Faculté de Technologie *Département D'Informatique*

Interopérabilité Sémantique des Systèmes d'Informations d'Entreprise : Approche Dirigée Par les Modèles

THÈSE

Présentée et soutenue publiquement le : **16 / 12 / 2014**

Pour l'obtention du

Doctorat en Sciences

de l'Université Djilali Liabes – Sidi Bel Abbes

(Spécialité Informatique)

Par

SOLTANI Mokhtar

Composition du jury

Président: *Pr MALKI Mimoun*	*Pr à l'Université de Sidi Bel Abbes*
Directeur de thèse: *Dr BENSLIMANE Sidi Mohamed*	*MCA à l'Université de Sidi Bel Abbes*
Examinateurs:	
Dr ADJOUDJ Réda	*MCA à l'Université de Sidi Bel Abbes*
Dr BOUCHIHA Djeloul	*MCA au Centre Universitaire de Naama*
Dr DEBBAT Fatima	*MCA à l'Université de Mascara*
Dr HAMOU RÉDA Mohamed	*MCA à l'Université de Saida*

Résumé

L'évolution des méthodes et des technologies dans le domaine du génie logiciel fait émerger de nouvelles approches pour le développement d'applications. Ainsi, ont fait leur apparition, le paradigme orienté objets, ensuite le paradigme orienté composants et de nos jours, le paradigme orienté services. Tous ces paradigmes reposent sur les principes de base du génie logiciel : l'abstraction, la séparation de préoccupations et la modularité. Chaque nouvelle approche est basée sur les anciens paradigmes tout en y ajoutant de nouveaux atouts.
L'idée de l'approche orientée services est de construire rapidement des applications par l'assemblage d'un ensemble de services. Le résultat de cet assemblage est appelé « application composite ». La mise en place effective de cette approche permet de faciliter l'intégration et l'interopérabilité des systèmes d'informations d'entreprise qui sont hétérogènes et n'ont pas été conçus pour être interopérables. L'approche orientée services propose un cadre pour résoudre cette problématique par le développement des applications interopérables, dynamiques, et complètement réparties.
L'intégration des paradigmes modélisation des processus métiers (BPM), le développement orienté service (SOC) et le développement dirigé par les modèles (MDD) pour améliorer le développement des solutions orientées services à partir des modèles métiers est devenu actuellement une exigence fondamentale pour beaucoup d'entreprises. La modélisation des processus métiers est également positionnée au centre des efforts de développement de logiciel, car la construction de ces modèles constitue explicitement la base pour la définition des services. Dans ce contexte, nous avons proposé à travers cette thèse une méthode permettant d'identifier et de spécifier les composants d'une architecture orientée service SOA à partir d'un modèle de processus métier collaboratif de haut niveau afin d'assurer la construction automatique des applications interopérables.

Mots clés: interopérabilité, modélisation des processus métiers (BPM), architecture orientée services (SOA), architecture dirigée par les modèles (MDA), identification des services, spécification des services.

Remerciement

Je tiens à remercier tout particulièrement Dr Sidi Mohamed BENSLIMANE pour avoir accepté la direction de cette thèse, pour ses remarques pertinentes, pour son soutient tout au long de ce doctorat, et pour la grande liberté qu'il m'a laissée pour mener à bien ce travail.

J'adresse mes plus vifs remerciements aux membres du jury pour avoir accepté cette charge ainsi que l'intérêt qu'ils ont porté à mes travaux de recherche : je cite Monsieur Mimoun MALKI, Monsieur Réda ADJOUJ, Monsieur Djeloul BOUCHIHA, Madame Fatima DEBBAT, et Monsieur Mohamed HAMOU REDA.

Je remercie également tous mes enseignants et tous ceux qui ont contribué de près ou de loin à l'aboutissement de ce travail: je cite Monsieur Chakir MOKHTARI.

Un grand merci à mes amis et à mes collègues.

Je termine par un grand remerciement à ma famille et en particulier mes parents pour leurs encouragements et pour leur soutien moral.

Table des matières

Table des figures

Introduction Générale

Introduction générale

1. Contexte

L'environnement des entreprises a largement évolué ces dernières années, il est devenu plus complexe et imprédictible. Aujourd'hui les entreprises sont organisées en réseaux, au sein desquels différents acteurs peuvent interagir. 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. Cette dernière doit 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.).

La compétitivité de ces entreprises est profondément liée à la capacité de structurer, partager et échanger les connaissances et le savoir-faire avec l'ensemble des participants au réseau collaboratif, et aussi à la réactivité aux changements métiers. Ce besoin d'échanger des connaissances oblige les entreprises d'évoluer leurs systèmes d'informations et leurs applications afin de les rendre interopérables.

Pour relever ces défis, il est nécessaire d'adopter de nouveaux paradigmes et de nouvelles technologies. Aujourd'hui, l'ingénierie logicielle est marquée particulièrement par l'émergence du paradigme SOC (Service Oriented Computing).

En effet, SOC permet le développement des systèmes d'information distribués, flexibles, agiles et interopérables. SOC s'appuis principalement sur le concept « Service » qui se définit comme une entité informatique autonome, auto-descriptive, et indépendante des plateformes permettant le support de la composition des applications distribuées en couplage faible tout en optimisant les coûts et les délais [Papazoglou, 2007].

L'interopérabilité des applications d'entreprise permet d'assurer l'échange des données, des fonctionnalités et des services d'une manière transparente à l'utilisateur. Chaque fonctionnalité, service, ou donnée possède son propre modèle. Un certain nombre de transformations entre ces modèles sont indispensable pour assurer l'interopérabilité entre les différentes entités hétérogènes de l'entreprise.

Par ailleurs, l'OMG (Object Management Group) a proposé l'architecture dirigée par les modèles MDA (Model-Driven Architecture) [OMG, 2003] pour le développement des systèmes d'information distribués. MDA propose une nouvelle méthode de spécification des systèmes informatiques basée sur les différentes perspectives du système à développer. En premier lieu, elle décrit les fonctionnalités et le comportement du système sans tenir compte les caractéristiques technologiques. Puis, la spécification des fonctionnalités et du comportement fournis est utilisée pour créer les systèmes informatiques conformément à une technologie spécifique.

La séparation claire et explicite entre les modèles indépendants et dépendants d'une technologie permet d'apporter certains avantages au développement des logiciels. En effet, la logique métier est protégée contre les changements technologique. De plus, elle apporte une grande flexibilité quand les systèmes informatiques doivent évoluer pour utiliser une autre technologie ou atteindre d'autres exigences. L'approche MDA a comme but de rendre les systèmes d'informations plus indépendants d'une technologie particulière en fournissant la portabilité et l'interopérabilité au niveau modèles.

2. Problématique

La nécessité d'assurer l'interopérabilité entre les applications d'entreprise a causé une sous problématique que nous la définissant de la manière suivante : « comment définir une approche d'identification et de spécification des systèmes orientés services interopérables indépendamment des technologies caractérisées par leur bas niveau d'abstraction ?».

Notre objectif principal est de soutenir l'interopérabilité des systèmes d'informations d'un ensemble de participants à une collaboration en adoptant une approche dirigée par les modèles. Nous avons essayé de résoudre cette problématique dès la phase de conception. L'idée initiale consiste à utiliser une approche descendante (top down) pour dériver les modèles d'une architecture applicative plus adaptée au besoin d'interopérabilité à partir d'un modèle métier collaboratif de haut niveau.

Deux questions peuvent être adressées afin de réaliser cet objectif :

1) Est-il possible d'identifier les services logiciels à partir de modèle métier de haut niveau ?

2) Comment représenter la structure et le comportement des services identifiés indépendamment à la technologie ?

3. Contributions :

Comme réponse à ces questions, nous proposons une approche systématique pour transformer les modèles conceptuels des processus collaboratifs en modèles et spécifications concrètes d'un processus métier exécutable puis identifier à partir de ce dernier un ensemble de services implémentant le processus métier en entré.

Pour accomplir notre but, nous avons proposé une approche permettant de dériver les composants d'une Architecture Orientée Service à partir d'un modèles de processus métier collaboratif de haut niveau afin d'assurer la construction automatique des applications interopérables. L'approche proposée implique les étapes suivantes :

a) Analyse et conception des processus collaboratifs en se basant sur un point de vue métiers pour représenter la vue de la collaboration B2B, c.-à-d. la définition des besoins métiers et des objectifs métiers communs de la collaboration B2B.

b) Annotation de ce processus métier inter-organisationnel par une ontologie. Le but est de représenter formellement les connaissances métiers encapsulées dans le diagramme de

processus afin de faciliter la dérivation des processus exécutables ainsi que la génération automatique d'une architecture orientée services qui exécutent le processus métier globale.

c) Transformation du processus métier annoté en un processus exécutable qui ne comporte que des activités automatiques qui peuvent être implémentées par des services logiciels.

d) Génération de la solution technologique des modèles de processus métiers, c.-à-d. les servies requis pour exécuter le processus collaboratif.

Nous avons défini cet ensemble d'étapes constituant une méthode d'identification des services à partir d'un processus métier en se basant sur une ontologie de processus métier. Cette ontologie est utile pour l'annotation du modèle de processus métier et est considérée comme une source de connaissances utilisable pour identifier automatiquement les services logiciels. Elle est basée sur deux principes :

1) Unifier les différents méta-modèles du processus métiers existants.

2) Fournir les propriétés nécessaires pour dériver les services logiciels à partir d'un modèle de processus métier.

Nous avons formalisé le problème d'identification des services comme un problème d'optimisation multi-objectif qui consiste à regrouper les activités métiers en des clusters significatifs (optimaux) en optimisant certains métriques. Un prototype a été développé afin de valider l'approche proposée.

Une fois les services logiciels sont identifiés, une phase indispensable est nécessaire ; c'est la spécification des services identifiés. Pour réaliser cette phase, nous avons proposé une approche concrétisée par un modèle conceptuel de service, un profile UML nécessaire à la modélisation des services identifiés, et une étude de cas sur la spécification automatique des services d'un système d'enseignement à distance.

4. Organisation du mémoire

Le travail réalisé est organisé en six chapitres :

Le premier chapitre présente une vue d'ensemble sur la modélisation des processus métier d'entreprises, architecture dirigée par les modèles et l'interopérabilité dirigée par les modèles.

Le deuxième chapitre présent les différents concepts de base sur l'architecture orientée services SOA ainsi que les principes de modélisations SOA.

Le troisième chapitre présente les métaheuristiques et l'optimisation multi-objective, où nous mettons l'accent sur la présentation des métaheuristiques les plus répandues.

Le quatrième chapitre est divisé en deux parties. La première partie présente un état de l'art sur les approches d'identification de services au niveau duquel nous faisons une synthèse des approches d'identification des services déjà existantes, notamment les

approches de types descendantes, ascendantes et mixtes, pour pouvoir par la suite dégager les limites de chacun de ces types d'approches. La deuxième partie présente un état de l'art sur la spécification des services à base MDA.

Le cinquième chapitre présente l'identification automatique des services à base d'algorithmes génétiques. Dans ce chapitre nous allons présenter notre approche qui consiste à identifier automatiquement des services logiciels en se basant sur un algorithme génétique qui serve à produire des clusters de services optimaux. Par la suite, nous allons présenter notre prototype et les tests que nous avons réalisés pour valider notre approche.

Nous présentons, dans le sixième chapitre, un ensemble d'étapes ainsi qu'un modèle conceptuel indépendant des standards technologiques pour spécifier automatiquement les services identifiés précédemment par notre approche. La méthode proposée est illustrée par une étude de cas sur la spécification des services d'un système d'enseignement à distance.

En fin, nous concluons ce mémoire en récapitulant les points forts de notre approche, et les perspectives de ce travail.

Chapitre **1**

Modélisation d'entreprise, MDA et Interopérabilité

Chapitre 1 Modélisation d'entreprise, MDA et Interopérabilité

1. Introduction

La modélisation d'entreprise consiste à décrire l'organisation et les processus opérationnels d'une entreprise, soit dans le but de simuler ces processus pour comparer divers scénarios, soit dans le but de les analyser et de les restructurer pour améliorer les performances de l'entreprise. Elle propose un cadre formel où les processus et les interactions entre eux sont décrits.

Une approche de modélisation peut être décrite par un processus de modélisation, des vues, des concepts et des langages de modélisation. Un processus de modélisation consiste au développement d'un ensemble de modèles de description du système étudié. Grâce au modèle, nous pouvons décrire les flux d'information, de matière, ainsi que les interactions qui existe entre les diverses entités de l'entreprise (processus, information, acteurs, etc.)

Dans le contexte de développement du Système d'Information (SI), les modèles des processus métiers sont considérés comme un préalable nécessaire à la réalisation du SI. Ils sont représentés dans des niveaux d'abstraction plus élevé et avec des langages de modélisation conçus spécialement pour être compréhensible par tous les acteurs de l'entreprise. Nous citons par exemple le langage BPMN [OMG11, 2011]. Ces modèles sont considérés comme une base essentielle pour exprimer les besoins.

Un système d'information permet donc d'automatiser l'exécution de certaines activités des processus métiers (processus exécutable). Pour se faire il est nécessaire de disposer d'un moyen pour transformer les modèles métiers qui sont indépendant de toute spécification informatique en des modèles plus spécifique à la conception de l'architecture du SI. L'architecture dirigée par les modèles (MDA) [Mukerji et Miller, 2003] a fait son apparition pour remédier à ce problème.

2. La modélisation dans l'entreprise

Une entreprise est une structure économique et sociale composée de gens et de processus dans le but de fournir des produits et des services à des clients [Vernadat, 1996].

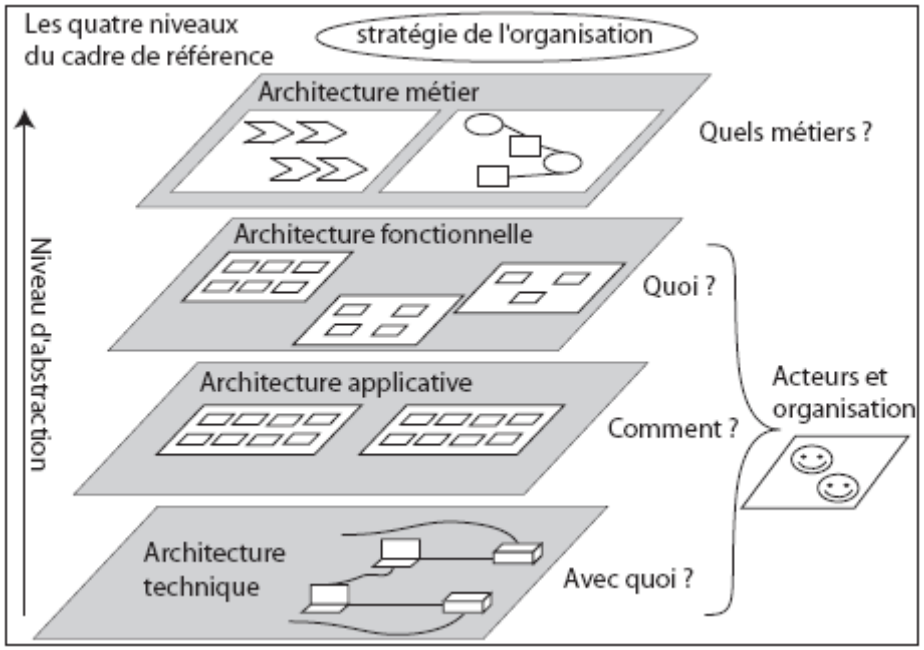
Aujourd'hui, la quasi-totalité des activités de l'entreprise, à tous les niveaux, est gérée par le système informatique de l'organisation. Dans un tel contexte, modéliser l'organisation se limite souvent à modéliser son système informatique. La figure 1.1 représente un cadre de référence de modélisation et de conception des SI d'entreprise. Selon le cadre de référence (figure 1.1), le SI peut se décomposer en 4 niveaux d'architecture [Mathieu, 2004] :

- niveau d'architecture métier : Il s'agit du niveau le plus abstrait et le plus proche de la stratégie de l'organisation. Il doit répondre à la question "Quels sont les métiers de l'entreprise ?". Une manière de répondre à cette question est de modéliser les processus de l'entreprise.

- niveau d'architecture fonctionnelle: Une fois que les métiers sont définis, il faut répondre à la question "que va-t-on faire pour être capable de faire ces métiers ?". C'est le "Quoi". Il s'agit de l'étape de spécification du système informatique en termes d'applications.

- niveau d'architecture applicative: Il a pour rôle de donner les clés permettant de réaliser les spécifications définies au niveau de l'architecture fonctionnelle. C'est le "Comment".

- niveau d'architecture technique: Il s'agit du niveau le moins abstrait dans la hiérarchie des niveaux d'architecture du SI. Il doit répondre à la question "Avec quels composants va-t-on réaliser les applications définies au niveau de l'architecture applicative. C'est le "Avec Quoi".



1Figure 1.1 Cadre de référence de modélisation des systèmes d'informations [Mathieu, 2004]

D'après ce cadre de référence, il est facile de comprendre l'importance de la modélisation des métiers de l'entreprise dans le but de modéliser son système d'information. En effet, ce sont les métiers de l'entreprise qui décident du contenu du système d'information.

Or les métiers se décrivent très bien à l'aide des processus créateurs de valeur ajoutée de l'entreprise. Dans ce cadre, on voit bien l'intérêt d'une modélisation orientée processus. Un processus, au sens métier, manipule à la fois des données statiques qui représentent le "cœur" du métier et des données dynamiques qui sont liées aux flux échangés par le processus. Les données statiques évoluent peu et peuvent donc être parfaitement décrites par un modèle de données. Les données dynamiques sont beaucoup plus volatiles, car elles sont liées aux flux, et les flux évoluent rapidement dans le contexte dynamique de l'entreprise. C'est ici où ce qu'on appelle le workflow ("science de la gestion des flux") prend toute son importance.

La conception d'un système informatique va utiliser la modélisation d'entreprise comme outil de spécification. On peut noter que même dans le cas d'un système complètement nouveau, une action de conception ne partira jamais de rien. Il est beaucoup plus courant de faire évoluer les installations existantes que d'en créer de nouvelles.

2.1 Concepts de la modélisation d'entreprise

Un modèle est une abstraction et une représentation simplifiée de la réalité. Il agit comme un filtre sur la partie du monde réel qu'il représente, ne conservant que l'information essentielle pour l'étude et supprimant les détails inutiles. *Un modèle est une représentation utile d'un sujet ; il représente une abstraction (plus ou moins formelle) de la réalité (ou de l'univers de discours) exprimé dans un formalisme (ou langage) défini par des concepts de modélisation adaptés au besoin de l'utilisateur. Ces concepts de modélisation forment les éléments d'un langage de modélisation défini par une syntaxe et une sémantique particulières* [Baïna, 2006]. La démarche générale consiste à abstraire les relations entre les entités modélisées. Ce processus d'abstraction est appelé *modélisation* [Mathieu, 2004]. Nous faisons la distinction entre la modélisation et le formalisme utilisé pour effectuer le modèle. La modélisation est une représentation abstraite du monde réel, tandis que le formalisme de modélisation est utilisé pour décrire cette représentation. Les modèles d'entreprise doivent permettre :

- la représentation du système, tant son état actuel que les différentes possibilités d'évolution,
- l'évaluation (du système existant ou à venir),
- l'optimisation des performances du système.

En plus de ces besoins techniques, les modèles doivent remplir un rôle de support aux acteurs humains du projet en favorisant :

- la communication à l'intérieur du projet,
- la communication dans l'espace (échanges entre les acteurs pouvant appartenir à des structures différentes de compétences différentes),
- la communication dans le temps (archivage et documentation),
- les décisions de conception ou de modification du système.

Pour réaliser un modèle, il est nécessaire d'adopter une démarche structurée, ceci afin d'obtenir un résultat utilisable. C'est l'intérêt des méthodes de modélisation. Dans le présent travail de thèse, nous avons focalisé seulement sur la modélisation des processus métier de l'entreprise.

2.2 Les objectifs de la modélisation

La modélisation de l'entreprise est vue comme un moyen pour :

- Offrir une meilleure représentation et une meilleure compréhension du fonctionnement d'une entreprise,
- Capitaliser la connaissance et le savoir-faire de l'entreprise pour une utilisation ultérieure,
- Rationaliser et structurer les échanges d'information,

- Concevoir et spécifier une partie de l'entreprise (aspects structurels, organisationnels, informationnels, fonctionnels ou comportementaux),
- Analyser certains aspects d'une partie de l'entreprise (analyse économique, organisationnelle, quantitative, qualitative, etc.)
- Simuler le comportement d'une partie de l'entreprise
- Contrôler, synchroniser ou coordonner certaines parties de l'entreprise (des processus, par exemple.)

En décrivant les processus et l'organisation, ces modèles d'entreprise recouvrent donc une part des connaissances nécessaires à l'architecture fonctionnelle du Système d'Information.

2.3 Différents types de modèles

Un modèle d'entreprise est le plus souvent composé de :

- Modèles économiques : définissent une vue analytique des coûts de l'entreprise, ces modèles sont utilisés pour étudier la compétitivité et l'efficacité des coûts des différentes sous-parties de l'entreprise.
- Modèles organisationnels : utilisés pour documenter de la structure et l'organisation de l'entreprise en termes de ligne de production, départements, cellules et centre de travail tout en représentant les responsabilités et les autorités affectées à chaque niveau de décision.
- Modèles informationnels : décrivent les structures et relations des éléments informationnels faisant parties des systèmes d'information de l'entreprise.
- Modèles d'activités : indiquent l'ensemble des opérations (ou actions) à enclencher lors de l'exécution du travail dans l'entreprise.
- Modèles de ressource : décrivent les caractéristiques règles de gestion et les actions supportées par les différents équipements en vue de l'exécution des activités de l'entreprise.
- Modèles de produits : sont utilisés pour représenter les caractéristiques géométriques ou non, les détails de conception des produits ainsi que leurs composants fabriqués dans l'entreprise à travers le cycle de vie du produit.

Ces modèles d'entreprise permettent d'aider à l'étude, la compréhension et la collection de la connaissance et du savoir de l'entreprise.

2.4 Modélisation des processus métiers

La vision processus joue un rôle important dans les théories des organisations comme dans le domaine système d'information où la modélisation des processus est considérée comme un préalable nécessaire à la conception d'un système d'information organisationnel [Morley, 2006].

La définition de référence d'un processus est aujourd'hui celle qui est donnée par (ISO 9000, 2000): " *Un ensemble d'activités corrélées ou interactives qui transforme des éléments d'entrés en éléments de sortie*". Dans la gestion par les activités, un processus est défini comme : " Une séquence d'activités différentes reliées par des relations client fournisseur qui s'enchaînent à

partir d'un facteur de déclenchement commun". Où une activité est " *un ensemble de tâches élémentaires réalisées par un individu ou un groupe, faisant appel à un savoir-faire spécifique, homogènes du point de vue de leurs comportements de coût et de performance, permettant de fournir un output à un client interne ou externe à partir d'un panier d'input*".

[Morley, 2006] présente un ensemble de concepts pour une représentation détaillée d'un processus. Il considère un processus comme un ensemble d'activités pour réaliser un objectif bien déterminé. La responsabilité d'exécution de tout ou partie des activités correspond à un rôle. Le déroulement du processus utilise des ressources et peut être conditionné par des événements, d'origine interne ou externe. L'agencement des activités correspond à la structure du processus. Cette définition est illustrée sous forme d'un diagramme de classe UML (figure 1.2).

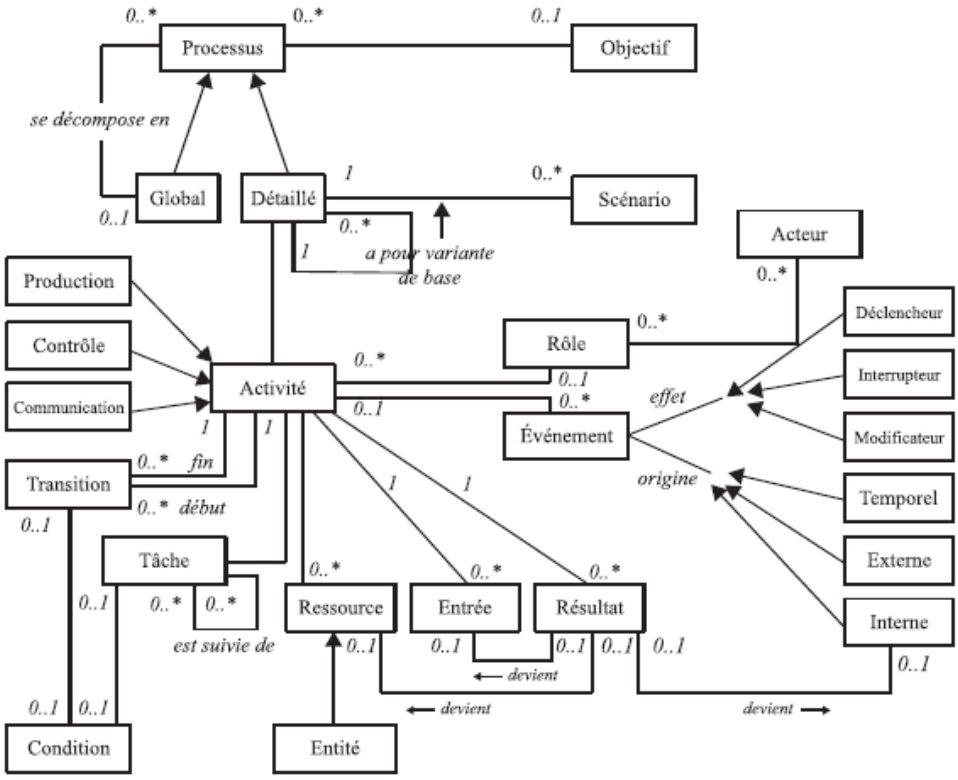


Figure 1.2 Méta-Modèle d'un processus [Morley, 2006]

2.4.1 Langages de modélisation des processus métiers

Réseau de Petri

Un réseau de Petri est un outil graphique et mathématique appliqué à plusieurs domaines où les notions d'événements et d'évolutions jouent un rôle important. Cette théorie est inventée par Carl Adam Petri en 1962 à l'université de Darmstadt. Parmi les applications des réseaux de Petri on peut citer: l'évaluation des performances des systèmes discrets, les protocoles de communications, la commandes des ateliers de fabrications, la conception des logiciels temps réels et/ou distribuées, les systèmes d'information (organisation d'entreprise).

Comme langage de modélisation graphique, les réseaux de Petri peuvent visualiser le comportement dynamique du système dont le réseau est composé par: (i) des noeuds de places représentés par des cercles qui correspondent aux activités des processus modélisés, à l'état d'un système ou à un événement, à une file d'attente, etc. (ii) des noeuds de transition représentés par des barres qui correspondent le plus souvent à des événements ou des conditions à vérifier. Elles peuvent cependant représenter des activités, en particulier si les places jouent le rôle d'événement. (iii) des arcs reliant les places avec des transitions. Ces derniers sont associés à des conditions de transitions ou des flux d'informations. La dynamique du système modélisé est représentée par la circulation d'un ou de plusieurs jetons entre les places. Le nombre de jetons étant défini par le poids de l'arc liant la transition à la place en sortie. Le passage de jetons d'une place à l'autre est soumis à la (aux) condition(s) ou l'événement (aux événements) représenté(s) par la transition. Dès que la condition est vérifiée, la transition est mise à feu et les jetons peuvent circuler.

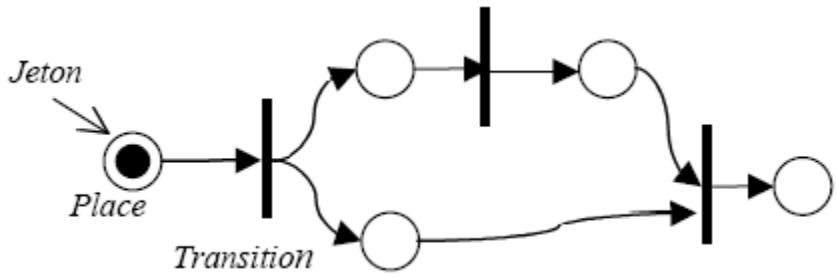


Figure 1.3 Réseaux de Petri

EPC (Event-Driven Process Chain)

EPC est un langage de modélisation intuitive utilisé pour la représentation des processus métiers. Le EPC a été développé par Keller, Nüttgens, et Scheer en 1992 [Mendlingy et al, 2007]. Il est utilisé largement dans les projets de modélisation d'entreprise. Un modèle EPC est un graphe directe et connecté composé de trois types de nœuds: (i) les fonctions qui correspond à des activités (task, process step) quelles besoins d'être exécutées. Une fonction est représentée par un rectangle à coins arrondis. (ii) Les événements décrits une situation avant et/ou après qu'une fonction est exécutée. Un événement correspond à des pré- ou à des post-conditions d'une fonction. Les événements sont représentés par des hexagones. (iii) Les connecteurs qui sont utilisés pour connecter les fonctions avec les événements. Dans un modèle EPC, on trouve trois types de connecteurs: $\wedge$ (AND), $\vee$ (OR), $\times$ (XOR). Les connecteurs sont représentés par des cercles dont le type du connecteur est indiqué aux centres. Les fonctions, les événements et les connecteurs peuvent être connectés par des flèches. Les événements possède au plus une flèche entrante et au plus une flèche sortante, et au moins une flèche associés (entrante ou sortante). Une fonction possède exactement une flèche entrante et une flèche sortante. Les connecteurs possèdent une flèche entrante et plusieurs flèches sortantes ou plusieurs flèches entrantes et une seule flèche sortante, et dans chaque chemin, les fonctions et les événements sont présentés alternativement (excepté les connecteurs intermédiaires).

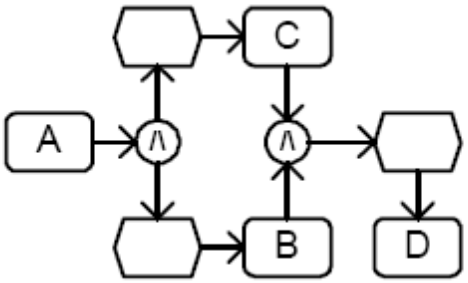


Figure 1.4 Un exemple de processus représenté en EPC

Diagramme d'activités de UML

UML (*Unified Modeling Language*) est un langage de modélisation orienté objets. Il doit être complété par une méthode pour obtenir un processus de génie logiciel complet. C'est un langage utilisable à la fois par les humains et les machines. UML est le résultat de l'unification de trois méthodes :

1. La méthode de Grady Booch, appelée parfois OOD,
2. OMT de James Rumbaugh,
3. OOSE et Objectory de Ivar Jacobson.

Jim Rumbaugh et Grady Booch décident fin 1994 d'unifier leurs travaux au sein d'une méthode unique : la méthode unifiée (The Unified Method). Une année plus tard, ils sont rejoints par Ivar Jacobson, le créateur des cas d'utilisation (use cases), une technique très efficace pour la détermination des besoins. UML permet de définir et de visualiser un modèle, à l'aide de diagrammes. Un diagramme UML est une représentation graphique, qui s'intéresse à un aspect précis du modèle. Chaque type de diagramme UML possède une structure et une sémantique précise. La Combinaison des différents types de diagrammes UML nous offre une vue complète des aspects statiques et dynamiques d'un système. On s'intéresse ici au diagramme d'activité qui peut être utile pendant la phase de modélisation des processus métiers.

Le diagramme d'activité permet de modéliser les dépendances entre les activités (séquentielles et/ou parallèles), en se déplaçant d'un point de départ initial vers le but désiré. Il est très similaire à un réseau de Petri. Chaque rectangle à coins ronds dans un diagramme d'activité représente une action; une flèche finie ouverte indique que l'action source doit être accomplie avant que l'action destinatrice soit commencé; un point noir indique le point de départ de l'activité; un point noir à l'intérieur d'un cercle indique la fin de l'activité; un diamant représente une décision; Des lignes noires épaisses, connues sous le nom de ***forks*** et ***joins***, sont employés pour indiquer le commencement et la fin d'un ensemble d'actions concourantes. Pour chaque action, nous pouvons montrer qui est le responsable de l'action en mettant un nom entre parenthèses, avant le nom de l'action lui-même. Ce nom peut être employé pour identifier des acteurs, des départements, des systèmes ou des objets. La figure suivante présente un exemple de diagramme d'activité.

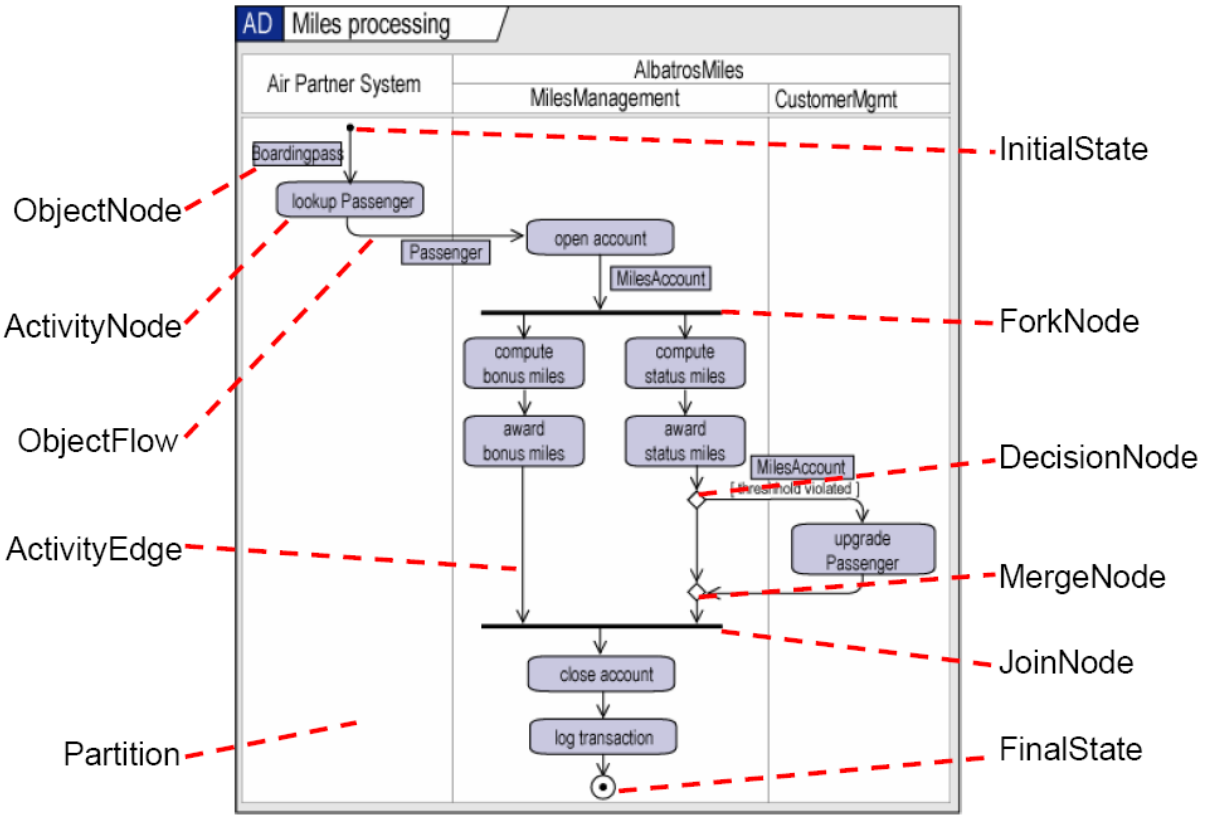


Figure 1.5 Exemple d'un diagramme d'activité

BPMN (Business Process Modeling Notation)

L'initiative BPMI (Business Process Management Initiative) a développé une notation standard de modélisation des processus métiers appelée BPMN (Business Process Modeling Notation). Les spécifications de BPMN 1.0 ont été publiées en mai, 2004. Le but initial de BPMI était de fournir une notation qui est aisément compréhensible par tous les utilisateurs métiers, par des analystes métiers qui créent les ébauches initiales des processus, par les développeurs techniques responsables de l'implémentation de la technologie qui exécutera ces processus, et finalement, par toute personnes d'affaires qui contrôleront et surveillent ces processus.

BPMN permette de fournir des modèles transformables facilement en un processus exécutable représentés par BPEL4WS (Business Process Execution Language for Web Services). BPMN définit un diagramme de processus métiers: BPD (Business Process Diagram), qui est basé sur une technique de schématisation utilisée pour créer les modèles graphiques des opérations de processus métiers. Un BPD est un réseau des objets graphiques, qui sont des activités et des objets de contrôle de flux qui définir leur ordre d'exécution.

Un BPD se compose d'un ensemble d'éléments graphiques. Ces éléments ont été choisis pour être distinctes aux autres et pour utiliser les formes les biens connus par la plupart des modélisateurs. Par exemple, les activités sont des rectangles et les décisions sont des diamants. Les quatre catégories de base sont :

- Flow Objects
- Connecting Objects
- Swimlanes
- Artifacts

Un BPD a trois objets de flux (Flow Objects) qui sont:

Event Un événement est représenté par un cercle et est quelque chose qui se produite pendant le déroulement d'un processus métiers. On distingue trois types d'événements: événement de début (start), événement intermédiaire (intermediate), événement de fin (end).

Activity Une activité est un terme générique dans le travail, elle est représentée par un rectangle à coins arrondis. Une activité peut être atomique ou non (composé). Les types d'activités sont : Tâche (Task) et Sous-processus (Sub-Process). Le sous-processus est distingué par un petit plus (+) au centre inférieur de la forme.

Gateway Un Gateway est représenté par la forme familière de diamant, il est employé pour commander la divergence et la convergence de l'ordre d'exécution des activités. Ainsi, il déterminera des décisions traditionnelles, aussi bien que diviser, fusionnant, et se joignant des chemins. Les marqueurs internes indiquer le type de la décision.

Les objets de connexion:

Les objets de flux sont reliés ensemble dans un diagramme pour créer la structure squelettique de base d'un processus métiers. Il y a trois objets de connexion qui fournissent cette fonction:

Sequence Flow Un Flux de séquence est représenté par une ligne solide avec une pointe de flèche pleine, il est employée pour montrer l'ordre dont lequel les activités sera exécuté dans un processus.

Message Flow Un flux de message est représenté par une ligne tirée avec une pointe de flèche ouverte, il est employé pour montrer l'écoulement des

messages envoyés entre deux participants de processus séparés (business entities ou business roles). Dans BPMN, deux contenants séparés (Pools) dans le diagramme représenteront les deux participants.

Association Une association est représentée par une ligne pointillée avec une pointe de flèche, elle est employée pour associer les données, les textes, et d'autres objets avec les objets de flux. Les associations sont employées pour montrer les entrées et les sorties des activités.

Les conteneurs (Swimlanes)

Beaucoup des méthodologies de modélisation de processus utilisent le concept des swimlanes comme mécanisme pour organiser les activités dans des catégories visuelles séparées afin d'illustrer différentes possibilités ou responsabilités fonctionnelles. BPMN porte deux types de swimlanes:

Pool Une *Pool* représente un participant à un processus. C'est également un conteneur graphique pour partitionner un ensemble d'activités, habituellement dans le contexte des B2B (Business to Business).

Lane Une *Lane* est une sous partition dans La *Pool*, elle prolongera la longueur entière de la *Pool* verticalement ou horizontalement. Les *Lane* sont employées pour organiser et classer les activités par catégorie.

Les Artifacts

BPMN fournit la flexibilité d'ajouter un contexte additionnel spécifique à toute situation de modélisation. BPMN supporte trois genres d'Artifacts: Un objet de données (Data Object), ce qui montre les données nécessaires et/ou produites par les activités. Un groupe qui est représenté par un rectangle à coins arrondis ayant des bords à tirets. Noter que le groupe est juste pour analyser le but et il n'affecte pas l'ordre de flux. Enfin nous avons les annotations qui sont des mécanismes utilisés pour fournir des informations textuelles additionnelles pour le lecteur d'un diagramme BPMN.

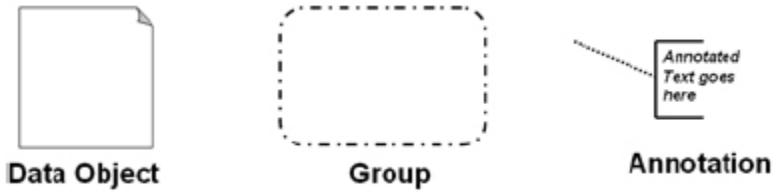


Figure 1.6 Les Artifacts de BPMN

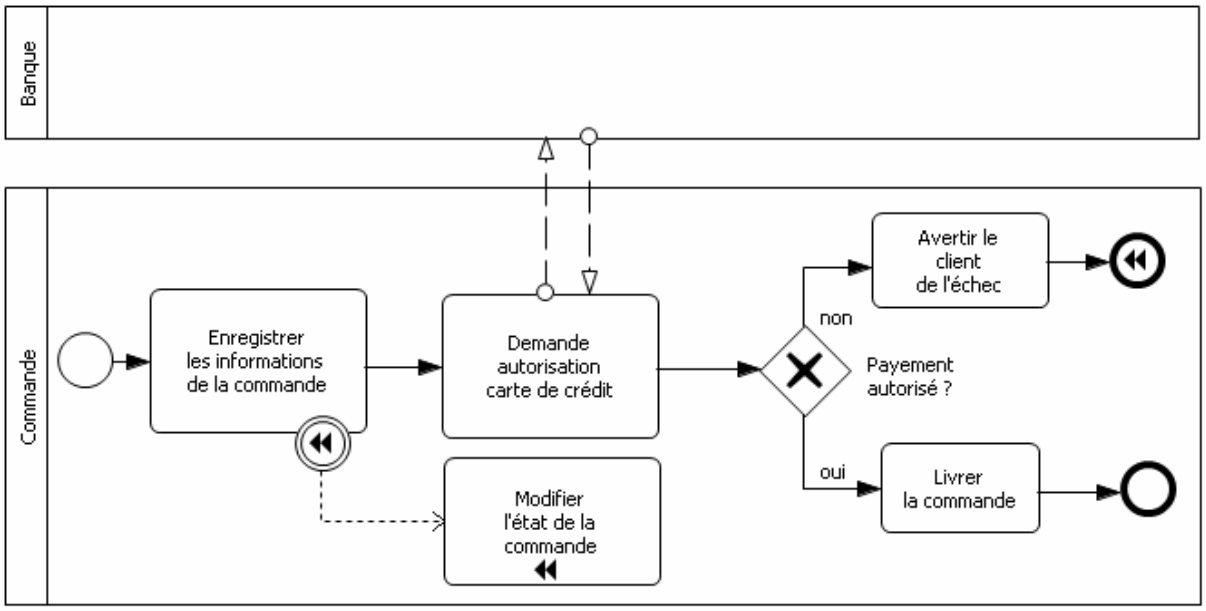


Figure 1.7 Exemple d'un BPD représenté avec BPMN

3. Ingénierie dirigée par les modèles et interopérabilité

L'interopérabilité des entreprises (et plus particulièrement l'interopérabilité des systèmes d'informations inter-organisationnels) est l'un des problèmes complexes auxquels l'entreprise est confrontée. Ce problème est crucial, car les systèmes d'informations d'entreprises qui sont par nature hétérogènes ont de plus en plus besoin de travailler ensemble. Dans la suite de ce chapitre, nous présentant la définition des concepts « collaboration », « interopérabilité » et « MDA ». Nous citons aussi les différents cadres d'interopérabilité d'entreprise ainsi que l'utilisation du cadre de modélisation MDA pour résoudre le problème d'interopérabilité.

3.1 Concepts de base

3.1.1 Collaboration

Une collaboration est une collection de participants qui travaillent ensemble pour réaliser un but commun. [Lemrabet, 2012] a distingué deux types de collaboration: (1) la *Collaboration Intégrée* qui signifie que les logiques métiers et applicatives sont intégrées dans un seul système monolithique utilisé par tous les participants ; (2) la *Collaboration Externe* qui a lieu entre deux ou plusieurs systèmes autonomes.

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

1. Dans la **vue centralisée de la coopération** un seul processus contient les activités de tous les participants. Dans cette vue, chacun des participants déploie et exécute le même processus collaboratif sur son propre Moteur d'Exécution de Processus (MEP). Puis, les MEP négocient entre eux pour déterminer qui exécutera l'activité suivante du processus collaboratif.

2. Dans la **vue distribuée de la coopération** les activités de chaque participant sont modélisées dans son propre processus. Dans ce cas, la collaboration entre les participants s'exprime par des échanges de messages entre leurs processus.

3.1.2 Interopérabilité

Il existe plusieurs définitions du mot Interopérabilité dans la littérature. [IEEE, 1990] considère l'interopérabilité comme « *la capacité que possèdent deux ou plusieurs systèmes ou composants à échanger des informations puis à exploiter les informations venant d'être échangées* ». C'est la définition de l'interopérabilité technique qui se focalise seulement sur l'échange d'informations entre des systèmes différents. Une autre définition de l'interopérabilité métier a été proposé par [ATHENA01, 2007] qui considère l'interopérabilité comme *la capacité opérationnelle et organisationnelle d'une entreprise pour coopérer avec ses partenaires métiers pour mieux créer, diriger et développer des relations basées sur le support de l'IT sans effort particulier des utilisateurs des systèmes d'informations d'entreprise.*

Plusieurs définitions de l'interopérabilité mais le but est le même : échanger et exploiter les informations. [Lewis et Wrage, 2004] ont distingué trois niveaux d'interopérabilité : le niveau syntaxique, le niveau sémantique et le niveau technologique :

- **L'interopérabilité syntaxique** concerne la capacité d'échange de l'information en proposant une intégration syntaxique de premier niveau en définissant notamment la nature, le type et le format des messages échangés,
- **L'interopérabilité sémantique** concerne l'interprétation commune du sens des informations et des fonctionnalités échangées et de la façon dont celle-ci devra être exploitée. Elle assure que les échanges qui s'effectuent entre les composants interconnectés conservent leur sens, c'est à dire que les parties communicantes ont une compréhension commune de la signification des données et des services qu'elles échangent. En effet, des conflits sémantiques surviennent puisque les systèmes n'utilisent pas la même interprétation de l'information qui est définie différemment d'une organisation à une autre,
- **L'interopérabilité technologique** qui concerne la coopération entre plusieurs entités logicielles issues de différentes technologies d'implémentation.

Trois grands axes de recherche marquent les travaux sur le domaine de l'interopérabilité :

- la modélisation de l'entreprise s'intéresse à la représentation de l'entreprise en réseau pour mettre en évidence les besoins en interopérabilité,
- les architectures et les plateformes définissent les solutions à implémenter pour atteindre l'interopérabilité,
- les ontologies qui offrent un vocabulaire partagé pour assurer l'interopérabilité sémantique.

L'interopérabilité peut être vue comme la capacité des entreprises à structurer, formaliser et partager le savoir-faire. Du point de vue application, l'interopérabilité vise à assurer la coopération entre deux applications sans un effort particulier d'interfaçage. Les applications initialement conçues pour fonctionner dans des environnements isolés, doivent pouvoir fonctionner ensemble.

[Bondé, 2006] a classé l'interopérabilité en trois catégories par référence aux différentes solutions et approches qui ont été proposées :

- L'interopérabilité dirigée par les cadres méthodologiques : dans cette catégorie d'approche, la résolution du problème d'interopérabilité passe par la définition d'une méthodologie de transformation de modèles afin de les implémenter sur des plateformes hétérogènes, la chose qui permettra d'aboutir à des systèmes interopérables. On pourrait citer, comme exemple de cette famille d'approches, la MDA (Model-Driven Architecture).
- L'interopérabilité dirigée par les modèles et les échanges de données : cette catégorie d'approche d'interopérabilité repose sur l'existence d'un langage commun permettant de définir les modèles ou données à échanger. Elle répond principalement à l'interopérabilité au niveau syntaxique. L'exemple le plus caractéristique de cette famille d'approches est l'interopérabilité par échange de modèles représentés en format XMI (XML Metadata Inter-change).
- L'interopérabilité centrée sur les services : cette catégorie contient toutes les approches d'interopérabilité centrées sur la notion de service. L'interopérabilité dans ces approches passe par la mise en œuvre de services connus dans l'environnement. Un exemple de cette catégorie d'approches est l'architecture orientée services (SOA).

Les approches d'interopérabilité basées sur les SOA se placent au niveau de l'interopérabilité syntaxique et sémantique. Interopérabilité syntaxique, à cause du format commun de codage des messages, et interopérabilité sémantique, par l'existence d'un contrat entre les consommateurs et fournisseurs de services.

3.2 Architecture Dirigée par les Modèles (MDA)

L'approche MDA (*Model Driven Architecture*) a était proposée par l'OMG (Object Management Group) en novembre 2000 [Soley, 2000]. L'approche MDA s'inscrit dans une tendance plus générale appelée Ingénierie Dirigée par les Modèles (MDE) qui pose les

modèles au centre du développement des logiciels. Les deux principaux artefacts de l'ingénierie dirigée par les modèles sont les modèles et les transformations de modèles.

Pour permettre aux organisations de faire évoluer leurs modèles d'applications indépendamment de l'évolution des plateformes technologiques, MDA préconise l'élaboration de quatre catégories de modèles en distinguant les modèles indépendants et spécifiques aux plateformes [Mukerji et Miller, 2003] :

- **Computation Independent Model – CIM :** appelé aussi modèle de domaine ou modèle métier, le CIM montre une vue qui se focalise sur l'environnement et les exigences du système.
- **Platform Independent Model – PIM :** modèle indépendant des plateformes techniques, le PIM représente une vue du système qui se focalise sur les entités fonctionnelles du système indépendamment des détails nécessaires à une plateforme particulière.
- **PDM** (*Platform Description Model*) ou **PM** (*Platform Model*) : il décrit la plateforme sur laquelle le système va être exécuté. Un PDM contient des informations pour la transformation de modèles vers une plateforme en particulier et il est spécifique de celle-ci. C'est un modèle de transformation qui va permettre le passage du PIM vers le PSM.
- **Platform Specific Model – PSM :** une vue du système qui se focalise sur les informations détaillées concernant l'utilisation d'une plateforme spécifique.

Quant à la transformation de modèles, elle consiste à utiliser des techniques de transformations pour mettre en relation les quatre principaux modèles CIM, PIM, PM et PSM. Ces techniques de transformation peuvent être utilisées pour générer le PIM à partir du CIM ou le PSM à partir du PIM (voir figure 1.8).

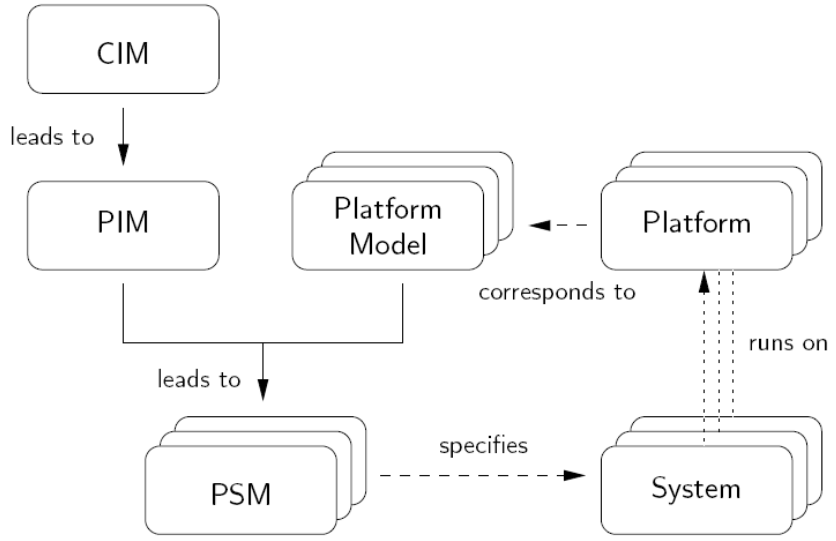


Figure 1.8 Relations entre les quatre catégories de modèles MDA

3.2.1 La hiérarchie des modèles MDA

Un méta-méta-modèle est un modèle qui décrit les éléments de modélisation nécessaires à la définition des langages de modélisation. Il a de plus la capacité de se décrire lui-même. C'est sur ce concept que se base l'organisation de la modélisation MDA. Les modèles MDA sont organisés sous une forme hiérarchique. Le monde réel est représenté à la base de la hiérarchie (niveau M0). Les modèles représentant cette réalité constituent le niveau M1. Les méta-modèles permettant la définition de ces modèles (p. ex. UML) constituent le niveauM2. Enfin, le méta-méta-modèle, unique et méta-circulaire, (MOF) est représenté au sommet de la hiérarchie (niveau M3). (Voir figure 1.9)

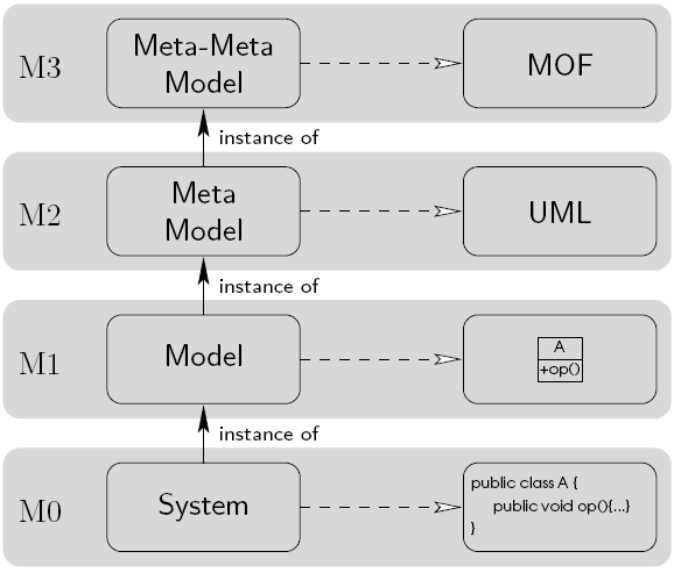


Figure 1.9 Hiérarchie de modélisation MDA

3.3 Interopérabilité dirigée par les modèles

Le but de l'interopérabilité dirigée par les modèles (*Model Driven Interoperability - MDI*) est d'adapter MDA pour résoudre les problèmes d'interopérabilité. L'approche MDI peut être utilisée par deux entreprises qui ont comme but d'améliorer leurs performances et qui veulent interopérer non seulement au niveau du code, mais aussi au niveau de la Modélisation d'Entreprise en utilisant les ontologies comme support [Bourey et al, 2007]. La Figure 1.10 représente le modèle de référence de l'approche MDI de [Bourey et al, 2007]. Elle montre les différents types de modèles qu'il est possible d'utiliser aux différents niveaux d'abstraction et les transformations de modèles qu'il faut réaliser. Dans le modèle de référence MDI la transformation d'un niveau plus haut à un niveau plus bas est définie comme une **transformation verticale**, alors que la transformation au même niveau entre deux entreprises est définie comme une **transformation horizontale**.

MDI définit 4 niveaux de modèles inspirés de la démarche MDA [Bourey et al, 2007] :

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

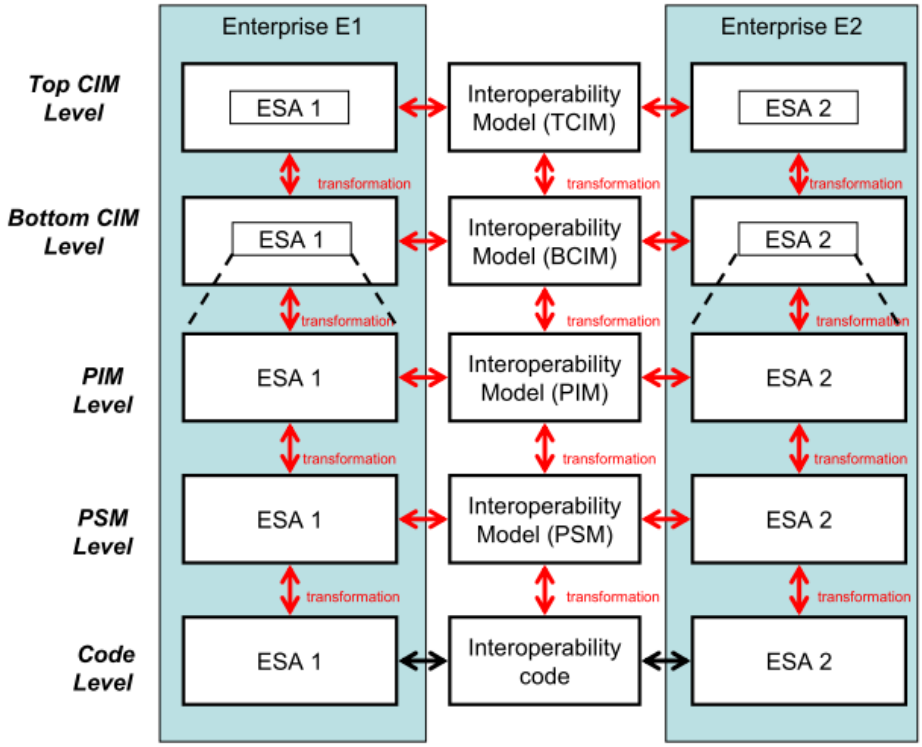


Figure 1.10 Modèle de référence MDI [Bourey et al, 2007]

Cette description pose les grandes lignes du modèle de référence de MDI. Cependant, elle n'est pas suffisante pour aborder concrètement un problème d'interopérabilité. Il faut affiner la description de chaque niveau et les modalités de passage d'un niveau à l'autre.

3.4 Cadres d'interopérabilité d'entreprise

Il y a eu plusieurs efforts dans différents contextes, pour caractériser le concept d'interopérabilité. La plupart de ces travaux proposent la mise en place de cadres spécifiquement dédiés à l'interopérabilité : (*Enterprise Interoperability Framework - EIF) [Chen*

et Daclin, 2006], (*ATHENA Interoperability Framework - AIF*) [ATHENA02, 2007], (*European Interoperability Framework - EIF*) [EIF, 2004], (*Interoperability Development for Entreprise Application and Software - IDEAS*) [IDEAS, 2003], etc.

Pour positionner notre travail d'identification et spécification des services réalisé dans le cadre de cette thèse, on présente le cadre d'interopérabilité *(Enterprise Interoperability Framework - EIF)*, développé dans le cadre du projet InterOp Noe (*Interoperability Research for Networked Entreprise Applications and software*). Ce cadre de référence distingue trois dimensions d'interopérabilité : les approches, les niveaux et les barrières.

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

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

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

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

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

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

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

- L'approche intégrée : les organisations partagent les mêmes modèles d'information. Cette approche consiste à construire un format commun pour tous les modèles afin de développer un système unique. Suite à l'action d'intégration, les deux systèmes en interaction ne forment plus qu'un seul système utilisant un modèle unique. Elle assure l'interopérabilité par le partage d'un environnement d'exécution et des conventions de communication.
- L'approche unifiée : les organisations partagent le même méta-modèle. Elle consiste à conserver le modèle propre à chaque système et de définir un format commun pour assurer la communication entre eux. Chaque système conserve alors sa propre structure avant et après la communication.
- L'approche fédérée : les organisations n'ont pas de modèle ou méta-modèle commun, mais utilisent une ontologie pour partager les données. Elle établie et maintient la collaboration entre des services locaux autonomes, chacun d'eux exécute un processus métier local.

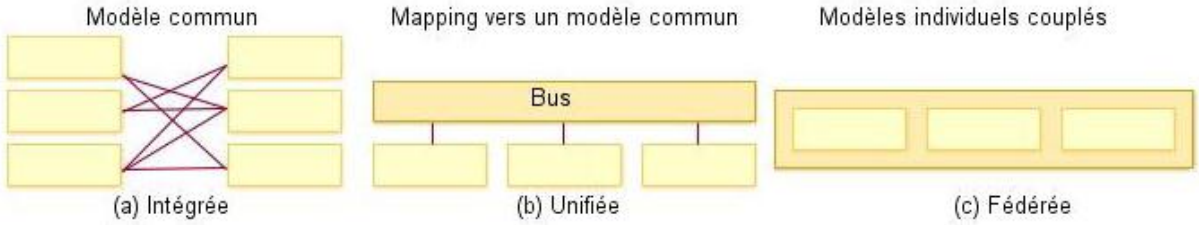


Figure 1.11 Les approches d'interopérabilité intégrée, unifiée et fédérée proposées par [Berre et al, 2004]

3.5 Architecture Orientée Service et Interopérabilité Dirigée par les Modèles

Pour tenter de répondre à la question de recherche suivante : Comment combiner les approches MDI et SOA dans un contexte de collaboration pour améliorer l'interopérabilité des systèmes d'informations d'entreprise ? On s'est basé sur les résultats du projet ATHENA, notamment le cadre interopérabilité AIF [ATHENA02, 2007]. En effet, pour développer l'interopérabilité d'entreprise le projet ATHENA recommande une approche multidisciplinaire qui réunit trois disciplines de recherche : (i) la modélisation d'entreprise qui définit les exigences d'interopérabilité et supporte l'implémentation de la solution, (ii) Les architectures et les plateformes qui fournissent la base technologique de l'interopérabilité des systèmes et (iii) l'ontologie qui identifie la sémantique de l'interopérabilité dans l'entreprise. Dans nos travaux de thèse on prend en considération les trois disciplines: modélisation des processus métiers d'entreprise, architecture orientée services et ontologie de processus métier. Ces trois éléments sont adoptés pour dériver automatiquement les modèles des services interopérables à partir d'un modèle de processus métier de haut niveau (voir chapitre 6).

La Figure 1.12 montre une vue simpliste du cadre conceptuel fourni par AIF. Ce modèle de référence indique les artefacts exigés et fournis par deux entreprises collaboratives. Chaque

entreprise est décrite par des modèles d'entreprise sur différents niveaux d'abstraction et différents points de vue (métier, processus, service et informations).

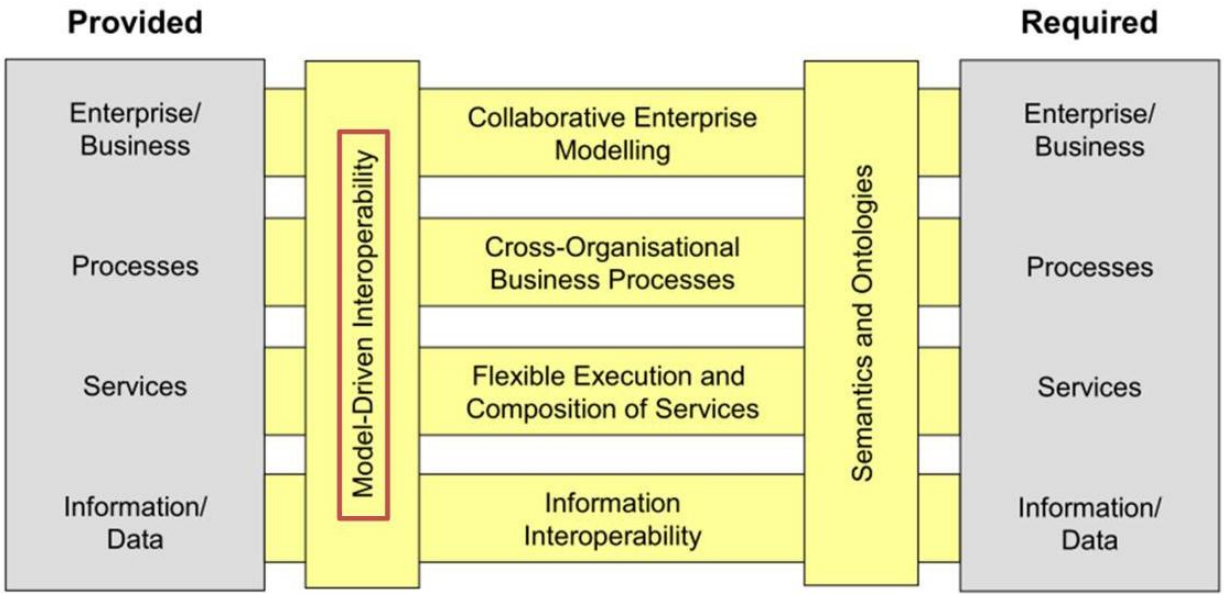


Figure 1.12 Vue simplifiée du Framework conceptuel défini par AIF [ATHENA02, 2007]

[Lemrabet , 2012] a combiner les approches AIF et MDI pour construire une grille pour capter les bonnes pratiques à chaque niveau de MDI (CIM-Haut, CIM-Bas, PIM, PSM) pour chaque aspect de l'AIF (métier, processus, service et donnée).

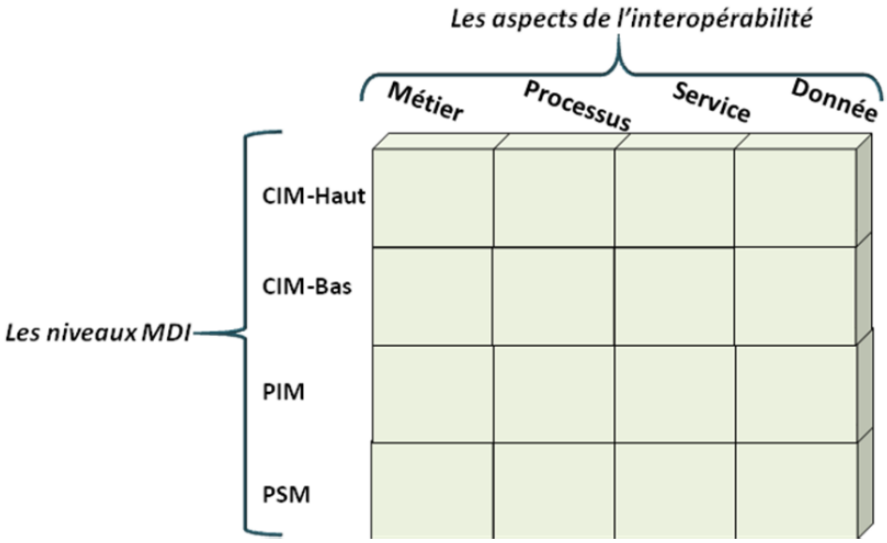


Figure 1.13 La grille pour identifier les différents aspects de l'interopérabilité [Lemrabet , 2012]

La grille représentée dans la Figure 1.13 définie l'interopérabilité comme un ensemble de seize sous-domaines d'interopérabilité. Ces sous-domaines facilitent la définition des domaines d'expertise entre les participants de la collaboration. Nous pouvons positionner notre contribution de thèse (chapitre 5 et chapitre 6) dans les sous-domaines (CIM-Haut et CIM-Bas d'un processus, et PIM d'un service) car elle se focalise sur la génération automatique des modèles PIM de service logiciels à partir des modèles CIM d'un processus

collaboratif. Nous expliquant dans les paragraphes suivants les quatre aspects d'interopérabilité présentés dans la figure 1.13

3.5.1 L'aspect Métier

L'interopérabilité à ce niveau est considérée comme la capacité d'une organisation de collaborer avec d'autres organisations externes qui ont des pratiques de travail, des législations, une culture ou des approches commerciales différentes [ATHENA03, 2005]. Les collaborateurs doivent partager une même vision et chacun d'eux doit définir ses propres objectifs en les formalisant avec un langage comme *(Business Motivation Model - BMM)* [OMG12, 2008]. Les partenaires doivent aussi choisir un style d'architecture commun à mettre en œuvre (ex : SOA).

3.5.2 L'aspect Processus

Les modèles de processus contiennent ce qu'il faut faire au niveau métier pour réaliser les objectifs métiers et la vision des collaborateurs. Dans un premier temps, il faut décrire globalement la collaboration en se concentrant sur les échanges entre les participants. Ensuite, il faut détailler la logique du flux des processus collaboratifs. Le formalisme BPMN 2.0 [OMG11, 2011] peut être utilisé pour spécifier les processus aux différents niveaux de l'approche MDI.

3.5.3 L'aspect Service

Le principal objectif de cet aspect est l'identification et la conception de services réutilisables pour supporter les processus métiers. L'identification des services peut être dirigée soit par les objectifs décrits dans l'aspect métier [Han et al, 2009] soit par les processus spécifiés dans l'aspect processus. Dans un premier temps les services doivent être modélisés avec un formalisme comme SoaML [OMG13, 2009] ou UML [OMG09, 2009 ; OMG10, 2009] qui offre un haut niveau d'abstraction, puis il faut raffiner les services pour les rendre spécifiques à une plateforme donnée.

3.5.4 L'aspect Donnée

Les modèles de données doivent être étudiés en parallèle avec les modèles des processus et des services. L'objectif de cet aspect est de définir les différents structures et types de données qui peuvent être échangé entre les différents processus et de les représenter en un format standard connu par tous les partenaires. BPMN ne précise pas comment spécifier les données utilisées par les processus.

4. Conclusion

La modélisation d'entreprise permet d'établir la représentation d'une partie ou l'ensemble d'entreprise selon un point de vue bien défini et avec un certain point de détail, afin de capitaliser la connaissance et le savoir-faire de l'entreprise, de rationaliser et structurer les échanges d'informations, d'analyser, concevoir et spécifier une partie de l'entreprise, et de simuler le comportement d'une partie de l'entreprise.

Dans ce premier chapitre, nous avons présenté une vue d'ensemble sur la combinaison des approches de modélisation d'entreprise, MDA et SOA pour supporter l'interopérabilité dans le cadre d'une collaboration. Pour cela, nous avons commencé par un rappel sur la

modélisation d'entreprise. Puis nous avons expliqué les concepts *collaboration* et *interopérabilité*. Puis, nous avons présenté le cadre de référence de l'interopérabilité dirigée par les modèles ainsi que les différents cadres d'interopérabilité existants. Enfin, nous avons positionné notre contribution dans une grille d'interopérabilité proposée par [Lemrabet, 2012]. Cette grille est formée de différents sous-domaines d'interopérabilité qu'il faut considérer pour permettre aux partenaires d'une collaboration d'analyser et de comprendre leurs besoins métiers et leurs exigences techniques.

L'intérêt de ce chapitre est de poser la base sur laquelle nous avons développé notre méthode d'identification et de spécification des modèles de services logiciels à partir d'un modèle de processus métier de haut niveau.

La modélisation des processus métier de l'entreprise peut être utilisée comme un point de départ à la conception de l'architecture du système d'information. Nous étudions, dans le chapitre suivant, la modélisation de l'architecture orientée services qui est considérée comme une architecture plus adaptée au besoin d'interopérabilité.

Chapitre 2

Modélisation des architectures orientées services

Chapitre 2 Modélisation des architectures orientées services

1. Introduction

Aux débuts du développement de logiciels, le code a été souvent écrit dans un seul bloc cohérent. Pour faciliter la résolution des problèmes, il est plus efficace de décomposer un grand problème en plusieurs sous-problèmes plus petit. Dans le développement de logiciel, ceci a été fait par l'utilisation des sous-routines et plus tard par l'introduction de la Programmation Orientée Objet (POO) qui coupe le code en plusieurs objets plus petits qui communiquent entre eux. Aujourd'hui, la POO est la base du développement des applications modernes.

D'autre part, les techniques modernes essayent d'éviter la réécriture du même code plusieurs fois qui signifie que le code devrait être autant réutilisable que possible. Le code réutilisable peut plus tard être employé pour implémenter de nouveaux systèmes. C'est l'une des idées du Développement Basé sur les Composants (DBC). Le DBC essaye de couper le logiciel en composants autonomes qui sont légèrement connectés et peuvent agir l'un sur l'autre à travers des interfaces. Chaque composant pourrait plus tard être employé et intégré dans d'autres logiciels.

Après le DBC, la prochaine étape dans l'évolution est l'Architecture Orientée Service (SOA). L'architecture orientée services est très semblable au développement basé sur les composants. Elle définit des services légèrement connectés qui agit l'un sur l'autre à travers des interfaces bien définies. Mais SOA ajoute plusieurs détails qui font la différence avec les autres paradigmes de développement logiciel. Les services SOA ont une granularité plus forte que les composants et sont alignés sur des processus métier. Une autre différence c'est l'auto-description des services, la chose qui permet aux applications de découvrir automatiquement les services qui répondent bien à leurs besoins. L'application consommatrice de services peut composer automatiquement des petits services afin d'accomplir ces buts métiers, et cela grasse au concept d'auto-description.

SOA permet d'augmenter la flexibilité (nous pouvons reconstruire facilement notre application par le remplacement des services par d'autres services plus adaptés aux nouveau besoins), améliorée la productivité, gagner le temps et réduire le cout de développement par la réutilisation des services prédéveloppés, et amélioré la maintenance par la substitution automatique des services mal-fonctionnés par d'autres service opérationnels.

2. Architecture orientée service (SOA)

Un certain nombre de définitions similaires et parfois complémentaires de SOA peuvent être trouvées dans la littérature. Parmi les définitions les plus appropriées de SOA, nous citons :

1. Un paradigme pour organiser et utiliser les capacités distribuées qui peuvent être sous la commande de différents domaines propriétaires. Elle fournit des moyens

standardisés pour offrir, découvrir, interagir avec et d'utiliser les capacités pour produire des effets désirés compatibles aux pré-conditions et aux espérances mesurables [MacKenzie et al, 2006].

2. Une architecture de logiciel qui emploie des services logiciels légèrement connectés pour soutenir les exigences des processus métier et des utilisateurs des logiciels " [Wikipedia, 2005]
3. Les politiques, pratiques, les cadres qui permettent à la fonctionnalité d'application d'être fournie et consommée en tant qu'ensembles de services publiés en une granularité correspondante aux besoins du consommateur de service. Les services peuvent être appelés, publiés et découverts à travers d'une forme simple d'interface standardisée [Sprott et Wilkes, 2004]
4. Approche de technologie de l'information ou stratégies dans lesquelles les applications utilisent des services disponibles dans un réseau tel que le World Wide Web " [Ort, 2005].

À partir de ces définitions nous pouvons extraire les principales caractéristiques de SOA telles que réutilisabilité, interopérabilité, abstraction, couplage faible, etc.

1. Encapsulation de service : les services encapsulent un morceau donné de fonctionnalité ou, plus généralement, une valeur.

2. Couplage faible : les services maintiennent seulement les relations qui réduisent les dépendances interservices et qui augmentent les dépendances intra-services.

3. Contrat de service : les services adhèrent à un agrément de communications, comme il est définit collectivement par un ou plusieurs documents de description de service.

4. Abstraction de service : au-delà de ce qui est décrit dans le contrat de service, les services cachent la logique métier du monde extérieur.

5. La réutilisabilité de service : la logique est divisée en petits services avec l'intention de favoriser la réutilisation.

6. Composability de service : des collections de services peuvent être coordonnées et assemblées pour former des services composés.

7. Autonomie de service : les services ont le contrôle de la logique qu'ils s'encapsulent.

8. Discoverability de service : des services sont conçus pour être extérieurement descriptifs de sorte qu'ils puissent être trouvés et accédés par l'intermédiaire des mécanismes de découverte.

2.1 Services Web

SOA, qui est aussi connu comme le calcul orienté service, est un paradigme architectural indépendant de la technologie. Cependant, il existe des langues, des technologies et des infrastructures qui peuvent être employées pour mettre en application une architecture orientée services.

Les Services Web sont considérés comme des technologies populaires utilisées pour réaliser une SOA. Ils exposent explicitement les fonctionnalités métiers en une manière interopérable.

Plusieurs définitions de service de Web sont trouvées dans la littérature. Nous citons dans la suite quatre définitions :

1. Un système logiciel conçu pour soutenir l'interaction interopérable de machine-à-machine sur un réseau. Il a une interface décrite dans un format compréhensible par machines (spécifiquement WSDL). D'autres systèmes interagissent avec le service Web en une manière prescrit par sa description à travers des messages SOAP exprimés en XML et transportés à travers le protocole HTTP et d'autres standards liées au WEB " [Haas et Brown, 2004].
2. Composants de logiciel légèrement connectés et réutilisables qui encapsulent sémantiquement la fonctionnalité discrète et sont distribués et accessible par des programmes au-dessus des protocoles standard d'Internet [Dormeur, 2001]
3. Applications d'un seul bloc, auto-descriptive, modulaires qui peuvent être publiées, localisés et appelé à travers le Web. Les services Web remplissent des fonctions, qui peuvent être quelque chose de simples requêtes aux processus métier compliqués [Tidwell, 2000].
4. Applications (programmes) auto descriptives, modulaires, indépendantes et faiblement couplées qui fournissent un modèle simple de programmation et de déploiement d'applications, basé sur des normes s'exécutant à travers l'infrastructure web, et qui peuvent être découvertes et invoquées dynamiquement via Internet ou un intranet par d'autres services [Daniel, 2003].

Grâce aux services Web, les entreprises peuvent encapsuler leurs activités métiers et les publier comme des services, chercher d'autres services et échanger des informations au-delà des frontières des entreprises. La figure 2.1 montre l'architecture de référence SOA.

2.1.1 SOAP, WSDL and UDDI

Les services Web encapsulent les fonctionnalités et permettent l'accès à n'importe quel composant d'une manière interopérable. À cette fin, les services Web utilisent trois spécifications principales qui permettent la description des services Web, l'échange des messages avec d'autres services Web, et localisation des services Web, à savoir : WSDL, SOAP et UDDI. Ces spécifications sont brièvement présentées dans la section suivante.

SOAP The Simple Object Access Protocol (SOAP) [W3C, 2003]. Est un protocole qui définit une manière uniforme de circuler des données codées en XML. Il définit également une manière d'employer le protocole HTTP comme protocole de transmission fondamental pour passer des messages entre deux points finaux. SOAP est un protocole indépendant à la plateforme, simple et extensible. Il fournit une manière de communication entre les applications fonctionnant sur différents systèmes d'exploitation avec différentes technologies et langages de programmation. Ces dispositifs montre la

puissance de SOAP pour l'échange des messages dans les environnements distribués et hétérogènes.

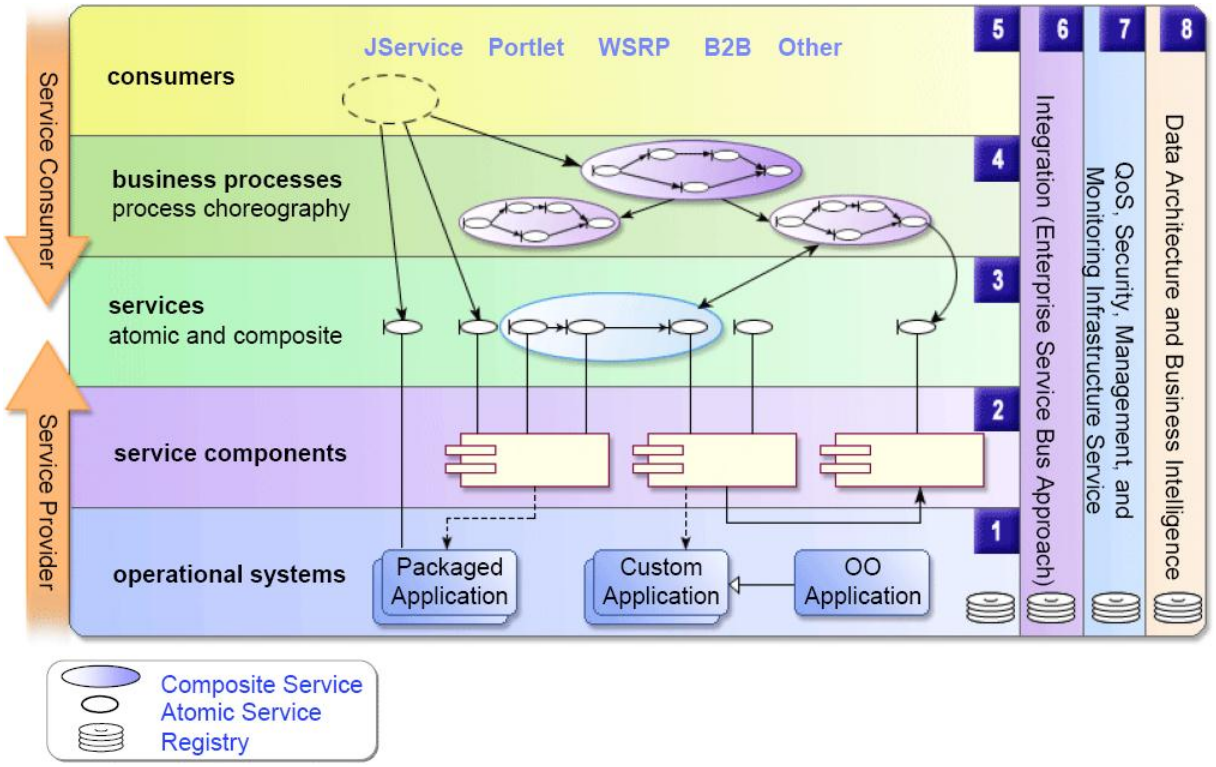


Figure 2.1 Architecture de référence SOA [Hoidn, 2012]

WSDL fournit un modèle et un format XML pour décrire des Services Web [Christensen et al., 2001]. WSDL permet de séparer la description de la fonctionnalité abstraite d'un service, des détails concrets d'une description de service, c.-à-d. la séparation de "quelle" fonctionnalité est fournie de "comment" et "où" celle-ci est offerte (?). Un document WSDL est composé essentiellement de définitions. Chaque définition est composée d'interfaces, de messages, de liaisons (bindings) et de services. Les types, les messages et les interfaces constituent la partie abstraite d'un document WSDL, les liaisons (bindings) et les services en constituent la partie concrète.

UDDI Universal Description, Discovery and Integration [Bellwood et al., 2002] fournit la définition d'un ensemble de services qui permettent la description et la découverte :

1. des entreprises, des organismes, et d'autres fournisseurs de Services Web,
2. des Services Web qu'ils proposent,
3. des interfaces techniques qui peuvent être utilisées pour accéder à ces services.

UDDI est similaire à un annuaire téléphonique qui présente donc des structures similaires aux pages blanches (i.e. Business Entity), pages jaunes (i.e. Business Service) et pages vertes (i.e. Binding Template).

Les pages blanches sont utilisées pour trouver un service par le contact, nom et adresse du fournisseur. Les pages jaunes sont utilisées pour trouver un service par une taxonomie

standardisée. Les pages vertes sont utilisées pour trouver un service par les caractéristiques techniques demandées. La figure 2.2 montre la relation entre WSDL, SOAP, et UDDI.

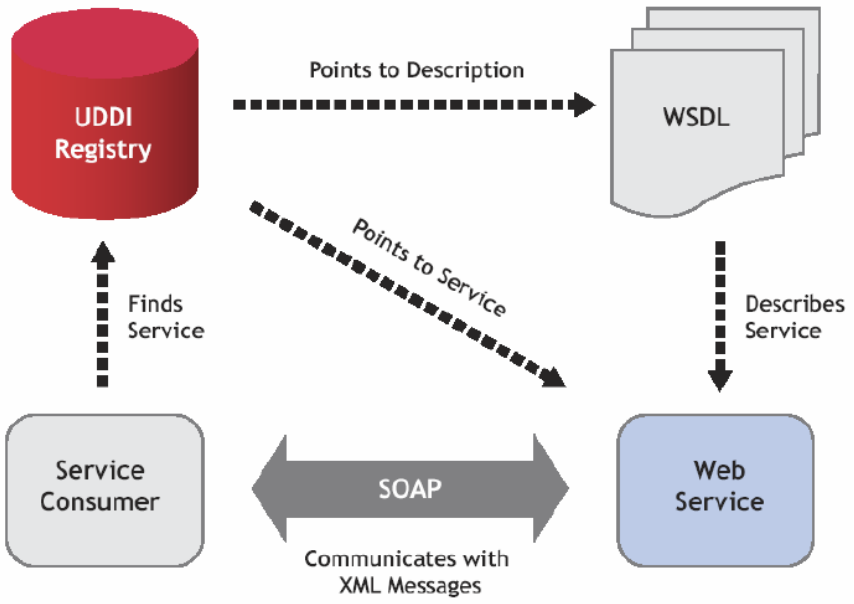


Figure 2.2 WSDL, SOAP, et UDDI [Hernández, 2007]

2.1.2 Fournisseur, consommateur, et annuaire de services

Un système à base SOA se compose de services légèrement connectés. Pour fournir, contrôler et employer des services, SOA utilise une topologie fournisseur/consommateur/annuaire (figure 2.3).

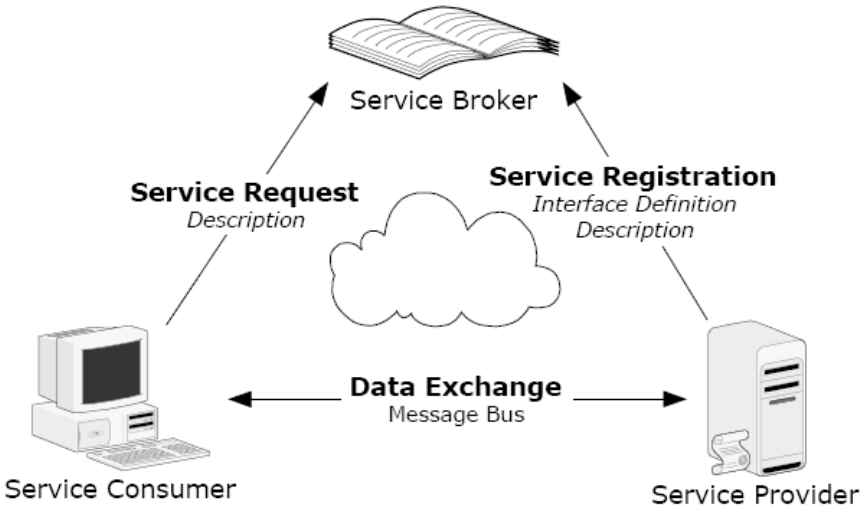


Figure 2.3 les trois nœuds SOA

Dans SOA, un fournisseur de services implémente un service. C'est une structure composée qui encapsule les composants implémentant le service et réalisant les spécifications de service. Le fournisseur de services publié une description de service dans un annuaire de service. La description de service est une métadonnée, qui décrit la fonction du service et définis les attributs de la qualité de service. L'annuaire de service contrôle tous les services existants, gère les requêtes des consommateurs de service et

fournit des connexions vers les fournisseurs de service. Un consommateur de service peut chercher des services dans l'annuaire, sélectionne les services les plus pertinents, puis établir une connexion au fournisseur de services.

La mise en œuvre d'une architecture orientée service n'est pas une tâche facile. Elle nécessite une méthodologie complète qui incluse une phase d'analyse des processus métiers de l'entreprise et identification des services candidats, une deuxième phase de modélisation et de spécification de la structure et du comportement des services identifiés dans la phase précédente, une troisième phase de réalisation et implémentation des services spécifiés. Nous présentons dans les sections suivantes les principes de modélisation d'une architecture orientée service.

3. Modélisation Orientée Services

En principe, le processus MDA débute avec la définition des modèles CIM ou de domaine, alors ces modèles peuvent être transformés par des professionnels en modèles PIM et PSM. Ce genre de transformation s'appelle transformation verticale. Une transformation verticale directe n'est pas toujours possible parce qu'il y a une grande distance entre les modèles qui sont trop grands pour les transformer en une simple étape. Dans ce cas, des transformations additionnelles sont nécessaires, par exemple, transformer un certain modèles abstraits PIM en des PIM plus détaillé.

Le niveau CIM décrit les modèles métiers qui incluent des buts, des règles métiers, des processus métiers et des services métiers. Ceci a pu être fait avec Business Motivation Model (BMM) et Business Process Modeling Notation (BPMN).

Le niveau PIM décrit les modèles SOA qui sont indépendant de la plateforme d'exécution /de la technologie. Il inclut des modèles d'interfaces de service, des contrats de service, des participants, etc. Les modèles à ce niveau d'abstraction sont considérés comme des modèles de spécifications de logiciel.

Le niveau PSM décrit des modèles spécifiques à la plateforme en tant que composants exécutables. Ces modèles peuvent être considérés comme des modèles de réalisation de logiciel. Si l'application SOA est implémentée avec, par exemple, des services Web, les interfaces de service peuvent être représentées en utilisant Web Service Description Language (WSDL). Un autre exemple est Business Process Execution Language (BPEL) pour spécifier le comportement de processus métier basé sur des services Web.

3.1 Méta-modèle SOA

Pour MDE, le méta-modèle est un concept très important. Chaque modèle exprimé en un certain langage de modélisation se conforme uniquement au méta-modèle de ce langage. Un méta-modèle peut être considéré comme une description explicite de la façon dont un modèle est établi. Afin de s'appliquer MDE (et en particulier MDA) au développement des applications SOA, il est important d'avoir un méta-modèle concret pour l'application SOA à modéliser. Si un tel méta-modèle n'est pas disponible il doit être défini.

Premièrement, un mécanisme est nécessaire pour définir clairement le langage de modélisation, de sorte qu'un outil de transformation puisse alors lire, écrire, et comprendre les modèles. Deuxièmement, les règles de transformation qui définissent une transformation décrivent comment un modèle dans un langage source peut être transformé en un modèle dans un langage cible. Ces règles emploient les méta-modèles des langages sources et cibles pour définir la transformation.

3.2 Critères de modélisation SOA

[Niels Schot, 2012] a organisé les critères de modélisation SOA selon les dimensions suivantes :

- **Syntaxe / Sémantique**

Les dimensions syntaxe et sémantique doivent être considérés dans n'importe quel langage de modélisation. Tous les langages ont besoin d'une syntaxe qui définit les combinaisons des symboles qui sont utilisés pour définir correctement la structure des modèles. Les langages ont également une sémantique, qui se rapporte à la signification des symboles disposés avec cette structure.

- **Abstrait / Concret**

Le niveau d'abstraction dans lequel les concepts SOA sont modélisés doit être considéré durant le processus de modélisation. Les concepts SOA peuvent être modélisés dans un niveau abstrait, ou dans un niveau concret avec plus de détail. Par exemple, un consommateur de service pourrait avoir besoin d'un modèle de service avec beaucoup de détails pour consommer le service, qui inclut les spécifications des opérations disponibles et du protocole d'interaction. Cependant, dans certains cas, les vues abstraites des services peuvent être suffisantes, par exemple, pour montrer comment les participants travaillent ensemble dans une application SOA.

- **Fonctionnel /Non Fonctionnel**

Les critères peuvent se rapporter aux concepts fonctionnels qui sont employés pour définir les opérations d'une application SOA. En outre, d'autres critères se rapportent aux aspects non fonctionnels de SOAs, parce que la modélisation des aspects non fonctionnels est souvent nécessaire pour décrire les conditions sur la qualité de service comme, par exemple, la disponibilité, la performance et la sécurité.

- **Comportement / Structure**

Nous pouvons classer les modèles SOA selon la structure ou le comportement. Des modèles sont employés pour représenter des structures (structures de données, composants) ou des comportements (fonctionnement interne du service).

- **Points de vue d'Acteurs**

La quatrième dimension est le point de vue des acteurs. Nous pouvons regarder des points de vue des différents acteurs qui emploient un service durant son cycle de vie.

3.2.1 Concepts fonctionnels de la SOA

a. Service

Le concept principal d'une SOA est le service. Un service est consommé par une application ou un système donné, et il représente une partie du comportement interne de l'application ou du système implémenté à l'extérieure du système consommateur. Un service devrait avoir des « capability » pour effectuer le travail, des spécifications du travail offert, et les offres fournis après l'exécution du service. Finalement les services doivent être modélisés syntaxiquement et sémantiquement dans différents niveaux d'abstraction [Niels Schot, 2012].

Les vues structurales et comportementales sont nécessaires pour modéliser toute les informations nécessaires, comme il est expliqué ci-dessous.

1. Que ce soit pour un consommateur ou un fournisseur de service, il est important de connaitre communiquer l'un avec l'autre, c.-à-d. comment un consommateur peut utiliser des services fournis par un fournisseur. Ceci, par exemple, exige des spécifications des interfaces de service, des messages, et du protocole d'interaction nécessaire.

2. Pour le consommateur de service, il est important de connaitre les conditions et les effets d'utiliser le service. Les Pré- et les post-conditions des services doivent être définis aussi.

b. Interface et implémentation de service

Un développeur de service doit spécifier les interfaces de service que le fournisseur offres. Les spécifications complètes d'interface de service devraient décrire toute information nécessaire pour qu'un consommateur emploie le service.

Pour la séparation des préoccupations, l'interface est indépendante mais elle est compatible à la façon dont le développeur implémente le service et à la façon dont le consommateur le consomme. Ainsi nous pouvons distinguer L'extérieur (interface) et l'intérieur (implémentation) d'un service. Pour les consommateurs éventuels, il est important de savoir comment ils peuvent employer le service en termes de format de message (structure) et ordre des messages et des opérations (comportement). Ces consommateurs n'ont pas besoin d'informations sur l'intérieur du service et peuvent employer seulement les modèles abstraits qui concerne la structure interne de service.

D'un point de vue comportemental nous pouvons sembler s'il est possible que le concepteur de service modélise l'implémentation interne de service d'une manière indépendante à la plateforme. Un concepteur de service peut modéliser l'implémentation de service dans le niveau PIM pour profiter les avantages de MDE. Par exemple, la modélisation du comportement de service dans un niveau PIM renforce la compréhension pour d'autres acteurs, tels que programmeur de service, et elle facilite les transformations possibles des modèles PIM vers les Modèles PSM [Niels Schot, 2012].

c. Description et découverte des services

Le fournisseur, le consommateur et l'annuaire de service sont impliqués dans la découverte de service. La réalisation de la découverte de service par l'intermédiaire d'un annuaire de service a besoin des descriptions de service et d'un registre de service. La description de service est employée par un fournisseur de services pour publier le service. L'information supplémentaire du registre devrait fournir des informations sur l'interface, les effets, l'adresse, et les conditions additionnelles possibles du service. Le consommateur emploie ces informations, probablement trouvées dans le registre, pour employer le service.

d. Contrat de service et interaction

Parfois des spécifications plus avancées sont nécessaires pour modéliser un comportement bidirectionnel plus complexe qu'un simple scénario demande/réponse, par exemple, en termes de contrat de service. Ces contrats de service peuvent se rapporter à des spécifications de l'interaction nécessaire entre les participants. Un tel contrat peut aider à spécifier comment les participants travaillent ensemble. Quand les consommateurs admettent ces contrats, ils acceptent d'employer le service comme il est défini dans le contrat.

Le comportement de contrat de service devrait être modélisé par le développeur, ceci est souvent fait sous forme d'une chorégraphie. La chorégraphie sert à établir un accord entre des multiples acteurs en termes d'échanges de message. Elle leur permet de comprendre comment leurs systèmes devraient agir l'un sur l'autre. Avec une chorégraphie il est possible de définir quelle information est envoyée entre les fournisseurs et les consommateurs et dans quel ordre. Une chorégraphie spécifie l'interaction entre les participants en détail [Niels Schot, 2012]. La figure 2.4 montre un exemple d'une Chorégraphie.

e. Composition de service

La composition de service est un concept important utilisé par le développeur et le consommateur de service. SOA stimule la réutilisation des services disponibles comme composants d'un service plus grand appelé service composé. La composition de service est identifiée en tant que critère supplémentaire pour le consommateur de service. Nous considérons deux points de vue en ce qui concerne la composition de service. (i) Le point de vue du développeur de service, la composition de service est réalisé au temps de conception pour construire un nouveau service à partir des services existants. De cette façon le développeur de service implémente un nouveau service en se basant sur d'autres services. (ii) Le point de vue du consommateur de service, la composition dynamique de service peut être employée pour laisser plusieurs services fonctionnent ensemble (au temps d'exécution) pour accomplir une certaine tâche dans une application SOA. Dans ce cas-ci, le consommateur joue le rôle d'un concepteur de composition.

f. Orchestration

Un concept utilisé pour décrire une composition est l'orchestration. De la même manière d'une chorégraphie, l'orchestration décrit le comportement de service, mais elle est diffère de la chorégraphie parce qu'elle décrit un flux de processus entre les services, contrôlé par

une partie unique. Par conséquent, les deux concepts sont connexes pour se rapporter à la description des collaborations de service, mais différemment. Une orchestration de service garantit que les services dans une application SOA peuvent être coordonnés pour travailler ensemble pour accomplir les buts métiers. Une orchestration est une spécification détaillée et probablement exécutables d'un service composé. Dans une chorégraphie, nous regardons le processus public entre des parties multiples, alors qu'avec une orchestration nous décrivons le processus d'interaction privé d'une simple partie. La figure 2.5 montre un exemple d'une Orchestration.

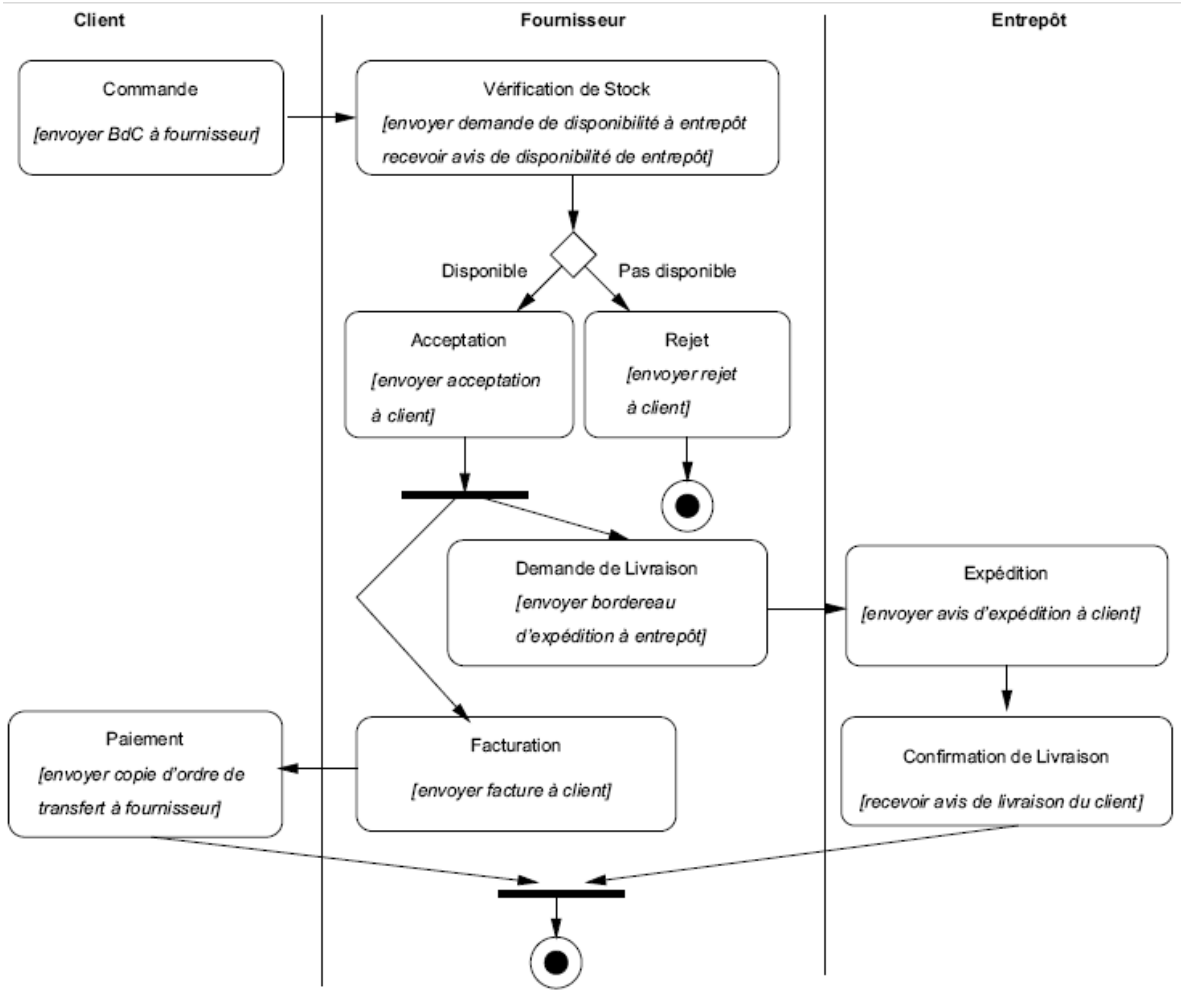


Figure 2.4 Exemple d'une Chorégraphie

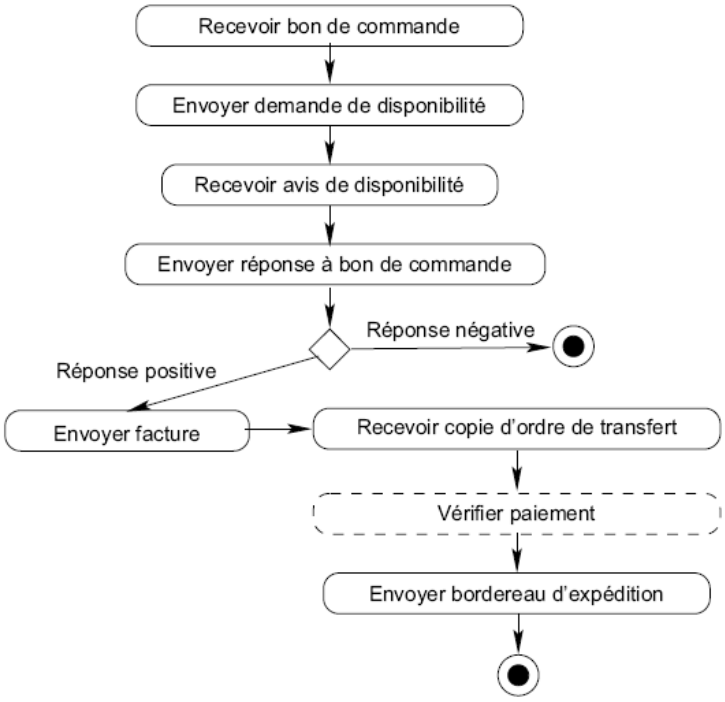


Figure 2.5 Exemple d'une Orchestration

g. Application SOA

Quelques acteurs de service sont responsables pour modéliser une application SOA qui emploie des services. La description de la collaboration des services utilisés est nécessaire, mais un modèle complet pour comprendre l'application SOA globale pourrait également être nécessaire dans des applications SOA plus avancées quand multiple services, fournisseurs et consommateurs sont impliqués. Un tel modèle devrait donner une vue d'ensemble de l'architecture décrivant comment les participants (avec leurs rôles possibles comme fournisseur ou consommateur) travaillent ensemble dans un niveau plus abstrait.

3.2.2 Concepts non-fonctionnels du SOA

Les besoins non fonctionnels spécifient les critères qui peuvent être employés pour juger le bon fonctionnement d'un système, plutôt que des comportements spécifiques. Des exemples sont les conditions de qualité qui décrivent les qualités voulues du produit qui ne sont pas directement liées à la fonctionnalité. Dans la modélisation des applications orientées services, le concept qualité de service (QoS) est souvent utilisé pour se rapporter à la collection de contraintes et de conditions de qualité pour un service. Il est important que QoS puisse être modélisée parce que ces contraintes et conditions de qualité peuvent alors être reconnues par les différents utilisateurs de services. Nous nous focalisons, dans le présent travail, seulement sur la modélisation des concepts fonctionnels du SOA.

3.3 Modèle conceptuel SOA

Une étape remarquable vers l'établissement d'un consensus sur la définition des éléments de modélisation d'une SOA a été la définition d'un modèle de référence pour l'architecture orientée services par OASIS. Ce modèle de référence est un cadre abstrait qui se compose d'un ensemble minimal de concepts unifiés, axiomes et relations utilisés pour modéliser les services indépendamment des normes spécifiques, technologies, réalisations, ou

d'autres détails d'implémentation, réduisant de ce fait la confusion créée par la prolifération des terminologies utilisées dans le domaine des SOA. Nous citons dans la suite la définition d'un sous ensemble de concepts utilisés pour modéliser les services SOA.

Les principaux concepts présentés par le modèle de référence d'OASIS sont : service, effet, capacité, et interface de service. Nous nous concentrerons dans ce chapitre sur ces concepts qui sont particulièrement appropriés au processus de d'identification et de spécification des services.

Définition 2.1 (Capability) :
(Capability of service) est la capacité d'effectuer quelque actions ou actions avec une valeur perçue, dans le sens qu'elle peut constituer la solution (peut-être partiel) d'un certain problème.

Les capacités ont associé certains effets, que nous définissons ci-dessous ; le but d'employer des capacités est de réaliser l'ensemble ou une partie de tels effets.

Définition 2.2 (Effect) :
Un effet est le résultat d'employer des capacités, et il peut être d'un des deux types suivants :

1. Une information évidente est rendue à la partie utilisatrice de la capacité de service, ou

2. Il y a un changement à un certain état partagé par le fournisseur de la capacité et le consommateur de la capacité.

Définition 2.3 (Service) :
Un service est un mécanisme par lequel certaines capacités sont accédées et ses effets sont réalisés.

Chaque service est fourni par une partie particulière que nous appelons le fournisseur de services, et il est employé par un consommateur de service. Il doit noter que les fournisseurs de service peuvent également agir en tant que consommateurs de service et vice versa.

Les fournisseurs d'un service exigent souvent de certaines informations fournies par le consommateur et certaines conditions à tenir sur un certain état partagé pour permettre l'accès aux capacités fondamentales associées à un service particulier. Nous appelons ces conditions « pré-conditions ».

Définition 2.4 (Precondition) :
Une pré-condition est une condition imposée par un service donné pour permettre réellement l'accès à ses capacités fondamentales. Elles peuvent être un des deux types suivants :

1. Quelques informations sont exigées pour être fournies par le consommateur du service, ou

2. Quelques conditions doivent se tenir sur un certain état partagé par le consommateur de service et le fournisseur de services.

Nous appelons le premier type de pré-condition « pré-condition d'information », et le deuxième « pré-condition du monde réel ».

Un service peut avoir zéro pré-conditions ou plus, et il n'y a aucune limite supérieure pour le nombre de pré-conditions nécessaire à l'exécution d'un service.

Pour qu'un service soit exécuté, des interactions entre le consommateur de service et le fournisseur de services sont nécessaires. Ces interactions peuvent seulement être effectuées de quelques manières explicitement prescrites par le fournisseur de services. L'interface de service définit comment l'interaction avec le service doit s'effectuée.

Définition 2.5 (Interface de service) :
L'interface de service est un moyen d'interagir avec un service. Elle définit les protocoles, les commandes, et l'échange des informations spécifiques par lequel un service est exécuté. L'information exigée par le service est représentée par des entrées d'interface, et l'information fournie par le service est représentée par des sorties d'interface.

L'interface de service inclut la définition de la façon dont les informations seront fournies au service, comment le service renverra l'information au consommateur, dans lequel ordre cette interaction se produira, quelle information particulière sera échangé (par l'intermédiaire de l'échange de message ou de toute autre manière) et, généralement comment l'interaction avec le service doit s'effectuée.

Ces définitions sont indispensables pour les travaux de spécification structurelle et comportementales des services SOA (voire chapitre 6).

4. Méthodologie de développement des architectures orientées services

La plupart des méthodologies de SOA proposent de diviser le cycle de vie de développement des SOA en six phases : Analyse orientée services, conception orientée services, développement/construction de service, teste de service, déploiement de service, administration/gestion de service [SVANIDZAITĖ, 2012]. Les deux premières phases sont les plus importantes parce que le succès du développement de SOA dépend principalement de elles.

Un certain nombre de méthodologies de développement des SOA telles qu'IBM RUP/SOMA, SOAF, SOUP, méthodologie par Thomas Earl et méthodologie par Michael Papazoglou ont été proposées pour assurer un développement réussi de SOA. On présente dans les sections suivantes une liste non exhaustive des méthodologies proposées dans le cadre d'analyse et de conception des SOA ainsi que les différentes caractéristiques de ces méthodologies

4.1 Caractéristiques des méthodologies SOA

- Stratégies d'analyse et de conception SOA: Trois stratégies (top-down, bottom-up and meet-in-the-middle) existent dans le développement de SOA, chacune est variée dans la quantité d'analyse du domaine métier et dans les dépendances entre les différents sous-systèmes.
- Couverture des phases d'analyse et de conception de SOA : Les phases d'analyse et de conception orientées services dans les méthodologies SOA peuvent être divisées en cinq activités principales qui sont encore raffinées dans des étapes. Ces étapes

sont employées pour l'évaluation de la couverture des phases d'analyse et de conception de SOA. Les activités principales des phases d'analyse et de conception des SOA sont :

- Analyse métier de l'organisation cible. Le but de cette étape est d'identifier : les objectifs de l'organisation, les buts métiers, la technologie également utilisée, les applications et les qualifications des personnes, les termes du vocabulaire métier commun, les règles métiers, les acteurs métiers et les cas d'utilisation principaux. Le résultat de cette étape est un modèle métier de haut niveau d'abstraction.
- Planification du projet SOA. Le but de cette étape est de formuler la vision et la portée du projet SOA, définir la stratégie choisie de la construction de SOA (créer les services à partir de zéro, créent des services à partir composants logiciels existants, achètent des services de tiers fournisseurs), créer le plan du projet et accomplir l'analyse financière.
- L'identification des services. Le but de cette étape est d'identifier les services candidats. Toutes les exigences fonctionnelles et non fonctionnelles pour le développement de SOA sont recueillies. Créer le modèle de processus métier est le décomposer en sous domaines métiers. Après cette étape, les services candidats, leurs spécifications initiales, les communications et les dépendances initiales sont définis. Des applications existantes sont analysées afin de trouver quels composants de logiciel peuvent être réutilisés dans le développement de SOA.
- Analyse et spécification des services candidats. Le but de cette étape est de choisir, parmi les services candidats, les services les plus pertinents pour créer les spécifications détaillées nécessaires à la réalisation de ces services. Des modèles structurels (diagrammes de classe UML par exemple) et des modèles comportementaux de services (diagrammes d'activités d'UML ou diagramme de processus BPMN) sont créés dans cette phase.
- La réalisation des services. Le but de cette étape est la construction finale des services spécifiés dans l'étape précédente.

- Degré de description : Les méthodologies de développement des SOA varient de les plus descriptives au moins descriptives. Le degré de description est évalué selon le nombre de paramètres fournis dans la description de processus, comme par exemple : les phases, les activités, étapes et entrées/sorties pour chaque étape.
- Adoption des techniques et notations existantes : La plupart de méthodologies de SOA sont basées sur des techniques telles que OOAD, BPM, EA et notations telles qu'UML et BPMN, alors que les autres n'adressent pas des techniques spécifiques ni des notations et laissent l'utilisateur de décider quelles techniques et notations sont appropriées dans une situation concrète, la chose qui rendre la méthodologie plus dure à comprendre et à utiliser par les utilisateurs inexpérimentés [SVANIDZAITĖ, 2012].

4.2 Analyse de quelques méthodologies existantes

4.2.1 IBM RUP/SOMA [IBM01, 2012]

Est une méthodologie intégrée développée par IBM dans une volonté pour apporter des aspects uniques de SOMA à RUP. Cependant, parce que SOMA est une méthodologie de propriété industrielle d'IBM, ses pleines spécifications ne sont pas disponibles.

La méthodologie se compose de quatre phases : analyse, identification, spécifications, et réalisation des services. Toutes ces phases sont de grande importance. Cependant IBM RUP/SOMA ne couvre pas le déploiement et l'administration des services.

La première phase d'analyse des transformations métiers peut être tracée à la phase de commencement de la méthodologie classique RUP. Cette phase est facultative et peut être omise si la pleine analyse des transformations du métier de l'organisation est exécutée. Elle vise à décrire le processus métier de l'organisation, pour comprendre les domaines des problématiques et les potentiels d'amélioration aussi bien que n'importe quelle information sur les demandes externes telles que des concurrents ou les tendances sur le marché. L'analyse des transformations métier comporte des activités telles que : évaluation de l'organisation cible et de ses objectifs, identification des buts métiers, définition du vocabulaire métier commun et les règles métiers, définition des acteurs métiers et ces principales cas d'utilisation, analyse de l'architecture métier.

La deuxième phase d'identification de service peut être tracée à la phase d'élaboration de RUP classique dont l'objectif est d'identifier les services candidats. L'identification de service comporte des activités telles que : La décomposition du domaine, la modélisation des buts des services et analyse des applications existante.

La troisième phase de spécification des services peut être tracée à la phase d'élaboration de RUP classique et elle focalisé sur la sélection des services candidats qui seront développés. La phase de spécification des services comporte des activités telles que : spécification des services, analyse des sous-systèmes et spécifications des composants.

La quatrième phase de réalisation des services peut être tracée à la phase de construction de RUP classique et elle est concentrée sur l'accomplissement de la conception des composants pour les implémenter. La réalisation de service comporte des activités telles que : la documentation des décisions de réalisation de service, attribution des composants de service aux différentes couches de l'architecture orientée service.

4.2.2 Service Oriented Architecture Framework (SOAF) [Erradi et al, 2006]

Cette méthode est composée de cinq phases principales : acquisition de l'information, identification de service, définition de service, réalisation de service, et planification.

Le but de SOAF est de faciliter l'identification de service, et les activités de définition et de réalisation en combinant une modélisation de haut en bas (top-down) d'un processus métier existant avec une analyse de bas en haut (bottom-up) des applications existantes.

La première phase d'acquisition de l'information a pour but de définir la portée et les contraintes du processus métiers existants et de la technologie utilisée.

Un modèle de processus métier est défini. Les services candidats qui automatisent le modèle de processus métier sont identifiés. Les besoins non fonctionnelles et les agréments de niveau métier devraient être également définis, classés par ordre de priorité.

Un mapping Processus/Application est exécuté afin d'examiner les applications logiciels existants dont le but est de découvrir les fonctionnalités d'applications candidat à exporter comme des services SOA.

La phase d'identification de service vise à définir un ensemble optimal de services. La phase de réalisation de service vise à définir les stratégies de transformation qui seront employées pour la transition de l'architecture à base d'applications logicielles fortement couplées vers la future architecture à base de services.

La phase de planification a pour but de planifier en détail la transformation et l'identification des métiers et des risques techniques.

4.2.3 Méthodologie par Papazoglou [Papazoglou et al, 2006].

(Papazoglou et autres) fournissent une méthodologie de développement de SOA qui couvre un plein cycle de vie de SOA. Elle est partiellement basée sur des méthodologies de développement bien établies comme RUP, développement à base de composants et BPM. La méthodologie est basée sur le processus itératif et incrémental et comporte une phase de planification préparatoire et huit phases principales : analyse de service, la conception de service, la construction de service, le test de service, l'approvisionnement de service, le déploiement de service, l'exécution de service et la surveillance de service.

La phase de planification est préparatoire. Les activités dans cette phase incluent l'analyse des besoins métiers et de l'examen de technologie courante, l'analyse financière du projet et une création de programme de développement de SOA. Le but de la phase d'analyse orientée services est d'obtenir les exigences pour l'application SOA. Les analystes métiers créent un modèle de processus métier qui permet aux acteurs de comprendre les services et les processus métiers disponibles. Le résultat de cette phase est la création d'un modèle de processus métier qui sera implémenté par la solution SOA. La phase d'analyse se compose de quatre activités principales : identification de processus, portée de processus, analyse de l'espace métiers et réalisation de processus.

La phase de conception des services a pour but de transformer les processus métiers en interfaces, descriptions, et modèles de compositions de services. La phase de conception se compose de deux activités : spécifications des services et spécifications des processus métiers.

4.2.4 Méthodologie de Thomas Erl [Erl, 2005], [Erl, 2008].

Cette méthodologie présente un guide étape-par-étape à travers les deux phases principales : analyse et conception orientées services. L'analyse orientée services comporte trois étapes principales : définition des exigences métiers, identification des systèmes d'automatisation

existants et modélisation des services candidats. Elle peut être divisée en deux parties principales : la première partie dans laquelle les exigences métiers sont définies et la deuxième partie dans laquelle les services candidats sont modelés.

La première partie de la phase inclut l'examinassions des buts et des objectifs métiers, l'analyse des changements potentiels sur les applications existantes pour trouver quels processus et composants d'application peuvent être employés dans le développement de la future application SOA.

Les analystes métier préparent le modèle de processus qui exprime la situation actuelle et permettent aux différents acteurs de comprendre quels processus métiers sont déjà implémentés et ce qui doit être exposé et automatisé, quel composants d'application peuvent être réutilisés. L'analyse orientée services a comme conséquence la préparation d'un modèle de processus métier qu'une application SOA l'implémentera. La deuxième partie d'analyse orientée services est un sous-processus de modélisation des services par lequel des services candidats sont identifiés et modélisés. Le sous-processus de modélisation des services permet de produire quelques objets comme : modèles conceptuels de service, modèles de capacités de service et modèles de composition des services.

4.2.5 Service-Oriented Unified Process (SOUP) [Mittal, 2012]

SOUP est une méthodologie de génie logiciel hybride qui est visée aux projets SOA. Comme il est indiqué dans le nom, cette méthodologie est principalement basée sur Rational Unified Process (RUP). Son cycle de vie se compose de six phases : Initiation, définition, conception, construction, et déploiement. La méthodologie SOUP peut être employée dans deux variations légèrement différentes : une adopte RUP pour des projets SOA initiaux et l'autre adopte la combinaison de RUP et XP pour la maintenance des applications SOA existantes.

La première phase d'Initiation vise à comprendre les besoins métier pour le développement des SOA et comment intégrer SOA dans l'organisation. L'objectif de cette phase est de décider si le projet SOA est profitable ou pas en évaluant la portée et les risques de projet. La phase d'initiation comporte des activités telles que : La formulation de la vision et de la portée du système, définition de la stratégie SOA, et la création d'un plan de communication.

La seconde phase de définition est la phase la plus critique dans le projet SOA. Elle vise à définir les exigences et à développer les cas d'utilisation.

Les objectifs de cette phase sont :

1) comprendre entièrement les processus métiers étudiés,
2) rassembler, définir et analyser les besoins fonctionnelles et non fonctionnelles en employant un processus formel de collection d'exigences comme RUP,
3) concevoir le modèle de soutien qui explique comment l'organisation soutiendra SOA,

4) préparer un plan de réalisation du projet,
5) définir une infrastructure technique qui est exigée pour soutenir entièrement la SOA.

La troisième phase de conception a comme but de traduire les réalisations des cas d'utilisations et l'architecture SOA en documents de conception détaillée. Les objectifs de cette phase sont :

1) créer le document de conception détaillée et le modèle de base de données qui expliquent la structure des services,
2) la structuration du processus de développement en définissant la technologie, les standards de codage etc.

4.3 Comparaison des méthodologies de développement des SOA

La méthodologie de SOA la plus descriptive est IBM RUP/SOMA qui est de propriété industrielle et employée couramment dans des projets industriels [SVANIDZAITĖ, 2012]. Elle supporte la stratégie d'analyse et de conception « meet-in-the-middle », elle couvre toutes les activités d'analyse et de conception de SOA. Elle a également le meilleur degré de description, parce qu'elle fournit la description d'activités, d'étapes, d'entrées et de sorties pour chaque phase. Elle adopte les techniques et notation existantes comme : BPM, UML, BPEL, WSDL, WS-BPEL.

La méthodologie de Thomas Erl ne fournit pas des descriptions détaillées sur comment lancer le projet SOA, comment exécuter l'analyse du métier de l'organisation et comment formuler la vision et la portée du projet. Cependant, elle fournit des descriptions orientées services détaillées pour les phases d'analyse et de conception, ce qui signifier qu'elle ne peut pas être utilisée dans le début du projet mais elle peut être employée conjointement avec une autre méthodologie qui fournit des recommandations détaillées sur le lancement du projet SOA. Elle supporte une stratégie d'analyse et de conception SOA descendante, elle a un bon degré de description et elle adopte également des techniques existantes telles que : BPM, WSDL, WS-BPEL, et les spécifications WS-*[SVANIDZAITĖ, 2012].

La méthodologie SOUP n'est pas assez mûre pour assurer le développement réussi de SOA parce qu'elle manque de la prescription : des phases, des activités, des artefacts, les travailleurs de processus et leurs rôles ne sont pas définis clairement. Elle supporte la stratégie d'analyse et de conception « meet-in-the-middle », mais elle ne couvre pas certaines des activités d'analyse et de conception de SOA. Il manque, dans la méthodologie SOUP, l'adoption des notations existantes telles qu'UML et BPMN qui sont largement employés dans l'analyse et la conception orientées services [SVANIDZAITĖ, 2012].

La méthodologie SOAF supporte la stratégie d'analyse et de conception « meet-in-the-middle », mais elle ne couvre quelques activités d'analyse et de conception SOA, elle n'utilise pas les techniques et les notations existantes pour assurer le développement réussi de SOA.

La méthodologie de Papazoglou supporte la stratégie d'analyse et de conception « meet-in-the-middle » de SOA, elle adopte les techniques et les notations telles que : BPM, BPMN, WSDL, BPEL, UML. Elle fournit des recommandations détaillées pour la conception et les spécifications de service, mais comme méthodologie d'analyse et de conception SOA, elle nécessite plus de description. Il ne raffine pas des activités dans les étapes concrets et ne fournit pas des entrées et des sorties de chaque étape [SVANIDZAITĖ, 2012].

5. Conclusion

Dans le présent chapitre nous avons défini les concepts de base de l'architecture orientée services ainsi que les concepts nécessaires à la modélisation de cette architecture. En particulier, nous avons définit la SOA et les services Web puis nous avons présenté les critères de modélisation de SOA ainsi que la définition des concepts de base nécessaire à la modélisation conceptuel de SOA. Nous avons présenté dans la section 4 de ce chapitre une vue d'ensemble sur les différentes méthodologies existantes dans le domaine de la modélisation des SOA avec une brève comparaison de ces méthodologies. Le cycle de vie d'une SOA commence toujours par l'identification des services candidats nécessaire à l'exécution des processus métier de l'entreprise. Le processus d'identification est considéré comme un problème d'optimisation difficile (chapitre 5) qu'on ne peut pas le résoudre par des méthodes exactes en un temps raisonnable d'où la nécessité des méthodes approchées qui font l'objet du chapitres suivant.

Chapitre 3

Les métaheuristiques et optimisation multi-objectif

Chapitre 3 Les métaheuristiques et optimisation multi-objectif

1. Introduction

Les problèmes d'optimisation occupent actuellement une place importante dans la communauté des ingénieurs et dans les travaux de recherche scientifique. En effet, ce genre de problèmes intervient dans leurs domaines d'activité qui sont très divers, comme la conception de systèmes mécaniques, le traitement d'images, l'électronique ou la recherche opérationnelle. Le problème à résoudre peut fréquemment être exprimé sous la forme générale d'un problème d'optimisation, dans lequel on définit une fonction objectif, ou fonction de coût, que l'on cherche à minimiser par rapport à tous les paramètres concernés.

Les problèmes d'optimisation sont rarement uni-objectif: il y a généralement plusieurs critères contradictoires à satisfaire simultanément. Contrairement à l'optimisation mono-objectif, la solution d'un problème multi-objectif n'est pas une solution unique, mais un ensemble de solutions, appelé *surface de pareto* [Collette et Siarry, 2002].

Dans le présent chapitre, nous présentons les concepts de base de l'optimisation suivie d'un ensemble d'algorithmes métaheuristiques utilisés pour l'optimisation mono-objectif, puis nous présentons la définition et les méthodes de l'optimisation multi-objectif.

2. Vocabulaire et définitions

Dans cette section, nous donnons un ensemble de définitions de certains termes utilisés couramment dans le domaine de l'optimisation mono-objectif et multiobjectif.

Définition 1

Le *vecteur de décision* $\vec{x}$ représente l'ensemble des variables du problème.

Définition 2

L'*espace de recherche* représente l'ensemble des valeurs pouvant être prises par les variables de décision, on le représente par χ.

Définition 3

L'*espace réalisable* représente l'ensemble des valeurs des variables satisfaisant les contraintes du problème.

Définition 4

Un point $\vec{x} \in \chi$ est un *minimum global* du problème si et seulement si : $\forall \vec{x} \in \chi, f(\vec{x}) \leq f\left(\overrightarrow{x'}\right)$.

Définition 5

Un point $\vec{x} \in \chi$ est un *minimum local* du problème si et seulement si : $\forall \overrightarrow{x'} \in \mathcal{V}(\chi), f(\vec{x}) \leq f\left(\overrightarrow{x'}\right)$, où $\mathcal{V}(\chi)$ définit un « voisinage » de $\vec{x}$.

Définition 6

Soient u et , deux vecteurs de même dimension,

$$u = v \quad ssi \quad \forall i \in \{1,2,\dots,m\}, \qquad u_i = v_i$$

$$u \leq v \quad ssi \quad \forall i \in \{1,2,\dots,m\}, \qquad u_i \leq v_i$$

$$u < v \quad ssi \quad u \leq v \wedge u \neq v$$

Les relations $\geq et >$ sont définies de manière analogue.

Définition 7

La dominance au sens de Pareto : Considérons un problème de minimisation. Soient u et v deux vecteurs de décision :

$u \prec v$ (u *domine* v) $\quad ssi \quad f(u) < f(v)$

$u \preceq v$ (u *domine faiblement* v) $\quad ssi \quad f(u) \leq f(v)$

$u \sim v$ (u *est incomparable (ou nondominé) avec* v) $\quad ssi \quad f(u) \nleq f(v) \wedge f(v) \nleq f(u)$

Définition 8

Une solution $x \in \chi_f$ est dite non dominée par rapport à un ensemble $\chi_a \subseteq \chi_f$ si et seulement si :

$$\nexists x_a \in \chi_a, x_a \prec x$$

Définition 9

Un vecteur de décision $x \in \chi$ est dit *Pareto globalement optimal* si et seulement si :

$\nexists y \in \chi, y \preceq x$. Dans ce cas, $f(x) \in \boldsymbol{\mathcal{F}}$ est appelée solution efficace.

Définition 10

Un vecteur de décision $x \in$ est dit *Pareto optimal localement* si et seulement si, pour $\delta > 0$ *fixé*: $\nexists y \in \chi, f(y) \in B(f(x), \delta)$ *et* $y \prec x$, *où* $B(f(x), \delta)$ représente une boule de rayon $f(x)$ et de rayon δ.

3. Optimisation mono-objectif

3.1 Optimisation mono-objectif difficile

Lorsqu'un seul objectif (critère) est recherché, le problème d'optimisation est mono-objectif. Un problème d'optimisation en général est défini par un espace de recherche $\boldsymbol{S}$ et une fonction objective $\boldsymbol{f}$. Le but est de trouver la solution $\boldsymbol{s}^* \in \boldsymbol{S}$ de meilleure qualité $\boldsymbol{f}(\boldsymbol{s}^*)$. Suivant le problème posé, on cherche soit le minimum soit le maximum de la fonction $\boldsymbol{f}$. Dans les sections suivantes, nous aborderons les problèmes d'optimisation essentiellement sous l'aspect minimisation, maximiser une fonction $\boldsymbol{f}$ étant équivalent à minimiser $-\boldsymbol{f}$. L'équation (3.1) résume la définition précédente.

$$\boldsymbol{s}^* = \mathbf{min}(\boldsymbol{f}(\boldsymbol{s})|\boldsymbol{s} \in \boldsymbol{S}) \qquad (\mathbf{3.1})$$

De plus, un problème d'optimisation peut présenter des contraintes d'égalité et/ou d'inégalité sur les solutions candidates $\boldsymbol{s} \in \boldsymbol{S}$, être multiobjectif si plusieurs fonctions objectifs doivent être optimisées ou encore dynamique, si la topologie de $\boldsymbol{f}$ change au cours du temps.

La valeur de fitness d'une solution de l'espace de recherche est la valeur de la fonction objective pour cette solution. L'évaluation d'une solution correspond au calcul de sa valeur de fitness.

Il existe de nombreuses méthodes déterministes qui permettent de résoudre certains types de problèmes d'optimisation en un temps fini. Cependant, ces méthodes nécessitent que la fonction objectif présente un certain nombre de caractéristiques, telles que la convexité, la continuité ou encore la dérivabilité. Parmi les méthodes les plus connues, on peut citer les méthodes de programmation linéaire, dynamique ou quadratique, la méthode de Newton, la méthode du simplex ou encore la méthode du gradient.

Certains problèmes restent cependant trop complexes à résoudre pour les méthodes déterministes. Certaines caractéristiques peuvent être problématiques pour ces méthodes. Parmi celles-ci, on peut citer la non-dérivabilité, la discontinuité ou encore la présence de bruit. Dans ce cas, le problème d'optimisation est dit difficile, car aucune méthode déterministe ne peut résoudre ce problème en un temps raisonnable. Ces méthodes permettent de trouver des solutions optimales. Mais ces méthodes s'avèrent, malgré les progrès réalisés, plutôt inefficaces à mesure que la taille du problème devient importante.

Les problèmes d'optimisation difficile se divisent en deux catégories (les problèmes à variables discrètes et les problèmes à variables continues):

- De façon générale, un problème d'optimisation à variables discrètes, ou combinatoire, consiste à trouver, dans un ensemble discret, la meilleure solution réalisable, au sens du problème défini par (3.1). Les problèmes d'optimisation combinatoire représentent une catégorie de problèmes très difficiles à résoudre et de nombreux problèmes pratiques peuvent être formulés sous la forme d'un problème d'optimisation. Le problème majeur réside ici dans le fait que le nombre

de solutions réalisables est généralement très élevé, donc il est très difficile de trouver la meilleure solution dans un temps raisonnable. Le problème du voyageur de commerce est un exemple classique de problème d'optimisation à variables discrètes. L'utilisation d'algorithmes d'optimisation stochastiques, tels que les métaheuristiques, permet de trouver une solution approchée, en un temps raisonnable.

- Les variables du problème d'optimisation sont continus. C'est le cas par exemple des problèmes d'identification, où l'on cherche à minimiser l'erreur entre le modèle d'un système et des observations expérimentales. Ce type de problèmes est moins formalisé que le précédent, mais un certain nombre de difficultés sont bien connues, comme l'existence de nombreuses variables présentant des corrélations non identifiées, la présence de bruit ou plus généralement une fonction objectif accessible par simulation uniquement.

La plupart des problèmes d'optimisations appartiennent à la classe des problèmes NP-difficile classe où il n'existe pas d'algorithme qui fournit la solution optimale en temps polynomial en fonction de la taille du problème et le nombre d'objectifs à optimiser. En outre, les méthodes de résolution exacte peuvent avoir un temps de résolution trop long. Dans ce cas, le problème d'optimisation est dit difficile, car aucune méthode exacte n'est capable de le résoudre en un temps raisonnable. Il est alors nécessaire d'avoir recours à des heuristiques de résolution dites méthodes approchées, qui fournissent un résultat sans garantir l'optimalité.

3.2 Métaheuristiques pour l'optimisation mono-objectif

Les métaheuristiques sont des algorithmes d'optimisation visant à résoudre des problèmes d'optimisation difficile (souvent issus des domaines de la recherche opérationnelle, de l'ingénierie ou de l'intelligence artificielle) pour lesquels on ne connaît pas de méthode classique plus efficace.

Les métaheuristiques sont généralement des algorithmes stochastiques itératifs, qui progressent vers un optimum global, c'est-à-dire l'extremum global d'une fonction, par échantillonnage d'une fonction objectif. Leur particularité réside dans le fait que celles-ci sont adaptables à un grand nombre de problèmes sans changements majeurs dans leurs algorithmes, d'où le qualificatif méta. Leur capacité à optimiser un problème à partir d'un nombre minimal d'informations est contrebalancée par le fait qu'elles n'offrent aucune garantie quant à l'optimalité de la meilleure solution trouvée. Seule une approximation de l'optimum global est donnée. Cependant, du point de vue de la recherche opérationnelle, ce constat n'est pas forcément un désavantage, étant donné que l'on préférera toujours une approximation de l'optimum global trouvée rapidement qu'une valeur exacte trouvée dans un temps rédhibitoire.

Les métaheuristiques sont des méthodes qui ont, en général, un comportement itératif, c'est- à-dire que le même schéma est reproduit un certain nombre de fois au cours de l'optimisation, et qui sont directes, dans le sens où elles ne font pas appel au calcul du gradient de la fonction.

Ces méthodes tirent leur efficacité du fait qu'elles sont moins facilement piégeables dans des optima locaux, car elles acceptent, au cours du traitement, des dégradations de la fonction objective et la recherche est souvent menée par une population de points et non un point unique.

Les métaheuristiques sont majoritairement conçues pour résoudre des problèmes à variables discrètes, mais font de plus en plus l'objet d'adaptations aux problèmes à variables continues. De plus, de par leur variété et leur capacité à résoudre des problèmes très divers, les métaheuristiques sont assez facilement sujettes à des extensions. Le monde des métaheuristiques est un monde en constante évolution. De nombreuses méthodes sont proposées chaque année pour tenter d'améliorer la résolution des problèmes les plus complexes [EL DOR 2012]. En général, l'utilisateur demande des méthodes efficaces et rapides permettant d'atteindre un optimum avec une précision acceptable dans un temps raisonnable, mais il a besoin aussi des méthodes simples à utiliser. Un des enjeux des métaheuristiques est donc de faciliter le choix d'une méthode et de simplifier son réglage pour l'adapter au mieux à un problème posé. De nombreuses méthodes existent dans la littérature. Dans la suite, nous présentons quelques une les plus connues.

3.3 Algorithme de recuit simulé

Le recuit simulé a été Introduit par Kirkpatrick en 1982 [Kirkpatrick et al., 1983]. Est une métaheuristique inspirée d'un processus utilisé en métallurgie. Pour modifier l'état d'un matériau, on dispose d'un paramètre de commande : la température. La technique consiste à chauffer préalablement le matériau pour lui conférer une énergie élevée. Puis on refroidit lentement le matériau en marquant des paliers de température de durée suffisante. Ce processus est appelé le recuit. (Figure 3.1)

Le recuit simulé repose sur l'algorithme de Metropolis [Metropolis et al., 1953] décrit par l'Algorithme 3.1. Cette procédure permet de sortir des minimas locaux avec une probabilité élevée si la température T est élevée et, quand l'algorithme atteint de très basses températures, de conserver les états les plus probables.

L'algorithme de recuit simulé est devenu rapidement populaire, du fait de son efficacité et sa facilité d'adaptation à un grand nombre de problèmes. En revanche, cet algorithme présente l'inconvénient de disposer d'un nombre élevé de paramètres (température initiale, règle de décroissance de la température, durée des paliers de température, etc.) qui rendent les réglages de l'algorithme assez empiriques. L'algorithme 3.2 montre la procédure du recuit simulé.

Algorithme 3.1 : Algorithme de Metropolis

Initialiser un point de départ x_0 et une température T
Pour i = 1 à n **faire**
  Tant que x_i n'est pas accepté **faire**
    Si $f(x_i) \leq f(x_{i-1})$: accepter x_i
    Si $f(x_i) > f(x_{i-1})$: accepter x_i avec la probabilité $e^{-\frac{f(x_i)-f(x_{i-1})}{T}}$
  Fin Tant que
Fin Pour

Algorithme 3.2 : recuit simulé

Déterminer une configuration aléatoire S
Choix des mécanismes de perturbation d'une configuration
Initialiser la température T
Tant que la condition d'arrêt n'est pas atteinte **faire**
    Tant que l'équilibre n'est pas atteint **faire**
        Tirer une nouvelle configuration S'
        Appliquer la règle de Metropolis
        Si $f(S') < f(S)$
            $S_{min} = S'$
            $f_{min} = f(S')$
        Fin Si
    Fin Tant que
    Décroître la température
Fin Tant que
Retourner S_{min}

Par contre, la méthode du recuit simulé a l'avantage d'être souple vis-à-vis des évolutions du problème et facile à implémenter. Elle a donné d'excellents résultats pour un certain nombre de problèmes, le plus souvent de grande taille. Bien qu'initialement créée pour fonctionner avec des variables discrètes, la méthode du recuit simulé possède des versions continues [Courat 1994].

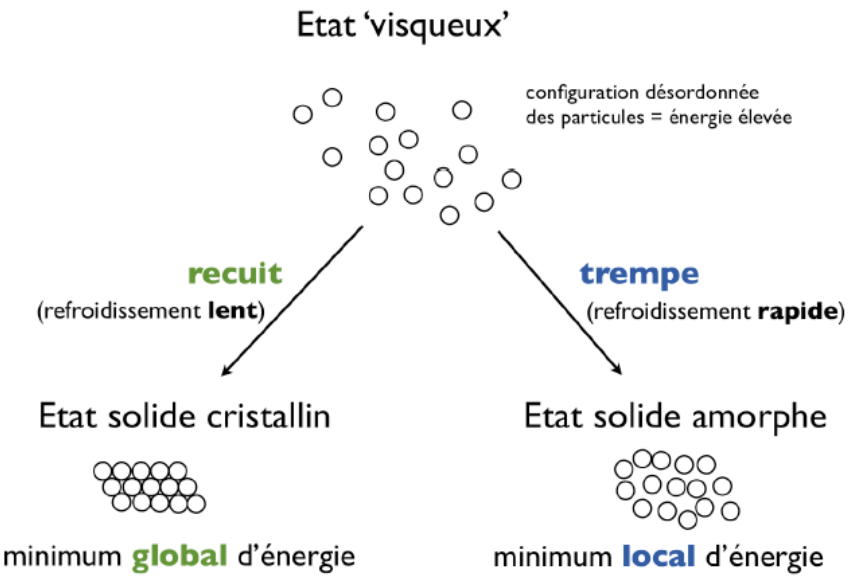


Figure 3.1 Comparaison des techniques du recuit et de la trempe [Siarry et al, 2003]

3.4 Recherche Taboue

La recherche taboue a été introduite par Glover en 1986 [Glover, 1986] comme une nouvelle stratégie pour échapper aux optima locaux en utilisant une notion de mémoire. L'idée ici est d'ajouter au processus de recherche une mémoire qui permette de mener une recherche plus intelligente dans l'espace des solutions. Comme l'algorithme de recuit simulé, la méthode de recherche taboue fonctionne avec une seule configuration courante, qui est actualisée au cours des itérations successives. La nouveauté ici est que, pour éviter le risque de retour à une configuration déjà visitée, on tient à jour une liste de mouvements interdits, appelée liste tabou. Cette liste contient m mouvements $(t \rightarrow s)$ qui sont les inverses des m derniers mouvements $(t \rightarrow s)$ effectués. L'algorithme modélise ainsi une

forme primaire de mémoire à court terme. L'algorithme de recherche tabou peut être résumé par l'Algorithme 3.3

Dans sa forme de base, l'algorithme de recherche tabou présente l'avantage de comporter moins de paramètres que l'algorithme de recuit simulé. Cependant, l'algorithme n'étant pas toujours performant, il est souvent approprié de lui ajouter des processus d'intensification et/ou de diversification, qui introduisent de nouveaux paramètres de contrôle.

Algorithme 3.3 : Recherche Taboue

Déterminer une configuration aléatoire s
Initialiser une liste taboue vide
Tant que le critère d'arrêt n'est pas atteint **faire**
  Perturbation de s suivant N mouvements non tabous
  Évaluation des N voisins
  Sélection du meilleur voisin t
  Actualisation de la meilleure position connue s^*
  Insertion du mouvement $t \rightarrow s$ dans la liste tabou
$s = t$
Fin Tant que
Retourner s^*

3.5 Algorithmes évolutionnaires

Les algorithmes évolutionnaires, élaborés au cours des années 1950, forment une famille d'algorithmes de recherche inspirés de l'évolution biologique des espèces. L'idée ici est de s'inspirer de la théorie darwiniste de sélection naturelle pour résoudre des problèmes d'optimisation. Au cours des années 1970, avec l'avènement des calculateurs de forte puissance, de nombreuses tentatives ont été réalisées pour modéliser cette approche. Trois approches se sont détachées :

- Les stratégies d'évolution qui ont été conçues comme une méthode d'optimisation à variables continues.
- La programmation évolutionnaire, qui visait à faire évoluer les structures d'automates à états finis par des successions de croisements et de mutations.
- Les algorithmes génétiques, qui ont été conçus pour résoudre des problèmes d'optimisation à variables discrètes.

Un algorithme évolutionnaire est typiquement composé de trois éléments fondamentaux :

- une population constituée de plusieurs individus représentant des solutions potentielles (configurations) du problème donné, permettant de mémoriser les résultats à chaque étape du processus de recherche.
- un mécanisme d'évaluation (fitness) des individus permettant de mesurer la qualité de l'individu,
- un mécanisme d'évolution de la population permettant, grâce à des opérateurs prédéfinis (tels que la sélection, la mutation et le croisement), d'éliminer certains individus et d'en créer de nouveaux. Ces méthodes sont applicable dans la plupart

des problèmes d'optimisation (multimodaux, non continu, contraints, bruités, multiobjectif, dynamiques, etc.).

Algorithme 3.4 Algorithme évolutionnaire générique

```
Initialisation de la population de μ individus
Évaluation des μ individus
Tant que le critère d'arrêt n'est pas atteint faire
   Sélection de λ  individus en vue de la phase de reproduction
   Croisement des λ individus sélectionnés
   Mutation des λ individus sélectionnés
   Évaluation des λ' enfants obtenus
   Sélection pour le remplacement
Fin Tant que
Retourner les meilleurs individus
```

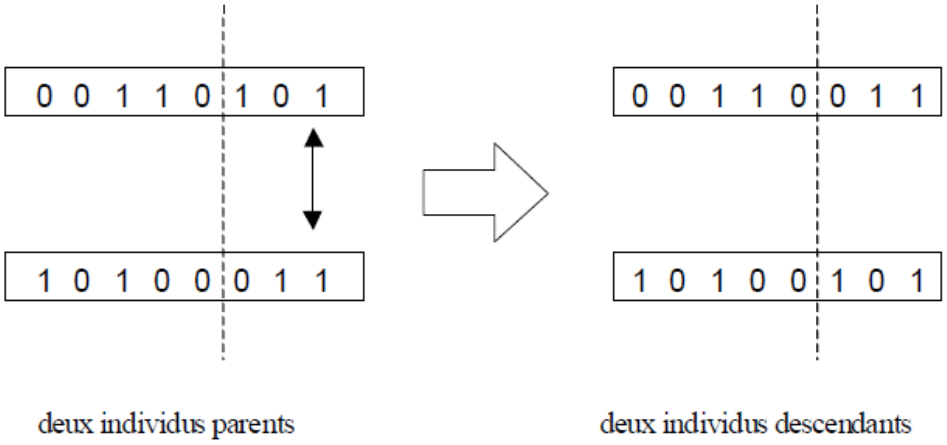
Les approches évolutionnaires s'appuient toutes sur un modèle commun présenté par l'Algorithme 3.4. Les individus soumis à l'évolution sont des solutions possibles du problème posé. L'ensemble de ces individus constitue une population. Cette population évolue durant une succession d'itérations, appelées générations. Au cours de chaque génération, une succession d'opérateurs est appliquée aux individus de la population pour engendrer une nouvelle population, en vue de la génération suivante. Chaque opérateur utilise un ou plusieurs individus de la population appelé(s) parent(s) pour engendrer de nouveaux individus, appelés enfants. A la fin de chaque génération, un certain nombre d'individus de la population sont remplacés par des enfants créés durant la génération.

Un algorithme évolutionnaire dispose donc de trois opérateurs principaux :

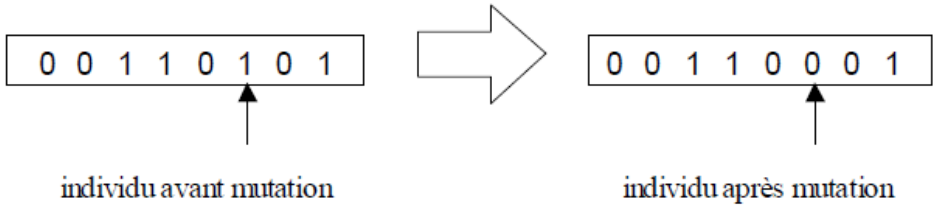
1. Un opérateur de sélection, qui favorise la propagation des meilleures solutions dans la population, tout en maintenant une certaine diversité génétique au sein de celle-ci ;
2. Un opérateur de croisement, mis en œuvre lors de la phase de création des enfants. Son but est d'échanger les gènes des différents parents pour créer les enfants.
3. Un opérateur de mutation, consistant à tirer aléatoirement une composante de l'individu parent et à la remplacer par une valeur aléatoire. L'apport d'un caractère aléatoire à la création de la descendance permet ainsi de maintenir une certaine diversité dans la population.

La Figure 3.2 présente l'exemple d'un croisement et de mutation simple pour des individus codés en représentation binaire.

L'opérateur de mutation consiste à tirer aléatoirement une composante de l'individu parent et à la remplacer par une valeur aléatoire. L'opérateur de mutation apporte un caractère aléatoire à la création de la descendance, qui permet de maintenir une certaine diversité dans la population.



a – Croisement (mono-point).



b – Mutation (d'un seul bit).

Figure 3.2 Exemple d'opérateur de croisement et de mutation [Siarry et al, 2003]

3.6 Optimisation par Essaim Particulaire

L'Optimisation par Essaim Particulaire (OEP), ou Particle Swarm Optimization (PSO) en anglais, est un algorithme évolutionnaire qui utilise une population de solutions candidates pour développer une solution optimale au problème. Cet algorithme a été proposé par Russel Eberhart (ingénieur en électricité) et James Kennedy (socio-psychologue) en 1995 [Kennedy et al, 1995]. Il s'inspire à l'origine du monde du vivant, plus précisément du comportement social des animaux évoluant en essaim, tels que les bancs de poissons et les vols groupés d'oiseaux. En effet, on peut observer chez ces animaux des dynamiques de déplacement relativement complexes, alors qu'individuellement chaque individu a une « intelligence» limitée, et ne dispose que d'une connaissance locale de sa situation dans l'essaim. L'information locale et la mémoire de chaque individu sont utilisées pour décider de son déplacement. Des règles simples, telles que « rester proche des autres individus », « aller dans une même direction » ou « aller à la même vitesse », suffisent pour maintenir la cohésion de l'essaim, et permettent la mise en œuvre de comportements collectifs complexes et adaptatifs.

Kennedy et Eberhart se sont inspirés de ces comportements socio-psychologiques pour créer l'OEP. Un essaim de particules, qui sont des solutions potentielles au problème d'optimisation, survole l'espace de recherche, en quête de l'optimum global. Le déplacement d'une particule est influencé par les trois composantes suivantes :

- Une composante *physique* : la particule tend à suivre sa direction courante de déplacement ;
- Une composante *cognitive* : la particule tend à se diriger vers le meilleur site par lequel elle est déjà passée ;
- Une composante *sociale* : la particule tend à se fier à l'expérience de ses congénères et, ainsi, à se diriger vers le meilleur site déjà atteint par ses voisins.

Dans le cas d'un problème d'optimisation, la qualité d'un site de l'espace de recherche est déterminée par la valeur de la fonction objective en ce point. La Figure 3.3 illustre la stratégie de déplacement d'une particule.

Dans un espace de recherche de dimension D, la particule i de l'essaim est modélisée par son vecteur position $\vec{X}_i = (x_{i1}; x_{i2}; \dots ; x_{iD})$ et par son vecteur vitesse $\vec{V}_i = (v_{i1}; v_{i2}; \dots ; v_{iD})$. Cette particule garde en mémoire la meilleure position par laquelle elle est déjà passée, que l'on note $\vec{P}_i = (P_{i1}; P_{i2}; \dots ; P_{iD})$. La meilleure position atteinte par toutes les particules de l'essaim est notée $\vec{g} = (g_1; g_2; \dots ; g_D)$. Au temps t, le vecteur vitesse est calculé à partir de (3.2).

$$v_{ij}(t) = w \cdot v_{ij}(t-1) + c_1 \cdot r_1 \left(p_{ij}(t-1) - x_{ij}(t-1)\right) + c_2 \cdot r_2 \cdot \left(g_j(t-1) - x_{ij}(t-1)\right), j \in \{1, \dots, D\} \quad (3.2)$$

Où w est en général une constante appelée, *coefficient d'inertie*, c_1 et c_2 sont deux constantes, appelées *coefficients d'accélération*, r_1 et r_2 sont deux nombres aléatoires tirés uniformément dans [0,1] à chaque itération et pour chaque dimension.

$w \cdot v_{ij}(t-1)$ Correspond à la composante *physique* du déplacement. Le paramètre w contrôle l'influence de la direction de déplacement sur le déplacement future. Il est à noter que, dans certaines applications, le paramètre w peut être variable.

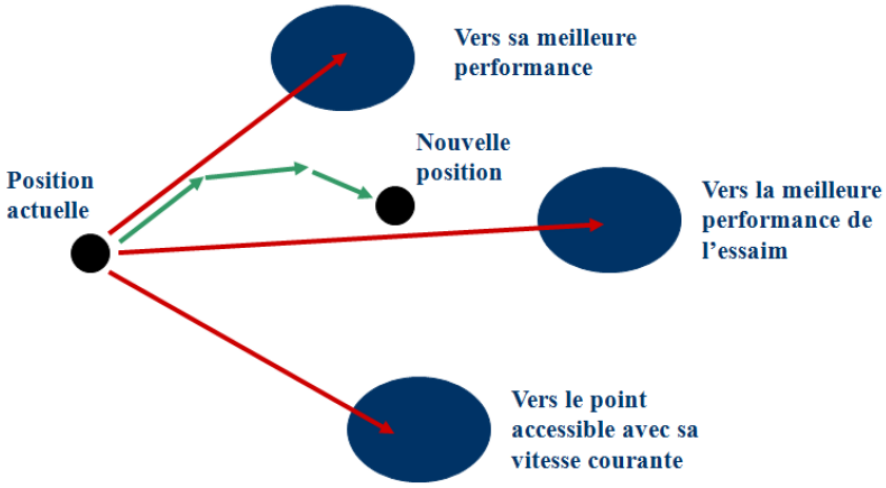


Figure 3.3 Déplacement d'une particule [COOREN 2008]

$c_1 \cdot r_1 \left(p_{ij}(t-1) - x_{ij}(t-1)\right)$ Correspond à la composante cognitive du déplacement. c_1 contrôle le comportement cognitif de la particule.

$c_2 \cdot r_2 \cdot \left(g_j(t-1) - x_{ij}(t-1)\right)$ Correspond à la composante sociale du déplacement. c_2 Contrôle l'aptitude sociale de la particule.

La combinaison des paramètres $w, c_1 \; et \; c_2$ permet de régler la balance entre les phases diversification et intensification du processus de recherche. Il est à noter que le terme vitesse est ici abusif car les vecteurs $\overrightarrow{V_i}$ ne sont pas homogènes à une vitesse. Il serait plus approprié de parler de direction de déplacement. Cependant, pour respecter l'analogie avec le monde animalier, les auteurs ont préféré utiliser le terme vitesse.

La position au temps t de la particule i est alors définie par (3.3).

$$x_{ij}(t) = x_{ij}(t-1) + v_{ij}(t), j \in \{1, \dots, D\} \tag{3.3}$$

L'OEP est un algorithme à population. Il commence par une initialisation aléatoire de l'essaim dans l'espace de recherche. A chaque itération de l'algorithme, chaque particule est déplacée suivant (3.2) et (3.3). Une fois le déplacement des particules effectué, les nouvelles positions sont évaluées. Les $\overrightarrow{P_i}$ ainsi que $\vec{g}$ sont alors mis à jour. Cette procédure est résumée par l'Algorithme 3.5. N est le nombre de particules de l'essaim.

Le critère d'arrêt peut être différent suivant le problème posé. Si l'optimum global est connu a priori, on peut définir une erreur acceptable ε comme critère d'arrêt. Sinon, il est commun de fixer un nombre maximum d'évaluations de la fonction objectif ou un nombre maximum d'itérations comme critère d'arrêt. Cependant, au regard du problème posé et des exigences de l'utilisateur, d'autres critères d'arrêt peuvent être utilisés.

Algorithme 3.5 : Algorithme d'optimisation par essaim particulaire

Initialisation aléatoire des positions et des vitesses de chaque particule
Pour chaque particule $i, \overrightarrow{p_i} = \overrightarrow{X_i}$
Tant que le critère d'arrêt n'est pas atteint **faire**
  Pour $i = 1$ *à* N **faire**
    Déplacement de la particule à l'aide de (3.2) et (3.3)
    Évaluation des positions
    si $f(\overrightarrow{X_i}) < f(\overrightarrow{p_i})$
      $\overrightarrow{p_i} = \overrightarrow{X_i}$
    Fin si
    Si $f(\overrightarrow{p_i}) < f(\vec{g})$
      $\vec{g} = \overrightarrow{p_i}$
  Fin Pour
Fin Tant que
Retourner Meilleures particules

Il est possible que le déplacement d'une particule la conduise à sortir de l'espace de recherche. Dans ce cas, on peut assister à une amplification des rétroactions positives, qui

conduit à une divergence de système. Pour s'affranchir de ce problème, on peut introduire un nouveau paramètre $Vmax$, qui va permettre de contrôler le mouvement des particules.

3.7 Algorithmes de colonies de fourmis

Les algorithmes à base de colonies de fourmis ont été introduits par Dorigo [Dorigo et al., 1996]. Une des applications principales de la méthode originale était le problème du voyageur de commerce et depuis elle a considérablement évolué. Les algorithmes de colonies de fourmis sont nés d'une constatation simple : les insectes sociaux, et en particulier les fourmis, résolvent naturellement des problèmes complexes. Un tel comportement est possible car les fourmis communiquent entre elles de manière indirecte par le dépôt de substances chimiques, appelées phéromones, sur le sol. Ce type de communication indirecte est appelé *stigmergie*.

La principale illustration de ce constat est donnée par la Figure 3.4. On voit sur cette figure que, si un obstacle est introduit sur le chemin des fourmis, les fourmis vont, après une phase de recherche, avoir tendance à toutes emprunter le plus court chemin entre le nid et l'obstacle. Plus le taux de phéromone à un endroit donné est important, plus une fourmi va avoir tendance à être attirée par cette zone. Les fourmis qui sont arrivées le plus rapidement au nid en passant par la source de nourriture sont celles qui ont emprunté la branche la plus courte du trajet. Il en découle donc que la quantité de phéromones sur ce trajet est plus importante que sur le trajet plus long. De ce fait, le plus court chemin a une probabilité plus grande d'être emprunté par les fourmis que les autres chemins et sera donc, à terme, emprunté par toutes les fourmis.

L'Algorithme 3.6 présente la méthode proposée par les auteurs. Si l'on considère un problème de voyageur de commerce à N villes, chaque fourmi k parcourt le graphe et construit un trajet de longueur $n = |N|$. Pour chaque fourmi, le trajet d'une ville i à une ville j dépend de :

- la liste des villes déjà visitées, qui définit les mouvements possibles à chaque pas, quand la fourmi **k** est sur la ville i: J_i^k.
- l'inverse de la distance entre les villes $\mathrm{n_{ij}} = \frac{1}{\mathrm{d_{ij}}}$, appelée *visibilité*. Cette information est utilisée pour diriger les fourmis vers des villes proches et, ainsi, éviter de trop longs déplacements.
- la quantité de phéromone déposée sur l'arête reliant deux villes $r_{ij}(t)$, appelée intensité de la *piste*. Cette quantité définit l'attractivité d'une piste et est modifiée après le passage d'une fourmi. C'est la pseudo-mémoire du système.

La règle de déplacement est la suivante :

$$p_{ij}^k(t) = \begin{cases} \dfrac{\left(r_{ij}(t)\right)^{\alpha}(n_{ij})^{\beta}}{\sum_{l \in J_i^k}\left(r_{il}(t)\right)^{\alpha}(n_{il})^{\beta}} & si\ j \in J_i^k \\ 0 & si\ j \notin J_i^k \end{cases} \qquad (3.4)$$

où α *et* β sont deux paramètres contrôlant l'importance relative de l'intensité et de la visibilité.

Après un tour complet, chaque fourmi dépose une quantité de phéromone $\Delta r_{ij}^{k}(t)$ sur l'ensemble de son parcours. Cette quantité dépend de la *qualité* de la solution trouvée et est définie par :

$$\Delta r_{ij}^{k}(t) = \begin{cases} \dfrac{Q}{L^{k}(t)} & si\ (i,j) \in T^{k}(t) \\ 0 & si\ (i,j) \notin T^{k}(t) \end{cases} \qquad (3.5)$$

Où $T^{k}(t)$ est le trajet effectué par la fourmi k à l'itération $t, L^{k}\ (t)$ est la longueur de $T^{k}\ (t)$ et Q est un paramètre fixé.

Enfin, il est nécessaire d'introduire un processus d'évaporation des phéromones. En effet, pour éviter de se faire piéger dans des solutions sous-optimales, il est nécessaire qu'une fourmi oublie les mauvaises solutions. La règle de mise à jour est donc :

$$r_{ij}(t+1) = (1-p).r_{ij}(t) + \Delta r_{ij}(t) \qquad (3.6)$$

Où $\Delta r_{ij}(t) = \sum_{k=1}^{m} \Delta r_{ij}^{k}(t)$ et m est le nombre de fourmis.

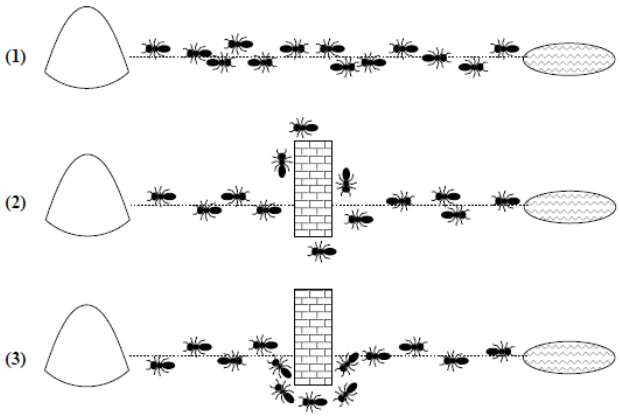


Figure 3.4 Détermination du plus court chemin par une colonie de fourmis [Siarry et al, 2003]

Algorithme 3.6 : Algorithme de colonies de fourmis pour le problème du voyageur de commerce

Tant que le critère d'arrêt n'est pas atteint **faire**
  Pour $k = 1$ *à* m **faire**
    Choisir une ville au hasard
    Pour chaque ville non visitée i **faire**
    Choisir une ville **j,** dans la liste J_i^k des villes restantes selon (3.4)
    Fin Pour
    Déposer une piste $\Delta r_{ij}^{k}(t)$ sur le trajet $T^{k}(t)$ conformément à (3.5)
  Fin Pour
  Évaporer les pistes selon (3.6)
Fin Tant que
Retourner Meilleure solution

3.8 Algorithme à estimation de distribution

Les algorithmes à estimation de distribution (EDA pour Estimation of Distribution Algorithm) ont été conçus comme une variante des algorithmes évolutionnaires [Mülhenbein et al., 1996]. Cependant, ces méthodes n'utilisent ni opérateur de croisement, ni opérateur de mutation. Un modèle probabiliste est utilisé pour engendrer des solutions candidates, la distribution de probabilité étant estimée à chaque itération à partir des informations apportées par la précédente génération. L'estimation de la distribution de probabilité permet d'estimer les relations entre les différents individus, alors que celles-ci sont exprimées de manière implicite dans le cas des algorithmes évolutionnaires.

Le principe général d'un EDA est similaire à un algorithme évolutionnaire. On initialise tout d'abord aléatoirement une population D_0. Ensuite, à l'itération i, on utilise un opérateur de sélection pour déterminer la population D_{i-1}^{se}. L'information apportée par cette population est alors utilisée pour estimer la distribution de probabilité $pl(x)$ de l'itération courante. Une nouvelle population D_i est créée aléatoirement à partir de la distribution de probabilité $pl(x)$. Le parallèle avec les algorithmes évolutionnaires est ici respecté, le couple croisement-mutation étant remplacé par l'estimation de la distribution. La phase de diversification est ainsi assurée par l'utilisation d'une distribution de probabilité explicite, alors que la phase d'intensification n'est assurée que par la présence d'un opérateur de sélection. Cette procédure est résumée par l'Algorithme 3.7.

La principale difficulté de cette méthode est le choix et l'estimation de la distribution de probabilité. En effet, il faut choisir au préalable quel modèle de distribution va être utilisé par l'algorithme et, à chaque itération, estimer les paramètres de cette distribution. De nombreux modèles de distributions ont été proposés dans la littérature. Ces modèles peuvent se diviser en trois catégories :

- Les modèles sans dépendance : la distribution de probabilité est factorisée à partir de distributions indépendantes univariantes pour chaque dimension. Ce cas a le défaut d'être le plus souvent inapplicable dans les cas réels, les variables étant la plupart du temps corrélées entre elles.
- Les modèles à dépendances bivariantes : la distribution de probabilité est factorisée à partir de distributions bivariantes.
- Les modèles à dépendances multiples : la distribution de probabilité est factorisée à partir de statistiques d'ordre supérieur à deux. Il est à noter que, dans le cas continu, le modèle de distribution le plus souvent utilisé est la distribution gaussienne. L'utilisation de modèles avec dépendances introduit l'utilisation de réseaux probabilistes, les réseaux bayésiens ou gaussiens étant le plus souvent utilisés.

Algorithme 3.7 : Algorithme à Estimation de Distribution

Engendrer aléatoirement M individus pour constituer la population D_0
Tant que le critère d'arrêt n'est pas atteint **faire**
$i = i + 1$
Sélectionner N individus dans D_{i-1} à l'aide d'un opérateur de sélection pour constituer la population D_{i-1}^{se}

Estimer la distribution de probabilité $pl(x) = p(x|D_{i-1}^{se})$

Échantillonner M individus selon $pl(x)$ pour constituer Di
Fin Tant que
Retourner Meilleurs individus

4. Optimisation multiobjectif

Les métaheuristiques ont toutes été conçues à l'origine pour résoudre des problèmes mono objectifs. Cependant, devant l'omniprésence des problèmes multi-objectifs dans les cas réels, de nouvelles méthodes ont dû être créées pour résoudre de tels problèmes où il faut optimiser plusieurs objectifs contradictoires. Le but ne consiste pas ici à trouver un optimum global, mais à trouver un ensemble d'optima, qui forment une surface de compromis pour les différents objectifs du problème.

4.1 Définition du problème

Un problème d'optimisation multi-objective est un problème du type :

$$Minimiser\ \vec{f}(\vec{x}) = \left(f_1(\vec{x}), f_2(\vec{x}), \dots, f_k(\vec{x})\right) \tag{3.7}$$

Sous les contraintes suivantes :

$$g_i(\vec{x}) \leq 0, i = 1,2, \dots, m \tag{3.8}$$

$$h_i(\vec{x}) = 0, i = 1,2, \dots, p \tag{3.9}$$

Où $\vec{x} = (x_1, x_2, \dots, x_d)$ est une position dans l'espace de recherche, $f_i: \mathbb{R}^D \to \mathbb{R}\ , i = 1, \dots, k$ sont les fonctions objectifs du problème, $g_i: \mathbb{R}^D \to \mathbb{R}\ , i = 1, \dots, m$ sont les contraintes d'inégalité du problème et $h_i: \mathbb{R}^D \to \mathbb{R}\ , i = 1, \dots, p$ sont les contraintes d'égalité du problème.

Le problème de l'optimisation multiobjectif réside dans le fait que l'on ne sait pas « classer » suivant plusieurs critères car $\mathbb{R}^K$ ne possède pas de relation d'ordre total si $k \neq 1$.

Dans le cas multiobjectif, le concept d'optimum est différent que dans le cas mono-objectif. En effet, on n'est plus ici à la recherche d'un unique optimum global, mais plutôt d'une surface de solutions qui offrent un bon « compromis » entre les différents objectifs. Pour bien comprendre la notion d'optimalité dans le cas multiobjectif, on introduit les définitions suivantes, basées sur les travaux de Pareto :

Définition 11

Étant donné $\vec{x}, \vec{y} \in \mathbb{R}^K$, on dit que $\vec{x} \leq \vec{y}$ *si* $x_i \leq y_i$ pour $i = 1, \dots, k$, *et que* $\vec{x}$ *domine* $\vec{y}$ (*noté* $\vec{x} \prec \vec{y}$) si $\vec{x} \leq \vec{y}$ et $\vec{x} \neq \vec{y}$.

Définition 12

On dit qu'un vecteur solution $\vec{x} \in \chi \subset \mathbb{R}^D$ est *non-dominé* dans χ s'il n'existe pas d'autre vecteur $\overrightarrow{x'} \in \chi$ tel que $\vec{f}(\vec{x}') \prec \vec{f}(\vec{x})$.

Définition 13

On dit qu'un vecteur solution $\vec{x} \in \mathcal{F} \subset \mathbb{R}^D$ est Pareto-optimal s'il est non-dominé dans $\mathcal{F}$.

Définition 14

L'ensemble optimal de Pareto est défini par :

$$P^* = \{\vec{x} \in \mathcal{F} | x \; est \; Pareto - Optimal \} \tag{3.10}$$

Définition 15

Le front optimal de Pareto est défini par :

$$\mathcal{F}_{p*} = \left\{\vec{f}(\vec{x}) \in \mathbb{R}^k \middle| \vec{x} \in P^*\right\} \tag{3.11}$$

Le but d'un algorithme d'optimisation multiobjectif est donc de trouver les positions de l'espace de recherche qui correspondent au front de Pareto $\mathcal{F}_{p*}$. Les algorithmes d'optimisation multiobjectif se divisent en trois familles : les approches agrégatives, les approches non-Pareto et les approches Pareto.

4.2 Approche agrégative

Cette approche, dite approche naïve, consiste à transformer un problème multiobjectif en problème mono-objectif pour y appliquer les méthodes déjà existantes. Pour cela, on réalise une combinaison des f_i pour obtenir un objectif unique F.

$$F(\vec{x}) = \sum_{i=1}^{k} a_i f_i(\vec{x}) = \left\langle \vec{a} \middle| \vec{f} \right\rangle \tag{3.12}$$

Où les paramètres a_i sont strictement positifs et $\sum_{i=1}^{k} a_i = 1$ Ils représentent les poids affectés aux différents critères. D'un point de vue géométrique, l'agrégation peut être vue comme la projection du vecteur objectif $\vec{f}$ sur le vecteur « poids » $\vec{a}$.

Cette méthode présente pour avantage d'être simple à mettre en œuvre et elle ne fournit qu'une seule solution, ce qui évite l'intervention d'un décideur. Les principaux problèmes sont la détermination des paramètres a_i et le choix des interactions entre les différents critères.

4.3 Approche non-Pareto

Les méthodes basées sur une approche non-Pareto ont pour caractéristique de traiter les objectifs séparément. Deux groupes de méthodes existent : les méthodes à sélection lexicographique et les méthodes à sélection parallèle.

Dans une approche classique de sélection lexicographique, les fonctions sont optimisées séquentiellement, suivant un ordre déni a priori. Cet ordre permet de définir le poids des objectifs. Plusieurs métaheuristiques ont été utilisées pour la résolution des problèmes multiobjectifs à sélection lexicographique.

4.4 Approche Pareto

Contrairement aux deux précédentes approches, l'approche Pareto utilise la notion de dominance pour sélectionner des solutions faisant converger la population vers un ensemble de solutions qui approchent avec justesse le front de Pareto. Cette approche a été initiée par Goldberg en 1989. Cette approche assure un traitement équitable de chaque critère, car il n'y a pas de classement a priori de l'importance des critères. Ainsi, l'algorithme fournit un ensemble de solutions qui approchent le front de Pareto. Le choix de la solution finale revient donc à l'utilisateur, qui doit choisir parmi l'ensemble fourni la solution qui lui convient le mieux. Ces méthodes se sont avérées être les plus efficaces donc, de nos jours, la majorité des algorithmes utilisent une approche Pareto pour traiter les problèmes multiobjectifs.

5. Conclusion :

Nous avons introduit dans ce troisième chapitre les problèmes d'optimisation difficile, qu'ils soient mono-objectif ou multi-objectif. Nous avons présenté une famille de méthodes stochastiques pour résolution de ces problèmes : les métaheuristiques, ces méthodes visent à résoudre un large panel de problèmes, sans pour autant que l'utilisateur ait à modifier leur structure. La plupart de ces méthodes sont inspirées d'analogies avec des domaines aussi variés que la physique ou la génétique.

Toutes les méthodes présentées dans ce chapitre peuvent être employées pour résoudre notre problématique d'identification automatique des services. Nous avons choisi l'approche des algorithmes génétiques pour la mise en œuvre de notre prototype à cause de sa rapidité, sa simplicité, et son adaptabilité aux problèmes de clustering.

Chapitre 4

Approches d'identification et de spécification des services : état de l'art

Chapitre 4 Approches d'identification et de spécification des services : état de l'art

1. Introduction

L'ingénierie dirigée par les modèles (MDE) est une approche de développement basée sur l'utilisation des modèles dans la construction de logiciel. Ces modèles guident alors le processus de développement, c.-à-d. ils peuvent être employés pour comprendre, estimer, communiquer et produire le code [Gherbi et al, 2009]. MDE se concentre sur les modèles de domaine qui peuvent être employés dans diverses transformations de modèles pour produire d'autres modèles utiles tels que le code. La productivité peut être améliorée avec des transformations automatisées.

L'architecture dirigée par les modèles (MDA) est une initiative bien connue proposés par Object Management Group (OMG) pour mettre en application une approche centrée sur les modèles en fournissant un ensemble d'outils qui gèrent les modèles [OMG02]. Elle propose une séparation du processus de développement en différent niveaux d'abstraction, tels qu'un PIM (modèle indépendant de plateforme) et un PSM (modèle spécifique de plateforme).

MDE est plus complète que MDA parce qu'elle considère multiple dimensions de modélisation en plus de dimensions indépendantes/spécifiques aux plateformes, telles que les différents types de modèles d'organisation. La vraie motivation pour MDE est de préserver des efforts d'analyse, de conception et de développement, d'améliorer la productivité et faciliter la migration des applications d'une plateforme à une autre.

L'architecture orientée services (SOA) est un modèle architectural qui permet à des concepteurs de concevoir et développer les logiciels en tant que services interopérables. Elle facilite le développement des services qui sont modulaires et qui peuvent être facilement intégrés et réutilisés par d'autres applications. Le point fort d'un service est la description en un format compréhensible par ordinateur. La chose qui permet aux consommateurs d'accéder aux services SOA d'une manière normalisée et sans devoir comprendre comment le service est implémenté.

Puisque SOA peut se composer de plusieurs artefacts tels que les services, les contrats, les participants et les relations, ils pourraient devenir très complexes. Une bonne manière de la comprendre est de la modéliser. Les modèles de SOA nous aideraient à expliquer, formaliser et comprendre cette architecture.

Il existe plusieurs initiatives de modélisation des SOAs. Une approche prometteuse est d'appliquer MDE au développement des SOAs. L'avantage potentiel est que les applications orientées services sont modélisées à différents niveaux d'abstraction, la chose qui permette de rendre les applications orientées services plus compréhensible par tous les acteurs SOA. MDE peut être employé pour accomplir la séparation des modèles technologiques de ceux indépendants de la technologie. Ceci a pu augmenter la vitesse du

développement des applications SOA et de réduire de ce fait les coûts de logiciel. La migration d'une technologie à une autre technologie plus nouvelle devrait coûter moins d'effort en suivant l'approche MDE. Mais avant de modéliser les services, il faut identifier les services candidats nécessaire à l'exécution du processus métier de l'entreprise.

Plusieurs approches pour le développement des services dans SOA suggèrent le processus métier comme point de départ. L'identification joue un rôle critique puisque elle est la fondation des autres étapes dans le cycle de vie de l'architecture SOA. L'attention portée à cette problématique est due à la complexité de la mise en place d'une telle architecture et le besoin d'une approche systématique pour faciliter l'identification (ou extraction) des services. Ce chapitre est divisé en deux parties, dans la première nous présentons quelques travaux de recherche qui ont porté sur les techniques d'identification des services SOA, et dans la deuxième partie, nous présentons les approches de spécification des services à base MDA.

2. Approches d'identification de services

Le service est considéré comme étant la brique de base d'une SOA. Partant de ce principe, la réussite d'un projet SOA passe par une identification précise des services les plus appropriés à l'entreprise et qui respectent les principes de base cités au chapitre précédent (la réutilisabilité, la compossibilité, etc.). Malgré le consensus qui existe sur ces principes, il n'existe pas une approche ou une méthodologie standard pour atteindre cette fin d'identification de service. Plusieurs approches ont été avancées pour guider le processus d'identification de services. Ces approches peuvent être comparées selon des critères techniques, économiques, métiers, etc [Boerner et Goeken, 2009; Klose et al 2007], ou encore selon les entrées utilisés et les différents types de services identifiés. En se basant sur la nature des entrées utilisées, nous pouvons distinguer trois types d'approches d'identification de services, à savoir : l'approche descendante (Top-Down), l'approche ascendante (Bottom-up) et l'approche mixte.

2.1 Approche descendante (Top-Down)

Étant donné que l'objectif principal d'une SOA, est de pouvoir aligner le métier de l'entreprise avec son SI, les partisans de l'approche Top-Down se basent sur l'analyse des processus et objectifs métiers afin d'identifier les services les plus appropriés à l'entreprise. L'approche Top-Down consiste principalement à décomposer les processus métiers en tâches métier plus fines, et ce, pour permettre l'identification des services de granularités différentes. Ce type d'approches est mené généralement par des analystes des processus.

[Suntae et al, 2008] proposent une approche, qui consiste principalement, à identifier les services à partir des objectifs métiers et desceller les différents scénarios suivis pour les atteindre. Ceci, tout en gardant la traçabilité entre les objectifs métiers, les changements possibles du métier et les différentes parties du système informatique y impliquées. Ainsi, on pourrait prédire les changements potentiels qui peuvent impacter le système informatique et lui assurer une meilleure agilité. L'approche proposée est composé de quatre étapes :

1. Modélisation des scénarios : l'objective de cette étape est d'identifier les besoins et les scénarios pour le système en cours de développement afin de répondre aux exigences métiers de l'entreprise (Figure 4.1).
2. La modélisation de scénarios variable : Les changements potentiels pour les systèmes orientés services sont analysés à partir de scénarios dans le modèle des scénarios afin de garantir l'agilité des systèmes de l'entreprise.
3. Identification des services à partir du modèle de scénario variable : L'idée de base est que les exigences et les scénarios à différents niveaux sont des bonnes entrées pour identifier des services.
4. Conversion des services identifiés en profil de service : Cette étape consiste à modifier les services identifiés en un profil de service pour la mise en œuvre des services en utilisant des technologies spécifiques. Une fois la conversion terminée, les outils tels que les outils de support de MDA peuvent aider les développeurs à construire des systèmes SOA basée sur le profil généré.

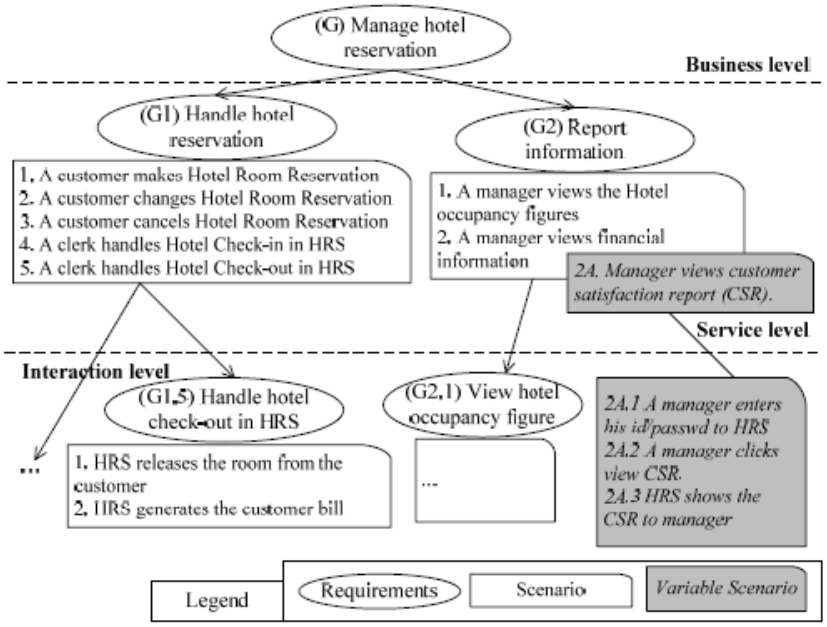


Figure 4.1 Modélisation d'un scénario [Suntae et al, 2008]

Une des solutions proposées consiste à faire appel au concept d'ontologie pour représenter, organiser et raisonner à travers un ensemble complexe de connaissances métiers [DongSu et al, 2008]. Ils ont utilisé un Feature model, Les feature model se présentent comme des arbres dans lesquels chaque nœud est une caractéristique et chaque arête peut avoir quatre valeurs possibles :

- Obligatoire : Si le nœud parent est présent, alors le nœud enfant aussi.
- Optionnel : Le nœud enfant peut ou peut ne pas être présent alors que le nœud parent l'est.
- Alternative : Si le nœud parent est présent, alors un seul des nœuds enfants reliés par une arête alternative l'est.

- Ou : Si le nœud parent est présent, alors au moins un des nœuds enfants reliés par une arête « ou » l'est.

Approche proposé par [DongSu et al, 2008] est composé de trois activités (Figure 4.2) :

Activité 1. Identifiez les nœuds qui contiennent les concepts et fonctionnalités dans le feature model. Les fonctions sont regroupées en utilisant la définition des fonctionnalités de regroupement. Enfin, le candidat de service (SC) est créé comme l'artefact de son activité.

Activité 2. Raffinement des candidats de service par rapport au couplage (faible ou fort), les services sont optimisé en granularité appropriée à la suite de cette activité.

Activité 3. Évaluer les services identifiés par le niveau de réutilisation en ligne de produits logiciels. Dans cette activité, le service est comparé avec les autres services en termes d'interopérabilité.

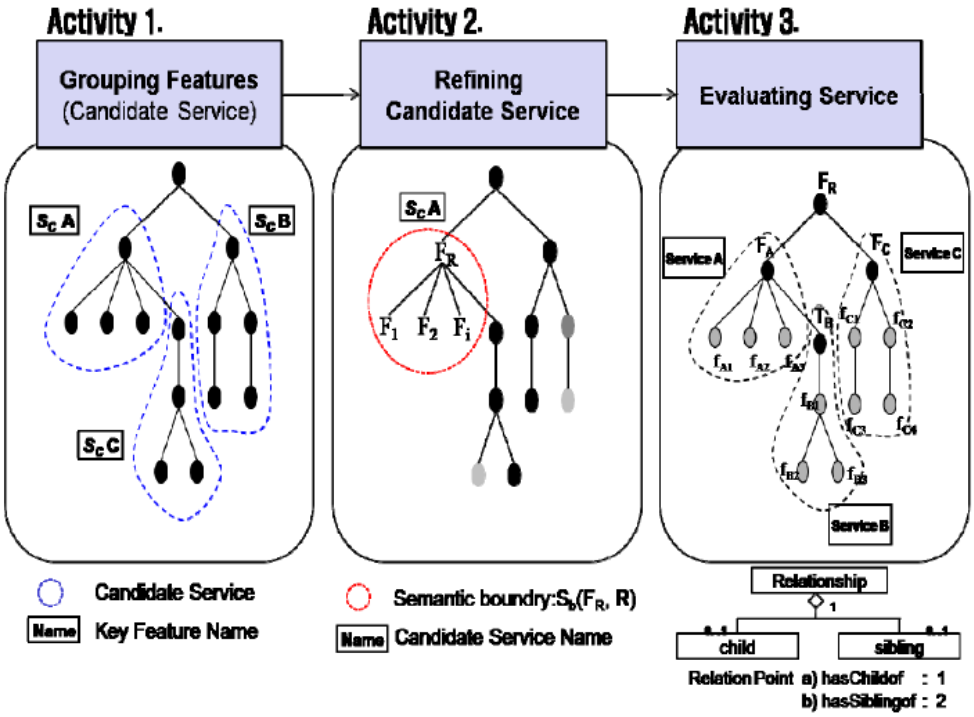


Figure 4.2 Processus d'identification [DongSu et al, 2008]

[Yousef et al, 2009], proposent une méthode pour la génération automatique des services à partir des processus métiers, en utilisant les ontologies ils ont développé le framework BPAOntoSOA (Figure 4.3).

Les étapes suivantes décrivent le processus d'identification des services selon BPAOntoSOA:

1. Associez chaque pool dans un modèle de processus BPMN à une ligne d'activité en BPAOnt.
2. Associez chaque tâche dans le modèle BPMN à une fonction atomique dans BPAOnt.
3. Identifier les fonctions appropries qui peuvent être automatisées en fonction de l'ontologie BPAOnt.
4. Étiqueter chacune des fonctions identifiées à la ligne d'activité auquel il appartient.
5. Spécifiez les pairs pools qui interagissent, et pour chaque paire, spécifier la tâche principale qui démarre le flux de messages.
6. Éliminer les fonctions redondantes;
7. Relier les résultats de fonction pour tous les EBE dans le BPAOnt selon les fonctions CRUD.
8. Créer la matrice CRUD et appliquer la technique clustering pour le regroupement des fonctions dans les services afin d'améliorer l'exactitude de services par apport à la cartographie des services d'entreprises aux services logiciels associés.

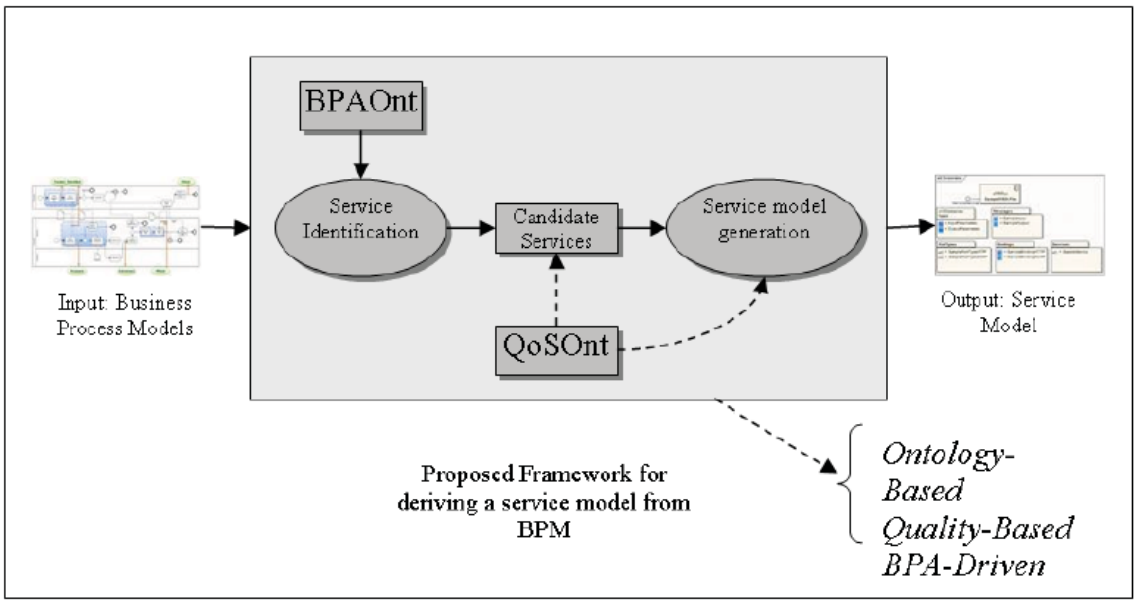


Figure 4.3 BPAOntoSOA Framework [Yousef et al, 2009]

Une autre solution consiste à stocker les informations liées aux processus métiers sous un format exploitable et bien structuré, tel que XML, qui est facile à analyser par n'importe quel outil informatique. Une telle solution est adoptée par [Dwivedi et Kulkarni 2008] qui utilisent, dans leur approche d'identification de service, des cartographies de processus métiers à base de UML, représentées en XMI (XML Metadata Interchange). XMI étant un format d'échanges d'informations de métadonnées UML basé sur XML. Ce format de persistance (XML) des informations métiers a permis aux auteurs de cette approche d'exploiter automatiquement les processus métiers et d'identifier automatiquement les services par l'utilisation des heuristiques spécifiques (Figure 4.4).

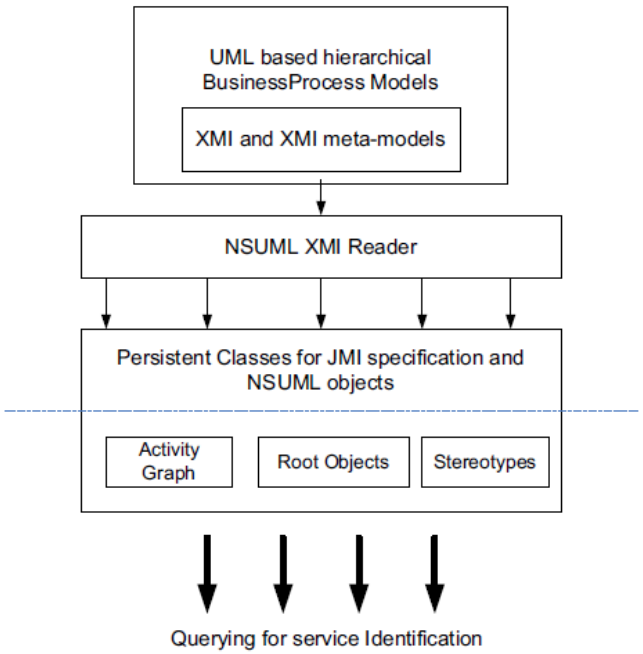


Figure 4.4 Approche SQUID [Dwivedi et Kulkarni 2008]

En plus de la non-automatisation, la majorité des approches d'identification de services sont non systématiques, ce qui rend difficile leur application dans le monde réel. [Arsanjani et al, 2008] dans SOMA; [Boerner et Goeken 2009]; [Erradi et al, 2006]; [Huysmans et al, 2007]; [Klose et al, 2007]; [Papazoglou , van den Heuvel 2006]; présentent un ensemble de lignes directrices et de bonnes pratiques à suivre, cependant ils ne décrivent pas réellement comment l'identification des services doit être menée. Alors que [Azevedo et al, 2009], proposent une approche systématique basée sur un ensemble d'heuristiques bien détaillées à suivre dans le processus d'identification des services, et ce, en se basant sur une analyse :

- sémantique des éléments de processus (règles métiers, besoins métiers);
- syntaxique des modèles de processus (structural pattern).

La méthode est constituée de trois phases (**Figure 4.5**) :

Phase 1. Sélection des activités, un ensemble d'activités sont sélectionné du modèle de processus métier. Une activité de processus est sélectionnée si elle est soit automatique (réalisée entièrement par un système sans intervention manuelle), ou en partie supportés par des systèmes ou automatisables (exécuté manuellement, mais qui devrait être pris en charge par un système). Les activités manuelles ne sont pas sélectionnées.

Phase 2. Identification et classification des services candidats, les services sont identifiés par l'application d'un ensemble d'heuristiques à l'ensemble des activités choisies dans la phase 1.

Phase 3. Consolidation des Services, les informations sur les services est consolidées. Consolidation des services vise à rassembler plusieurs caractéristiques sur chaque service afin de soutenir le concepteur de service pour décider de la mise en œuvre.

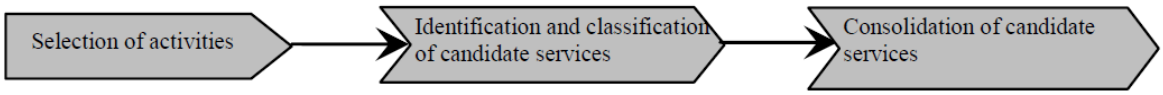


Figure 4.5 La méthode d'identification de [Azevedo et al, 2009]

Afin de pouvoir exploiter le plein potentiel de SOA, certaines démarches d'identification de services adoptent une gouvernance SOA [Boerner et Goeken 2009; Dan et al, 2008; Strnadl, 2007], cette gouvernance a pour objectif de traiter les problèmes liés à la mise en œuvre d'une SOA, et ce, en assurant une bonne intégration des ressources humaines, des processus et des données. En effet, une gouvernance SOA permet de :

1. éviter la multiplication incontrôlée des services ;
2. gérer le cycle de vie de ces services ;
3. assurer leur alignement avec la stratégie de l'entreprise.

Il existe dans le marché plusieurs outils, dits de gouvernance SOA, telle que IBM Rational Method Composer, Oracle SOA Governance 11g, etc. Ces outils permettent de faciliter une telle gouvernance et appliquer ses règles.

[Jamshidi et al, 2008] ils ont formulé le problème comme étant un problème d'optimalisation multi objectifs. Ils ont résolue le problème avec un nouvel algorithme de clustering. Elementary business process and business Entity Affinity analysis Technique (EEAT) sont utilisés pour traiter une matrice CRUD (Figure 4.6). Bien qu'ils affirment que leur processus proposé a la capacité d'automatisation basée sur orienté modèle et ont mis l'accent sur l'adoption de mesures techniques, ils ne pouvaient pas définir formellement le problème et de proposer une méthode automatisée basée sur des principes de conception de services quantitatifs.

[Jamshidi et al, 2012] Présentent une nouvelle méthode ASIM (Automated Service Identification Method) pour identifie et spécifié les services de niveau entreprise à partir du processus métier. Ils ont formulé l'identification de service come un problème d'optimalisation multi objectifs et ils ont résolue en utilisant un algorithme d'optimalisation méta heuristique (Recuit simulé) qui dérive les services appropriés par la technique de clustering en utilisant des mesures quantitatives pour granularité, le couplage, la cohésion, la réutilisabilité et la maintenabilité. Cependant, l'algorithme qui est utilisé pour le clustering n'est pas rapide.

[Kazemi et al, 2011] ont présenté une méthode automatisée d'identification des services aux entreprises en adoptant des mesures de conception basée sur la décomposition top-down des processus. Cette méthode prend un ensemble de processus métiers de

l'entreprise en entrée et produit un ensemble de solutions non dominées représentant les services appropriés en utilisant un algorithme génétique multi-objectif.

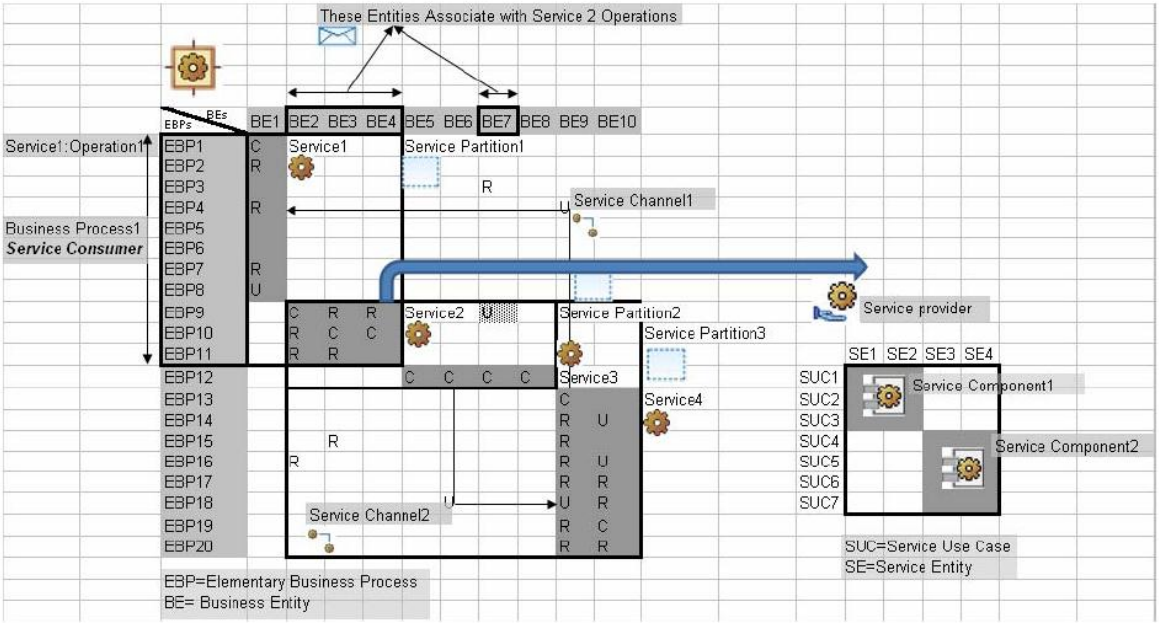


Figure 4.6 La matrice CRUD partitionnée [Jamshidi et al, 2008]

Trois phases composent la méthode proposée par [Kazemi et al, 2011] :

- Phase de définition des Métriques : Dans cette phase, un ensemble d'objectifs de l'entreprise sont prises et un ensemble appropriés de métriques sont définis ou choisis. Parvenir à un ensemble de métriques appropriées pour répondre aux objectifs de l'entreprise est le but majeur de cette phase.
- Phase d'identification d'ensemble de services : Le but ultime de cette phase est l'identification automatique de l'ensemble des services. Modèles de processus métiers ainsi que les entités métiers et les mesures définies dans la première phase constituent les entrées de cette phase. Dans cette phase ils utilisent l'algorithme génétique pour résoudre le problème multi objectif.
- Phase de sélection d'ensemble de service approprié : Dans cette phase, selon l'importance des attributs de qualité, un ensemble de règles floues avec leurs fonctions d'appartenance sont définies. Le classement des solutions Pareto est l'activité suivante qui est faite juste après la logique floue.

[Jain et al, 2004] ont fourni une méthode pour l'identification des services Web MOGA WSI. La méthode, qui a été la première tentative pour automatiser l'identification de service Web, prend un modèle d'analyse en entrée et dérive les services Web candidats. Un groupement initial hiérarchique des classes est dérivé en utilisant l'algorithme spanning tree. Puis par l'adoption d'un algorithme génétique, une solution plus souhaitable peut être sélectionnée. Cependant, afin d'utiliser cette méthode dans le cycle de vie, des efforts doivent être fournis pour la construction du modèle d'analyse et rendre le MOGA-WSI moins pratique. [Zhang et al, 2008] ont mis en place un environnement à savoir

SOMA-Modélisation de l'environnement (SOMA-ME) qui agit comme un framework pour la conception axée sur les modèles SOA en utilisant la méthode SOMA. Dans ce cadre, l'identification des services a été automatisée avec MOGA-WSI.

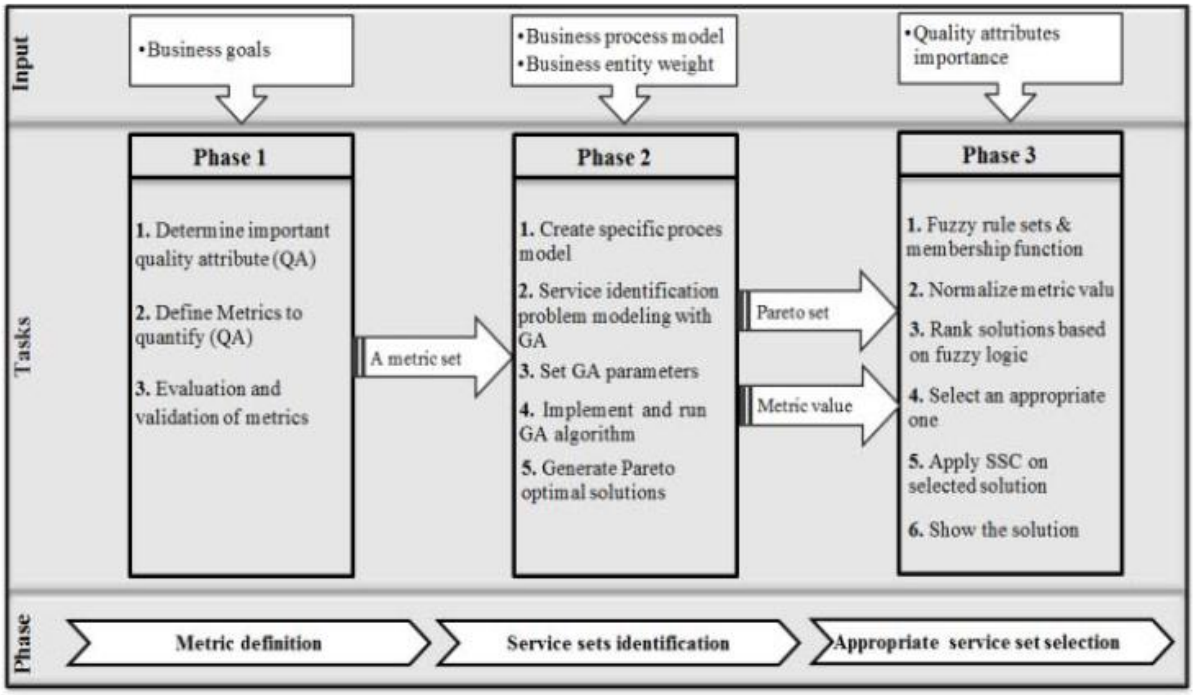


Figure 4.7 La méthode proposée par [Kazemi et al, 2011]

[Strosnider et al, 2008] ont introduit une nouvelle approche en utilisant model-driven business transformation (MDBT) qui est basé sur l'architecture dirigée par les modèles pour la génération semi-automatique des services à partir des processus métiers, ils ont présenté business entity life cycle analysis (BELA) une technique pour MDBT qui est basé sur SOA.

[Gacitua-Decar et al, 2009] ont présenté une approche globale à l'identification des services basé sur les processus métiers. Ils ont introduit une solution pour identifier les modèles BP basés sur un mécanisme de matching de graphes. Les aspects structurels et la sémantique, y compris le traitement du langage naturel, sont abordés. Cette méthode est semi automatisée et ne considèrent pas les mesures des principes de conception des services. Par conséquent, les services qui en découlent ne peuvent pas être évalués en fonction de critères quantitatifs et comparables.

[Birkmeier et al, 2013] proposent une méthode systématique pour l'extraction automatique des services à partir d'un modèle de processus métier pour le gouvernement fédérale Allemand, en utilisant des techniques de conception à base d'heuristiques. L'approche de [Birkmeier et al, 2013] consiste de plusieurs étapes (**Figure 4.8**):

- Dans la première phase, l'analyste métier identifie les activités du processus métier qui devrait être automatisé.

- Pendant la deuxième phase, l'architecte aide l'analyste métiers avec son expérience pour consolider la liste initiale des fonctionnalités.
- L'objectif de la troisième phase consiste à identifier les services existants réutilisables et les nouveaux services qui fournissent conjointement les fonctionnalités requises. Durant cette phase les auteurs utilisent des heuristiques pour automatiser le processus.
- Dans la quatrième phase, l'architecte doit aligner la liste des services requis avec le catalogue de services.
- La cinquième phase se concentre sur l'alignement avec l'architecture d'entreprise.
- Lors de la sixième phase, l'architecte classifie les nouvelles fonctionnalités et services par rapport à leur pertinence pour l'organisation générale.

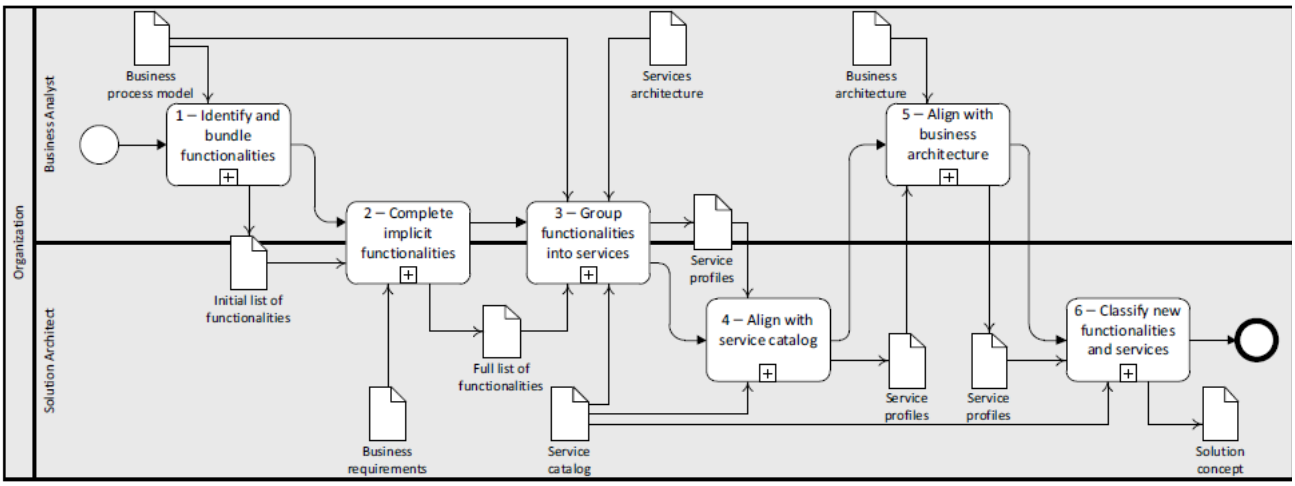


Figure 4.8 Processus Global de [Birkmeier et al, 2013]

[Bianchini et al, 2013] proposent la méthode P2S (Process-to-Services), une méthode semi-automatique pour permettre l'identification des services qui composent les processus métiers collaboratifs. La méthode est basée sur des métriques (granularité, cohésion, couplage, réutilisation). La méthode est constituée de trois étapes :

- L'analyse des processus métier : Dans cette phase, les descripteurs de tâches, les dépendances et les dépendances de flux de données sont exploitées pour analyser la structure des processus métier.
- L'identification des services candidats : Dans cette phase, les résultats de l'analyse précédente sont utilisées pour identifier les services qui composent le processus métier.
- La réconciliation des services : dans cette dernière phase, les auteurs utilisent des ontologies pour mesurer la distance de similarité entre les services identifies.

L'adoption d'une approche descendante, pour l'identification des services, permet d'avoir non seulement une architecture orientée services complètement alignée aux besoins métiers de l'entreprise, mais aussi d'éviter la redondance de services. Cependant, cette démarche n'est pas épargnée de certains travers que certains essayent de les esquiver en

complétant les approches Top-Down par d'autres techniques. [Senthil et al, 2008], par exemple utilisent les interfaces utilisateurs et les informations qu'elles contiennent pour optimiser le processus d'identification de services.

Malgré les avantages qu'elle apporte, la démarche descendante reste souvent très couteuse et trop risquée, puisqu'elle nécessite une refonte complète ou partielle du SI de l'entreprise qui l'adopte.

2.2 Approche ascendante (Bottom up)

L'approche ascendante est réalisée principalement par des analystes système, qui ont l'expertise et les compétences nécessaires qui leurs permettent de [Iheb, 2011] :

- analyser la plateforme applicative du SI actuel,
- desceller la logique métier et les fonctionnalités redondantes dans le SI,
- et les exposer sous forme de services, qui peuvent ou non être composés afin de participer à la construction des processus d'affaires.

Ainsi, la migration vers SOA se fera sans la négligence du vaste héritage logiciel existant et applicatif développer par l'entreprise depuis sa création, ce qui réduit énormément le coût pour une migration souple.

Une approche ascendante commence par une évaluation rigoureuse de la plateforme technique de l'entreprise [Reddy et al, 2009] et la réalisation d'une cartographie complète de l'architecture applicative, qui décrit les différentes fonctions offertes par le SI actuel. Dans cette cartographie, les logiciels et les programmes du SI peuvent être classés selon plusieurs critères en utilisant des métriques comme la cohésion ou le couplage. [Sneed 2006], dans son approche ascendante d'identification de service par l'implémentation d'une sous-couche pour encapsuler les applications existantes et les proposer sous forme de service.

[Sneed, 2006] propose aussi trois phases pour procéder à la migration du SI actuel vers une architecture orientée services : La première phase consiste à analyser le code par des technique de rétro-ingénierie (reverse engineering) et ré-implémenter le code dans un autre langage. La deuxième phase consiste à envelopper le code exécutable et y accéder via une interface. La troisième et dernière phase consiste à transformer le code source pour qu'il puisse être traduit en service **(Figure 4.9)**.

[Komondoor et al, 2012] proposent une nouvelle solution basée sur l'analyse statique pour résoudre le problème de l'identification des services au sein des programmes de traitement des transactions. Ils proposent une caractérisation formelle des services en termes de propriétés de flux de données et de flux de contrôle, ce qui est bien adapté pour les idiomes généralement exhibés par les applications d'entreprise. Leur technique combine le découpage des programmes avec la détection des régions de code conditionnel pour identifier les services conformément à une caractérisation formelle.

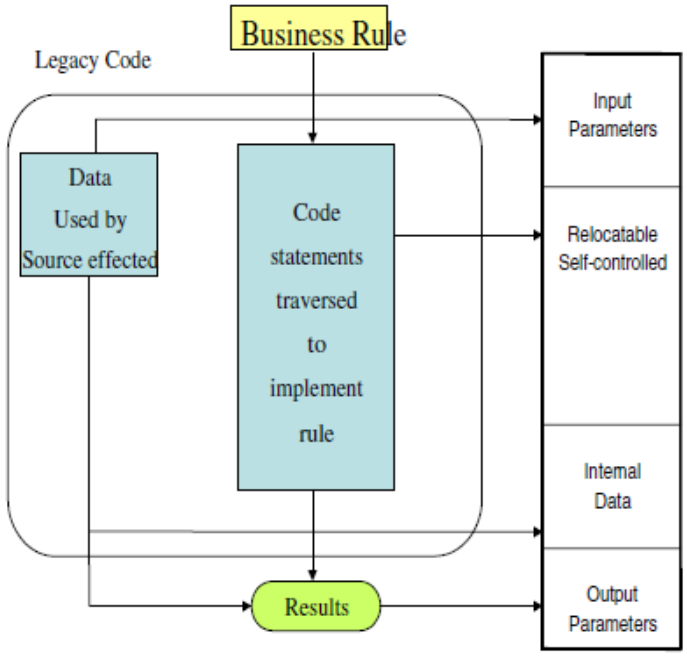


Figure 4.9 Approche de [Sneed 2006]

[Feng et al, 2005] introduisent une approche basée sur l'analyse des fonctionnalités des applications (**Figure 4.10**)

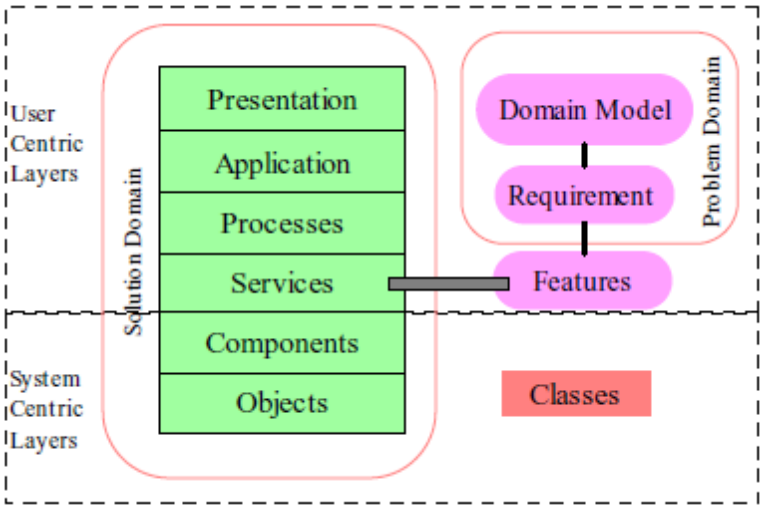


Figure 4.10 les fonctionnalités dans l'architecture orientée service [Feng et al, 2005]

Le choix des fonctionnalités comme un intermédiaire dans toute la réingénierie orientée services dépend des critères suivants :

- Granularité
- Traçabilité
- Sémantique
- Interaction entre entité

Grâce à l'analyse de fonctionnalité, les services peuvent être identifiés à partir des systèmes existants pour une implémentation SOA (**Figure 4.11**).

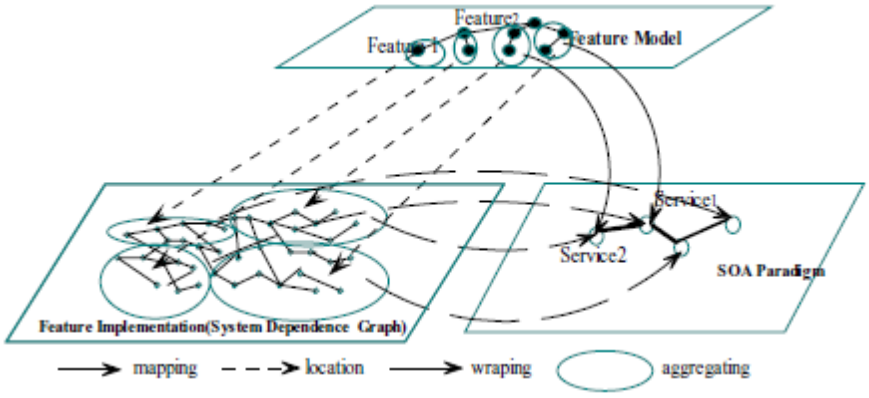


Figure 4.11 Les Relations entre les fonctionnalités et les services [Feng et al, 2005]

[Nakamura et al, 2009] présentent une méthode pour dérivation automatique des services à partir des programmes procédural. Ils supposent que chaque procédure représente un service. L'identification se base sur l'analyse du code source ainsi que les flux de données.

L'adoption d'une approche ascendante dans la mise en place d'une SOA n'assure pas systématiquement l'alignement du métier avec le SI. En effet, cette approche présuppose que le SI actuel est, d'ores et déjà, aligné avec le métier de l'entreprise et répond aux besoins de cette dernière, ce qui n'est pas toujours le cas. En plus, en suivant cette approche, les bénéfices de mise en place d'une SOA ne seront perçus, qu'une fois les services auront été rendus exploitables au sein des processus métiers, ce qui rend très difficile la justification de l'investissement auprès des décideurs [Heubès, 2007].

2.3 Approche mixte

Les approches mentionnées précédemment (Top-down et Buttom-up) peuvent être combinées dans une approche intitulée « mixte ». Cette approche préconise de mener à la fois une analyse ascendante et une analyse descendante et faire une corrélation entre les services identifiés par chacune de ces approches [Iheb ,2011].

Une approche mixte d'identification de service peut être résumée en quatre étapes [Srikanth et al, 2007] :

- Dans une première étape, les analystes du métier commencent par analyser les processus métiers de l'entreprise et y recenser les activités. Cette étape nécessite d'avoir des processus métiers documentés, offrant un certain niveau de détail.
- Durant la deuxième étape, menée en parallèle avec la première, les analystes commencent à analyser les applications au niveau du SI et identifier les fonctionnalités qui peuvent être transformées en services.
- Pendant la troisième étape, les activités identifiées au niveau métier et les services recensés au niveau du système existant, vont être confrontés afin de pouvoir trouver un lien de corrélation entre eux. Ce lien peut être un lien direct ou un lien

de composition, c'est-à-dire, un service identifié à partir du système existant peut être directement associé à une activité métier ou dans le cas contraire, combiné avec un ou plusieurs autres services, recensé à partir de la couche des programme existant, afin de pouvoir bien mener l'activité du processus métiers.

- La quatrième et dernière étape consiste à raffiner les processus métier et définir les sous-processus, ceci afin d'essayer d'atteindre, idéalement, un niveau de corrélation un à un entre les services et les activités métiers.

La figure 4.12 schématise l'architecture globale de l'approche de [Srikanth et al, 2007].

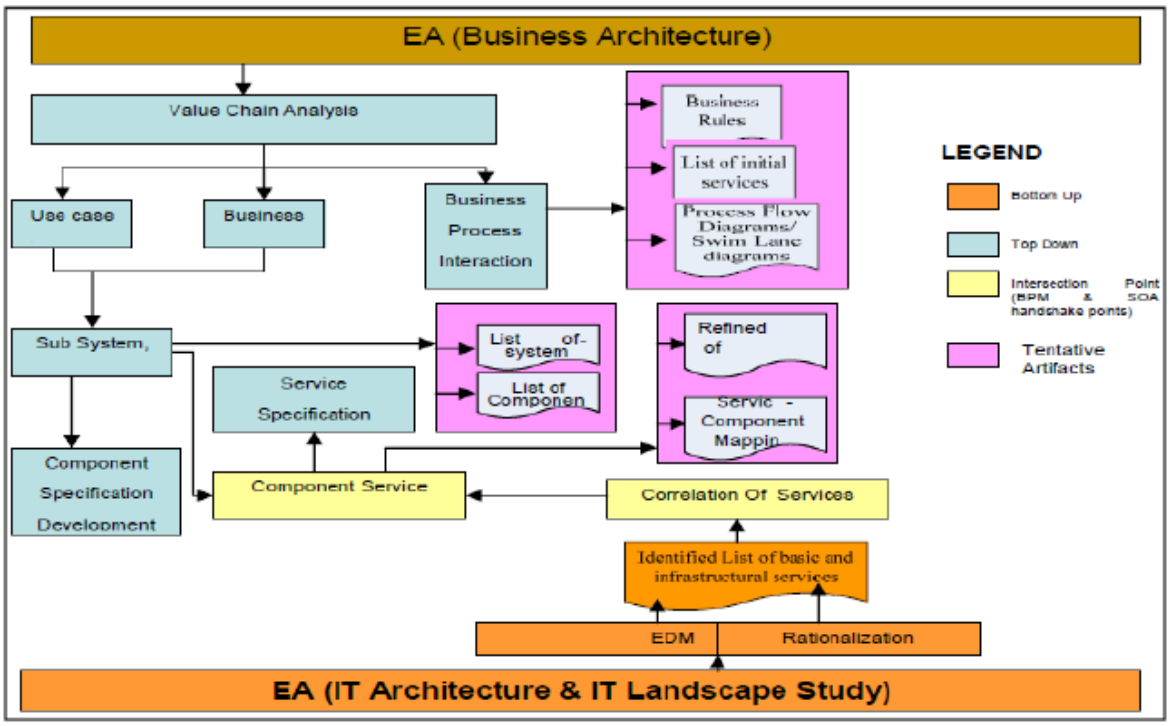


Figure 4.12 L'approche d'identification de services de [Srikanth et al, 2007]

[Nikravesh et al, 2011] ont proposé une approche semi-automatique appelée 2PSIM en utilisant un algorithme pour le partitionnement des graphes pour l'identification des services à partir des processus métier et les entités métier. 2PSIM utilise la stratégie middle-out pour identifier les services réutilisables avec une granularité adéquate et un niveau acceptable de cohésion et de couplage (**Figure 4.13**).

L'adoption d'une approche mixte permet de combiner les avantages de l'approche ascendante et descendante, cependant, cela ne veut pas dire que cette combinaison va éviter les contraintes de ces deux approches. En effet, la recherche d'un lien de corrélation entre les activités métiers et les services identifiés au niveau SI présente une tâche très critique. Elle nécessite non seulement la compréhension du métier de l'entreprise, mais aussi la maîtrise de son SI. Dans une approche mixte, la réussite ou non de la mise en œuvre d'une SOA dépend principalement de cette étape de corrélation.

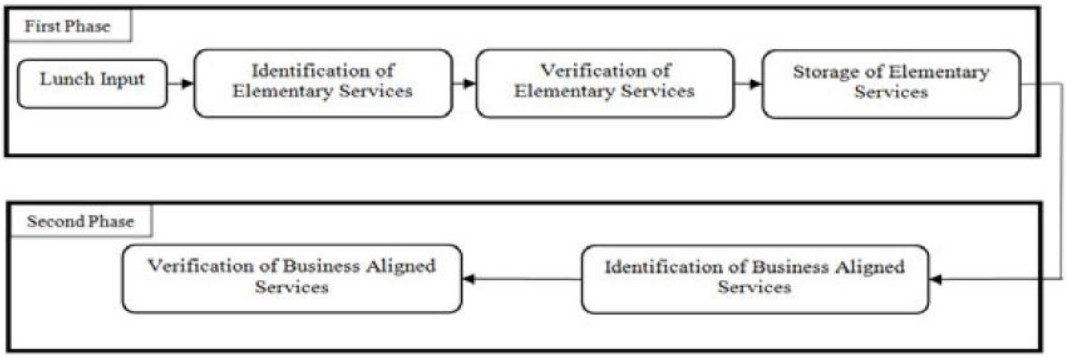


Figure 4.13 Le processus de 2PSIM [Nikravesh et al, 2011]

Le tableau ci-dessous présente une synthèse sur les trois approches d'identification de services :

Approches	Inputs d'analyse	Services identifiés	Avantages	Inconvénients
Descendante	- Processus Métiers. - Domaines et objectifs métiers.	Métiers	- Alignement SI / Métier. - Éviter la redondance des services.	- Coûteuse - Risquée
Ascendante	Applications informatiques du SI actuel.	Applications	- Coût réduit. - Cartographie du SI.	- Alignement SI/Métier non garanti. - Investissement non justifié au départ.
Mixte	Couche applicative et métier.	-Métiers - Applications	Combinaison des avantages des approches descendantes et ascendantes	Combinaison des inconvénients des approches descendantes et ascendantes

Tableau 4.1 : Synthèse des approches d'identification de services [Iheb ,2011]

2.4 Comparaison des approches d'identification de services

Après avoir fait le tour des différentes méthodes d'identification de services trouvées dans la littérature, nous remarquons que les approches les plus répondues et les travaux les plus récents sont ceux basées sur l'analyse des processus métiers de l'entreprise. Cependant, les approches telles que l'approche mixte et l'approche descendante sont le plus souvent prescriptives et exigent une intervention manuelle humaine dans l'analyse des processus métiers. La non-automatisation de ces approches est due principalement à la nature des informations liées aux processus métiers. En effet, dans une entreprise, on trouve généralement la modélisation concernant les processus métier, sous forme de graphiques ou des rapports écrits en langage naturel. Ces formes d'information ne facilitent pas le traitement automatique de ces processus métiers et rendent la participation humaine nécessaire, surtout dans la phase d'analyse métier.

Afin de comparer les différentes approches existantes, nous avons proposé les critères d'évaluations suivant en se basant sur les travaux de [Jamshidi et al, 2012] :

Degré d'automatisation : Une méthode d'identification de service peut être une technique, une procédure prescriptive, une méthode semi-automatique avec un outil de case, un algorithme sans outil, ou totalement automatisés.

Métriques techniques employées : les mesures techniques qui influence les décisions architecturales afin d'aboutir à une conception efficace des services dans le contexte SOA.

L'utilisation de l'approche dirigée par les modèles : Les approches Model Driven utilisent des notations abstraites pour créer des modèles qui expriment le modèles source et cible.

Démarche d'identification : l'Approche sélectionné, qui est utilisé pour l'identification de service. Différentes approches peuvent être adoptées pour l'identification de service, à savoir de haut en bas (décomposition de domaine), bottom-up (analyse du système existant) et approche mixte (combinaison de l'approche ascendante et descendante).

Technique d'identification : Différentes techniques peuvent être utilisées pour identifier les services des processus métiers telles que le clustering des matrice et des graphes ou d'autres techniques d'analyse.

Applicabilité : Les échelles d'application de méthodes varient d'une entreprise à une société de petite échelle selon les techniques mises en œuvre.

Réutilisation de la méthode : Si une méthode est indépendante de son contexte, elle peut être réutilisée dans un autre contexte. Toutefois, certaines méthodes ont plusieurs restrictions et ne peuvent être réutilisé dans un autre contexte.

Degré de formalisme : Description des méthodes rigoureuses avec des fondations mathématiques, (semi-) formelle avec des modèles graphiques, informelle avec des explications textuelles

Modèle d'entrée : Méthodes d'identification de service doit faire appel à un modèle prétraitée en tant que source d'information pour identifier les services.

Critères / Méthodes	Degré d'automatisation	Métriques techniques employées	L'utilisation de l'approche dirigée par les modèles	Démarche d'identification	Technique d'identification	Applicabilité	Réutilisation de la méthode	Degré de formalisme	Modèle d'entrée
[Bianchini et al, 2013]	Semi-automatique avec des algorithmes pour assister l'utilisateur	granularité, cohésion, couplage, réutilisation.	Entièrement basé sur les modèles	Top down	Value analysis et ontologie	Niveau de l'entreprise	réutilisable	Rigoureuse avec fondements mathématiques	CRUD Matrix
[Birkmeier et al, 2013]	Semi - Automatique (basée sur un framework)	granularité, couplage, réutilisation	Entièrement basé sur les modèles	Top down	Heuristique	Niveau de l'entreprise	réutilisable	Semi-formel	BPMN Modèle
ASIM [Jamshidi et al, 2012]	Algorithme et outils case	granularité, couplage, la cohésion, la réutilisabilité et la maintenabilité	Entièrement basé sur les modèles	Top down	Matrix Clustering(recuit simulée)	Niveau de l'entreprise	réutilisable	Rigoureuse avec fondements mathématiques	CRUD matrix
[Nikravesh et al, 2011]	Semi-automatique	Granularité, couplage et cohésion.	Partiellement basé sur les modèles	Mixte	Partitionnement des graphs	Petite à moyenne échelles	réutilisable	Semi Formelle	Processus Métier et le code source
[Ali Kazemi et al, 2011]	Algorithme	granularité, couplage, la cohésion, Convergence entité	Entièrement basé sur les modèles	Top down	Graph Clustering (algorithme génétique)	Niveau Entreprise	réutilisable	Rigoureuse avec fondements mathématiques	Matrice représentant le graphe
[Azevedo et al, 2009]	Algorithme et outils case	Réutilisation, couplage et cohésion.	Entièrement basé sur les modèles	Top Down	heuristique	Niveau Entreprise	réutilisable	Semi Formelle	EPC Modèle
[Yousef et al, 2009]	Outils case et algorithme	Réutilisation, couplage et cohésion.	Entièrement basé sur les modèles	Top down	EEATclustering et Ontologie	Niveau de l'entreprise	réutilisable	Semi Formel	CRUD Matrix
[Gacitua Decar et al, 2009]	Algorithme (basé sur graphe)	aucune métrique quantitative	Entièrement basé sur les modèles	Top down	Pattern matching and discovery	Niveau de l'entreprise	La Réutilisation de cette méthode a de nombreuses contraintes	Rigoureuse avec fondements mathématiques	Business Process Graph
[Suntae et al, 2008]	outils case	Granularité	Entièrement basé sur les modèles	Top down	Goal-Scenario Modeling	Niveau Entreprise	réutilisable	Semi-formelle	Goal-scenario model

[Strosnider et al, 2008]	Basé sur humain, outils case(Semi-automatique)	aucune métrique quantitative	Entièrement basé sur les modèles	Top down	State Machine Analysis	Niveau de l'entreprise	réutilisable	Semi-formelle avec modèle de graphe	Modèle de données
[DongSu et al, 2008]	Outils case	Couplage	Entièrement basé sur les modèles	Top down	Modeling Feature et Ontologie	Niveau de l'entreprise	réutilisable	Rigoureuse avec fondements mathématiques	Feature model
[Jamshidi et al, 2008]	Outils case	Réutilisation, maintenabilité et granularité.	Entièrement basé sur les modèles	Top down	Clustering avec EEAT	Niveau de l'entreprise	réutilisable	Semi-formelle	CRUD Matrice
[Dwivedi et Kulkarni 2008]	Outils case et algorithme	Réutilisation, couplage, compossibilité et autonomie.	Entièrement basé sur les modèles	Top down	Heuristique	Niveau de l'entreprise	réutilisable	Semi Formel	Modèle UML(XMI)
[Srikanth et al, 2007]	Procédure manuelle	aucune métrique quantitative	Partiellement basé sur les modèles	Mixte	Prescriptive	Petite à moyenne échelles	Une technique qui impose de nombreuses restrictions	Informel en général des procédures textuelles	Processus Métier et le code source
[Sneed, 2006]	Outils case et algorithme	aucune métrique quantitative	Partiellement basé sur les modèles	Bottom UP	Wrapping	Petite à moyenne échelles	Une technique qui impose de nombreuses restrictions	Semi Formelle	Code source
[Feng et al, 2005]	Outils	Granularité	Entièrement basé sur les modèles	Bottom UP	Wrapping avec Analyse des fonctions	Petite à moyenne échelles	Une technique qui impose de nombreuses restrictions	Semi Formelle	Code source
[Jain et al, 2004]	Algorithme et outils case	cohésion, coûts de développement, facilité de montage, personnalisation, réutilisabilité et la maintenabilité	Entièrement basé sur les modèles	Top down	Graph Clustering (algorithme génétique)	petite échelle	réutilisable	Rigoureuse avec fondements mathématiques	Modèle objet

Tableau 4.2 : Étude comparative des approches d'identification de services

3. Les approches de spécification des services à base MDA

[Kalantari et al, 2011] ont classé les approches de développement des services dirigé par les modèles en trois catégories à savoir : approches à base de méthodologie logiciel, approches à base de formalismes UML, et approches à base de méta-modèle.

3.1 Approches à base de méthodologie logicielle

Cette catégorie d'approches est basée sur les principes du génie logiciel. Elle fournit des solutions méthodologiques pour créer des annotations sémantiques pendant le développement de service. Les approches basées sur la méthodologie logicielle peuvent combiner les bons pratiques dans la spécification sémantique et le développement des services.

L'approche de [Torres et al, 2006] est une méthode d'ingénierie Web orientées objets utilisée pour développer les services Web sémantique basé sur l'ontologie OWL-S. Cette approche étendre un modèle d'ontologie avec la méthode d'ingénierie Web pour spécifier les fonctionnalités et les données du système qui vont être publiées vers les systèmes externes. En outre, dans cette approche les fonctionnalités et les opérations de service sont spécifiées par des modèles structuraux et comportementaux tels que diagramme de classe, diagramme d'activité, et diagramme d'état/transition. L'approche ne fournit pas le profil de service et elle devrait être créée à la main. Les constructions OWL et de OWL-S sont modelées par la représentation de l'ontologie de domaine qui est formée par le diagramme de classe, diagramme d'état/transition et diagramme d'activité.

L'approche MIDAS-S [Acuna et al, 2006] est une extension de l'approche MIDAS [Caceres et al, 2003]. MIDAS est un cadre méthodologique dirigée par les modèles pour le développement des systèmes d'informations sur le Web. MIDAS cherche à modéliser les SIW selon deux dimensions orthogonales:

D'abord, MIDAS spécifie le système entier par des (CIMs), des (PIMs) et des (PSMs). Deuxièmement, il modélise le système selon trois aspects de base : **hypertexte**, **contenu** et **comportement**. En outre, MIDAS suggère d'employer UML en tant que la notation unique pour modéliser les PIMs et les PSMs. La figure 4.14 présente le cadre MIDAS. MIDAS-S est une méthodologie de logiciel dirigée par les modèles utilisée pour développer les services Web sémantiques basé sur WSMO. Elle est basée sur le cadre MIDAS. L'approche MIDAS-S ajoute l'aspect sémantique dans le niveau PIM et PSM (cf. Figure 4.15 et Figure 4.16). Cet aspect est représenté par des diagrammes UML avec des spécifications de WSML. Par conséquent, le développement du service Web sémantique peut être intégré avec les autres aspects comme l'hypertexte, le contenu, et le comportement. La figure 4.15 montre le cadre MIDAS-S.

Dans cette approche, les profils d'UML emploient le langage OCL pour la représentation des axiomes logiques de WSMO. En fait, dans MIDAS-S, les axiomes sont représentés par le diagramme de classe d'UML avec la définition d'axiome comme valeur étiquetée. Une partie du code de WSML qui représente les axiomes est incluse dans la définition

d'axiome. Cette approche spécifie les modèles des services, les ontologies, les médiateurs, et les buts dans le niveau de PSM. Les buts, les Ontologies, et les modèles des services Web sont divisés en deux modèles appropriés : modèle de contexte et modèle de contenu. Les informations sur les espaces de noms, les médiateurs utilisés, et les ontologies importés sont rassemblés dans le modèle de contexte d'ontologie qui est représenté par le diagramme de paquetage d'UML. Le modèle de contenu d'ontologie est représenté par le diagramme de classe d'UML. En outre, les concepts, attributs, éléments d'axiomes sont représentés par le modèle de contenu d'ontologie. Les modèles de contenu et de contexte de service Web sont similaires aux modèles d'ontologie susmentionnés avec la différence que les modèles de contenus représentent les éléments du service Web tels que les capacités, les interfaces, etc. (cf. figure 4.17 et figure 4.18).

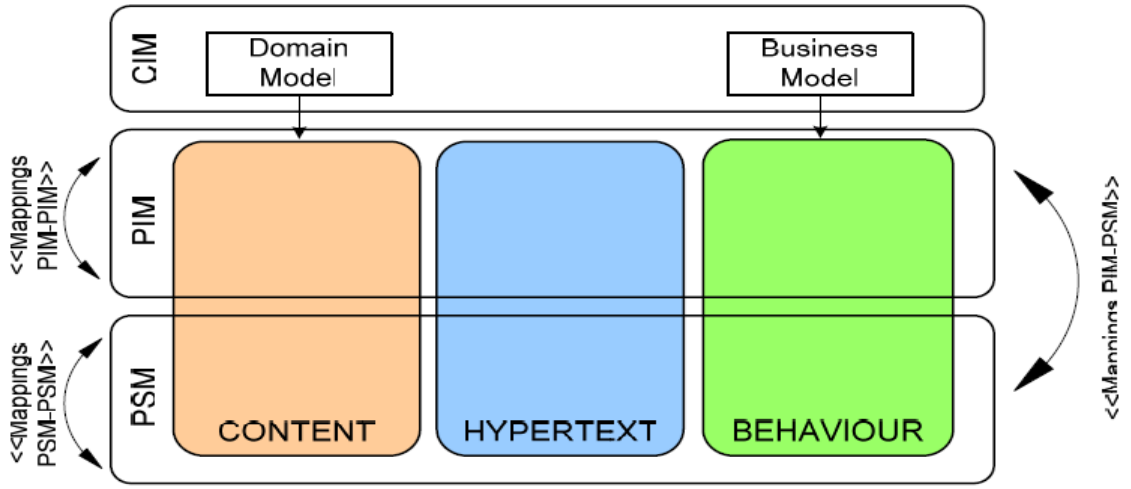


Figure 4.14 le cadre de spécification des services MIDAS [Caceres et al, 2003]

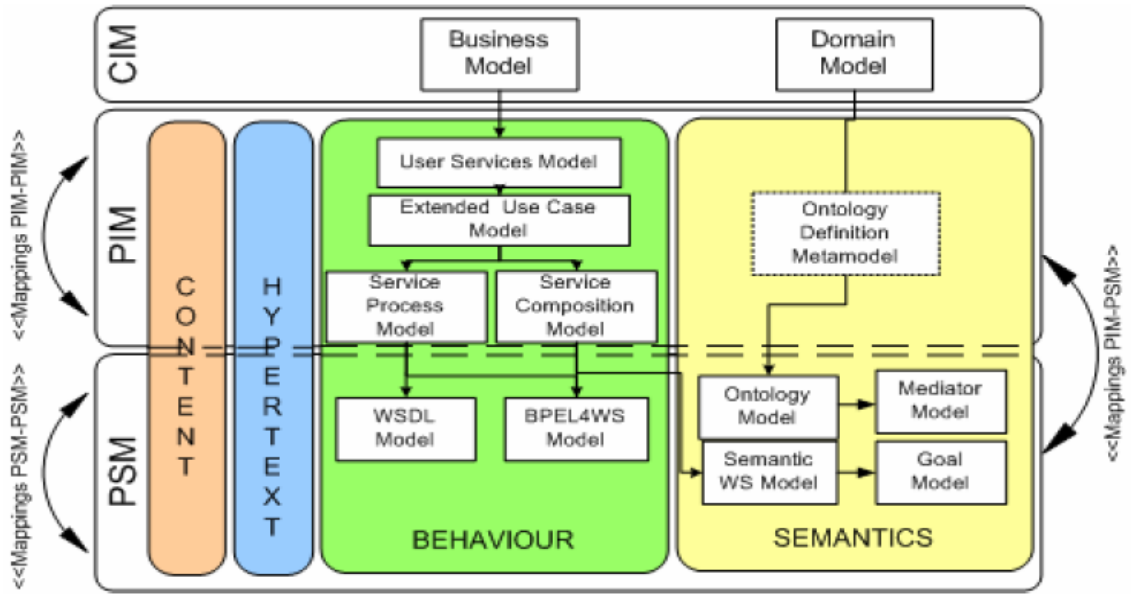


Figure 4.15 le cadre MIDAS-S [Acuna et al, 2006]

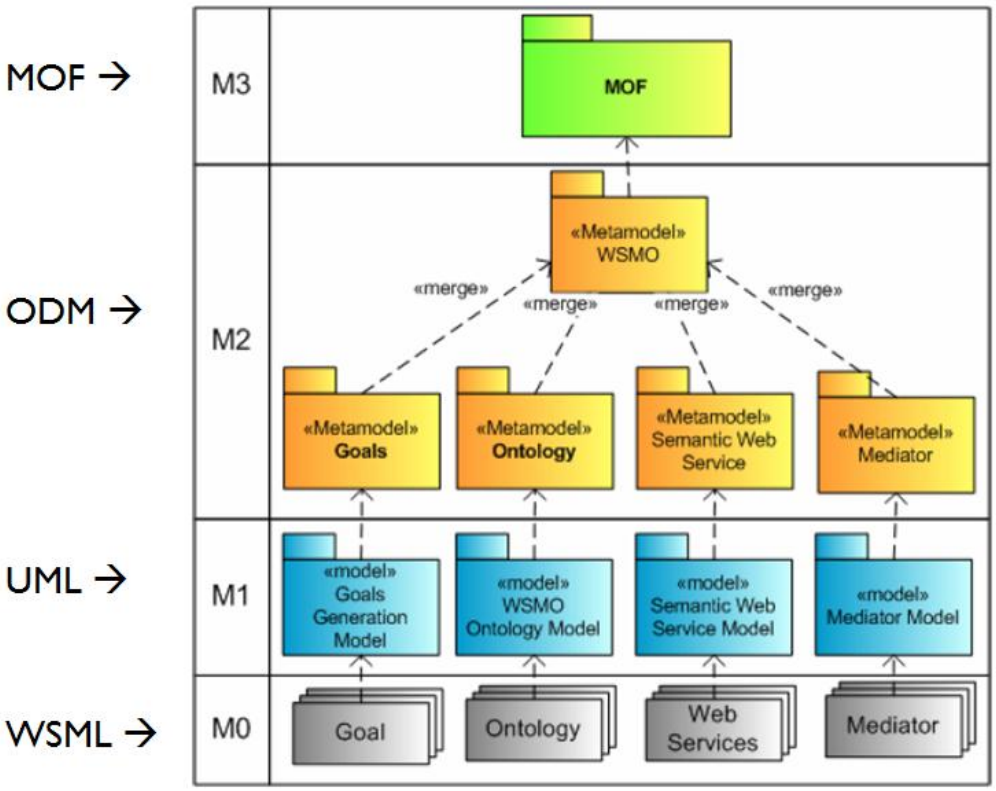


Figure 4.16 Projection de la spécification du WSMO sur les 4 niveaux MDA [Acuna et al, 2006]

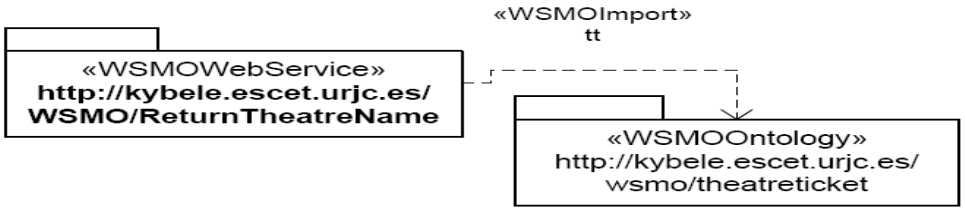


Figure 4.17 Modèle de contexte de Web service [Acuna et al, 2006]

L'approche de [Brambila et al, 2007] utilise Business Process Modeling Notation (BPMN) et Web Modeling Language (WebML) pour concevoir et développer les services Web sémantiques basé sur WSMO. Le but principal de cette approche est de diminuer la difficulté de fournir la description sémantique d'une application Web par l'utilisation des technologies sémantiques disponibles. Cette approche fournit un générateur semi-automatisé pour extraire des descriptions sémantiques de service à partir de la conception d'application. En outre, WSMX est employé en tant qu'environnement sémantique d'exécution. Dans cette approche des diagrammes WebML et BPMN représentés en une sérialisation XML sont créés par l'intermédiaire du langage de transformation XSLT. Puis, des descriptions sémantiques de service Web (WSMO) de l'application sont produites à partir de ces modèles par XSLT. En outre, cette approche utilise l'outil CASE

WebRatio [Webratio, 2009] pour convertir les squelettes WebML en codes avant qu'elles soient exécutées par WSMX.

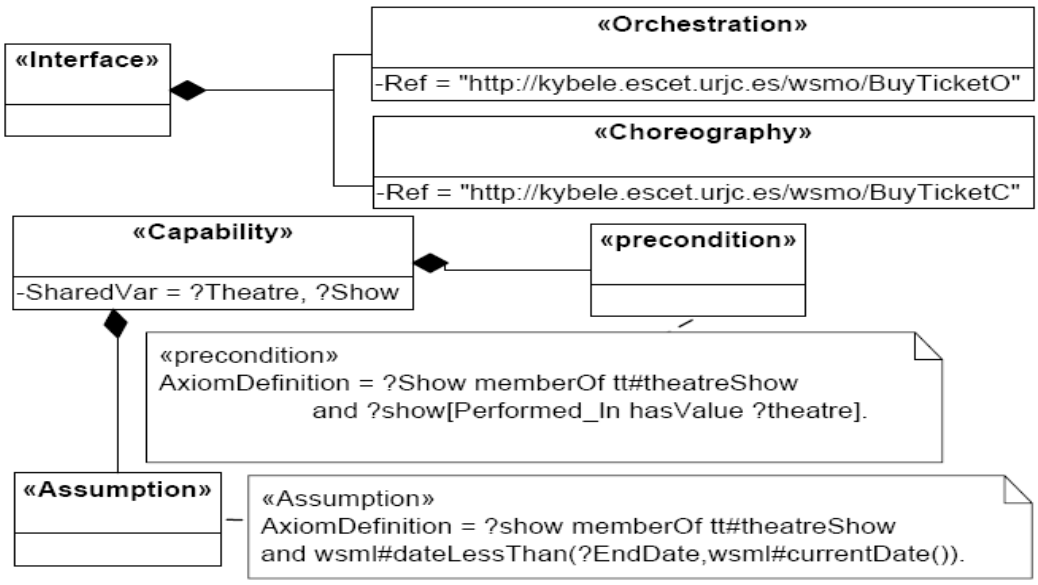


Figure 4.18 Modèle de contenu de Web service [Acuna et al, 2006]

3.2 Approches à base de formalismes UML

Les approches à base de formalismes UML extraient la description des services Web sémantique à partir d'un ensemble de diagrammes fourni par des outils UML. En fait, ces approches élaborent des descriptions sémantiques indépendamment du développement réel des services.

L'approche de [Yang et Chang, 2006] utilise les diagrammes d'état/transition et les diagrammes de classe d'UML pour générer le modèle de service de l'ontologie OWL-S. Cette approche est centrée sur le modèle de service de l'ontologie OWL-S qui décrit les informations sur l'interopérabilité de service. Le processus de génération de la sous-ontologie du modèle de service est divisé en deux sous-processus : l'information est extraite à partir des diagrammes de classe d'UML pour produire des services atomiques et l'information de génération des services composés est extraite à partir des diagrammes d'état/transition. En outre, pour transformer les diagrammes UML en des spécifications OWL-S, un ensemble de règles de transformation sont fournies. En outre, pour convertir les documents XMI qui représente les modèles UML en des spécifications OWL-S, cette approche utilise XSLT comme langue de transformation.

L'approche de [Timm et Gannod, 2008] est une architecture dirigée par les modèles pour développer les services Web sémantiques basé sur OWL-S par l'utilisation des diagrammes de classe et d'activité d'UML avec leurs profils UML. Cette approche utilise WSDL et le modèle du service Web sémantique pour accomplir les tâches des spécifications, de liaison et de l'exécution. L'approche suit quatre étapes principales : modélisation, conversion, liaison, et exécution. Le diagramme de classe d'UML est employé pour modéliser la structure du service et le diagramme d'activité d'UML facilite la modélisation de la composition. Dans l'étape de modélisation, l'approche utilise le langage OCL pour symboliser les conditions dans les diagrammes UML. Cependant, ils sont transformés en SWRL dans la sérialisation OWL-S à travers le langage de transformation XSLT. D'ailleurs, un outil automatisé de spécification et d'exécution est fourni pour charger n'importe quel document WSDL afin de faciliter la description, la liaison, et l'exécution. La figure 4.19 présente une vue d'ensemble sur l'approche de [Timm et Gannod, 2008].

L'approche de [Kim et Lee, 2009] est une méthode dirigée par les modèles qui a pour but d'extraire les descriptions sémantiques des services Web à partir des diagrammes UML complexes. La méthode concentre sur la génération des spécifications OWL-S à partir d'un ensemble de diagrammes UML tel que diagramme de classe, diagramme d'activité, et diagramme de séquence. La méthode est divisée en trois phases : la modélisation d'ontologie, de processus, et de transformation. Dans la première étape, les diagrammes de classe sont représentés à partir de l'ontologie de domaine importée. Après, cette approche emploie le diagramme de séquence et d'activité avec le profil UML et ses éléments comme le stéréotype, les valeurs étiquetées, et les contraintes.

Les objets du diagramme de séquence sont liés à un diagramme de classe approprié. Les fragments d'interaction sont employés pour décrire des constructions de contrôle de l'ontologie OWL-S pendant le processus de conception des diagrammes de séquences. Les diagrammes d'activité sont employés pour représenter le processus composé parce que leurs notations peuvent décrire diverses constructions de contrôle de OWL-S. Enfin, un document XMI est produit à partir des diagrammes UML. Ce document alors est transformé en description OWL-S par l'intermédiaire du script XSLT.

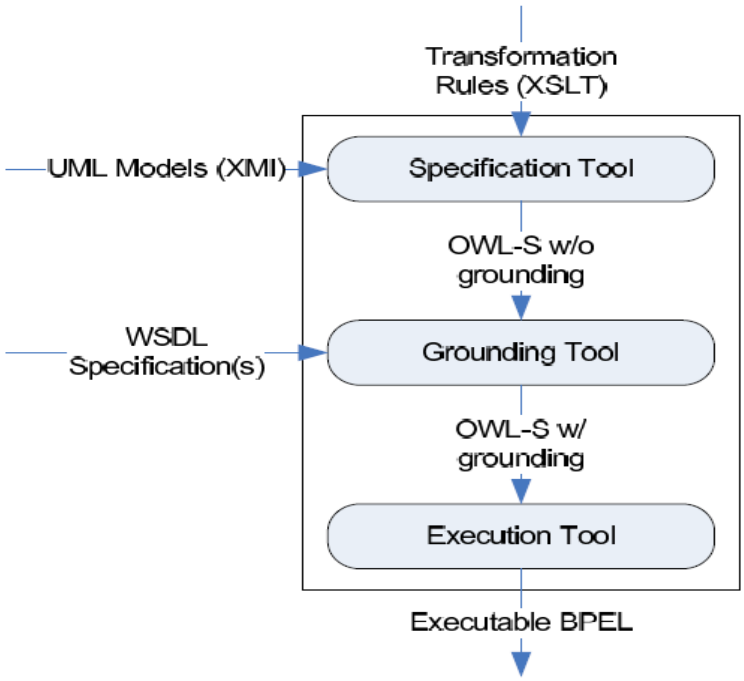


Figure 4.19 Framework de [Timm et Gannod, 2008]

3.3 Approches à base de méta-modèle

Cette catégorie contient les approches qui fournissent un méta-modèle général et indépendant pour concevoir et développer les services Web sémantiques. En particulier, il y a un profil UML pour chaque méta-modèle pour représenter les processus atomiques et composés.

L'approche de [Gronmo et al, 2005] fournit un profil UML qui est employé pour modéliser les aspects sémantiques des services Web. Le profil d'UML est créé sur la base de deux méta-modèles: le méta-modèle d'UML Ontology Profile (UOP) et le méta-modèle des éléments du modèle d'activité d'UML 2.0. En plus du profil UML, cette approche fournit un mécanisme de transformation qui transforme l'ontologie OWL-S en diagrammes UML et vice versa par l'intermédiaire d'un outil de transformation de modèles. Ce mécanisme de transformation emploie le script XSLT pour convertir un modèle en un autre. Cette approche est composée de trois étapes principales : modélisation de la composition, la découverte, et la sélection. Dans l'étape de modélisation, le modèle de composition est conçu par l'utilisation d'OCL pour représenter les conditions. La découverte est effectuée par des algorithmes d'arrangement. Enfin, les services appropriés sont choisis pour effectuer les tâches désirées dans un modèle UML.

L'approche de [Lautenbacher et Bauer, 2007] Cette approche présente un méta-modèle et un profile UML pour fournir une manière indépendante de description des services Web sémantiques. Le méta-modèle fourni est basé sur Ontology Definition Metamodel (ODM) pour soutenir OWL-S, WSMO, WSDL-S, et SWSF. Cette approche a spécifié un ensemble de règles de transformation informelles qui sont implémentées par open Architecture Ware-language Xpand [Xpand01] pour générer le code à partir un méta-modèle. Le méta-modèle dans l'approche se compose de cinq paquetages cohérents. Tous les concepts de l'ontologie sont représentés en paquetage d'ontologie semblable à ODM. Le paquetage d'interface représente le modèle de service WSDL et sa description sémantique. Les descriptions non fonctionnelles des services Web sémantique sont modélisées dans le paquetage « Service Provider ». Le paquetage fonctionnel inclut tous les aspects pour annoter chaque étape. Le paquetage « Process Flow » fusionne les concepts étendus qui sont définis dans le paquetage « Service Provider ».

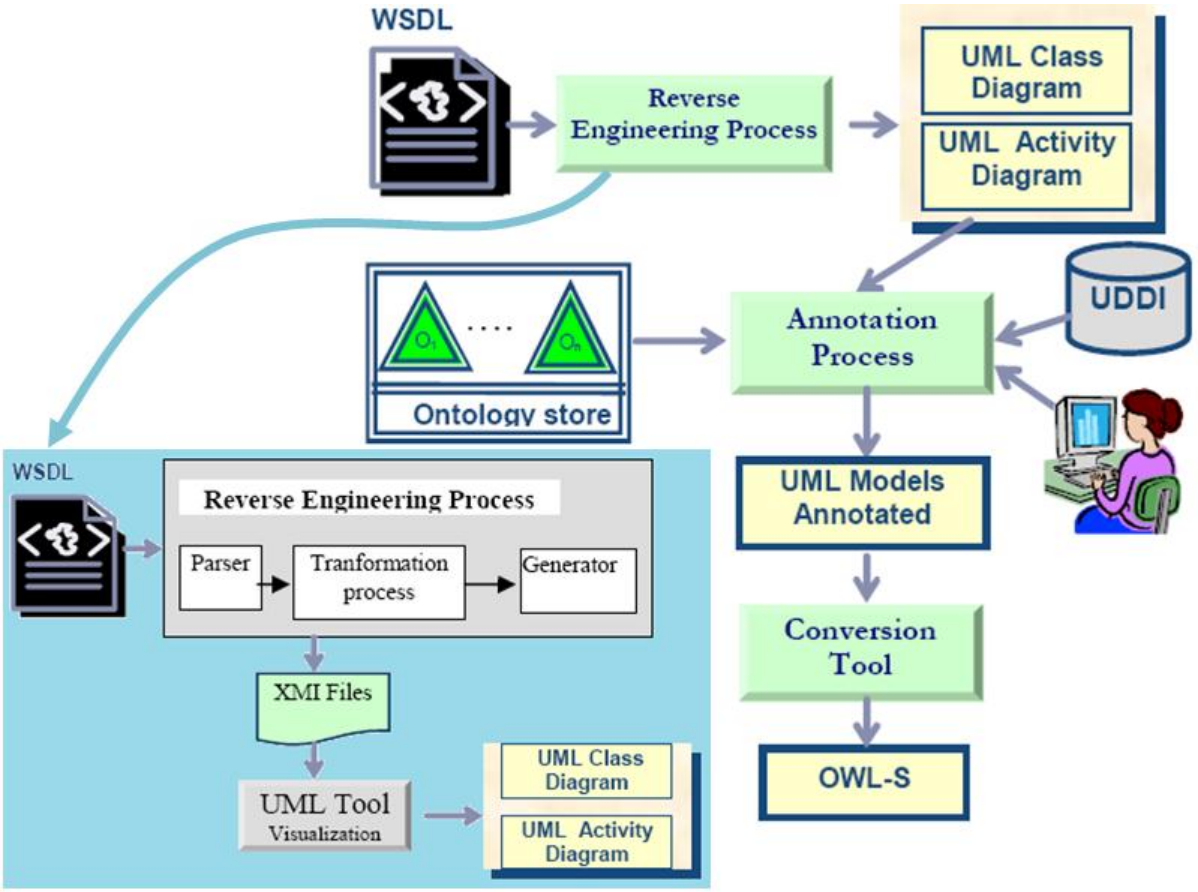


Figure 4.20 Framework de [Amar Bensaber et Malki, 2008]

L'approche de [Amar Bensaber et Malki, 2008] Comme l'approche de [Gronmo et al, 2005] fournit un profil UML basé sur UOP et les méta-modèles d'activité d'UML standard.

Cette approche utilise la rétro-ingénierie pour convertir les documents WSDL en modèle de profil UML afin de gagner un degré élevé du modèle graphique et puis, en employant des ontologies importés, elle définit le modèle UML qui représente les aspects sémantiques. Finalement, des descriptions OWL-S sont produites à partir du modèle UML. Cette approche contient trois étapes : Le processus de rétro-ingénierie, le processus d'annotation, et la conversion. (cf. figure 4.20) Cette approche effectue automatiquement le processus de liaison parce que l'accès aux URIs et les liens étaient déjà transformé à partir des documents WSDL.

Approche de [Belouadha et al, 2010] est une approche dirigée par les modèles basée sur la composition des Méta-modèles des services Web sémantiques. Le méta-modèle est indépendant de tout langage de service Web sémantique. Dans cette approche le service Web est modélisé comme un service métier qui réalise un ensemble d'opérations (interface). Chaque opération doive avoir des paramètres d'entrés, de sorties, les préconditions et les post-conditions. L'approche est basée sur SAWSDL et emploie la notation BPMN pour modéliser le comportement de composition de service Web et pour produire le code exécutable en BPEL. L'approche emploie ATL comme langue de transformation. Le tableau 4.3 présente une synthèse des approches présentées dans le présent chapitre.

	Langage de modélisation								**Langage de transformation**					**Standards pour Service Web Sémantique**				
	UML						Non-UML											
Approches	*CD*	*AD*	*SQD*	*STD*	*UP*	*Condition*	*BPMN*	*WebML*	*XPand*	*Java*	*XSLT*	*M2T*	*ATL*	*OWL-S*	*WSMO*	*SWSO*	*WSDL-S*	*SAWSDL*
Approches à base de formalismes UML																		
Yang & Chung (2006)	√			√		GUI					√			√				
Timm & Gannod (2008)	√	√			√	OCL					√			√				
Kim & Lee (2009)	√	√	√		√	Contra int					√			√				
Approches à base de méthodologie logicielle																		
Torres et al (2006)	√	√		√	√	OWL						√		√				
MIDAS-S (2006)	√				√	OCL									√			
Brambila et al (2007)							√	√			√				√			
Approches à base de méta-modèles																		
Gronmo et al (2005)	√	√			√	OCL				√	√			√	√			
Lautenbacher & Bauer (2007)	√	√			√	Contra int			√					√	√	√	√	
Bensaber & Malki (2008)	√	√			√	OCL					√			√				
Belouadha et al (2010)	√				√		√						√					√

CD=Class Diagram, AD= Activity Diagram, SQD= Sequence Diagram, STD= Statechart Diagram, UP= UML Profile, M2T= Model to Text

Tableau 4.3 synthèse des approches de spécification des services à base MDA

4. Conclusion

L'étape de l'identification dans le cycle de SOA est fondamentale pour la bonne réussite du projet. Ainsi, le choix d'une bonne approche (descendante, ascendante ou mixte) doit être fait en prenant en considération l'état de l'existant de l'entreprise, processus métier et la complexité du SI actuel.

Une fois les services candidats sont identifiés, il est indispensable de spécifier (modéliser) la structure et le comportement de chaque service. Le but de la deuxième partie de ce chapitre est de fournir une vue d'ensemble sur les différentes approches dirigées par les modèles qui spécifient les services Web et la sémantique des services Web. Ces approches sont classées en trois catégories à savoir, approches à base de méthodologie logicielle, à base de formalismes UML, et à base de méta-model. Un tableau comparatif est présenté à la fin de ce chapitre. D'après ce tableau, nous remarquons que la majorité des approches utilisent le diagramme de classe pour représenter la structure d'un service, un diagramme d'activité pour représenter le comportement d'un service, et un profile UML est fourni pour décrire les éléments de modélisation. Le langage de transformation le plus utilisé est XSLT, et la pluparts des approches utilise l'ontologie OWL-S pour représenter la sémantique des services.

La quasi-totalité des approches d'identification des services proposées sont descendantes ce qui permet un alignement avec le métier de l'entreprise. Cependant, la plupart sont prescriptives ou semi-automatiques. Les deux chapitres suivants présentent notre modeste contribution pour identifier et spécifier automatiquement les services.

Chapitre 5

Identification des services à base d'algorithmes génétiques

Chapitre 5 Identification des services à base d'algorithmes génétiques

1. Introduction

Plusieurs approches de développement orienté services utilisent les modèles de processus métiers comme point de départ pour dériver des services logiciel. La première étape dans le cycle de vie d'une architecture orientée service est l'identification des services. Elle joue un rôle critique parce qu'elle est la fondation des autres étapes. L'attention portée sur le développement automatique d'une architecture orientée service est due à la complexité de la mise en place d'une telle architecture et le besoin d'une approche méthodique pour faciliter l'identification (ou dérivation) automatique des services. Cependant, la majorité de méthodes existantes pour l'identification de service sont développées manuellement parce que, d'une part, ils sont basés sur la compétence des développeurs et, d'autre part, les modèles de processus métiers ne comportent pas suffisamment de connaissance pour identifier des services automatiquement. L'intégration de la modélisation de processus métier (BPM), le développement Dirigé par les Modèles (MDD), et l'Annotation Sémantique à base d'Ontologies (OSA) permet l'automatisation du développement de services SOA.

Trois étapes sont employées pour développer une solution orientée services: l'identification des services, la spécification de services et finalement la réalisation de service. Dans ce chapitre, nous allons présenter une méthode appelée MOOSI (Multi-Objective Optimization-based Service Identification) qui identifie automatiquement les éléments structuralement significatifs à partir d'un modèle de processus métier annoté afin de spécifier les différents artefacts des modèles de services. L'objectif principal de ce travail est de soutenir l'automatisation du processus de développement du système d'information d'entreprise orienté services. Les résultats d'exécution de notre méthode sont discutés. Ces résultats montrent que MOOSI peut atteindre une haute performance en termes de temps d'exécution et une qualité meilleure de modularisation des services identifiés en comparaison à d'autres solutions.

2. Processus d'Identification des Service Dirigé par les Modèles

La principale idée de l'architecture orientée services est la restructuration des systèmes d'information d'entreprise en un ensemble de services légèrement connectés et indépendants. Ces services devraient permettre la réutilisation des fonctionnalités implémentées auparavant afin de réduire le temps entre la conception et l'implémentation quand les exigences métiers changent. Les principaux défis pour développer les systèmes orientés services sont le mapping des modèles de processus métier vers des modèles de services. Les modèles de services jouent un rôle important pendant les phases d'analyse et de conception des services.

Selon [Arsanjani, 2004], le cycle de vie de modélisation orienté services est basé sur trois phases principales :

- Identification des services. Cette phase consiste à identifier les éléments structuralement significatifs de la solution cible. Le résultat de cette phase est un modèle de service au niveau d'analyse.
- Spécification des services. Cette phase consiste à décrire un service : ce qu'il offre, ce qu'il demande et comment il est exposé. Elle décrit également des dépendances avec d'autres services, composition des services, et messages des services. Le modèle principal lié à cette phase est un modèle de service au niveau conception.
- Réalisation des services. Cette phase consiste à fournir une solution pour un service particulier. Elle représente comment un service est réalisé. Le modèle connexe avec cette phase est un modèle de conception détaillé.

Dans ce chapitre, nous adressons en particulier la phase d'identification de service en fournissant un processus d'identification automatique de service. Le processus se compose de trois étapes pour identifier des services logiciel à partir un modèle de processus métier: L'annotation de processus métier, la transformation de processus métier, et l'identification de services de candidat [Soltani et al, 2012b] (cf. Figure 5.1).

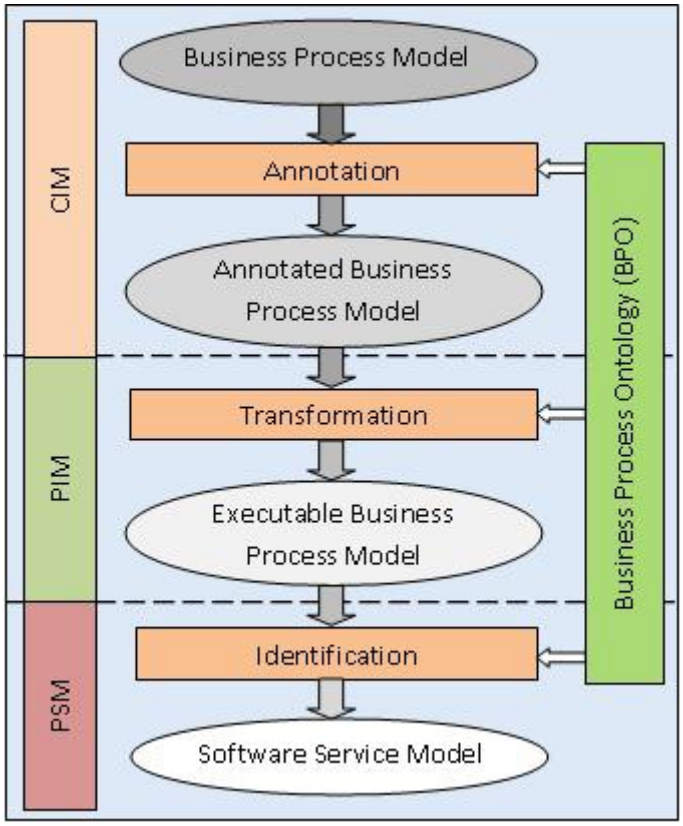


Figure 5.1 Processus d'identification des services

2.1 Annotation des processus métier

Dans cette étape, des informations complémentaires sont ajoutées au modèle métier qui est représenté comme étant un modèle indépendant au calcul (Computation Independent Model). Des métadonnées sont ajoutées à ce modèle tel que la nature des activités (c.-à-d., manuel, semi-automatique, automatique), la composabilité des activités (c.-à-d., atomique,

composé), et le but des activités (commande enregistré, commande validée etc.) [Soltani et al, 2012a].

Toutes les connaissances sur le processus métier initial sont définies à l'avance dans l'Ontologie de processus métier (BPO) par l'analyste métier et l'ingénieur de connaissance. BPO est considérée comme un méta-modèle générique du processus métier et elle est employée comme une base de connaissance pour dériver les services logiciels à partir un modèle de processus métier de haut niveau [Soltani et al, 2011]. BPO capture les concepts génériques liés aux processus métier et les relations entre eux au niveau métier. Cet Ontologie définit les concepts (Process, Activity, Event, Control Node, Message Flow, Sequence Flow, etc.) et les relations entre eux. La Figure 5.2 est un extrait qui illustre les relations du concept Activité dans BPO.

Figure 5.2 Les relations du concept Activity dans BPO.

BPO est basée sur deux principes : unifiant les différents Méta-modèles existants de processus métier et fournissant les propriétés nécessaires pour dériver des services logiciels à partir d'un processus métier de haut niveau.

Pour faciliter l'extraction des multiples vus sur un modèle de processus, le BPO permet à l'analyste métier et à l'ingénieur de connaissance de marquer la visibilité des activités à différents rôles de collaboration. Ainsi, de diverses vues sur le modèle de processus d'affaires peuvent être extraites.

2.2 Transformation des processus métier

Nous ne pouvons pas directement transformer un modèle de processus métier de niveau élevé en une solution SOA puisque il est indépendant de n'importe quelles spécifications informatique et il comporte des activités manuelles, semi-automatiques et automatiques. Comme les activités de haut niveau ont une grande granularité, la même activité métier peut être transformée en plusieurs services SOA. Ainsi, afin d'identifier les services candidats, il est nécessaire de transformer à l'avance le processus métier de haut niveau en un processus intermédiaire appelé Processus exécutable [Soltani et al, 2012b]. Car les modèles de processus métier sont à un niveau d'abstraction plus élevé que les modèles de processus exécutables, la connaissance additionnelle de domaine devra être ajoutée pendant cette étape. BPO est interrogée pour transformer le modèle de processus métier annoté en un processus exécutable exprimé comme un modèle indépendant de plate-forme (PIM) [Soltani et al, 2010a; Soltani et al, 2010b]. Pendant cette étape, cinq opérations de transformation sont effectuées :

- Renommer une activité métier,
- Décomposer une activité métier en plusieurs sous activités,
- Fusionner deux activités métier en une seule,
- Insérer une nouvelle activité dans le processus exécutable,
- Enlever une activité du processus métier initial.

2.3 Identification des services candidats

L'identification des services correspondants aux activités exécutables est réalisée en interrogeant l'ontologie de processus métier pour rechercher et extraire les propriétés du concept Activité et ses relations avec les autres concepts de l'ontologie (c.-à-d. ressource, participant, et but). Les propriétés et les relations extraits sont employés pour calculer les différentes dépendances sémantiques comme (activité - Acteur), (activité - entité métier) et (activité - but métier).

Les différentes dépendances sémantiques sont modélisées comme un graphe de dépendance d'activités qui est considéré comme donnée d'entrée pour identifier un ensemble de services candidats implémentant le processus métier initial comme modèle spécifique de plate-forme (PSM).

Pendant le processus d'identification, un ensemble de métrique de conception telle que le couplage, la cohésion, et la qualité de modularisation sont calculés. Ces métriques de conception sont employées comme des paramètres de commande de l'algorithme de Clustering qui produise comme résultat un ensemble optimal de Clusters de services.

Nous formulons notre algorithme d'identification comme un problème d'optimisation multi-objectif qui classifie les services candidats selon des valeurs optimales des métriques de conception.

3. MOOSI: A Multi-Objective Optimization-based Service Identification

La première phase dans le cycle de vie de développement de service est l'identification de service. Le but de l'identification de service à partir un modèle de processus métier est d'organiser les activités métiers en un ensemble de clusters significatifs (cohésion forte et couplage faible). Les activités d'un cluster sont considérées comme des opérations du service candidat.

Nous proposons de combiner trois techniques pour grouper les activités métiers afin de former des services logiciels [Soltani et al, 2013]:

- Technique centrée sur les Entités Métier : elle consiste à mettre toutes les activités associées avec une entité métier particulière dans un service. Dans ce cas-là, Une matrice CRUD (Create Read Update Delete) notée (CRUDM) est employée comme entrée dans l'algorithme de clustering.
- Technique centrée sur les Acteurs : elle consiste à mettre toutes les activités exercées par un acteur particulier dans un service. Dans ce cas-là, une matrice Activity-Actor Matrix (AAM) est employée comme entrée dans l'algorithme de clustering.
- Technique centrée sur les Buts métiers : elle consiste à mettre toutes les activités participantes à l'accomplissement d'un but particulier dans un service. Dans ce cas-là, la matrice Activity-Goal Matrix (AGM) est employée comme entrée dans l'algorithme de clustering.

Nous pouvons employer un graphique dirigé pour représenter des dépendances entre les activités métiers afin de rendre la structure complexe d'un processus métier plus compréhensible. Nous appelons un tel graphe Activity Dependency Graph (ADG). Dans le graphe de dépendance d'activité, les fonctions métiers (activités métier, sous processus) sont représentées comme nœuds et les relations entre eux comme lignes qui relient les nœuds. Ainsi, une activité A est reliée à une activité B si les deux activités manipulent la même entité métier et/ou est exercée par le même acteur et/ou participe à l'accomplissement du même but métier. Une manière de rendre l'ADG plus cohérent est de le diviser en groupant étroitement des activités fortement liées dans des clusters.

Définition 1 (Cluster)

Soit $As = (A_1, A_2, \ldots, A_n)$ un ensemble d'activités métiers d'un système d'information. Chaque activité manipule un ou plusieurs entité métier et est exécutée par un ou plusieurs acteur et participe à l'accomplissement de un ou plusieurs but métier.

Formellement, un cluster C_i peut être défini comme:

$$C_i = \{A_j \mid 1 \leq j \leq |As| \text{ où } A_j \in As\} \quad (5.1)$$

Définition 2 (Partition)

Une partition est un ensemble de sous-ensembles non vides de *As*. Formellement, une partition π peut être définie comme :

$\pi = \{C_k\} \mid 1 \leq k \leq |As|$, *où:*

$$\begin{cases} \bigcup_{i=1}^{k} C_i = As \\ \bigcap_{i=1}^{k} C_i = \emptyset \end{cases} \quad (5.2)$$

En outre, la partition de $\boldsymbol{As}$ en $\boldsymbol{k}$ clusters non vide s'appelle une k-partition de $\boldsymbol{As}$

Étant donné un ensemble $\boldsymbol{As}$ qui contient $\boldsymbol{n}$ éléments, le nombre $\boldsymbol{S_{(n,k)}}$ appelés *Stirling number* du k-partition distincte de $\boldsymbol{As}$ satisfait l'équation récurrente présentée par [Mancoridis et al, 1998].

$$S_{n,k} \begin{cases} 1 & si\ k = 1 \ \ or\ k = n \\ S_{n-1,k-1} + k * S_{n-1,k} & si\ non \end{cases} \quad (5.3)$$

La création d'une partition significative d'un ADG est difficile parce que le nombre de partitions possible est très grand même pour un petit graphe.

$\boldsymbol{S_{(n,k)}}$ se développe exponentiellement en fonction de la taille de As. Par exemple, un graphe de dépendance d'activité de 7 nœuds admis 877 partitions distinctes, alors qu'un ADG de 25 nœuds admis 4.638.590.332.229.999.353 partition distinctes.

3.1 Métrique de conception pour le développement de service

Une façon efficace pour augmenter la qualité des logiciels consiste à utiliser des métriques afin de guider le processus de développement. Pour évaluer et mesurer la qualité des services produits par notre prototype de clustering dans chaque itération, nous adoptons trois métriques de conception qui sont : couplage, cohésion, et qualité de modularisation.

Définition 3 (couplage entre deux clusters)

Le couplage entre deux clusters noté ($CoupBC$) mesure le degré de connectivité entre deux clusters distincts. Un niveau important d'interconnectivité est indésirable parce qu'il indique que les services sont fortement dépondant les uns aux autres. Réciproquement, un bas degré d'interconnectivité est souhaitable parce qu'il indique que les différents services sont largement indépendant les uns aux autres.

Formellement, $CoupBC_{ab}$ (le couplage entre le cluster a et le cluster b) mesure le rapport des interdépendances entre le cluster a et le cluster b et le nombre maximal possible des interdépendances entre ces clusters.

$$CoupBC_{ab} = \begin{cases} 0 & si\ a = b \\ \dfrac{2 * \sum_{i=1}^{Na} \sum_{j=1}^{Nb} TDM_{ACa_i, ACb_j}}{2 * Na * Nb} & si\ a \neq b \end{cases} \quad (5.4)$$

Où $\boldsymbol{Na}$ est le nombre d'activités dans le cluster $\boldsymbol{a}$, $\boldsymbol{Nb}$ est le nombre d'activités dans le cluster $\boldsymbol{b}$, $\boldsymbol{ACa_i}$ est la i[eme] activité du cluster $\boldsymbol{a}$, $\boldsymbol{ACb_j}$ est la j[eme] activité du cluster $\boldsymbol{b}$, et la valeur $\boldsymbol{TDM_{ACa_i, ACb_j}}$ (*Total Dependency Matrix*) représente le degré de dépendance entre la i[eme] activité du cluster $\boldsymbol{a}$ et la j[eme] activité du cluster $\boldsymbol{a}$.

Définition 4 (le couplage d'un cluster donné)

Le couplage d'un cluster donné noté (CoupOC) mesure le degré de connectivité entre un cluster donné et tous les autres clusters du système. Formellement, le couplage d'un cluster est défini comme :

$$CoupOC_a = \begin{cases} 0 & si\ M = 1 \\ \dfrac{\sum_{b=1}^{M} CoupBC_{ab}}{M - 1} & si\ non \end{cases} \quad (5.5)$$

Où M est le nombre du cluster dans le system.

Définition 5 (cohésion d'un cluster)

La cohésion de service se rapporte au degré de la force de pertinence fonctionnelle des activités effectuées par un service pour réaliser un processus d'affaires [Ma et al, 2009].

Le degré de cohésion dépend de la force des dépendances entre les activités du cluster approprié (service). Formellement, la cohésion d'un cluster a ($CohOC_a$) est définie comme:

$$CohOC_a = \begin{cases} 0 & si\ \ n = 1 \\ \dfrac{\sum_{i=1}^{n-1} \sum_{j=i+1}^{n} (2 * TDM_{ACa_i, ACa_j})}{n * (n - 1)} & si\ \ n > 1 \end{cases} \quad (5.6)$$

où $\boldsymbol{n}$ est le nombre d'activités dans le cluster , $\boldsymbol{TDM}$ (Total Dependency Matrix) est la matrice de dépendance total, $\boldsymbol{ACa_i}$ est la i[eme] Activité du Cluster $\boldsymbol{a}$, et $\boldsymbol{ACa_j}$ est la j[eme] Activité du Cluster $\boldsymbol{a}$.

Définition 6 (facteur de modularisation)

Le facteur de modularisation (MF) est le rapport entre la cohésion et de le couplage de chaque cluster. Formellement, MF est définie comme suit :

$$MF_a = \begin{cases} 0 & si\ CohOC_a + CoupOC_a = 0 \\ \dfrac{CohOC_a}{CohOC_a + \left(\frac{CoupOC_a}{2}\right)} & si\ non \end{cases} \quad (5.7)$$

Où $\boldsymbol{CohOC_a}$ est la cohésion du cluster $\boldsymbol{a}$, et $\boldsymbol{CoupOC_a}$ est le couplage du cluster $\boldsymbol{a}$.

Définition 7 (qualité de modularisation)

La qualité de modularisation (MQ) détermine la qualité d'une partition d'un ADG, parfaitement, elle permet de mesurer la différence entre l'interconnectivité (c.-à-d., dépendances entre les activités de deux services distincts) et l'intra-connectivité (c.-à-d., dépendances entre les activités du même service). Cette différence est basée sur l'hypothèse que des systèmes logiciels orientés services bien conçus sont organisés en sous-systèmes cohésifs qui sont faiblement interconnectés. Par conséquent, une haute MQ permet la création des clusters fortement cohésifs qui ne sont légèrement couplés. Formellement, la qualité de modularisation (MQ) est définie comme la moyenne des MF de tous les clusters :

$$MQ = \frac{1}{n} * \sum_{i=1}^{n} MF_i \quad (5.8)$$

Où $\boldsymbol{n}$ est le nombre total des clusters dans le system, et MF_i est le Facteur de Modularisation du i^{eme} cluster.

3.2 Processus d'identification des services

Le processus d'identification des services commence par interroger l'Ontologie de processus métier, celui contiennent toute la connaissance sur le processus métier d'entrée, pour extraire les propriétés qui décrivent les dépendances existantes entre le concept Activité et les concepts Entité métier, Acteur et le but respectivement.

Trois matrices sont créées : (i) CRUD Matrix(CRUDM) qui décrit les relations entre les activités et les entités métier, (ii) Activité-Acteur Matrix (AAM) qui décrit les relations entre les activités et les acteurs, et (iii) Activité-But Matrix (AGM) qui décrit les relations entre les activités et les buts métier (cf. figure 5.3).

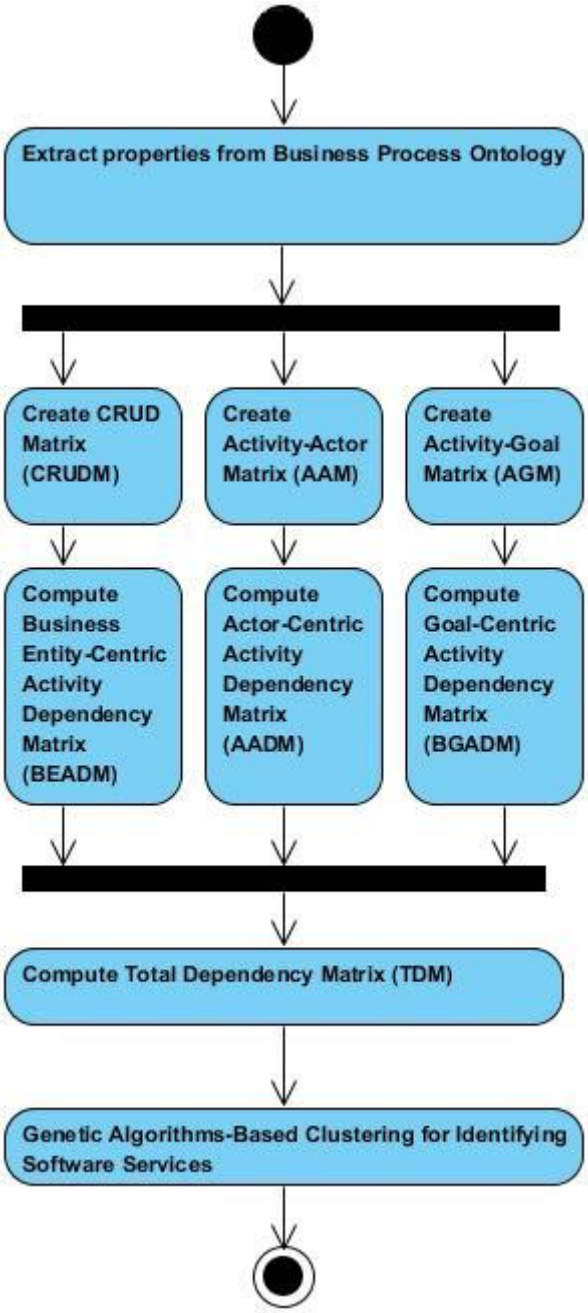


Figure 5.3 Vue d'ensemble sur la procédure d'identification

3.2.1 Matrices des relations d'activité

- **CRUD Matrix**

Pour grouper les activités dans des clusters en utilisant la technique centrée sur les Entité métiers, nous avons besoin d'un taux qui représente le degré de la relation sémantique entre l'Activité et l'Entité métier. Quatre opérations peuvent être appliquées par l'activité sur chaque entité métier. Ces opérations s'appellent CRUD (C : Create, R : Read, U : Update, D : Delete).

Afin de calculer la valeur de chaque métrique, telle que la cohésion, le couplage, et la qualité de modularisation, il est nécessaire d'associer à chaque opération une valeur entre 0 et 1.

La majorité de travaux existants sur des relations sémantiques CRUD adoptent les substitutions : C=1 ; U=0.75, D=0.50, R=0.25. Ces valeurs représentent le coût de l'opération sur chaque entité métier.

Nous stockons ces valeurs dans la Matrice CRUD (CRUDM), qui a des activités métier en tant que lignes, des entités métiers en tant que colonnes, et des relations sémantiques (« C », « U », « D », « R », avec l'intensité distincte 1=>C>U>D>R>0) en tant que ses cellules. Chaque colonne dans cette matrice doit avoir exactement une opération (Create). Le tableau 5.1 montre un exemple de CRUDM.

	BE1	BE2	BE3	BE4	BE5	BE6
Activity1		C	U	D	C	D
Activity2	R	D	C	R		U
Activity3	C				D	
Activity4			D	C		
Activity5	D	R			R	C

Tableau 5.1 Exemple de *CRUD Matrix*

- **Activity-Acteur Matrix**

Afin de regrouper des activités en utilisant la technique centrée sur l'Acteur, nous représentons le degré des relations sémantiques entre l'activité et l'acteur par une deuxième matrice appelé Activité-Acteur Matrix (AAM).

La valeur de chaque cellule dans AAM représente un pourcentage d'accomplissement des activités par les acteurs. par exemple la valeur $AAM_{1,3}$=0.75 indique que l'acteur 3 exercent 0.75% de l'activité 1.

La somme de chaque ligne dans AAM doit être égale à 1, pour indiquer que l'activité correspondante est exercée complètement.

- **Activity-Goal Matrix**

Pour grouper des activités métiers en se basant sur la technique centrée sur les But, nous employons une troisième matrice appelée Activity Goal Matrix (AGM). Chaque cellule d'AGM représente le degré des relations sémantiques entre l'activité et le but métier.

La valeur de chaque cellule d'AGM indique le taux d'accomplissement d'un but métier après l'exécution d'une activité. Par exemple la valeur $AGM_{2,4}$=0.50 indique que l'activité 2 réalisent 50% du but 4. La somme de chaque colonne de AGM doit être égale à 1, pour indiquer que le but correspondant est atteint complètement.

3.2.2 Dérivation de la matrice de dépendance totale

Trois matrices carrées sont dérivées à partir CRUDM, AAM , et AGM. La première s'appelle *Business Entity-centric Activity Dependency Matrix* (BEADM), la second s'appelle *Actor-centric Activity Dependency Matrix* (AADM), et la troisième s'appelle *Business Goal-centric Activity Dependency Matrix* (BGADM).

Ces matrices carrées représentent le degré de dépendance entre chaque activité et les autres activités du processus métier d'entrée.

Pour calculer la dépendance entre deux activités, nous prenons la valeur minimale des rapports entre les deux activités avec la même entité métier.

Par exemple, le degré de dépendance entre l'activité 1 qui lire l'entité métier 5 et l'activité 3 qui créer la même entité métier 5 est égal à 0.25 (nous employons la valeur minimale entre R : 0.25 et C : 1). Si aucune opération n'est appliquée par l'activité 2 sur les entités métier manipulée par l'Activité 1, nous concluons qu'il y a zéro dépendance entre l'activité 1 et l'activité 2.

Nous identifions les services candidats à partir de trois matrices de dépendance d'activités AADM, BEADM, et BGADM afin d'augmenter la qualité des clusters de services.

L'algorithme 5.1, l'algorithme 5.2, et l'algorithme 5.3 illustrent la méthode de calcul des dépendances entre différentes activités afin de dériver BEADM, AADM, et BGADM respectivement.

Algorithme 5.1: La création de Business Goal-Centric Activity Dependency Matrix (BGADM)

Entré: CRUDM /* La Matrice CRUD */

Sortie: BEADM /* Business Entity-Centric Activity Dependency Matrix */

Pour i allant de 1 à nombre_d'Activités-1 **faire**

  Pour j allant de i+1 à nombre_d'Activités **faire**

    $BEADM_{i.i}$=0;

    $BEADM_{j.j}$=0;

    $BEADM_{i.j}$=0;

    Pour k allant de 1 à nombre_d'entités_métiers **faire**

      $BEADM_{i,j}$= $BEADM_{i,j}$ + min($CRUDM_{i,k}$,$CRUDM_{j,k}$);

    Finpour

    $BEADM_{i,j}$= $BEADM_{i,j}$ / nombre_d'entités_métiers;

    $BEADM_{j,i}$=$BEADM_{i,j}$;

  Finpour

Finpour

Algorithme 5.2: La création de Actor-Centric Activity Dependency Matrix (AADM)

Entré: AAM /* Activity-Actor Matrix */

Sortie:AADM /*Actor-Centric Activity Dependency Matrix*/

Pour i allant de 1 à nombre_d'Activités-1 **faire**

  Pour j allant de i+1 à nombre_d'Activités **faire**

    $AADM_{i.i}$=0;

    $AADM_{j.j}$=0;

    $AADM_{i.j}$=0;

    Pour k allant de 1 à nombre_d'acteurs **faire**

      $AADM_{i,j}$= $AADM_{i,j}$ + min($AAM_{i,k}$,$AAM_{j,k}$);

    Finpour

    $AADM_{i,j}$= $AADM_{i,j}$ / nombre_d'acteurs;

    $AADM_{j,i}$=$AADM_{i,j}$;

  Finpour

Finpour

Algorithme 5.3: La création de Business Goal-Centric Activity Dependency Matrix (BGADM)

Entré: AGM /* Activity-Goal Matrix */

Sortie: BGADM /*Business Goal-Centric Activity Dependency Matrix*/

Pour i allant de 1 à nombre_d'Activités-1 **faire**

  Pour j allant de i+1 à nombre_d'Activités **faire**

    $BGADM_{i.i}$=0;

    $BGADM_{j.j}$=0;

    $BGADM_{i.j}$=0;

    Pour k allant de 1 à nombre_de_buts **faire**

      $BGADM_{i,j}$= $BGADM_{i,j}$ + min($AGM_{i,k}$,$AGM_{j,k}$);

    Finpour

    $BGADM_{i,j}$= $BGADM_{i,j}$ / nombre_de_buts;

    $BGADM_{j,i}$=$BGADM_{i,j}$;

  Finpour

Finpour

Une fois que les matrices BEADM, AADM, et BGADM sont créées, elles sont intégrées dans une seule matrice appelée Total Dependency Matrix (TDM) qui représente la dépendance moyenne entre chaque couple activités. Chaque cellule de TDM est calculée comme suit :

$$TDM_{ij} = \frac{1}{3} * \left(AADM_{ij} + BEADM_{ij} + BGADM_{ij}\right) \quad (5.9)$$

3.2.3 Clustering à base d'Algorithmes génétique

Le clustering est une opération qui consiste à diviser l'ensemble de données d'entrée dans des sous-ensembles (clusters), de sorte que les données dans chaque sous-ensemble partagent des aspects communs. Le partitionnement des données est souvent indiqué par une mesure de similarité exprimé par une distance.

Plusieurs techniques du clustering sont présentées dans la littérature. Le problème de l'identification de service est un problème NP-difficile dont la résolution avec une méthode déterministe n'est pas possible dans une durée de calcul raisonnable. Pour cela, nous avons utilisés un algorithme génétique pour le résoudre.

Les algorithmes génétiques travaillent sur un ensemble (population) d'individus représentant les données d'entrée du problème. Chaque itération de l'algorithme s'appelle une génération. Dans chaque génération, une nouvelle population est créée par les opérateurs génétiques qui sont : la mutation, le croisement, et la sélection des individus les plus puissants de la génération précédente.

L'idée fondamentale du clustering à base d'algorithme génétique est de combiner les individus afin de dériver une meilleure partition. Par conséquent, chacun individu représente une partition et non seulement un cluster.

Un individu, dans le vocabulaire des algorithmes génétiques, est appelé chromosome. Chaque chromosome se compose d'un ensemble de gènes. Nous avons codé un individu par un ensemble de lettres comme il est illustré dans l'exemple suivant.

C	A	B	A	B	C
0	1	2	3	4	5

←chromosome C[]

C[0] =C signifié que l'activité 0 est affectée dans le cluster C, c[1] =A signifié que l'activité 1 est affectée dans le cluster A, et ainsi de suite. Le tableau 5.2 illustre la partition représentée par le chromosome C [].

Cluster C	Cluster A	Cluster B
Activity 0	Activity 1	Activity 2
Activity 5	Activity 3	Activity 4

Tableau 5.2 Partition représentée par le chromosome.

Nous illustrons dans l'algorithme suivant la procédure de clustering à base d'algorithme génétique

Algorithme 5.4:Genetic-algorithm-based clustering

Entré: TDM /*Total Dependency Matrix*/

Sortie: un ensemble d'individus représentant les partitions optimales

1. Générer aléatoirement une population initiale de taille fixe.

2. Evaluer la population courante.

3. Selectionner 50% des meilleurs individus de la population courante.

4. Appliquer le croisement entre les individus sélectionnés afin de générer une nouvelle population.

5. Appliquer la mutation sur chaque individu de la nouvelle polulation.

6. Remplacer l'ancienne population par la nouvelle.

7. Si le compteur d'itération est inférieur au nombre maximale d'itération alors aller à 2, si non arrêter le processus et afficher les individus qui représentent les partitions optimales.

- **La sélection**

La sélection est un opérateur essentiel dont le principe consiste à choisir les meilleurs individus de la population courante afin de produire une nouvelle population plus puissante.

Nous avons adopté, dans notre processus de clustering, une stratégie simple de sélection qui consiste à choisir l'individu qui a une fonction objective plus grande ou égale la moyenne de toutes les fonctions objective de la population courante.

L'algorithme suivant illustre notre fonction de choix.

Algorithme 5.5:La sélection des meilleurs individus

```
Entré: une Population
Sortie: un ensemble d'individus
initialser la list des meilleurs individus
Pour i allant de 1 à nombre_d'individus faire
Chromosome c=individus(i) ;
  Si c.fitness>= La_Moyenne_des_fitness Alors
     Ajouter c à la liste des meilleurs Individus;
  Finsi
Finpour
Retourner la liste des meilleurs individus ;
```

La forme physique d'un individu est calculée selon les trois objectifs suivants : (i) la qualité de la modularisation (ii) la somme d'intra-connectivité, et (iii) la somme d'interconnectivité. Formellement,

$$fitness = MQ * \left(\frac{1}{n} * \sum_{i=1}^{n} CohOC_i\right) * \left(1 - \left(\frac{1}{n} * \sum_{j=1}^{n} CoupOC_j\right)\right) \quad (5.10)$$

où n est le nombre maximal des clusters représentés par l'individu courant, $CohOC_i$ est la cohésion du i^{eme} cluster, et $CoupOC_j$ est le couplage du j^{eme} cluster.

Un meilleur individu est un individu qui a une qualité de modularisation forte, une cohésion forte, et un couplage faible qui implique une fonction objective élevée.

- **Croisement**

Le croisement est un opérateur génétique basé sur le principe que les enfants héritent des qualités de leurs parents. Le format standard du croisement est $C: E \times E \rightarrow E \times E$, qui croise, avec un certain probabilité $Pc(0 \leq Pc \leq 1)$, deux parents $(p1, p2) \in E \times E$.

Nous avons utilisé, dans notre processus de clustering, un croisement uniforme qui consiste à produire aléatoirement un masque binaire de même taille que les chromosomes afin de reproduire deux nouveaux enfants E1 et E2 à partir de deux parents P1 et P2.

$$\begin{cases} E_{1i} = P_{1i} \quad and \quad E_{2i} = P_{2i} & si \;\; Mask_i = 1 \\ E_{1i} = P_{2i} \quad and \quad E_{2i} = P_{1i} & si\ non \end{cases} \quad (5.11)$$

- **Mutation**

L'idée générale de la mutation est la modification (avec une certaine probabilité P.M., 0≤pm≤1) d'un ou plusieurs gènes de l'individu choisi, afin de présenter une variabilité dans la population.

4. Implémentation et évaluation

Afin d'évaluer les diverses phases de notre méthode d'identification automatique de services logiciel exposés en ce chapitre, nous avons développé un outil logiciel basé sur un algorithme génétique qui prend comme entrée une Ontologie de processus métier et produit comme sortie un ensemble d'activités organisées en groupes optimaux.

Les activités du même groupe doivent avoir une cohésion forte, un couplage faible, et une qualité de modularisation élevée. La figure 5.4 montre le graphe de dépendance d'activité qui représente l'entrée du processus de clustering. À la fin du processus de groupement, les activités fortement dépendantes sont mises dans le même cluster comme il est présenté dans la figure 5.5

Pour évaluer l'exécution et la qualité des clusters produits par notre outil, nous avons exporté une courbe qui représente la valeur de la fonction objective dans chaque itération ainsi que les propriétés des partitions optimales telles que la qualité de la modularisation, la cohésion, le couplage et le temps d'exécution (voir figure 5.6).

Afin d'évaluer l'exécution et la qualité du groupement, nous avons appliqué notre algorithme de clustering sur cinq processus métier BP01, BP02, BP03, BP04, et BP05

correspondant respectivement à la gestion des clients, la gestion d'achat, la gestion des ventes, la gestion des commandes, et à la gestion de la production. Les résultats obtenus sont présentés dans le tableau 5.3.

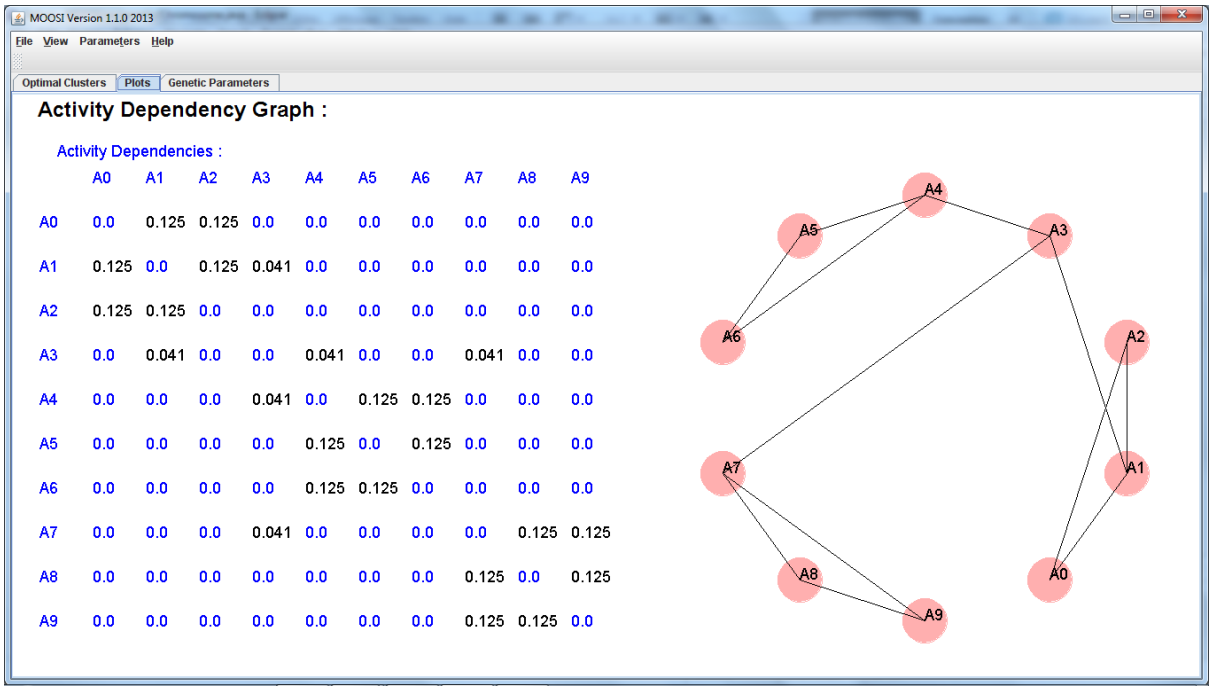


Figure 5.4 Activity Dependency Graph (input).

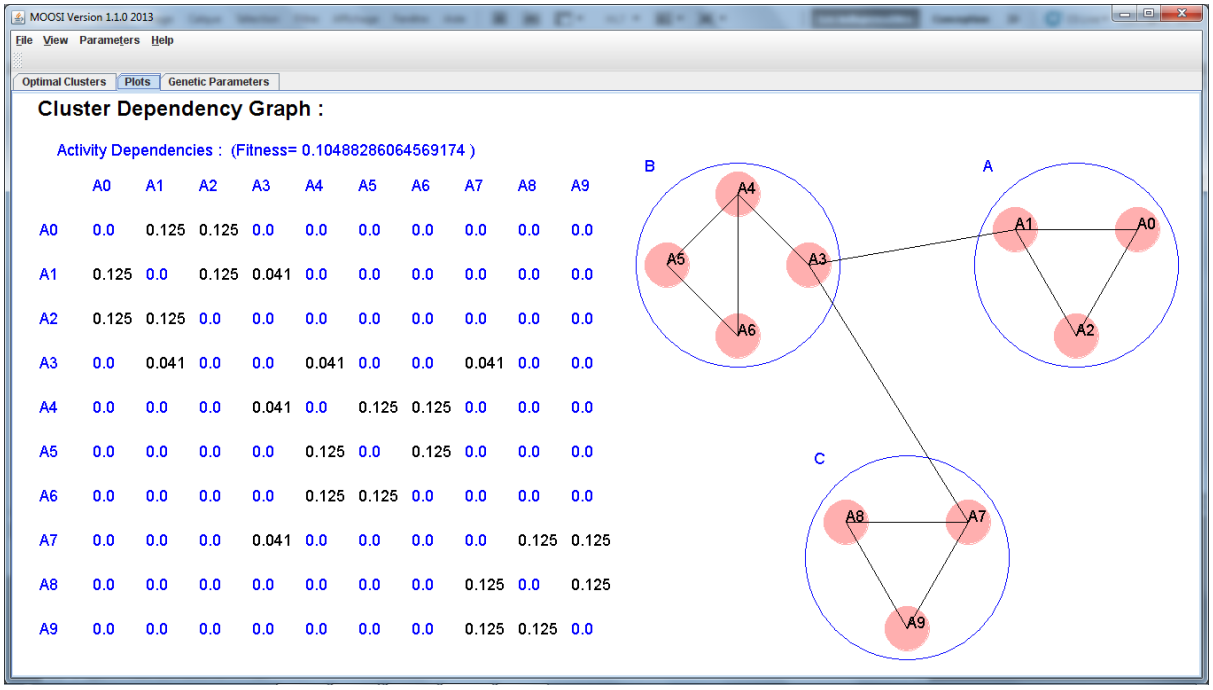


Figure 5.5 Cluster Dependency Graph (output).

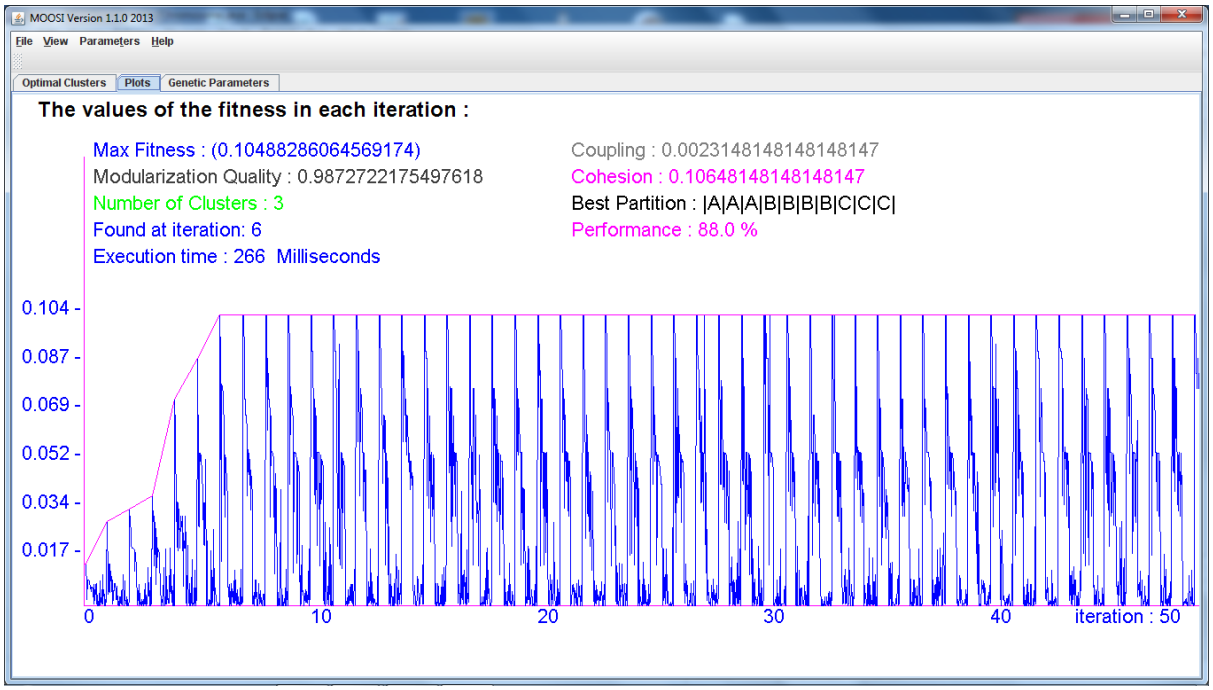


Figure 5.6 Etapes d'exécution de MOOSI tool.

	BP01	BP02	BP03	BP04	BP05
Number of Activities	7	9	10	9	13
Number of Edges	15	15	12	15	39
Number of Clusters	2	4	3	3	5
Fitness	0.108	0.101	0.104	0.202	0.055
Modularization Quality	0.877	0.888	0.987	0.855	0.878
Average of Cohesion	0.128	0.118	0.106	0.25	0.064
Average of Coupling	0.034	0.028	0.002	0.052	0.016
Found at iteration	2	7	6	8	13
Execution time in milliseconds	32	203	266	265	1685

Tableau 5.3 résultats expérimentaux

L'analyse des résultats fourni par MOOSI, prouve que les clusters produits une cohésion forte, un couplage faible et une qualité de modularisation supérieur à 85%. La majorité de solutions optimales sont trouvées dans un nombre réduit d'itération qui correspond à un temps d'exécution minimal.

En appliquant notre algorithme sur le même processus métier utilisé par [Jamshidi, 2012], nous avons trouvé les clusters optimaux dans un nombre réduit d'itération qui implique une performance élevée. D'ailleurs, le nombre de clusters optimaux dans MOOSI est supérieur au nombre de clusters optimaux dans l'approche de [Jamshidi, 2012] qui implique une qualité de modularisation importante (voir le tableau 5.4).

[Jamshidi, 2012] propose un algorithme de clustering à deux dimensions qui implique une augmentation de la complexité. Il groupe les activités liées à la même entité métier dans le même cluster et les entités métiers manipuler par la même activité dans le même cluster.

Contrairement, l'algorithme adopté par MOOSI est un algorithme de clustering à simple dimension. Il calcule des dépendances entre les activités en se basant sur les relations sémantiques existantes entre les activités et les entités métiers afin de grouper les activités les plus dépendantes dans le même cluster. Ceci implique une réduction de la complexité de l'algorithme.

	MOOSI	Jamshidi's approach
Number of Activities	13	13
Number of Edges	39	39
Number of Clusters	5	4
Found at iteration	13	5313

Tableau 5.4 Comparaison entre MOOSI et le prototype de [Jamshidi, 2012].

Noter qu'à la fin du processus de groupement, il est facile de grouper les entités métier manipulées par la même activité dans le même cluster en assignant chaque entité métier au cluster qui contient l'activité créatrice.

5. Conclusion

Dans ce chapitre, nous avons présenté une méthode appelée MOOSI (Multi-Objective Optimization-based Service Identification) pour identifier automatiquement des services SOA candidat à partir d'un processus métier de haut niveau. La méthode illustre comment un ensemble de services candidats peuvent être dérivés automatiquement à partir un modèle de processus métier. Une Ontologie de processus métier notée(BPO) est adoptée pour annoter le modèle de processus métier d'entrée. Le modèle de processus annoté est considéré comme une entrée d'un moteur de transformation qui le transforme, après l'interrogation de l'ontologie, en processus exécutable. Enfin un moteur d'identification interroge l'ontologie pour extraire toutes les propriétés utilisées comme données d'entrée pour produire des services candidats automatiquement. Le moteur d'identification applique une méthode de clustering à base d'algorithme génétique. Les résultats d'implémentation prouvent que MOOSI peut atteindre une haute performance en termes de temps de calcul et une qualité importante en termes de qualité de modularisation des services identifiés comparés à autre solution.

Chapitre **6**

Spécification des services à base MDA-Etude de cas

Chapitre 6 Spécification des services à base MDA : Étude de cas

1. Introduction

Les services logiciel évoluent jour après jour, et gagnent plus d'intérêt par les entreprises en particulier dans les réseaux de collaboration. L'évolution rapide des technologies de service Web joue un rôle central dans le processus de développement de logiciel. Le développement des services logiciel a devenu un sujet très intéressant, qui mérite plus d'investissement en matière de recherche.

Analogiquement à l'architecture de logiciel traditionnelle, les services logiciels disposent une structure et un comportement spécifiques. Les spécifications structurales du service représentent la partie statique des applications orientées services qui sont composées d'entités fonctionnelle candidates et de leurs relations. Les spécifications comportementales du service représentent la partie dynamique, qui définisse comment le service logiciel fonctionne à l'intérieur. Les deux parties sont importantes pour modéliser et implémenter les applications orientées services.

Une direction récente pour développer les applications orientées services est d'exploiter les principes de développement de logiciel dirigé par les modèles (MDD) en modélisant la structure et le comportement des services sous une forme plus abstraite et en définissant les transformations des modèles qui permettent de passer d'une spécification abstraite vers un système exécutable spécifique à une plateforme donnée [Soltani et al, 2010a ; Soltani et al, 2010b].

Dans ce chapitre nous allons présenter une méthode et un modèle conceptuel indépendant à la technologie qui est employé pour spécifier automatiquement la structure et le comportement des services logiciel à partir d'un modèle de processus métier de haut niveau. Notre méthode de spécification des services est illustrée par une étude de cas.

2. Modèle conceptuel SOA

Nous avons présenté dans ce travail un modèle conceptuel de niveau supérieur inspiré par le modèle de référence d'OASIS [OASIS01, 2006] (qui n'est pas lié directement à aucun standard, technologies ou à d'autres détails d'implémentation concrets). Ce modèle définit un cadre abstrait pour comprendre les relations significatives entre les entités de l'environnement SOA et permet le développement des architectures concrètes en utilisant des standards conformes supportant cet environnement. Les concepts abstraits dans le modèle conceptuel forment la base pour le développement des solutions SOA concrètes en utilisant les standards technologiques. En plus que la documentation des projets SOA, le modèle conceptuel est considéré comme une base pour la compréhension globale du projet SOA. En fait, il définit tous les éléments conceptuels principaux qui devraient être considérés dans la conception rigoureuse d'un cadre intégrateur pour la modélisation, l'analyse, le développement, la validation, et le déploiement des services légers robustes au-dessus des réseaux informatiques. La figure 6.3 montre les différents éléments du modèle conceptuel de service SOA.

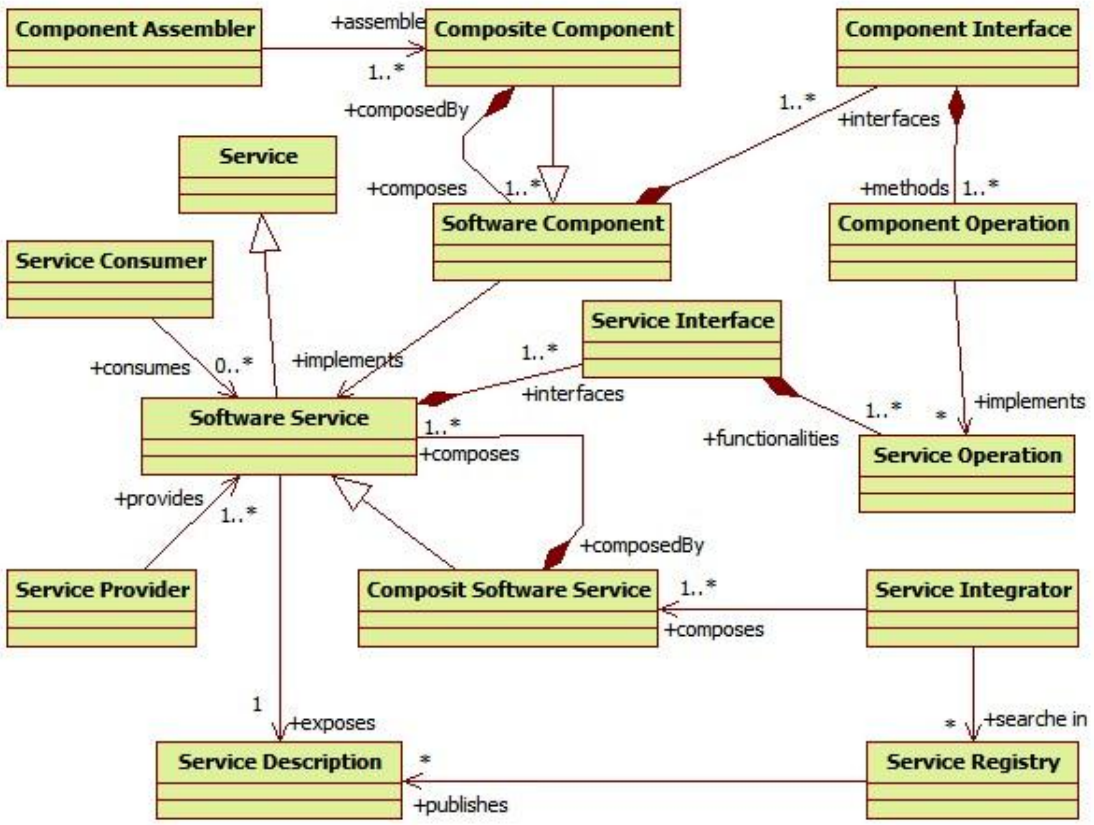


Figure 6.1 Modèle conceptuel de service simplifié de [PLASTIC01, 2008]

2.1 Spécification fonctionnelle de service

La figure 6.2 montre un modèle simple avec les spécifications fonctionnelles de service où la description de service décrit le service de sorte qu'elle puisse être directement consommée par des consommateurs de service ou puisse être combiné avec d'autres services pour produire l'application orientée services finale.

Les spécifications fonctionnelles de service sont des éléments d'une description de service qui spécifient les possibilités de service (capability) qui représentent des cas d'utilisation pour le service, par exemple, « acheter un livre ». Une fois que les services sont définis, un certain nombre de descriptions de processus métiers doivent être fournies. En particulier, pour chaque capability d'un grand service, un diagramme de description de processus métier devrait être spécifié afin de décrire les interactions entre les services impliqués. Chaque capability se rapporte à une ou plusieurs logiques métiers (c.-à-d., conversations), par exemple, « trouver le livre pertinent », et « m'envoyer ce livre ». La conversation « trouver le livre pertinent » peut encore se composer par une séquence de deux invocations d'opération : « obtenir une liste de mes achats précédents », et « proposer des recommandations en se basant sur ces achats ». La conversation « envoyer le livre à moi » peut encore se composer par une séquence de deux invocations d'opération : « payer en ligne », et « charger le livre à mon adresse ».

Une conversation (et par conséquent une opération de service) spécifie le type d'opération (par exemple, synchrone, asynchrone, etc.), ainsi que, la signature d'opération (par exemple, exceptions, paramètre de type de données, type d'entrée-sortie, pré- et post-conditions). Par exemple, une pré-condition pourrait être « la carte de crédit est-elle valide ?» et une post-condition pourrait être « le livre devrait être livré ».

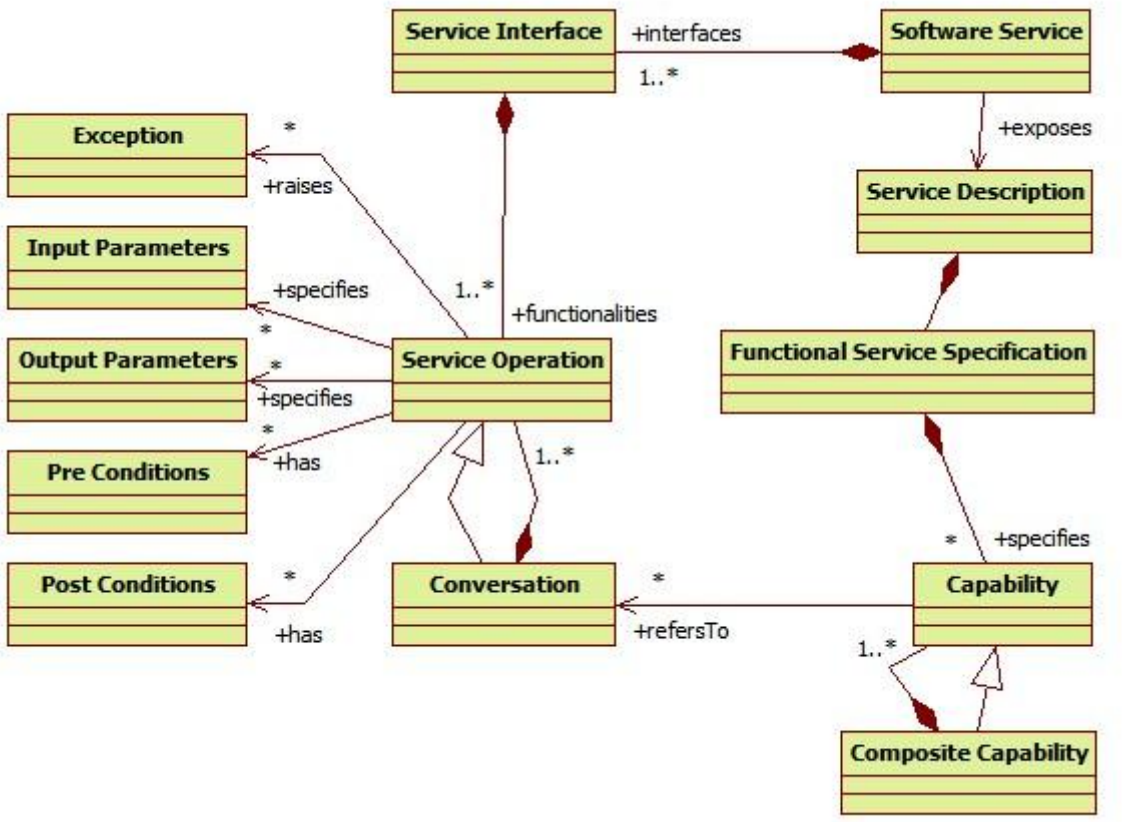


Figure 6.2 Spécification fonctionnelle de service [PLASTIC01, 2008]

2.2 Profile UML pour SOA

Un profile UML est un mécanisme d'extension basé sur les stéréotypes utilisé pour décrire les modèles spécifique à un domaine particulier dont les méta-classes des concepts de ce domaine ne sont pas prédéfinis dans l'infrastructure d'UML.

Pour faciliter la spécification automatique des services logiciels, nous devons décrire les concepts du modèle conceptuel de service d'une manière compréhensible par l'ordinateur. Ceci peut être fait en fournissant une description formelle écrite dans un langage d'ontologie (voir la figure 6.3). Le modèle conceptuel de service peut être vu comme une ontologie qui fournit une description formelle des termes de service convenablement fortement au développement et à l'exécution des services logiciels [PLASTIC02, 2007]. La Figure 6.3 représente les entités globales impliquées dans les activités de conceptualisation et de formalisation des services.

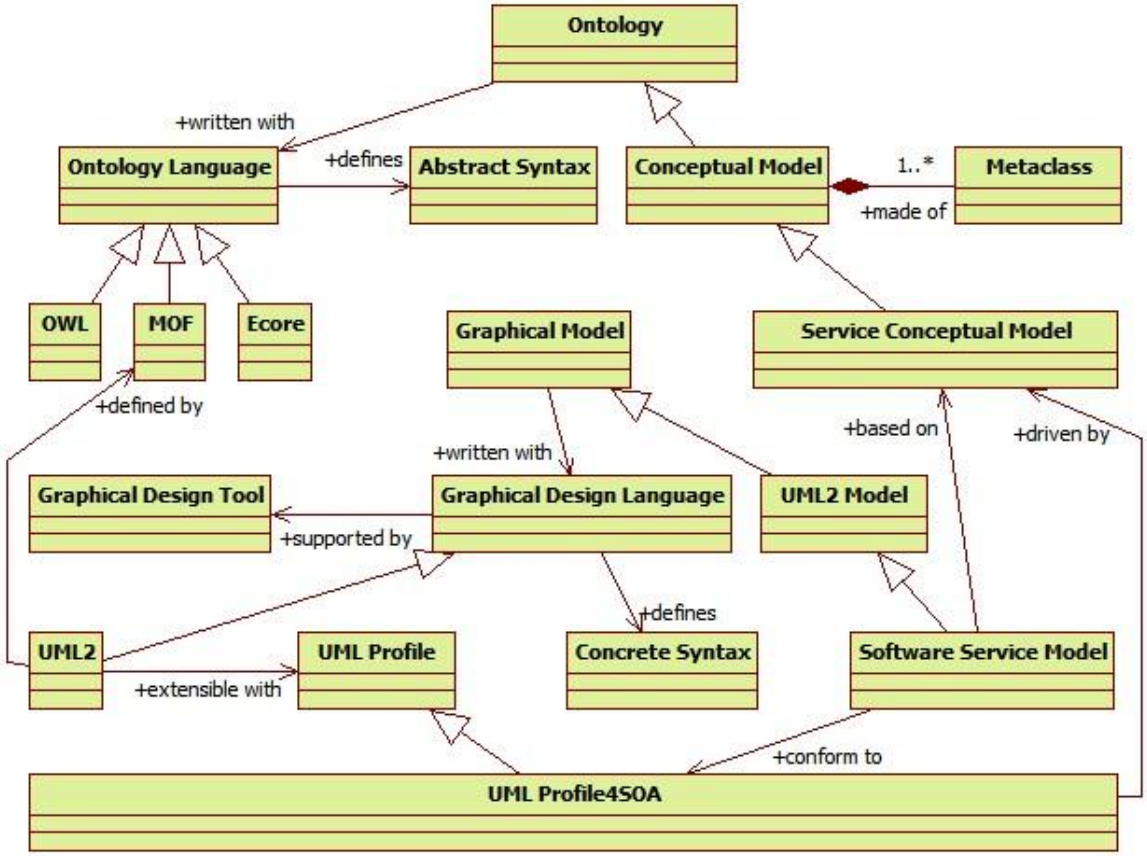


Figure 6.3 éléments de conceptualisation et de formalisation SOA [Simplifié de PLASTIC02, 2007]

Le modèle conceptuel de service peut être représenté en un langage d'Ontologie, tel que Meta Object Facility (MOF), Ecore, ou OWL.

Un langage de conception graphique associer une notation graphique aux éléments du langage d'ontologie qui doivent être exposés dans les modèles graphiques.

UML2 est un langage de modélisation graphique utilisé principalement pour le développement des logiciels. Il est écrit en langage d'ontologie MOF.

Pour faire face au problème de spécification des services, nous avons présenté un profile UML décrivant des modèles spécifiant la structure et le comportement des services. Nous avons concentré sur deux vues principale : la vue d'exigence et la vue de service. La vue d'exigence est décrite par un diagramme de cas d'utilisation de service et la vue de service est décrite par un ensemble de diagrammes présentant la structure et un autre ensemble diagrammes montrant le comportement des services. (Voir figure 6.4)

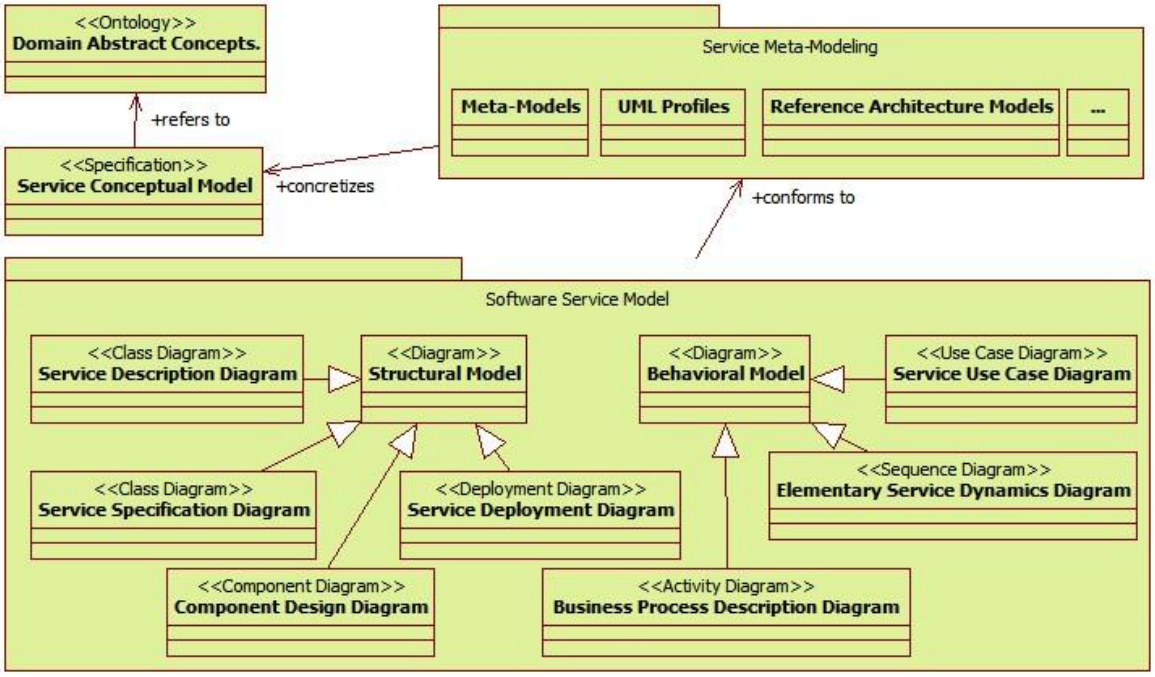


Figure 6.4 niveaux d'abstraction pour la modélisation des services [inspiré de PLASTIC02, 2007]

2.2.1 La vue d'exigence

La description d'une application orientée service commence toujours par la spécification des besoins en utilisant un ensemble de diagrammes de cas d'utilisation de service définis comme une extension des diagrammes de cas de l'utilisation d'UML2 (voir la figure 6.5).

Cinq acteurs peuvent être distingués durant le processus de développement des services : l'assembleur des composants, l'intégrateur des services, le développeur de service, le consommateur de service, et le fournisseur de service. Les cas d'utilisation spécifient des possibilités de service et se rapportent à des spécifications de conversation exprimé sous forme de diagramme d'activité étendu.

2.2.2 La vue de service

Une fois que la vue d'exigence a été produite, le développement se poursuit par la spécification des services qui servent à implémenter l'application orientée service à modéliser. De telle spécification est donnée par des perspectives structurales et comportementales. En particulier, une vue structural est donné par les diagrammes de description de service qui décrivent les services qui doivent être agrégés ou combinés pour produire l'application finale. La vue comportementale contient les spécifications des possibilités de service définies dans la vue d'exigence.

Le diagramme de description de service est un diagramme de classe personnalisé et les éléments de modélisation qu'il fournit sont montrés dans la figure 6.6 Le concept principal est « *Service Description* » qui est la base de l'unité structurale pour la description des applications orientées service. C'est un stéréotype étendant la méta-classe « Interface » d'UML2.

Le diagramme de description de service permet aussi de spécifier les services composés à l'aide du stéréotype « Service Composition ». C'est une prolongation d'une « Dépendance UML » et il est employé pour établir des liens entre la description des services atomiques et la description des services composés. D'ailleurs, un service peut utiliser d'autres services, et cette situation peut être modélisée à l'aide du stéréotype « Service Usage » qui est une extension de la relation de dépendance d'UML « Usage ».

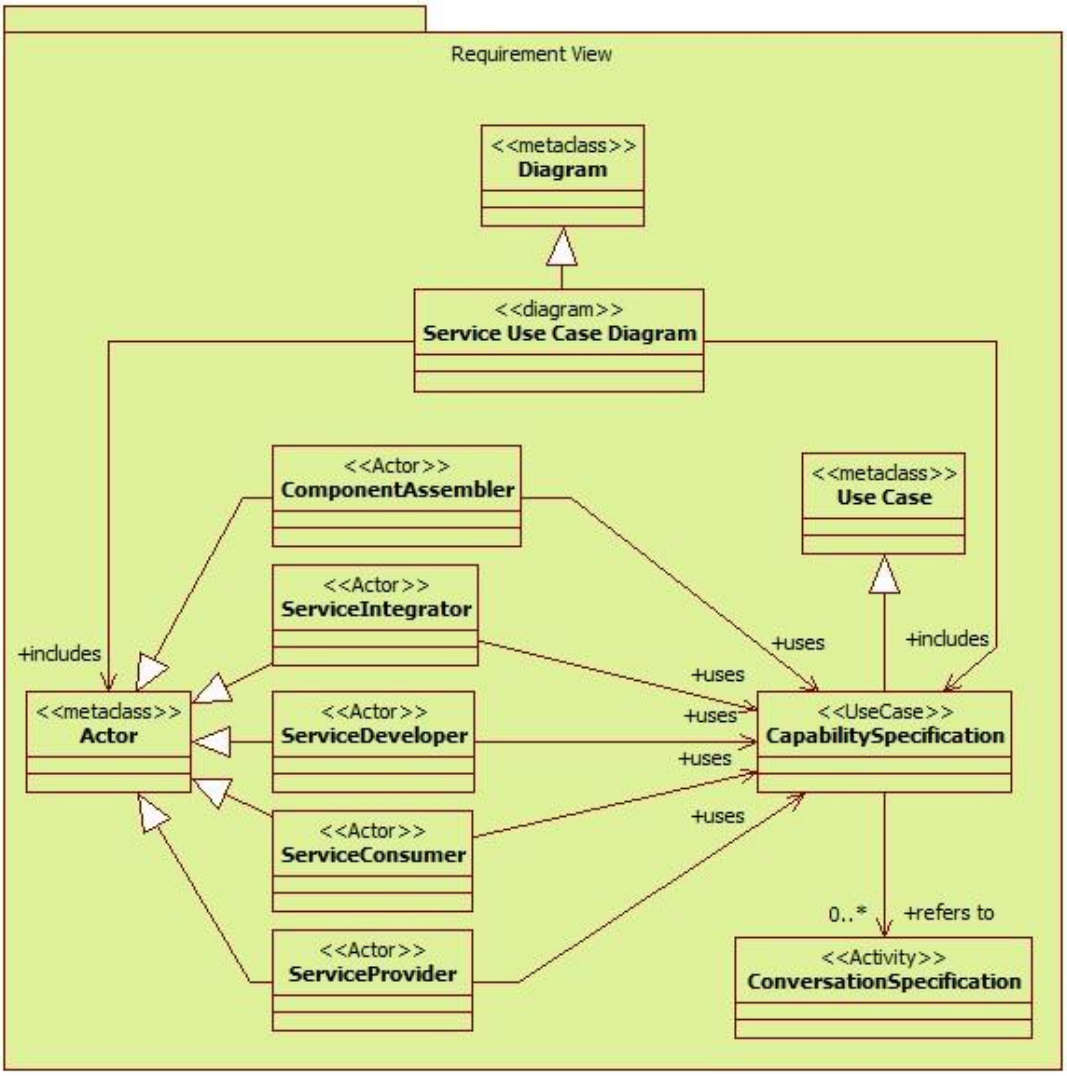


Figure 6.5 éléments de modélisation de diagramme de cas d'utilisation de service [PLASTIC02, 2007]

Une fois que tous les services sont définis, un certain nombre de descriptions de processus métier doivent être fournies. En particulier, pour chaque possibilité d'un grand service, un diagramme de description de processus métier doit être spécifié afin de décrire les interactions entre les petits services impliqués.

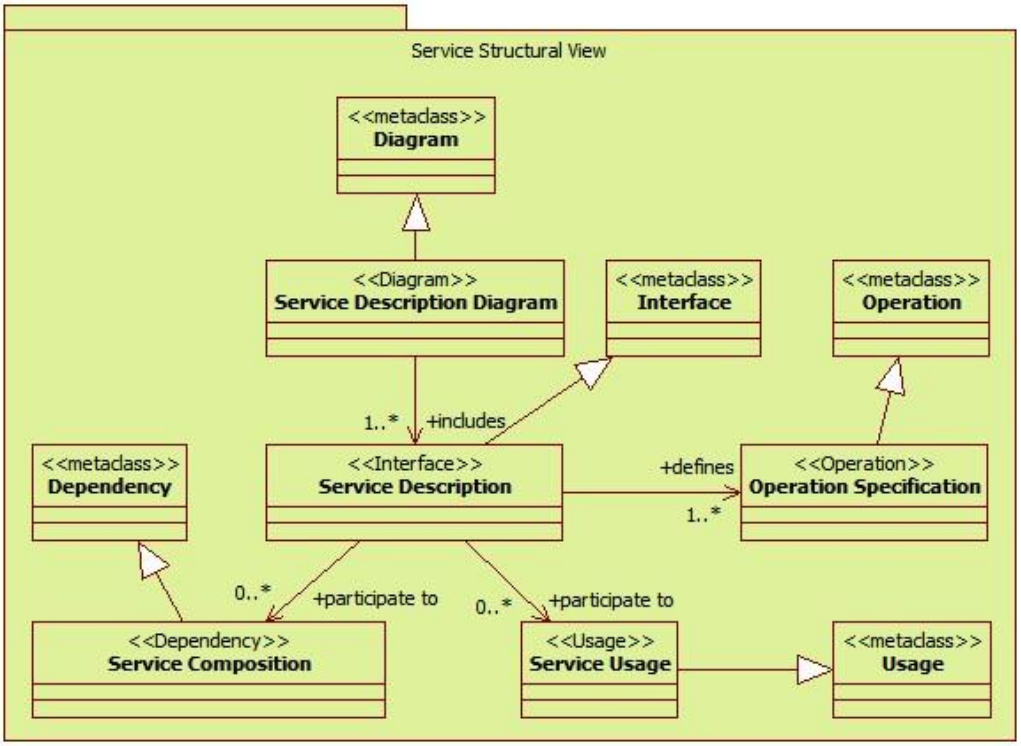


Figure 6.6 éléments de modélisation du diagramme de description de service [PLASTIC02, 2007]

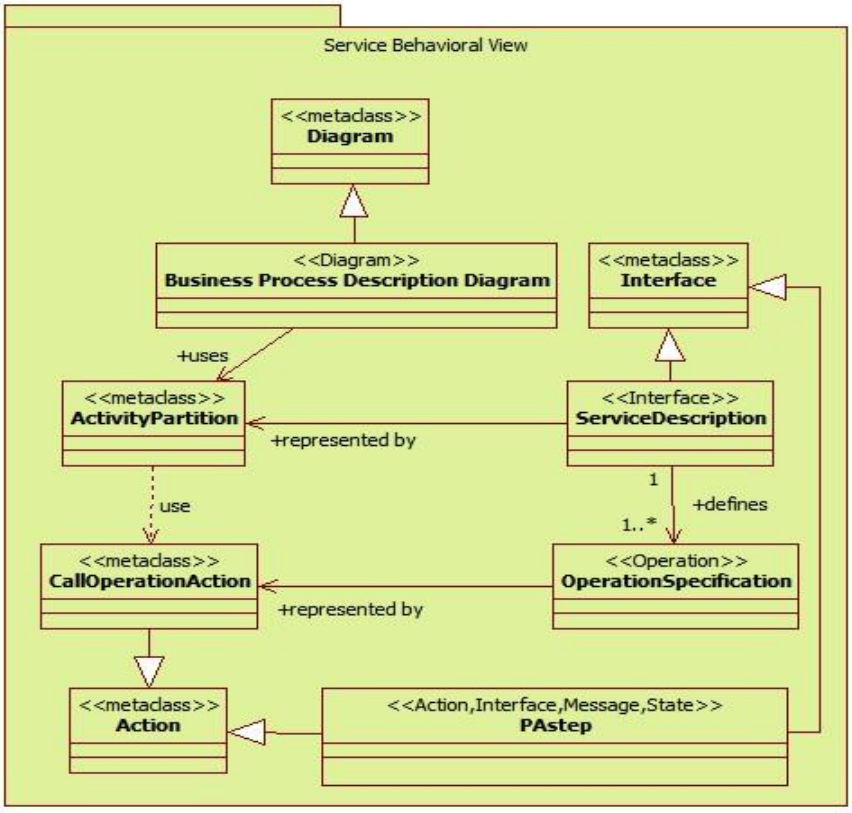


Figure 6.7 éléments de modélisation d'un diagramme de description de processus métier [PLASTIC02, 2007]

3. Processus de spécification automatique des services

La spécification des services logiciels est une activité qui consiste à décrire, sous forme de modèles conceptuels, la structure, le comportement interne de chaque service, ainsi que les interactions entre services. La première étape dans le processus de spécification des services est l'extraction des dépendances existante entres les activités métiers de l'entreprise. Trois modèles CIM sont impliqués dans cette étape :

(i) un diagramme de processus métier de haut niveau qui représente la logique métier des applications d'entreprise.
(ii) (ii) un diagramme de cas d'utilisation qui représente les différents acteurs et ses exigences.
(iii) (iii) et une ontologie de processus métier qui est considérée comme une base de connaissance constituée de tous les concepts, les propriétés, et les relations associés à la logique métier (par exemple, le type des activités, le niveau d'automatisation, le but de chaque activité etc.).

Après cette première étape, un moteur d'identification de service regroupe les activités métiers fortement dépendante dans des clusters en optimisant certain métriques tel que la cohérence et le couplage [Soltani et al, 2013]. Le résultat fournis par l'identificateur de service est un graphe de dépendance de clusters d'activités.

L'étape qui suive est une étape très importante, c'est la spécification des services qui consiste à représenter en UML la structure de chaque service (nom du service, les différentes opérations du service, les entrés et les sortie des opérations du service etc.) et la dynamique interne de chaque service. Un ensemble de modèles PIM sont générés par un modélisateur automatique de service durant cette phase tel que :

- Service Description Diagram ; est un diagramme de classe UML qui représente la structure des services.
- Service Use Case Diagram ; est un diagramme de cas d'utilisation qui décrit les acteurs et les capacités de chaque service.
- Business Process Description Diagram ; est un diagramme d'activités qui représente les interactions entre services.

Elementary Service Dynamics Diagram ; est un diagramme de séquence utilisé pour représenter le comportement interne des opérations de service. (La figure 6.8 illustre bien les phases de spécification de service).

La dernière phase dans le processus de spécification des services est la transformation des modèles PIM produits en modèles spécifique à la plateforme et la génération des codes correspondants.

Pour bien illustrer les étapes et les modèles de la spécification de service, nous avons étudié le cas du système d'enseignement à distance (SED) dont le but est de dériver automatiquement les services logiciels implémentant les activités de l'enseignant.

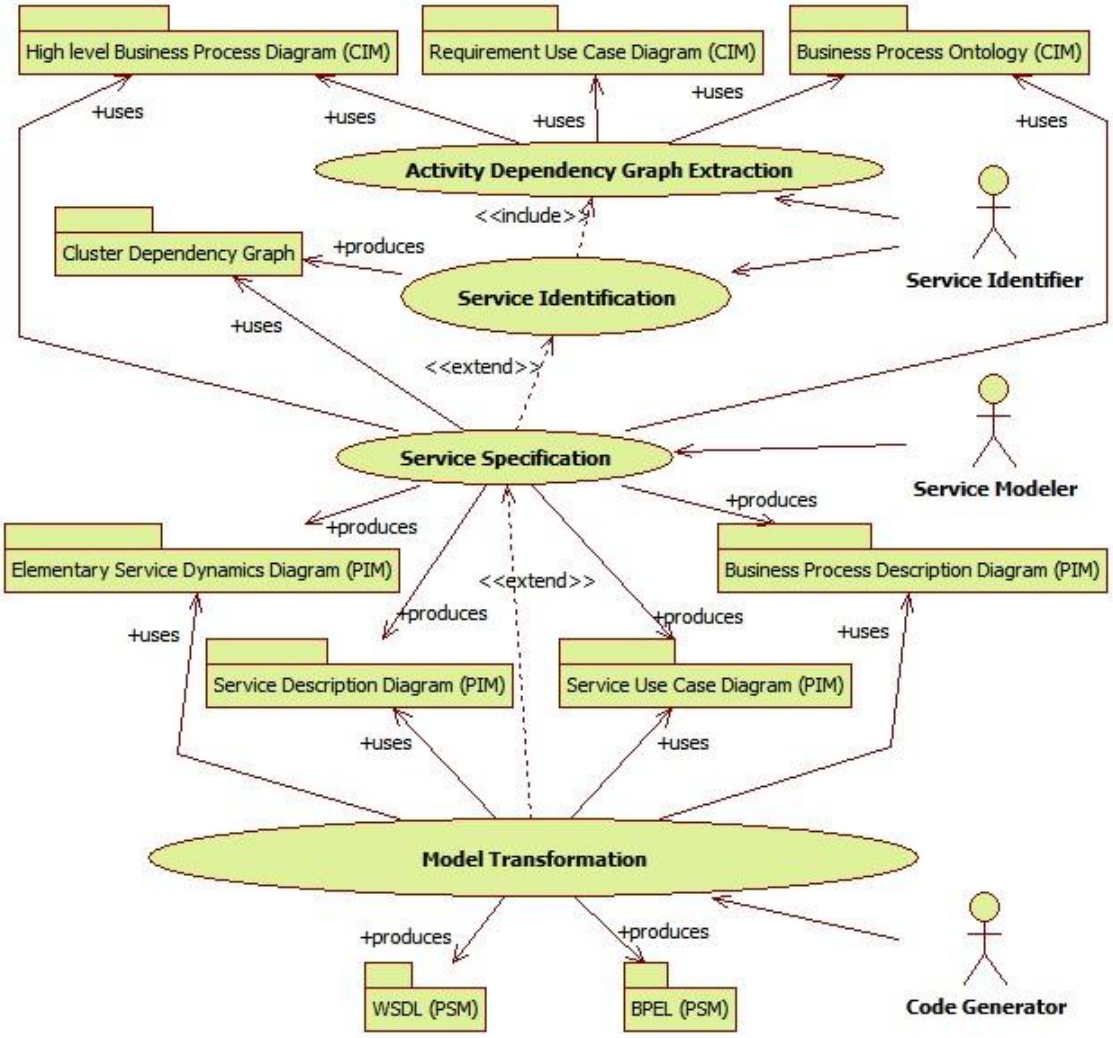


Figure 6.8 Processus de spécification automatique des services

4. Étude de cas : Spécification des services d'un Système d'Enseignement à Distance

Un système d'enseignement à distance (SED) est un système implémenté sur un réseau ouvert tel que le Web. Il a pour but de (i) faciliter les tâches de l'enseignant à travers l'automatisation des activités manuelles, (ii) réduire la distance entre l'enseignant et l'étudiant par l'implémentation des mécanismes de communication. (iii) augmenter la disponibilité des enseignants par l'intervention des professeurs qui travaillent dans des établissements qui sont géographiquement éloignés.

Trois acteurs principaux sont impliqués dans ce système: l'étudiant, l'enseignant, et l'administrateur. Pour chacun de ces acteurs, le système d'enseignement à distance fournit les fonctionnalités qui correspondent à ses besoins. Ainsi le système fournit les fonctionnalités à l'étudiant pour faire son inscription. Il lui permet de suivre les formations, d'accéder aux exercices, de poser les questions, d'envoyer des messages aux forums de discussion spécifiques à une formation donnée, de passer les tests et les

examens, et de consulter les notes. Il permet aussi aux étudiants de télécharger les documentations nécessaires (pdf, audio, vidéo, logiciels, etc.).

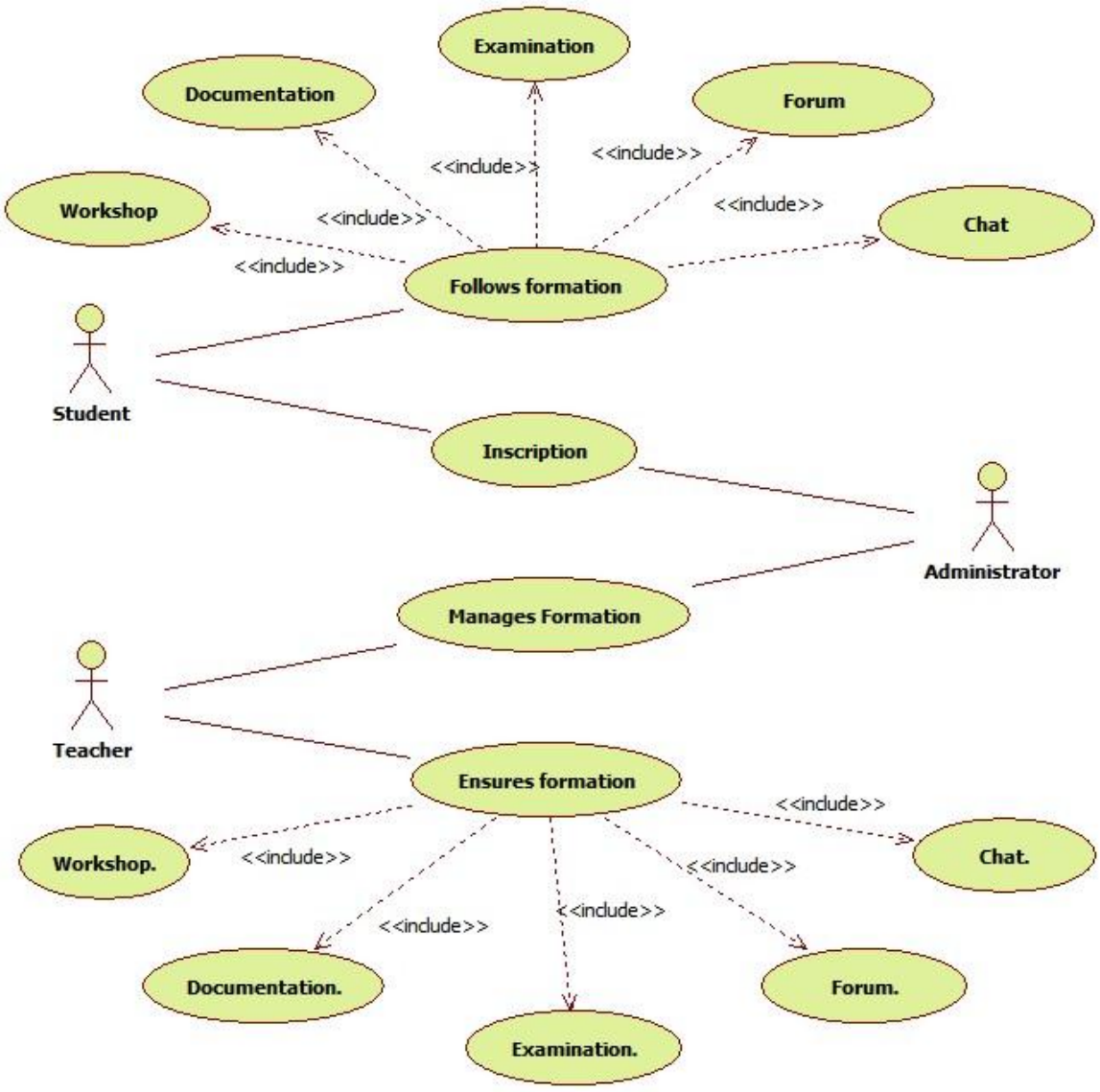


Figure 6.9 Système d'enseignement à distance (cas d'utilisation)

Quant à l'enseignant, il utilise le système pour créer des cours, pour répondre aux questions des étudiants, pour donner des exercices et pour répondre aux messages des forums de discussion. Le système offre aussi à l'enseignant la possibilité de charger la documentation concernant une formation spécifique ainsi que la possibilité de mettre à jours et/ou supprimer le contenu des cours.

Le système permet à l'administrateur de gérer les inscriptions des étudiants, de gérer le calendrier d'une formation et pour affecter les responsables des formations. Il permet de gérer, en collaboration avec l'enseignant, les sessions d'examens. La figure 6.9 illustre bien le SED par un diagramme de cas d'utilisation UML.

La figure 6.10 montre le processus métier « Assurer Formation » d'un enseignant. C'est un processus composé de sept sous processus qui s'exécutent en parallèles y compris la gestion des ateliers, la gestion de la documentation, la gestion des tests et d'examinassions, la gestion des forums, la participation dans un « Chat », la gestion des devoirs, et finalement la gestion des leçons.

Afin d'illustrer la spécification des services de ce système d'enseignement à distance, nous avons détaillé le sous-processus « gestion des leçons » dans la figure 6.11. Ce dernier est composé de douze activités parallèles : l'ajout d'une leçon dans la base de données des cours, la mise à jour d'une leçon existante, la suppression d'une leçon, l'ajout d'un exercice, la mise à jour d'un exercice, la suppression d'un exercice, ajouter une solution d'exercice, mettre à jour une solution d'un exercice, supprimer une solution, poser une question, lister les réponses des questions, consulter les solutions d'exercices proposées par les étudiants.

4.1 Extraction des dépendances d'activités

Pour identifier les services logiciels candidats à l'accomplissement de ces besoins métiers, nous avons commencé par l'extraction des dépendances existantes entre les activités métiers du sous-processus « gestion des leçons » à partir de la matrice CRUD présentée par le tableau 6.1. Cette dernière est composée de douze activités A0, A1, A2, A3, A4, A5, A6, A7, A8, A9, A10, A11 en lignes et cinq entités métiers en colonnes (leçon, exercice, exercice solution, question et réponse). Chacune de ces activités exécute quatre types d'opération sur les entités métiers. Ces opérations sont notées C: CREATE, R : READ, U : UPDATE, D : DELETE.

ID	Business Entities / Activities	Lessons	Exercice	Exercice Solution	Question	Answer
A0	T.Add lesson()	C				
A1	T.Add exercice ()		C			
A2	T.Add exercice solution ()			C		
A3	T.Ask question ()				C	
A4	T.Update lesson ()	U				
A5	T.Update exercice ()		U	U		
A6	T.Update exercice solution ()			U		
A7	T.Delete lesson ()	D				
A8	T.Delete exercice ()		D	D		
A9	T.Delete exercice solution ()			D		
A10	T.View question answers ()				R	R
A11	T.View exercice solutions ()			R		

Tableau 6.1 la matrice CRUD du SED

L'identification des services consiste à regrouper les activités fortement dépendantes dans un même groupe. La figure 6.12 montre les trois services candidats identifiés par notre moteur d'identification de services MOOSI. Le premier service (cluster A) est composé de trois activités A0, A4 et A7 intitulées respectivement « T.Add lesson », « T.Update lessons », et « T.Delete lessons ». Nous remarquons que ces trois activités participent à l'accomplissement du même but métier qui est « la gestion des leçons » (voir tableau 6.2)

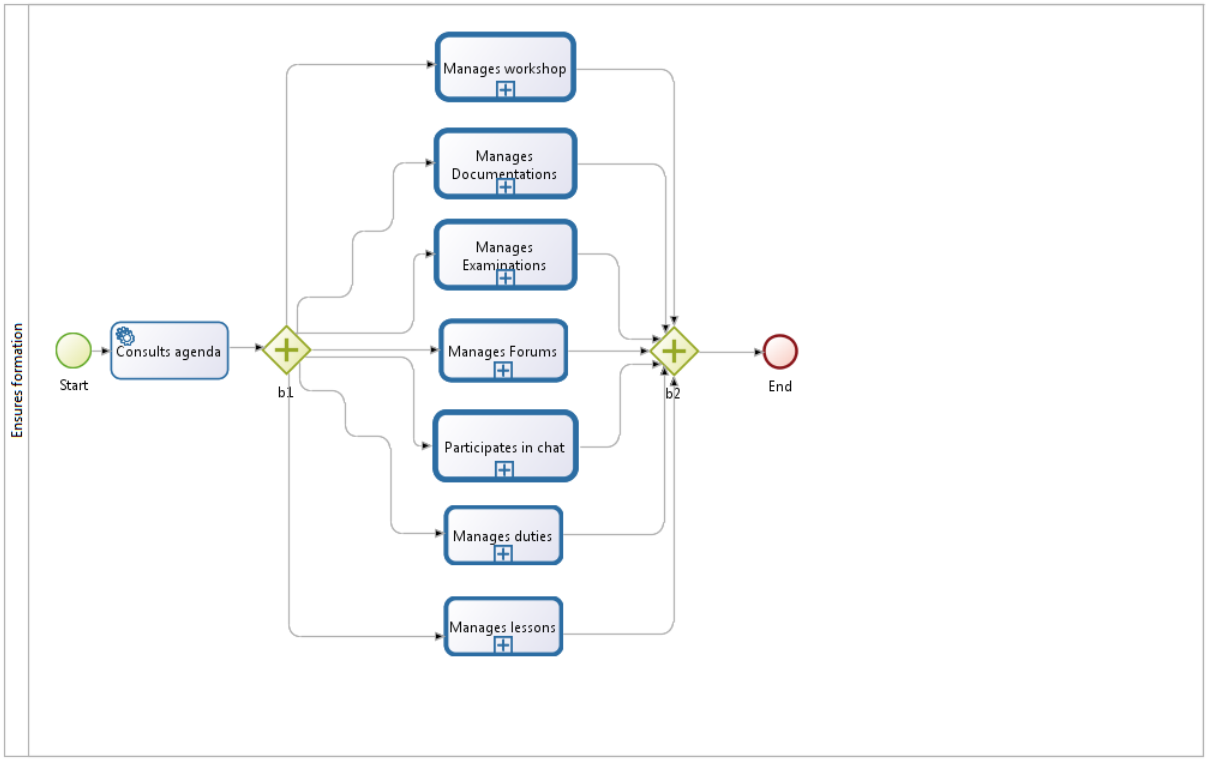


Figure 6.10 Processus métier « Assurer Formation »

Le deuxième service (Cluster B) est constitué de sept activités A1, A2, A5, A6, A8, A9, A11 nommées respectivement « T.Add exercice », « T.Add exercice solution », « T.Update exercice », « T.Update exercice solution », « T.Delete exercice », « T.Delete exercice solution », « T.View exercice solution ». Les sept activités du cluster B participent à l'accomplissement du même but métier qui est «la gestion des exercices ». Le dernier service candidat identifié par MOOSI (Cluster C) est composé par deux activités A3 et A10 nommées respectivement « T.Ask question » et « T.View question answers » les deux activités du cluster C participent à la réalisation du but métier « gérer question » afin d'évaluer la compréhension des étudiants.

Cluster ID	Activities (service operations)	Service Goal
A	A0: T.Add lessons() A4: T.Update lessons() A7: T.Delete lessons()	Manages lessons
B	A1:T.Add exercice() A2:T.Add exercice solution() A5: T.Update exercice() A6: T.Update exercice solution() A8: T.Delete exercice() A9: T.Delete exercice solution() A11: T.View exercice solution()	Manages exercices
C	A3: T.Ask question() A10:T.View question answers()	Manages questions

Tableau 6.2 Analyse des services identifiés

La deuxième phase est la spécification des services identifiés sous formes de diagrammes UML dont le but est de faciliter leurs implémentations à travers un générateur automatique de code. Le diagramme de cas d'utilisation des trois services identifiés est présenté dans la figure 6.13. La structure de ces services est illustrée par le diagramme de classe montré dans la figure 6.14

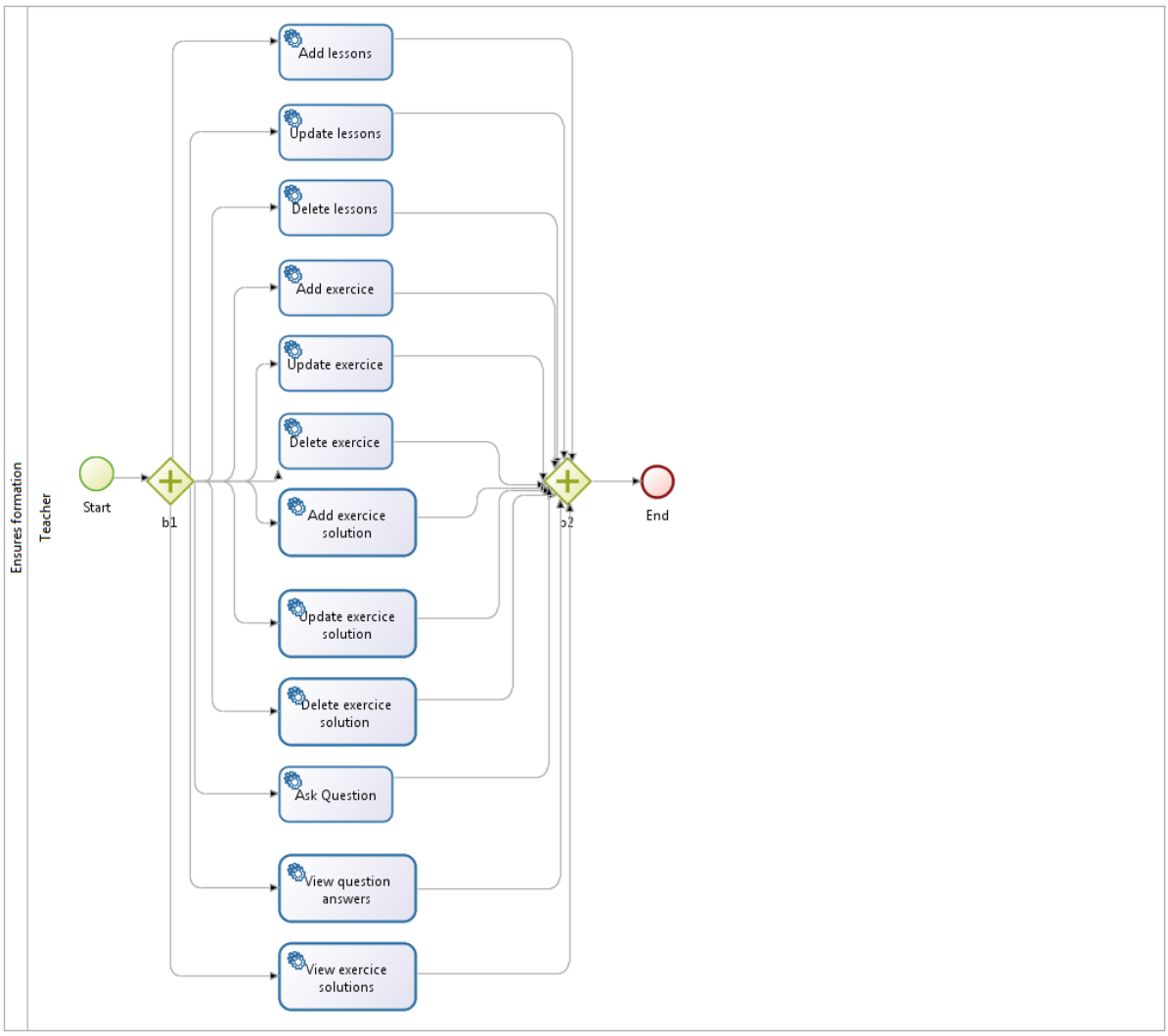


Figure 6.11 Sous-Processus métier « gérer leçon »

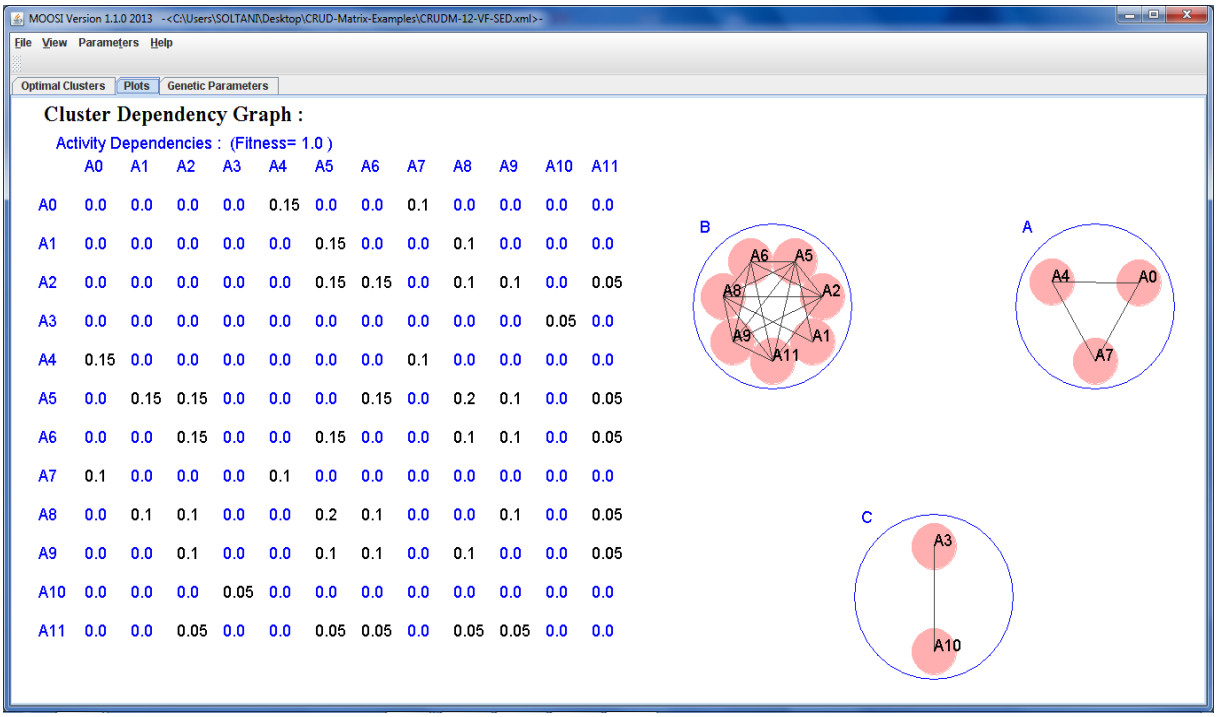


Figure 6.12 Identification des services du SED par MOOSI

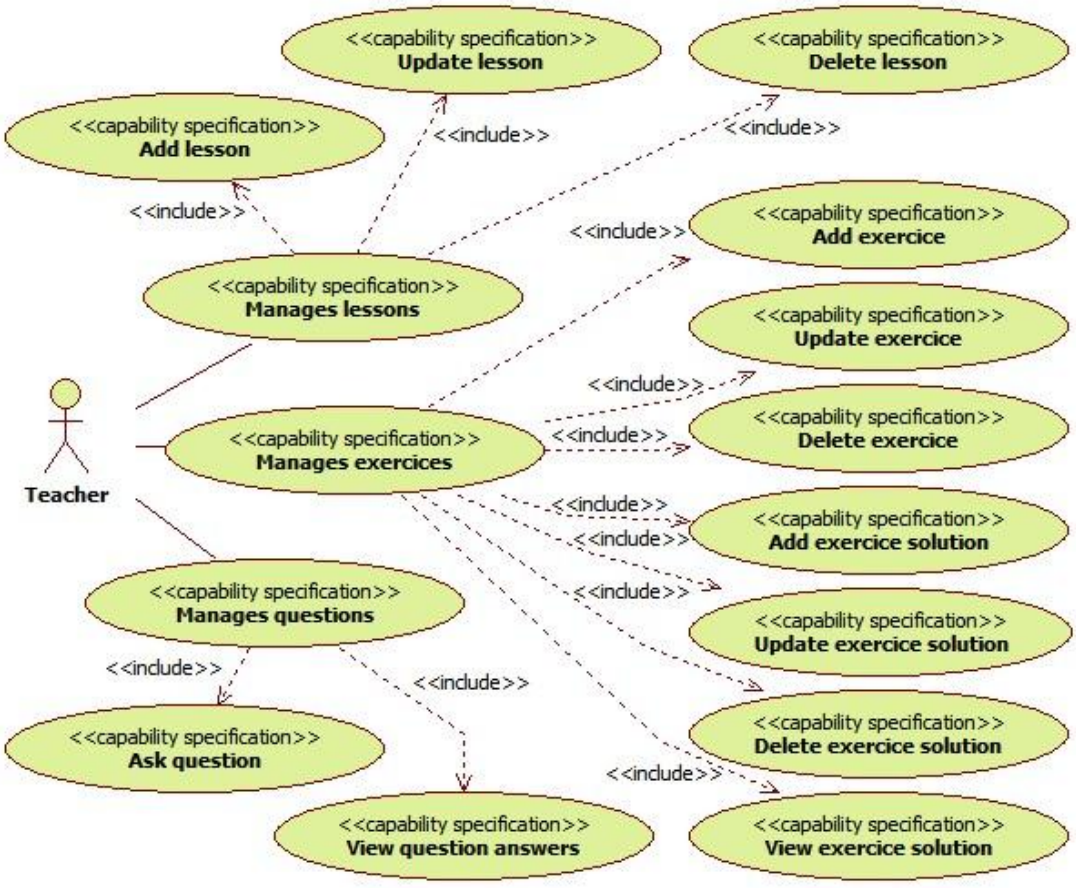


Figure 6.13 SED Service Use Case Diagram

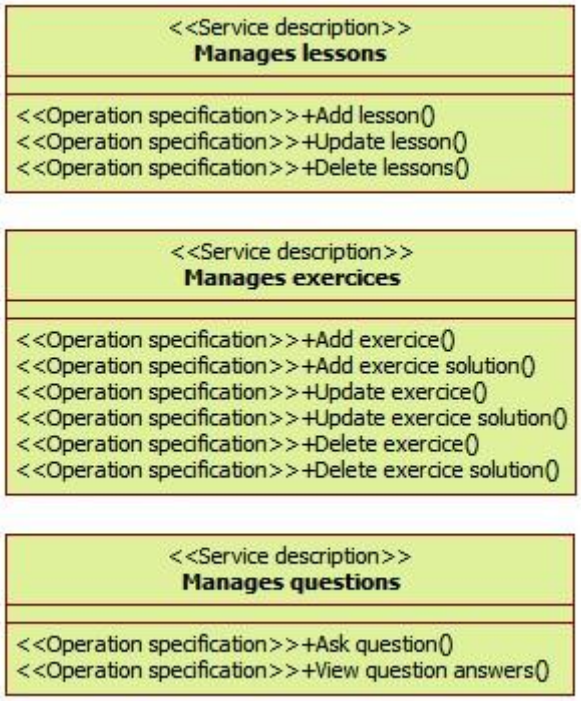


Figure 6.14 SED – Service Description Diagram

La dynamique intra- et inter services est illustrée dans les figures (figure 6.15 et figure 6.16)

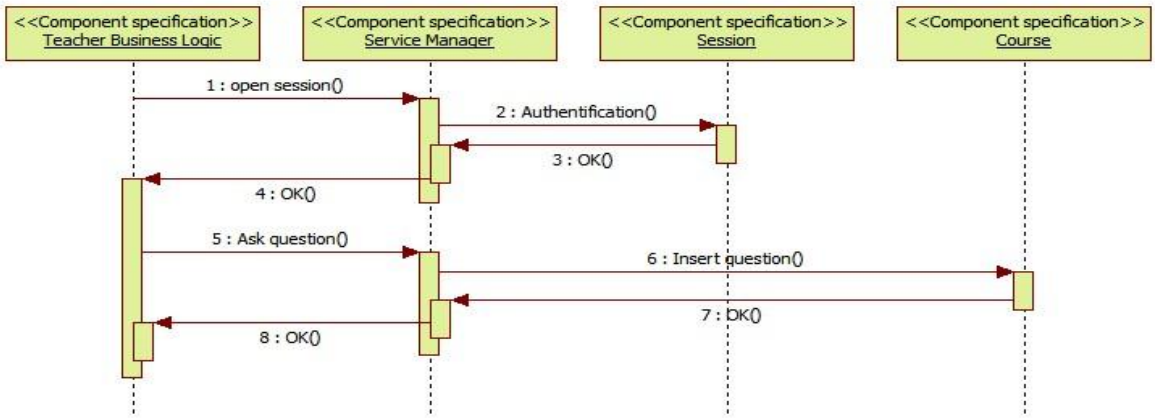


Figure 6.15 SED- Elementary Service Dynamics Diagram (Manages questions->Ask question ())

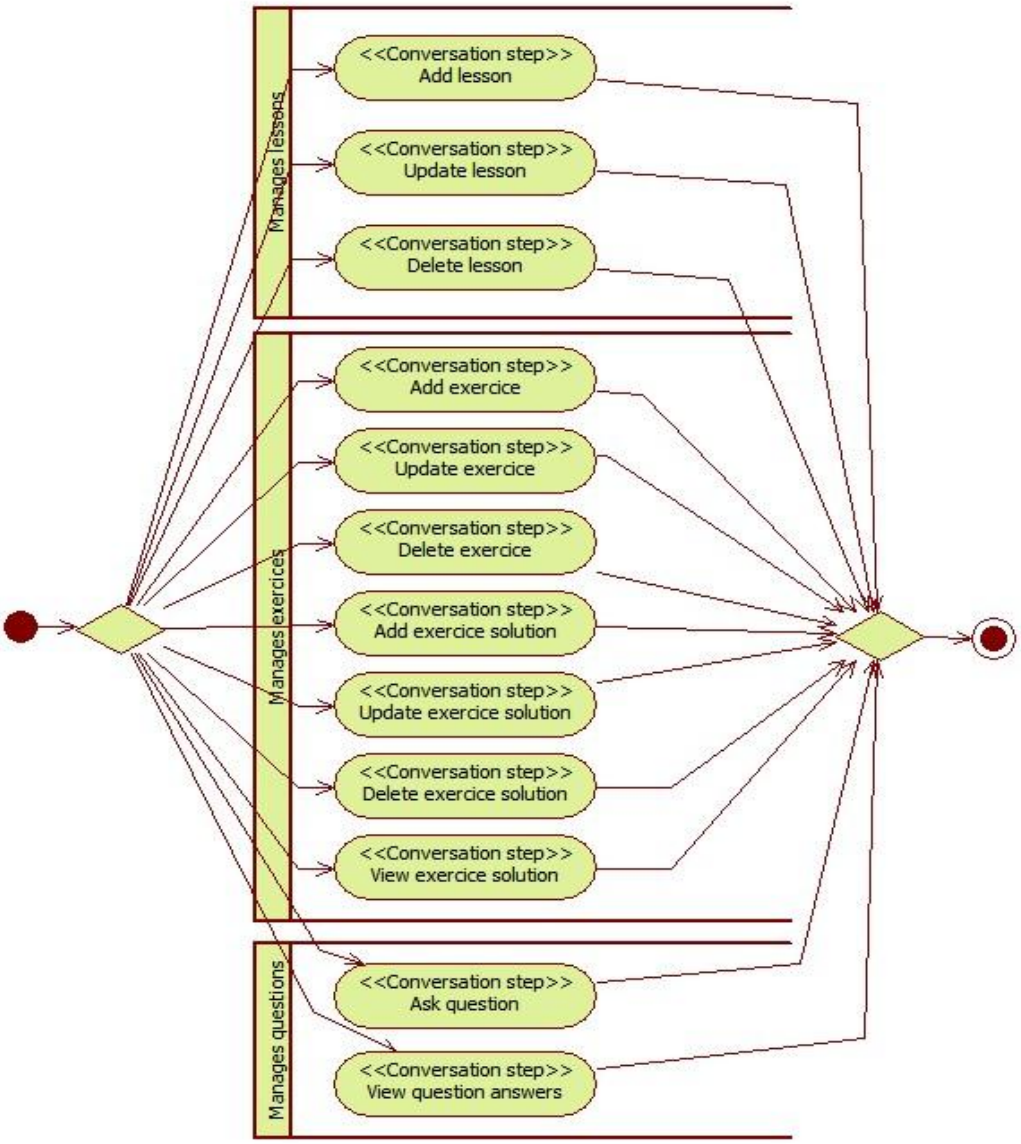


Figure 6.16 SED – Business Process Description Diagram

5. Conclusion

La modélisation des logiciels est une activité indispensable dans n'importe quel projet logiciel. Dans le présent chapitre, nous avons présenté les concepts de base nécessaire à la conceptualisation de SOA. Nous avons proposé une méthode de spécification automatique des services logiciels concrétisée par un modèle conceptuel SOA de haut niveau d'abstraction, un profile UML pour l'architecture orientée service, un processus de spécification automatique de services logiciels et finalement une étude de cas sur la spécification automatique des services d'un système d'enseignement à distance.

Nous avons remarqué que l'identification automatique des services en utilisant une approche descendante permet de produire des entités cohérentes de fonctionnalités qui peuvent être modélisés par la suite automatiquement afin de faciliter et accélérer la construction des services logiciels interopérables.

Conclusion générale et perspectives

Conclusion générale

Les travaux de recherche effectués dans le cadre de cette thèse portent sur la problématique de l'interopérabilité sémantique des systèmes d'information d'entreprise dirigée par les modèles. Nos travaux de recherche s'intéressent à la question de développement automatique d'une architecture interopérable en considérant que les modèles sont des éléments centraux de cette problématique.

Nous avons porté l'essentiel de nos efforts sur la définition d'une méthode qui permet de dériver automatiquement une architecture plus adaptée au besoin d'interopérabilité à partir d'un modèle métier de haut niveau.

En effet, l'adhésion d'une entreprise à des scénarios d'interopérabilité est régie par une double préoccupation. D'une part, l'entreprise présente un manque de flexibilité au niveau du système d'information et d'autre part, mettre en œuvre une interopérabilité exige le développement d'un cadre bien défini qui permettra aux entreprises d'interconnecter leurs différents processus d'une manière transparente.

L'architecture orientée services et la technologie des services Web semblent proposer des réponses crédibles au besoin d'interopérabilité. Nos travaux de recherche ont eu pour objectif principal de développer une nouvelle approche qui supporte la construction automatique d'une architecture orientée services au sein de l'entreprise.

Dans le cadre de cette thèse, nous avons concentré sur l'identification et la spécification des services logiciels appropriés au cadre d'une interopérabilité interentreprises. Les principales contributions que nous avons apportées durant la phase de préparation de cette thèse portent sur la définition d'une démarche d'identification automatique de services logiciels à partir d'un modèle de processus métier de haut niveau. La démarche d'identification appelée MOOSI (Multi-Objective Optimization-based Service Identification) permet de définir un ensemble de phases qui comportent des étapes nécessaires pour la dérivation d'une architecture de services. Il s'agit essentiellement d'une démarche qui vise à définir des services candidats dédiés à l'interopérabilité. Trois étapes pertinentes sont utilisées dans notre démarche. (i) Une Ontologie de processus métier notée(BPO) est adoptée pour annoter le modèle de processus métier d'entrée. (ii) Le modèle de processus annoté est considéré comme une entrée d'un moteur de transformation qui le transforme, après l'interrogation de l'ontologie, en processus exécutable. (iii) Enfin un moteur d'identification interroge l'ontologie pour extraire toutes les propriétés utilisées comme données d'entrée pour produire des services candidats automatiquement. Le moteur d'identification applique une méthode de clustering à base d'algorithme génétique. Les résultats d'implémentation prouvent que MOOSI peut atteindre une haute performance en termes de temps de calcul et une meilleure qualité de modularisation des services identifiés comparé à d'autres solutions.

Une fois les services logiciels sont identifiés, une phase fondamentale est nécessaire ; c'est la spécification des services identifiés. Pour réaliser cette phase, nous avons proposé une démarche bien détaillée concrétisée par un modèle conceptuel de service, un profile UML nécessaire à la modélisation des services identifiés, et une étude de cas sur la spécification automatique des services d'un système d'enseignement à distance.

Les prochains travaux seront focalisés sur la sémantisation automatique des services spécifiés à travers la définition des règles de transformation du modèle conceptuel de haut niveau qui représente les services en modèles de description des services Web sémantiques existants comme par exemple OWL-S et WSMO.

Bibliographies

Bibliographies

[Abdul et al, 2008] N. M. Abdul Latiff C. C. Tsimenidis, B. S. Sharif, C. Ladha." Dynamic Clustering using BinaryMulti-Objective Particle Swarm Optimization forWireless Sensor Networks". Personal, Indoor and Mobile Radio Communications, PIMRC 2008. IEEE 19th International Symposium on.2008.

[Alter, 2002] Alter, S, "Information Systems: The foundation of E-Business", PrenticeHall,NJ. 2002.

[Arsanjani et al, 2008] Arsanjani, A., S. Ghosh, A. Allam, T. Abdollah, S. Ganapathy et K. Holley. « SOMA: a method for developing service-oriented solutions ». IBM Systems Journal, vol. 47, no 3.2008.

[Arsanjani, 2004] Arsanjani, A. (2004). (SOMA) Service-oriented modeling and architecture: How to identify, specify, and realize services for your SOA. *SOA and Web Services Center of Excellence, IBM, Software Group*.

[ATHENA01, 2007] *Deliverable Number: D.B3.5 / D.B3.6 : ATHENA Contribution to Interoperability Policy Action Plan Version 3 and Long-Term Research Recommendations version 2*, ATHENA European Integrated Project, 2007.

[ATHENA02, 2007] *Deliverable Number : D.A4.2 : Specification of Interoperability Framework and Profiles, Guidelines and Best Practices* ATHENA European Integrated Project, 2007, p. 215.

[ATHENA03, 2005] *D.A1.3.1: Report on Methodology description and guidelines definition, Version 1.0*, 2005.

[Azevedo et al, 2009] Azevedo, L. G., Santoro, F., Baiao, F., Souza, J., Revoredo, K., Pereira, V., & Herlain, I. (2009). A method for service identification from business process models in a SOA approach. In *Conference on Advanced Information Systems Engineering (CaiSE)* (pp. 99–112).

[Baïna et al, 2006] Baïna, S., Panetto, H., & Benali, K. (2006). Apport de l'approche MDA pour l'interopérabilité des systèmes d'entreprise. *Ingénierie des systèmes d'information. 11*(3), 11–29.

[Baïna, 2006] Baïna, S. (2006). *Interopérabilité dirigée par les modèles: Une approche orientée produit pour l'interopérabilité des systèmes d'entreprise*. Doctoral dissertation, University of Henri Poincaré, Nancy I.

[Bellwood et al., 2002] Bellwood, T., Clement, L., Ehnebuske, D., Hately, A., Hondo, M., Husband, Y., Januszewski, K., Lee, S., McKee, B., Munter, J., and von Riegen, C. (2002). UDDI version 3.0. http://uddi.org/pubs/uddi-v3.00-published-20020719.htm.

[Belouadha et al, 2010] Belouadha, F.-Z., H. Omrana, and O. Roudies, A model-driven approach for composing SAWSDL semantic Web services.

[Amar Bensaber et Malki, 2008] Amar Bensaber, D. and M. Malki. Development of semantic Web services: Model driven approach. 2008. Lyon, France: Association for Computing Machinery.

[Berre et al, 2004] Berre A.-J., Hahn A., Akehurst D., Bezivin J., Tsalgatidou A., Vermaut F., Kutvonen L., F. Linington P., *State of the art for interoperability architecture approaches*, INTEROP, 2004, p. 362, disponible sur: http://interop-vlab.eu/ei_public_deliverables/interop-noe-deliverables/dapdomain-architecture-and-platforms/D91/?searchterm=d9.1.

[Bianchini et al, 2013] Bianchini Devis, Cinzia Cappiello, Valeria De Antonellis, Barbara Pernici. « Servic identification in inter-organizational process design ». IEEE Transactions on Services Computing, 20 May 2013.

[Birkmeier et al, 2013] Dominik Q. Birkmeier, Andreas Gehlert, Sven Overhage, Sebastian Schlauderer. "Alignment of Business and IT Architectures in the German Federal Government:A Systematic Method to Identify Services from Business Processes". 46th Hawaii International Conference on System Sciences, 2013.

[Boerner et Goeken, 2009] Boerner, R., et M. Goeken. « Service identification in SOA Governance literature review and implications for a new method ». In., p. 588-93. Coll. « 2009 3rd IEEE International Conference on Digital Ecosystems and Technologies (DEST) ». Piscataway, NJ, USA: IEEE. 2009.

[Bondé, 2006] Bondé L. Transformations de Modèles et Interopérabilité dans la Conception de Systèmes Hétérogènes sur Puce à Base d'IP. Thèse de doctorat, Université des Sciences et Technologies de Lille, 2006.

[Bourey et al, 2007] Bourey J.-P., Grangel R., Doumeingts G., Berre A. J., Deliverable DTG2.3 : Report on Model Driven Interoperability, I. Society, InterOP, 2007, p. 91, disponible sur: www.interop-noe.org

[Brambilla et al, 2007] Brambilla, M., et al., Model-driven design and development of semantic Web service applications. ACM Transactions on Internet Technology, 2007. 8(1).

[Briol, 2008] P. Briol. « Ingénierie des processus métiers, De l'élaboration à l'exploitation » Lulu. Com, 2008.

[Catton, 2000] M. Catton, "Management des processus, une approche innovante", Paris : Afnor, 2000.

[Chen et al, 2006] Chen D., Dassisti M., Elvesæter B., *Interoperability Knowledge Corpus, Intermediate Report. Deliverable DI.1b*, Network of Excellence InterOp, Contract No.IST-508011, 2006.

[Chen et al, 2007] Chen D., Dassisti M., Elvesæter B., Deliverable DI.3 : Enterprise Interoperability Framework and knowledge corpus Final report INTEROP, 2007, p. 44, disponible sur: http://interopvlab. eu/ei_public_deliverables/interop-noe-deliverables/delivlist.

[Chen et Daclin, 2006] Chen D., Daclin N., *Framework for enterprise interoperability*, *IFAC TC5.3 workshop EI2No6*, Bordeaux, France, 2006.

[Chen et Daclin, 2010] Chen D., Daclin N., *Framework for Enterprise Interoperability,* (Interoperability for Enterprise Software and Applications), ISTE, 2010, p. 77-88, ISBN. 9780470612200, disponible sur: http://dx.doi.org/10.1002/9780470612200.ch6.

[Chergui, 2013] M. E. Chergui, S. M. Benslimane. Using Combinatorial Particle Swarm Optimization to Automatic Service Identification. 13th International Arab Conference on Information Technology ACIT'2013, December 17-19, 2013, Khartoum, Sudan.

[Christensen et al., 2001] Christensen, E., Curbera, F., Meredith, G., and Weerawarana, S. (2001). Web Services Description Language (WSDL) 1.1. Note, W3C.

[Collette et Siarry, 2002] Yann Collette, Patrick Siarry "Optimisation multiobjectif", EYROLLES, 2002.

[COOREN 2008] Yann COOREN,"Perfectionnement d'un algorithme adaptatif d'Optimisation par Essaim Particulaire. Applications en génie médical et en électronique", Thèse De Doctorat, De L'université Paris 12 Val De Marne,2008.

[Courat 1994] J. P. Courat, G. Raynaud, I. Mrad, and P. Siarry. "Electronic component model minimization based on Log Simulated Annealing".1994.

[Dan et al, 2008] Dan, Asit, Robert D. Johnson et Tony Carrato. « SOA service reuse by design ». In., p. 25-28. Coll. « Proceedings - International Conference on Software Engineering ». Leipzig, Germany: Inst. of Elec. and Elec. Eng. Computer Society.2008.

[Daniel, 2003] Jérôme Daniel. Extrait de livre "SERVICES WEB Concepts, techniques et outils" Edition vuibert Informatique 2003.

[Debauche et Mégard, 2004] B. Debauche and P. Mégard. BPM: Business Process Management: pilotage métier de l'entreprise. Hermes Science Publications, 2004.

[DongSu et al, 2008] DongSu, Kang, Song Chee-yang et Baik Doo-Kwon.« A method of service identification for product line ». In. Vol. vol.2, p. 1040-5. Coll. « 2008 Third International Conference on Convergence and Hybrid Information Technology (ICCIT) ». Los Alamitos CA, USA: IEEE Computer Society. 2008.

[Dorigo et al., 1996] M. Dorigo, V. Maniezzo, A. Colorni. (1996). Ant System : optimization by a colony of cooperating agents, IEEE Transactions on Man. Cyber., Part B, Vol. 26, N°1, pp. 29-41, 1996.

[Dufresne et Martin, 2003] Thomas Dufresne and James Martin, "Process Modeling for E-Business", INFS 770 – Methods for Information Systems Engineering: Knowledge Management and EBusinessm.Spring 2003.

[Dussart et al, 2004] Aymeric Dussart, Benoit Aubert, and Michel Patry. "An evaluation of interorganizational workflow modeling formalisms".JDM.2004

[Dwivedi et Kulkarni 2008] Dwivedi, V., et N. Kulkarni.« A model driven service identification approach for process centric systems ». In., p. 65-72. Coll. « 2008 IEEE Congress on Services Part II (SERVICES-2) ». Piscataway, NJ, USA: IEEE. 2008.

[EIF, 2004] EIF, *European Interoperability Framework for pan-European eGovernment services V1.0*, La commission Européenne, Bruxelles, 2004, p. 79, disponible sur: http://ec.europa.eu/idabc/en/document/3473/5585.html.

[EL DOR, 2012] Abbas EL DOR, "Perfectionnement des algorithmes d'Optimisation par Essaim Particulaire. Applications en segmentation d'images et en électronique", Thèse de doctorat, Université Paris-Est 2012.

[ElMansouri, 2009] El Mansouri Raida.«Modélisation et Vérification des processus métiers dans les entreprises virtuelles : Une approche basée sur la transformation de graphes ». Phd thesis. Université Mentouri Constantine2009.

[Erl 2008] Erl, Thomas. *SOA : Principles of Service Design*. Prentice Hall PTR. 2008.

[Erl, 2005] T. Erl, *Service-Oriented Architecture: Concepts, Technology and Design*. Prentice Hall PTR, 2005.

[Erradi et al, 2006] A. Erradi, S. Anand, and N. Kulkarni, SOAF: an architectural framework for service definition and realization. In: *IEEE International Conference on Services Computing (SCC 2006)*, 18-22 September 2006, Chicago, Illinois, USA. IEEE Computer Society, 2006, 151-158.

[Erradi et al, 2006] Erradi, Abdelkarim, Naveen Kulkarni et Sriram Anand. 2006. « Service design principles: A case study in modeling services for the securities trading domain ». Computer Systems Science and Engineering, vol. 21, no 4.

[Esper, 2010] Alida ESPER, « Intégration des approches SOA et orientée objet pour modéliser une orchestration cohérente de services », Thèse de Doctorat, INSA Lyon 2010.

[Feng et al, 2005] Feng Chen, Shaoyun Li and Hongji Yang ,Ching-Huey Wang ,William Cheng-Chung Chu." Feature Analysis for Service-Oriented Reengineering". Proceedings of the 12th Asia-Pacific Software Engineering Conference 2005.

[Gacitua-Decar et al, 2009] V. Gacitua-Decar and C. Pahl, "Automatic Business Process Pattern Matching for Enterprise Services Design", IEEE 5CC - Services II - Cloud II @ SOPOSE.2009.

[Gaubert-Macon, 2006] Christine Gaubert-Macon, "Approche des processus organisationnels et modélisation", http://www.reseaucerta.org © CERTA - juillet 2006 – v1.0.

[Gherbi et al, 2009] I.B. Tahar Gherbi, Djamel Meslati, « MDE between Promises and Challenges », 11th International Conference on Computer Modelling and Simulation, 2009.

[Gillot, 2008] J. Gillot." The Complete Guide to Business Process Management: BusinessProcess Transformation Or a Way of Aligning the Strategic Objectives of the Company and the Information System Through the Processes". Lulu. com, 2008.

[Girish et al, 2007] Girish, Juneja, Dournaee Blake, Natoli Joe et Steve Birkel. 2007. Service Oriented Architecture Demystified, 1st. Coll. « IT Best Practices Series ». Intel Press, 326 p.

[Glover, 1986] F. Glover. (1986). Future paths for integer programming and links to artificial intelligence, Computers and Operations Research, Vol. 13, pp. 533-549, 1986.

[Grnmo et al, 2005] Gronmo, R., M.C. Jaeger, and H. Hoff. Transformations between UML and OWL-S. 2005. Nuremberg, Germany: Springer Verlag.

[Gunnar et al, 2012] Gunnar, T., & Gottfried, V. (2012). Identification, specification, and development of web-oriented architectures. [IJISSS]. *International Journal of Information Systems in the Service Sector*, *4*(1), 1–21. doi:10.4018/jisss.2012010101

[Han et al, 2009] Han F., Moller E., Berre A. J., *Organizational interoperability supported through goal alignment with BMM and service collaboration with SoaML*, *The 2009 International Conference on Interoperability for Enterprise Software and Applications - IESA '09*, Chine, 2009, p. 268 – 274.

[Harris, 2007] Harris, Torry. « SOA Test Methodology ». Torry Harris.Boerner, R., et M. Goeken. 2009. « Service identification in SOA Governance literature review and implications for a new method ». In., p. 588-93. Coll. « 2009 3rd IEEE International Conference on Digital Ecosystems and Technologies (DEST) ». Piscataway, NJ, USA: IEEE.2007.

[Hernández, 2007] Rubén Lara Hernández. A Flexible Model for the Semi-automatic Location of Services. A dissertation presented in The Department of Computer Science in partial fulfillment of the requirements for the degree of Doctor of Philosophy in the subject of Computer Science, Universidad Autónoma de Madrid, Spain, July 2007.

[Heubès, 2007] Heubès, Christophe. « Mise en œuvre d'une SOA : Les clés du succès ». In Blog Xebia France : J2EE, Agilité et SOA. <http://blog.xebia.fr/2007/08/16/Mise enoeuvre- dune-soa-les-cles-du-succes/>. 2007.

[Hoidn, 2012] Hans-Peter Hoidn. « Enterprise IT Architectures: SOA (Service Oriented Architecture)" white paper, IBM Corporation, 2012.

[Huysmans et al, 2007] Huysmans, Philip, Jan Verelst et Herwig Mannaert. « Towards systematic identification of services: A domain-specific approach ». In, WSEHS/-. Vol. ISDM, p. 267-277. Coll. « ICSOFT 2007 - 2nd International Conference on Software and Data Technologies, Proceedings ». Barcelona, Spain: INSTICC Press.2007.

[IBM01, 2012] *Introduction to RUP*, IBM Corp. [cited April 2012]. Available from: http://www.michael-richardson.com/rup_classic/#core.base_rup/guidances/supportingmaterials/introduction_to_rup_36B63436.html.

[IDEAS, 2003] IDEAS, *A Gap Analysis - Required Activities in Research, Technology and Standardisation to close the RTS Gap - Roadmaps and Recommendations on RTS activities*, 2003.

[IEEE, 1990] *Standard Computer Dictionary- A Compilation of IEEE Standard Computer Glossaries*, New York, 1990, p. 215, ISBN: ISBN 1-55937-079-3

[Iheb, 2011] Iheb Abdellatif, "Vers Une Démarche D'aide À La Décision Pour L'identification DesServices D'une Architecture Orientée Services", Mémoire de maîtrise, Université Du Québec 2011.

[Inaganti et al, 2007] Inaganti, S., & Behara, G. K. (2007). Service identification: BPM and SOA handshake. *White Paper*.

[ISO, 2000] ISO. Quality Management Standard (ISO9001:2000). Rapport technique, International Organization for Standardization, December 2000.

[ISO-14258, 1998] *ISO-14258. Industrial automation systems and integration – Concepts and rules for enterprise models,*, 1998.

[Jain et al, 2004] Jain, H. Zhao, and N.R. Chinta, "A Spanning Tree Based Approach to Identifying Web Services, International Journal of Web Services Research 1, No.1.2004

[Jamshidi et al, 2008] Jamshidi, P., Sharifi, M., & Mansour, S. (2008, July 8–11). To establish enterprise service model from enterprise business model. In *Proceedings of the IEEE International Conference on Services Computing (SCC 2008)*, Honolulu, HI (pp. 93–100).

[Jamshidi et al, 2012] Jamshidi P, Mansour S, Sedighiani K, Jamshidi S, and Shams F., "*An Automated Service Identification Method*", Technical Report. Department of Electrical and Computer Engineering. Shahid Beheshti University, 2012.

[Jarboui et al, 2007] B. Jarboui , M. Cheikh , P. Siarry, A. Rebai , "Combinatorial particle swarm optimization (CPSO) for partitional clustering problem", Applied Mathematics and Computation 192, 337–345, Elsevier 2007.

[Jarboui et al, 2008] Bassem Jarboui , Saber Ibrahim , Patrick Siarry , Abdelwaheb Rebai."A combinatorial particle swarm optimization for solving permutation flowshop problems". Computers & Industrial Engineering 54 526–538(2008).

[Kapil et Matjaz ,2008] Kapil, Pant, et Juric Matjaz. "Business Process Driven SOA using BPMN and BPEL:From Business Process Modeling to Orchestration and Service OrientedArchitecture".2008.

[Karthi et al, 2009] R. Karthi, S. Arumugam, and K. Ramesh Kumar." Discrete Particle Swarm OptimizationAlgorithm for Data Clustering". Nature Inspired Cooperative Strate. for Optimization, SCI 236, pp. 75–88. Springer-Verlag Berlin Heidelberg 2009.

[Kazemi et al, 2011] Kazemi, A., Rostampour, A., Jamshidi, P., Nazemi, E., Shams, F., & Azizkandi, A. N. A genetic algorithm based approach to service identification. In *Proceedings of the IEEE World Congress on Services,* Washington, DC. (2011, July 4-9).

[Kennedy et al, 1995] J. Kennedy, R.C. Eberhart," Particle swarm optimization", in: Proceedings of the IEEE International Conference on Neural Networks, vol. IV, 1995, pp. 1942–1948.1995.

[Kennedy et al, 1997] Kennedy J. and Eberhart R. C., "A Discrete Binary Version of the Particle Swarm Optimization", Proc. Of the conference on Systems, Man, and Cybernetics SMC97, pp. 4104-4109, 1997.

[Kim et Lee, 2009] Kim, I.-W. and K.-H. Lee, A model-driven approach for describing semantic Web services: From UML to OWL-S. IEEE Transactions on Systems, Man and Cybernetics Part C: Applications and Reviews, 2009. 39(6): p. 637-646.

[Kirkpatrick et al., 1983] S. Kirkpatrick, C.D. Gelatt, M.P. Vecchi. (1983). Optimization by simulated annealing, Science, Vol. 220, N° 4598, pp. 671-680, 1983.

[Klose et al, 2007] Klose, K., Knackstedt, R., & Beverungen, D. (2007). Identification of services - A stakeholder based approach to SOA development and its application in the area of production planning. In *Proceedings of the Fifteenth European Conference on Information Systems (ECIS)* (pp. 1802–1814). 2007

[Komondoor et al, 2012] Raghavan Komondoor ,V. Krishna Nandivada ,Saurabh Sinha." Identifying Services from Legacy Batch Applications". Proceedings of ISEC '12, Feb. 22-25, 2012 Kanpur, UP, India.

[Kumari et al, 2013] Kumari, C. A., & Srinivas, K. (2013). Software module clustering using a fast multi-objective hyper-heuristic evolutionary algorithm. *International Journal of Applied Information Systems*, 5(6), 12–18. doi:10.5120/ijais13-450925

[Lautenbacher et Bauer, 2007] Lautenbacher, F. and B. Bauer. Creating a meta-model for semantic Web service standards. 2007. Barcelona, Spain: INSTICC Press.

[LeMoigne, 1990] J.L Le Moigne, "La modélisation des systèmes complexes", Afcet-systèmes,Dunod, 1990.

[Lemrabet, 2012] Lemrabet Youness. Proposition d'une méthode de spécification d'une architecture orientée services dirigée par le métier dans le cadre d'une collaboration inter-organisationnelle. Thèse de doctorat en génie industriel délivré par l'école centrale de Lille, 2012.

[Lewis et Wrage, 2004] Lewis G. A. et Wrage L.: approaches to constructive interoperability. Rap. tech., Carnegie Mellon Software Engineering Institute, 2004.

[Ma et al, 2009] Ma, Q., Zhou, N., Zhu, Y., & Wang, H. (2009). Evaluating service identification with design metrics on business process decomposition. In *Proceedings of the IEEE International Conference on Services Computing* (pp. 160–167).

[Magnus, 2010] Magnus Erik Hvass Pedersen,"Good Parameters or Particle Swarm Optimization" ,Hvass Laboratories Technical Report no. HL1001, 2010.

[Mancoridis et al, 1998] Mancoridis, S., Mitchell, B. S., Chen, Y., & Gansner, E. R. (1998, June 24-26). Using automatic clustering to produce high-level system organization of source code. In *Proceeding of the International Workshop on Program Understanding*, Ischia, Italy (pp. 45–53).

[MATHIEU, 2004] Hervé MATHIEU, Modélisation Conjointe De L'infrastructure Et Des Processus Pour L'administration Pro-Active De L'entreprise Distribuée, thèse PHD, Institut National Des Sciences Appliquées De Lyon, 2004.

[Megartsi, 1997] Riad MEGARTSI"Etude comparative des méthodes d'analyse des systèmes de production ", Mémoire DEA, Aix-Marseille III, 1997.

[Mendlingy et al, 2007] Jan Mendlingy and Boudewijn Van Dongenz and Wil Van Der Aalst, On The Degree Of Behavioral Similarity Between Business Process Models, 2007

[Metropolis et al., 1953] N. Metropolis, A.W. Rosenbluth, M.N. Rosenbluth, A.H. Teller, E. Teller.(1953). Equation of state calculations by fast computing machines, Journal of Chemical Physics, Vol. 21, pp. 1087-1092, 1953.

[Mittal, 2012] K. Mittal, *Service Oriented Unified Process*. [cited April 2012]. Available from: http://www.kunalmittal.com/html/soup.html.

[Morley et al, 2002] C. Morley and J. Hugues and B. Leblanc, UML Pour L'analyse D'un Système D'informations, Dunod, 2002.

[Morley et al, 2005] C. Morley, J. Hugues, B. Leblanc and O. Hugues. « Processus Métiers et SI: Evaluation, modélisation, mise en œuvre ». 2005.

[Morley, 2000] C. Morley, Changement Organisationnel Et Modélisation Des Processus, 5ème Congrès De l'Association Information Et Management(AIM), Montpellier, 2000.

[Morley, 2002] C. Morley, La Modélisation Des Processus : Typologie Et Proposition Utilisant UML Processus Et Systèmes D'information, Journées ADELI, Paris, 2002.

[Morley, 2006] Chantal MORLEY, Un Cadre Unificateur Pour La Représentation Des Processus, GET/INT, Département Systèmes d'Information, 2006.

[Mukerji et Miller, 2003] Jishnu Mukerji and Joaquin Miller. "*Model Driven Architecture (MDA) Guide Version 1.0.1*. Object Management Group (OMG), Needham, MA, USA, June 2003.

[Mülhenbein et al., 1996] H. Mülhenbein, G. Paa. (1996). From recombination of genes to the estimation of distributions I. Binary parameters, Proceedings of the International Conference on Parallel Problem Solving from Nature IV, 1996, LNCS 1411, pp. 178-187, Springer.

[Nakamura et al, 2009] Nakamura Masahide, Hiroshi Igaki, Takahiro Kimura and Kenich Matsumoto. "Extracting Service Candidates from Procedural Programs based on Process Dependency Analysis". IEEE Asia-Pacific Services Computing Conference 2009.

[Nakamura et al, 2011] Nakamura, M., Igaki, H., Kimura, T., & Matsumoto, K. (2011). Identifying services in procedural programs for migrating legacy system to service oriented architecture. [IJISSS]. *International Journal of Information Systems in the Service Sector*, *3*(4), 54–72. doi:10.4018/jisss.2011100104

[Naseem et al, 2013] Naseem, R., Maqbool, O., & Muhammad, S. (2013). Cooperative clustering for software modularization. *Journal of Systems and Software*, *86*(8), 2045–2062. doi:10.1016/j.jss.2013.03.080

[Niels Schote, 2012] Niels Schote. Model-Driven SOA. Report for research topics, University of Twente, EEMCS – Software Engineering Group, April 2012.

[Nikravesh et al, 2011] Ali Nikravesh, Fereidoon Shams, Soodeh Farokhi, and Amir Ghaffari." 2PSIM: Two Phase Service Identifying Method". OTM 2011, Part II, LNCS 7045, pp. 625–634, 2011.

[OASIS01, 2006] Reference Model for Service Oriented Architecture 1.0. http://docs.oasis-open.org/soa-rm/v1.0/soa-rm.html, 2006.

[OASIS02, 2011] Reference Architecture Foundation for Service Oriented Architecture Version 1.0. http://docs.oasis-open.org/soa-rm/soa-ra/v1.0/soa-ra.html, 2011.

[OMG01, 2001] OMG. "*Model Driven Architecture (MDA)*". Object Management Group (OMG), Needham, MA, USA, July 2001.

[OMG02, 2011] "MDA Specification". http://www.omg.org/mda/specs.htm,2011.

[OMG03, 2002] OMG. MOF 2.0 Query/View/Transformations Request for Proposal. Technical report, Object Management Group (OMG), October 2002.

[OMG04, 2002] OMG. *UML Profile for CORBA Specification*. Object Management Group (OMG), Needham, MA, USA, April 2002.

[OMG05, 2005] OMG. *A Proposal for an MDA Foundation Model*. Object Management Group (OMG), Needham, MA, USA, April 2005.

[OMG06, 2006] OMG. *Meta Object Facility (MOF) Core Specification v2.0*. Object Management Group (OMG), Needham, MA, USA, January 2006.

[OMG07, 2008] OMG. *Common Object Request Broker Architecture (CORBA) Specification, Version 3.1 – CORBA Interfaces*. Object Management Group (OMG), Needham, MA, USA, January 2008.

[OMG08, 2008] OMG. *Common Object Request Broker Architecture (CORBA) Specification, Version 3.1 – CORBA Interoperability*. Object Management Group (OMG), Needham, MA, USA, January 2008.

[OMG09, 2009] OMG. *OMG Unified Modeling Language (OMG UML) – Infrastructure v2.2*. Object Management Group (OMG), Needham, MA, USA, February 2009.

[OMG10, 2009] OMG. *OMG Unified Modeling Language (OMG UML) – Superstructure v2.2*. Object Management Group (OMG), Needham, MA, USA, February 2009.

[OMG11, 2011] OMG. *Business Process Model and Notation (BPMN)*. Object Management Group (OMG), Needham, MA, USA, January 2011.

[OMG12, 2008] *Business Motivation Model Version 1.0*, 2008, disponible sur: http://www.omg.org/spec/BMM/1.0/PDF.

[OMG13, 2009] *Service Oriented Architecture Modeling Language (SoaML) 1.0 - Beta 2*, 2009, p. 167, disponible sur: http://www.omg.org/SoaML/1.0/Beta2/PDF.

[OSG, 2011] OSGi Alliance. *OSGi Service Platform Core Specification – Release 4, Version 4.3*, April 2011.

[Papazoglou et al, 2006] M. P. Papazoglou and W.-J. van den Heuvel, Service-oriented design and development methodology, *International Journal of Web Engineering and Technology* **2**(4) (2006), 412-442. doi:10.1504/IJWET.2006.010423

[PLASTIC01, 2008] Deliverable D1.2, Formal description of the PLASTIC conceptual model and of its relationship with the PLASTIC platform toolset, IST STREP Project, http://www.ist-plastic.org.

[PLASTIC02, 2007] Deliverable D2.2, Graphical design language and tools for resource-aware adaptable components and services, IST STREP Project, http://www.ist-plastic.org.

[Plouin et al, 2007] Plouin, Guillaume, Bruno Pennec, Pascal Grojean et Cyril Rognon. « SOA Perception des entreprises françaises ». 2007.

[Pottinger et al, 2003] Rachel A. Pottinger and Philip A. Bernstein. "Merging models based on given correspondences". In *VLDB '2003: Proceedings of the 29th international conference on Very large data bases*, pages 862–873. VLDB Endowment, 2003.

[Praditwong et al, 2011] Praditwong, K., Harman, M., & Xin, Y. (2011). Software module clustering as a multi-objective search problem. *IEEE Transactions on Software Engineering*, *37*(2), 264–282. doi:10.1109/TSE.2010.26

[Ramollari et al, 2007] E. Ramollari, D. Dranidis, and A. J. H. Simons, A survey of service oriented development methodologies. In: *Proceedings of the 2nd European Young Researchers Workshop on Service Oriented Computing*, Leicester, UK, June 2007.

[Rathipriya et al, 2011] R.Rathipriya, K.Thangavel, J.Bagyamani." Binary Particle Swarm Optimization basedBiclustering of Web usage Data". International Journal of Computer Applications (0975 – 8887)Volume 25– No.2, July 2011.

[Reddy et al, 2009] Reddy, V., A. Dubey, S. Lakshmanan, S. Sukumaran et R. Sisodia.. « Evaluating legacy assets in the context of migration to SOA ». Software Quality Journal, vol. 17, no 1.2009.

[Rehab, 2010] Rehab F. Abdel-Kader.: Genetically Improved PSO Algorithm for Efficient Data Clustering.In: Proceedings of the IEEE Second International Conference on Machine Learning andComputing (2010).

[Rouillard, et al, 2007] Rouillard, José, Thomas Vantroys et Vincent Chevrin. « Architectures orientées services : une approche pragmatique des SOA Coll ». Série génie logiciel . Vuibert, 317 p. 2007.

[Sandy, 2007] Sandy, Carter. "The New Language of Business: SOA & Web 2.0", 1st. United States IBM Press, 320 p. 2007.

[Santorum, 2011] Marco Oswaldo SANTORUM GAIBOR. « Une méthode ludique et participative pour la représentation et l'amélioration des processus métier ». Thèse de doctorat.Université de Gronoble 2011.

[Senthil et al, 2008] Senthil, Mani, V. S. Sinha, Sukaviriya Noi et T. Ramachandra. «Using user interface design to enhance service identification ». In., p. 78-87. Coll. « 2008 IEEEInternational Conference on Web Services (ICWS) ». Piscataway, NJ, USA: IEEE2008.

[Shirazi et al, 2009] H. M Shirazi, N. Fareghzadeh, and A. Seyyedi, A combinational approach to service identification in SOA, *Journal of Applied Sciences Research* **5**(10) (2009), 1390-1397.

[Siarry et al, 2003] Patrick Siarry , Johann Dréo , Alain Pétrowski , Eric Taillard." Métaheuristiques pour l'optimisation difficile".Edition Eyrolles 2003.

[Sneed, 2006] Sneed, H. M. « Integrating legacy software into a service oriented architecture ». In., p. 11 pp. Coll. « 10th European Conference on Software Maintenance and Reengineering ». Los Alamitos, CA, USA: IEEE Computer Society.2006.

[Soley, 2000] Soley R. M., *Model driven architecture (mda), draft 3.2. Rapport technique*, 2000, p. 12, disponible sur: http://www.omg.org/cgi-bin/doc?omg/00-11-05.

[Soltani et al, 2013] Soltani, M., & Benslimane, S. M. (2013). Ontology-based Multi-Objective Evolutionary Algorithm for Deriving Software Services from Business Process Model. *International Journal of Information Systems in the Service Sector, 5(3), 35-53.* DOI: 10.4018/jisss.2013070103

[Soltani et al, 2012a] Soltani M. and Benslimane S. M., "From A High Level Business Process Model To Service Model Artifacts: A Model-Driven Approach". 14-th International Conference on Enterprise Information Systems (ICEIS 2012), pp. 265-268 28 June – 1 July, 2012, Wroclaw, Poland.

[Soltani et al, 2012b] Soltani M., Benslimane .S M. « AMSI : An Automatic Model-Driven Service Identification from Business Process Models », 4th international conference on Web and Information Technologies (ICWIT'2012). P275-276, 29-30 April 2012, Sidi Bel Abbes, Algeria.

[Soltani et al, 2011] Soltani M., Benslimane .S « Génération automatique des modèles de services à partir des modèles de processus métiers : Approche dirigée par les ontologies », (COSI'2011) Colloque sur l'Optimisation et les Systèmes d'Information, P389-400, 24-27 Avril 2011, Guelma, Algérie.

[Soltani et al, 2010a] Soltani M., and Benslimane S. M., "Towards automatic model-driven transformation: a business process ontology based Approach" (ICWIT'2010). The Third International Conference on Web and Information Technologies 16-19 June, 2010, Marrakech – Morocco.

[Soltani et al, 2010b] Mokhtar Soltani, Sidi Mohamed Benslimane. Towards automatic transformation from CIM to PIM. (ICMOSS'2010) Congrès International sur les modèles, Optimisation et Sécurité des Systèmes, 29 au 31 Mai 2010, Tiaret, Algérie

[Srikanth et al, 2007] Srikanth Inaganti, Gopala Krishna Behara. « Service Identification: BPM and SOAHandshake ». (Mars), p. 12.2007.

[Strnadl, 2007] Strnadl, C. F. « Bridging architectural boundaries design and implementation of a semantic BPM and SOA governance tool ». In., p. 518-29. Coll. « Service-Oriented Computing - ICSOC 2007. Proceedings Fifth International Conference. (Lecture Notes in Computer Science vol. 4749) ». Berlin, Germany: Springer-Verlag.2007.

[Strosnider et al, 2008] J. K. Strosnider, P. Nandi, S. Kumaran, S. Ghosh, A. Arsanjani, "Model-driven synthesis of SOA solutions", IBM Systems Journal, Vol 47, NO 3.2008.

[Suntae et al, 2008] Suntae, Kim, Kim Minseong et Park Sooyong. 2008. « Service identification using goal and scenario in service oriented architecture ». In., p. 419-26. Coll. « 2008 15th Asia-Pacific Software Engineering Conference ». Piscataway, NJ, USA: IEEE.Thèse de doctorat en Génie Industriel de l'Ecole des Mines de Saint-Etienne et l'Université JeanMonnet. Septembre 2005.

[SVANIDZAITĖ, 2012] Sandra SVANIDZAITĖ. A Comparison of SOA Methodologies Analysis & Design Phases. *Institute of Mathematics and Informatics, Vilnius University*

[Talbot, 2003] Jean Talbot,"Les T.I. et la réingénierie des processus", HEC Montréal- MBA, 2003.

[Terje et al, 2009] Terje, W., & Guttorm, S. (2009). A survey of development methods for semantic web service systems. [IJISSS]. *International Journal of Information Systems in the Service Sector*, *1*(2), 1–16.

[Timm et Gannod, 2008] Timm, J.T.E. and Gannod, G. C, A model-driven framework for the specification, grounding, and execution of semantic Web services. 2008, Arizona State University. p. 170.

[Touzi, 2007] J. TOUZI, "Aide à la conception de Système d'Information Collaboratif support de l'interopérabilité des entreprises", thèse de doctorat spécialité : systèmes industriels, Préparée au Centre de Génie Industriel - Ecole des Mines d'Albi Carmaux, soutenue le 09/11/2007

[Vernadat, 1996] F. Vernadat, "Enterprise modelling and integration, Principles and applications", Chapman & Hall, 1996.

[W3C, 2003] SOAP version 1.2 part 0: Primer. http://www.w3.org/TR/2003/REC-soap12-part0-20030624/

[Webratio, 2009] Webratio site development. 2009; Available from: http://www.Webratio.com.

[Xiaohui et al, 2005] Xiaohui Cui, Thomas E. Potok, Paul Palathingal. "Document Clustering using Particle Swarm Optimization". Swarm Intelligence Symposium, 2005. SIS 2005. Proceedings 2005 IEEE.

[Xpand01] Xpand. Available from: http://www.eclipse.org/workinggroups/oaw/.

[Yang et Chung, 2006] Yang, J.H. and I.J. Chung. Automatic generation of service ontology from UML diagrams for semantic Web services. 2006. Beijing, China: Springer Verlag.

[Yousef et al, 2009] Yousef, R., Odeh, M., Coward, D., & Sharieh, A. (2009). BPAOntoSOA: A generic framework to derive software service oriented models from business process architectures. In *Proceedings of the Second International Conference on the Applications of Digital Information and Web Technologies (ICADIWT '09)* (pp. 50–5). Piscataway, NJ, USA: IEEE. 2009.

[Yukyong et al, 2009] Yukyong, K., & Kyung-Goo, D. (2009). Formal identification of right-grained services for service-oriented modeling. In *Proceedings of the 10th International Conference on Web Information Systems Engineering (WISE '09)* (pp. 261–273). Springer-Verlag Berlin, Heidelberg.

[Zaidat, 2005] Zaidat, "Spécification d'un cadre d'ingénierie pour les réseaux d'organisations", Thèse de doctorat, l'Université de Jean Monnet 2005.

[Zhang et al, 2008] L.J. Zhang, N. Zhou, Y.M. Chee, A. Jalaldeen, K. Ponnalagu, R.R. Sindhgatta, A. Arsanjani, F. Bernardini, "SOMA-ME: A platform for the model-driven design ofSOA solutions", IBM Systems Journal, Vol 47, NO 3.2008.