Web (applet) ou encore comme langage serveur (jsp).

- **SWING** est une bibliothèque graphique pour le langage de programmation Java, faisant partie du package *Java Foundation Classes* (JFC), inclus dans J2SE. Swing est considérée comme l'une des principales évolutions apportées par Java 2 par rapport aux versions antérieures. Swing permet de créer des interfaces graphiques identiques quel que soit le système d'exploitation sous-jacent.

4.2 Présentation de l'atelier de CSOMA

Dans cette section, nous allons présenter le principe de fonctionnement de notre atelier génie logiciel à travers un enchaînement représentatif de quelques écrans.
Le lancement de l'atelier développé génère la création d'un nouveau projet. Par la suite, l'analyste métier peut choisir la phase et l'étape qu'il désire effectuer. Une fenêtre, contenant une liste des phases de l'approche CSOMA, la liste des étapes de chaque phase, ainsi qu'un aperçu du résultat du choix, sont proposés à l'analyste métier (voir Figure A.28).

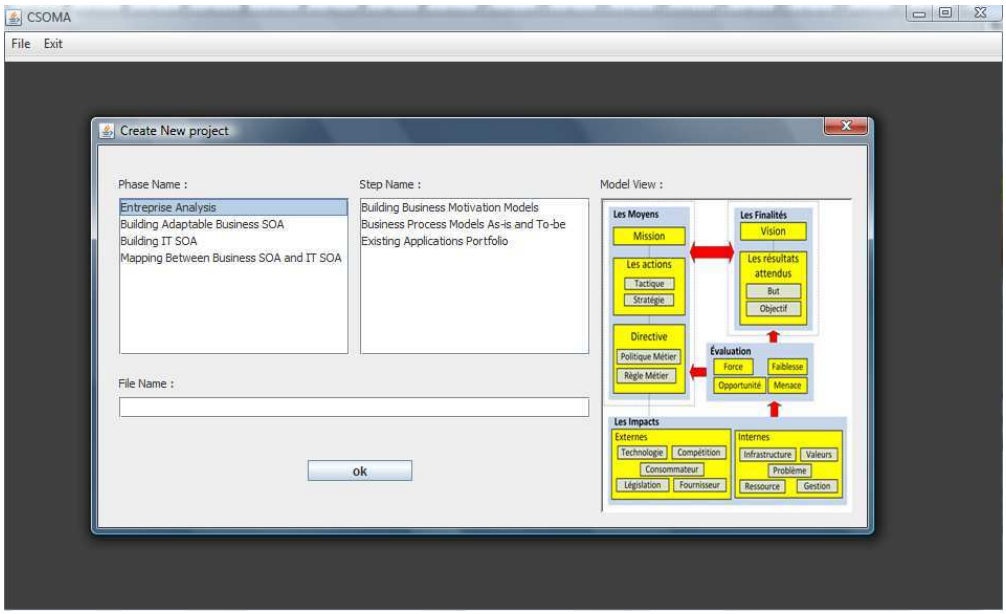


Figure A.28. Ecran principal de l'atelier support à CSOMA

L'analyste métier peut par exemple démarrer la phase de construction de la SOA métier et modélise ainsi un processus métier grâce à l'interface proposée par l'atelier développé. Cette interface utilise le standard BPMN (*Business Process Modeling Notation*) avec les annotations contextuelles (voir figure suivante).

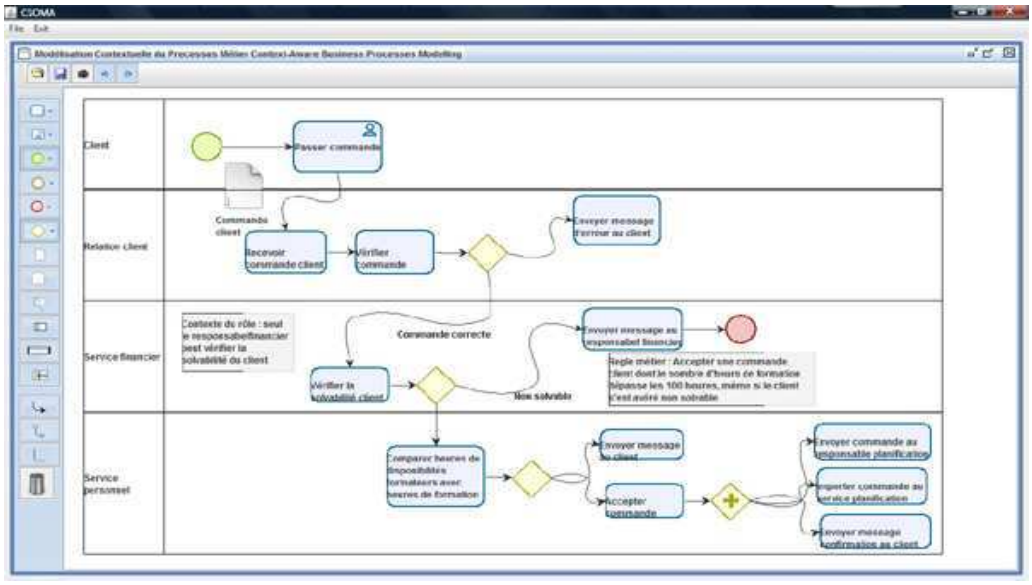


Figure A.29. Élaboration du modèle contextuel des processus métiers

Par exemple, l'analyste métier peut procéder à la décomposition des buts métier formant ainsi l'arbre de décomposition (voir Figure A.30). Cet arbre contient le but stratégique d'un processus, les buts tactiques et les buts opérationnels.

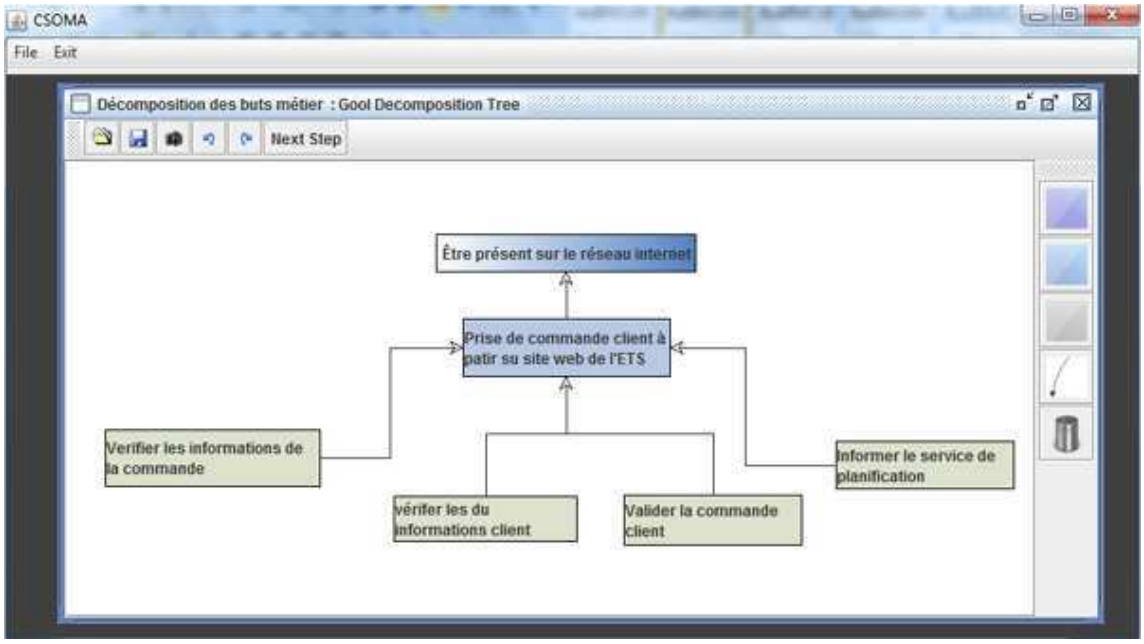


Figure A.30. Élaboration de l'arbre de décomposition des buts

Une fois que les processus métiers et les buts opérationnels sont identifiés, l'atelier crée le tableau de correspondance entre les buts opérationnels et les activités métier (voir Figure A.31). Les lignes du tableau, illustrent les activités métier alors que les colonnes présentent les buts.

Par la suite le système applique l'algorithme de regroupement des activités métier en des services fonctionnels. Il offre également la possibilité à l'analyste métier d'intervenir pour corriger certains regroupements d'activités qui ne lui conviennent pas.

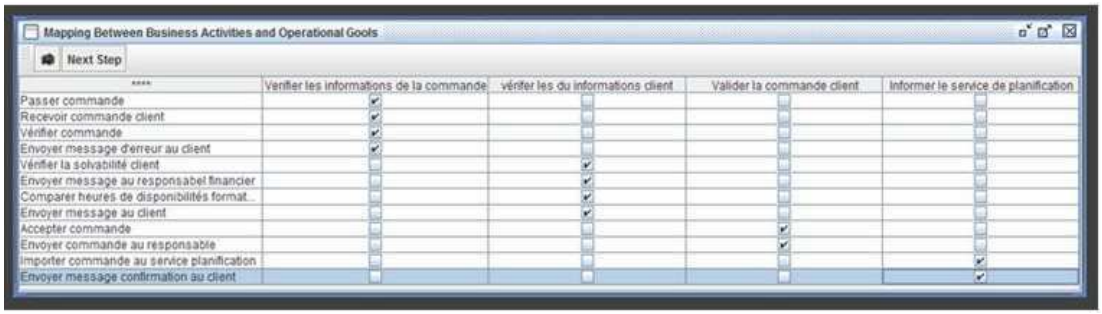


Figure A.31. Élaboration du tableau de correspondance entre les activités métier et les buts opérationnels

Finalement, en appuyant sur le bouton « *next step* » (voir Figure A.31), le système élabore les services fonctionnels (voir Figure A.32) résultant de la correspondance pour permettre à l'analyste métier de les nommer et de les enregistrer afin de les utiliser dans l'étape suivante.

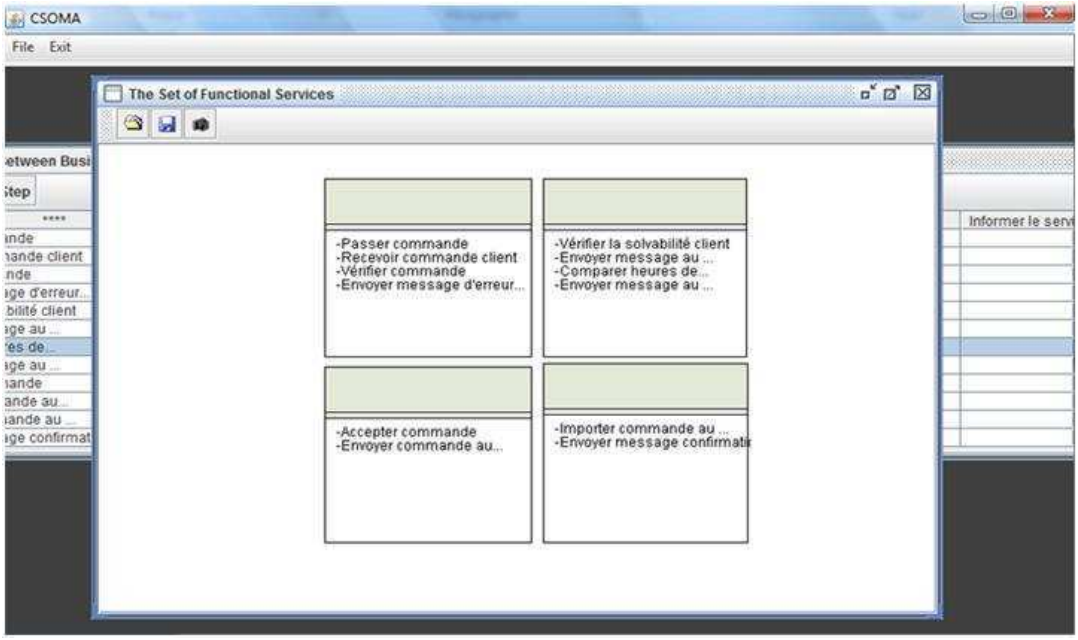


Figure A.32. Élaboration de services fonctionnels

L'analyste métier peut procéder par la suite à une modélisation détaillée de chaque service fonctionnel (voir Figure A.33) ou encore à un raffinement d'un ou de plusieurs services (voir Figure A.34, Figure A.35).

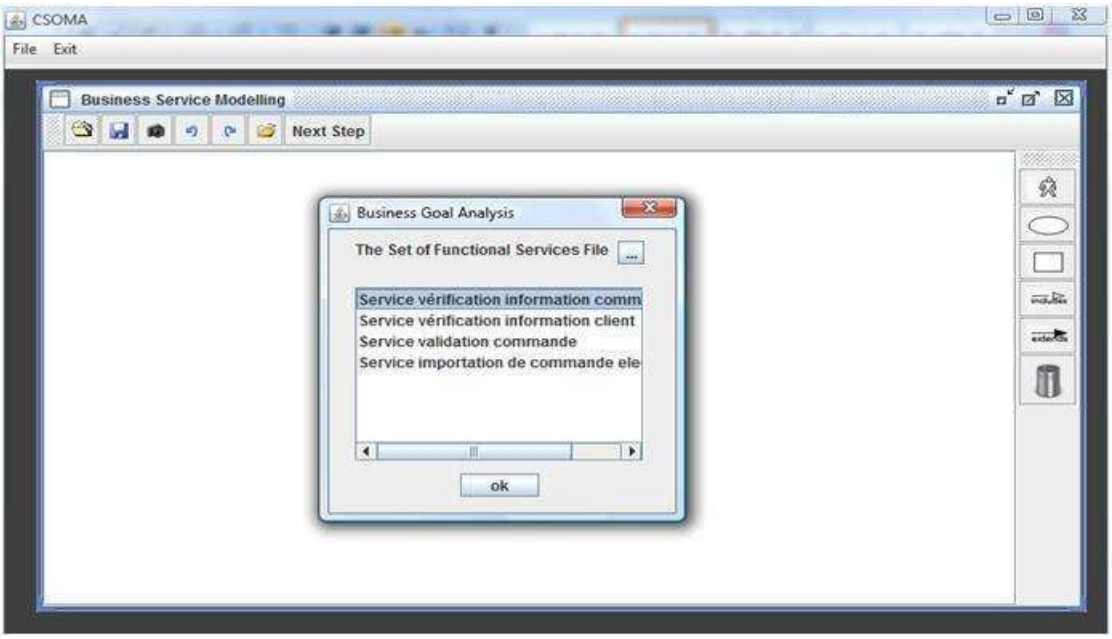


Figure A.33. Choix du service à modéliser

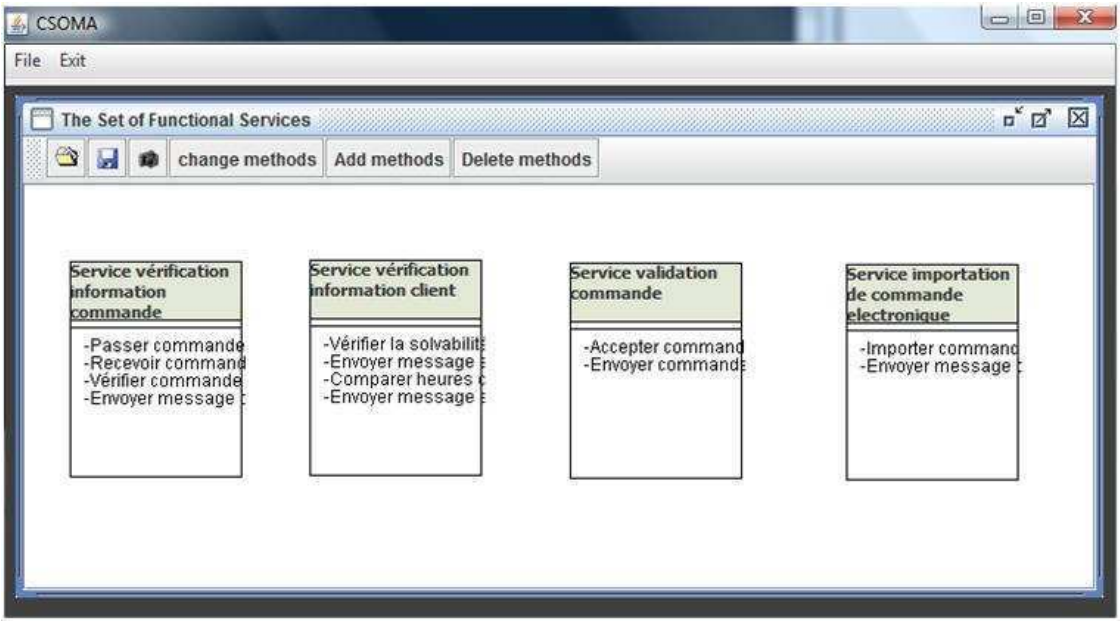


Figure A.34. Raffinement des services fonctionnels

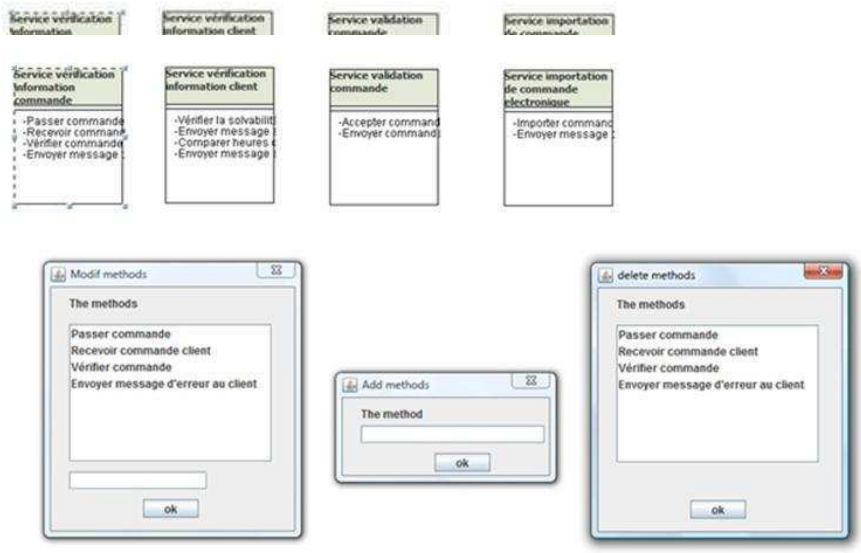


Figure A.35. Exemples de raffinement des services

4.3 Test et expérimentation

Nous avons montré précédemment quelques prises d'écrans relatives à l'atelier génie logiciel développé. Cet atelier a été testé par un ensemble d'utilisateurs afin d'évaluer les différentes fonctionnalités proposées. Le nombre de tests effectués sur une collection de cinq utilisateurs a donné des résultats prometteurs. Il s'agit essentiellement d'élaborer les différents modèles (modèle de motivation métier, modèle de processus métier, modélisation des services) et de les réutiliser par la suite pour l'identification des services et la réalisation des différents modèles de schémas d'orchestration des services.

Nous avons également réalisé deux tests qui évaluent la capacité du système à répondre à certaines spécificités. Ces deux tests ont été réalisés en utilisant l'environnement décrit dans le tableau suivant.

Système d'exploitation	Windows Vista business
Java Virtual machine	Sun Java Run Time Environment 1.5.0_10
RAM	2 GB
Processeur	Intel centrino Duo

Tableau A.4. Caractéristique de l'environnement de test

Le premier test consiste à identifier le temps relatif à la réponse du système concernant le regroupement des activités métier. Il s'agit d'évaluer le temps d'exécution de l'algorithme de regroupement présenté (cf. chapitre2, section 2.4.2, étape 2.3). Comme l'illustre la courbe, présentée dans la Figure A.36, l'augmentation du nombre d'activités au sein d'un processus métier, entraîne une augmentation exponentielle du temps d'exécution. Étant donné que dans la majorité des cas les activités métier contenues dans le processus ne dépassant pas les 60 activités, le temps écoulé reste raisonnable.

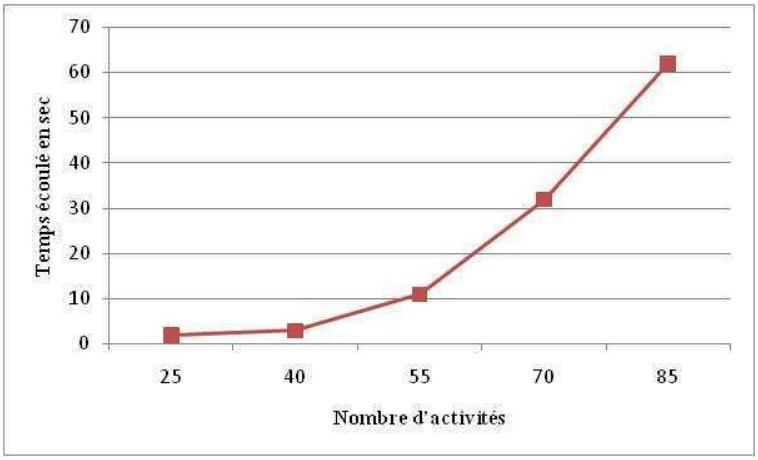


Figure A.36. Temps exécution de l'algorithme de regroupement des activités métier en des services fonctionnels

Quant au deuxième test, il s'agit d'évaluer la capacité du système à réaliser les transformations du modèle de service fonctionnel au modèle de spécification du service fonctionnel. Le temps indispensable pour réaliser les règles de transformation automatique (cf. chapitre 2, section 2.4.2, étape 2.4) évolue exponentiellement en fonction du nombre d'opérations contenues dans le modèle de service fonctionnel. Cette évolution est de plus en plus croissante quand le nombre d'opérations de service dépasse huit opérations. Cependant, par souci de granularité, le nombre d'opérations contenues dans un service ne doit pas dépasser dans la majorité des cas les six opérations. Ainsi, le temps écoulé pour les transformations reste gérable.

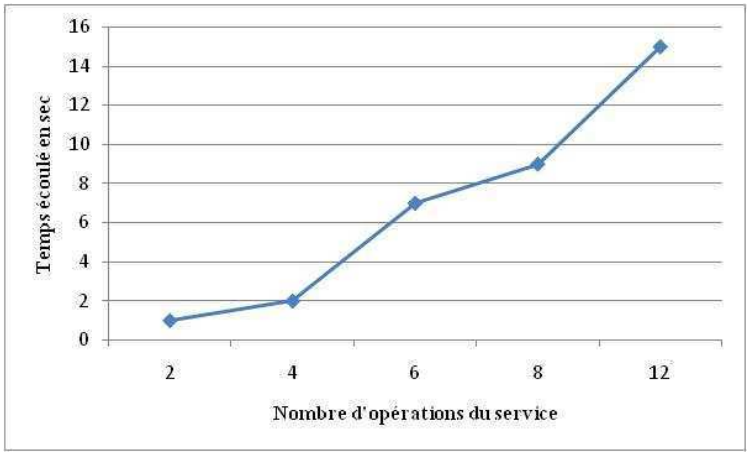


Figure A.37. Temps d'exécution des transformations entre les modèles

Actuellement notre atelier de génie logiciel a servi comme un support à la démarche CSOMA et à son application dans le cadre de l'entreprise *ETS*. Certes, notre atelier ne prend pas fin avec l'achèvement du travail de thèse. Plusieurs améliorations sont possibles telle que l'ajout d'autres modules ou son implémentation en réseau.

Annexe B

Prototype de construction du processus coopératif

1. Présentation du prototype CP^2 (Collaborative Process creation Prototype)

Cette partie présente le prototype relatif à la construction du processus coopératif. Ce travail a été réalisé en collaboration avec [Chaari et al., 2008] et a donné naissance à quelques publications [Boukadi et al., 2008a], [Chaari et al., 2008].

Le prototype envisagé dans [Boukadi et al., 2008a] se compose de quatre modules différents (voir Figure B.1) :

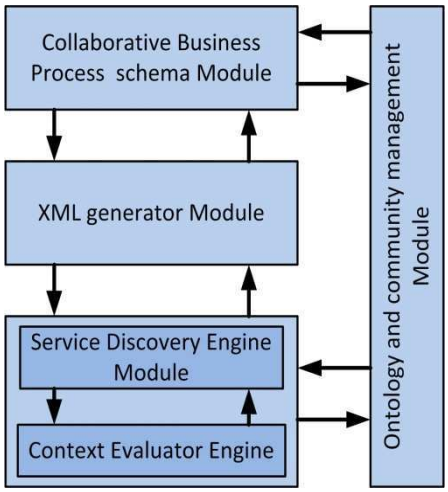


Figure B.1. Modules du prototype [Boukadi et al., 2008a]

- **Module de construction du schéma abstrait du processus coopératif**

Ce module offre une interface graphique pour la modélisation du processus coopératif abstrait à l'aide d'un ensemble de Goal Template, des branchements et des connecteurs. En utilisant ce module, le concepteur peut désigner la communauté d'entreprises en question, ses besoins fonctionnels (les méthodes génériques) ainsi que son contexte (à travers les paramètres de contexte de la Goal Template).

- **Module de génération de description XML**

Ce module réalise deux fonctions de base. La première consiste à générer le fichier XML contenant la description du Goal Template conformément à la DTD présentée dans la Figure B.2. La deuxième fonction consiste à réaliser le *mapping* du schéma abstrait du processus

coopératif vers un schéma exécutable sous le format BPEL. Ce processus doit être transmis par la suite au moteur d'exécution BPEL.

```
<?xml version="1.0" encoding="iso-8859-1"
standalone="no" ?>
<! Element GT (Id-Template, Id-Community, IGM+, ICx*>
<! Element Id-Template ID #REQUIRED >
<! Element Id-Community ID #REQUIRED >
<! Element IGM (#PCDATA)>
<! Element ICx (#PCDATA)>
```

Figure B.2. DTD pour la description d'un Goal Template

- **Module de découverte de service**

Ce module permet de découvrir les services domaines selon l'approche de découverte présentée dans le chapitre 3 de ce manuscrit de thèse (cf. section 3.4.3). Il est composé de deux sous modules : (i) le module de *matching* et (ii) l'évaluateur de contexte.

- **Module d'ontologie et de gestion de communauté**

Ce module permet la gestion de l'ontologie de catégorisation du contexte et des ontologies de domaines ainsi que la gestion des communautés (création, mise à jour et suppression).

2. Implémentation

Nous présentons dans ce qui suit les technologies utilisées pour l'implémentation du prototype ainsi que quelques prises d'écran. Il est important de souligner qu'au moment de la rédaction du manuscrit de thèse, seul le premier module du prototype CP^2 a été implémenté. Nous envisageons de développer la totalité de la plateforme de coopération dans nos travaux futurs.

- **Technologies utilisées**

Le prototype a été développé en utilisant le langage Java et sous l'environnement de développement Eclipse.

- **Prise d'écran du prototype**

L'interface principale du prototype est présentée dans la Figure B.3. Elle correspond au module de conception du schéma abstrait du processus. Cette interface comporte une barre d'outils contenant les différents éléments de modélisation, un explorateur de processus qui permet de naviguer au sein des différents éléments du processus, une partie représentant le résultat d'exécution du processus coopératif (apparaît en bas de la Figure B.3).

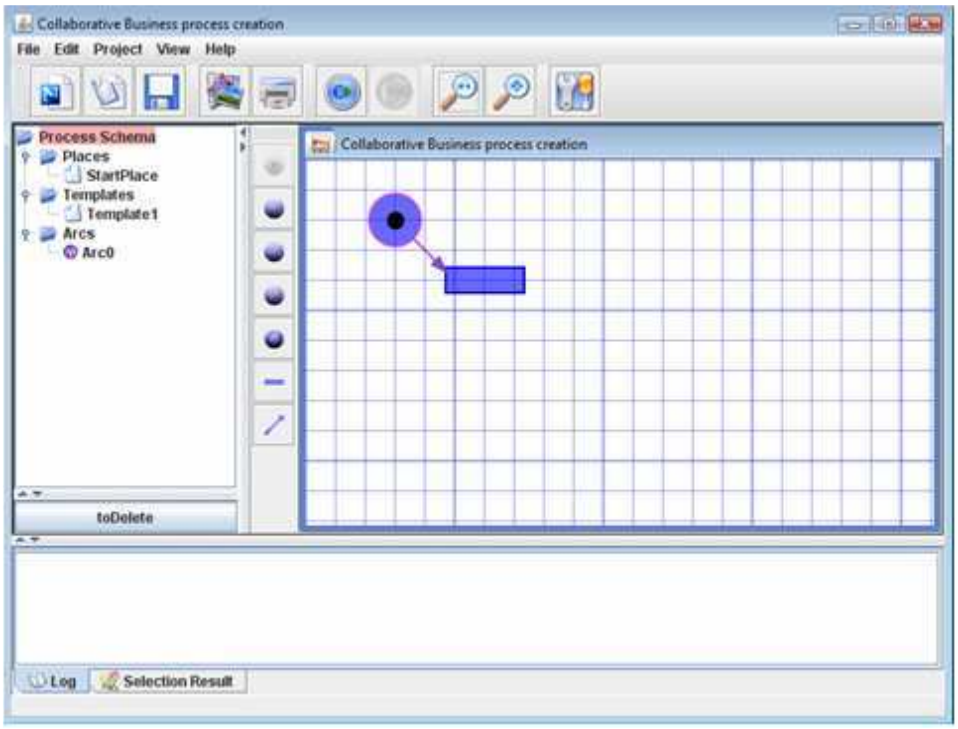


Figure B.3. Conception du schéma abstrait du processus coopératif

Annexe C

Adaptation des services domaines au contexte à base de tissage d'aspects

Cette partie complète la dimension technique du chapitre 4 relatif à l'adaptation des services domaines au contexte à base de tissage d'aspects. Nous allons commencer par présenter les différents outils supports à la programmation orientée aspect et les situer les uns par rapport aux autres. Ensuite, nous allons exposer quelques détails d'implémentation de l'architecture du service domaine.

1. Outils supports à la programmation orientée aspects

La programmation orientée aspect est un paradigme de programmation qui n'est pas lié à un langage de programmation bien particulier. De ce fait, elle peut être appliquée à n'importe quel langage, à condition qu'il propose un tisseur (*weaver*).
Les outils les plus utilisés pour la programmation orientée aspect sont : *AspectJ*, *JAC*, Jboss *AOP*, *Spring AOP*. Ces outils sont classés selon deux approches : l'approche « langage » (*AspectJ*) et l'approche « framework » (*JAC, Jboss AOP, Spring AOP*). Chacune de ces approches possède ses propres techniques d'implémentation des concepts de la programmation orientée aspect et présente quelques points de différence qui seront présentés dans ce qui suit.

- **Au niveau de l'aspect**

Dans le langage *AspectJ*, l'aspect est une entité logicielle qui définit les *pointcuts*, les *advices* ainsi que les méthodes. L'aspect dans *JAC* est une classe qui hérite de la classe *AspectComponent* contenue dans la bibliothèque *JAClibrary*. L'aspect définit les *pointcuts* ainsi que les *advices* qui sont écrits dans une classe nommée *wrapper*.
Le concept d'aspect dans *JBoss* est présenté par deux types de fichiers : d'une part, les méthodes d'interception ou les intercepteurs qui implémentent les codes d'*advice* définies dans des classes java et d'autre part, les *pointcuts* qui associent ces intercepteurs avec les *join points* définis dans des fichiers XML.
Dans Spring, *l'advice* et le *pointcut* sont implémentés dans des classes java qui sont configurées comme des *Beans*.

- **Au niveau de pointcut**

Dans le langage *AspectJ*, les *pointcuts* sont définies avec une syntaxe bien déterminée (le mot clé est *pointcut*). L'outil *JAC* les définit grâce à une API en utilisant des expressions régulières

qui précisent les *join points* en se basant sur les noms des classes, les noms des objets et les signatures des méthodes. Quant à *Jboss*, il définit les *pointcuts* en utilisant le langage XML. *Spring* emploie le langage XML mais peut aussi utiliser la programmation java pour définir les *pointcuts*.

- **Au niveau de types de join points**

Il existe plusieurs types de *join point*. Dans ce qui suit nous allons examiner quelques exemples de types de *join point* supportés par les outils de la programmation orienté aspect. Un *join point* peut correspondre à :

- L'exécution d'une méthode : ce type de *join point* est supporté par tous les outils,
- l'appel d'une méthode : supporté par tous les outils sauf l'outil *JAC*,
- l'exécution des constructeurs : supportée par tous les outils sauf *Spring*,
- les exceptions : l'*AspectJ* peut intercepter le début de l'exception. Cet événement n'est pris en charge par aucun des autres outils.

Il est important à noter que l'approche « framework » supporte moins de types de *join point* que l'approche langage.

- **Au niveau de types d'advice**

AspectJ et *Spring* assurent tous les trois types d'*advice* (« *before* », « *after* » et « *around* ») tandis que les outils *JAC* et *Jboss* ne fournissent que les *advices* de type « *around* ».

- **Au niveau du tissage**

Le tisseur *AspectJ* permet d'utiliser le code source ou le byte code comme entrée. C'est un tisseur au moment de la compilation. Les classes et les aspects dans *JAC*, *Jboss*, *Spring* sont compilés séparément et leur tissage se fait lors du chargement et de l'exécution de l'application.

– Synthèse de la comparaison des outils

Nous nous sommes basés sur un ensemble d'études réalisées dans le domaine de la programmation orientée aspect afin de synthétiser la comparaison entre les différents outils déjà présentés [Kersten, 2005], [Baltus, 2001].

Caractéristique / **Outils**	**Source**	**Compilateur**	**Détection du *pointcut***	***Weaving***	**Déploiement**	**Exécution**
AspectJ	.java ou .aj	Aspectj compile	Nom des *pointcuts*	Compilation Chargement	Déploiement statique	Programme Java
JAC	.java	Java compile, post processing	Nom des *pointcuts*	Compilation Chargement	Déploiement statique	Programme Java

Jboss AOP	.java	Java compile, post processing	Nom des *pointcuts*	Exécution	Déploiement à chaud et *proxies*[18]	Framework
Spring AOP	.java	Java compile	-	Exécution	Déploiement à chaud	Framework

Tableau C.1. Comparaison des outils supports à la programmation orientée aspect, inspirée de [Kersten, 2005]

Caractéristique / **Outils**	**IDE**	**Éditeur**	**Vues**	***Débuggeur***	**Librairies**	**Documentation**
AspectJ	eclipse Jdevelopper Jbuilder netbeans	Assistance pour les *advices*	*Visualizer*	Java	-	++++
JAC	eclipse	Assistance pour les *advices*	-	Java	-	+++
Jboss AOP	eclipse	Interface utilisateur pour les *pointcuts* et les *advices*	Aspect manager	Java	++++	++
Spring AOP	eclipse	-	-		+++	+

Tableau C.2. Comparaison des outils supports à la programmation orientée aspect, inspirée de [Kersten, 2005]

L'analyse des deux tableaux nous montre l'importance de l'utilisation de l'outil *Jboss AOP*. Le déploiement à chaud, le *Weaving* au moment de l'exécution et les *proxies* constituent des outils indispensables pour la réalisation de notre approche d'adaptation de la logique métier des services d'une manière dynamique. En effet, le déploiement à chaud est un outil très puissant pour la flexibilité et l'extensibilité de la solution. Par ailleurs, grâce au serveur d'application associé à *Jboss AOP*, appelé *Jboss AS*, l'interception des invocations est réalisée au moment de l'exécution. Cette caractéristique constitue un atout garantissant une solution dynamique. Les différents avantages cités nous ont guidés dans le choix du Framework *Jboss AOP* et de son serveur d'application afin de réaliser notre approche d'adaptation.

Nous avons également réalisé une comparaison entre l'utilisation de *Jboss AOP* et l'intégration des aspects en utilisant le langage BPEL (notamment le langage XML). Le Tableau C.3 montre une comparaison des deux possibilités.

[18] Un serveur mandataire ou (*proxy*) est un serveur qui a pour fonction de relayer des requêtes entre un poste client et un serveur.

Jboss AOP	Intégration des aspects en utilisant le langage BPEL (XML)
- Le *Pointcut* est exprimé en utilisant le langage XML - L'*advice* est exprimée en utilisant le langage java - Le *weaving* s'effectue dynamiquement (pendant exécution) et ceci sans aucune intervention du programmeur (le serveur se charge de la réalisation du *weaving*) - La possibilité d'avoir plusieurs *advices* pour le même *joint point* et leur ordonnancements - La réutilisation des aspects dans plusieurs processus BPEL - Les *Join points* peuvent être de nature différente (appel de méthode, appel de constructeur, *etc.*)	- Les *pointcuts* et les *advices* sont exprimés en utilisant le langage XML - Le *weaving* s'effectue d'une manière dynamique (appelé aussi à l'exécution) et nécessite une programmation (gestion des *threads*) - L'*advice* est limitée aux instructions BPEL - *Join points* sont limités aux activités BPEL

Tableau C.3. Comparaison entre Jboss AOP et l'intégration des aspects en utilisant le langage BPEL

La comparaison présentée dans le Tableau met en évidence l'avantage de l'utilisation de *Jboss AOP*. En effet, *Jboss AOP* permet d'assurer l'indépendance vis-à-vis du langage et du moteur d'exécution BPEL. Par contre, l'utilisation du langage XML pour les aspects et leurs tissages au moment de l'exécution nécessite un couplage fort avec le moteur BPEL dans le sens où les développeurs doivent modifier le moteur BPEL pour intégrer le tissage d'aspects.
Ces différentes constatations nous ont motivées dans le choix de l'outil *Jboss AOP* pour l'adaptation des services.

2. Détails d'implémentation

Nous allons préciser dans ce qui suit quelques détails d'implémentation relatifs à l'architecture du service domaine. Tout d'abord, nous allons présenter une étape importante pour la réalisation de notre approche présentée dans le chapitre 4 de ce manuscrit, à savoir le déploiement du moteur d'exécution BPEL au sein du serveur *Jboss AS*. Ensuite, nous allons exposer quelques prises d'écran du code des modules développés au sein du prototype.

2.1 Déploiement d'ActiveBpel Engine dans Jboss AS

Nous avons déjà précisé dans le chapitre 4 de ce manuscrit de thèse (cf. section 4.5.2) que nous avons retenu *ActiveBpel Engine* sous sa version 5.0.2 comme moteur d'exécution des services domaines.
ActiveBpel est le moteur retenu et ce pour plusieurs raisons :

- La compatibilité avec le serveur d'application *Jboss AS*

- L'extensibilité
- La stabilité
- La présence d'interface d'administration
- Présence des modules de *monitoring* qui permettent de surveiller le moteur ainsi que les différentes exécutions des processus

Bien que le moteur *ActiveBpel Engine* et le serveur *Jboss AS* soient compatibles, il est fondamental pour assurer le tissage d'aspects, de déployer le moteur *ActiveBpel Engine* au sein du serveur *Jboss AS*. Cette constatation nous l'avons déduite à partir des différents tests réalisés. En effet, comme le montre la Figure C.1, *Jboss AS* et *ActiveBpel Engine* possèdent chacun un *class loader* (*i.e.*, class loader est une classe qui permet à une plateforme Java de charger en mémoire des classes Java compilées) indépendant, ce qui rend par conséquent, l'activité du *weaving* entre les classes impossible.

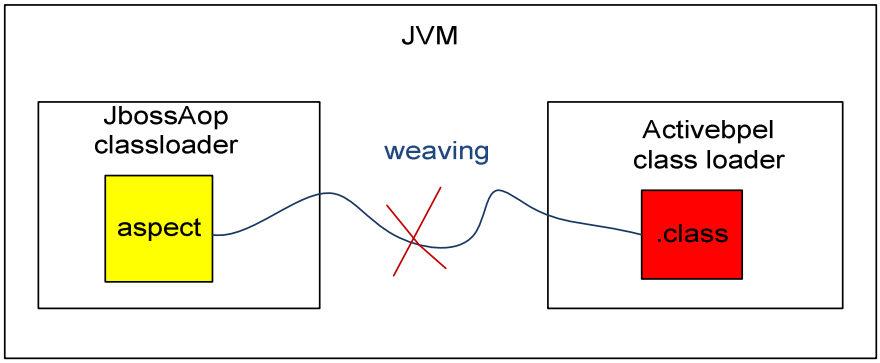


Figure C.1. État du *Java Virtuel Machine* lors de l'exécution d'un processus non déployé au sein de *Jboss AS*

Pour remédier à ce problème, nous avons proposé une démarche de déploiement décrite dans [Boukadi, 2009]. Cette démarche s'appuie sur un certain nombre d'études comme celles réalisées par [Umberto, 2008].

2.2 Prises d'écran du prototype du service domaine

Nous présentons dans ce qui suit quelques prises d'écran du prototype du service domaine. Dans un premier temps, nous allons présenter le dévelopement des modules d'adaptation du service domaine. Dans un deuxième temps, nous allons montrer quelques détails relatifs au déploiement et au test de l'architecture d'adaptation.

- **Modules d'adaptation de la logique métier du service domaine**

– Gestionnaire de contexte

La Figure C.2 montre un extrait du code relatif au gestionnaire de contexte que nous avons développé. Il s'agit essentiellement des trois opérations prévues pour ce module : lire les paramètres de contexte, détecter les changements contextuels, établir la liste des paramètres de contexte afin d'informer le module d'activation d'aspect.

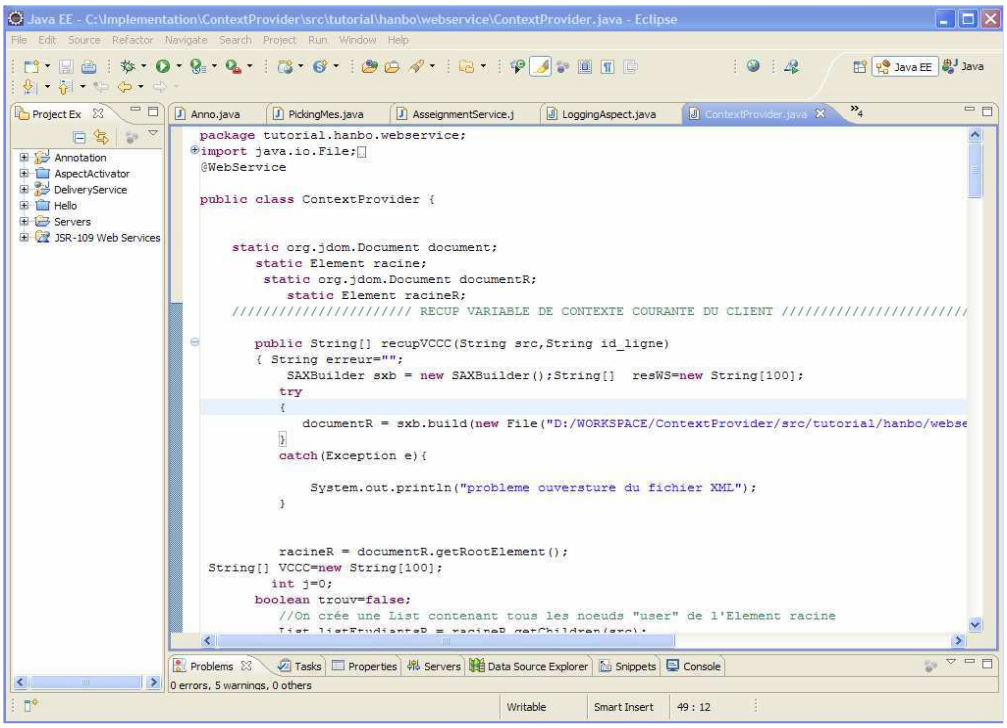


Figure C.2. Gestionnaire de Context (*Context Manager Module*)

– Module d'activation d'aspect

La Figure C.3 montre un extrait du code relatif au module d'activation d'aspect. Les différentes fonctionnalités attribuées à ce module ont été dévelopées. Il s'agit des deux opérations : de sélection d'aspect et de création du fichier de *Binding* indispensable pour le tissage d'aspect.

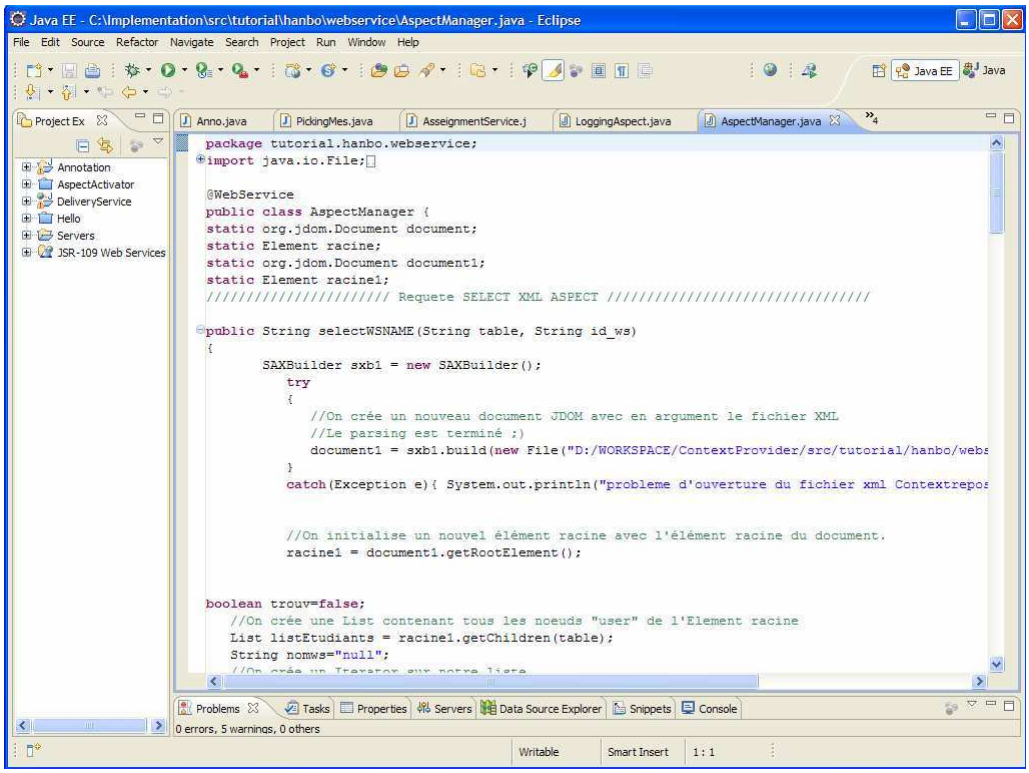


Figure C.3. Module d'activation d'aspect (*Aspect Activator Module*)

- **Déploiement et test de l'architecture d'adaptation**

La Figure C.4 montre le déploiement des deux modules d'adaptation au sein du moteur de service Web *Jboss WS* fourni avec le serveur *JBoss AS*.

Figure C.4. Déploiement du gestionnaire de contexte et du module d'activation d'aspects sous *Jboss WS*

Quant à la Figure C.5, elle illustre le déploiement du processus BPEL de l'exemple du service domaine de livraison déjà décrit en détail précédemment (cf. chapitre 4, section 4.6)

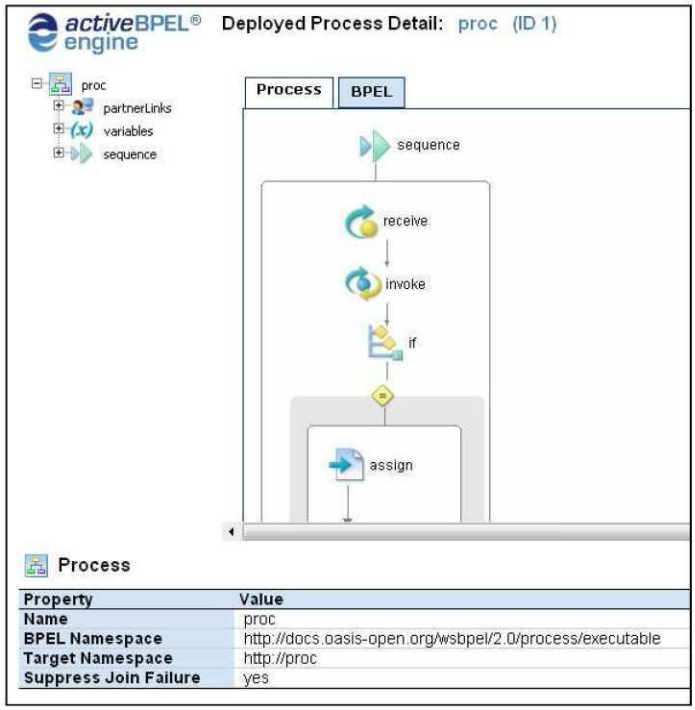


Figure C.5. Déploiement du processus BPEL relatif au service domaine de livraison sous *ActiveBpel Engine*

Afin de valider l'exemple présenté dans le chapitre 4 de ce manuscrit de thèse (cf. section 4.6) et visionner par la suite le comportement du service, nous avons implémenté une classe de test (voir Figure C.6). Cette classe permet de garder trace de l'exécution des deux modules d'adaptation (gestionnaire de contexte et le module d'activation d'aspect). Le résultat du test est représenté dans la Figure C.7.

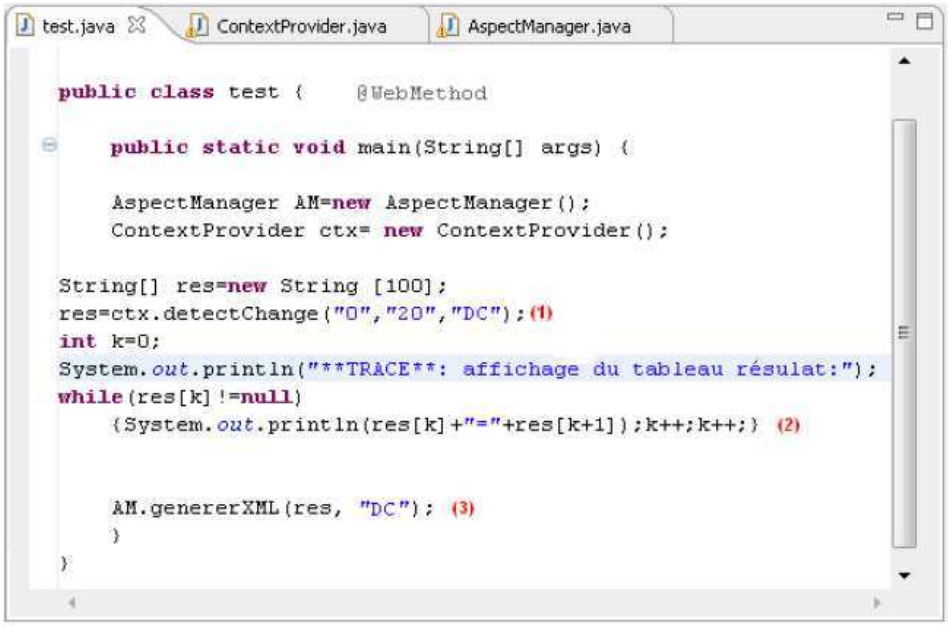


Figure C.6. Classe Test

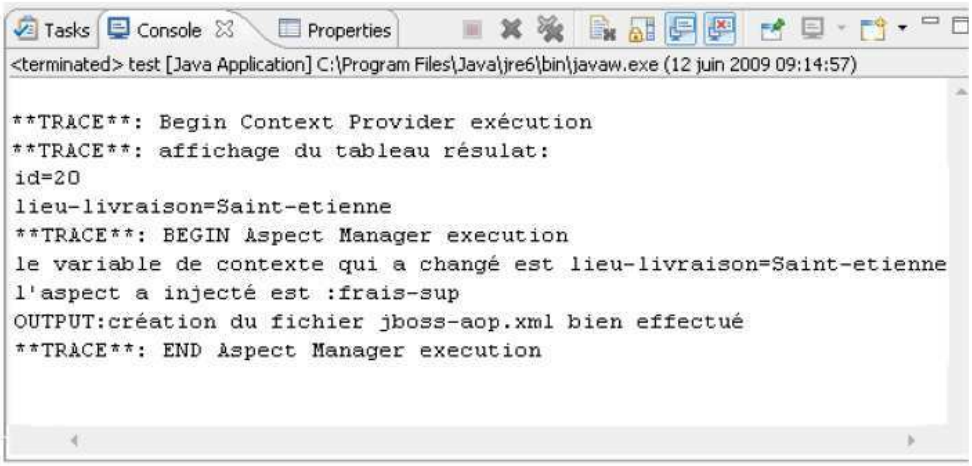


Figure C.7. Résultat d'execution de la classe Test