Intégration des systèmes d'information industriels : une approche flexible basée sur les services sémantiques

Saïd Izza

Ecole Nationale Supérieure des Mines
SAINT-ETIENNE

N° d'ordre : 418 I

THESE

présentée par

Saïd IZZA

Pour obtenir le grade de Docteur
de l'Ecole Nationale Supérieure des Mines de Saint-Etienne

Spécialité : Informatique

INTEGRATION DES SYSTEMES D'INFORMATION INDUSTRIELS

Une approche flexible basée sur les services sémantiques

Soutenue à Saint-Etienne, le 20 novembre 2006

Membres du jury

Président :	Hervé PINGAUD	Professeur/Ecole des Mines d'Albi
Rapporteurs :	François VERNADAT	Professeur/Université de Metz
	Michel SCHNEIDER	Professeur/Université de Clermont-Ferrand
Examinateurs :	Pierre LEBRUN	Ingénieur-Responsable Architecture IT /STMicroelectronics
Directeur(s) de thèse :	Patrick BURLAT	Professeur/Ecole des Mines de Saint-Etienne
	Lucien VINCENT	Maître de Recherche/Ecole des Mines de Saint-Etienne

● Spécialités doctorales : **Responsables :**

SCIENCES ET GENIE DES MATERIAUX	**J. DRIVER** Directeur de recherche – Centre SMS
MECANIQUE ET INGENIERIE	**A. VAUTRIN** Professeur – Centre SMS
GENIE DES PROCEDES	**G. THOMAS** Professeur – Centre SPIN
SCIENCES DE LA TERRE	**B. GUY** Maitre de recherche
SCIENCES ET GENIE DE L'ENVIRONNEMENT	**J. BOURGOIS** Professeur – Centre SITE
MATHEMATIQUES APPLIQUEES	**E. TOUBOUL** Ingénieur
INFORMATIQUE	**O. BOISSIER** Professeur – Centre G2I
IMAGE, VISION, SIGNAL	**JC. PINOLI** Professeur – Centre CIS
GENIE INDUSTRIEL	**P. BURLAT** Professeur – Centre G2I
MICROELECTRONIQUE	**Ph. COLLOT** Professeur – Centre CMP

● Enseignants-chercheurs et chercheurs autorisés à diriger des thèses de doctorat (titulaires d'un doctorat d'Etat ou d'une HDR)

BENABEN	Patrick	PR 2	Sciences & Génie des Matériaux	SMS
BERNACHE-ASSOLANT	Didier	PR 1	Génie des Procédés	CIS
BIGOT	Jean-Pierre	MR	Génie des Procédés	SPIN
BILAL	Essaïd	MR	Sciences de la Terre	SPIN
BOISSIER	Olivier	PR 2	Informatique	G2I
BOUDAREL	Marie-Reine	MA	Sciences de l'inform. & com.	DF
BOURGOIS	Jacques	PR 1	Sciences & Génie de l'Environnement	SITE
BRODHAG	Christian	MR	Sciences & Génie de l'Environnement	SITE
BURLAT	Patrick	PR 2	Génie industriel	G2I
COLLOT	Philippe	PR 1	Microélectronique	CMP
COURNIL	Michel	PR 1	Génie des Procédés	SPIN
DAUZERE-PERES	Stéphane	PR 1	Génie industriel	CMP
DARRIEULAT	Michel	ICM	Sciences & Génie des Matériaux	SMS
DECHOMETS	Roland	PR 2	Sciences & Génie de l'Environnement	SITE
DELAFOSSE	David	PR 2	Sciences & Génie des Matériaux	SMS
DOLGUI	Alexandre	PR 1	Informatique	G2I
DRAPIER	Sylvain	PR 2	Mécanique & Ingénierie	CIS
DRIVER	Julian	DR	Sciences & Génie des Matériaux	SMS
FOREST	Bernard	PR 1	Sciences & Génie des Matériaux	SMS
FORMISYN	Pascal	PR 1	Sciences & Génie de l'Environnement	SITE
FORTUNIER	Roland	PR 1	Sciences & Génie des Matériaux	CMP
FRACZKIEWICZ	Anna	MR	Sciences & Génie des Matériaux	SMS
GARCIA	Daniel	CR	Génie des Procédés	SPIN
GIRARDOT	Jean-Jacques	MR	Informatique	G2I
GOEURIOT	Dominique	MR	Sciences & Génie des Matériaux	SMS
GOEURIOT	Patrice	MR	Sciences & Génie des Matériaux	SMS
GRAILLOT	Didier	DR	Sciences & Génie de l'Environnement	SITE
GROSSEAU	Philippe	MR	Génie des Procédés	SPIN
GRUY	Frédéric	MR	Génie des Procédés	SPIN
GUILHOT	Bernard	DR	Génie des Procédés	CIS
GUY	Bernard	MR	Sciences de la Terre	SPIN
GUYONNET	René	DR	Génie des Procédés	SPIN
HERRI	Jean-Michel	PR 2	Génie des Procédés	SPIN
JOYE	Marc	Ing. (Gemplus)	Microélectronique	CMP
KLÖCKER	Helmut	CR	Sciences & Génie des Matériaux	SMS
LAFOREST	Valérie	CR	Sciences & Génie de l'Environnement	SITE
LE COZE	Jean	PR 1	Sciences & Génie des Matériaux	SMS
LI	Jean-Michel	EC (CCI MP)	Microélectronique	CMP
LONDICHE	Henry	MR	Sciences & Génie de l'Environnement	SITE
MOLIMARD	Jérôme	MA	Sciences & Génie des Matériaux	SMS
MONTHEILLET	Frank	DR 1 CNRS	Sciences & Génie des Matériaux	SMS
PERIER-CAMBY	Laurent	MA1	Génie des Procédés	SPIN
PIJOLAT	Christophe	PR 1	Génie des Procédés	SPIN
PIJOLAT	Michèle	PR 1	Génie des Procédés	SPIN
PINOLI	Jean-Charles	PR 1	Image, Vision, Signal	CIS
STOLARZ	Jacques	CR	Sciences & Génie des Matériaux	SMS
SZAFNICKI	Konrad	CR	Sciences de la Terre	SITE
THOMAS	Gérard	PR 1	Génie des Procédés	SPIN
TRAN MINH	Cahn	MR	Génie des Procédés	SPIN
VALDIVIESO	Françoise	CR	Génie des Procédés	SPIN
VALDIVIESO	François	MA	Sciences & Génie des Matériaux	SMS
VAUTRIN	Alain	PR 1	Mécanique & Ingénierie	SMS
VIRICELLE	Jean-Paul	CR	Génie des procédés	SPIN
WOLSKI	Krzysztof	CR	Sciences & Génie des Matériaux	SMS
XIE	Xiaolan	PR 1	Génie industriel	CIS

Glossaire :

PR 1	Professeur 1ère catégorie
PR 2	Professeur 2ème catégorie
MA(MDC)	Maître assistant
DR 1	Directeur de recherche
Ing.	Ingénieur
MR(DR2)	Maître de recherche
CR	Chargé de recherche
EC	Enseignant-chercheur
ICM	Ingénieur en chef des mines

Centres :

SMS	Sciences des Matériaux et des Structures
SPIN	Sciences des Processus Industriels et Naturels
SITE	Sciences Information et Technologies pour l'Environnement
G2I	Génie Industriel et Informatique
CMP	Centre de Microélectronique de Provence
CIS	Centre Ingénierie et Santé

Ecole Nationale Supérieure des Mines
SAINT-ETIENNE

N° d'ordre : 418 I

THESE

présentée par

Saïd IZZA

Pour obtenir le grade de Docteur
de l'Ecole Nationale Supérieure des Mines de Saint-Etienne

Spécialité : Informatique

INTEGRATION DES SYSTEMES D'INFORMATION INDUSTRIELS

Une approche flexible basée sur les services sémantiques

Soutenue à Saint-Etienne, le 20 novembre 2006

Membres du jury

Président :	Hervé PINGAUD	Professeur/Ecole des Mines d'Albi
Rapporteurs :	François VERNADAT	Professeur/Université de Metz
	Michel SCHNEIDER	Professeur/Université de Clermont-Ferrand
Examinateurs :	Pierre LEBRUN	Ingénieur-Responsable Architecture IT /STMicroelectronics
Directeur(s) de thèse :	Patrick BURLAT	Professeur/Ecole des Mines de Saint-Etienne
	Lucien VINCENT	Maître de Recherche/Ecole des Mines de Saint-Etienne

● Spécialités doctorales : **Responsables :**

SCIENCES ET GENIE DES MATERIAUX	**J. DRIVER** Directeur de recherche – Centre SMS
MECANIQUE ET INGENIERIE	**A. VAUTRIN** Professeur – Centre SMS
GENIE DES PROCEDES	**G. THOMAS** Professeur – Centre SPIN
SCIENCES DE LA TERRE	**B. GUY** Maitre de recherche
SCIENCES ET GENIE DE L'ENVIRONNEMENT	**J. BOURGOIS** Professeur – Centre SITE
MATHEMATIQUES APPLIQUEES	**E. TOUBOUL** Ingénieur
INFORMATIQUE	**O. BOISSIER** Professeur – Centre G2I
IMAGE, VISION, SIGNAL	**JC. PINOLI** Professeur – Centre CIS
GENIE INDUSTRIEL	**P. BURLAT** Professeur – Centre G2I
MICROELECTRONIQUE	**Ph. COLLOT** Professeur – Centre CMP

● Enseignants-chercheurs et chercheurs autorisés à diriger des thèses de doctorat (titulaires d'un doctorat d'Etat ou d'une HDR)

BENABEN	Patrick	PR 2	Sciences & Génie des Matériaux	SMS
BERNACHE-ASSOLANT	Didier	PR 1	Génie des Procédés	CIS
BIGOT	Jean-Pierre	MR	Génie des Procédés	SPIN
BILAL	Essaïd	MR	Sciences de la Terre	SPIN
BOISSIER	Olivier	PR 2	Informatique	G2I
BOUDAREL	Marie-Reine	MA	Sciences de l'inform. & com.	DF
BOURGOIS	Jacques	PR 1	Sciences & Génie de l'Environnement	SITE
BRODHAG	Christian	MR	Sciences & Génie de l'Environnement	SITE
BURLAT	Patrick	PR 2	Génie industriel	G2I
COLLOT	Philippe	PR 1	Microélectronique	CMP
COURNIL	Michel	PR 1	Génie des Procédés	SPIN
DAUZERE-PERES	Stéphane	PR 1	Génie industriel	CMP
DARRIEULAT	Michel	ICM	Sciences & Génie des Matériaux	SMS
DECHOMETS	Roland	PR 2	Sciences & Génie de l'Environnement	SITE
DELAFOSSE	David	PR 2	Sciences & Génie des Matériaux	SMS
DOLGUI	Alexandre	PR 1	Informatique	G2I
DRAPIER	Sylvain	PR 2	Mécanique & Ingénierie	CIS
DRIVER	Julian	DR	Sciences & Génie des Matériaux	SMS
FOREST	Bernard	PR 1	Sciences & Génie des Matériaux	SMS
FORMISYN	Pascal	PR 1	Sciences & Génie de l'Environnement	SITE
FORTUNIER	Roland	PR 1	Sciences & Génie des Matériaux	CMP
FRACZKIEWICZ	Anna	MR	Sciences & Génie des Matériaux	SMS
GARCIA	Daniel	CR	Génie des Procédés	SPIN
GIRARDOT	Jean-Jacques	MR	Informatique	G2I
GOEURIOT	Dominique	MR	Sciences & Génie des Matériaux	SMS
GOEURIOT	Patrice	MR	Sciences & Génie des Matériaux	SMS
GRAILLOT	Didier	DR	Sciences & Génie de l'Environnement	SITE
GROSSEAU	Philippe	MR	Génie des Procédés	SPIN
GRUY	Frédéric	MR	Génie des Procédés	SPIN
GUILHOT	Bernard	DR	Génie des Procédés	CIS
GUY	Bernard	MR	Sciences de la Terre	SPIN
GUYONNET	René	DR	Génie des Procédés	SPIN
HERRI	Jean-Michel	PR 2	Génie des Procédés	SPIN
JOYE	Marc	Ing. (Gemplus)	Microélectronique	CMP
KLÖCKER	Helmut	CR	Sciences & Génie des Matériaux	SMS
LAFOREST	Valérie	CR	Sciences & Génie de l'Environnement	SITE
LE COZE	Jean	PR 1	Sciences & Génie des Matériaux	SMS
LI	Jean-Michel	EC (CCI MP)	Microélectronique	CMP
LONDICHE	Henry	MR	Sciences & Génie de l'Environnement	SITE
MOLIMARD	Jérôme	MA	Sciences & Génie des Matériaux	SMS
MONTHEILLET	Frank	DR 1 CNRS	Sciences & Génie des Matériaux	SMS
PERIER-CAMBY	Laurent	MA1	Génie des Procédés	SPIN
PIJOLAT	Christophe	PR 1	Génie des Procédés	SPIN
PIJOLAT	Michèle	PR 1	Génie des Procédés	SPIN
PINOLI	Jean-Charles	PR 1	Image, Vision, Signal	CIS
STOLARZ	Jacques	CR	Sciences & Génie des Matériaux	SMS
SZAFNICKI	Konrad	CR	Sciences de la Terre	SITE
THOMAS	Gérard	PR 1	Génie des Procédés	SPIN
TRAN MINH	Cahn	MR	Génie des Procédés	SPIN
VALDIVIESO	Françoise	CR	Génie des Procédés	SPIN
VALDIVIESO	François	MA	Sciences & Génie des Matériaux	SMS
VAUTRIN	Alain	PR 1	Mécanique & Ingénierie	SMS
VIRICELLE	Jean-Paul	CR	Génie des procédés	SPIN
WOLSKI	Krzysztof	CR	Sciences & Génie des Matériaux	SMS
XIE	Xiaolan	PR 1	Génie industriel	CIS

Glossaire :

PR 1	Professeur 1ère catégorie
PR 2	Professeur 2ème catégorie
MA(MDC)	Maître assistant
DR 1	Directeur de recherche
Ing.	Ingénieur
MR(DR2)	Maître de recherche
CR	Chargé de recherche
EC	Enseignant-chercheur
ICM	Ingénieur en chef des mines

Centres :

SMS	Sciences des Matériaux et des Structures
SPIN	Sciences des Processus Industriels et Naturels
SITE	Sciences Information et Technologies pour l'Environnement
G2I	Génie Industriel et Informatique
CMP	Centre de Microélectronique de Provence
CIS	Centre Ingénierie et Santé

Remerciements

Il me serait impossible de citer nommément toutes les personnes qui m'ont aidé, encouragé et soutenu afin que ce travail puisse voir le jour. Que toutes ces personnes trouvent ici l'expression de ma sincère reconnaissance.

Je tiens tout d'abord à remercier vivement Monsieur Hervé PINGAUD, Professeur à l'Ecole des Mines d'Albi, pour m'avoir fait l'honneur d'accepter d'examiner mes travaux et de présider le jury de ma soutenance de thèse.

Je tiens aussi à remercier particulièrement Monsieur François VERNADAT, professeur à l'université de Metz, et Monsieur Michel SCHNEIDER, professeur à l'université de Clermont-Ferrand, pour avoir accepté d'être rapporteurs de cette thèse. Je leur suis reconnaissant pour leur lecture attentive ainsi que pour les critiques et suggestions constructives qu'ils ont faites sur ce travail.

Je suis très reconnaissant envers Monsieur Pierre LEBRUN, responsable de l'équipe Architecture et Industrialisation du site Rousset de la société STMicroelectronics, et Monsieur Hugues SOLIGNAC, architecte IT au sein de STMicroelectronics, d'avoir été mes tuteurs industriels. Je les remercie pour leurs conseils, leur aide, leur disponibilité et leur soutien tout au long de cette thèse.

Je remercie bien entendu mes tuteurs scientifiques, Monsieur Patrick BURLAT, professeur à l'Ecole des Mines de Saint-Etienne, et Monsieur Lucien VINCENT, maître de recherche à l'Ecole des Mines de Saint Etienne, pour avoir accepté de diriger mes travaux de recherche. Leur aide, leur disponibilité, leurs conseils, et leur soutien durant toute cette période, m'ont toujours redonné confiance et volonté. Qu'ils trouvent ici l'expression de ma sincère reconnaissance et de ma profonde gratitude.

J'adresse un grand merci à Ali ZAIDAT, sans qui cette thèse n'aurait jamais lieu. Je remercie aussi Hocine IHABCHIENNE et Rabah HARBANE qui n'ont toujours soutenu.

Je tiens aussi à remercier tous les membres du laboratoire G2I et du CMP de Gardanne. Je remercie vivement Monsieur Philippe COLLOT, directeur du CMP qui m'a accueilli chaleureusement au sein de son centre tout au long de la durée de ma thèse. Je remercie particulièrement Monsieur Stéphane DAUZERE-PERES responsable du département SFL du CMP, où j'ai élu domicile. Je le remercie pour sa disponibilité et toute l'aide qu'il m'a apportée. Je remercie aussi tous les membres de SFL avec qui j'ai partagé de beaux moments de bonheur. Un grand merci à Hassan ETTALEB, avec qui j'ai surmonté, avec bonheur, les longs bouchons de l'autoroute A7 et A51. Merci aussi à tous les doctorants du CMP.

J'adresse également tous mes remerciements à tous les membres de l'équipe "Architecture & Industrialisation" de ST Rousset. Un grand merci à Julie Chapron qui a initié le projet d'intégration et d'urbanisation du système d'information au sein de STMicroelectronics, grâce à ses travaux sur l'urbanisme organisationnel.

Je n'omettrais pas de remercier tous mes amis de Marseille avec qui j'ai partagé de très beaux moments de bonheur. Je cite entre autres: Philomène DOURMIAN, Anais ARZOUMANIAN, Abderrahmane MALEK, Djamel BERKANE. Enfin je remercie ma famille qui me manque tant. Il s'agit en particulier de ma femme Malika, de mon fils Sifaks, et de ma mère Fetta. Je remercie aussi tous mes frères et sœurs, mes beaux parents (Amar et Fathma BENZABA), mes beaux frères, mes belles sœurs et mes neveux, et aussi tous mes amis et cousins de Tizi-Ouzou qui m'ont soutenu en toutes circonstances.

À ma mère Fetta
À ma femme Maly et mon fils Sifaks
À tous ceux qui me sont chers

Résumé

Le domaine des systèmes d'information industriels s'est profondément transformé ces dernières années sous l'influence de l'évolution des technologies logicielles (objets, composants, service web, …), de l'évolution des technologies matérielles (loi de Moore), et aussi de l'évolution des organisations (fusions, acquisitions, mondialisation). Conséquence de tous ces facteurs, les systèmes d'information deviennent de plus en plus complexes et hétérogènes qu'il convient alors d'intégrer afin de les faire communiquer et les faire coopérer. Il s'agit du problème d'intégration des systèmes d'information. Notre travail s'inscrit dans cette problématique, et plus précisément dans le cadre de l'intégration sémantique de systèmes d'information de grandes entreprises industrielles. Il propose une approche flexible basée sur les services sémantiques, en combinant à la fois les ontologies et les Services Web.

Après avoir exposé la problématique, nous avons présenté les différentes techniques d'intégration des systèmes d'information industriels. L'analyse de l'état de l'art nous a permis de retenir deux niveaux d'intégration: l'intégration syntaxique et l'intégration sémantique. Cette dernière constitue un problème crucial de l'intégration des systèmes d'information. Jusqu'à présent, ce problème n'est toujours pas correctement traité. Les solutions actuelles se focalisent plutôt sur les techniques d'intégration syntaxique. La prise en compte de l'aspect sémantique peut promouvoir l'intégration en lui apportant plus de consistance et de flexibilité.

En nous focalisant sur les services sémantiques, nous avons constaté un certain nombre de lacunes dont l'inadéquation des architectures actuelles des ontologies à capturer de façon flexible et efficace la sémantique des applications industrielles, le manque de méthodologie à mettre en œuvre pour définir les ontologies et aussi les services sémantiques, le manque d'approches de découverte et de médiation de services dans le contexte intra-entreprise, et la complexité inhérente à l'utilisation des technologies associées à l'exploitation de la sémantique.

Partant de ce constat, nous avons alors proposé une approche flexible d'intégration des applications industrielles qui s'intitule *ODSOI* (Ontology-Driven Service-Oriented Integration). Cette approche se focalise principalement sur trois sous-problématiques complémentaires qui sont respectivement la construction d'une architecture de services ($\boldsymbol{P_{Syn}}$) permettant de définir et de structurer les services d'entreprise, la construction d'une architecture sémantique ($\boldsymbol{P_{Sem}}$) permettant de définir et de structurer les ontologies d'entreprise servant à enrichir sémantiquement les services d'entreprise, et la construction d'une architecture d'intégration ($\boldsymbol{P_{Int}}$) permettant d'offrir des mécanismes d'intégration basés sur la sémantique. Notre approche repose sur trois principes majeurs qui sont l'ouverture, l'unification et l'urbanisation. Le principe d'ouverture impose de s'inscrire dans le cadre d'utilisation de standards industriels tels que WSDL et OWL. Le principe d'unification permet d'uniformiser les composants du système d'information. Et en dernier lieu, le principe d'urbanisation permet de mieux structurer l'architecture des services, l'architecture sémantique et aussi l'architecture d'intégration.

Nous basant sur ces trois architectures, nous avons implémenté un prototype permettant de créer, de gérer, et de mettre en œuvre des projets d'intégration. Nous avons enfin réalisé diverses expérimentations portant sur le domaine de la maintenance préventive en milieu industriel.

MOTS-CLÉS: Système d'Information, Application, Intégration, Interopérabilité, Ontologie, Sémantique, Service, Découverte, Médiation, Urbanisation, Flexibilité.

Abstract

Over the last decade, the field of industrial information systems was deeply transformed under the influence of the evolution of software technologies (objects, components, web services,…), the evolution of hardware technologies (Moore law), and also the evolution of organisations (fusions, acquisitions, globalisation). Consequently, the information systems became more and more complex and heterogeneous. Those need to be integrated in order to make them communicate and cooperate. It is the problem of information system integration. This work treats this latter problem and precisely the semantic integration one. It proposes a flexible approach that is based on semantic services and that combines both ontologies and web services in order to overcome some issues related to the semantic integration problem.

After having exposed our research problematic, we reviewed the most important related works that concern the integration of industrial information systems. The analysis of the state of the art let us to consider mainly two integration levels: syntactic and semantic integration. This latter constitutes a crucial problem that is not is not correctly addressed by today's integration solutions that focus mainly on the syntactical integration. Addressing the semantic aspect will promote the integration by providing it more consistency and robustness.

Focalising our work on semantic services, that constitute the most efficient and flexible approaches that deal with semantic integration, allowed us to note some important limitations that are mainly: the discrepancy of current ontology architectures to correctly capture the semantics of industrial applications, the lack of methodologies in order to build ontologies and also semantic services, the lack of pertinent discovery and mediation approaches for intra-entreprise integration issues, and the complexity of the technologies related to the exploitation of the enterprise semantics.

Thus, we have proposed a flexible approach for integrating industrial applications that is named *ODSOI* (Ontology-Driven Service-Oriented Integration). This approach focuses on three complementary problematics that are respectively: the building of the architecture of enterprise services ($\boldsymbol{P_{Syn}}$) that defines and structures enterprise services, the building of the semantic architecture ($\boldsymbol{P_{Sem}}$) that semantically describes enterprise services, and the building of the integration architecture ($\boldsymbol{P_{Int}}$) that defines integration mechanism based on enterprise semantics. Our approach is based on three major principles that are openness, unification and urbanisation. First, the openness principle imposes us to use and to conform to industrial standards such as WSDL and OWL. Second, the unification principle allows to make information system components uniform. Third, the urbanisation principle allows to correctly structuring the service architecture, the semantic architecture and also the integration architecture.

Basing on these three complementary architectures, we implement a prototype that creates, manages and exploits integration projects. Finally, we led various experimentations of the prototype that concern the domain of preventive maintenance within an industrial enterprise.

KEYWORDS: Information System, Application, Integration, Interoperability, Ontology, Semantics, Web Service, Discovery, Mediation, Urbanisation, Flexibility.

Sommaire

LISTE DES FIGURES

LISTE DES TABLEAUX

INTRODUCTION

Chapitre I

INTRODUCTION

I.1. Introduction à la problématique

L'intégration, actuellement terme très en vogue, est devenue pour beaucoup d'entreprises, notamment les grandes entreprises industrielles, une nécessité incontournable pour faire face aux exigences sans cesse évolutives du marché. Les contraintes qui pèsent de nos jours sur les entreprises sont souvent directement répercutées sur leurs systèmes d'information, à qui on demande d'être sans cesse plus agiles afin de pouvoir mieux soutenir la stratégie de l'entreprise. Pour faire face à ces nouvelles exigences, les entreprises ont très souvent recours au concept d'intégration, et parfois de façon plus spécifique au concept d'interopérabilité afin d'interconnecter leurs applications informatiques. Fondamentalement, l'intégration fournit des outils traditionnels tels que les intergiciels ou les outils EAI (Enterprise Application Integration) pour faire communiquer et coopérer des applications d'entreprise hétérogènes dont la difficulté majeure relève du fait que ces applications ont été généralement conçues de façon indépendante, voire de façon incompatible.

Récemment, les Services Web ont émergé avec l'apparition et l'évolution de l'Internet et permettent d'offrir une panoplie de standards pour l'intégration. Bien que les Services Web ne constituent pas encore une technologie totalement mûre, ils sont souvent considérés comme étant une technologie prometteuse qui permettra de rendre l'intégration plus simple à travers l'utilisation de standards Web. En dépit de la disponibilité d'un grand nombre d'outils, une utilisation de plus en plus répandue de standards, et l'existence d'un certain nombre d'efforts en cours de développement, il n'en demeure pas moins que l'objectif de l'intégration, qui est l'intégration sémantique des applications, n'est toujours pas atteint. En effet, l'intégration fournit des solutions pour l'interconnexion des applications au niveau technique et au niveau syntaxique mais ne traite pas correctement, ou du moins pour le moment, la problématique liée à l'intégration sémantique.

L'intégration sémantique devient alors un enjeu majeur pour l'intégration des systèmes d'information. Elle se propose d'interconnecter les applications au niveau conceptuel, permettant ainsi de rendre plus flexible le processus d'intégration. A l'origine la sémantique constitue une branche de la linguistique qui s'occupe de l'étude des sens des termes. Dans cette thèse, la sémantique réfère au sens des différents éléments d'un système d'information d'entreprise qui peuvent être des données, des fonctions, voire des processus. Un exemple concret de problème sémantique, est par exemple le problème lié

à la définition de l'entité *"client"*. Pour certains, un *"client"* est quelqu'un qui a déjà acheté des produits à l'entreprise. Pour d'autres, un *"client"* est une personne ayant manifesté des intérêts quelconques pour les produits de l'entreprise. L'existence de problèmes liés à la sémantique est une évidence dès lors que l'on dispose de plusieurs systèmes hétérogènes. Il devient vital que l'identification de ces conflits, puis leur résolution se fasse le plutôt possible, de préférence dès les phases amont du projet d'intégration.

La difficulté liée à la sémantique des données, des fonctions voire des processus tient du fait qu'elle peut différer d'un système à un autre, en fonction du contexte d'utilisation. La sémantique constitue un aspect fondamental de l'intégration. Pour que cette intégration puisse être menée avec succès, il est nécessaire que les différentes applications utilisées puissent interpréter de la même manière les données échangées et les fonctions réutilisées. Les conflits sémantiques surviennent lorsque, par exemple, l'information change de sens en fonction du contexte. La résolution des hétérogénéités sémantiques nécessite la mise en place de mécanismes de découverte et de médiation basées sur la sémantique. Le même raisonnement tient pour l'interopérabilité des systèmes, qui est actuellement au cœur des solutions dites faiblement couplées pour l'intégration d'entreprise.

Bien qu'il existe à l'heure actuelle certains efforts dans le domaine de l'intégration ou de l'interopérabilité sémantique (par exemple OWL-S, METEOR-S et WSMO), il n'en demeure pas moins que la plupart de ces initiatives sont en cours de développement et/ou manquent de maturité. Par conséquent, l'intégration sémantique constitue une problématique toujours ouverte, aussi bien dans le contexte de l'intégration intra-entreprise que dans le contexte de grandes entreprises caractérisées par un environnement dynamique. En particulier, l'intégration sémantique apparaît comme un réel besoin dans le contexte spécifique de notre projet industriel qui concerne une grande entreprise (STMicroelectronics) du secteur de la microélectronique, secteur par ailleurs caractérisé par de fréquentes évolutions liées en grande partie aux changements fréquents de la technologie et de la stratégie déployée.

I.2. Cadre de la thèse

Cette thèse est réalisée dans le cadre d'une collaboration industrielle entre l'Ecole des Mines de Saint-Étienne et la société STMicroelectronics. Elle s'inscrit dans la démarche d'intégration initiée par cette société pour répondre à une situation complexe, liée à la fois à la complexité des processus technologiques, à l'hétérogénéité des applications déployées et à la volonté de rendre le système d'information plus flexible et plus cohérent. Elle complète la thèse qui vient de s'achever de Julie Chapron [Chapron 2006] qui porte sur le management de l'évolution des systèmes d'information dans un environnement réactif, et elle est complétée par la thèse en cours de Rahee Ghurbhurn [Ghurbhurn et al. 2006] qui porte sur l'intégration sémantique des données.

Pour notre thèse, la collaboration portait sur le projet MiMISI (**Mi**croelectronics **M**anufacturing **I**nformation **S**ystem **I**ntegration) qui concerne la problématique d'intégration d'applications industrielles dans le secteur de la microélectronique. Plus précisément, le projet portait sur l'élaboration d'un cadre méthodologique global qui va permettre de guider et d'aider les utilisateurs dans leur processus d'intégration sémantique d'applications industrielles.

Le projet MiMISI peut être considéré comme un projet pilote pour l'intégration d'applications industrielles au sein de la société STMicroelectronics. Cette société est une compagnie multinationale franco-italienne spécialisée dans la fabrication de composants microélectroniques. Elle constitue l'un des leaders mondiaux dans la fabrication des semi-conducteurs. Elle dispose de plus de 17 sites et environ 40 000 employés répartis dans près de 30 pays dans le monde et elle produit une panoplie de 3000 types de produits qu'elle livre à plus de 1500 clients. En plus de cette complexité quantitative, l'entreprise est aussi dynamique et multidisciplinaire dans le sens où elle comporte plusieurs domaines métiers hétérogènes et évolutifs, chacun d'eux comportant une multitude d'applications hétérogènes. En outre, STMicroelectronics nécessite de la flexibilité dans le but de mieux gérer les multiples changements qui sont fréquents dans le domaine de la microélectronique et qui sont la conséquence de la loi de Moore[1] qui conduit ainsi à changer souvent de technologie de fabrication et donc souvent d'applications servant de support à ces technologies.

Cette thèse est réalisée en collaboration avec le site de Rousset de la société STMicroelectronics. Ce dernier est un site qui comprend deux usines de production ST6 et ST8 (et qui produisent respectivement des plaquettes de dimensions 6 et 8 mm) et des ateliers de tests. Plus précisément, la collaboration s'est effectuée avec l'équipe "Architecture et Industrialisation" qui est localisée au sein du département "Manufacturing Application Systems". L'objectif de cette équipe est le développement et le suivi des projets d'architecture et d'industrialisation des applications informatiques supportant le manufacturing.

I.3. Objectifs et contributions de la thèse

Ce travail de recherche a pour principal objectif de concevoir des architectures d'intégration flexibles dans le cadre du projet MiMISI. La recherche de flexibilité nous a poussé à prendre en compte deux aspects fondamentaux qui sont: (i) l'orientation service et (ii) la prise en compte de la sémantique à travers la notion d'ontologie. L'orientation services permet d'offrir le meilleur cadre architectural (architecture orientée services) des applications en se basant sur la technologie service web qui se base sur des standards industriels (SOAP, WSDL, UDDI, etc.). La sémantique permet d'offrir le meilleur moyen pour rendre les échanges inter-applicatifs ainsi que l'orchestration des applications plus flexibles et plus intelligibles à travers l'utilisation des technologies de web sémantique (RDF, DAML, OWL, etc.).

Nous nous focalisons donc exclusivement sur l'intégration sémantique d'applications d'entreprise. Notre contribution principale dans de ce travail de thèse porte sur les problématiques liées à l'intégration sémantique orientée services des applications d'entreprises. L'approche que nous proposons s'intitule ODSOI (**O**ntology-**D**riven **S**ervice-**O**riented **I**ntegration) elle est basée sur l'utilisation des services sémantiques d'entreprise (services d'entreprise qui sont enrichis sémantiquement). Elle se résume comme suit :

- (i) la mise en place d'une architecture orientée services d'entreprise;
- (ii) l'enrichissement sémantique de l'architecture orientée services d'entreprise;

[1] Rappelons que la loi de Moore (de l'ingénieur Gordon E. Moore) postule le doublement tous les dix huit mois des performances des semi-conducteurs.

- (iii) et la définition de mécanismes d'intégration basés sur la sémantique des services d'entreprise.

Ces trois préoccupations sont complémentaires et s'inscrivent dans la continuité des problématiques déjà traitées dans les travaux de [Chapron 2006]. De plus, notre approche se base sur certains principes fondateurs qui ont été retenus tout au long du projet MiMISI, et qui sont : le principe d'ouverture (qui impose l'utilisation de standards), le principe d'unification (qui permet de représenter de façon uniforme les différents composants du système d'information d'entreprise), et le principe d'urbanisation (qui permet de structurer les services d'entreprise, les sémantiques d'entreprise et les services d'intégration). Chacun de ces principes est fondamental dans la mesure où il intervient dans la réalisation de chacune des trois préoccupations précédentes.

I.4. Méthodologie de travail

Dans ses grandes lignes, la démarche adoptée pour notre travail est guidée par les nombreuses questions issues des préoccupations de notre collaborateur industriel. Etant donné que la problématique d'intégration est complexe, nous nous sommes efforcés tout au long de cette thèse de prendre suffisamment de recul afin de proposer un cadre méthodologique relativement global et suffisamment complet afin de mieux aider et mieux guider les utilisateurs dans leur processus d'intégration.

Notre démarche de travail est essentiellement itérative et incrémentale du fait que nous nous sommes basés sur un cycle spiral permettant de répondre progressivement à chacun des sous-objectifs précédemment cités. Plus précisément, notre démarche repose principalement sur les étapes suivantes :

- (i) étape de 'servicisation'[2] ou de construction des services d'entreprise : qui définit le modèle ainsi que la trajectoire de migration vers un système orienté services;
- (ii) étape de 'sémantification'[3] ou d'enrichissement sémantique: qui permet de définir un modèle sémantique et la trajectoire de migration du système orienté services vers un système 'sémantifié';
- (iii) étape d'intégration: qui définit des mécanismes d'intégration basés sur la sémantique pour intégrer des services d'entreprise.

Bien entendu, la réalisation de chacune des étapes précédentes nécessite un processus complexe qui comprend principalement les phases suivantes : une phase préliminaire de recherche bibliographique, une phase de recherche et enfin une phase d'implémentation et d'expérimentation. Ces différentes phases sont très souvent menées de façon itérative et incrémentale.

I.5. Organisation du document

Cette thèse est structurée en trois grandes parties. La première partie porte sur l'état de l'art en matière de techniques et démarches d'intégration et/ou d'interopérabilité des

[2] Nous empruntons les anglicismes 'serviciser' et 'servicisation' pour désigner respectivement l'action et le processus de migration vers une architecture de services d'entreprise.

[3] Nous empruntons les anglicismes 'sémantifier' et 'sémantification' pour désigner respectivement l'action et le processus d'enrichissement sémantique des services d'entreprise.

systèmes d'information. Elle comporte trois chapitres. Le chapitre II porte sur la problématique de l'intégration et/ou d'interopérabilité dans les entreprises industrielles en général et dans les entreprises du secteur de la microélectronique en particulier. Le chapitre III et IV étudient respectivement les approches d'intégration syntaxiques et les approches d'intégration sémantiques des systèmes d'information industriels.

La deuxième partie est consacrée à la conceptualisation de notre approche d'intégration d'applications industrielles. Le chapitre V pose les bases de notre approche en motivant les choix retenus et en mettant en évidence les trois sous-problématiques essentielles qui découlent de notre orientation et qui sont : (i) la construction de la couche de services, (ii) la construction de la couche d'ontologies (iii) et la construction de la couche d'intégration. Le chapitre VI répond à la problématique de construction de la couche de services à travers la proposition d'une extension du modèle de base de l'architecture orientée services. Le chapitre VII traite de la construction de la couche d'ontologies à travers la proposition d'un modèle sémantique d'entreprise. Le chapitre VIII adresse la problématique de construction de la couche d'intégration à travers la définition et l'amélioration de certains services d'intégration tels que la découverte et la médiation sémantique.

La troisième et dernière partie porte sur l'implémentation et l'expérimentation de notre approche. Elle est composée d'un seul chapitre (le chapitre IX) qui présente les différents aspects liés à l'implémentation du prototype que nous développerons et aussi aux diverses expérimentations que nous effectuerons dans le cadre de déploiement de ce prototype sur le cas réel de la société STMicroelectronics.

Enfin, cette thèse est clôturée par le chapitre X qui donne les conclusions et perspectives de ce travail. Ce chapitre propose une synthèse et un bilan du travail effectué durant cette thèse et un ensemble de perspectives liées notamment à la poursuite de ce travail ainsi qu'aux nouveaux thèmes de recherche qui nous paraissent les plus pertinents.

PARTIE 1

ETAT DE L'ART

Chapitre II

LA PROBLEMATIQUE DE L'INTEGRATION ET DE L'INTEROPERABILITE DANS LES ENTREPRISES INDUSTRIELLES

II.1. Introduction

L'intégration des systèmes d'information constitue de nos jours l'un des problèmes complexes auxquels l'entreprise est confrontée. Au cœur de cette problématique se pose entre autres le problème de l'interopérabilité. Les problèmes d'intégration et de l'interopérabilité sont devenus aujourd'hui cruciaux car les applications composant les systèmes d'information d'entreprise ont besoin de plus en plus de travailler ensemble. Cela peut être dans le cadre intra-entreprise où il s'agit principalement d'interconnecter diverses applications hétérogènes d'une même entreprise, ou dans le cadre inter-entreprise pour faire communiquer des applications de plusieurs partenaires, ou encore dans cadre des logiques de fusion-acquisition d'entreprises pour unifier les systèmes d'information impliqués.

Ce chapitre a pour objet de présenter la problématique générale de l'intégration et de l'interopérabilité des systèmes d'information. Après avoir présenté les systèmes d'information d'entreprise et leurs caractéristiques générales, nous allons présenter de façon macroscopique le concept d'intégration à travers les principales dimensions qu'il présente et le situer par rapport au concept d'interopérabilité.

II.2. Système d'information d'entreprise

Dans cette section nous allons présenter d'une façon générale la notion de système d'information d'entreprise ainsi que les principales caractéristiques des applications qui composent ce système.

II.2.1. Notion de système d'information d'entreprise

Signalons d'emblée que la notion de *système d'information* est liée à celle d'organisation. Rappelons brièvement qu'une organisation est le siège d'activités les plus diverses visant à créer de la valeur et dans laquelle l'information est considérée comme une essence et une ressource vitale pour son fonctionnement. Dans le cadre de nos travaux, nous nous intéressons aux entreprises et aux systèmes d'information d'entreprise.

Diverses définitions ont été données pour cette notion de système d'information. Nous pouvons, cependant, retenir qu'un système d'information est un ensemble organisé de ressources:

- matériels (machines informatiques, supports, ...)
- personnel (utilisateurs, informaticiens, ...)
- données (connaissances, modèles, …)
- logiciels et procédures (programmes informatique, des méthodes de travail, …)

permettant

- d'acquérir
- de traiter
- de stocker
- et de communiquer

des *informations* sous des formes variées au sein d'une entreprise [Reix 1995] (figure II.1).

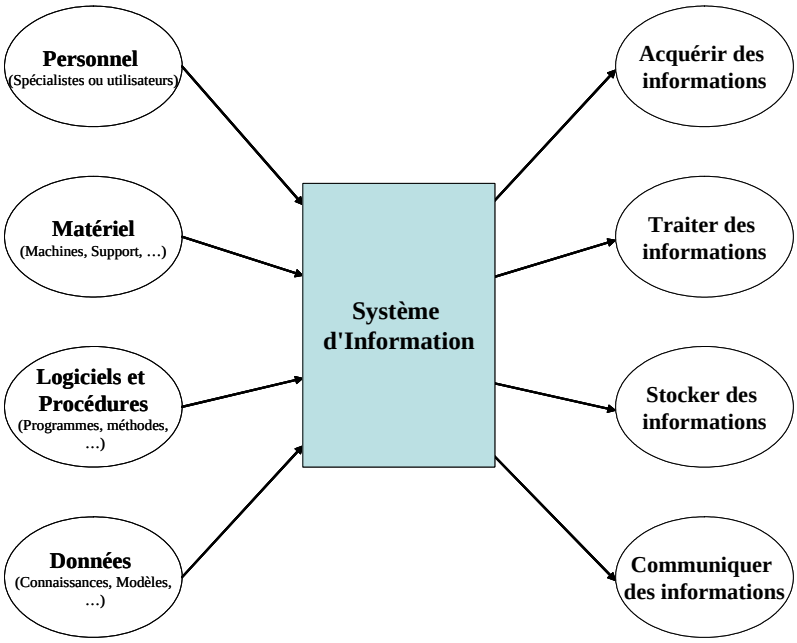

Figure II.1. Structure d'un système d'information

Du point de vue systémique, un système d'information d'Entreprise peut être perçu comme le sous-système de l'entreprise, donc un système, qui englobe tous les composants dont les interactions sont de type *informationnel*. Son objectif est de fournir aux différents niveaux de l'entreprise les informations nécessaires permettant de la faire fonctionner [Ermes 1994]. On retrouve dans ce même courant d'idées la définition de J. L. Lemoigne [Lemoigne 1984] qui aborde les Systèmes d'Information avec une

approche systémique qui consiste à définir une organisation comme la composition de trois types de systèmes qui sont (figure II.2):

- Le *système opérant*, qui réagit aux événements quotidiens, qui viennent de l'environnement, selon des règles définies. Il est chargé de transformer des ressources ou flux primaires (flux financiers, flux de personnel, flux de matière, flux d'information).
- Le *système de pilotage*, qui permet d'engager le processus de décision tout en définissant au préalable les objectifs, les critères d'évaluation et les règles de gestion. Il dirige l'entreprise et maintient le cap sur les objectifs choisis.
- Le s*ystème d'information*, qui relie les deux systèmes précédents tout en jouant un rôle de coupleur. Il correspond à la partie chargée de la collecte, du traitement, du stockage et de la diffusion des informations. Il peut être perçu comme une représentation de l'activité du système opérant et/ou du système de pilotage.

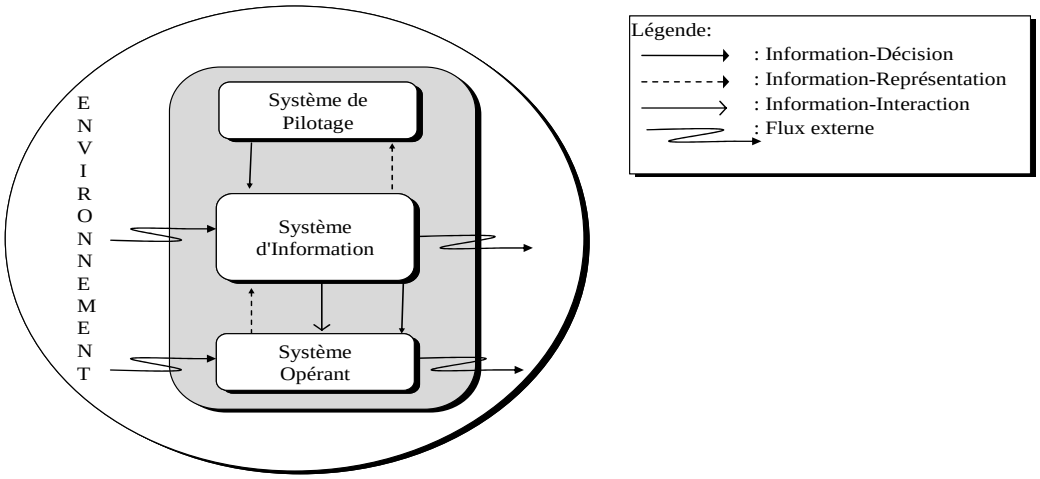

Figure II.2. Vue systémique d'un système d'information [Tardieu et al. 2002]

Cette vue systémique du système d'information est à l'origine même de la typologie des Systèmes d'Information qui distingue selon leur finalité principale: des systèmes d'information supports d'opérations (traitement de transaction, contrôle de processus industriels, etc.) et des systèmes d'information supports de gestion (aide à la production de rapports, aide à la décision, etc.) [Tessier 1995].

Il est fondamental de noter que l'informatique joue un très grand rôle dans la réalisation des Systèmes d'Information. Aussi, il nous arrive souvent de confondre s*ystème d'information* et *système informatique*, négligeant souvent ainsi l'existence de Systèmes d'Information manuels. Le Système Informatique ne représente en fait qu'une partie du système d'information, plus précisément la partie automatisée du système d'information, également appelée la partie TIC pour Technologie d'Information et de Communication ou la partie IT pour Information Technology dans les pays anglo-saxons. Du point de vue informatique, nous considérons un système d'information d'entreprise principalement comme un ensemble d'applications (logiciels, bases de données) qui équipent le système informatique de l'entreprise.

II.2.2. Notion d'application d'entreprise

Dans un système d'information d'Entreprise, nous pouvons trouver une multitude d'applications pouvant répondre à certains besoins. Dans la littérature, le terme "application" peut avoir plusieurs sens. Cependant, dans le cadre de nos travaux, nous utilisons ce terme au sens d'application informatique de l'entreprise et nous appelons "application" (ou application d'entreprise) un ensemble de programmes (logiciels) articulés entre eux, utilisés pour automatiser ou assister des tâches de traitement de l'information dans un domaine particulier au sein de l'entreprise.

Une application est principalement caractérisée par [Leroux 2004][Reix 1995] :

- les composants applicatifs qu'elle comporte et qui représentent des sous-ensembles cohérents (modules logiciels) de l'application;
- le ou les champs d'application (contexte d'utilisation) définis soit structurellement (un poste de travail, un département, etc.) soit fonctionnellement (une fonction de gestion telle que la gestion de la maintenance, la gestion des commandes, etc.);
- des fonctionnalités qui correspondent à un ensemble de tâches qui peuvent être prises en charge par l'application;
- les données manipulées qui correspondent aux différentes données utilisées et produites par l'application;
- et les différentes ressources humaines, logicielles, et matérielles utilisées par l'application.

Nous avons synthétisé dans le méta-modèle de la figure II.3 les principales caractéristiques de la notion d'application et des composants applicatifs.

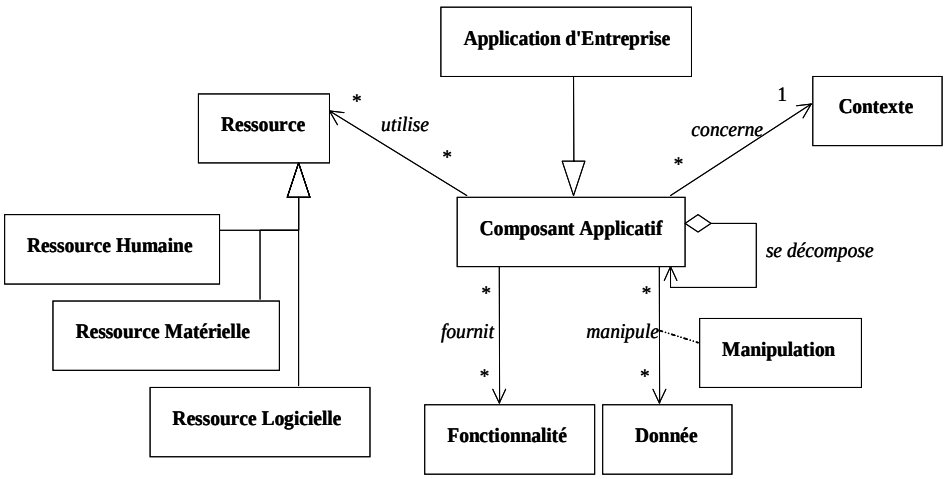

Figure II.3. Notion d'application[4]

Typiquement, dans une entreprise de production, on peut classifier les applications selon leur nature : progiciel ou application spécifique. Les progiciels sont des applications achetées clés en main et qui peuvent être éventuellement adaptées au contexte spécifique de l'entreprise. Les applications spécifiques, qui font généralement l'objet de conception et développement internes, sont développées à façon et qui permettent de

[4] [4] Signalons que nous nous basons tout au long de ce chapitre et des chapitres suivants sur le formalisme UML pour représenter nos différents modèles et méta-modèles. Aussi, le sens des symboles graphiques utilisés dans ces modèles est celui préconisé dans UML 2.0. En particulier, les flèches désignent la relation de spécialisation/généralisation entre classes, les arcs possédant un losange à l'extrémité désignent des relations d'agrégation entre classes, et les arcs simples désignent les autres types de relations entre classes.

répondre à des fonctionnalités qui ne sont pas couvertes par les standards ou les progiciels du marché.

Généralement au sein des entreprises de production, on peut trouver différents types de progiciels qui sont :

- un PGI - pour Progiciel de Gestion Intégré (en anglais : ERP- Enterprise Resource Planning) – permettant d'offrir des fonctions de base pour la gestion des stocks et de la comptabilité de l'entreprise, etc.;
- une application de GRC - Gestion de la Relation Client (en anglais : CRM - Customer Relationship Management) – qui regroupe toutes les fonctions permettant d'intégrer les clients dans le système d'information de l'entreprise;
- une application de GCL - Gestion de la Chaîne Logistique (en anglais : SCM - Supply Chain Management) qui regroupe toutes les fonctions permettant d'intégrer les fournisseurs et la logistique au système d'information de l'entreprise;
- une application de GRH - Gestion des Ressources Humaines (en anglais : HRM – Human Resource Management) - qui permet d'offrir toutes les fonctions permettant la gestion du personnel de l'entreprise;
- une application de GDT - Gestion de Données Techniques (ou PDM - Product Data Management) – qui permet d'offrir des fonctions d'aide au stockage et à la gestion des données techniques, surtout utilisé par les bureaux d'études;

L'état de l'art actuel des systèmes d'information montre une certaine convergence en matière d'architecture vers la structuration des applications selon trois tiers dénommés le *front-end*, le *middle-end* et le *back-end*. Le *front-end* comprend l'ensemble des applications qui interagissent directement avec l'utilisateur. Le *back-end* comprend les applications liées aux systèmes de production de l'entreprise dans lesquels se concentre la valeur ajoutée de l'entreprise. Le *middle-end* comprend l'ensemble des applications et des technologies nécessaires pour interconnecter le front-end et le back-end.

De plus, l'état de l'art en matière d'architecture applicative propose de classifier les applications selon trois principaux styles architecturaux en fonction de la structuration physique des couches logiques d'une application (couche présentation, couche applicative et couche données). Les différents styles architecturaux sont : 1-tier (qui correspond aux situations d'applications monolithiques construites sur le principe d'un niveau physique qui englobe les trois couches logiques), 2-Tiers (qui correspond aux applications construites selon le mode client/serveur) et 3-tiers (qui implique une séparation physique des trois couches logiques au sein des applications).

Au delà de ces typologies, il est généralement admis de classifier les applications d'entreprise selon le mode de traitement des événements : mode par lot (différé) et mode unitaire (immédiat). Ceci permet de distinguer [Manouvrier 2001]:

- les *applications batch* (AB) : qui sont les applications les plus anciennes et qui sont conçues avec une philosophie d'application monolithique lourde et qui permettent de traiter en différé un ensemble d'événements groupés dans des fichiers ou éventuellement dans des bases de données;
- les *applications transactionnelles* (AT) : qui sont des applications qui permettent de traiter les événements les uns après les autres et dont la plupart peuvent assurer des interfaces avec les utilisateurs;
- les *applications client/serveur* (AC) : qui sont une forme évoluée des applications transactionnelles et qui reposent sur le modèle client-serveur où une application cliente peut contacter le serveur et lui envoyer des requêtes que le serveur traite pour retourner ses réponses à l'application appelante;

- les *applications web* (AW) : qui sont une forme particulière d'applications transactionnelles qui tournent sur une technologie web;
- et les applications au fil de l'eau (AF) : qui sont des applications relativement récentes et essentiellement asynchrones qui utilisent des technologies de messagerie inter-applications (MOM - Message Oriented Middleware - cf. § III.5.6) pour communiquer avec d'autres applications. Elles fonctionnent selon des logiques qui peuvent relever à la fois des modes batch et transactionnel.

II.2.3. Caractéristiques des applications d'entreprise

Les principales caractéristiques des applications d'entreprise sont [Bussler 2003-b][Hasselbring 2000] :

- l'autonomie;
- la distribution;
- et l'hétérogénéité.

Ces trois caractéristiques sont mutuellement orthogonales les unes avec les autres dans le sens où elles constituent des dimensions indépendantes de l'application. Chacune de ces caractéristiques peut engendrer des problèmes spécifiques pour faire travailler ensemble différentes applications. L'objectif étant de gérer autant que faire se peut ces dimensions afin d'accroître la capacité des applications en matière. Les flèches en pointillé de la figure II.4 indiquent quelques approches générales permettant de gérer ces trois dimensions.

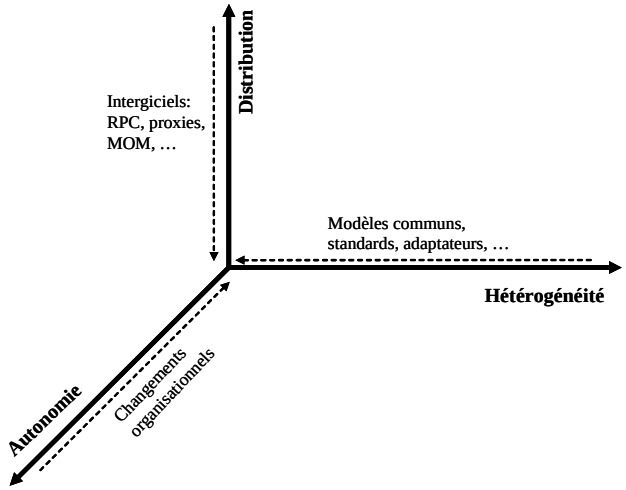

Figure II.4. Dimensions des applications d'entreprise (adapté de [Hasselbring 2000])

A ces trois dimensions, certains auteurs comme [Singh et Huhns 2005] rajoutent une autre dimension appelée dynamisme due au fait que les applications peuvent évoluer avec le temps en fonction des évolutions et des changements qui s'opèrent dans leur environnement.

II.2.3.1. L'autonomie

Les applications d'entreprise sont autonomes dans la mesure où elles peuvent être conçues et exécutées indépendamment les unes des autres. Dans le contexte des bases de données, [Özsu et Valduriez 1999] propose une classification de la notion d'autonomie en définissant plusieurs aspects de cette notion :

- *autonomie de conception* : qui signifie qu'une application est indépendante d'autres applications dans sa conception intrinsèque (son modèle de donnée, son modèle de traitement, etc.);
- *autonomie de communication* : qui signifie qu'une application peut localement choisir avec quelles applications elle peut communiquer;
- *autonomie d'exécution* : qui signifie l'indépendance de l'application pour gérer les interactions avec son environnement extérieur.

II.2.3.2. La distribution

La seconde caractéristique des applications d'entreprise est la distribution et qui désigne le fait que les applications sont très souvent réparties physiquement sur le réseau d'entreprise. Cette répartition physique est réalisée grâce à une répartition des données et/ou des traitements permettant ainsi l'implémentation au niveau local de certaines données et/ou de certains traitements du système d'information.

L'une des techniques qui sont le plus souvent utilisées pour permettre la distribution des applications se base sur la mise en oeuvre d'intergiciels tels que CORBA, Java/RMI, COM/DCOM, MOM, etc. Une présentation brève de ces techniques est donnée au chapitre III.

II.2.3.3. L'hétérogénéité

L'hétérogénéité est une dimension inhérente au fait que les applications d'entreprise peuvent être développées et déployées de façons indépendantes et selon des approches et des méthodologies différentes. L'hétérogénéité peut survenir à différents niveaux et pour des raisons multiples et on distingue principalement [Wiederhold 1992] :

- *l'hétérogénéité technique :* qui correspond aux différences présentes dans les matériels et logiciels de base utilisés. L'hétérogénéité au niveau matériel (hétérogénéité matérielle) comprend les différences liées aux ordinateurs et les réseaux utilisés. L'hétérogénéité au niveau des logiciels de base (hétérogénéité de plateforme) comprend les différences associées aux systèmes d'exploitation, aux systèmes de gestion de bases de données, aux plateformes d'exécution, etc.;
- *l'hétérogénéité syntaxique :* qui correspond aux différences présentes dans les format des données et des interfaces des applications (signature des fonctions). Par exemple, dans un schéma de données, il existe un type de donnée "Matricule_Emp" de type chaîne de caractères et dans un autre schéma on a le type "Numéro_Emp" de type entier. Si ces deux types désignent la même chose (par exemple, le matricule d'un employé), nous pouvons alors dire qu'il existe une différence d'ordre syntaxique et qui peut être résolue par une transformation syntaxique du schéma de données;
- *l'hétérogénéité sémantique :* qui correspond aux différences liées à l'interprétation et au sens associé aux données et aux fonctions d'une application. Cette hétérogénéité exprime le fait que le nom symbolique d'un concept peut être interprété de façon différente suivant les applications considérées. Ces conflits sémantiques se produisent principalement lorsque (1) un même nom symbolique recouvre différents concepts (on parle alors dans ce cas d'homonymie), ou alors (2) que plusieurs noms symboliques recouvrent le même concept (et on parle dans ce cas de synonymie). Un exemple classique de problèmes sémantiques qui peut survenir entre deux applications ou deux personnes, est le problème lié à la

définition de l'entité "client". Un client est-il quelqu'un qui a déjà acheté quelque chose à l'entreprise ? S'agit-il d'une personne ayant manifesté ses intérêts pour les produits de l'entreprise ? S'agit-il d'une personne physique ou morale ? L'existence de problèmes liés à la sémantique est une évidence au sein de toute entreprise. Et il devient vital que l'identification de ces conflits, puis leur résolution se fasse le plutôt possible, de préférence dès les phases amont du projet.

La plupart des auteurs [Hasselbring 2000][Bussler 2004][Dogac 2004] s'accordent sur le fait que l'hétérogénéité des applications constitue la véritable difficulté dans le domaine de l'intégration. Bien que toutes les autres dimensions soient importantes, nous nous focalisons principalement dans le cadre de nos travaux sur la dimension d'hétérogénéité des applications et plus particulièrement sur l'hétérogénéité sémantique qui constitue la difficulté majeure de la problématique d'intégration des applications d'entreprise.

II.2.3.4. Dynamisme

Une autre caractéristique introduite dans [Singh et Huhns 2005] est le dynamisme des applications d'entreprise. En effet, du fait que les systèmes d'information actuels sont ouverts et soumis à de fréquents changements, en réponse à des changements d'ordre stratégique, métier ou technologiques que subit l'entreprise, les applications de ces systèmes doivent alors être capables d'évoluer de façon dynamique pour faire face à ces changements.

Le dynamisme est une dimension qui peut se manifester généralement selon deux aspects. Le premier aspect concerne le dynamisme dans le comportement qu'une application peut afficher de façon autonome selon sa configuration interne. Le second aspect concerne les changements qui peuvent s'opérer au sein des composants applicatifs d'une application tels que la modification de certains composants, l'arrivée de nouveaux composants, la suppression de certains composants jugés obsolètes, l'absence temporaire de certains composants, et la substitution de certains composants.

II.3. Spécificités des systèmes d'information industriels (SII)

Dans le cadre de nos travaux, nous nous intéressons aux systèmes d'information d'entreprises industrielles, et en particulier aux systèmes d'information d'entreprise du domaine de la fabrication de produits microélectroniques. Dans de telles entreprises, de nombreux domaines de natures différentes peuvent collaborer en vue d'atteindre l'objectif final qui est de satisfaire le client. Ceci sous-entend la réussite de l'activité métier qui est, rappelons-le, la production de produits (dans notre cas de composants microélectroniques) avec des contraintes de délai, de qualité, de coût, de performances techniques et niveau de service toujours de plus en plus exigeantes.

II.3.1. Caractéristiques des SII

Mener à bien le processus de production de composants microélectroniques requiert de déployer différentes catégories d'applications à différents niveaux de l'entreprise, chacun devant assurer une fonction particulière comme la gestion des équipements, des tâches de maintenance, etc. Nous avons choisi de présenter brièvement quelques applications les plus représentatives suivant les différents niveaux de l'entreprise (tableau II.1) et qui sont [Toublant et al. 2002] :

- Les systèmes de contrôle-commande;
- Les systèmes de supervision;
- Les systèmes MES (Manufacturing Execution System);
- Les systèmes de GPAO (Gestion de production assistée par ordinateur);
- Les systèmes SGDT (Système de gestion de données techniques).

Domaine	Niveau	Fonctions	Principaux logiciels associés
Planification	5	Gestion de l'entreprise (finances, trésorerie, ressources humaines)	GPAO
	4	Gestion de production Études Achats Maintenance	GPAO
Exécution	3	Suivi de production (qualité, traçabilité....) Ordonnancement en temps réel	MES GPAO
Commande	2	Conduite Pilotage, Supervision	Supervision
	1	Automates Système numérique de contrôle-commande (SNCC) - commande numérique à calculateur (CNC)	Contrôle-commande
	0	Capteurs/actionneurs	
SGDT		Base de données	

Tableau II.1. Principaux progiciels en fonction du niveau de l'entreprise [Toublant et al. 2002]

Les systèmes de contrôle de commande permettent de relier les systèmes d'automatisme de la fabrication aux systèmes de GPAO et de supervision. Les systèmes de supervision ont pour rôle l'acquisition des données provenant des équipements, de fabrication, l'affichage et le traitement de ces données, et la communication avec d'autres applications. Le MES (Manufacturing Execution System) permet d'assurer la remontée intelligente de données quantitatives issues du procédé de fabrication en décloisonnant ainsi l'entreprise entre les niveaux de la supervision et de la GPAO. La GPAO (Gestion de Production Assistée par Ordinateur) intervient principalement pour la planification et le suivi de la fabrication. En outre, on distingue également des systèmes de gestion de données techniques (SGDT) qui constituent une base de données transverse comportant un ensemble d'informations techniques relatives aux différentes étapes du cycle de vie des produits fabriqués.

Ce qui est fondamentalement distinctif dans ces systèmes d'information du domaine de la microélectronique est

- une *forte automatisation* des procédés: cette automatisation implique une forte dépendance de l'informatique. Ce qui nécessite alors d'extraire toute la connaissance embarquée par ces systèmes automatisés en vue d'éventuels transferts de connaissances, apprentissages ou tout simplement de maintenances;
- une forte *évolution des systèmes* : ces systèmes sont fortement évolutifs en raison de l'évolution permanente du métier[5] ce qui implique que ces systèmes doivent être relativement autonomes et faiblement couplés afin de procurer la plus grande souplesse et flexibilité pour le système d'information,
- une *coopération impérative* : ces systèmes doivent impérativement opérer ensemble afin d'assurer l'atteinte de l'objectif global. Chacun de ces systèmes gère les informations qui lui sont propres en utilisant des traitements qui lui sont propres.

[5] Selon la loi de Moore, la technologie des semi-conducteurs évolue rapidement, d'où l'évolution rapide des systèmes d'information.

Dans de tels systèmes, on peut concevoir que la représentation des informations et des fonctionnalités soit spécifique à chacun des systèmes. Il est par contre, impératif que ces systèmes coopèrent de façon flexible afin de permettre les échanges, faciliter la réutilisation de leurs services et favoriser leur modernisation. De plus, il est impératif que la sémantique portée par les données et les fonctions échangés soit la même pour pouvoir assurer une certaine cohérence entre les systèmes. Ainsi, en plus des problèmes d'ingénierie technique que posent ces Systèmes d'Information industriels, vient se greffer un problème crucial, celui de *l'intégration des systèmes*.

II.3.2. Principaux besoins des SII

II.3.2.1. Besoins en intégration

Pour atteindre leur objectif, les entreprises industrielles, et en particulier celles du domaine de la microélectronique, ont besoin de faire interagir différentes applications de leur système d'information. Ce mécanisme d'interaction ou de coopération entre différentes applications est généralement appelé le mécanisme d'intégration. Des exemples typiques de besoins qui peuvent inciter à recourir à l'intégration au sein d'une entreprise sont par exemple [Erl 2004] :

- une application A qui a périodiquement et/ou régulièrement besoin d'accéder aux données de l'application B;
- une application A qui nécessite une fonction de l'application B;
- les applications A et B qui ont besoin de partager un ensemble de données afin d'éviter des redondances;
- les applications A et B qui doivent collaborer par échange de données;
- les applications A et B qui sont orchestrées par une application ad hoc (moteur de workflow) pour réaliser un processus métier;
- etc.

Des exemples plus concrets dans le domaine industriel peuvent être :

- le système de GPAO qui accède périodiquement au système de gestion des données techniques pour extraire des informations portant sur les caractéristiques des équipements;
- le système de GPAO qui accède régulièrement au système de gestion des ressources humaines pour extraire des informations portant sur la disponibilité des opérateurs;
- ou encore la cas d'un utilisateur métier qui désire construire un workflow applicatif en réponse à une demande spécifique impliquant ainsi l'utilisation de plusieurs composants applicatifs, par exemple ceux du système de gestion de production et des relations humaines.

II.3.2.2. Besoin en flexibilité

Outre l'intégration des applications qui constitue le besoin principal des SII, un autre besoin non moins important des entreprises microélectroniques est la flexibilité. La *flexibilité* constitue aujourd'hui une préoccupation majeure des entreprises de production microélectronique qui cherchent plus d'agilité et de réactivité. En effet, les systèmes d'information industriels ne sont généralement pas stables, au contraire ils évoluent constamment. Derrière cette évolution constante se cache l'une des principales

motivations d'une architecture d'intégration flexible appropriée à ce type d'entreprises. Les raisons de l'évolution de ces systèmes d'information peuvent être génériques ou spécifiques.

Les raisons génériques le plus souvent évoquées sont principalement [Octo 2005] :

- Evolution des organisations: Ces évolutions peuvent considérablement impacter le système d'information et peuvent être d'origine interne (évolution engendrée par exemple par la restructuration des organisations ou la création de nouvelles filiales ou de nouvelles implantations géographiques) ou externe (évolution engendrée par exemple par la fusion ou acquisition d'organisations).
- Evolution des réglementations: Ces évolutions peuvent impacter le système d'information de manière exogène (par exemple lors du passage à l'Euro ou lors du changement des règles de calcul imposées par l'Etat ou par les collectivités locales). Ces évolutions obligent souvent à repenser et/ou à moderniser le système d'information.
- Evolution du métier: Ces évolutions sont principalement dues à l'évolution de l'activité de l'entreprise (par exemple la production d'une nouvelle gamme de produit). Ces évolutions du métier impactent les processus métier et par conséquent le système d'information.
- Evolution des technologies : Ces évolutions sont principalement dues à l'évolution des produits logiciels, des modèles d'architecture (mainframe, client-serveur, n-tiers, …), des paradigmes de programmation (C++, Java, …) et des infrastructures informatiques (J2EE, .NET). Ces évolutions impactent les systèmes d'information dans la mesure où elles se traduisent le plus souvent par des efforts d'intégration entre technologies hétérogènes.
- Maîtrise des coûts: Elle constitue souvent une exigence formulée aux directions des systèmes d'information à qui on demande de réduire et/ou de maîtriser ses coûts. Ceci se traduit le plus souvent par une démarche de rationalisation du système d'information (par exemple l'unification des architectures applicatives ou la mutualisation des applicatifs de l'entreprise).

En ce qui concerne les raisons spécifiques au domaine microélectronique, elles sont principalement dues au fait que les entreprises de fabrication microélectronique sont soumises en particulier au phénomène de changements fréquents de leur technologie de fabrication ce qui engendre une forte évolutivité de leurs systèmes d'information [Chapron 2006].

Face à toutes ces évolutions et exigences, il devient nécessaire de structurer le système d'information de manière à faciliter ces évolutions tout en s'insérant dans le cadre d'une approche d'intégration de systèmes hétérogènes.

II.4. Concept d'intégration et d'interopérabilité

Les problèmes d'intégration et d'interopérabilité ne sont pas nouveaux et ils constituent des domaines complexes auxquels s'intéressent plusieurs communautés dont celle des systèmes d'information, celle des bases de données, celle du génie logiciel, celle de la fouille de données, etc. Plusieurs approches, standards et/ou technologies ont été proposés pour résoudre ces problèmes. Dans cette section, nous allons présenter quelques notions générales de ces concepts.

II.4.1. Définitions

II.4.1.1. Intégration

Selon le dictionnaire, l'intégration désigne l'action d'intégrer. De son étymologie latine "integrare", intégrer signifie "rendre entier"; faire entrer dans un ensemble, dans un groupe plus vaste. L'intégration est de ce fait l'acte d'incorporation d'éléments constitutifs par lequel on va rendre un ensemble complet, lui conférer les propriétés attendues, et dans notre cas des propriétés essentiellement liées à la connexion et à la cohérence des systèmes d'information.

De façon générale, l'intégration consiste à combiner des composants de telle façon à former un nouvel ensemble constituant un tout pour créer de la synergie [Weston 1993]. Dans le contexte de l'entreprise, le problème d'intégration est communément appelé intégration d'entreprise (EI - Enterprise Integration). [Vernadat 1996] définit l'intégration d'entreprise comme étant le processus qui "consiste à faire interopérer fortement les personnes, les machines et les applications afin d'accroître la synergie au sein de l'entreprise" [Vernadat 1996]. Dans le même ordre d'idées, [Williams et Li 1999] définissent l'intégration d'entreprise comme "la coordination de tous les éléments incluant les processus métier, les ressources humaines, et la technologie d'une entreprise fonctionnant ensemble dans le but d'atteindre la réalisation optimale de la mission de cette entreprise telle que définie dans la stratégie de l'entreprise".

L'intégration peut revêtir plusieurs formes. Elles s'étendent des approches faiblement couplées jusqu'aux approches fortement couplées et qui sont [Petrie 1992] : approche unifiée, approche fédérée, approche intégrée. Au sein d'une entreprise, il existe différents niveaux d'intégration. Chaque niveau englobe les niveaux inférieurs. Ces niveaux d'intégration sont (figure II.5) [Vernadat 2002] :

- *L'intégration des systèmes physiques :* Elle concerne essentiellement les systèmes de communication, c-à-d l'interconnexion et les échanges de données au moyen de réseaux et de protocoles de communication. Ce type d'intégration date du début des années 70 et sont encore en cours d'évolution. Les principaux travaux qui ont été réalisés dans le cadre de ce type d'intégration sont notamment la définition du standard ISO/OSI (Open System Inteconnection), la définition des protocoles industriels tels que MAP, TOP, ou encore les récents développements autour d'ATM, fast Ethernet et Internet.
- *L'intégration d'applications :* Elle porte aussi bien sur l'interconnexion et la coopération des applications hétérogènes que sur l'accès aux données partagées par des applications distantes. L'intégration d'applications a vu le jour vers le milieu des années 80 et elle constitue de nos jours un domaine très actif. Parmi les travaux réalisés dans ce cadre, nous pouvons citer STEP, EDI, HTML, XML, eb-XML, Web services, les plateformes d'intégrations (OSF/DCE, CORBA, EAI) et les systèmes de gestion des Workflow.
- *L'intégration métier :* Elle porte sur l'intégration au niveau entreprise, c-à-d la coordination des processus métier. Ceci concerne principalement la coopération d'entreprises et qui nécessite la modélisation précise des règles métier. Parmi les travaux effectués dans ce domaine, on peut citer les initiatives ICAM, IPAD, CALS, CIMOSA, AIT.

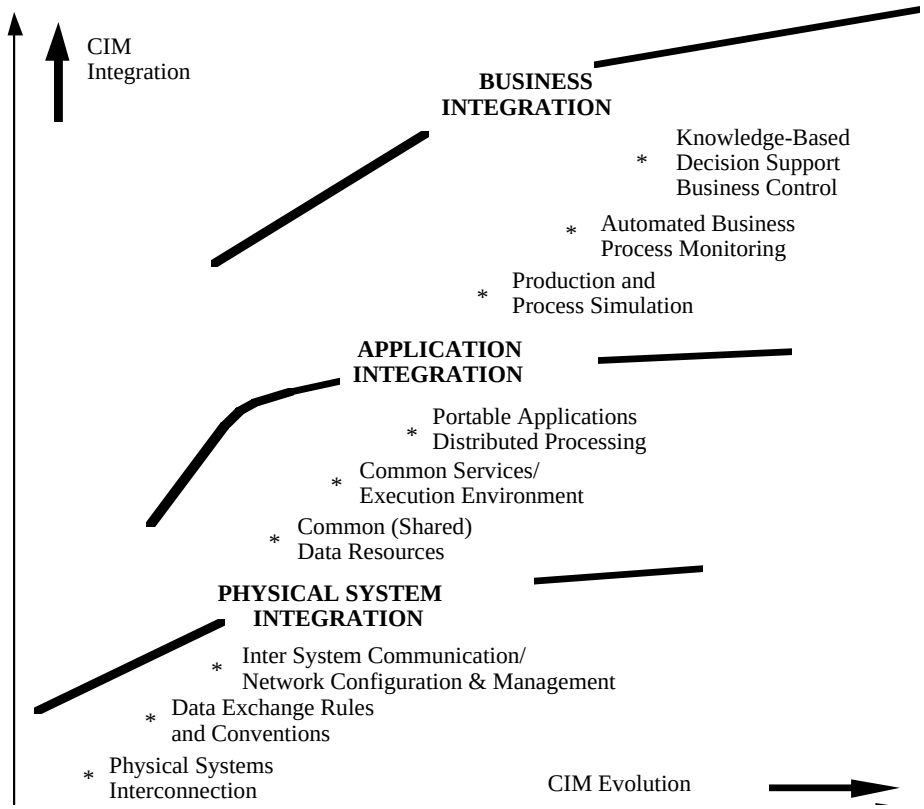

Figure II.5. Les trois niveaux d'intégration d'entreprise [AMICE 1993]

Dans le cadre de nos travaux, nous nous intéressons à l'intégration d'applications d'entreprise. Au coeur de cette problématique se pose parfois le problème d'*interopérabilité* des applications. Nous allons, dans ce qui suit, définir le concept d'interopérabilité (§ II.4.1.2) et nous allons ensuite le situer par rapport au concept d'intégration (§ II.4.1.3).

II.4.1.2. Interopérabilité

Le dictionnaire définit l'*interopérabilité* comme étant la compatibilité des équipements, des procédures ou des organisations permettant à plusieurs systèmes d'agir ensemble. Dans le domaine des applications d'entreprise, l'interopérabilité de deux applications ou de deux composants d'un système hétérogène distribué est défini comme l'aptitude de ces composants ou de ces applications à échanger des données et des fonctionnalités [Vernadat 1996][Wegner 1996]. Le standard d'IEC [IEC 2000] définit le concept d'interopérabilité dans le domaine du génie logiciel comme un degré de compatibilité (figure II.6). Dans ce standard, on considère que l'interopérabilité est réalisée si l'interaction peut au moins exister à l'un des trois niveaux : données, fonctionnalités et processus (comportement). Comme on peut le remarquer sur la figure II.6, le plus haut degré de compatibilité du standard IEC et qui englobe le concept d'interopérabilité est l'interchangeabilité. Cette dernière repose sur le fait que chaque système puisse être remplacé par un autre système similaire, - présentant des données similaires, des fonctionnalités similaires et des comportements similaires - tout en continuant à fonctionner.

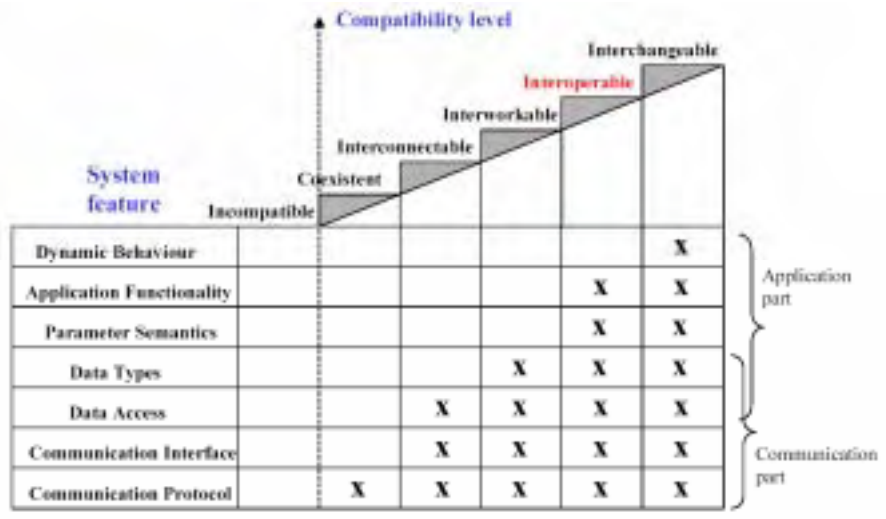

Figure II.6. Interopérabilité en tant que niveau de compatibilité [IEC 2000]

II.4.1.3. Discussion

A travers les définitions précédentes, on peut remarquer qu'il existe une certaine ambiguïté entre l'intégration et l'interopérabilité. Ces deux termes peuvent devenir synonymes seulement dans le cas où l'on s'intéresse à l'intégration faiblement couplée, c'est-à-dire que l'on cherche à préserver l'hétérogénéité et l'autonomie des systèmes constituants. L'interopérabilité est alors considérée comme une forme intrinsèquement plus souple dans la mesure où, contrairement à l'intégration, les systèmes composants sont censés préserver leur identité [Lapassat 2003]. Ainsi, on utilise généralement dans la littérature le vocable d'interopérabilité de systèmes quand les systèmes concernés sont aptes à s'échanger des informations et à agir ensemble, et on réserve le vocable d'intégration de systèmes pour le cas où les systèmes coopèrent au sein d'un système unique et homogène [Meinadier 2002]. A ce titre, [Meinadier, 2002] précise que l'interopérabilité de systèmes se place ainsi au niveau du partage d'informations et de services et implique des adaptations, notamment sémantiques, entre les modèles de données, tandis que l'intégration entre systèmes se place au niveau du comportement global: elle implique, de plus, l'intégration fonctionnelle et l'enchaînement des traitements entre systèmes d'information. Ceci rejoint [Vernadat 1996] qui considère aussi le concept d'intégration comme un concept plus large que celui de l'interopérabilité dans la mesure où il englobe selon lui des capacités liées à la communication, la coopération et la coordination des composants du système.

D'autres auteurs, par contre, s'accordent plutôt sur le fait que les solutions d'interopérabilité peuvent osciller entre deux types de couplage entre les systèmes devant interopérer, selon que l'on privilégie l'indépendance des systèmes constituants, on parle de fédération (ou d'interopérabilité), ou plutôt la cohérence globale d'ensemble, on parle alors d'intégration. Dans ce même courant d'idées, [ISO 14258] et [Chen et Doumeingts 2003] en reprenant la classification de [Petrie 1992] ont proposé trois formes fondamentales d'interopérabilité qui peuvent exister entre deux entités et qui sont: l'intégration, l'unification et la fédération.

En résumé, nous allons considérer, dans le cadre de nos travaux, que l'intégration est un concept générique incluant le concept d'interopérabilité, tel qu'il a été défini dans [Vernadat 1996]. De plus, nous allons nous focaliser, par la suite, sur le concept d'intégration.

II.4.2. Enjeux et défis de l'intégration

Traditionnellement, on peut citer deux facteurs principaux qui peuvent pousser au développement et à la mise en place d'un projet d'intégration dans les entreprises [Manouvrier 2001]. Il s'agit, tout d'abord, du facteur lié à la nature hétérogène du système d'information qui oblige alors à établir des passerelles entre ses différentes composantes pour en exploiter toute sa richesse et sauvegarder sa cohérence. Il s'agit, ensuite, du facteur lié à l'agilité de l'entreprise qui se décline en flexibilité du système d'information et qui est désormais devenue une qualité nécessaire et fondamentale à tout système d'information moderne, qui se doit de prendre en compte les différents changements qui peuvent survenir dans le contexte actuel de concurrence rude caractérisé par un "time to market" de plus en plus réduit.

Nous pouvons souligner que le facteur lié à l'hétérogénéité du système d'information s'est trouvé accéléré par la récente transition au best-of-breed[6] ou aux applications COTS[7] [Boehm et Abts 1999] qui commencent à devenir le quotidien de nos systèmes d'information actuels. Ces composants très hétéroclites, additionnées aux applications patrimoines héritées du passé informatique, imposent une structure hétérogène que beaucoup qualifient d'architecture spaghetti [Octo 1999]. Quant au facteur associé à l'agilité des systèmes d'information, ce dernier est lié à la nécessité d'intégrer les données, les applications et les processus de l'entreprise. L'intégration est d'autant plus nécessaire que chaque application constitue très souvent un îlot quasi autonome aussi bien du point de vue informationnel [Stonebraker 1999] que du point de vue fonctionnel [Lubinsky 2000]. Et comme aucun des îlots ne contient à la fois ni toute l'information complète ni toutes les fonctions, il en résulte alors la nécessité d'intégrer ou de faire interopérer ces multiples îlots pour pouvoir unifier et partager les données et les fonctions. D'autres raisons peuvent être citées dont notamment la mise en ligne et l'intégration entre partenaires, qui ne peuvent être mis en œuvre que si les systèmes internes sont intégrés [Lubinsky 2000].

II.4.3. Approches d'intégration

Traditionnellement, on distingue la typologie qui comprend principalement deux approches permettant de faire communiquer et coopérer des applications en pratique [Chen et Doumeingts 2003] : l'adhésion à des interfaces standardisées et l'utilisation d'intergiciels pour faire communiquer et faire coopérer diverses applications. Un exemple typique de la première approche est l'ensemble des standards industriels qui ont été développés tels que XML [W3C-XML 2004], EDI [Chiaramonti 2005], STEP [Sharon et Kemmerer 1999], etc. La seconde approche comprend les architectures et les technologies liées aux intergiciels telles que l'architecture CORBA [OMG-CORBA 1998], l'architecture DCOM [Microsoft et DEC 1995], ou encore les intergiciels orientés messages (MOM – Message-Oriented Middleware) [Ribière 1998] ou les outils EAI (Enterprise Application Integration) [Linthicum 2004].

La typologie du paragraphe précédent porte beaucoup plus sur les techniques d'intégration qui peuvent être mises en oeuvre. Une analyse plus fine de l'état de l'art du domaine de l'intégration des applications d'entreprise permet de révéler en fait une

[6] Best of breed : Littéralement, le meilleur de sa catégorie, se dit d'une solution logicielle prétendant offrir des fonctions avancées sur un segment de marché bien délimité. Cette notion s'oppose à celle de "tout intégré", solution se démarquant par la polyvalence de sa couverture fonctionnelle.

[7] COTS (Commercial On The Shelf) : Littéralement des composants applicatifs disponibles sur le marché (sur étagère).

multitude de dimensions permettant à la fois de caractériser la notion d'intégration et de classifier les approches d'intégration existantes. Aussi, nous avons synthétisé[8] dans le cadre de nos travaux les principaux aspects de l'intégration à travers quatre dimensions fondamentales qui sont :

- les périmètres d'intégration [Singh et Huhns 2005][Vernadat 1996][Wangler et Paheerathan 2000],
- les points de vue d'intégration [Hasselbring 2000],
- les couches d'intégration [Dip 2005][NIST 2005][Linthicum 2004][IAC 2003][Wasserman 2000][Linthicum 1999][Stonebraker 1999],
- et les niveaux d'intégration [Xu et al. 2003][Sheth 1999][Bruijn 2003-b].

Chacune de ces dimensions est étudiée ci-dessous.

II.4.3.1. Les périmètres de l'intégration

Il est possible de classifier les approches de l'intégration en fonction du périmètre couvert par les applications mises en jeu. On distingue essentiellement deux champs ou périmètres d'intégration: l'intégration intra-entreprise et l'intégration inter-entreprise [Vernadat 2002][Singh et Huhns 2005].

L'intégration intra-entreprise concerne les scénarios d'intégration impliquant des applications internes à l'entreprise. L'intégration intra-entreprise est généralement décomposée en deux sous-classes d'intégration [Vernadat 2002][Wangler et Paheerathan 2000] : l'intégration horizontale et l'intégration verticale.

L'intégration horizontale concerne l'intégration logique (conceptuelle) et physique (interne) des processus métier de la demande de produit jusqu'à la livraison de celui-ci. Un exemple typique de ce type d'intégration est la gestion de la chaîne logistique où il s'agit principalement d'optimiser l'ensemble des activités liées à la commande, à la production et à la livraison du produit.

L'intégration verticale concerne l'intégration des différents niveaux de gestion d'une entité organisationnelle et concerne fondamentalement le flux décisionnel qui transite habituellement de façon verticale, i.e.: les décisions sont transmises d'un niveau décisionnel vers le niveau inférieur et le feedback informationnel est transmis de façon ascendante d'un niveau vers le niveau supérieur (figure II.2). Un exemple typique de ce type d'intégration est l'intégration du système de gestion de la production (GPAO) avec les autres applications permettant de gérer les ressources de l'entreprise (GRH).

L'intégration inter-entreprise implique, quant à elle, des applications appartenant à des entreprises différentes en s'insérant ainsi dans le cadre de l'intégration B2B (Business to

[8] On trouve dans la littérature plusieurs classifications, cadres et modèles d'intégration et /ou d'interopérabilité qui définissent un certain nombre de dimensions de l'intégration (et de l'interopérabilité) :
- le modèle de maturité d'AMS (Amercian Management Systems) [Schmidt 2000],
- le cadre AIF (Athena Interoperability Framework) [Athena 2005],
- le cadre IIF (Ideas Interoperability Framework) [Ideas 2004],
- le cadre EIF (European Interoperability Framework) [EIF 2004],
- le modèle LISI (Levels of Conceptual Interoperability Model) [C4ISR 1998],
- le modèle LCI (Layers of Coalition Interoperability) [Tolk 2003][Tolk et Muguira 2003],
- le modèle SOSI (System of Systems Interoperability Model) [Morris et al. 2004],
- le modèle NC3TA (NATO C3 Technical Architecture) [Hoffman 2003],
- le modèle OIMM (Organizational Interoperability Maturity Model) [Clark et Jones 1998],
- le modèle d'ECIMF (E-Commerce Integration Mata-Framework) [ECIMF 2001][Bialecki 2001],
- le modèle ECMA/NIST (European Computer Manufacturers Association/National Institute of Standards and Technology) [NIST 2005],
- le modèle Double Toaster de ECMA/NIST [NIST 2005],
- le modèle ISO 19101 et 19119 ODP/IRM (ODP/Interoperability Reference Model) [Ideas 2004].
- le modèle 7LP (7 layer Protocol) [Herzum 1999].

Business). Très souvent, on ajoute au précédent type d'intégration, l'intégration B2C (Business to Customer) et on parle ainsi d'intégration extra-entreprise (figure II.7). Bien que les périmètres d'intégration peuvent différer, les techniques utilisées, comme nous allons le voir par la suite, demeurent toutefois quasiment les mêmes [Manouvrier 2001].

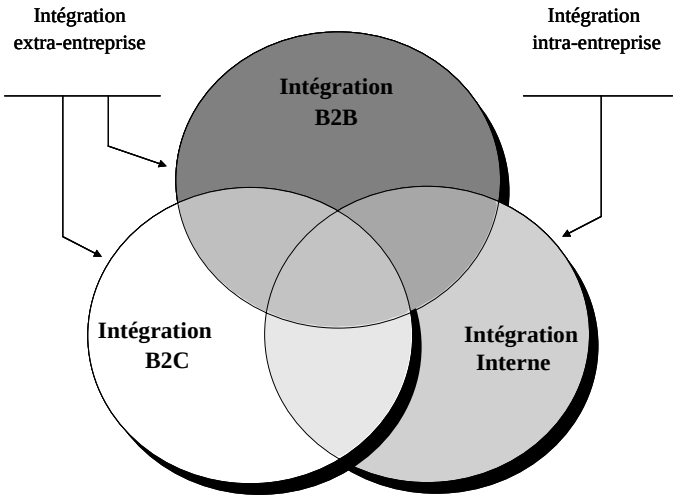

Figure II.7. Les périmètres de l'intégration

Des conflits peuvent apparaître aussi bien au niveau intra-entreprise qu'inter-entreprise. Dans le premier cas, on parle d'hétérogénéités internes à l'entreprise tandis que dans le second cas on parle d'hétérogénéités externes. L'intégration doit être réalisée aussi bien en interne qu'en externe.

II.4.3.2. Les points de vue de l'intégration

Le second critère de classification des approches d'intégration est la notion de point de vue dans lequel l'intégration peut être examinée. Nous nous inspirons de ce qui se fait pour l'interopérabilité, pour définir les points de vues de l'intégration. A ce titre, [Hasselbring 2000] considère trois principaux points de vue qui sont : la vue externe (utilisateur), la vue conceptuelle (concepteur) et la vue interne (programmeur). La figure suivante permet d'illustrer la structure d'un ensemble d'applications et leur intégration au niveau de chacun de ces points de vue.

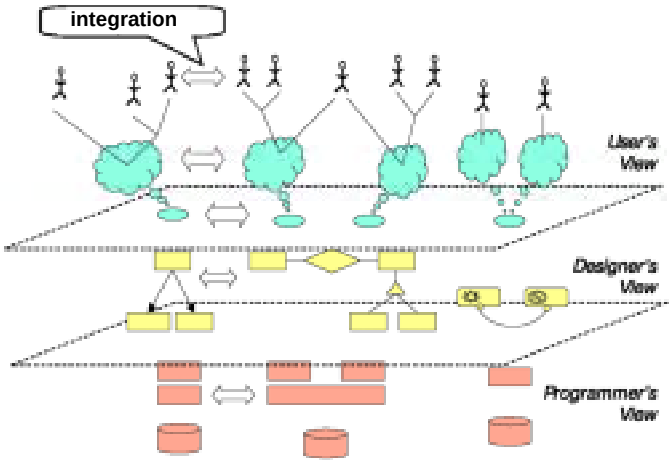

Figure II.8. Les points de vue de l'intégration (adapté de [Hasselbring 2000])

Le point de vue utilisateur concerne les différentes vues que possèdent des experts de domaine et les utilisateurs métier. Le point de vue concepteur concerne les différents modèles utilisés lors de la conception des systèmes d'information ou des applications. Le point de vue programmeur réfère à l'implémentation du système d'information ou des applications.

Des conflits peuvent apparaître au niveau de chacun de ces points de vue, et l'intégration doit être réalisée au niveau de tous les points de vue : les utilisateurs d'une application doivent comprendre les informations qui parviennent d'autres applications, les concepteurs d'une application doivent interpréter les données et les modèles des autres applications, et les programmes doivent automatiser les échanges des données et le partage des fonctionnalités. Bien entendu, la difficulté majeure de l'intégration se trouve au niveau externe et conceptuel [Abiteboul et al. 2000].

II.4.3.3. Les couches de l'intégration

L'intégration peut concerner une ou plusieurs couches du système d'information d'Entreprise. L'analyse de la littérature sur les couches d'intégration de systèmes d'information permet de mettre en évidence une multitude d'approches pour intégrer des applications d'entreprise. Linthicum définit deux niveaux d'intégration : intégration au niveau des données et l'intégration au niveau des processus. L'intégration au niveau des processus se décline à son tour en intégration au niveau des interfaces d'application, au niveau des méthodes et au niveau de l'interface utilisateur [Linthicum 1999][Linthicum 2004]. Stonebraker identifie, quant à lui, deux niveaux d'intégration, l'intégration par les données et l'intégration par les événements [Stonebraker 1999]. De son côté, Lublinsky définit trois types d'intégration d'applications: intégration par les données, par les messages, et par les processus [Lublinsky 2001].

Analysées de près, ces taxonomies sont en fait très similaires. Comme nous pouvons le constater, la couche de données est un élément commun à toutes les taxonomies. La combinaison du niveau interface d'application et niveau interface utilisateur de Linthicum correspond plus ou moins au niveau message de Lublinsky. Le niveau événement de Stonebraker correspond à un concept plus large incorporant le niveau méthode de Linthicum et le niveau message et processus de Lublinsky. Ces multiples taxonomies sont donc globalement équivalentes et elles rejoignent toutes les approches classiques proposées par Wasserman et NIST dans le cadre de l'intégration d'outils en génie logiciel et qui sont principalement [Wasserman 2000][NIST 2005] (figure II.9):

- l'intégration par les données (data integration),
- l'intégration par les traitements (control integration),
- l'intégration par les présentations (presentation integration),
- et enfin, l'intégration par les processus (process integration).

Ces différentes approches d'intégration seront par la suite brièvement présentées selon cette dernière taxonomie (figure II.9). Comme nous pouvons le remarquer, ces différentes couches d'intégration résultent des multiples niveaux de structuration du système d'information d'entreprise et qui sont principalement [Club-Urba 2005][Alonso 2002] :

- la couche données qui traite l'accès aux données manipulées,
- la couche de traitements d'application qui correspond aux règles métier implémentées,
- la couche présentation qui gère l'interaction homme-machine,
- et la couche processus qui permet l'orchestration des applications.

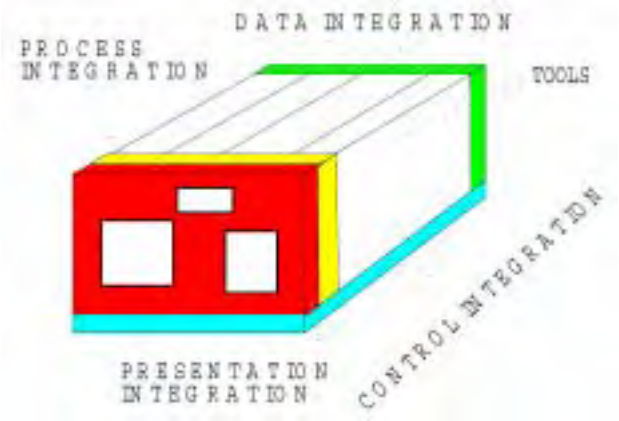

Figure II.9. Les différentes couches d'intégration [NIST 2005]

Signalons que des conflits peuvent apparaître au niveau de chacune de ces couches, et idéalement l'intégration doit être réalisée au niveau de toutes les couches.

L'intégration par les données constitue la manière la plus répandue pour intégrer des applications. Elle consiste généralement à déplacer les données entre les applications en utilisant très souvent des outils de transfert, de réplication ou de fédération de données. Bien que relativement élémentaire, cette approche présente tout de même l'avantage de ne pas induire de changement ni au niveau de la couche présentation ni au niveau de la couche traitement, ce qui permet d'économiser énormément sur le plan financier. Cependant, l'inconvénient majeur réside parfois dans le fait que l'on peut facilement introduire des incohérences au niveau des données en outrepassant les contraintes d'intégrité [Schmidt 2000].

L'intégration par les traitements permet d'intégrer des applications en partageant des services offerts par les composants applicatifs (méthodes, objets, etc.). Cette approche repose le plus souvent sur la notion d'événements qui permet d'invoquer ces services. A titre d'exemple, on peut citer des techniques basées sur le mécanisme traditionnel d'API (Application Programming Interface) [Linthicum 2004], d'application composites [Manouvrier 2001], ou encore les récents Web Services [Linthicum, 2004]. C'est ce type d'approche qui se trouve généralement à la base de la mise en place d'une application client sur le web ou sur l'intranet. Cette approche offre ainsi l'avantage du partage de la logique applicative entre les applications et/ou les processus, ce qui permet une plus grande réutilisation, distribution ainsi que le support des transactions [Linthicum 2001].

L'intégration par les présentations représente sans doute le type d'intégration le plus léger dans la mesure où elle permet d'intégrer les applications au niveau de l'interface utilisateur en fournissant le plus souvent une interface commune. Largement utilisée pour les applications de l'Internet et d'Intranet à travers notamment les techniques de screen-scrapping et de portails d'entreprise, cette approche ne se préoccupe cependant pas de l'intégration ni des données ni des traitements et qui sont laissés très souvent ainsi à l'initiative de l'utilisateur [Schmidt 2000].

L'intégration par le processus constitue sans doute une des formes les plus évoluées afin d'interconnecter et de faire coopérer des applications. Cette approche se base très souvent sur l'utilisation de la notion de gestion de processus (BPM – Business Process Management) et/ou d'intégration de processus métier (BPI - Business Process Integration) qui permettent respectivement de définir et d'intégrer des processus en mettant en œuvre un moteur d'orchestration (BPM ou BPI engine) ou un moteur de Workflow (WFMS - Workflow Management System). Ces moteurs qui constituent des

intergiciels perfectionnés permettant de relier les applications aux processus [Stohr et Nickerson 2001], permettent alors d'interconnecter dynamiquement les multiples applications en fonction des événements métiers. Ceci permet généralement d'obtenir une plus forte valeur ajoutée [Linthicum 2004]. Bien qu'elle soit assez intéressante, cette forme d'intégration demeure pour l'instant à l'état embryonnaire, ce qui laisse supposer qu'elle manque de maturité.

Retenons le fait que ces différentes approches d'intégration ne sont pas nécessairement exclusives, si bien qu'elles peuvent présenter des interdépendances et peuvent alors être mises en œuvre simultanément [Linthicum 2004][Chevassus 2005]. Autrement dit, on peut intégrer au sein d'un même projet d'intégration, certaines applications par les donnés, d'autres par les traitements, etc. Signalons également que, dans la suite de nos travaux, nous ne nous intéressons pas à l'intégration par des présentations qui, précisons le, peut par ailleurs constituer une perspective pertinente pour les travaux que nous menons.

II.4.3.4. Les niveaux de l'intégration

Une autre dimension de l'intégration est le niveau d'intégration. On distingue généralement plusieurs niveaux où l'on peut réaliser l'intégration [Xu et al. 2003][Sheth 1999][Bruijn 2003-b]. Dans le cadre de nos travaux, nous nous focalisons principalement sur deux niveaux d'intégration qui sont :

- l'intégration syntaxique (niveau syntaxique);
- et l'intégration sémantique (niveau sémantique).

Nous devons noter que ces niveaux d'intégration présentent une certaine similitude avec les niveaux d'interopérabilité définis dans le cadre des frameworks EIF (European Interoperability Framework)[9] [EIF 2004], AIF (Athena Interoperability Framework)[10] [Athena 2005], et ECIMF (E-Commerce Integration Mata-Framework)[11] [ECIMF 2001].

II.4.3.4.1. L'intégration syntaxique

L'intégration syntaxique permet de résoudre les hétérogénéités d'ordre syntaxique (cf. § II.2.3.3) qui peuvent exister au niveau des modèles de données ou des signatures des fonctions et des processus. Les approches principales qui peuvent être dégagées des différents travaux sur l'intégration syntaxique sont l'approche par conversion et l'approche par modèle canonique. L'approche par conversion consiste à transformer la syntaxe d'un composant dans la syntaxe d'un autre composant. L'approche par modèle canonique suppose l'existence d'un modèle pivot par lequel on doit transiter pour pouvoir faire interopérer deux composants du système d'information. Des exemples de tels modèles canoniques dans le domaine industriel sont notamment STEP, PIF et XML.

STEP (STandard for the Exchange of Product model data) [Sharon et Kemmerer 1999] est un modèle destiné à décrire des données géométriques des produits. PIF (Process Interchange Format) [Lee et al. 94] est un modèle destiné à l'échange automatique des

[9] Le cadre EIF [EIF 2004] définit trois niveaux d'interopérabilité: niveau technique, niveau sémantique et niveau organisationnel. L'interopérabilité technique couvre les aspects techniques d'interconnexion des systèmes et des services et inclut principalement les techniques d'interfaces ouvertes, d'intégration, de présentation et des échanges de données. L'interopérabilité sémantique couvre les aspects liés au sens des informations échangées. L'interopérabilité organisationnelle cocnerne les aspects liés à la définition des objectifs métier, à la modélisation des processus, et la collaboration des administrations qui désirent échanger des informations.

[10] Le cadre AIF [Athena 2005] qui est relativement similaire au cadre EIF définit, quant à lui, trois niveaux d'interopérabilité: niveau technologique, niveau conceptuel et niveau organisationnel. Le niveau conceptuel inclut le niveau syntaxique et sémantique.

[11] Le cadre d'interopérabilité d'ECIMF (E-Commerce Integration Mata-Framework) définit les niveaux suivants: le niveau contexte métier, le niveau de médiation de processus, le niveau de translation sémantique, et le niveau de mapping syntaxique [ECIMF 2001][Bialecki 2001].

modèles de procédés de workflow entre des systèmes hétérogènes. XML (eXtended Markup Language) constitue un langage de formatage standardisé des données basé sur des techniques de balisage. Certains de ces modèles sont décrits au chapitre III qui présente un panorama des principales techniques utilisées pour l'intégration syntaxique des systèmes d'information industriels.

II.4.3.4.2. L'intégration sémantique

L'intégration sémantique a pour but de résoudre les hétérogénéités sémantiques qui peuvent exister entre les applications. Dans le paragraphe § II.2.3.3, nous avons introduit les principaux conflits sémantiques de données qui sont notamment les conflits de confusion, les conflits de mesure et les conflits de nommage. Les conflits sémantiques ne concernent pas seulement les données, mais peuvent aussi concerner les autres couches de l'intégration dont notamment les fonctions et les processus.

Parmi les solutions utilisées pour l'intégration sémantique, on peut citer l'utilisation des approches basées sur les ontologies qui permettent de définir une couche d'abstraction où tous les concepts d'un domaine y sont définis et où des mécanismes d'inférence peuvent être mis en œuvre en vue de résoudre les différences d'ordre sémantique. Le chapitre VI de cette présente partie présente les principales techniques utilisées pour l'intégration sémantique des systèmes d'information industriels.

Notons, que dans le cadre de nos travaux, on se focalise principalement sur l'intégration sémantique et à moindre mesure sur l'intégration syntaxique des systèmes d'information d'entreprise. On s'intéresse plus précisément aux scénarii d'intégration intra-entreprise[12], du point de vue conceptuel et concernant principalement la couche applicative (données et traitements) et la couche processus. Dans les sections suivantes, nous allons brièvement introduire les principes méthodologiques majeurs utilisés dans le cadre des projets d'intégration intra-entreprise.

II.5. Méthodologies pour l'intégration

Les méthodologies d'intégration actuelles des applications d'entreprise reposent fondamentalement sur les étapes de mise en œuvre d'un certain nombre de techniques d'intégration. Des exemples de techniques peuvent être la mise en place des multiples intergiciels (CORBA, serveurs d'applications, MOM, EAI, ESB, etc.) qui peuvent être mis en œuvre dans des projets intra-entreprise.

Comme tout projet informatique, les projets d'intégration[13] peuvent suivre l'un des cycles standards (cycle en cascade, en V, RUP, etc.) utilisés traditionnellement dans le cadre de développement de systèmes d'information [Leroux 2004] avec les étapes suivantes : étape préalable, spécification, réalisation et mise en œuvre. Dans ce qui suit, nous allons présenter une démarche générale d'intégration qui s'inspire des scénarios pratiques de mise en œuvre de solutions EAI au sein des entreprises [Rivard et Plantain 2003].

[12] Il est fondamental de préciser que la frontière entre les scénarii intra et inter-entreprise est relativement floue. C'est l'exemple typique d'une grande entreprise multinationale avec plusieurs filiales et dont les scénarios d'interopérabilité inter-filiales peuvent être considérés à la fois comme des scénarios intra-entreprise et inter-entreprise.

[13] Rappelons qu'un projet d'intégration d'applications est avant tout un projet informatique, et à ce titre ses délivrables (ou livrables) sont multiples et incluent l'étude de faisabilité, les cahiers de charges, le logiciel (socle d'intégration), la recette, etc.

II.5.1. Démarche globale d'intégration

Un projet d'intégration est généralement un projet complexe et d'envergure dans la mesure où il est rarement ponctuel, et de plus il peut concerner et impacter l'ensemble du système d'information. Pour cette raison, il doit s'inscrire dans une optique de projet global d'infrastructure du système d'information guidée par un plan d'urbanisation. Telle que présentée, cette démarche est descendante (top-down) et globalisante dans la mesure où elle touche à la globalité de l'entreprise, et on parle généralement de projets stratégiques ou d'infrastructure. Toutefois, il demeure possible d'adopter une démarche ascendante (bottom-up) conduisant ainsi à des projets tactiques ou départementaux. Hormis la portée du projet, l'objectif demeure le même: il s'agit de raccorder tous les systèmes d'information (du département ou de l'entreprise, selon le cas) aux infrastructures d'échange de l'information existantes et/ou qui vont exister au sein du département ou de l'entreprise.

Quelle que soit l'approche retenue, intégrer les applications nécessite souvent une analyse approfondie des applications existantes et de leurs interdépendances : c'est la démarche d'urbanisation. L'urbanisation du système d'information, dont la paternité revient à [Sassoon 1998], se base sur la métaphore de la cité en procédant à un découpage orienté métiers de l'entreprise. Ce découpage permet de définir des zones, qui sont ensuite découpées en quartiers, chaque quartier étant lui-même découpé en blocs. La notion de bloc constitue la brique de base de l'urbanisation. Il est considéré comme une partie relativement autonome, asynchrone, propriétaire de ses données qu'elle encapsule, et de plus traite généralement un événement de bout en bout sans se référer à un autre bloc. Chaque bloc peut éventuellement entretenir des échanges avec d'autres blocs en transitant par l'infrastructure de communication ou d'échanges [Longépé 2003]. Le processus d'urbanisation aboutit généralement à la décomposition du système d'information en sous-systèmes modulaires où deux types principaux d'infrastructures existent (figure II.10) :

- des infrastructures d'intégration locales permettant de relier les différents éléments à l'intérieur d'un sous-système;
- et des infrastructures globales utilisées pour relier les sous-systèmes entre eux.

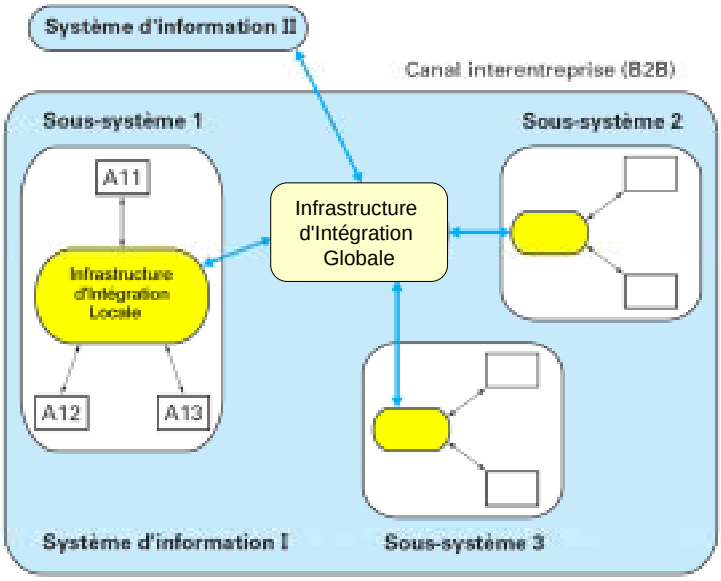

Figure II.10. Principe général de l'urbanisation du système d'information [Chevassus 2005]

II.5.2. Etapes du processus d'intégration

Les étapes du processus d'intégration d'applications peuvent sensiblement différer selon que la démarche d'intégration est ascendante (projet tactique ou départemental) ou descendante (projet stratégique ou global). Mais pour simplifier, on peut supposer que la différence principale se situe autour des deux premières étapes, qui correspondent à l'étape de stratégie et d'urbanisation du système d'information, qui est généralement absente ou à la limite se fait d'une manière peu importante dans le cas d'une démarche ascendante. Les principales étapes du processus d'intégration peuvent donc se résumer comme suit (figure II.11) [Rivard et Plantain 2003]:

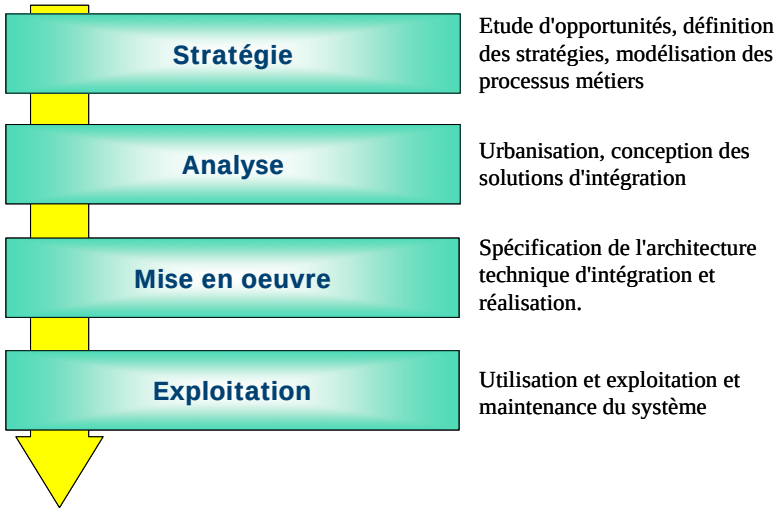

Figure II.11. Étapes du processus d'intégration [Rivard et Plantain 2003]

- Etape 1 – *Stratégie*: Cette étape consiste à déterminer le modèle métier le plus approprié aux besoins et à la volonté du client, à détecter les opportunités, les critères différentiateurs, et les stratégies de repli. Généralement effectuée par des consultants en métier (stratégie et marketing), cette étape nécessite le plus souvent l'intervention de spécialistes en stratégie et en e-Business.
- Etape 2 - *Analyse*: Cette étape comprend une analyse métier et applicative du domaine. L'analyse métier permet de fournir la cartographie fonctionnelle, tandis que l'analyse applicative permet de produire la cartographie applicative du domaine. Ces deux analyses peuvent être menées dans le cadre du plan d'urbanisation. Ce plan, qui peut servir de démarche structurante est d'une importance capitale pour les projets d'intégration. Il est à l'intégration ce que le schéma directeur (ou plan d'informatisation) était au développement de systèmes d'information. Le plan d'urbanisation permet de structurer le système d'information en le découpant selon les métiers de l'entreprise.
- Etape 3 - *Mise en place d'une solution d'intégration*: qui permet le raccordement progressif des applications du domaine d'intégration. Il est, bien entendu, possible que des projets de développement notamment de connecteurs puissent émerger lors de cette étape. On peut également signaler que sur le terrain, la tendance actuelle en matière d'intégration se penche beaucoup plus vers la sous-traitance et/ou à l'acquisition de solutions commerciales de type EAI (Enterprise Application Integration) par exemple. Il faut tout de même souligner l'intérêt d'une démarche qui privilégie une maîtrise interne du processus d'intégration et qui permettrait à

terme de s'affranchir des solutions souvent propriétaires des éditeurs et également de la politique de "tout-intégré", en adoptant éventuellement des politiques basées sur des standards et/ou sur la politique du "best of breed" qui sont souvent plus flexibles, plus robustes, et plus pérennes [Rivard et Plantain 2003].

- Etape 4 – *Exploitation* : Une fois le système déployé, des équipes vont alors l'exploiter au quotidien. Les équipes peuvent être techniques ou métier. Les équipes techniques vont s'occuper de l'administration et de l'exploitation de l'infrastructure d'intégration, alors que les équipes métier (managers) vont s'intéresser à l'exploitation de tableaux de bord fournis par la plate forme.

II.6. Conclusion

Ce chapitre a présenté la problématique de l'intégration et de l'interopérabilité des systèmes d'information industriels en général et en particulier celle des systèmes d'information microélectroniques où les défis majeurs consistent à définir des solutions d'intégration et/ou d'interopérabilité flexible permettant ainsi d'apporter le maximum d'agilité aux entreprises.

Ce chapitre a permis notamment de présenter la notion de système d'information et d'application ainsi que leurs principales caractéristiques. Il a ensuite permis d'exposer le concept d'intégration et celui de l'interopérabilité en mettant en évidence la différence qui existe entre ces deux concepts. Il a également exposé les multiples dimensions que peut présenter le concept d'intégration. Enfin, il a présenté un cadre méthodologique générique très souvent utilisé par les entreprises pour définir leurs solutions d'intégration de leurs systèmes d'information.

Dans les deux chapitres suivants, nous allons approfondir les principales techniques qui peuvent être mises en oeuvre pour la réalisation des projets d'intégration. Le chapitre III va plus particulièrement s'intéresser aux techniques d'intégration syntaxique, tandis que le chapitre IV va concerner les techniques d'intégration sémantique.

Chapitre III

INTEGRATION SYNTAXIQUE DES SYSTEMES D'INFORMATION INDUSTRIELS

III.1. Introduction

Comme nous l'avons souligné dans le chapitre précédent, les applications industrielles, et en particulier les applications microélectroniques, doivent impérativement opérer ensemble afin d'assurer l'atteinte de l'objectif global. Nous avons évoqué l'existence de deux principaux niveaux d'intégration qui sont le niveau syntaxique et le niveau sémantique. Nous allons, dans ce chapitre nous intéresser au niveau syntaxique, et nous allons présenter les principales techniques utilisées pour intégrer syntaxiquement des applications industrielles. En particulier, nous allons étudier les techniques basées sur les formats standardisés, les intergiciels de communication et de distribution (middlewares), les outils EAI (Enterprise Application Integration), les moteurs BPM (Business Process Management), les architectures orientées services (SOA – Service-Oriented Architecture) et les techniques d'ingénierie à base de modèles.

Ce chapitre présente donc un panorama des principales techniques d'intégration syntaxique pouvant être mises en œuvre au sein des entreprises industrielles et particulièrement dans les entreprises du secteur de la microélectronique. Après avoir présenté une typologie des principales techniques d'intégration syntaxique, nous allons brièvement décrire chacune d'elles et nous terminerons le chapitre par une synthèse des principales caractéristiques de chacune des techniques étudiées.

III.2. Typologie des techniques d'intégration syntaxique

Un des problèmes majeurs de l'intégration dans le domaine industriel est la capacité à interpréter et comprendre les données reçues. Une des solutions utilisées consiste à mettre en œuvre des technologies et techniques permettant d'adresser l'intégration syntaxique. Ces technologies et techniques sont aussi diverses que variées. Cependant, nous proposons de les classifier en sept grandes catégories qui sont :

- les techniques ad hoc basées sur la conversion de syntaxes;
- les techniques basées sur la standardisation des modèles et des formats d'échange;
- les techniques basées sur l'utilisation des intergiciels (middlewares);
- les techniques basées sur les outils EAI (Enterprise Application Integration);
- Les techniques basées sur la gestion des processus (BPM - Business Process Management)
- les techniques basées sur les architectures orientées services (SOA - Service-Oriented Architectures);
- et les techniques basées sur l'ingénierie des modèles (MDA - Model-Driven Architecture).

Nous avons choisi de résumer en figure III.1 les principales techniques d'intégration syntaxique en utilisant un méta-modèle UML. Dans les sections suivantes, nous allons détailler chacune de ces techniques.

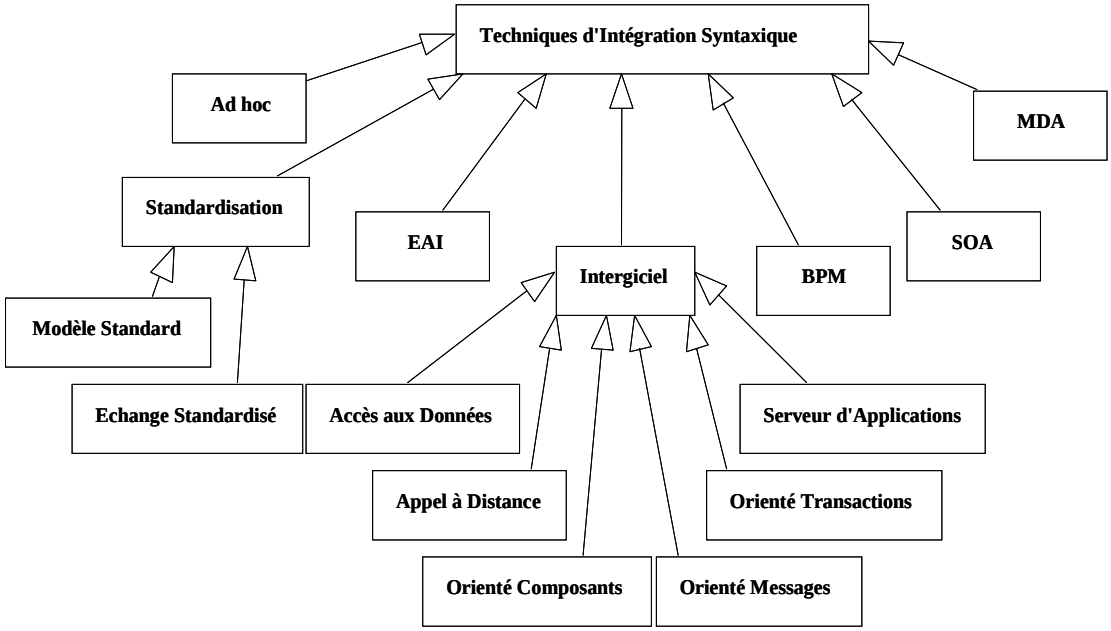

Figure III.1. Typologie des principales techniques d'intégration syntaxique[14]

Signalons qu'il existe de nombreux autres travaux plus ou moins proches du domaine d'intégration syntaxique et de la flexibilité des systèmes d'information. Il s'agit en particulier des travaux qui traitent de l'intégration de données [Schneider 2002][Dang Ngoc 2003] [Baril 2003][Devogele 1997], des travaux sur la composition et de la fédération de logiciels [Tuyet 2004][Villalobos 2003][Verjus 2001][Amiour 1999][Boyer 1994], des travaux portant sur la construction d'intergiciels spécifiques [Quinot 2003][Huet 2002][Denis 2003] et aussi des travaux de modélisation d'entreprise et de son système d'information qui peut être mise en œuvre dans le cadre d'intégration syntaxique ou de recherche de flexibilité [Blanc 2004] [Mathieu 2004]. Vu l'ampleur de l'état de l'art, nous avons préféré nous limiter aux approches évoquées en figure III.1.

[14] Signalons que nous nous basons tout au long de ce chapitre et des chapitres suivants sur le formalisme UML pour représenter nos différents modèles et méta-modèles. Aussi, le sens des symboles graphiques utilisés dans ces modèles est celui préconisé dans UML 2.0. En particulier, les flèches désignent la relation de spécialisation/généralisation entre classes, les arcs possédant un losange à l'extrémité désignent des relations d'agrégation entre classes, et les arcs simples désignent les autres types de relations entre classes.

III.3. Techniques ad hoc de conversion

Les techniques ad hoc de conversion se basent fondamentalement sur l'utilisation de techniques câblées (codage en dur) pour implémenter les scénarios d'intégration entre applications. Pour cela, elles se basent généralement sur certains intergiciels simples invoqués de façon programmatique pour définir des passerelles entre les applications. Dans ce type de techniques, les applications sont très souvent reliées les unes aux autres en n'utilisant pas de structure évoluée d'intégration intermédiaire.

La complexité en matière d'interfaces (figure III.2) est de l'ordre de $o(n^2)$ avec n étant le nombre d'applications. Parmi les intergiciels les plus traditionnellement utilisées, on peut citer le RPC (Remote Procedure Call) (cf. § III.5.4) qui consiste à mettre en oeuvre les appels distants entre les procédures des applications industrielles.

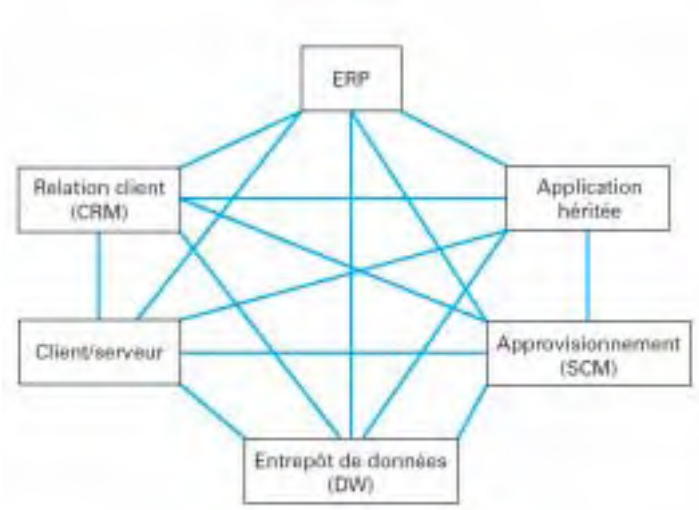

Figure III.2. Principe des techniques ad hoc de conversion (adapté de [Chevassus 2005])

III.4. Techniques de standardisation des représentations

Une des manières les plus intuitives de réaliser l'intégration entre applications informatiques et aussi entre les utilisateurs est de définir un langage commun. Dans l'industrie, on utilise une multitude de langages qui permettent de représenter les modèles de données, les modèles de processus ou encore les échanges de données. Cette section a pour objet de présenter brièvement les principaux standards utilisés dans l'industrie et en particulier dans l'industrie microélectronique.

III.4.1. Modèles standardisés

Les modèles utilisés dans le cadre de la formalisation dans l'industrie microélectronique sont principalement les modèles de données et les modèles de processus. Les modèles de données permettent de donner une représentation (très souvent graphique) de la réalité à travers principalement l'utilisation des objets ou des entités, des attributs et des relations. Les modèles de processus permettent de décrire les différents éléments d'un processus métier à travers principalement la notion d'activité, d'événement et de rôle. Dans ce qui suit, nous allons présenter quelques modèles standardisés des données et

des processus très souvent utilisés en industrie. De plus, nous allons présenter brièvement la norme industrielle CEN/ISO 19440 qui permet de définir les constructions nécessaires dans le cadre de la modélisation et l'intégration d'entreprise.

III.4.1.1. Modélisation standardisée des données

Il existe plusieurs modèles pour la description des données. On peut principalement citer dans le cadre industriel UML, STEP et XSD. UML (Unified Modeling Language) est un standard utilisé pour la modélisation des données. STEP (Standard for the Exchange of Product Model Data) [Sharon et Kemmerer 1999] constitue une norme en matière de données techniques utilisées lors des échanges impliquant des outils de conception. XSD (XML Schema Definition) constitue une norme en matière de modélisation des flux de données quelconques. Dans ce qui suit, nous allons succinctement présenter les deux modèles les plus représentatifs et les plus génériques qui sont UML et XSD.

III.4.1.1.1. UML

UML [Rumbaugh et al. 2005] qui est un standard de l'OMG, est à l'origine un langage de modélisation de systèmes d'information à base d'objets distribués, visant à unifier les concepts majeurs introduits par les méthodologies Booch [Booch 1992], OMT (Object Modeling Technique) [Rumbaugh et al. 1991] et OOSE [Jacobson 1993]. Il permet de modéliser les multiples aspects d'un système. En industrie, UML est généralement utilisé pour décrire les schémas de données des bases de données.

La figure suivante est un extrait du méta-modèle UML. On y distingue quatre concepts de base qui sont utilisés lors de la modélisation de données et qui sont : la classe, la relation, l'objet, le lien. Les deux premiers éléments permettent de définir le diagramme de classes, tandis que les deux derniers sont utilisés pour définir le diagramme d'objets.

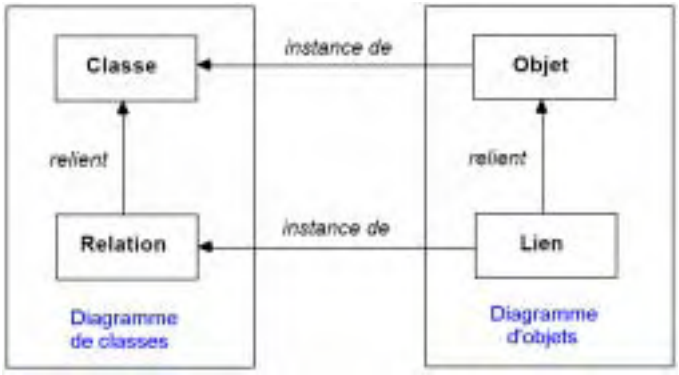

Figure III.3. Extrait du méta-modèle UML [Espinasse 2002]

III.4.1.1.2. XSD

XSD (XML Schema Definition) [W3C-XSD 2005] constitue un dialecte XML qui permet de définir de façon très précise la structure et le contenu d'un document XML (cf. § III.4.2.1). XSD, qui a été introduit en remplacement de la traditionnelle DTD[15] qui

[15] La DTD (Document Type Definition) est un modèle décrivant la structure d'un fichier XML et permettant de passer d'un document bien formé (respectant la syntaxe du langage) à un document valide (respectant également la structure d'une DTD). Les DTD peuvent être directement intégrées au sein des documents dont elles décrivent la structure (DTD internes) ou simplement référencées comme entité externe.

est devenue obsolète, fournit un inventaire exhaustif des balises XML qui peuvent être utilisées lors de la définition de classes de documents XML. Cette définition se fait via des composants de modélisation qui permettent de définir les contraintes et documenter la signification, l'utilisation et les relations de dépendances des constituants d'un document XML : les types de données, les éléments et leurs contenus, les attributs et leurs valeurs. Les schémas permettent également de spécifier des informations complémentaires au document, comme par exemple la normalisation et la gestion des valeurs par défaut des attributs et des éléments. Les schémas offrent des facilités pour être auto-documentés.

La figure III.4 permet de représenter un extrait du méta-modèle XSD. Tout document est défini à l'aide d'un ou plusieurs schémas XML. Chaque schéma est un ensemble structuré d'un certain nombre de composants qui peuvent être de types simples ou de types complexes. Des exemples de types simples peuvent être les types string, date, etc., tandis que les types complexes sont des types composés sur la base de types simples ou d'autres types composés en utilisant des constructeurs comme la séquence, le groupe ou le choix.

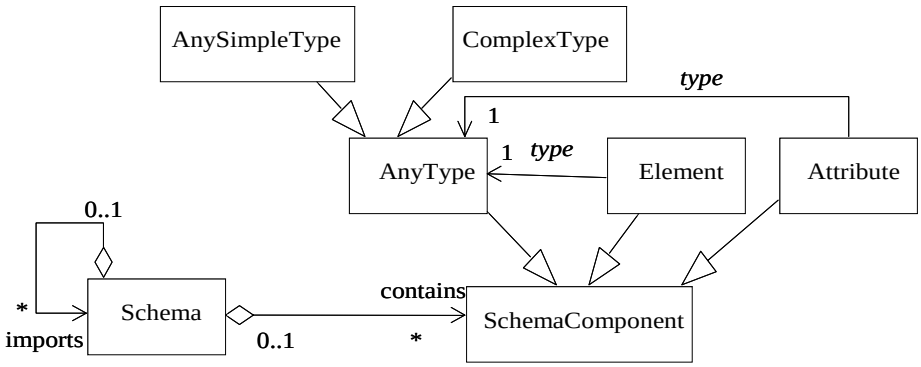

Figure III.4. Extrait du méta-modèle XSD

III.4.1.2. Modélisation standardisée des processus

De même que pour les données, il existe plusieurs modèles pour la description des processus. On peut principalement citer dans le cadre industriel le modèle IDEF3 et BPML (d'autres modèles seront introduits en § III.8).

III.4.1.2.1. IDEF3

IDEF3 [Maye 1992] a été proposé en 1992 pour remédier aux limites d'IDEF0 (Ross 1977) en matière de modélisation du comportement de l'entreprise. Il permet de représenter graphiquement les processus sous forme d'un enchaînement d'étapes, appelées unités de comportement. Ces dernières sont connectées entre elles par des boîtes de jonction (asynchrones ou synchrones). Les liens peuvent être de précédence, relationnels ou de flux d'objets. Une unité de comportement est une activité du processus, elle est représentée par une boîte rectangulaire et elle est le résultat de la décomposition modulaire hiérarchique et descendante des activités du système étudié. Les boîtes de jonction sont des connecteurs logiques entre les unités de comportement servant à modéliser les processus. Elles peuvent être convergentes ou divergentes pour représenter deux flux qui se rencontrent ou qui se séparent. Elles peuvent être utilisées en mode synchrone ou asynchrone selon que les flux démarrent ou finissent au même

moment ou non. Il existe trois types de boîtes de jonction dans IDEF3 : les connecteurs AND, OR et XOR.

La Figure II.5 présente le méta-modèle de IDEF3 [Maye 1995]. Comme on peut le constater, IDEF3 permet d'exprimer aisément les relations causales et de précédence rencontrées dans l'expression des processus d'entreprise. La description d'un processus s'articule autour du constructeur d'unité de comportement (UOB - Unit Of Behavior) qui définit une étape quelconque d'un processus. Pour former des scénarii, les UOB sont connectées par l'intermédiaire de liens et de connecteurs logiques (Junction Process).

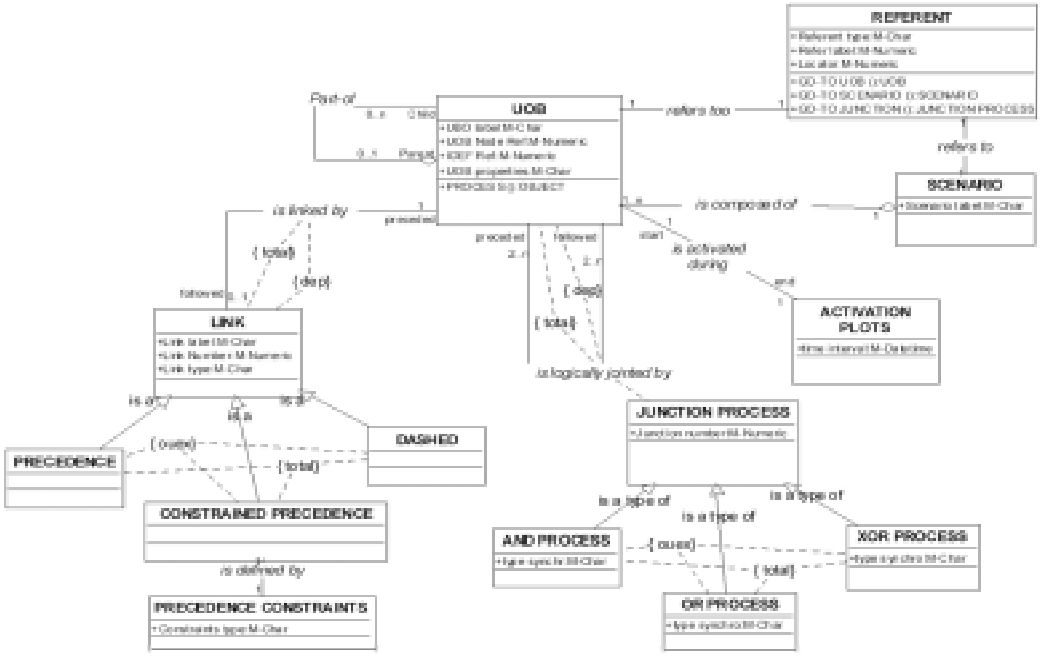

Figure III.5. Méta-modèle IDEF3

III.4.1.2.2. BPML

BPML (Business Process Modeling Language) [BPMI 2003] est un langage de modélisation des processus métiers dont les premières spécifications sont apparues au printemps 2001 au sein du consortium BMPI. Il permet de définir un modèle abstrait d'interaction entre collaborateurs participant à une activité de l'entreprise, voire entre une organisation et ses partenaires. Les processus métiers sont représentés par un flux de données, un flux d'événements sur lesquels on peut influer en définissant des règles métier, des règles de sécurité, des règles de transactions. Cette norme est à la base de la définition de BPEL4WS qui constitue un standard lié aux Services Web et qui est présenté dans le paragraphe § III.8.4.4.

Un extrait du méta-modèle BPML est introduit en figure III.6. Comme on peut le constater sur le schéma, le principal constructeur du méta-modèle est l'activité. On peut distinguer des activités simples et complexes. Cette dernière catégorie permet de définir les différents types de processus : processus complexes, processus d'exception, et processus de compensation (un processus de compensation permet une fois exécuté d'annuler l'effet des activités du processus métier qu'on désire défaire).

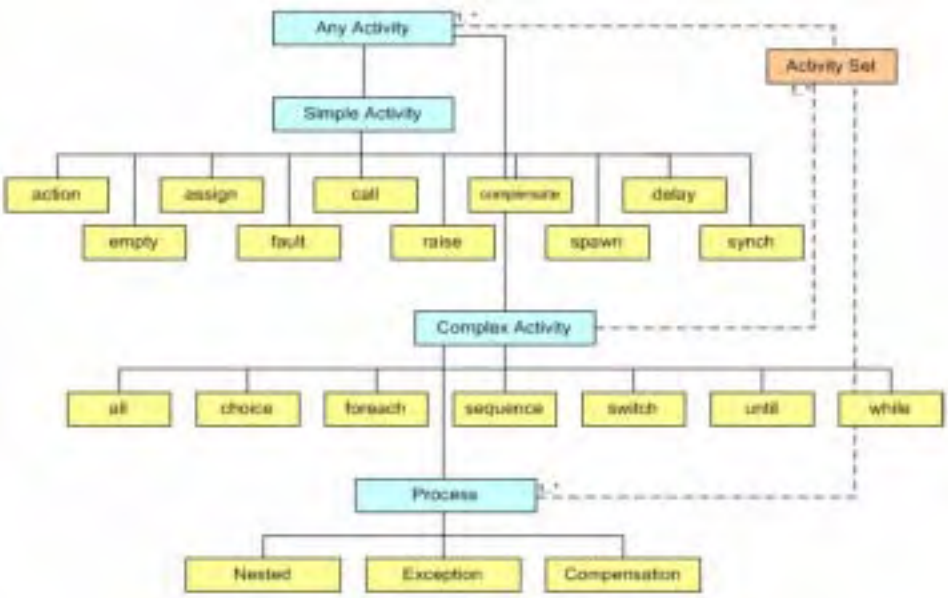

Figure III.6. Extrait du méta-modèle BPML

III.4.1.3. CEN/ISO DIS 19440

CEN/ISO DIS 19440 [CEN 2006] constitue une norme en matière de définition des constructions pour la modélisation d'entreprise. Elle est basée sur la norme CEN ENV 12204 et elle permet d'articuler les constructions liées à l'automation. De plus, elle permet d'élaborer la base de la modélisation CIMOSA. Elle repose sur la notion de vue et on distingue quatre vues principales qui sont : la vue fonctionnelle, la vue informationnelle, la vue de ressources et la vue organisationnelle. A chaque vue correspond un certain nombre de constructions dont les principales sont: domaine, processus métier, activité d'entreprise, événement, ressource, entité fonctionnelle, capacité, objet d'entreprise, vue d'objet, produit, ordre, unité organisationnelle, rôle organisationnel et centre de décision.

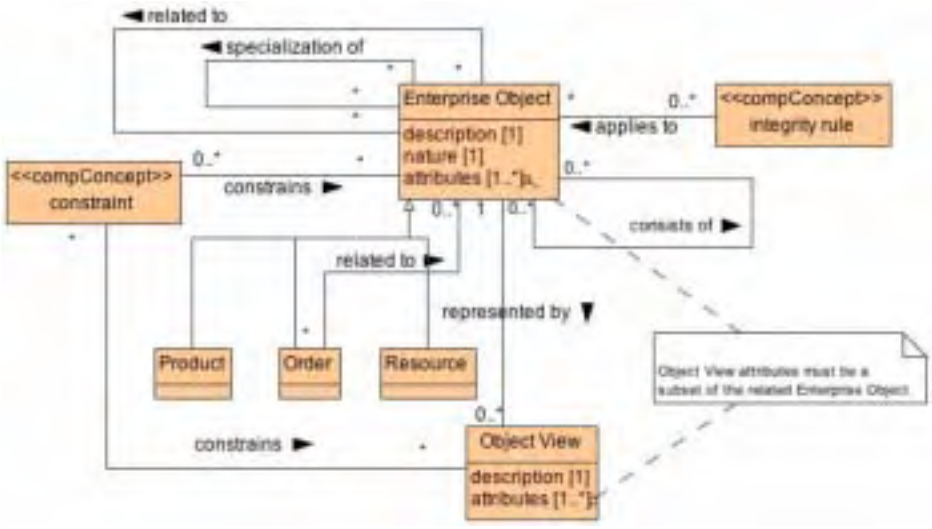

Figure III.7. Un extrait du méta-modèle CEN/ISO DIS 19440

Un extrait du méta-modèle CEN/ISO DIS 19440 nécessaire à la modélisation de la vue informationnelle est introduit en figure III.7. Comme on peut le constater sur le schéma,

le principal constructeur du méta-modèle est la vue d'objet (Object View). Ce dernier constitue une représentation des objets d'entreprise. Les objets d'entreprise (Enterprise Object) permettent de capturer toute l'information pertinente nécessaire pour la planification stratégique, le support de décision et la supervision et le contrôle des processus opérationnels.

III.4.2. Echanges standardisés

Dans le cadre des échanges informatisés dans les milieux industriels, on peut distinguer principalement l'utilisation du langage XML pour des échanges intra- et inter-entreprises et l'utilisation des langages dérivés de XML pour des échanges inter-entreprises. Parmi les langages dérivés utilisés dans des scénarios B2B, on peut citer deux formats majeurs qui sont ebXML et RosettaNet.

III.4.2.1. XML

XML (eXtensible Markup Language) a été développé, à l'origine, pour combler les lacunes du langage HTML, initialement conçu pour décrire les pages Web, et qui s'est très vite avéré limité dans la mesure où il ne distinguait pas les informations de présentation et de contenu d'un document. Or, la multiplication des canaux et des formats poussait à l'indépendance du contenu et de la présentation. D'autre part, l'accroissement des échanges B2B et les besoins d'intégration convergeaient vers une structuration plus explicite des documents et vers l'importance accrue de la sémantique. XML, dérivé du langage normalisé SGML (Standard Generalized Markup Language - ISO 8879), a été conçu comme un langage de structuration de données permettant de distinguer les informations de présentation et de structure d'un document, est basé sur l'utilisation de balises définissant un format universel de représentation des données.

De par son ouverture et sa flexibilité, XML est un langage aux possibilités d'évolution, d'expression et d'adaptation exceptionnelles. La figure III.8 donne un extrait d'un exemple de document (biblio.xml) qui permet de décrire une bibliothèque.

```
<?xml version='1.0' encoding='ISO-8859-1'
standalone='no'?>
<!DOCTYPE bibliothèque SYSTEM 'biblio.dtd'>
<?xml-stylesheet type='text/xsl' href='biblio.xsl'?>

<bibliothèque>
      <livre lang='fr' ref='ASI2002'>
            <achat date='10/11/2003' lieu='FNAC'/>
            <titre type="roman' genre='SF'> Prélude à
            fondation </titre>
            <auteurs>
                  <auteur>
                        <nom> Asimov </nom>
                        <prénom> Isaac </prénom>
                  </auteur>
            </auteurs>
            <éditeur> Pocket </éditeur>
      </livre>
```

Figure III.8. Exemple d'un document XML [Accary-Barrier et Calabretto 2002]

III.4.2.2. ebXML

ebXML (e-Business XML) [ebXML 2001] est un standard proposé en 1999 à la fois par l'OASIS et par l'UN/EDIFACT pour les échanges dans le cadre du commerce électronique. ebXML se focalise sur les processus métier et la gestion des transactions. Les processus métiers, au sens ebXML, s'entendent comme l'enchaînement de tâches donnant lieu à l'échange de messages conventionnels (documents commerciaux) entre les protagonistes. Un point fort de ebXML est sa généralité, ce qui suppose qu'il peut être utilisé dans différents secteurs industriels.
Les principales spécifications ebXML sont (figure III.9) [Chauvet 2002] :

- Core Component : qui définit les structures de données réutilisables de bas niveau (ex : party, address, phone, date, devise…) qui peuvent être instanciées dans plusieurs types d'échanges d'informations utilisés pour définir les processus métier et les modèles d'information. Ce composant permet de faciliter l'intégration entre différents systèmes;
- Business Process : qui permet de décrire les interfaces du processus métier (ex: processus "Acheter un produit");
- Trading Partner Profile : qui permet de définir deux types de contrats le CPP et le CPA. Le CPA (Collaboration Protocol Profile) permet de décrire les capacités d'affaire d'une compagnie dans un format standard et portable (Profile général, BP, protocoles supportés…). Le CPA (Collaboration Protocol Agreement) permet de décrit clairement les éléments et les mécanismes des transactions conduits entre deux compagnies (Contrat);
- ebXML Registry & Repository : permet de définir les mécanismes de sauvegarde et de recherche dans les registres et référentiels ebXML;
- Transport & Routing : permet de définir une méthode pour échanger des messages électroniques.

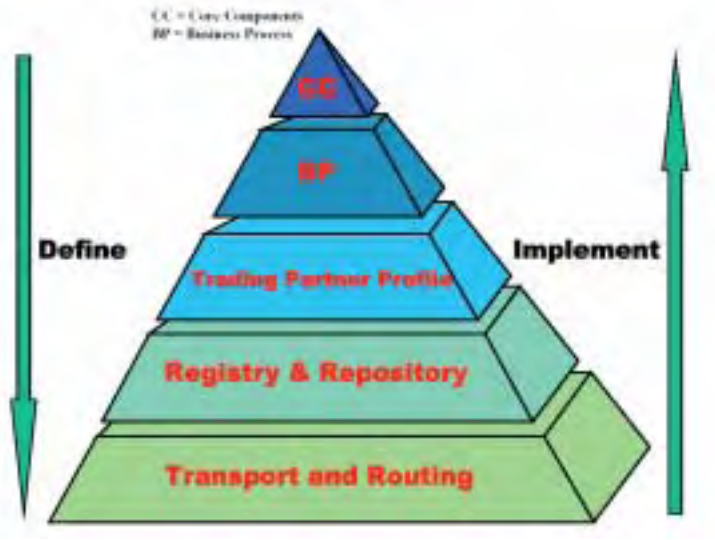

Figure III.9. Architecture des spécifications ebXML

III.4.2.3. RosettaNet

Créé en 1998, RosettaNet [RosettaNet 2002] est un standard similaire à ebXML proposé par le consortium RosettaNet, consortium indépendant d'industriels des secteurs

informatique-électronique et semi-conducteurs. Il a pour but de formaliser, en XML, le dialogue entre les partenaires d'une transaction commerciale, afin de définir les éléments nécessaires à l'automatisation des Web Services ayant trait à la chaîne logistique.

L'architecture de RosettaNet repose sur quatre notions essentielles (figure III.10) :

- des templates, ou modèles de processus métiers;
- les PIPs (Partner Interface Processes) qui permettent de définir les dialogues;
- une architecture d'échange permettant de définir des échanges entre partenaires;
- des dictionnaires pour harmoniser les vocabulaires XML.

Grâce à la conjugaison de ces quatre notions, il est possible d'échanger des messages RosettaNet, en respectant un enchaînement défini par le PIP, et avec une sémantique compréhensible par les parties, grâce aux dictionnaires. Les PIPs permettent de gérer les transactions entre partenaires de l'échange. Ils représentent des enchaînements de tâches que les entreprises ont rendu publiques, pour gérer leur échange. A charge pour elles de conserver leurs propres processus internes de traitement des informations entrantes et sortantes.

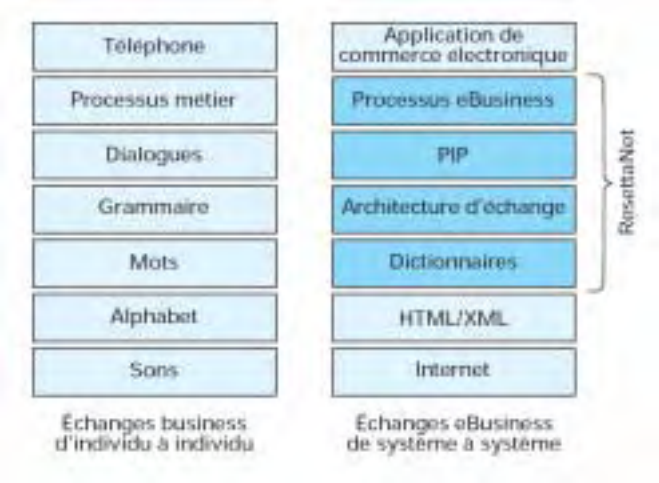

Figure III.10. Architecture RosettaNet [Pontacq 2002]

III.5. Intergiciels

De même que pour les formats standardisés qui sont très souvent utilisés dans l'industrie, les intergiciels, qui sont des logiciels principalement dédiés à la gestion des communications inter-applications, sont également très présents dans les milieux industriels.

III.5.1. Définition

Traditionnellement, le terme d'intergiciel (en anglais middleware**)** désigne l'ensemble des technologies permettant de faire communiquer des composants informatiques en fournissant des services de transport et de routage [Linthicum 2004][IDEAS 2004]. La figure suivante (figure III.11) permet d'illustrer le principe des intergiciels qui permettent ainsi de masquer toute l'hétérogénéité technique liée aux plates-formes (langages de programmation, systèmes d'exploitation et réseaux).

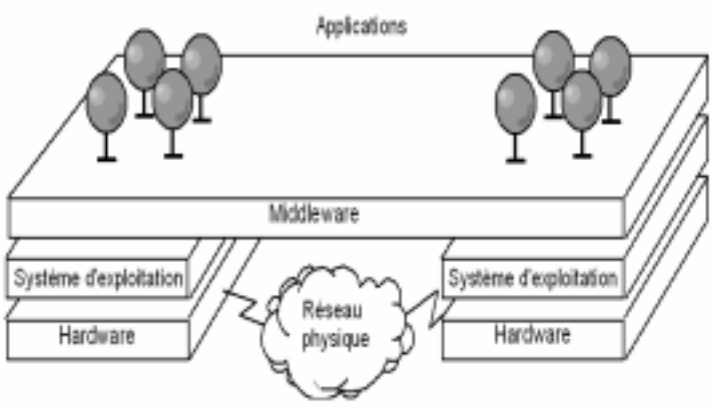

Figure III.11. Principe de l'intergiciel

Par rapport au modèle OSI (Open System Interconnexion) [ISO-OSI], l'intergiciel concerne généralement les trois couches supérieures du modèle OSI, c-à-d: il concerne les couches 5 à 7 (session, présentation et application) et s'appuie sur les couches de communication de bas niveau (transport, réseau, liaison et physique) dont il masque la technicité (figure III.12). Il s'agit d'une couche logicielle jouant le rôle d'intermédiaire entre une application (le client) et une autre (le serveur) [Serain 2001][Leroux 2005].

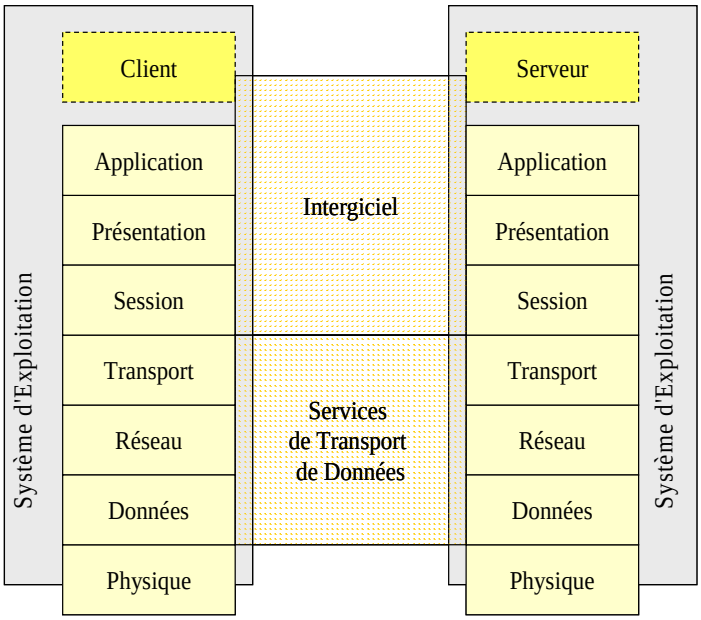

Figure III.12. Positionnement de l'intergiciel par rapport au modèle OSI (adapté de [Serain 2003])

Nous devons souligner le fait que la notion d'intergiciel a suffisamment évolué, ces dernières années, au point d'inclure d'autres services, dont certains de haut niveau, tels que la transformation des données, la gestion des transactions, la gestion des processus, la sécurité, le management des échanges (administration et exploitation), la qualité de services (performances, disponibilité, etc.) [Alonso 2002][Linthicum 2004]. Toutefois, dans le cadre de ce document, on réserve le terme intergiciel principalement aux outils qui se chargent de la communication, du routage des messages entre les applications et de la gestion des transactions, et nous préférons considérer les moteurs de workflow et les outils EAI comme des systèmes sophistiqués qui incluent les intergiciels de base présentés dans cette section.

III.5.2. Typologie des principaux intergiciels

Bien que les intergiciels qui peuvent être mis en œuvre dans les milieux industriels soient fort nombreux, nous pouvons toutefois les regrouper principalement autour de sept grandes familles qui sont (figure III.1) [Hohpe et Woolf 2003][Linthicum 2004][Linthicum 1999]:

- les accès aux bases de données;
- les appels de procédures à distance;
- les intergiciels orientés messages;
- les intergiciels orientés composants;
- les intergiciels orientés transactions;
- et les serveurs d'applications.

III.5.3. Intergiciels d'accès aux bases de données

Les intergiciels d'accès aux données permettent aux applications d'accéder de manière transparente et simultanée à des données, et ce quels que soient leur mode d'organisation et leur type (DB2, Oracle, Sybase). Il existe principalement deux types de middlewares de base spécialisés et normalisés pour l'accès aux données, ODBC (Open DataBase Connectivity) et JDBC (Java DataBase Connectivity) et qui jouent un rôle très important dans le cadre de l'intégration des applications dans le domaine industriel. Mais au dessus de ces deux standards, on a construit d'autres middlewares plus perfectionnés et dont on distingue principalement les middlewares de réplication et celui de fédération de données.

La technique généralement utilisée par les middleware d'accès aux données consiste à définir une interface logique utilisée par les applications pour accéder aux sources de données et ce en :

- convertissant le langage de l'application source vers un langage compréhensible par la base de données cible (généralement le langage SQL de cette base de données);
- puis en envoyant cette requête à la base de données cible;
- et ensuite en exécutant cette requête sur cette base de données cible;
- et enfin en récupérant les réponses à travers le réseau et en les convertissant en un format compréhensible par l'application source.

La norme ODBC [Microsoft 1992] est un standard qui a été développé par Microsoft dans les années 1990 permettant la communication entre des clients bases de données fonctionnant sous Windows et les SGBD (Système de Gestion de Bases de Données) du marché. ODBC fournit une API indépendante de la base de données cible. Avec ce standard, le client peut adresser sa requête dans un formalisme SQL classique sans se préoccuper des spécificités du SGBD cible. En fonction du type de base de données à accéder, la partie logicielle ODBC charge un pilote (driver) permettant la traduction au format propriétaire du SGBD cible (figure III.13). Bien que ODBC permette un interfaçage avec des bases de données indépendamment du SGBD, cette technologie demeure une solution propriétaire de Microsoft. De par son ouverture, l'API JDBC que nous allons présenter ci-dessous comble cette lacune.

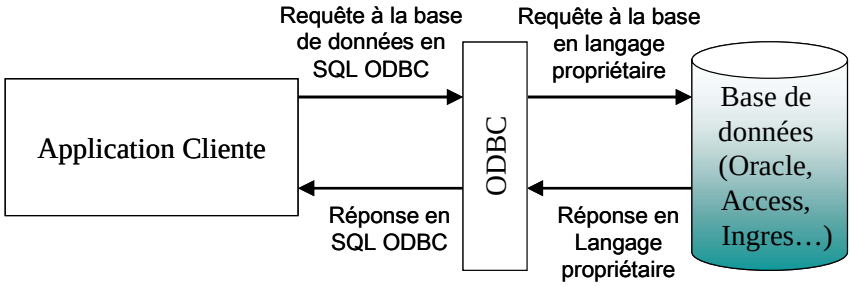

Figure III.13. Principe de l'ODBC [Toublant et al. 2001]

La norme JDBC [Sun-JDBC 2005] est développée par JavaSoft et elle est fonctionnellement équivalente à ODBC mais est adaptée par contre aux environnements Java. L'API standard JDBC définit un package (ensemble de classes Java) qui permettent à un composant java quelconque (applet, servlet, Java Bean ou toute application Java) de se connecter à une base de données cible. Notons qu'il existe deux façons majeures pour utiliser le JDBC pour accéder à une base de données cible (figure III.14): soit à travers un JDBC driver s'il est disponible, soit à travers un ODBC bridge (pont vers le driver ODBC).

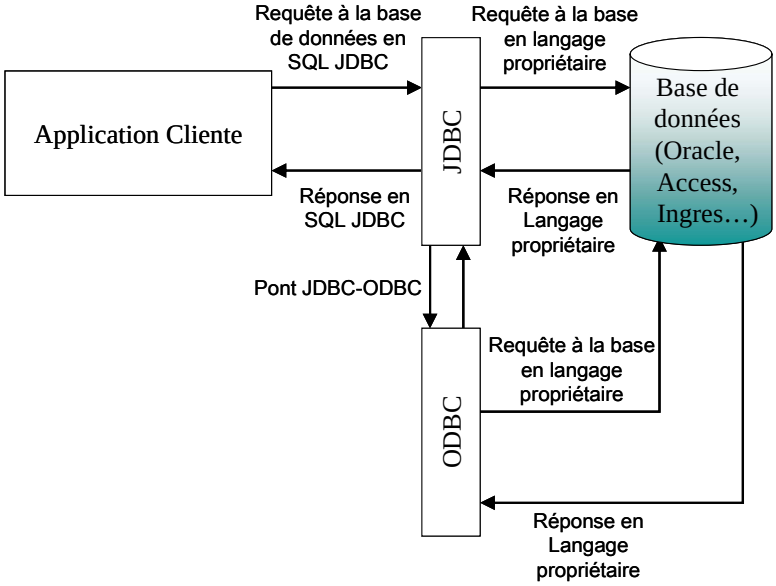

Figure III.14. Principe du JDBC

En plus de ces middlewares relativement primitifs, on utilise généralement au sein des environnements industriels d'autres techniques plus évoluées principalement basées sur la notion de réplication, de fédération de données et les entrepôts de données.

La réplication de données [Linthicum 2004] consiste à partager des données entre des bases physiquement distinctes. Les modifications apportées aux données partagées d'une base sont répliquées dans les autres bases du système de réplication (figure III.15). Un caractéristique fondamentale que ces outils doivent avoir est la prise en charge des différences de schéma de base de données. L'un des avantages de la réplication est sa simplicité et son coût de mise en œuvre alors que le principal inconvénient réside dans le fait que ce type de technique peut présenter certains problèmes à propos de la cohérence des données et surtout en ce qui concerne l'intégrité transaction sur chacune des bases de données répliquées.

Figure III.15. Principe de la réplication de données [Linthicum 2004]

Le principe de la fédération des bases de données [Linthicum 2004] consiste à accéder à des bases de données multiples à travers une couche logique unique et virtuelle.

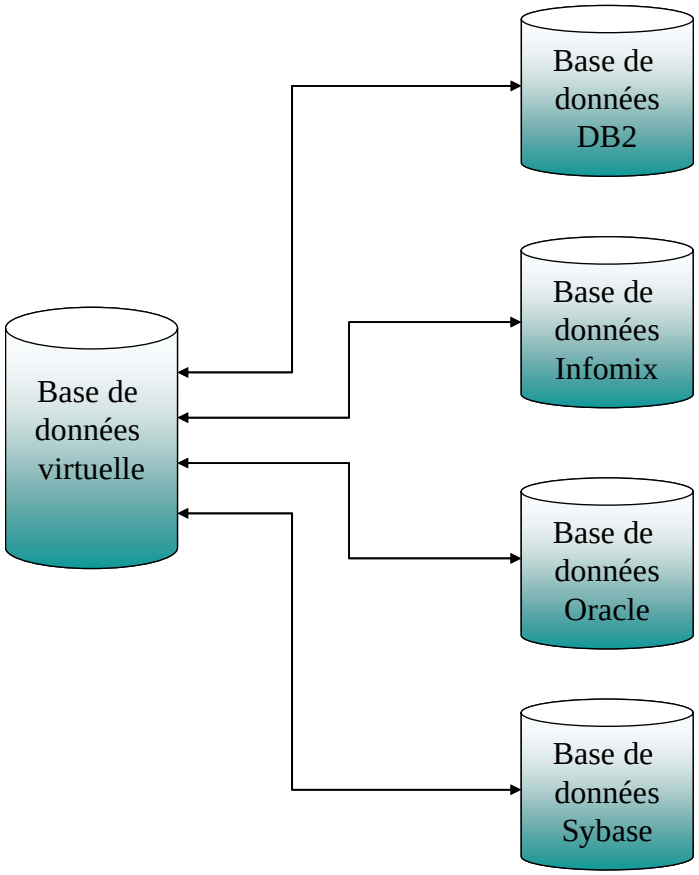

Figure III.16. Principe de la fédération de données [Linthicum 2004]

Une autre technologie qui est fortement utilisée dans les environnements industriels est la technologie ETL (Extract-Transform-Load) [Chrisment et al. 2004][Schneider 2002][Hellerstein et al. 1999] ou parfois appelée datapumping. Cette technologie permet de définir une catégorie de middleware orientée données se basant sur le déplacement massif et physique des données en utilisant des entrepôts de données ou les Data Marts (qui sont des entrepôts d'envergure assez limitée).

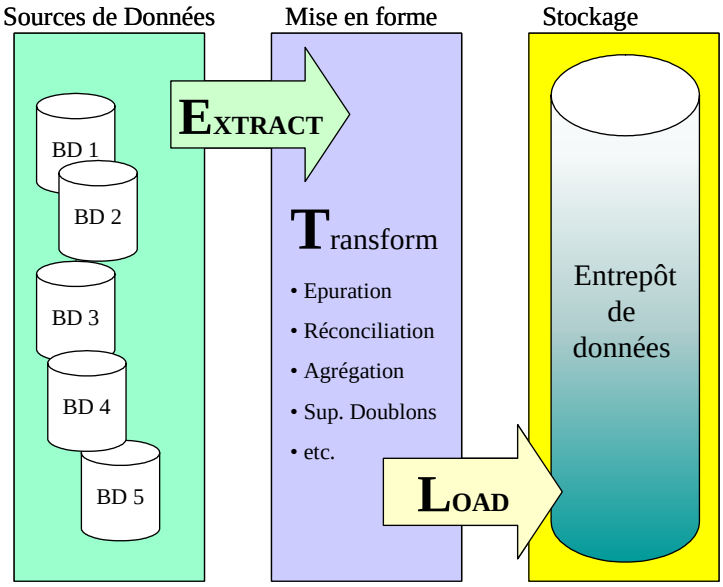

Figure III.17. Principe des entrepôts de données

Le principe de fonctionnement de ces outils repose sur trois phases essentielles qui sont (figure III.17):

- la phase d'extraction de données (Extract) des bases existantes est réalisée par un moniteur qui détecte les mises à jour sur les bases de l'entreprise afin de les envoyer vers l'entrepôt;
- la phase de transformation (Transform) qui permet d'épurer et de mettre en forme les données extraites;
- et la phase de chargement (Load) qui effectue le chargement des données dans l'entrepôt de l'entreprise.

III.5.4. Intergiciels d'appels de procédures à distance

L'appel de procédures distantes (RPC - Remote Procedure Call) constitue un des moyens les plus élémentaires pour exécuter des programmes distants à travers le réseau. Le but du service RPC est de permettre la programmation aisée de programmes client/serveur sous une forme transparente aux utilisateurs. Comme l'illustre la figure III.18, la réalisation du mécanisme RPC repose sur l'utilisation d'un serveur, qui exécute des procédures sur le site distant. Le mécanisme est synchrone dans le sens où l'appelant reste bloqué jusqu'à la terminaison de la procédure appelée. Le processus client reste bloqué pendant l'exécution de la procédure distante et il est réactivé avec la terminaison de la procédure. On introduit généralement deux procédures, appelées talons. Le talon client (stub), localisé sur le site appelant, fournit au processus client une interface identique à celle de la procédure appelée. De même, on utilise un talon serveur (skeleton), sur le site appelé, et qui réalise pour le processus serveur une interface identique à celle d'un appel local. Chacun des talons, lié au programme du processus client ou serveur sur le site correspondant, fonctionne également comme un représentant de l'autre processus (client ou serveur) sur ce site.

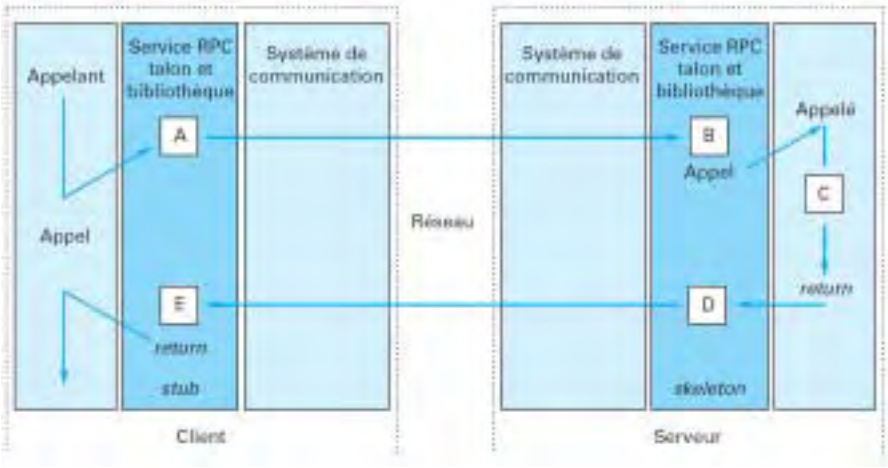

Figure III.18. Principe du RPC [Riveill 2000]

En l'absence de défaillances majeures, le scénario de fonctionnement des fonctions du talon client, exécutées par le processus appelant, sont principalement comme suit [Riveil 2000] :

- au point **A** : on exécute la fonction d'emballage (marshalling) qui permet de mettre les paramètres d'appel sous une forme permettant leur transport sur le réseau vers le site appelé;
- au point **E** : lorsque la réponse arrive du site distant, le processus client est réveillé; il poursuit son exécution dans le talon client en récupérant du message de réponse les résultats retournés après avoir été éventuellement transformés par une fonction de déballage (unmarshalling);

Sur le site appelé, le processus associé à l'exécution de la requête appelle le talon serveur et exécute les fonctions suivantes :

- au point **B** : réceptionner le message reçu du client, déterminer la procédure appelée et déballer les paramètres d'appel;
- au point **C** : exécuter la procédure appelée en lui passant les paramètres (qui constitue alors un appel de procédure local);
- au point **D** : au retour de cet appel, on prépare le message de retour vers le site appelant, en emballant les éventuels résultats et envoyer le message de retour au site appelant.

La difficulté essentielle qui caractérise le RPC est le maintien des propriétés implicitement associées à un appel de procédure locale dont notamment le codage et décodage des arguments des procédures, l'ininterruptibilité supposée du programme appelant qui peut parfois être suspendu, et la fiabilité du protocole dans la mesure où ce dernier suppose que la procédure appelée retourne toujours un résultat, ce qui n'est forcément pas toujours le cas.

Notons que les environnements à base de RPC sont aujourd'hui assez variés. A titre d'exemple on peut citer le RPC de Sun/ONC, le RPC d'OSF, le Java RMI ou encore le HTTP. Le RPC de Sun/ONC a été développé à l'origine pour la mise en oeuvre du système de fichiers répartis NFS dans des environnements Unix [Odsadzinski 1988]. Le RPC d'OSF (Open Software Foundation) appelé DCE (Distributed Computing Environment) qui permet de fournir des services distribués pour la mise en œuvre d'applications à grande échelle [OSF-DCE 1995]. Le Java RMI (Remote Method Invocation) permet de définir le protocole RPC au sein de la programmation objet

[Downing 1998]. Le HTTP (HyperText Transfer Protocol) qui permet de définir le protocole RPC dans le domaine du Web [Berners-Lee et al. 1997].

III.5.5. Intergiciels orientés composants

Les principaux intergiciels orientés composants sont :

- les intergiciels basés sur l'architecture CORBA
- les intergiciels basés sur les composants COM/DCOM

III.5.5.1. CORBA

CORBA (Common Object Request Broker Architecture) est une architecture proposée par l'OMG (Object Management Group), qui offre une solution au problème d'intégration des applications hétérogènes distribuées [OMG-CORBA 1998]. Comme toutes les autres solutions distribuées, elle exploite les objets distribués et le principe d'appel à des méthodes distantes. La particularité de l'architecture CORBA réside dans l'intégration de services implantés par des fournisseurs de logiciels en exploitant le langage IDL (Interface Definition Language) et le protocole IIOP pour la communication sur le réseau..

Dans cette architecture, les objets dialoguent par l'intermédiaire du bus CORBA qui se charge de masquer les différences entre les langages d'implantation des objets, les systèmes d'exploitation et les architectures matérielles (figure III.19). De plus, ce bus rend totalement transparente la localisation des objets permettant ainsi de simplifier l'utilisation des objets sur le réseau.

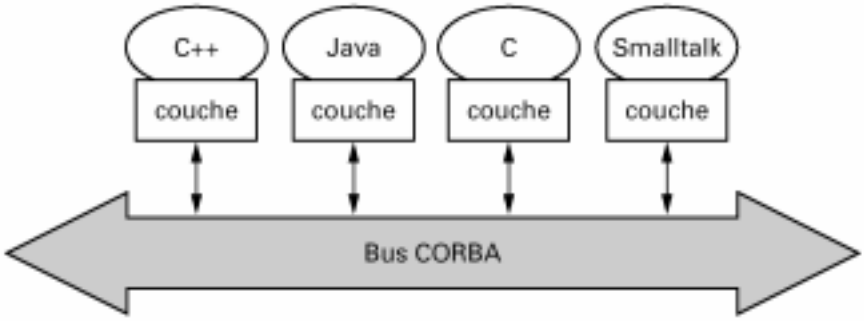

Figure III.19.Principe du bus CORBA [Geib 2000]

Le bus CORBA repose sur un modèle orienté objet client/serveur abstrait et de coopération entre les applications distribuées. Chaque application peut exporter certains de ses services sous forme d'objets CORBA : c'est la notion d'abstraction. Les interactions entre les différentes applications sont matérialisées par des invocations à distance des méthodes des objets : c'est la notion de coopération.

L'OMG définit également une vision globale de la construction d'applications distribuées, c'est l'architecture OMA (Object Management Architecture) [Soley et Stone 1995]. Cette architecture globale vise à classifier les différents objets qui interviennent dans une application en fonction de leurs rôles (figure III.20):

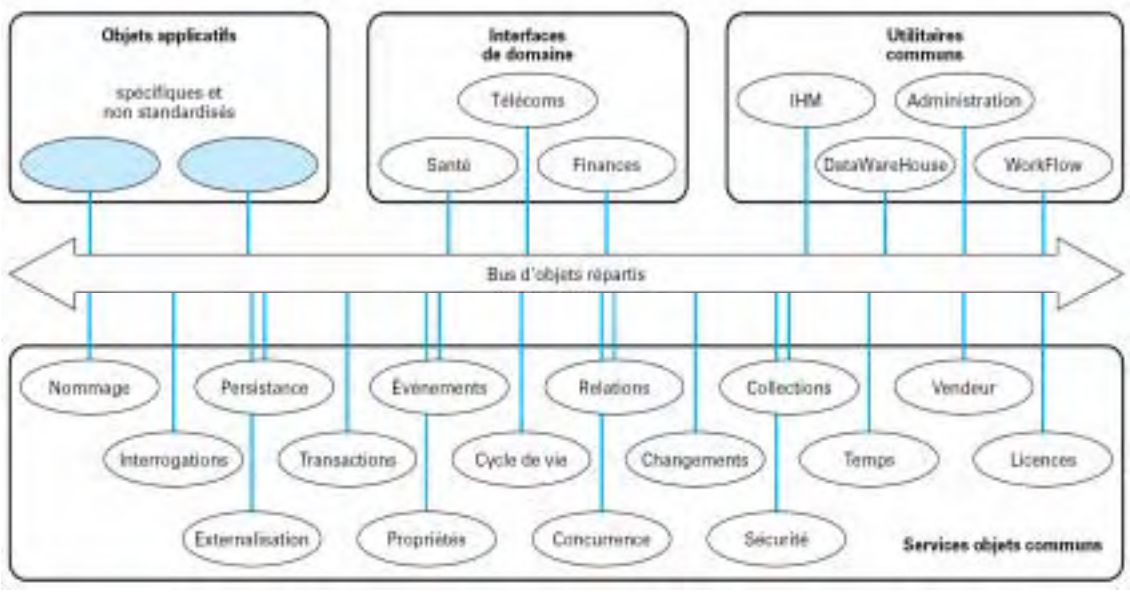

Figure III.20. Architecture OMA de l'environnement CORBA [Geib 2000]

- Le *bus d'objets répartis* (Object Request Broker) est le composant central de l'architecture globale de l'OMG. Il assure le transport des requêtes entre tous les objets CORBA. Il offre un environnement d'exécution aux objets en masquant l'hétérogénéité liée aux langages de programmation, aux systèmes d'exploitation, aux processeurs et aux réseaux.
- Les *services objet communs* (CORBA services) fournissent sous forme d'objets CORBA, spécifiés grâce au langage OMG-IDL, les fonctions systèmes nécessaires à la plupart des applications distribuées, comme les services de base pour la gestion des annuaires (Nommage et Vendeur), du cycle de vie des objets, des relations entre objets, des événements, des transactions, de la sécurité, de la persistance, etc.
- Les *utilitaires communs* (CORBA facilities) qui sont des canevas d'objets (Frameworks) qui permettent de répondre plus particulièrement aux besoins des utilisateurs. Ils permettent ainsi de standardiser l'interface utilisateur (par exemple, Fresco, OpenDoc), la gestion de l'information, l'administration, le Workflow, etc.
- Les *interfaces de domaine* (Domain Interfaces) permettent de définir des objets de métiers spécifiques à des secteurs d'activités comme la finance (par exemple, monnaie électronique), la santé (par exemple, dossier médical) et les télécoms (par exemple, transport multimédia). L'objectif de ces interfaces est de pouvoir assurer l'intégration entre les systèmes d'informations d'entreprises d'un même métier : les Business Object Frameworks (BOFs).
- Les *objets applicatifs* (Application Objects) sont ceux qui sont spécifiques à une application répartie et ne sont donc pas standardisés. Toutefois, dès que le rôle de ces objets évolue en apparaissant dans plusieurs applications ils peuvent alors migrer dans l'une des catégories précédentes et peuvent donc être standardisés par l'OMG.

La principale limitation de la technologie CORBA est son manque de standardisation. En effet, les composants développés dans les environnements CORBA sont fortement dépendants de l'environnement de développement, ce qui ne leur permet pas d'être réutilisés facilement ailleurs. Ce qui implique un manque en matière de flexibilité pour le système d'information.

III.5.5.2. COM/DCOM

COM (Component Object Model) [Microsoft et DEC 1995] est un modèle de composants introduit par Microsoft en 1995. Il constitue le format de base des applications OLE (Object Linking Embedding) [Brockschmidt 1995] qui a ensuite évolué en ActiveX[16] [Chappell 1996] très utilisés dans les environnements Windows. Le modèle COM permet la construction et l'exécution de composants binaires réutilisables, accessibles à travers une ou plusieurs interfaces, dont chacune propose un certain nombre de méthodes. Une méthode est une fonction qui exécute une action spécifique et qui peut être appelée par le logiciel utilisant l'objet COM (le client de cet objet). Les objets COM et leurs interfaces sont spécifiés en utilisant le langage de description d'interface propriétaire de Microsoft et qui s'appelle MIDL (Microsoft Interface Description Language). Ce langage est dérivé du langage de définition d'interface de DCE (Distributed Computing Environment). Ainsi, les objets COM et les clients peuvent être codés dans n'importe quel langage supportant la structure binaire des objets COM.

DCOM (Distributed COM) [Microsoft 1996] est une extension de COM pour le traitement distribué. Il permet à des objets appartenant à des machines distinctes de communiquer entre elles via un réseau, alors que COM se limite à une seule et même machine. Cette délocalisation est transparente aussi bien pour le client de l'objet que pour l'objet lui-même. DCOM est basé sur l'environnement DCE (Distributed Computing Environment), dont il exploite en particulier les appels de procédures distantes RPC (Remote Procedure Call), comparables dans leur principe au système RMI. Par conséquent, à chaque fois qu'on envoie un message DCOM à un objet distant, ce message est en réalité transporté via RPC.

La figure III.21 illustre le principe de fonctionnement du modèle COM/DCOM. Comme on peut le constater, chaque objet COM s'exécute au sein d'un serveur. Il existe trois manières distinctes, pour une application cliente donnée, d'accéder aux objets COM fournis par un serveur :

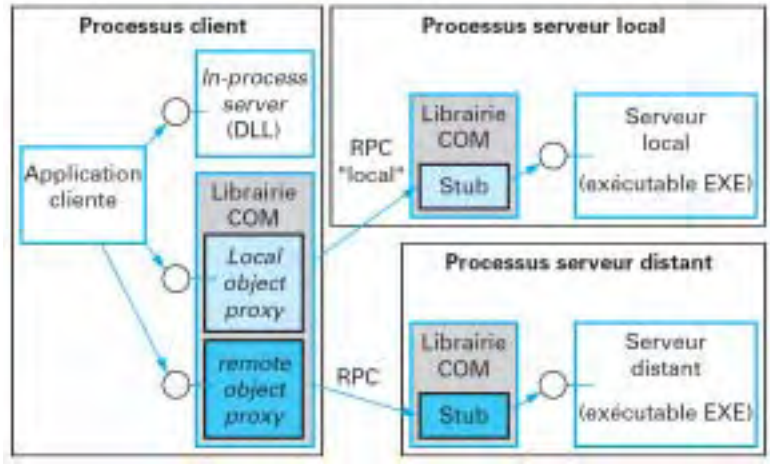

Figure III.21. Principe d'utilisation des composants COM (adapté de [Microsoft 1996])

[16] Rappelons que technologie OLE est utilisé pour la construction d'objets liés et "embarqués" à l'intérieur de documents composites de l'environnement Windows, alors que la technologie ActiveX étend la technologie OLE pour permettre aux composants d'être inclus dans des sites Web.

- l'application cliente établit un lien direct avec une bibliothèque ou une librairie (DLL) contenant le serveur. Le client et le serveur s'exécutent à l'intérieur d'un même processus (in-process server). La communication entre le client et le serveur se fait au travers d'appels de procédures;
- l'application cliente accède au serveur s'exécutant dans un processus différent mais localisé sur la même machine au travers d'un mécanisme de communication inter-processus réalisé par le proxy d'objets locaux (local object proxy). Ce mécanisme est basé sur le protocole appel de procédure à distance (RPC - Remote Procedure Call));
- l'application cliente accède au serveur distant s'exécutant sur une autre machine en utilisant le mécanisme DCOM qui se base sur le protocole RPC/DCE.

De même que pour la technologie CORBA, la technologie COM/DCOM est une technologie propriétaire et qui ne fonctionne que dans les environnements Windows de Microsoft. Ceci limite considérablement la flexibilité des applications développées avec ce type de technologie.

III.5.6. Intergiciels orientés messages

Les intergiciels orientés messages permettent l'échange de messages entre applications en utilisant une communication asynchrone. L'intérêt de ce mode communication (asynchronisme) réside dans le fait que les applications qui échangent des messages ne sont plus liées l'une à l'autre [Hohpe et Woolf 2004], elles peuvent donc avoir des caractéristiques de disponibilité complètement différentes. De plus, ce mode de communication permet aux applications de traiter les informations "au fil de l'eau", c'est à dire au fur et à mesure de l'arrivée des données.

Le modèle de communication des MOM est traditionnellement représenté par une succession de couches ou de modèles qui symbolisent chacune une évolution historique et pragmatique de la communication par messages où chaque couche utilise les fonctionnalités fournies par la couche précédente. Comme illustré en figure III.22, les principaux modèles de middlewares orientés messages sont le modèle de queue de message (message queuing), modèle par abonnement (ou modèle Publish/Subscribe) et le modèle événementiel.

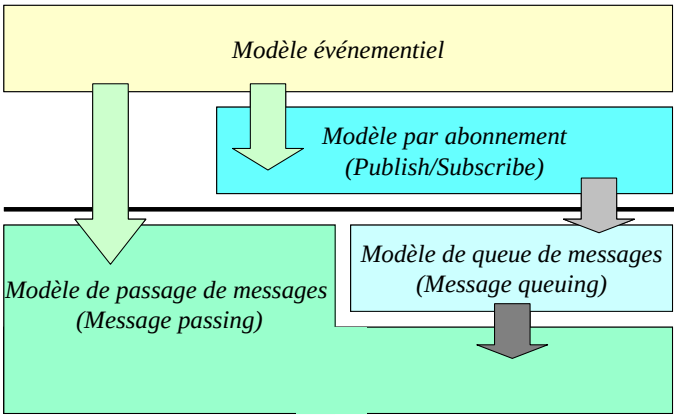

Figure III.22. Les principaux types de middlewares orientés messages

Le modèle de passage de messages (ou modèle de Message Passing) [Reiss 1990] constitue le mode de base de la communication par messages. Il réalise l'envoi de messages simples de façon unidirectionnelle et généralement non bloquante à la même manière d'un envoi d'une lettre. Ce mode est sans connexion car la disponibilité de l'application destinataire à l'instant de l'émission n'est pas nécessaire. La figure III.23 illustre le principe de fonctionnement de ce type de middleware. L'application émettrice qui désire envoyer un message exécute une instruction *SEND* en indiquant le nom de la boite aux lettres de l'application destinataire. Cette dernière doit exécuter, pour réceptionner le message, une instruction *RECEIVE*.

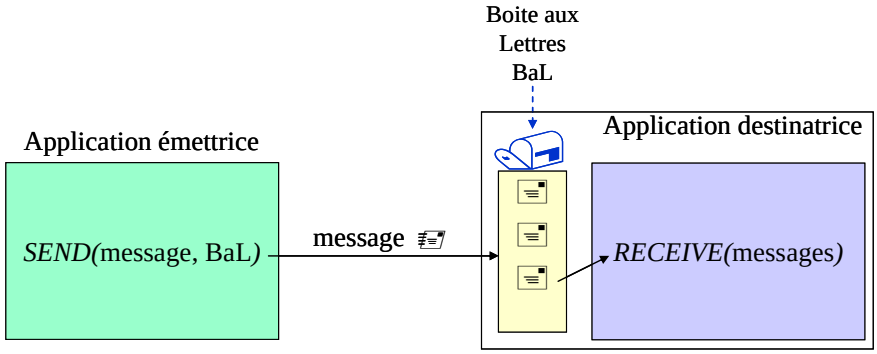

Figure III.23. Principe du Message Passing

Le Message Queuing [Ribière 1998] étend le modèle de passage de messages en rajoutant la notion de persistance ce qui apporte plus de fiabilité aux communications inter-applicatives. La persistance est réalisée en utilisant une file d'attente de message entre les applications communicantes (figure III.24). Le rôle de cette persistance est d'assurer la sécurisation des messages envoyés en permettant ainsi de conserver leur intégrité quelle que soit la situation qui peut survenir comme l'arrêt de l'application destinataire, arrêt du support physique contenant la file d'attente. Bien que pertinent, ce modèle peut s'avérer limité et contraignant dans la mesure où il est nécessaire que l'application émettrice connaisse le nom et l'adresse de l'application destinatrice, et de plus, le modèle ne supporte pas de façon souple les envois multiples car il faudrait dans ce cas effectuer plusieurs envois de la même information. Des exemples d'outils commercialisés à base de files d'attente de messages sont : MQSeries d'IBM, SonicMQ de Sonic Software ou MSMQ de Microsoft.

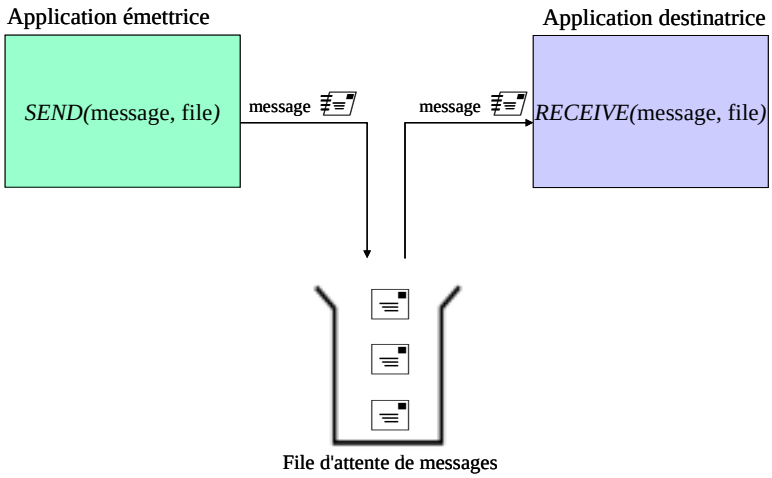

Figure III.24. Principe du Message Queuing

Le modèle par abonnement (ou modèle Publish/Subscribe) [Banavar et al. 1999], utilise les fonctions du Message Passing ou du Message Queuing et ajoute la notion d'anonymat et d'abonnement. Dans ce modèle, il y a deux types d'applications: des applications clientes et des applications fournisseur. Les applications clientes (consumers) s'abonnent à des sujets (types d'événements) tandis que les applications fournisseurs (providers) produisent et publient des événements (messages). Ces derniers sont alors envoyés au gestionnaire de communications qui se charge de le répartir aux applications qui se sont abonnées à ce type d'événement (figure III.25). L'avantage de ce modèle tient de sa flexibilité en comblant ainsi les lacunes du modèle du message queuing.

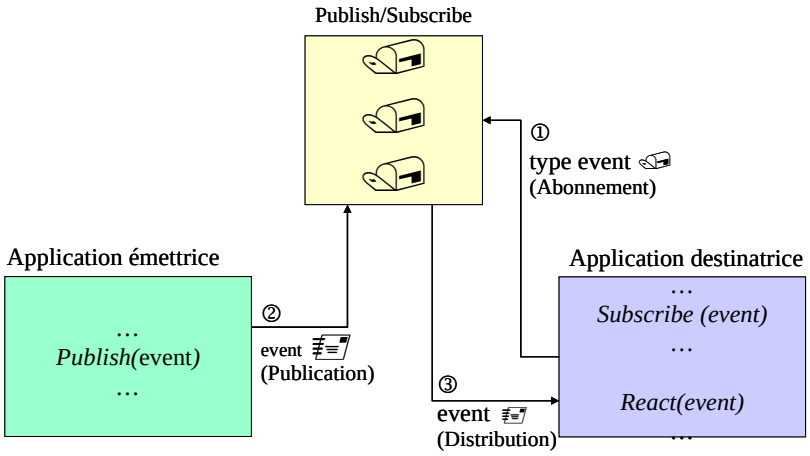

Figure III.25. Principe du modèle Publish/Subscribe

Le modèle événementiel [Oki et al. 1993] constitue le modèle le plus évolué des middlewares orientés messages. Il permet de gérer les événements. De par son perfectionnement, ce modèle est généralement considéré beaucoup plus comme un modèle de programmation que comme un modèle de Communication. Il utilise généralement les règles actives E-C-A (Evénement-Condition-Action) qui apportent une nouvelle façon de programmer. Le schéma de programmation typique des règles du modèle événementiel est le suivant (figure III.26) :

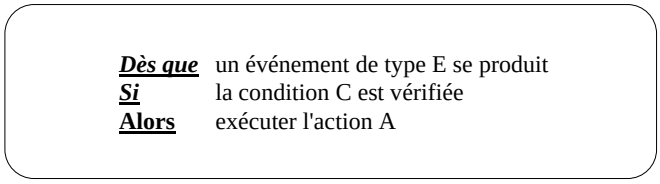

Figure III.26. Format des règles E-C-A

Le modèle se base alors sur un mécanisme dynamique qui permet de gérer les événements de façon dynamique et à la volée. Pour cela, le modèle se base sur le principe des écouteurs (*listeners*) et qui permettent à une application de s'enregistrer comme écouteur d'un certain événement. Les applications peuvent alors émettre des événements et ces derniers sont captés par le ou les écouteurs définis. La figure suivante (figure III.27) illustre le principe de fonctionnement de ce type de modèle.

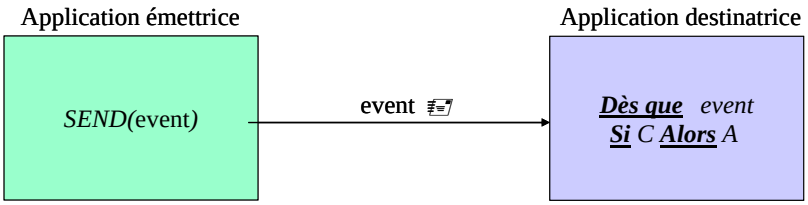

Figure III.27. Principe du modèle événementiel

III.5.7. Intergiciels orientés transactions

Les intergiciels orientés transactions, également appelés les moniteurs transactionnels, sont des systèmes qui permettent principalement de gérer les accès aux bases de données de façon à garantir l'intégrité des transactions qui y sont effectuées. L'intégrité des transactions est généralement assurée en définissant des transactions de type ACID. ACID est l'abréviation utilisée pour dénoter l'ensemble des quatre propriétés de la transaction et qui sont [Printz 2000] :

- la propriété d'atomicité (A) : qui signifie que la transaction est entièrement exécutée ou pas du tout. Si la transaction échoue pour une raison quelconque, tout doit être remis en place comme s'il ne s'était rien passé;
- la propriété de consistance (C) : qui signifie que la transaction préserve la consistance de la base de données;
- la propriété d'indépendance ou d'isolation (I) : qui désigne le fait qu'une transaction doit pouvoir être exécutée et reprise à son début sans influence sur les autres transactions; elles ne peuvent donc pas s'échanger sans précaution des données autrement que par la base de données;
- la propriété de durabilité (D): qui veut dire que le moniteur garantit que les résultats d'une transaction survivront à tout dysfonctionnement survenant ultérieurement. Une opération durable modifie la réalité. Pour annuler la modification, il n'y a pas d'autres moyens que d'effectuer l'opération inverse.

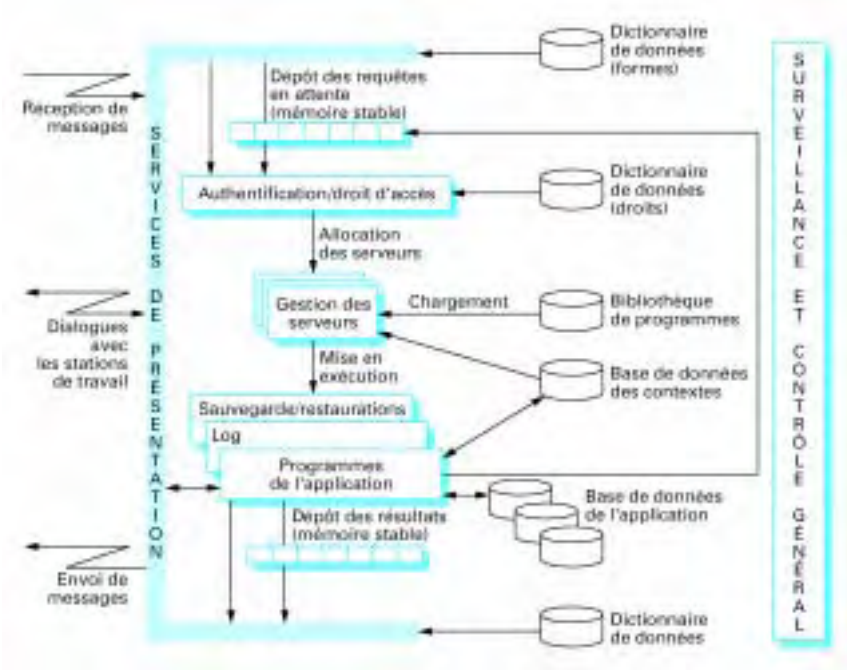

Figure III.28. Architecture générale d'un moniteur transactionnel [Printz 2000]

La figure III.28 permet d'illustrer l'architecture générale d'un moniteur transactionnel. Comme on peut le remarquer, en plus de la gestion des transactions ACID, le moniteur doit prendre en charge un certain nombre de tâches dont: les services d'authentification, les services de gestion des serveurs, et les services de surveillance et de contrôle du déroulement des transactions.

III.5.8. Serveurs d'applications

Les serveurs d'applications constituent une évolution des moniteurs transactionnels. Ils sont des environnements de développement et d'exécution permettant de masquer la complexité de réalisation d'applications distribuées, robustes, capables de monter en charge. Un serveur d'applications est donc un intergiciel sophistiqué qui permet généralement de combiner trois principaux types composants : des composants de communication avec les systèmes back-end, les composants de communication avec les clients front-end, et un framework qui permet d'exécuter la logique métier. Il existe principalement deux catégories de serveurs: les serveurs J2EE et les serveurs propriétaires, et parmi les serveurs d'application les plus populaires du marché, on peut citer pour les serveurs J2EE : BEA Weblogic, IBM Websphere, Sun/Netscape iPlanet, Jonass, Jboss, et pour les serveurs propriétaires : Microsoft .NET, Python Zope, OpenACS.

La figure III.29 illustre le principe des serveurs d'applications basés sur l'architecture J2EE [Cloux 2003]. Le serveur d'application héberge un certain nombre de composants dont des containers (EJB servers) servant à stocker des composants métiers (EJB – Enterprise Java Beans) et une panoplie de services et d'API permettant le nommage, , le déploiement des application, l'accès aux données d'entreprise (JDBC – Java DataBase Connectivity, JCA - J2EE Connector Architecture, Java CICS), la gestion des communications interprocessus (RMI - Remote Method Invocation), la gestion des messages (JMS - Java Message System), ou encore la gestion des transaction (JTS - Java Transaction Service, JTA - Java Transaction API, RMI-IIOP - Internet Inter-ORB Protocol).

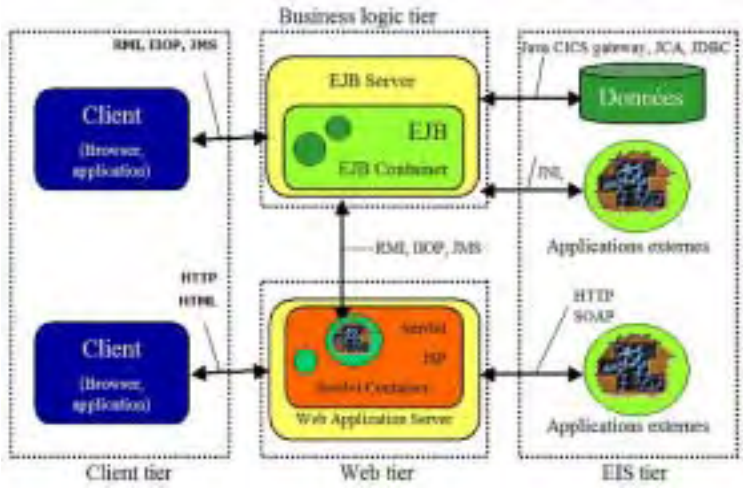

Figure III.29. Principe des serveurs d'applications J2EE

Tous les intergiciels présentés dans la section III.5 sont des outils qui peuvent être mis en œuvre en vue de l'intégration syntaxique des systèmes d'information industriels. Ces

outils sont simples et permettent d'interconnecter des applications de façon plus ou moins rudimentaire. Pour simplifier, on peut considérer qu'ils sont principalement destinés à prendre en charge l'aspect communicationnel de l'intégration des systèmes. Ces outils constituent plus précisément des outils de base sur lesquels d'autres outils plus sophistiqués sont généralement construits. Dans ce qui suit, nous allons étudier certains des ces outils sophistiqués qui sont construits sur la base des intergiciels. Nous allons en particulier étudier les outils EAI, les outils BPMS et les Services Web.

III.6. Intégration d'applications d'entreprise (EAI)

Les outils EAI (Enterprise Application Integration) constituent un type d'outils récent pour l'intégration.Ils sont apparus vers le milieur des années 90 dans le monde industriel pour interconnecter des applications existantes [Linthicum 2000]. On peut définir un EAI comme un outil intégré supportant trois grandes fonctions [Chevassus 2005] : transport- connectivité, transformation-adaptation des flux de données, et l'automatisation-orchestration des processus. Il s'agit en quelque sorte du mariage d'une multitude d'intergiciels (pour le transport et la connectivité), des outils ETL (Extract, Transform, Load) et des moteurs de workflow (cf. § III.7).

Le principe majeur des outils EAI se base sur la démultiplication des connexions entre les applications d'entreprise ce qui réduit considérablement la complexification des systèmes et donc leurs coûts d'intégration en développement et en maintenance. Le nombre des connexions nécessaires pour intégrer *n* applications est donc de l'ordre de *o(n)* (figure III.30). De plus, les outils EAI permettent généralement l'utilisation du format Pivot comme clé de l'interfaçage des systèmes pour assurer la communication entre les applications hétérogènes.

Figure III.30. Principe de l'EAI (adapté de [Chevassus 2005])

L'architecture générale d'un outil EAI comprend principalement quatre briques de base qui sont : transport, connecteurs, transformation et routage (figure III.31)**.** Ces briques permettent au logiciel d'EAI de récupérer des données d'une application, puis les router vers leurs applications destinatrices après les avoir converties dans le format approprié. Les briques de transformation et routage sont souvent regroupés dans la littérature sous l'expression de "moteur d'intégration". De plus, certains outils permettent de prendre en charge l'orchestration des applications à travers une brique supplémentaire appelée BPM (Business Process Management) ou Workflow.

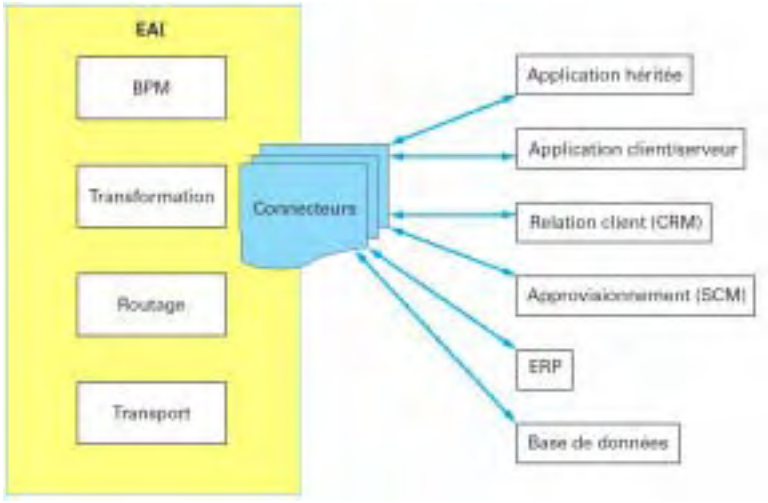

Figure III.31. Architecture des outils EAI (adapté de [Chevassus 2005])

La brique de transport (ou d'échange de message) permet de véhiculer les messages de manière sûre en utilisant les intergiciels de communication de type MOM (cf. § III.5.6). La brique de transformation permet de traduire dynamiquement les messages dans le format attendu par le destinataire. Le routage a pour rôle d'acheminer les messages en fonctions de règles basées sur des critères internes au message (ex, le type du message, la valeur d'une rubrique du message), des considérations opérationnelles (ex, une date d'échéance), etc. Les adaptateurs (ou connecteurs) qui permettent aux applications de se connecter facilement à l'infrastructure de l'EAI. Ils jouent le rôle de fils de raccordement (ou prise) à l'infrastructure EAI. Ces adaptateurs permettant notamment l'accès aux principaux progiciels du marché dont par exemple CRM de Siebel, PGI de SAP. La brique BPM, qui se place au-dessus des couches de transformation et de routage, permet la modélisation et la gestion des processus métier. Cette brique permet d'ajouter un niveau d'abstraction en donnant une vision plus métier, orientée processus, des flux de données inter-applicatifs.

En ce qui concerne les outils EAI les plus populaires du marché, on peut citer principalement [jdNet 2006][Programmez 2003] : BEA Weblogic Integration, Microsoft Biztalk, IBM Websphere Business Integration, SeeBeyond Integration Platform, Tibco Active Enterprise, Vitria BusinessWare, WebMethods Integration Platform. Une analyse de certains de ces outils est donnée en annexe 1.

Parmi les limitations majeures de ces outils, on peut citer leur lourdeur, leur caractère propriétaire et surtout le fait qu'ils ont pour ambition d'intégrer l'ensemble des fonctionnalités de l'entreprise au niveau d'une application monolithique, ce qui limite considérablement la flexibilité du système d'information. L'heure est plutôt à l'interopération d'applications modulaires et modulables reposant sur des infrastructures flexibles [Linthicum 2000].

III.7. Gestion des processus (BPM)

Bien qu'ils soient souvent incorporés dans les outils EAI, les outils de gestion de processus BPMS (Business Process Management System) [Dayal et al 2001][Johannesson et Perjons 2001][Linthicum 2004] peuvent être considérés comme

des outils à part entière. Le BPM est basé sur la notion de processus métier. Un processus métier est un ensemble d'activités ou de procédures, qui implique collectivement plusieurs domaines de l'entreprise pour satisfaire une demande externe dans le contexte d'une organisation définissant des relations et des rôles fonctionnels [WfMC 2000].

Le principe du BPM consiste à assouplir le fonctionnement des entreprises en modélisant le fonctionnement de l'entreprise en termes d'activités et d'événements métiers faisant intervenir aussi bien des humains que des applications et en exécutant ces processus à l'aide d'un moteur approprié. La notion de processus métier est très souvent confondue avec la notion de workflow. Ce dernier constitue une automatisation totale ou partielle d'un processus business dans lequel les documents, les informations et les tâches sont échangés entre les participants pour action, selon un ensemble de règles permettant de mener à bien, ou de contribuer à, une finalité de gestion [Pontacq 2002].

La différence fondamentale entre un workflow et un processus métier tient donc du fait que le workflow ne concerne que la coordination automatisée de tâches réalisées par des intervenants humains, alors qu'un processus métier peut inclure en plus des tâches automatisées qui peuvent être prises en charge par des applications informatiques. La figure III.32 illustre le principe d'intégration et de l'intégration des applications d'entreprise avec les moteurs BPM et/ou workflow. Comme on peut le remarquer, l'idée majeure du BPM est d'offrir une vue simplifiée à un utilisateur métier à travers le BPMS ou le courtier de processus (Process Broker) qui constitue un médiateur entre l'utilisateur et l'ensemble des applications et/ou des personnes humaines à intégrer.

Il est important de noter que le BPM n'impose généralement aucune contrainte sur le type de participant à utiliser dans les scénarios d'intégration. Toutes sortes d'élément logiciel, existant ou à venir, peuvent être utilisés pour implémenter le processus métier d'entreprise. En particulier, ceci rend la tâche relativement complexe. Une des approches les plus pertinentes à l'heure actuelle, pour mettre en œuvre les techniques basées sur le BPM est de s'inscrire dans le contexte des services Web et des architectures orientées services (cf. § III.8). A cet effet, de nombreux standards de BPM existent pour orchestrer les services Web tels que BPEL4WS (Business Process Execution Language for Web Services) [IBM et al. 2005], BPML (Business Process Modelling Language) [BPMI 2003], etc.

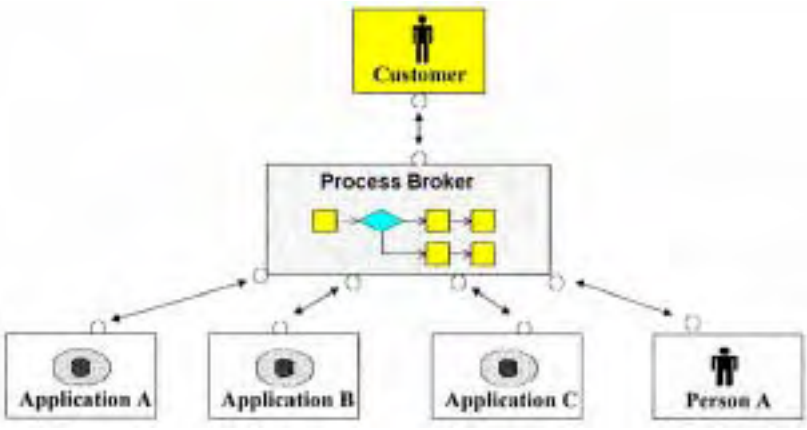

Figure III.32. Principe d'intégration par le BPM [Johannesson et Perjons 2001]

Remarquons qu'un processus métier peut être interne à une entreprise (gestion de la production par exemple), ou mettre en jeu des entreprises partenaires – on parle alors de

processus collaboratifs ou de processus métiers B2B (la gestion de commande impliquant plusieurs partenaires par exemple). Et notons que les modèles les plus utilisés pour représenter et/ou manipuler les processus métier sont BPMN (Business Process Modeling Notation), BPML (Business Process Modeling Language) (cf. § III.4.1.2.2), BPEL (Business Process Execution Language) et BPQL (Business Process Query Language).

III.8. Architectures de services (SOA)

Une autre technique relative à l'intégration qui a, ces dernières années, le vent en poupe et qui commence à investir les milieux industriels est la technique basée sur les architectures orientées services. L'architecture orientée services (SOA - Service-Oriented Architecture) constitue un style architectural de développement et d'intégration dynamique des applications d'entreprise. Il se base sur la notion de service qui est considéré comme la brique de base de cette architecture.

III.8.1. Notion de service

La notion de service logiciel correspond à une fonctionnalité offerte par un composant applicatif et dont l'interface est indépendante de la plate-forme ou de la technologie utilisée pour le développement du composant. Les principales caractéristiques d'un service logiciel sont les suivantes [Leroux 2004][Erl 2004][Singh et Huhns 2005]:

- il est auto-descriptif et permet la réutilisation dans le sens où il fournit une description de son interface;
- il est autonome, complet et cohérent dans la mesure où le service contient toute sa logique d'exécution (self-contained) ce qui lui permet de s'exécuter indépendamment des autres services;
- le service peut être localisé dans la mesure où il possède une adresse ce qui permet de l'invoquer dynamiquement.

III.8.2. Architecture Orientée services

Le principe d'une architecture orientée services consiste à structurer le Système d'Information d'Entreprise comme un ensemble de services qui exposent leur interface fonctionnelle et qui communiquent par messages. Dans une SOA, nous avons fondamentalement trois principaux rôles: le *fournisseur de services* (service provider), le *client du service* (service requestor) et le *registre de services* (service registry); ainsi que trois types d'opérations basiques : *publier* (publish), *rechercher* (find) et *interagir* (bind). Toute entité d'une SOA peut jouer un ou plusieurs de ces rôles et peut effectuer une ou plusieurs de ces types d'opérations (figure III.33) [Kontogiannis et al. 2002][Kadima et Monfort 2003].

Dans le modèle SOA, chaque service est défini par un fournisseur. Le fournisseur de services publie (publish) la description de son service dans des registres de services spécialement conçus en vue d'être interrogés par des clients. Les clients de services (applications clientes) localisent (find) leurs besoins en terme de services en effectuant des recherches sur le registre de services. Une fois le service localisé, le client récupère

sa description du registre et sur la base des informations fournies dans la description du service, le client interagit (bind) alors avec le service en vue de l'exécuter.

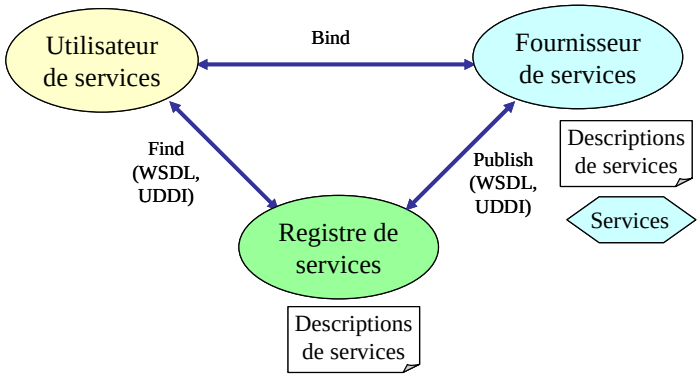

Figure III.33. Principe des SOA (adapté de [Kadima et Monfort 2003])

III.8.3. Services Web

Les Services Web sont considérés comme le résultat de convergence du Web et des technologies objet distribuées [Acharya 2003]. Ils sont définis comme des services logiciels déployés sur l'Internet [Kadima et Monfort 2003]. Les Services Web constituent l'instanciation la plus importante du modèle SOA dans le domaine industriel. Les principaux apports de la technologie des Services Web sont notamment l'universalité, la standardisation, le couplage faible entre les services, et la virtualisation qui permet de définir une indépendance vis à vis de la localisation et de l'implémentation des services.

Les Services Web peuvent être déployés aussi bien à l'intérieur comme à l'extérieur d'une entreprise. Dans tous les cas, les Services Web sont publiés avec des URLs appropriées par les fournisseurs de services Web à travers l'Internet ou l'Intranet. Une fois publiés, ces Services Web sont accessibles par des clients de Services Web. Ce qui permet alors aux applications d'entreprise de dialoguer ensemble à distance via Internet, indépendamment des environnements techniques et des langages sur lesquels tournent ces applications.

Une caractéristique majeure déjà évoquée des Services Web est qu'ils se basent sur des protocoles industriels standardisés. Ces derniers sont basés sur XML comme le standard SOAP (Simple Object Access Protocol) utilisé comme middleware pour le transport des messages, le standard WSDL (Web Service Description Language) utilisé comme langage pour la description des services et le standard UDDI (Universal Description, Discovery and Integration) utilisé pour le stockage et la découverte de services à travers le Web [Amjad 2004] (figure III.34).

En outre, les Services Web peuvent être intégrés en utilisant des langages de composition de services comme BPEL4WS (Business Process Execution Level For Web Services), XLANG (XML business process LANGuage) ou WSFL (Web Service Flow Language). Ces langages permettent de définir des implémentations, de manière procédurale, de processus métier. Dans ce qui suit, nous avons choisi de présenter brièvement quelques standards de base les plus représentatifs des services web et qui sont SOAP, WSDL, UDDI et BPEL4WS.

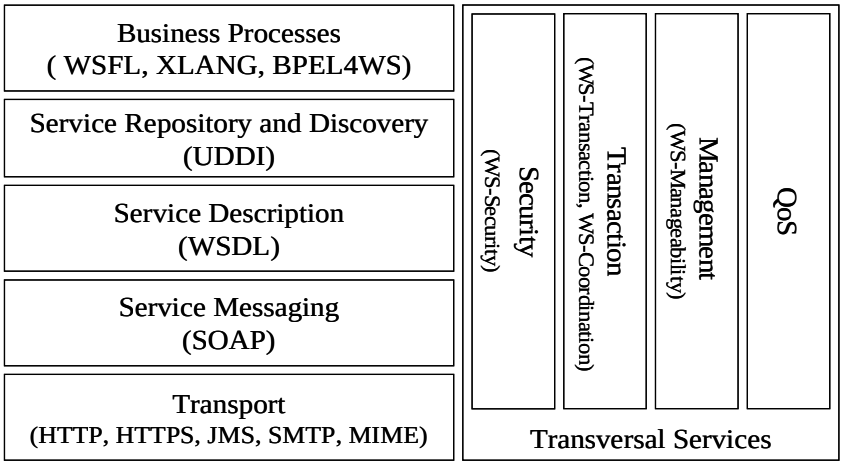

Figure III.34. Principaux standards des Services Web

III.8.4. Principaux standards des services Web

III.8.4.1. SOAP

SOAP (Simple Object Access Protocol) est un standard du consortium W3C définissant un protocole qui assure des appels RPC en s'appuyant principalement sur le protocole HTTP et sur XML. Il constitue un protocole léger d'échange de données dans un réseau de pair à pair, c'est-à-dire décentralisé. Il existe deux modes de fonctionnement de SOAP le mode RPC et le mode Messagerie. Le mode RPC (Remote Procedure Call) est un mode de fonctionnement requête-réponse synchrone, tandis que le mode Messagerie permet de fonctionner au choix, en mode synchrone ou asynchrone.

La structure des messages SOAP se divise en quatre parties (figure III.35):

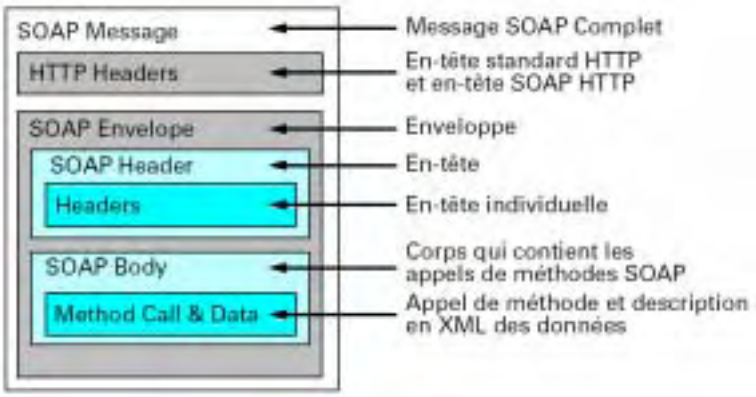

Figure III.35. Structure d'un message SOAP [W3C 2001]

- l'enveloppe SOAP, qui définit le contexte d'un message, son destinataire, son contenu et différentes options;
- un en-tête qui permet d'ajouter à un message SOAP des attributs spécifiques qui ne sont pas acceptés par toutes les parties et qui peuvent donc être « négociés » au cas par cas;
- un corps qui contient le message SOAP proprement dit et qui permet généralement de décrire le nom de la procédure et la valeur des paramètres d'appel dans le cas

d'une demande de service ou la valeur des paramètres de retour après l'exécution du service.

III.8.4.2. WSDL

WSDL (Web Services Description Language) est un standard du W3C qui permet de définir une syntaxe XML pour décrire les méthodes et paramètres des Web Services invocables par le biais de messages au format SOAP. Il permet de définir qu'est-ce qu'un Web Service est capable de faire, où est-ce qu'il réside et comment l'invoquer. De même que SOAP, WSDL est un langage qui se veut indépendant des protocoles de transport de message. Il spécifie, par défaut, comment employer directement HTTP, SOAP ou MIME[17] comme couche de transport de messages. Bien entendu, il reste évidemment possible d'employer des *middlewares* traditionnels notamment ceux déployés dans le cadre d'une EAI comme MQSeries d'IBM, MSMQ de Microsoft, voire des couches de transport RPC comme IIOP de Corba ou RMI pour le langage Java.

Dans WSDL, les services communiquent par échange de messages entre des terminaisons, constituées d'un ou de plusieurs ports, chacun doté d'un type – au sens des langages de programmation. La spécification WSDL 1.0 définit les entités suivantes (figure III.36):

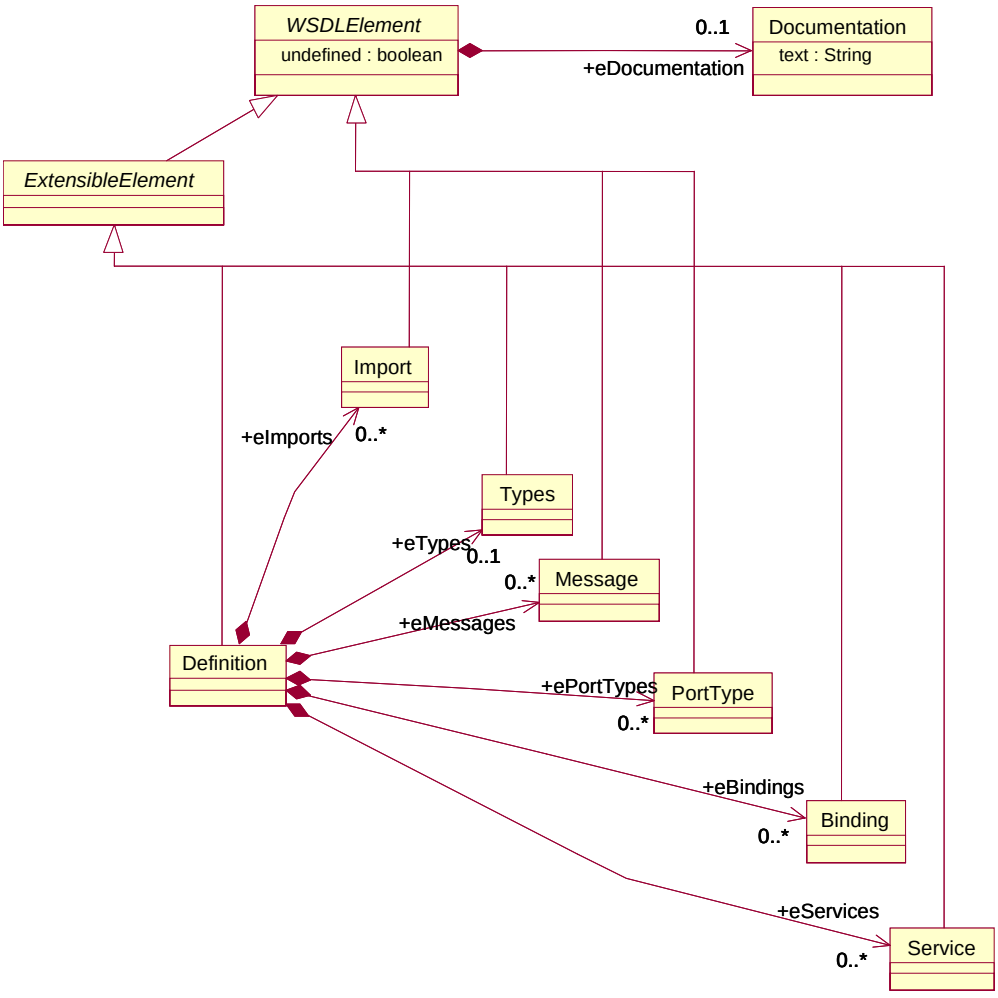

Figure III.36. Extrait du méta-modèle WSDL

[17] MIME (Multipurpose Internet Mail Extensions) [NWG 1993] est le format d'extension du courrier électronique pour la prise en compte de données non textuelles (son, image, vidéo…).

- types : un système de types applicable à des données, pour lequel WSDL emploie XSD (XML Schema);
- message : l'abstraction décrivant les données échangées entre services Web;
- opération : l'abstraction décrivant une action implémentée par un service Web;
- type de port : un ensemble d'opérations implémenté par une terminaison donnée;
- liaison (binding) : un protocole concret d'accès à un port et un format de spécification des messages et des données pour ce port;
- port : un point de terminaison identifié de manière unique par la combinaison d'une adresse Internet et d'une liaison;
- service Web : une collection de ports.

III.8.4.3. UDDI

UDDI (Universal Description, Discovery and Integration) est un standard qui né à l'initiative d'un regroupement d'entreprises dont Ariba, IBM, Microsoft. Il constitue un référentiel commun d'entreprises visant à établir un format d'annuaire des services Web. UDDI permet de décrire et de localiser une entreprise (ou toute unité organisationnelle) ou un service, à partir d'interfaces qui sont personnalisables. L'annuaire indique également comment accéder aux services Web.

UDDI se comporte lui-même comme un Web Service dont les méthodes sont appelées via le protocole SOAP. UDDI est composé de deux parties :

- l'UDDI Business Registry, annuaire d'entreprise et de Web Services
- les interfaces d'accès à ces annuaires, et les modèles de données.

L'objectif de standardisation UDDI a pour fin de construire un annuaire global qui soit pour les services Web et les entreprises qui les offrent ce que les noms de domaine et les adresses IP sont aux sites Web [Chauvet 2002-b]. Cet annuaire universel (UDDI Business Registry) rassemble l'équivalent des "pages blanches" (adresses, points de contact, identités des services), des "pages jaunes" (classement des services selon des taxinomies standardisées) et des "pages vertes ou roses" donnant des informations complémentaires et le mode d'emploi de l'annuaire. Le tableau suivant résume les trois types d'informations disponibles au format XML et que contient le UDDI Business Registry.

Information	Utilité
Pages blanches: Informations publiques de type nom, adresse, téléphone et autres coordonnées.	Permet au fournisseur du Web Service de se référencer lui-même.
Pages jaunes: Informations sur la classification de l'entreprise au moyen de taxinomies standard.	Permet de chercher et trouver un Web Service particulier.
Pages vertes: Description technique des différents Web Services offerts par l'entreprise.	Permet de comprendre comment une application peut se connecter et interagir avec le Web Service trouvé.

Tableau III.1. Les différents types d'utilisation d'un UDDI

Les données enregistrées au sein d'un UDDI sont organisées dans cinq structures de données principales qui sont (figure III.37) : businessEntity, businessService, bindingTemplate, tModel et publisherAssertion. Ces éléments sont décrits comme suit :

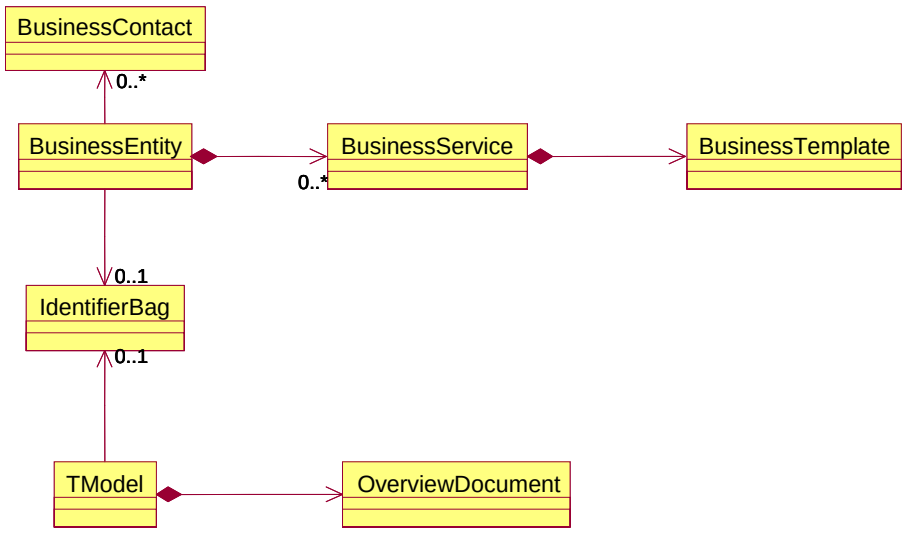

Figure III.37. Extrait du méta-modèle UDDI

- le businessEntity permet de fournir les informations concernant le business (fournisseur de services). Il peut comporter un ou plusieurs businessServices;
- le businessService et les bindingTemplate permettent d'enregistrer les descriptions techniques et métiers d'un service web;
- le bindingTemplate contient une référence à un ou plusieurs tModels;
- un tModel permet d'enregistrer la spécification technique d'un service;
- le publisherAssertion permet de définir une relation contractuelle de service entre deux services métiers.

III.8.4.4. BPEL4WS

Né d'une fusion de deux anciens standards rivaux WSFL (Web Services Flow Language) d'IBM, et XLANG (XML Business Process Language) de Microsoft, BPEL4WS (Business Process Execution Language for Web Services) [Andrews et al. 2003] est un langage de programmation basé sur XML dont le but est de décrire très précisément tous les paramètres techniques nécessaires à l'exécution d'un processus - faisant appel à des services - au sein d'un moteur d'orchestration. BPEL4WS permet plus précisément de spécifier les actions qui doivent s'effectuer au sein d'un processus opérationnel et décrire les services Web offerts par ce processus grâce à WSDL. Ces spécifications vont ensuite servir à la fois pour l'exécution du modèle de processus et aussi pour les échanges dans la mesure où les descriptions BPEL4WS qui spécifient les échanges entre partenaires sont assimilés à des protocoles d'échanges de messages.

Un document BPEL4WS utilise XML pour décrire les multiples aspects d'un processus métier (process) et qui sont (figure III.38) :

- partners : les web services invoqués en tant que parties du processus.
- activities : qui permettent de décrire les activités d'un processus.
- containers : décrivent les conteneurs de données utilisés par le processus et qui fournissent les définitions des données en termes de types de messages WSDL.

- faultHandlers : qui décrivent les routines d'exception.
- compensationHandler : décrivent les routines de compensation à exécuter en cas de rollback (annulation de transaction).
- correlationSets : représentent les dépendances qui peuvent exister entre les invocations de services Web.

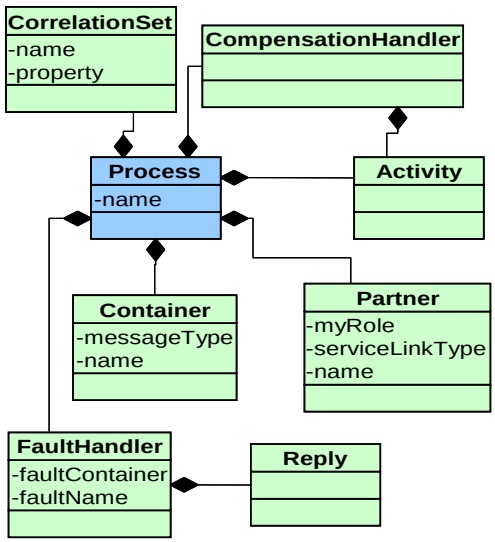

Figure III.38. Extrait du méta-modèle BPEL4WS

III.8.4.5. Autres standards

Il existe d'autres standards liés aux Services Web et qui sont habituellement utilisés dans le domaine industriel pour fournir des mécanismes divers liées à la gestion et/ou l'exécution des Services Web. Sans toutefois être exhaustif, on peut principalement citer [W3C-WSA 2006] :

- WS-Coordination [IBM et al. 2005] : qui permet de prendre en charge la coordination de services;
- WS-Transaction [Serviceoriented 2003] : qui permettent de gérer les transactions distribuées;
- WS-Reliability [Evans 2003] : qui permet de prendre en charge la fiabilité des échanges SOAP;
- WS-Security [Oasis 2004]: qui s'intéresse aux aspects de la sécurité des Services Web.

La figure III.39 récapitule à travers un méta-modèle simplifié les principales interactions qui peuvent exister entre les principaux standards lies aux Services et qui sont actuellement utilisés dans l'industrie.

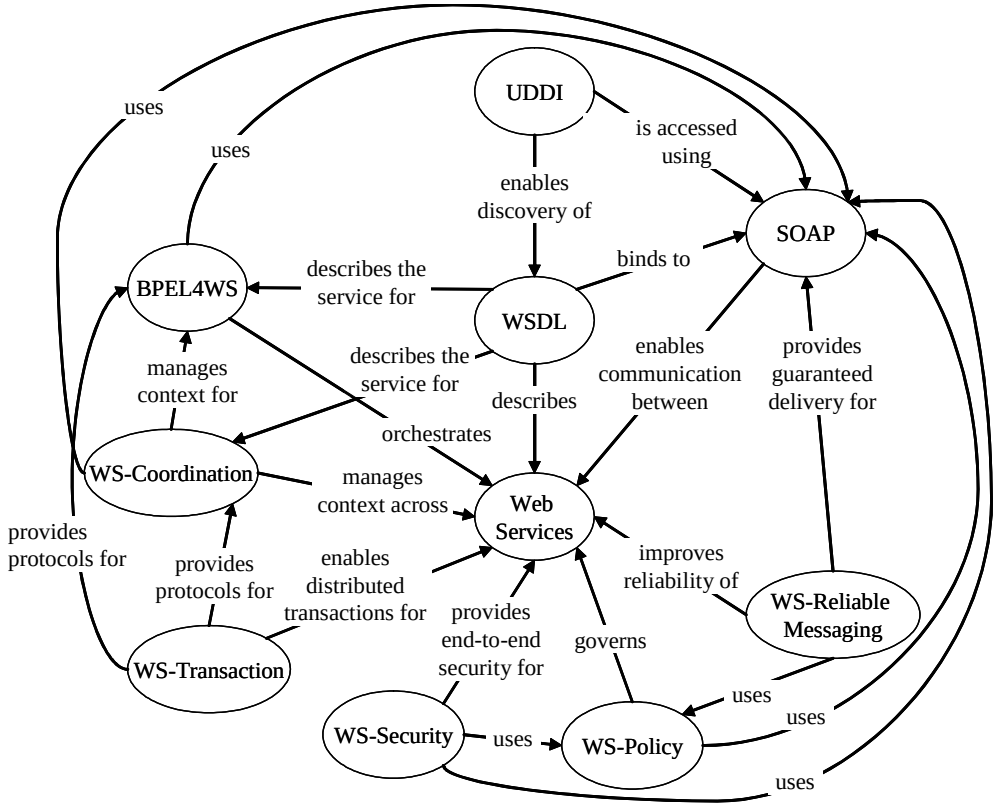

Figure III.39. Principaux standards des Services Web (Adapté de [Erl 2004])

III.8.5. Bus de services d'entreprise (ESB)

L'ESB (Enterprise Service Bus) ou Bus de Services d'Entreprise [Chappell 2005] est un concept qui a été défini en 2003 par le Gartner Group. Il est considéré comme la convergence des outils EAI et des Services Web mais en évitant l'aspect monolithique des EAI. L'ESB est une solution d'intégration distribuée basée sur un message-oriented middleware (MOM) et fournissant des services de transformation des données, de routage et ce en s'appuyant sur l'utilisation systématique des standards des Services Web (SOAP, WSDL, UDDI) et les normes WS-*.

Comme illustré par la Figure III.40, l'ESB est centré sur la notion de bus qui permet d'implémenter une solution d'intégration distribuée. Le bus permet aux différents services métiers (encapsulant des application d'entreprise) de communiquer et de coopérer. Le bus se base sur certains de services de base comme [Lublinsky 2003]:

- le moteur d'orchestration qui joue le rôle de service d'orchestration permet d'exécuter des processus basés sur BPEL4WS;
- le service de localisation qui permet de trouver de façon transparente les services;
- les services utilitaires qui sont des services techniques et qui sont sollicités par les services métier (ex: services permettant le routage, la transformation, le support des transaction, etc.);
- les services d'infrastructure qui sont des services qui permettent de fournir un support système et d'infrastructure aux services métiers (ex: services liés à la sécurité et au monitoring, etc.).

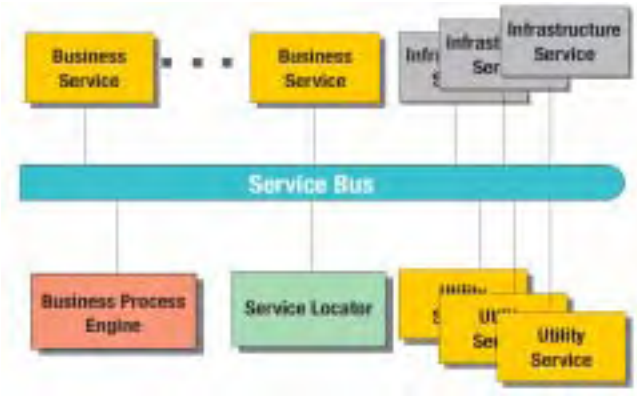

Figure III.40. Architecture d'un ESB [Lublinsky 2003]

III.9. Ingénierie à base de modèles (MDA)

Cette section étudie la contribution à l'intégration de l'approche basée sur les modèles ou approche MDA (Model-Driven Architecture), qui est une nouvelle orientation de développement et d'intégration proposée par l'OMG (Object Management Group) [Bézivin et Blanc 2002][Bezivin et al. 2003].

III.9.1. Définition

Le MDA [OMG-MDA 2006] est une approche proposée par l'OMG en 2000 comme une réponse à l'hétérogénéité des technologies utilisées dans le cadre du génie logiciel. En effet, pour faire face aux multiples technologies existantes et à venir il devient nécessaire de conserver dans l'entreprise des compétences diverses et variées pour des raisons de maintenance et d'intégration des applications. La solution que le MDA propose pour surmonter ces problèmes consiste à utiliser des modèles dans toutes les étapes du processus de développement et d'interaction des applications d'entreprise. Précisément, le MDA propose de se concentrer sur l'élaboration de modèles. Elle propose de séparer les spécifications fonctionnelles d'un système de spécifications de son implémentation sur une plateforme donnée, permettant ainsi définir une architecture indépendante pour pouvoir ensuite la projeter vers différents modèles de plateformes utilisables dans le multiples secteurs de l'industrie.

La figure III.41 représente les différentes couches de spécification du MDA. Au coeur se trouvent les techniques de base (qui sont notamment UML, MOF[18] et CWM[19]), autour quelques-unes des plates-formes supportées (CORBA, Web Services, Java, XML, etc.), en surface les services systèmes offerts (gestion des événements, des transactions, de la sécurité, etc.) et enfin à l'extérieur les domaines (manufacturing, finance, e-commerce, etc.) pour lesquels des composants métiers doivent être définis (Domain Facilities).

[18] MOF (Meta Object Facilities) [OMG-MOF] fournit le standard de méta-modélisation et d'échange de constructions utilisées par MDA. Le MOF est un exemple typique de méta-méta-modèle. Il définit les éléments essentiels, la syntaxe et la structure des méta-modèles utilisés pour construire des modèles orientés objet.

[19] CWM (Common Warehouse Metamodel) [OMG-CWM] est le standard de l'OMG pour les techniques liées aux entrepôts de données. Il définit un méta-modèle qui représente les méta-données aussi bien métiers que techniques qui sont le plus souvent trouvées dans les entrepôts de données.

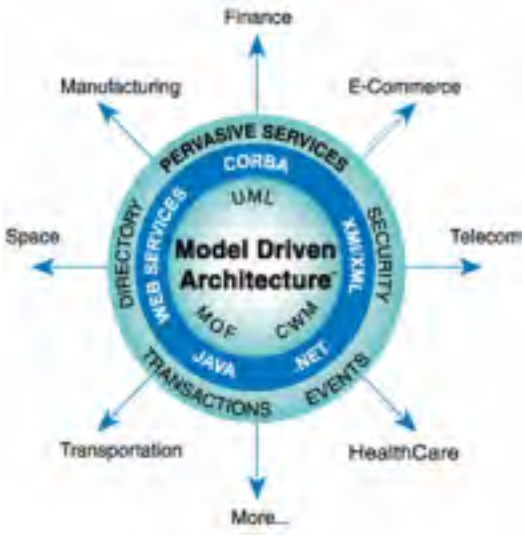

Figure III.41. Le Model-Driven Architecture

III.9.2. Principe

L'approche MDA définit plusieurs niveaux d'architecture. La version initiale de référence définit un modèle indépendant des plateformes qui après raffinements successifs se projette dans un modèle spécifique à une plateforme. La projection se base sur des techniques de transformation paramétriques des modèles métier vers des plates-formes techniques concrètes.

La figure III.42 résume le principe du processus MDA où deux principaux types de modèles sont utilisés :

- les PIM (Platform Independent Model) qui sont indépendants de toute plate-forme (modèles métiers);
- les PSM (Platform Specific Model) qui sont dépendants d'une plate-forme spécifique (modèles techniques spécifiques).

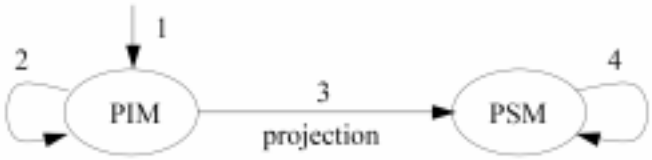

Figure III.42. Processus MDA

III.9.3. Intégration à base de modèles

Le MDA introduit un niveau supérieur en matière d'intégration: l'intégration des applications sur base de modèles. Ceci peut permettre une intégration plus large permettant de ne pas être lié à une plate-forme spécifique. Le principe d'utilisation du MDA dans le cadre de l'intégration d'applications d'entreprise peut être effectué en exprimant leurs relations au niveau PIM. Au moment de la génération des PSM, des modèles de connexions sont générés ce qui permet d'obtenir une implantation des ponts d'interconnexion au moment de la génération de code (figure III.43).

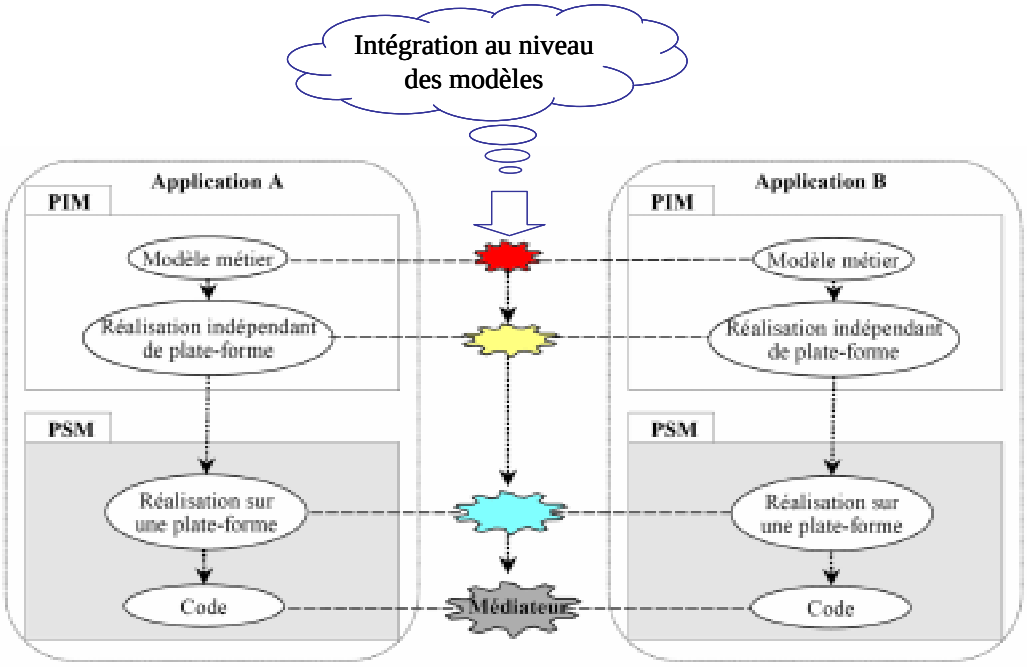

Figure III.43. Principe de l'intégration à base de modèles

L'idée proposée par le MDA est simple, intuitive et séduisante. Cependant, pour que cela devienne réaliste, il devient indispensable de créer des moyens permettant de supporter le processus MDA dont notamment des outils pour :

- Représenter des modèles PIM avec suffisamment de détails pour être implantés dans une plate-forme particulière;
- Représenter des modèles PSM;
- et surtout de transformer automatiquement des modèles PIM vers PSM et des modèles PSM vers le code.

De nombreux travaux importants dans le domaine MDA existent. Ce sont notamment les travaux portant sur les transformations de modèles, dont XSLT (Extensible Stylesheet Language Transformation) [W3C-XSLT 1999], MTRANS [Peltier et al. 2001] et les transformations relationnelles [Akehurst et Kent 2002]. Cependant, à l'heure actuelle, ces approches ne sont pas totalement opérationnelles pour qu'elles puissent être intégrées dans les milieux industriels.

III.10. Discussions

Nous avons voulu résumer, dans le tableau III.2, les principales caractéristiques des principales technologies permettant l'intégration syntaxique présentées précédemment. Les caractéristiques de ces techniques sont synthétisées à l'aide de neuf critères que nous avons choisis de retenir qui sont, à notre sens, les plus pertinents. Il s'agit essentiellement des critères étudiés au chapitre II, qui correspondent soit aux caractéristiques des applications, soit aux dimensions des approches d'intégration et qui sont :

- le critère *périmètre* typique qui précise le champ de mise en œuvre principale de la technique (A2A : intra-entreprise, B2B : inter-entreprise ou B2C : intégration du client);

- critère *point de vue* qui permet de préciser le niveau d'abstraction (conceptuel, technique) associé à la technique considérée;
- le critère *couche* d'intégration qui représente le concept (données, fonction, interface, processus) sur laquelle porte l'intégration;
- le critère *niveau* qui précise le niveau d'intégration où l'on peut réaliser l'intégration;
- le critère *autonomie* qui permet de définir le degré de couplage (faible, fort) entre les applicatifs intégrés;
- le critère *distribution* qui permet de définir le degré de support de la répartition des applications;
- le *dynamisme* qui permet de définir le degré d'aptitude de la technique à prendre en compte des composants applicatifs dynamiques;
- le critère *ouverture* qui permet de représenter le degré de standardisation permis par la technique;
- le critère *flexibilité* qui permet de représenter la capacité de la technique à faire face à de nouveaux changements ou à des scénarios d'évolution.

Critères d'analyse	Techniques d'intégration syntaxique						
	Ad hoc	Standard	Intergiciel	EAI	BPM	SOA	MDA
Périmètre	A2A	A2A B2B	A2A	A2A	A2A	A2A B2B	A2A
Point de vue	technique	conceptuel	technique	technique	conceptuel	conceptuel	conceptuel
Couche	données fonctions	données processus	données fonctions	données	processus	functions processus	modèle
Niveau	syntaxique	syntaxique	syntaxique	syntaxique	syntaxique	syntaxique	syntaxique sémantique
Autonomie	non	oui	(selon type)	oui	oui	oui	oui
Distribution	non	oui	oui	oui	oui	oui	oui
Dynamisme	non	non	non	non	oui	oui	oui
Ouverture	non	oui	non	non	non	oui	oui
Flexibilité	non	oui	non	non	non	oui	oui

Tableau III.2. Principales techniques d'intégration syntaxique

L'analyse du tableau III.2 permet de dégager principalement deux solutions émergentes, qui sont complémentaires et qui peuvent être mises en œuvre dans le cadre de l'intégration des systèmes d'information industriels. Il s'agit des architectures orientées services (SOA) et de l'approche basée sur l'ingénierie des modèles (MDA). Ces deux

approches prises conjointement peuvent favoriser considérablement la flexibilité du système d'information tout en apportant une solution au problème de l'intégration des systèmes hétérogènes dans les milieux industriels. Mais comme on peut le constater dans le tableau III.2, ces solutions sont essentiellement syntaxiques du fait qu'elles n'adressent l'aspect sémantique que de façon implicite. Il en résulte que la sémantique de ces systèmes d'information est très souvent embarquée de façon câblée au sein des programmes et/ou elle est présente mais de façon informelle dans les mémoires individuelles des acteurs de l'organisation. Ce qui peut expliquer la présence des multiples conflits d'ordre sémantique qui peuvent exister et qui doivent être adressés afin d'assurer une bonne intégration entre les systèmes d'information industriels.

III.11. Conclusion

Dans ce chapitre, nous avons présenté les principales techniques permettant l'intégration syntaxique qui peuvent être mises en œuvre dans le cadre de l'intégration des systèmes d'information industriels, et en particulier dans le cadre de systèmes d'information microélectroniques. Nous avons choisi de classifier ces techniques en six grandes catégories qui sont : les techniques *ad hoc* basées sur la conversion câblée de syntaxes, les techniques basées sur la *standardisation* des modèles et des formats d'échange, les techniques basées sur l'utilisation des *intergiciels*, les techniques basées sur les outils *EAI*, les techniques basées sur la *gestion des processus*, les techniques basées sur les *architectures orientées services* et enfin les techniques basées sur l'*ingénierie des modèles*.

L'analyse de ces techniques révèle deux points essentiels. Le premier point porte sur le fait que les architectures orientées services et les techniques basées sur l'ingénierie des modèles constituent les meilleurs paradigmes qui peuvent apporter le maximum de flexibilité aux systèmes d'information industriels. Le second point concerne un certain nombre de limites qui peuvent freiner considérablement la capacité d'intégration des applications dans les milieux industriels. Nous avons, en particulier, retenu le manque de prise en charge de l'aspect sémantique comme étant un des éléments critiques qui caractérisent les techniques syntaxiques actuelles.

Pour remédier aux limites des techniques syntaxiques, et dans le but également de favoriser la flexibilité des systèmes d'information, des solutions basées sur la sémantique ont été introduites dans certains milieux industriels. Le chapitre suivant a pour objet de présenter un panorama de ces techniques, tout en se focalisant principalement sur celles qui prônent l'orientation services, qui est à notre sens, l'un des meilleurs paradigmes pouvant offrir le maximum de flexibilité aux systèmes d'information industriels.

Chapitre IV

INTEGRATION SEMANTIQUE DES SYSTEMES D'INFORMATION INDUSTRIELS

IV.1. Introduction

Dans le chapitre précédent, nous avons présenté un panorama des principales techniques qui peuvent être mises en œuvre pour l'intégration syntaxique des systèmes d'information industriels. Nous avons en particulier insisté sur l'intérêt de l'approche orientée services (SOA) et l'approche basée sur les modèles (MDA) qui peuvent considérablement favoriser la flexibilité des systèmes. De plus, nous avons retenu le fait que ces approches présentent de sérieuses limites en ce qui concerne la prise en charge de l'aspect sémantique des systèmes d'information.

Dans ce chapitre, on étudiera le niveau sémantique de l'intégration des systèmes d'information. L'intégration sémantique consiste à faire interopérer et a coopérer des systèmes au niveau conceptuel en fournissant des mécanismes permettant de comparer, reconnaître les similitudes et les différences entre les différents concepts manipulés. On analysera les principales techniques utilisées pour intégrer sémantiquement des applications industrielles hétérogènes. En particulier, on s'intéressera aux techniques sémantiques qui se basent à la fois sur les ontologies (qui s'inscrivent dans le prolongement naturel du MDA) et l'orientation services (SOA).

Dans ce qui suit, on présentera la notion de sémantique et la notion d'ontologie qui est utilisée comme l’un des moyens les plus efficaces pour la représentation formelle de la sémantique. Ensuite, on exposera de façon succincte les principaux travaux sur les architectures et l'intégration des ontologies. Puis on présentera les différentes approches d'intégration sémantique des systèmes d'information, où l'on s'attardera en particulier sur l'intégration sémantique des services. Enfin on terminera par une synthèse des principales caractéristiques et également des principales limites des approches analysées.

IV.2. Notion de sémantique

IV.2.1. Définitions

A l'origine le terme *sémantique* vient du grec *semantikos* ("qui signifie"). De nos jours, la sémantique constitue une branche de la linguistique qui s'occupe de l'étude des sens des termes. La sémantique est généralement utilisée pour compléter la syntaxe qui constitue un élément nécessaire pour pouvoir parler de sémantique[20]. La syntaxe est à l'origine une branche de la linguistique qui étudie la façon dont les mots (morphèmes libres) se combinent pour former des syntagmes (nominaux ou verbaux) pouvant mener à des propositions (phrases), lesquelles peuvent se combiner à leur tour pour former des agrégats plus complexes, - les énoncés (textes). Il y a généralement entre la sémantique et la syntaxe le même rapport qui existe entre le fond et la forme.

Dans le cadre de nos travaux, nous utilisons les notions de sémantique et de syntaxe du point de vue informatique, où les définitions restent pratiquement similaires à celles pratiquées en linguistique, et où plus précisément la syntaxe se réfère au respect de la grammaire formelle d'un langage, alors que la sémantique concerne plutôt l'interprétation des modèles et des symboles informatiques. Plus précisément, la sémantique a trouvé son essor dans le domaine informatique avec l'évolution du web au web sémantique [Berners-Lee et al. 2001] où on la considère "comme une forme formelle de représentation des connaissances humaines" [Woods 1975].

Dans le domaine des systèmes d'information, la sémantique réfère plus précisément au sens des différents éléments d'un Système d'Information qui peuvent être des données, des fonctions, voire des processus. La difficulté liée à la sémantique des éléments d'un système d'information tient au fait qu'elle peut différer d'un système à un autre, en fonction du contexte d'utilisation. La sémantique constitue un aspect fondamental de l'intégration des applications d'entreprise. Pour assurer correctement cette intégration, il est nécessaire que les différentes applications utilisées puissent interpréter de la même manière les données échangées et/ou les fonctions réutilisées.

Les conflits sémantiques surviennent lorsque, par exemple, l'information change de sens en fonction du contexte. Dans le cadre de conflits sémantiques, et plus précisément de conflits sémantiques de données, [Goh 1997] identifie trois catégories principales d'hétérogénéités sémantiques:

- *des conflits de confusion :* où la donnée apparaît avoir le même sens alors qu'il n'en est rien;
- *des conflits de mesure :* quand différents systèmes mesurent la même valeur;
- *et des conflits de nomination* **:** quand les noms de schémas diffèrent significativement du fait de la présence d'homonymes ou de synonymes.

Ces conflits sémantiques ne concernent pas seulement les données d'une application, mais peuvent aussi concerner la logique métier des applications et qui est aussi appelée hétérogénéité d'invocation selon [Bussler 2003]. [Kavouras 2003] précise que l'hétérogénéité sémantique est plus complexe que les autres types d'hétérogénéité - l'hétérogénéité syntaxique et technique. Et il propose plusieurs causes qui peuvent

[20] On rejoint dans ce sens la théorie de l'information qui distingue fondamentalement trois niveaux pour la représentation du monde : la syntaxe, la sémantique et la pragmatique. Les travaux présentés dans ce document concernent fondamentalement les niveaux syntaxique et sémantique.

générer ces hétérogénéités sémantiques et qu'il résume dans deux points, à savoir différence de couverture (niveau de détail) et différence de classification (conceptualisation).

De plus, en considérant que les entités de la réalité sont caractérisées par un terme T_i et une définition D_i, il propose une typologie d'hétérogénéités sémantiques découlant du croisement de situations portant d'une part sur les situations existant entre les termes (T_1, T_2) et d'autre part sur les situations existant entre les définitions (D_1, D_2) associées à entités décrites (figure IV.1).

Les combinaisons les plus triviales illustrées par ce tableau sont la notion d'équivalence (même terme et même définition), et la notion de disjonction (termes différents et définitions différentes). La synonymie correspond au cas où deux entités sont représentées par deux termes différents et une même définition. Quant à l'homonymie, elle correspond à la situation dans laquelle deux entités sont représentées par le même terme et deux définitions différentes. Enfin, le chevauchement correspond à des situations dans lesquelles les définitions des deux entités se recouvrent.

	$T_1 = T_2$	$T_1 \neq T_2$
$D_1 = D_2$	Equivalence	Synonymie
$D_1 > D_2$	Complétude	Spécialisation
$D_1 \sim D_2$	Chevauchement	Chevauchement
$D_1 \neq D_2$	Homonymie	Disjonction

Légende:
- pas de conflict
- faible conflict
- moyen conflict
- grand conflict

Figure IV.1. Typologie des conflits sémantiques (adapté de [Kavouras 2003])

Dans le cadre des applications industrielles, la résolution des hétérogénéités sémantiques est fondamentale. Elle nécessite la mise en place de mécanismes de médiation. Ces mécanismes doivent couvrir l'ensemble des couches d'intégration dans le sens où ils doivent prendre en charge à la fois la médiation de données, la médiation fonctionnelle et la médiation de processus.

IV.2.2. Le continuum sémantique

La notion de sémantique peut être mieux appréhendée dans le cadre de l'intégration des systèmes d'information industriels à travers le *continuum sémantique* proposé par [Uschold et Gruninger 2002] et qui permet de classer les différents types de sémantique qui peuvent exister dans le contexte du Web. Aussi, on peut distinguer :

- La *sémantique implicite* : qui existe seulement dans le mental des gens;
- La *sémantique semi-informelle* (explicite et informelle) : qui est une sémantique explicite mais qui est très souvent représentée de façon informelle en utilisant généralement des langages naturels tels que le français ou l'anglais;
- La *sémantique semi-formelle* : qui désigne une sémantique explicite et relativement formelle qui est destinée principalement aux humains en utilisant des formalismes

le plus souvent graphiques tels que les modèles sémantiques, ou les diagrammes UML;

- La *sémantique formelle* : qui est une sémantique qui se base sur des formalismes mathématiques rigoureux (tels que la logique de description, la logique de premier ordre, etc.) qui lui permettent d'être traitée de façon automatique.

Ces types de sémantique ne sont généralement pas exclusifs. Aussi, on peut trouver au sein d'une entreprise la présence simultanée d'un ou de plusieurs de ces types de sémantique.

IV.2.3. Représentation de la sémantique

La représentation de la sémantique est nécessaire pour l'intégration des systèmes. Pour cela on utilise des techniques de représentation de la connaissance. La représentation des connaissances étudie comment transformer l'expression du sens en une représentation formelle manipulable par une machine. Il existe principalement deux approches pour la modélisation de la sémantique : la *modélisation procédurale* et la *modélisation déclarative*. La modélisation procédurale se fait à travers l'utilisation de procédures ou de règles, tandis que l'approche déclarative se base sur la modélisation des faits.

Même si l'approche procédurale de modélisation de la sémantique est une approche qui présente des performances très intéressantes en matière d'exécution, il n'en demeure pas moins que l'approche déclarative est généralement plus avantageuse en termes de standardisation, de capture, de réutilisation et d'inférence [Singh et Huhns 2005]. De plus, l'approche déclarative est plus flexible dans le sens où la modélisation ne dépend pas de l'ordre de représentation des faits, ce qui implique une modification plus aisée. L'un des outils les plus utilisés dans le cadre de la représentation déclarative de la sémantique est la notion d'ontologie. Cette dernière est introduite ci-après.

IV.3. Notion d'ontologie

Nous assistons, ces dernières années, à l'émergence de la notion d'*ontologie*, qui constitue un sujet de recherche répandu dans diverses communautés notamment celles liées à l'ingénierie des connaissances, à l'ingénierie des systèmes, à l'intégration des systèmes, etc. Ce fort engouement est motivé par le fait que les ontologies peuvent fournir un moyen efficace pour gérer les connaissances partagées et communes à un domaine particulier, tout en permettant des échanges tant au niveau syntaxique que sémantique, entre des personnes et/ou des systèmes hétérogènes. Nous allons présenter dans cette section un état de l'art sur les ontologies. Nous allons successivement présenter la notion d'ontologie, les méthodologies de construction, les langages ainsi que les outils de développement.

IV.3.1. Généralités

IV.3.1.1. Définitions

De nos jours, les *ontologies* constituent le fer de lance de l'ingénierie des connaissances (KE - Knowledge Engineering). Elles désignent une manière de représenter les

connaissances. De son étymologie grecque "*ontologia*", ontologie est issue du domaine de la philosophie et signifie "science de l'être" (*onto*: être et *logia*: science). L'ontologie constitue de ce fait, comme la définit le dictionnaire Robert, la *"partie de la métaphysique qui s'applique à l'être, indépendamment de ses déterminations particulières*" permettant "*une explicitation systématique de l'existence".*

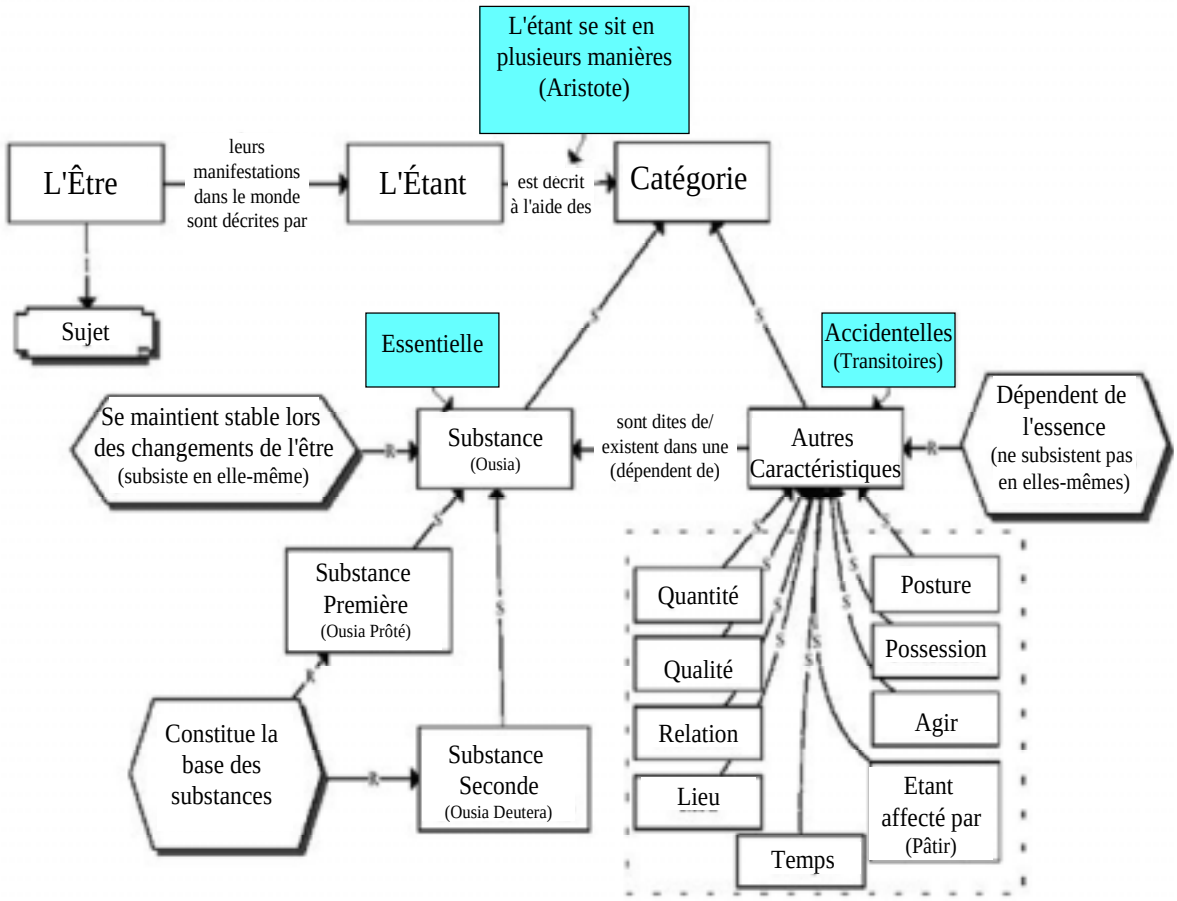

Figure IV.2. L'ontologie de l'être selon Aristote [Psyché et al. 2003]

La figure précédente (figure IV.2) permet d'illustrer l'une des premières ontologies de l'être que l'on peut retrouver en philosophie. Il s'agit de celle d'Aristote qu'il appelle la philosophie première. Cette ontologie permet de représenter l'être ainsi que les différentes situations associées aux multiples manifestations de l'être décrites par un certain nombre de catégories telles que la substance, la quantité, la qualité, etc.

Dans le domaine de l'informatique, il existe une multitude de définitions pour la notion d'ontologie. Cependant, la définition qui nous semble être la plus célèbre et la plus citée est celle de Gruber et qui stipule qu' "*une ontologie est une spécification explicite d'une conceptualisation*[21]*"* [Gruber 1993]. Le terme "conceptualisation" réfère à un modèle abstrait d'un certain phénomène de la réalité et qui permet d'identifier les concepts pertinents de ce phénomène. Le terme "explicite" signifie que le type des concepts utilisés ainsi que les contraintes sur leur emploi sont réellement définies d'une manière claire et précise.

[21] "Ontology is an explicit specification of a conceptualization" [Gruber 1993].

Plusieurs auteurs ont repris et affiné la définition précédente. Borst propose comme amélioration la définition suivante: *"les ontologies se définissent comme une spécification formelle d'une conceptualisation partagée"* [Borst 1997]. Cette définition précise d'une part, le fait que l'ontologie doit être formelle, c'est à dire exprimée sous forme d'une logique pouvant être manipulée sur machine, et d'autre part, le fait qu'elle doit être "partagée" dans la mesure où elle doit référer à la notion de groupe qui impose ainsi la mise en place d'un partage de connaissances entre ses différents individus. En 1995, Guarino et ses collègues ont recueilli plusieurs définitions dans la littérature scientifique, en ont fourni des interprétations sémantiques et ont proposé la définition suivante: *"Une ontologie est une théorie logique qui permet une spécification explicite et partielle d'une conceptualisation"* où le terme conceptualisation désigne l'idée de la réalité qu'un individu ou un groupe d'individus peuvent avoir [Guarino 1995]. D'autres auteurs offrent d'autres définitions fondées sur l'approche qu'ils ont adopté pour construire leurs ontologies. A ce titre, on peut citer par exemple Swartout et ses collègues qui proposent la définition: *"une ontologie est un ensemble de termes structurés de façon hiérarchique, conçu afin de décrire un domaine et qui peut servir de charpente à une base de connaissances"* [Swartout et al. 1997]. On peut aussi citer Gomez-Perez qui propose quant à elle la définition qui stipule qu' "*une ontologie fournit les moyens de décrire de façon explicite la conceptualisation des connaissances représentées dans une base de connaissances"* [Gomez-Perez 2000].

Ces multiples définitions, qui dans leur diversité offrent des points de vue à la fois différents et complémentaires sur un même concept, laissent augurer de quoi seront constituées les ontologies. Actuellement, force est de constater que le terme ontologie est de plus en plus utilisé, tout en référant à des objets assez différents. La figure IV.3 , tirée de [Smith et Welty 2001], permet d'illustrer le large spectre couvert par les artefacts qui ont pu, à un moment donné, être classifiés, à tort ou à raison, comme des ontologies.

Ainsi, des systèmes d'information aussi simples que des catalogues sont considérés par certains auteurs comme des ontologies. Légèrement plus complexes, l'ontologie peut être basée sur un ensemble de textes en langue naturelle sur lesquels on établit des correspondances de chaînes de caractères. Les glossaires, permettant de classer les termes référencés constituent également une ontologie, au même titre que les thésaurus qui peuvent être considérés comme des glossaires plus évolués. Les taxinomies ainsi que les systèmes à base de frames, largement utilisés dans la représentation des connaissances, peuvent être également perçus comme des ontologies. Finalement, les systèmes à base de connaissance utilisant des axiomes de la logique du premier ordre, d'ordre supérieur, la logique modale, la logique des descriptions représentent les ontologies les plus complexes.

Figure IV.3. Spectre couvert par les ontologies [Smith et Welty 2001]

Bien qu'ils soient disparates, les artefacts précédents présentent tout de même quelques points communs. Il s'agit pour l'essentiel de la nécessité de mettre en œuvre une classification partagée des concepts manipulés. Afin de mieux préciser notre perception de la notion d'ontologie, nous reprenons les définitions de [Gruber 1993] avec une légère connotation informatique, et nous considérons une ontologie comme "*un modèle (au sens large, composé d'objets et de relations) réutilisable et partageable d'un domaine particulier (dans notre cas le domaine industriel, et plus précisément le domaine de la microélectronique) spécifié pour créer un langage commun (éventuellement basé sur des standards basés sur XML) afin de faciliter l'échange d'informations et le partage de fonctionnalités entre les personnes et les systèmes d'information (intégration)*".

IV.3.1.2. Un exemple simplifié d'ontologie : l'ontologie de jouet

Pour mieux illustrer les définitions précédentes en ce qui concerne la notion d'ontologie, nous avons voulu présenter une ontologie simplifiée - l'ontologie de jouet [Gandon 2002]. Cette ontologie est celle de la situation bien connue des cubes disposés sur une table.

La figure IV.4 montre une situation décrivant une scène de trois cubes arrangés sur une table. [Gandon 2002] propose un vocabulaire conceptuel (*ontologie de jouet*) pour parler de quelques aspects de cette réalité (certains d'entre eux sont ignorés, par exemple il n'y a aucun vocabulaire pour exprimer les dimensions des cubes). Enfin l'état de la scène observée est décrit en utilisant les primitives de l'ontologie, qui sont dans notre exemple formalisées par les fonctions *Cube()* et *On()*.

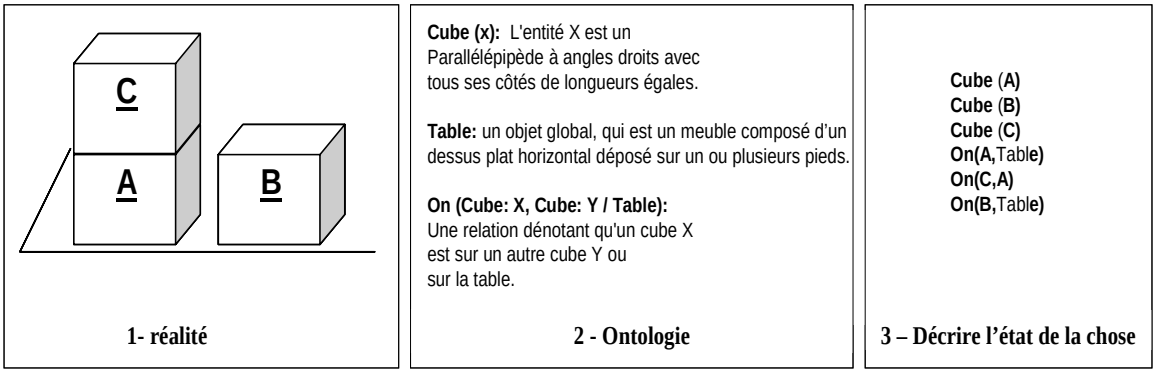

Figure IV.4. L'exemple des cubes [Gandon 2002]

IV.3.2. Structuration des ontologies

Comme tout formalisme de représentation, les ontologies sont basées sur l'utilisation d'un certain nombre de composantes de base dont la notion de concept, de relation et d'axiome.

IV.3.2.1. Constituants de base

Les connaissances traduites par une ontologie sont véhiculées à l'aide d'un certain nombre de constituants ou briques de base et qui sont principalement des: 1) *Concepts*, 2) *Relations*, 3) *Fonctions*, 4) *Axiomes*, 5) *Instances* [Gruber 1993], [Gomez-Perez 2000]. Le détail de ces constituants est comme suit :

- Les *concepts*, aussi appelés termes ou classes[22] de l'ontologie, constituent les objets de base manipulés par les ontologies. Ils correspondent aux abstractions pertinentes du domaine du problème, retenues en fonction des objectifs qu'on se donne et de l'application envisagée pour l'ontologie, par exemple, la description d'un ensemble d'objets, d'une tâche, d'une fonction, d'une stratégie, d'un processus de raisonnement, etc.
- Les *relations* traduisent les interactions existant entre les concepts présents dans le domaine analysé. Ces relations sont formellement définies comme tout sous-ensemble d'un produit cartésien de n ensembles, c'est à dire $R : C_1 \times C_2 \times \ldots \times C_n$ et incluent 1) la relation de spécialisation (subsomption), 2) la relation de composition (méronymie), 3) la relation d'instanciation, etc. Ces relations nous permettent de capturer, la structuration ainsi que l'interaction entre les concepts, ce qui permet de représenter une grande partie de la sémantique de l'ontologie**.**
- Les *fonctions* sont des cas particuliers de relations dans lesquelles le nième élément (extrant) de la relation est défini de manière unique à partir des n-1 éléments précédents (intrants). Formellement, les fonctions sont définies ainsi : $F : C_1 \times C_2 \ldots \times C_{n-1} \rightarrow C_n$. Comme exemple de fonctions binaires, nous pouvons citer la fonction *mère-de*.
- Les *axiomes* permettent de modéliser des assertions toujours vraies, à propos des abstractions du domaine traduites par l'ontologie. Ils permettent de combiner des concepts, des relations et des fonctions pour définir des règles d'inférences et qui peuvent intervenir, par exemple, dans la déduction, la définition des concepts et des relations, ou alors pour restreindre les valeurs des propriétés ou les arguments d'une relation.
- Les *instances* ou *individus* constituent la définition extensionnelle de l'ontologie. Ils représentent des éléments singuliers véhiculant les connaissances (statiques, factuelles) à propos du domaine du problème.

IV.3.2.2. Propriétés des concepts

Les concepts constituent les constituants centraux d'une ontologie. Selon [Gomez-Perez 2000], il est possible de les classifier selon plusieurs dimensions: 1) *niveau d'abstraction* (concrets ou abstraits), 2) *atomicité* (élémentaires ou composés), 3) *niveau de* réalité (réels ou fictifs). De plus, un concept est généralement défini par la donnée de trois parties : un *terme*, une *intention* et une *extension*. Le *terme* (nom) correspond à l'identité du concept. L'*intention* (notion) du concept, contient la sémantique du concept, exprimée en termes de *propriétés* et d'*attributs*, de *règles* et de *contraintes*. L'*extension* du concept regroupe les objets manipulés à travers le concept; ces objets sont appelés instances du concept. L'extension et l'intention permettent respectivement de doter le concept d'une sémantique référentielle (celle imposée par son extension) et d'une sémantique différentielle (celle imposée par son intention) [Bachimont 2000].

Il est possible d'associer aux concepts un certain nombre de propriétés qui peuvent porter aussi bien sur l'extension que sur l'intention et dont on peut citer principalement et sans prétendre à aucune exhaustivité celles présentées dans [Füster 2002] :

- la *généricité* : un concept est générique s'il n'admet pas d'extension. Par exemple, la vérité est un concept générique;

[22] Bien que le terme classe soit plus technique, il est cependant très utilisé dans la littérature pour désigner la notion de concept. Aussi, dans ce qui suit, nous allons, sauf indication contraire, considérer ces deux termes comme équivalents.

- l'*identité* : un concept porte une propriété d'identité si cette propriété permet d'identifier les instances de ce concept. Par exemple, le numéro de l'étudiant constitue une propriété d'identité pour le concept d'étudiant;
- la *rigidité* : un concept est rigide si toute instance de ce concept en reste instance dans tout les mondes possibles. Par exemple, humain est un concept rigide, étudiant est un concept non rigide;
- l'*anti-rigidité* : un concept est anti-rigide si toute instance de ce concept est essentiellement définie par son appartenance à l'extension d'un autre concept. Par exemple, étudiant est un concept anti-rigide car l'étudiant est avant tout un humain
- l'*unité* : un concept est un concept unité si, pour chacune de ses instances, les différentes parties de l'instance sont liées par une relation qui ne lie pas d'autres instances de concepts. Par exemple, les deux parties d'un couteau, manche et lame sont liées par une relation " emmanché " qui ne lie que cette lame et ce manche.

En plus de ces propriétés intrinsèques, nous pouvons aussi citer les propriétés inter-concepts qui portent sur des propriétés extrinsèques aux concepts. Il s'agit principalement de [Garino 1995][Füster 2002]:

- - l'*équivalence* : deux concepts sont équivalents s'ils ont la même extension. Par exemple, le concept salarié est équivalent au concept élève dans le cadre d'une entreprise qui dispense des formations à tous ses salariés;
- - la *disjonction* : deux concepts sont disjoints si leurs extensions sont disjointes. Par exemple, homme et femme sont disjoints dans la mesure où aucune instance de l'un ne peut être à la fois un homme et une femme;
- - la *dépendance* : un concept C1 est dépendant d'un concept C2 si pour toute instance de C1 il existe une instance de C2 qui ne soit ni partie ni constituant de l'instance de C2. Par exemple, parent est un concept dépendant de enfant (et vice-versa).

IV.3.2.3. Propriétés des relations

Tout comme pour les concepts, il existe aussi pour les relations un certain nombre de propriétés, dont principalement on peut distinguer des propriétés intrinsèques, des propriétés inter-relations, et des propriétés liant une relation et des concepts :

- Les *propriétés intrinsèques* à une relation permettent de définir la relation, et on distingue principalement :
  - les *propriétés algébriques* : symétrie, réflexivité, transitivité;
  - la *cardinalité* : nombre de participations possibles d'une instance de concept dans une relation.
- Les *propriétés inter-relations* portant sur plusieurs relations et qui peuvent être:
  - l'*incompatibilité* : deux relations sont incompatibles si elles ne peuvent lier les mêmes instances de concepts. Par exemple, les relations "être rouge" et "être vert" sont incompatibles;
  - l'*inverse* : deux relations binaires sont inverses l'une de l'autre si, quand l'une lie deux instances I1 et I2, l'autre lie I2 et I1. Par exemple, les relations "a pour père" et "a pour enfant"sont inverses l'une de l'autre;
  - l'*exclusivité* : deux relations sont exclusives si, quand l'une lie des instances de concepts, l'autre ne lient pas ces instances, et vice-versa. L'exclusivité entraîne

l'incompatibilité. Par exemple, l'appartenance et la non appartenance sont exclusives.

- Les *propriétés liant une relation et des concepts* et qui sont [Kassel 2002]:
  - le *lien relationnel* : il existe un lien relationnel entre une relation R et deux concepts C1 et C2 si, pour tout couple d'instances des concepts C1 et C2, il existe une relation de type R qui lie les deux instances de C1 et C2. Un lien relationnel peut en outre être contraint par une propriété de cardinalité, ou porter directement sur une instance de concept. Par exemple, il existe un lien relationnel entre les concepts "texte" et "auteur" d'une part et la relation "a pour auteur" d'autre part;
  - la *restriction de relation* : pour tout concept de type C1, et toute relation de type R liant C1, les autres concepts liés par la relation sont d'un type imposé. Par exemple, si la relation "mange" portant sur une "personne" et un "aliment" lie une instance de "végétarien", concept subsumé par "personne", l'instance de "aliment" est forcément instance de "végétaux".

IV.3.2.4. Représentation des connaissances

Tous les constituants d'une ontologie qui sont énumérés ci-dessus peuvent être utilisés pour modéliser certains aspects de la connaissance d'un domaine particulier. Cependant, une des questions le plus souvent soulevée est le choix de type de modélisation à effectuer et qui consiste à décider du choix du type de constituant (c'est à dire le choix d'une propriété, d'un concept, d'une relation, etc.) à utiliser pour modéliser une connaissance particulière du domaine. Des critères semblables à ceux utilisés dans les modèles de bases de données peuvent être utilisés. A titre d'exemple, un critère souvent retenu pour la modélisation consiste à dire qu'il s'agit d'une propriété dès lors que les valeurs prises possibles sont d'un type primitif, alors que les valeurs possibles d'une relation qui sont d'un type complexe correspondent à un concept de l'ontologie.

D'une manière générale, dans la terminologie des systèmes à bases de connaissances, un fait apparaît généralement comme une propriété d'une instance de concept, une règle apparaît plutôt comme une propriété de concept ou d'une relation, et un axiome est généralement intégré dans les ontologies comme des objets liés aux concepts et aux relations.

IV.3.3. Classification des ontologies

Il existe principalement, dans la littérature, une classification d'ontologies selon deux dimensions : le niveau de granularité (généralité) et le niveau de formalité (formalisme de représentation) [Guarino 1998], [Uschold et Gruninger 1996].

IV.3.3.1. Classification selon le niveau de granularité

Selon leur niveau de granularité, les ontologies peuvent être classifiées principalement en quatre catégories [Guarino 1997], [Uschold et Gruninger 1996]: 1) ontologies de haut niveau (supérieure), 2) ontologies de domaine, 3) ontologies de tâches, 4) et ontologies d'application (figure IV.5). Ces différents types d'ontologies sont décrits comme suit :

- Les *ontologies supérieures* (*Upper or Top-level Ontologies*) [Guarino 1999]: ont pour objet l'étude des catégories de choses qui existent dans le monde, comme les concepts de haute abstraction tels que: les entités, les événements, les états, les

processus, les actions, le temps, l'espace, les relations, les propriétés, etc. et qui sont indépendants d'un domaine particulier. Très souvent basées sur la théorie de l'identité, la méréologie (théorie de l'ensemble et de ses parties) et la théorie de la dépendance, ces ontologies intègrent en sus les fondements philosophiques comme étant des principes à suivre pour concevoir l'ontologie de haut niveau, et dont le but ultime est de standardiser la conception des ontologies.

- Les *ontologies de domaine (Domain ontologies)* [Mizoguchi 1995][Van Heijst et al. 1997] : ce sont des ontologies qui sont construites sur un domaine particulier de la connaissance. Elles fournissent le vocabulaire des concepts du domaine de connaissance (par exemple lot, wafer dans le domaine microélectronique) et les relations entre ces derniers, les activités de ce domaine (par exemple analyser, tester, fabriquer) ainsi que les théories et les principes de base de ce domaine. Les ontologies de domaine constituent donc des méta-descriptions d'une représentation de connaissances du domaine. De nombreuses ontologies de domaine existent déjà, telles que MENELAS dans le domaine médical [Zweigenbaum 1994], ENGMATH pour les mathématiques [Gruber et Olsen 1994], TOVE dans le domaine de la gestion des entreprises [Gruber 1995], etc.

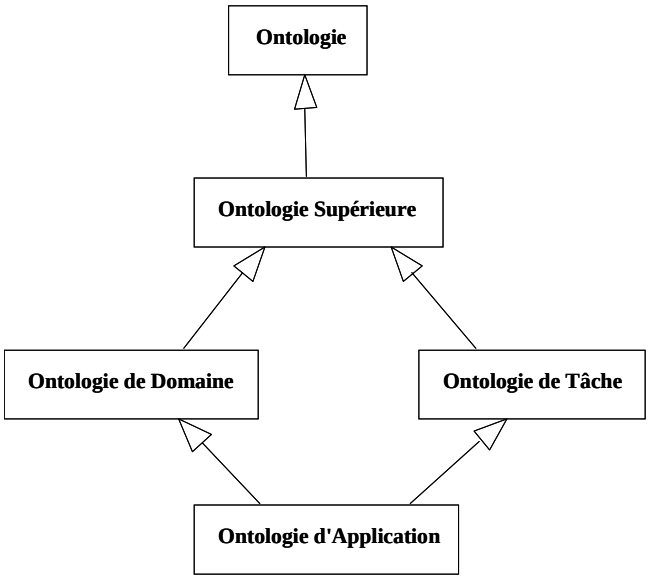

Figure IV.5. Classification des ontologies selon le niveau de granularité[23]

- Les *ontologies de tâches (Task ontologies)* [Mizoguchi 1995] : Ces ontologies sont utilisées pour gérer des tâches spécifiques liées à la résolution de problèmes dans les systèmes, telles que les tâches de diagnostic, de planification, de configuration, etc. Elles fournissent un ensemble de termes au moyen desquels on peut décrire au niveau générique comment résoudre un type de problème. Elles incluent, par exemple, dans le cas de domaines liés à la planification, des noms génériques (par exemple plan, objectif, contrainte), des verbes génériques (par exemple assigner, classer, sélectionner), des adjectifs génériques (comme assigné) et d'autre mots qui relèvent de l'établissement d'échéances, etc.

[23] Signalons que nous nous basons tout au long de ce chapitre et des chapitres suivants sur le formalisme UML pour représenter nos différents modèles et méta-modèles. Aussi, le sens des symboles graphiques utilisés dans ces modèles est celui préconisé dans UML 2.0. En particulier, les flèches désignent la relation de spécialisation/généralisation entre classes, les arcs possédant un losange à l'extrémité désignent des relations d'agrégation entre classes, et les arcs simples désignent les autres types de relations entre classes.

- Les *ontologies d'application* [Van Heijst et al. 1997] : ce sont les ontologies les plus spécifiques. Elles permettent de décrire des concepts dépendants à la fois d'un domaine et d'une tâche. Dans cette classification, la notion d'ontologie d'application définit le *contexte* d'une application qui décrit la sémantique des informations et des services manipulés par une ou un ensemble d'applications sur un même domaine.

IV.3.3.2. Classification selon le niveau de formalité

La classification en fonction du niveau de formalité du langage utilisé pour la représentation de l'ontologie permet de mettre en évidence quatre catégories principales (qui ressemblent quelque peu aux types de sémantique définis dans le continuum sémantique donné en § IV.2.2) qui vont des ontologies hautement informelles jusqu'aux ontologies rigoureusement formelles et qui sont (figure IV.6) [Uschold et Gruninger 1996] :

- *Ontologies informelles* : qui sont des ontologies opérationnalisées dans un langage naturel, ce qui correspond à ce qui est communément appelé la sémantique ouverte;
- *Ontologies semi-informelles* : ce sont des ontologies qui sont décrites à l'aide d'un langage naturel structuré et limité;
- *Ontologies semi-formelles* : qui sont des ontologies qui utilisent un langage artificiel défini formellement;
- *Ontologies formelles* : qui sont des ontologies qui se basent sur l'utilisation d'un langage artificiel contenant une sémantique formelle telle que les logiques de description.

Cette typologie met en évidence plusieurs niveaux de représentation des ontologies. Le processus de développement que nous allons voir plus loin, repose le plus souvent sur les transformations de niveau de représentation. Le principe général du processus de représentation qui repose sur ces niveaux peut donc être découpé en trois 3 phases principales qui sont la conceptualisation, l'ontologisation et l'opérationnalisation et qui permettent respectivement d'obtenir des ontologies informelles ou semi-informelles, des ontologies semi-formelles, et enfin des ontologies formelles.

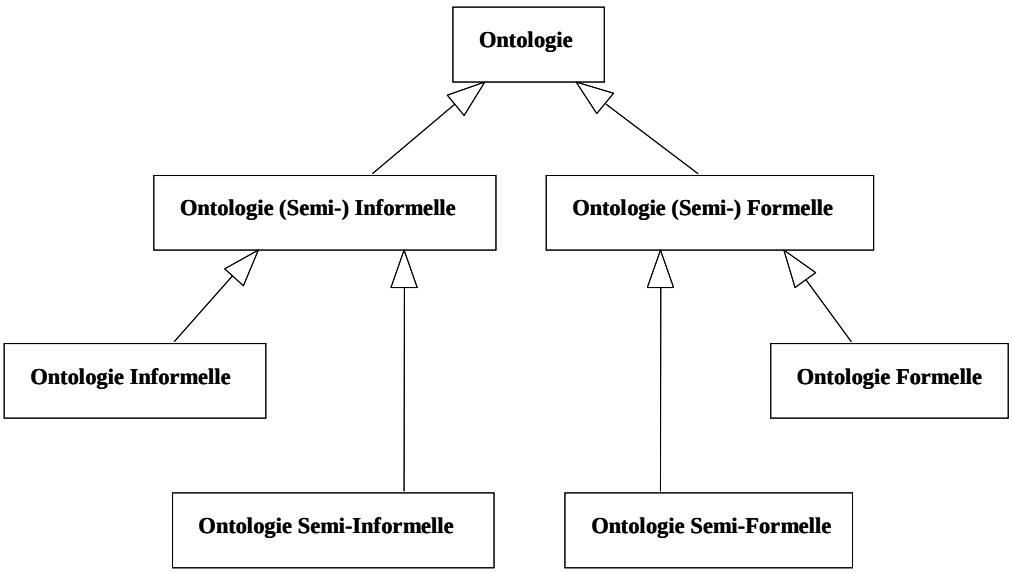

Figure IV.6. Classification des ontologies selon le niveau de formalité

IV.3.4. Construction d'ontologies

La construction d'ontologies peut se révéler complexe, impliquant plusieurs acteurs, diverses connaissances et formalismes. Afin de maîtriser cette complexité, la mise en place d'une méthodologie peut se révéler nécessaire. L'utilisation d'une méthodologie a pour objectif de rationaliser et d'optimiser le processus de développement, dans le sens où il permet de réduire les délais et les coûts ainsi que l'amélioration de la qualité du processus et du produit de développement.

A l'heure actuelle on peut recenser dans la littérature une multitude de méthodologies, un peu plus d'une trentaine selon [Gomez-Perez 2003]. Les auteurs s'accordent à reconnaître qu'il n'existe pas encore de consensus en matière de normes de construction, et par conséquent il n'existe pas encore de méthode universellement reconnue pour la construction d'ontologies. Ceci relève beaucoup plus du savoir-faire que de l'ingénierie. En dépit de cela, de nombreux critères et principes permettant de guider la construction d'ontologies ont été proposés. Certains de ces principes sont résumés dans les prochaines sections.

IV.3.4.1. Cycle de développement ontologique

Du fait que les ontologies peuvent s'apparenter à des composants logiciels, leur développement peut donc s'appuyer sur les mêmes principes que ceux appliqués en génie logiciel. Ainsi, on retrouve pratiquement les mêmes activités liées au développement d'ontologies. Il s'agit d'une part des activités de développement (spécification, formalisation implémentation), et d'autre part des activités de gestion de projet (organisation, planification, estimation, contrôle, assurance-qualité), s'y ajoutent un certain nombre d'activités transversales de support (évaluation, documentation, gestion de la configuration).

EU-UNSF (European Commission - US National Science Foundation) propose un *cycle de vie* pour le développement d'ontologies dans le contexte du Web sémantique. Il est itératif et incrémental et comprend quatre phases: Conception initiale, raffinement conceptuel, évaluation et évolution, avec plusieurs possibilité de feedback (figure IV.7).

Figure IV.7. Cycle de vie ontologique de EU-NSF [EU-NSF 2002]

Un autre cycle de vie semblable au précédent est proposé par [Dieng et al. 2001] (figure IV.8) et il comprend principalement une étape initiale d'évaluation des besoins, une étape de construction, une étape de diffusion, et une étape d'utilisation. Après chaque utilisation significative, l'ontologie et les besoins sont réévalués et l'ontologie peut être étendue et, si nécessaire, en partie reconstruite.

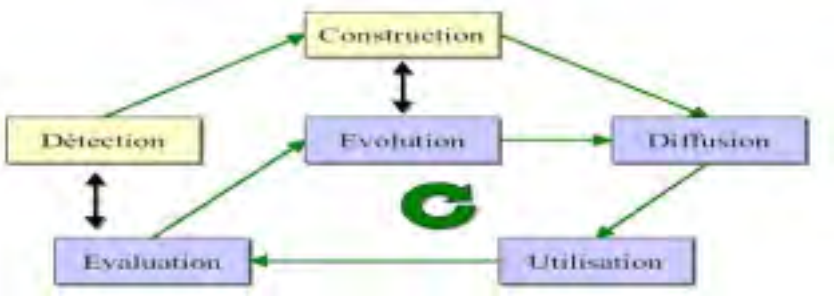

Figure IV.8. Cycle de vie global d'une ontologie [Dieng et al. 2001]

Un autre cycle de vie qui nous paraît intéressant, est celui proposé par l'équipe de Fernandez [Fernandez et al. 1997]. Il est fortement inspiré du cycle en spirale utilisé en génie logiciel, et permet de développer des ontologies d'une manière prototypique (figure IV.9).

Figure IV.9. Cycle de vie de Fernandez & al [Fernandez et al. 1997]

Une fusion de ces deux derniers cycles de vie a été proposée dans [Gandon 2002] dans le cadre du projet CoMMA mené par l'équipe ACACIA de l'INRIA. Ce cycle de vie propose, comme le montre la figure IV.10, une meilleure articulation entre les différentes activités liées à la construction d'ontologies.

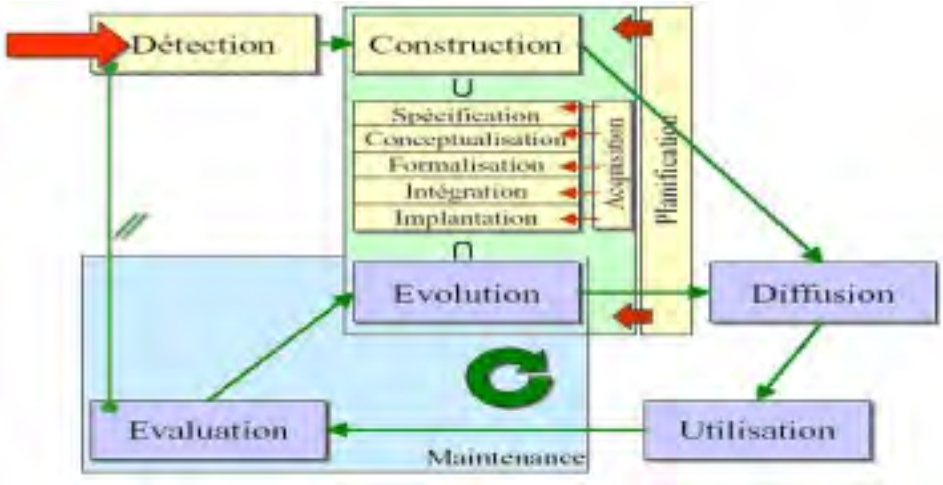

Figure IV.10. Cycle de vie fusionné [Gandon 2002]

Quelle que soit la méthodologie retenue, la principale phase du cycle de vie est celle de construction qui définit le cycle de développement d'une ontologie et permet de définir les différentes étapes du processus de représentation des connaissances. Il peut être

découpé en 3 phases principales: la spécification, la conceptualisation, et la formalisation. L'étape de formalisation peut se décliner en deux sous-étapes: l'ontologisation et l'opérationnalisation. Elle peut être complétée d'une part, par une étape d'intégration au cours de laquelle une ou plusieurs ontologies vont être importées dans l'ontologie à construire [Fernandez et al. 1997], et d'autre part par une étape d'implémentation permettant de coder les ontologies dans un environnement technique spécifique.

L'étape de construction peut se résumer alors aux phases suivantes:

- *La spécification :* identification des objectifs et analyse des scénarios de l'ontologie.
- *La conceptualisation :* identification des connaissances contenues dans un corpus représentatif du domaine. Ce travail doit être mené par un expert du domaine, assisté par un ingénieur de la connaissance; Ce travail permet d'aboutir à une modélisation informelle, voire semi-formelle, partiellement cohérente.
- *La formalisation* : qui peut à son tour se décliner en:
  - *L'ontologisation :* formalisation, autant que possible, du modèle conceptuel obtenu à l'étape précédente. Ce travail doit être mené par l'ingénieur de la connaissance, assisté de l'expert du domaine.
  - *L'opérationnalisation :* transcription de l'ontologie dans un langage formel et opérationnel de représentation de connaissances. L'ontologie obtenue est opérationnelle au sens où elle peut inclure des mécanismes de raisonnement. Ce travail doit être mené par l'ingénieur de la connaissance.

Nous devons retenir le fait que le processus de développement n'est fondamentalement pas linéaire et que de nombreux allers-retours sont a priori nécessaires pour bâtir une ontologie opérationnelle adaptée aux besoins du domaine. Il ne s'agit donc pas nécessairement d'un cycle en cascade, mais plutôt d'un cycle en spirale qui est donc itératif et incrémental. Remarquons également que le modèle de construction d'ontologie tel que présenté part des connaissances à représenter, pour aboutir à une représentation formelle. Cependant, retenons également le fait qu'une autre construction est possible, et cette dernière consiste à choisir un modèle opérationnel de représentation, en fonction de l'objectif d'utilisation de l'ontologie, puis à instancier ce modèle avec les connaissances du domaine.

IV.3.4.2. Principes pour la construction d'ontologies

Pour mieux guider la construction d'ontologies, plusieurs auteurs ont proposé quelques principes qui peuvent s'avérer efficaces durant la conception d'ontologies (ontologisation). Nous ne citerons que ceux qui sont à notre sens assez représentatifs, et il s'agit essentiellement des principes proposés par [Gruber 1993] et [Bachimont 2001].

Gruber propose cinq critères génériques permettant de guider le processus d'ontologisation. Il s'agit de [Gruber 1993]:

- *Clarté et objectivité* : l'ontologie devrait fournir des définitions claires et objectives pour les termes, indépendantes de tout choix d'implémentation.
- *Cohérence* : consistance des axiomes afin de pouvoir formuler par la suite des inférences cohérentes.
- *Extensibilité*: c'est à dire la possibilité d'étendre l'ontologie sans modification.
- *Minimalité des postulats d'encodage*: ce qui assure une bonne portabilité.

- *Minimalité de l'engagement ontologique:* c'est-à-dire l'expressivité maximum de chaque terme qui implique la minimalité et la consistance du vocabulaire utilisé.

A ces principes qui sont relativement génériques, nous ajoutons des principes plus opérationnels, qui sont ceux proposés par [Bachimont 2001]. Ces principes qui sont appelés des principes différentiels permettent de guider le processus de construction d'une ontologie, qu'il appelle *ontologie différentielle,* en organisant ses concepts [Bachimont 2001] :

- *Le principe de communauté avec le père ou principe de similarité :* exprime le fait qu'un concept hérite l'intention de son concept père;
- *Le principe de différence avec le père ou principe de différence :* exprime le fait que l'intention d'un concept est différente de celle de son concept père, sinon il n'y aurait pas besoin de définir le concept fils;
- *Le principe de communauté avec la fratrie ou principe de sémantique unique:* exprime le fait qu'une propriété est commune aux concepts frères issus du même concept père mais s'exprime différemment pour chaque frère. Par exemple, les concepts "homme"et "femme"portent la propriété "sexe" héritée de leur concept père "humain", mais cette propriété vaut "masculin"chez "homme"et "féminin"chez "femme";
- *Le principe de différence avec la fratrie ou principe d'opposition:* exprime le fait que les frères doivent tous être incompatibles, sinon il n'y aurait pas besoin de tous les définir.

IV.3.4.3. Méthodologies de construction d'ontologies

Comme nous l'avons déjà signalé, il existe une multitude de méthodologies. On retrouve dans les travaux du groupe OntoWeb [OntoWeb 2002] plusieurs méthodologies classées selon le type de processus de construction: ex-nihilo, ré-ingénierie, construction collaborative. En plus de ces méthodologies de construction, ces travaux citent d'autres types de méthodologies destinés à d'autres tâches comme les méthodologies d'apprentissage, d'évaluation, d'évolution, de fusion, de maintenance, etc.. Dans le cadre de nos travaux, nous allons nous focaliser uniquement sur les méthodologies de construction d'ontologies. Et nous ne citerons que celles qui nous paraissent les plus représentatives, à savoir, les méthodologies d'Uschold et King, Grüninger et Fox, et Methontology [Corcho et al. 2003].

La *méthodologie d'Uschold et King* [OntoWeb 2002], inspirée de l'expérience de ces auteurs dans la construction de plusieurs ontologies pour la modélisation des entreprises, propose les quatre étapes suivantes: 1) Identifier le but et la portée de l'ontologie, 2) la construire, 3) l'évaluer, et 4) la documenter (figure IV.11). Le processus de construction repose sur la capture des connaissances, leur codage, et leur intégration éventuelle à d'autres ontologies. Pour l'identification des concepts, les auteurs proposent trois stratégies: une approche descendante, dans laquelle les concepts les plus abstraits sont définis et qui sont ensuite spécialisés; une approche ascendante, dans laquelle les concepts spécifiques sont identifiés et qui sont ensuite généralisés pour définir d'autres concepts plus génériques; et une approche mixte dans laquelle les concepts les plus importants sont identifiés et auxquels on applique les techniques de spécialisation et/ou de généralisation pour déterminer les autres concepts.

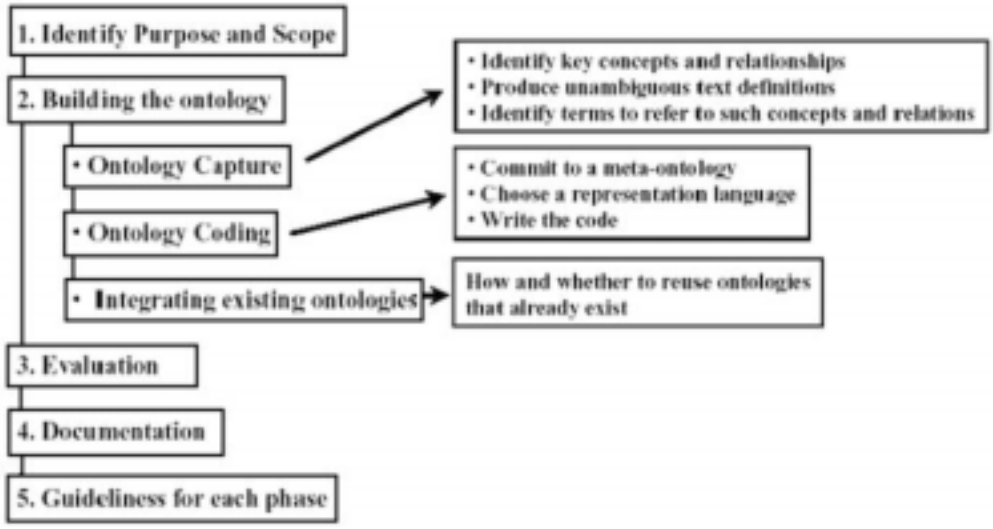

Figure IV.11. La méthode d'Uschold et King [OntoWeb 2002]

La *méthodologie de Grüninger et Fox* [OntoWeb 2002] est inspirée des techniques de développement de systèmes à base de connaissances utilisant la logique de premier ordre. Elle repose essentiellement sur les étapes suivantes (figure IV.12): 1) identifier les principaux scénarios (les applications possibles de l'ontologie), 2) identifier les questions de compétences (questions auxquelles le système projeté est censé pouvoir répondre), 3) extraire ensuite, à partir des résultats précédents, les concepts et les axiomes de l'ontologie. Cette méthodologie, où chaque composante est exprimée en logique de premier ordre, est considérée comme une méthodologie très formelle tirant ainsi profit de la robustesse et la rigueur de la logique mathématique. Elle peut être utilisée en tant que guide de transformation de scénarios informels en modèles formels.

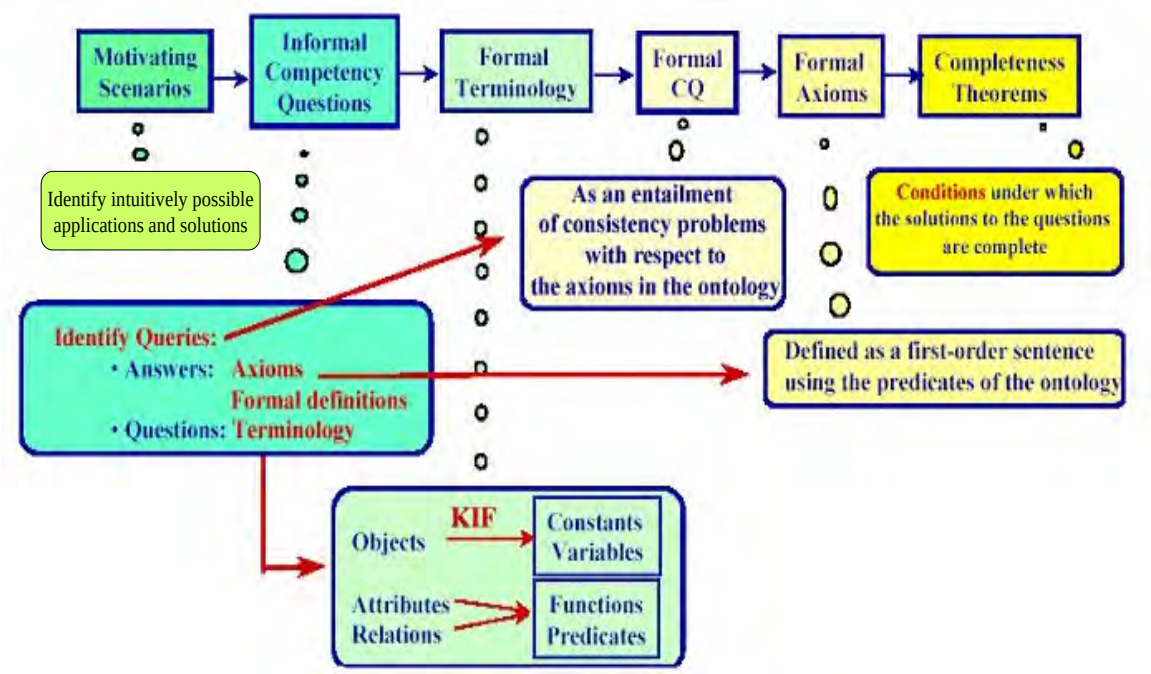

Figure IV.12. La méthodologie de Grüninger et Fox [OntoWeb 2002]

La *méthodologie Methontology*, proposée en 1998 par l'équipe du LAI de l'Université Polytechnique de Madrid, permet de couvrir tout le cycle de vie d'une ontologie. Elle s'intéresse pratiquement à toutes les activités liées aux ontologies, c'est à dire aux activités de développement, de gestion de projet et de support. Le cycle de vie de cette méthodologie est basé sur la notion de prototype évolutif (figure IV.13) et sur des techniques particulières pour la réalisation des activités. Le processus de développement se base sur plusieurs représentations intermédiaires, obtenues le plus souvent par des mappings, permettant ainsi de changer de niveau. La phase principale est la conceptualisation. Cette méthodologie est soutenue par l'outil WebODE et ODE, que nous allons présenter dans les prochains paragraphes.

On trouve dans [Corcho et al. 2003][OntoWeb 2002][WonderWeb 2005] une étude comparative de plusieurs méthodologies, y compris celles présentées précédemment. L'étude révèle que ces méthodologies n'ont pas encore atteint le degré de maturité souhaité, comparé à celui atteint par le génie logiciel. De plus l'étude révèle que Methontology représente la méthode la plus mûre, ce qui l'a sans doute amenée à devenir une recommandation de FIPA en matière de construction d'ontologies.

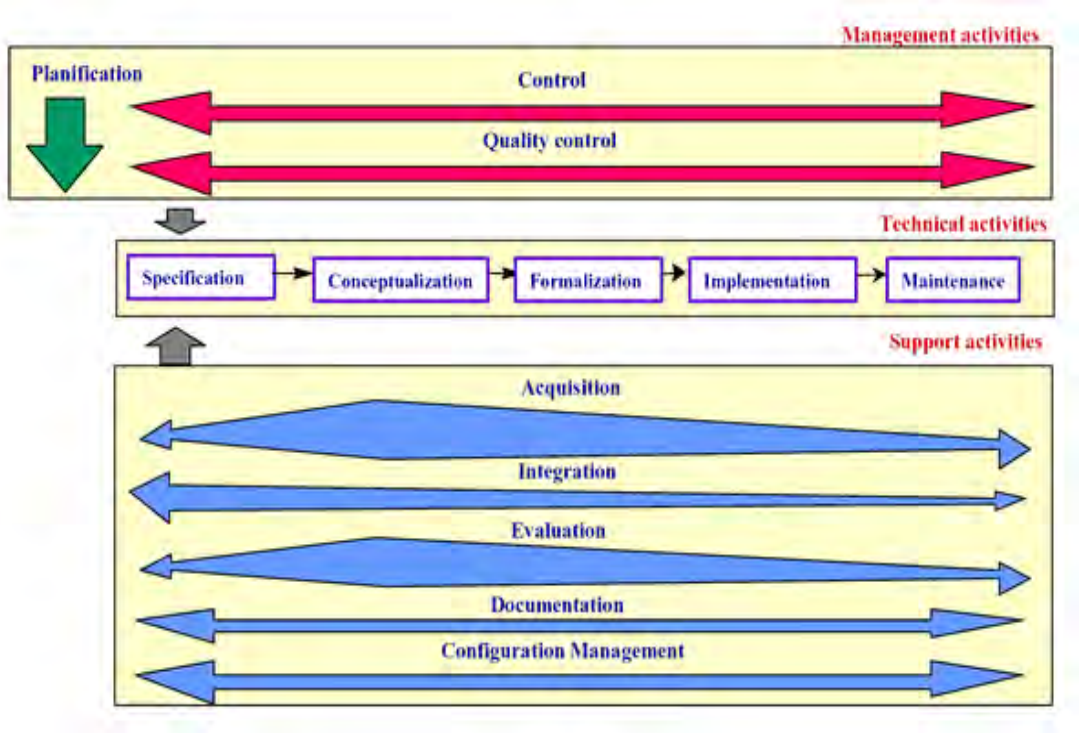

Figure IV.13. Les composantes de Methontology [OntoWeb 2002]

IV.3.4.4. Outils de développement des ontologies

On assiste actuellement à la prolifération d'outils autour des ontologies. Ces outils qui peuvent généralement offrir des fonctionnalités d'édition, de catalogage et/ou de recherche, permettent de systématiser quelque peu l'ingénierie des ontologies. Nous citerons ici que certains outils qui nous semblent les plus significatifs, à savoir Ontolingua, Ontosaurus, ODE, Protégé 2000, Tadzebao, WebOnto, HOZO, KAON, OILEd, et DUET.

Une liste plus exhaustive peut être trouvée dans [Corcho et al. 2003][GómezPérez 2000][Ideas 2003][OntoWeb 2002]. Certains de ces outils ont fait l'objet de plusieurs

études comparatives comme celle proposée par l'équipe WonderTools de l'Université d'Amsterdam [Duineveld et al. 1999] ou encore plus récentes celles issues du projet IST WebOnto [OntoWeb 2002].

Loin de constituer une étude comparative, nous pouvons toutefois reprendre la classification très simplifiée qui consiste à répartir les outils en deux catégories. Dans la première on retrouve les outils les plus anciens historiquement et qui permettent de spécifier les ontologies au niveau symbolique (Ontolingua, Ontosaurus, WebOnto), tandis que dans la seconde catégorie, on retrouve des outils qui ont mûri en abstraction et permettent à l'utilisateur de créer l'ontologie de manière relativement indépendante de toute implémentation (Protégé 2000, KAON, WebODE, etc.). Ces derniers outils sont plus efficaces dans la mesure où ils permettent de s'abstraire de toute contrainte d'implémentation, en permettant notamment de s'affranchir du niveau symbolique des ontologies. Ils peuvent très souvent prendre en charge d'une manière quasi-automatique la phase d'opérationnalisation de l'ontologie, en permettant ainsi de la transposer automatiquement dans divers langages à travers des primitives de mapping ou d'import/export. La figure IV.14 reproduite de [Corcho et al. 2003] illustrant les correspondances entre les outils et les langages, permet de mettre en évidence ces deux catégories d'outils.

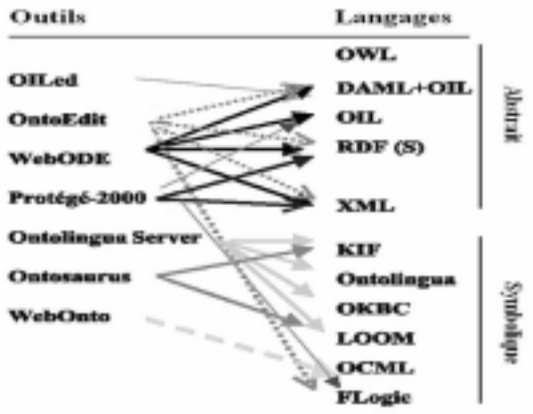

Figure IV.14. Langages, et outils (adapté de [Corcho et al. 2003])

L'outil Ontolingua, de l'Université Stanford, représente le plus ancien et sans doute le plus connu des environnements de construction d'ontologies. Développé au début des années 90, il peut être perçu comme un serveur d'édition d'ontologies au niveau symbolique en utilisant un formalisme de représentation également nommé Ontolingua. Il comprend un ensemble d'outils et de services qui supportent la construction en coopération d'ontologies, entre des groupes séparés géographiquement.

De même que l'outil Ontolingua, l'outil *Ontosaurus*, développé au début des années 90 par l'Information Science Institute de l'Université de South California, consiste en deux modules: un serveur d'ontologies et un navigateur d'ontologies. Le serveur qui se base sur le langage *LOOM* comme langage de représentation des connaissances, utilise des formulaires HTML pour permettre à l'usager d'éditer l'ontologie. Quant au serveur de navigation, il permet de créer dynamiquement des pages HTML qui affichent la hiérarchie de l'ontologie. Des traducteurs d'ontologies LOOM en Ontolingua, KIF, KRSS et C^{++}, ont été développés.

En 1997, les deux outils *Tadzebao* et *WebOnto* ont été développés par le Knowledge Media Institute d'Open University. A l'instar des outils cités précédemment, ces deux outils dépendent du formalisme de représentation. Il s'agit en l'occurrence du langage

OCML. Les deux outils Tadzebao et WebOnto sont complémentaires. Le premier permet aux ingénieurs des connaissances de tenir des discussions sur les ontologies, en mode synchrone et asynchrone, tandis que le second supporte la navigation collaborative, la création et l'édition d'ontologies sur le Web.

Contrairement aux outils ci-dessus (Ontolingua, Ontosaurus, WebOnto) qui appartiennent à la première génération d'outils, les outils de la seconde génération sont plus mûrs et plus ou moins complets. En plus du gain en abstraction qu'ils permettent, les fonctionnalités de ces outils sont améliorées et peuvent s'étendent pour certains afin de couvrir la quasi-totalité du cycle de vie ontologique. Parmi les outils de cette seconde génération, on peut citer Protégé 2000, ODE et OntoEdit.

ODE (Ontology Design Environment), développé au laboratoire d'Intelligence Artificielle de l'Université de Madrid, permet de construire des ontologies au niveau connaissance, comme le préconise la méthodologie Methontology, elle-même proposée par le même laboratoire. La formalisation avec ODE s'effectue avec un langage de frames, tandis que l'opérationnalisation utilise des formalismes de type Ontolingua ou F-Logic. L'outil *WebODE*, également développé au sein du même laboratoire, constitue l'adaptation de ODE pour le Web.

OntoEdit (Ontology Editor) est un environnement de construction d'ontologies indépendant de tout formalisme. Il a été développé par AIFB à Karlsruhe University. Il offre principalement des fonctionnalités d'édition et de navigation d'ontologies. Il inclut des plugins permettant d'inférer en utilisant Ontobroker et aussi d'importer et exporter vers de multiples langages. Il existe en deux versions: une version gratuite et une version professionnelle. Récemment l'outil *KAON* (Karlsuhe Ontology) a été développé comme le successeur d'OntoEdit.

OILEd (OIL Editor), développé sous la responsabilité de l'Information Management Group de l'Université de Manchester, se veut un éditeur freeware d'ontologies, destiné à supporter le développement d'ontologies de petites et moyennes tailles, basées sur le formalisme standard OIL, un des précurseurs de OWL. OILEd n'offre pas des fonctionnalités supportant le cycle complet de conceptualisation et d' opérationnalisation, mais il offre des mécanismes de tests de cohérence grâce à son moteur d'inférence, FaCT, lui-même bâti sur OIL.

L'outil *DUET,* développé par AT&T, permet de visualiser les ontologies en utilisant le formalisme UML (Unified Modeling Language) et de les construire en utilisant le langage DAML+OIL. Cet outil a été intégré comme un plugin dans la suite Rational Rose.

L'outil *Protégé-2000,* développé au département d'Informatique Médicale de l'Université de Stanford, est le dernier de la lignée des outils Protégé (successeur de *Protégé WIN).* C'est un outil de construction d'ontologies qui comprend un éditeur et une librairie de plugins (figure IV.15). L'outil permet : 1) de construire une ontologie du domaine, 2) de personnaliser des formulaires d'acquisition de connaissances et 3) de transférer la connaissance du domaine.

Cependant, nous devons rappeler que, dans l'ensemble, les outils de l'ingénierie ontologique ont mûri considérablement au cours de ces dernières années, mais qu'ils doivent encore s'enrichir de certaines caractéristiques propres à l'ontologie afin d'aboutir à un degré comparable à celui atteint par le génie logiciel. En effet, bien que la plupart de ces outils offrent un support pour l'ontologisation et l'opérationnalisation, très peu d'entre eux offrent par exemple une aide à la phase de conceptualisation. Tel est le cas, par exemple, de Terminae qui propose ainsi, à travers le module Lexter, une

assistance par des mécanismes de traitement du langage naturel opérant sur des corpus de textes.

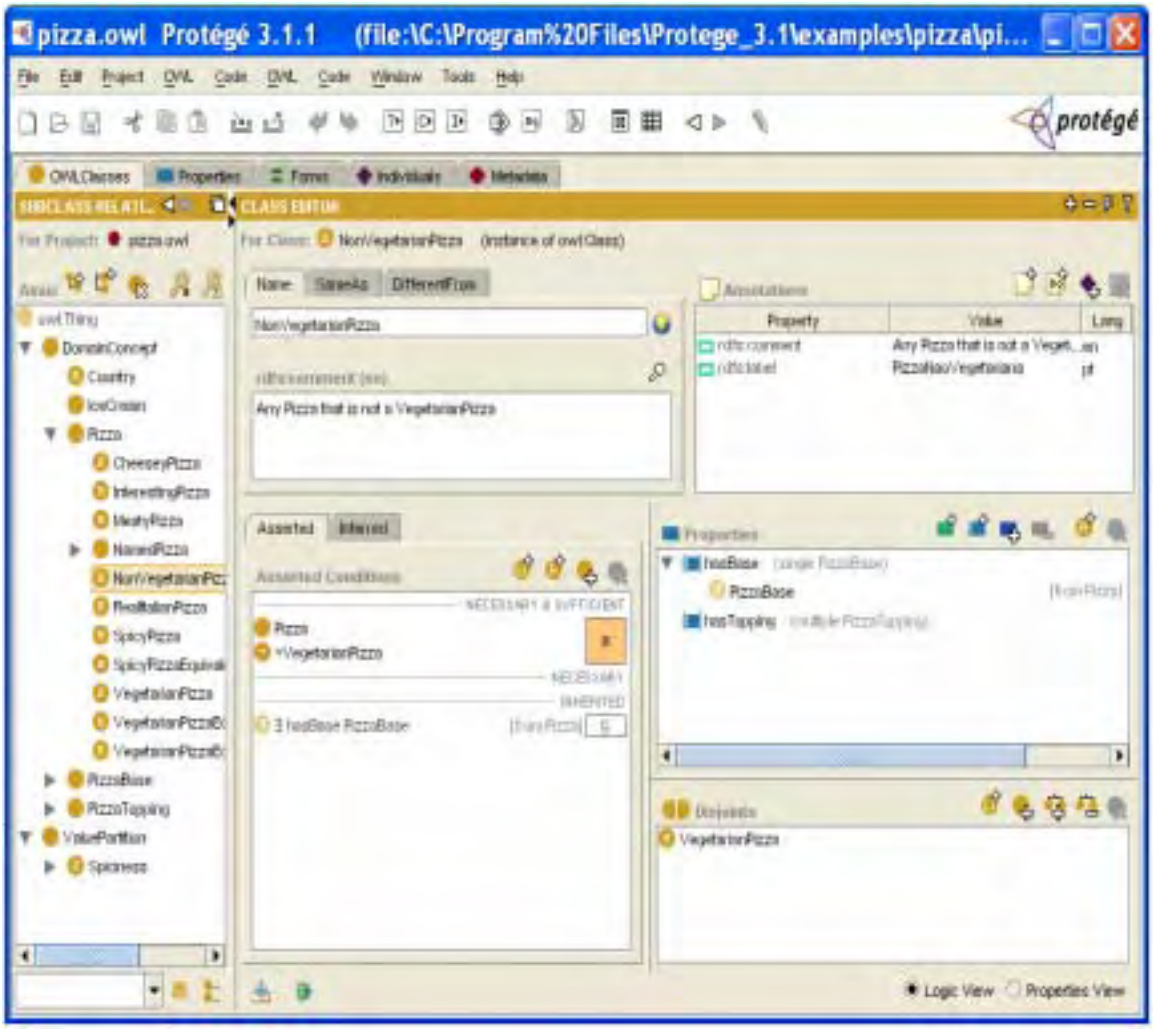

Figure IV.15. L'outil open source Protégé 2000

En plus de ces outils, destinés à la construction d'ontologies, nous devons signaler l'existence d'autres types outils destinés à d'autres activités complémentaires. Ainsi il existe des outils spécialisés dans un certain nombre d'activités liées aux ontologies comme la fusion d'ontologies (Protégé-PROMPT) [Noy et Musen 2003], la traduction d'ontologies d'un langage à un autre (Ontomorph) [Chalupsky 2003], l'annotation de pages Web (COHSE [Wroe et al. 2003], OntoMat [OntoMat 2002], SHOE Knowledge Annotator [Shoe 2001]), l'évaluation d'ontologies (OntoAnalyser [Sure et al. 2002], ONE-T [Gomez-Perez 2001], etc.), l'interrogation d'ontologies (RDF Suite [Alexaki et al. 2001], Sesame [Broekstra et al. 2001], Jena [McBride 2001], Racer [Haarslev et Moller 2003] etc.), etc, [OntoWeb 2002]. Une étude plus complète de ces outils est présentée dans le déliverable D1.3 du projet OntoWeb [OntoWeb 2002].

IV.3.5. Langages de représentation des ontologies

De nos jours, de nombreux langages de représentation d'ontologies existent et ils peuvent intervenir à plusieurs niveaux dans le processus de construction d'ontologies.

Les premiers langages spécifiquement dédiés aux ontologies remontent au début des années 90. Au commencement, un certain nombre de langages basés sur l'intelligence artificielle ont été créés. Il s'agissait pour l'essentiel de langages basés sur la logique de premier ordre (KIF), sur des frames combinées à la logique de premier ordre (Ontololingua, OCML, F-Logic), ou la logique descriptive (LOOM).

Avec l'explosion de l'Internet vers le milieu des années 90, d'autres langages ont été créés pour pouvoir exploiter la richesse du Web. Ce sont les langages basés Web (SHOE), appelés aussi langages d'annotation d'ontologies (Ontology Markup Languages). Certains de ces langages sont basés XML comme XOL, OML, RDF(S), OIL, DAML+ OIL, OWL. Bon nombre de ces langages sont encore à l'heure actuelle en phase de développement et/ou d'évolution, tel est le cas de OWL.

Au delà de ces langages relativement récents, nous pouvons également citer les formalismes de représentation des connaissances, dont certains sont déjà utilisés vers la fin des années 60 et qui sont proposés par l'intelligence artificielle. Ces formalismes peuvent aussi être considérés comme de véritables langages de représentation des ontologies. On peut citer entre autres: les réseaux sémantiques, les langages de frames, les graphes conceptuels et les logiques de description.

Dans ce qui suit, nous allons présenter une synthèse des langages ontologiques en s'inspirant principalement des délivrables des projets européens OntoWeb [Ontoweb 2002] et IDEAS [Ideas 2003].

IV.3.5.1. Langages traditionnels de représentation d'ontologies

Les *réseaux sémantiques*, introduits par Quillian en 1968, sont fondés sur un modèle graphique permettant de combiner la représentation de concepts sous forme de nœuds et de relations sous forme d'arcs, et qui permettent de supporter la notion d'héritage ainsi que des mécanismes de raisonnement fondés sur le parcours du réseau.

Le *langages des frames*, introduits par Minsky en 1975, ont pour but de représenter les connaissances dans un schéma incluant des frames, des slots permettant de décrire ces frames, et des relations d'héritage permettant d'hiérarchiser ces frames.

Au milieu des années 80, deux formalismes inspirés à la fois des réseaux sémantiques et des frames ont été développés. Il s'agit du formalisme des graphes conceptuels et de celui des logiques de description. Chacun de ces formalismes est doté d'une sémantique formelle et dispose d'un ensemble de mécanismes permettant non seulement de représenter les connaissances mais aussi de permettre le raisonnement sur celles-ci.

Les *graphes conceptuels* (GC), introduits par Sowa en 1984, permettent précisément de gérer la connaissance avec des graphes bipartites finis connexes décrivant comment les connaissances d'un domaine sont organisées. Les deux composants de ces graphes en sont les concepts et les relations. Chaque nœud relation doit relier des concepts par des arcs. Chaque concept peut désigner des entités, des propriétés, des états, voire des événements. Les exemples suivants (figure IV.16) permettent de représenter en GC la connaissance exprimée en langue naturelle "Lucien va à Marseille en train"en utilisant respectivement les multiples formes permises par les CG à savoir, DF (Display Form), LF (Linear Form), CGIF (Conceptual Graph Interchange Format).

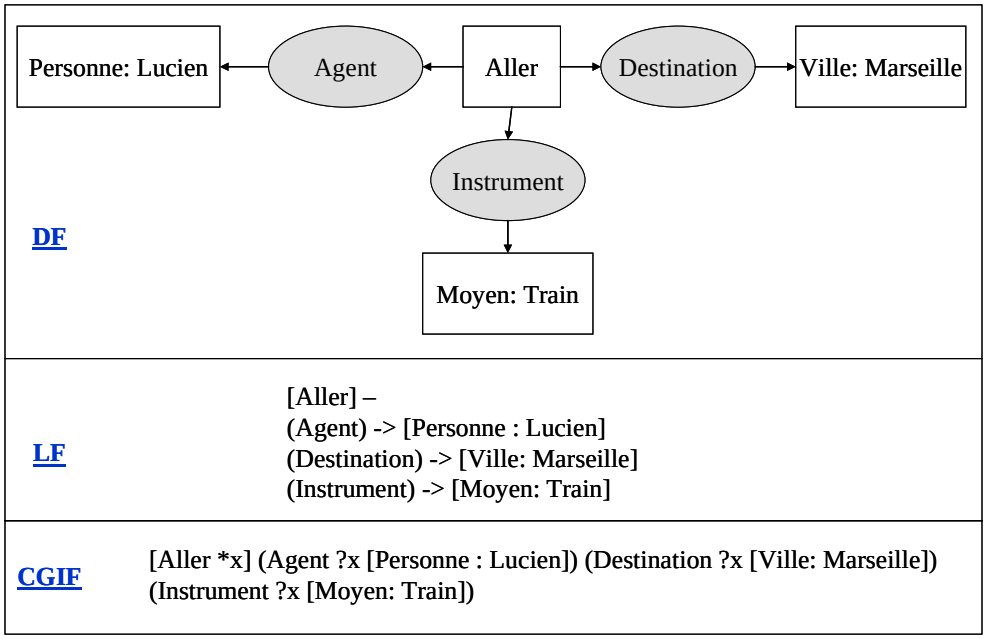

Figure IV.16. Exemple de représentation avec les graphes conceptuels

Les *logiques de description* [Napoli 1997] (également appelées logiques terminologiques, ou encore langages basés sur les termes), introduit au début des années 90, constituent l'aboutissement des longs travaux de recherche, menés entre autres par Nebel, dans le cadre de la formalisation dans les langages de représentation des connaissances. Ces logiques manipulent trois types d'entités qui sont les individus (entités concrètes de l'univers), les concepts (entités génériques) et les rôles (relations binaires entre individus), et offrent deux types de langages pour pouvoir représenter les connaissances: les langages assertionnels et les langages terminologiques. Les langages assertionnels (A-Box) permettent de décrire les individus, tandis que les langages terminologiques (T-Box) permettent de décrire les concepts et les rôles. Le schéma ci-après (figure IV.17) permet d'illustrer le méta modèle de la logique des descriptions qui montre ainsi les principaux constructeurs utilisés pour la modélisation : expression, constructeur ensemblistes (union, intersection) et les quantificateurs (existentiel, universel), etc.

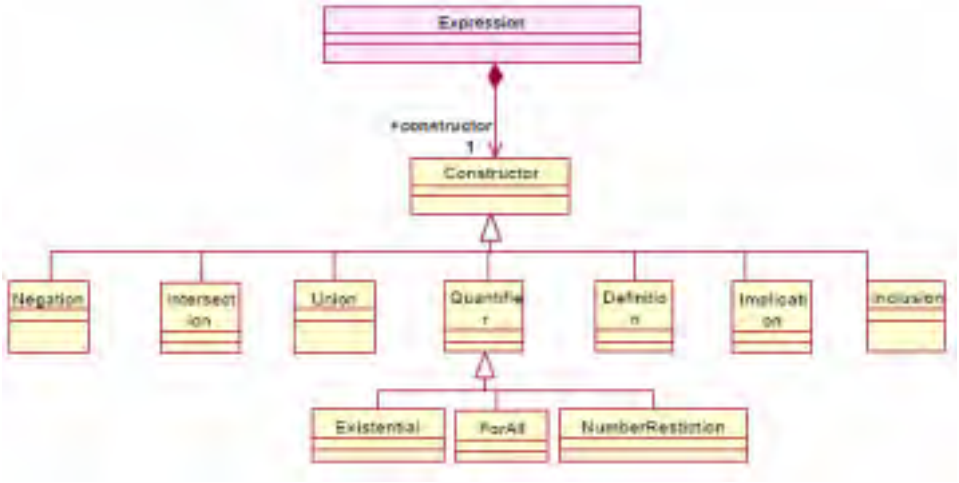

Figure IV.17. Extrait du méta-modèle DL

A partir des années 90, plusieurs langages ontologiques ont vu le jour. Certains se basent sur la logique et d'autres sur les frames. Ils sont brièvement présentés ci-dessous.

Le langage KIF (Knowledge Interchange Format) constitue sans doute l'un des langages spécialement dédié aux ontologies. Il est basé sur la logique de premier ordre et il est créé en 1992 comme format d'échange pour divers systèmes à base de règles. Si on reprend l'exemple de la figure IV.16 qu'on traduit en KIF, on obtiendra la formule suivante.

```
(exists ((?x Aller) (y Personne) (?z Ville) (?w Bus)
   (and (Name ?y 'Lucien') (Name ?z 'Marseille')
     (Agent ?x ?y) (Destination ?x ?z) (Instrument ?x?w)))
```

Figure IV.18. Exemple de représentation en KIF

Le langage Ontolingua, qui est construit sur le langage KIF, est développé en 1992 par l'université de Stanford. Il permet de représenter des connaissances en combinant des frames et la logique des prédicats de premier ordre. Il a constitué un référentiel pendant de nombreuses années en matière de langage de construction d'ontologies. Bien qu'il représente les ontologies au niveau symbolique sans abstraction réelle, il présente tout de même une assez forte expressivité grâce à ses nombreux éléments permettant de formaliser les concepts, les relations, les fonctions, les instances et les axiomes. Néanmoins, ce langage demeure assez limité dans la construction de mécanismes de raisonnement.

Le langage LOOM a été développé en 1992 par l'ISI de l'université de South California. Initialement conçu pour les bases de connaissances générales, il peut toutefois être utilisé pour représenter des ontologies. LOOM est basé sur les logiques de description et les règles de production, et il peut fournir une classification automatique des concepts. Il permet de représenter les connaissances avec les composantes suivantes: concepts, taxonomies, relations, fonctions et règles.

OCML (Operational Conceptual Modelling Language) est développé en 1993 par KMI de Open University. Il a été conçu comme une extension du langage Ontolingua, dans la mesure où il comble les lacunes de ce dernier en prenant en charge les règles de production, ce qui permet d'améliorer les mécanismes de raisonnement d'Ontolingua. Son orientation vers les mécanismes de raisonnement fait d'OCML un langage adapté aux méthodes de résolution de problèmes.

F-Logic, développé en 1995 par Karlsruhe University, combine aussi bien les frames que la logique de premier ordre. Il permet de représenter des connaissances avec les éléments suivants: concepts, taxonomies, relations, axiomes et règles de déduction. Il possède un moteur d'inférence, Ontobroker, qui peut être utilisé pour dériver de nouvelles connaissances.

OKBC (Open Knowledge Base Connectivity), anciennement connu sous le nom de Generic Frame Protocol, a été développé en 1997 au sein du programme américain HPKB (High Performance Knowledge Base) sponsorisé par DARPA. Il constitue un protocol, et non pas un langage, et permet d'accéder aux bases de connaissances implémentées avec différents de langages de représentation.

IV.3.5.2. Langages de représentation basés Web

Avec l'explosion de l'Internet, de nouveaux langages basés Web ou langages d'annotation d'ontologies ont vu le jour pour gérer et exploiter toute la sémantique du Web. La figure suivante (figure IV.19) reproduite à partir de [Corcho et al. 2003] permet de résumer les principaux langages de cette catégorie sous forme d'une pile de langages construite sur la base du langage XML (voire HTML) et qui permet de procurer une syntaxe aux documents structurés à travers le Web.

Le langage OWL [OWLC 2004] renforce le vocabulaire de description des propriétés et des classes : entre autres, les relations entre les classes (par exemple, la disjonction), la cardinalité (par exemple, un seul exactement), l'égalité, le typage enrichi des propriétés, les caractéristiques des propriétés (par exemple, la symétrie) et les classes énumérées.

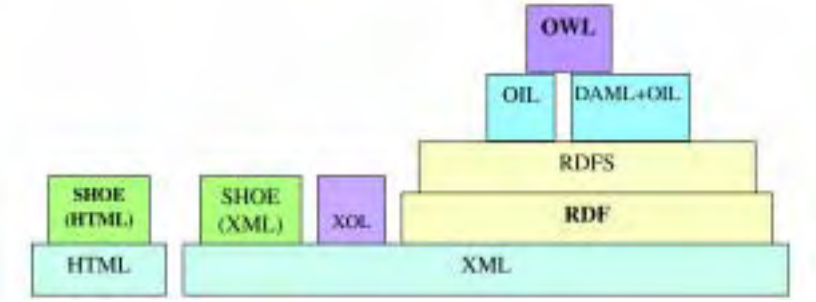

Figure IV.19. Pile des langages d'annotation d'ontologies

Le premier langage d'annotation d'ontologies est SHOE (Simple HTML Ontology Extensions). Il a été créé en 1996 comme une extension de HTML à l'université de Maryland. Il utilise des balises particulières qui permettent d'insérer des ontologies dans des documents HTML. Ce langage combine le frames et les règles de production ce qui lui permet de représenter des concepts, des taxonomies, des relations, et aussi des règles. Ces dernières permettent d'inférer de nouvelles connaissances.

Quelques temps après, avec l'avènement du XML en 1998, qui a été très vite adopté comme un standard pour les échanges d'informations sur le Web par le W3C (World Wide Web Consortium), SHOE a été modifié de telle sorte qu'il puisse supporter des documents structurés décrits en XML. D'autres langages ont par la suite été créés sur la base de la syntaxe de XML.

Le langage XOL (XML Ontology Language) a été développé par le centre AI de SRI International en 1999 comme un langage basé XML qui permet de spécifier des concepts, des taxonomies et des relations binaires. Bien qu'il ne permette pas d'effectuer des raisonnements, ce langage a été longtemps utilisé dans les échanges d'ontologies dans le domaine bio-médical.

Le langage RDF (Ressource Description Framework) a été développé par W3C comme un langage basé sur les réseaux sémantiques pour décrire les ressources du Web. Afin de renforcer ce langage, RDF Schema a été construit par W3C pour comme extension de RDF comportant des primitives basées sur des frames. RDF Schema permet notamment de déclarer les propriétés des ressources ainsi que le type des ressources. La combinaison de RDF et RDF Schema est connue sous le nom RDF(S). Bien que relativement limités dans la mesure où ils ne sont pas très expressifs, les langages RDF(S) peuvent cependant spécifier des concepts, des taxonomies et des relations

binaires. De plus, certaines règles existent pour notamment définir des contraintes sur les recherches.

La figure qui suit (figure IV.20) est une méta-vue de RDF et permet d'illustrer les principaux constructeurs utilisés pour modéliser la sémantique d'un domaine. On peut à titre d'exemple, citer le constructeur Class, Property et Resource qui représente respectivement la notion concept (ou classe), la notion d'attribut (propriété) et la notion d'objet (ressource) décrit.

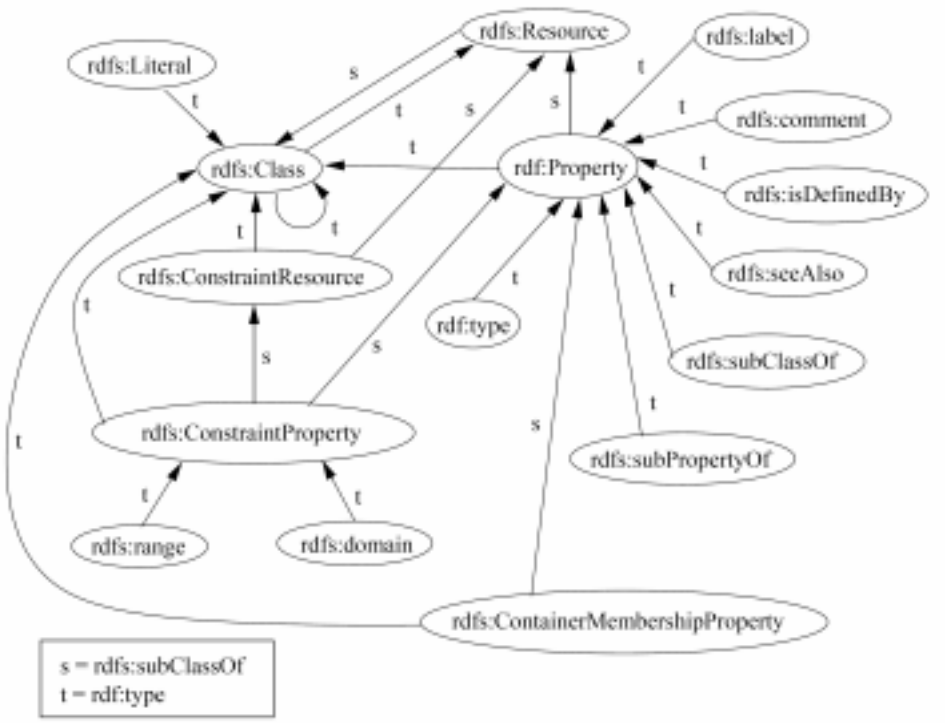

Figure IV.20. Extrait du méta-modèle RDF [W3C-RDFS 2002]

Les langages SHOE, XOL RDF(S) constituent la fondation du Web sémantique. Et dans ce contexte précis, d'autres langages ont été élaborés sur la base de RDF(S), il s'agit essentiellement de OIL, DAML+OIL et OWL.

OIL (Ontology Inference Layer) est développé au cours du projet européen IST On-To-Knowledge. Basé sur RDF(S), le langage OIL permet de décrire les ontologies en combinant les primitives de modélisation utilisées dans les langages de frames et le raisonnement formel des logiques de description pour exprimer des ontologies sur le Web [Fensel 2001]. Le langage OIL est limité du point de vue expressivité, puisque par exemple il ne supporte pas les types concrets, restriction qui se justifie par ailleurs par la volonté de pouvoir conserver un test de subsomption décidable [Fensel 2001].

Parallèlement à cette initiative européenne, deux autres langages ont été développés au cours du projet américain DARPA (DARPA Agent Markup Language). Il s'agit du langage DAML-ONT et du langage DAML-LOGIC qui permettent respectivement de spécifier des ontologies et d'ajouter des règles d'inférences. Très rapidement le langage DAML-ONT a évolué, en 2000, en DAML+OIL, créé également dans le cadre du projet DARPA et qui représente la fusion des deux langages DAML-ONT et OIL. Ce nouveau langage supporte désormais des types primitifs, tels qu'on les retrouve dans XML, et supporte également la définition d'un certain nombre d'axiomes comme l'équivalence de

classes ou de propriétés. DAML+OIL est équivalent à une logique descriptive très expressive, mais demeure assez limité dans le contexte d'intégration des applications sur le Web.

Finalement, en 2001, et dans le cadre du groupe de travail WebOnto, un nouveau langage pour l'annotation des documents Web a été développé. Il s'agit de OWL (Web Ontology Language) qui a été construit sur DAML+OIL. OWL est développé pour pallier les lacunes de DAML+OIL, et plus précisément pour être utilisé dans des situations où les informations contenues dans les documents Web doivent être traitées par des applications logicielles, par opposition aux situations dans lesquelles le contenu est simplement présenté aux humains, cas traditionnel de l'utilisation du Web de nos jours. La section IV.3.5.4 va effectuer une présentation plus détaillée de OWL.

IV.3.5.3. Critères de sélection d'un langage

En ce qui concerne la sélection de langages, les critères le plus souvent retenus sont principalement l'expressivité et la performance du langage [Gandon 2002][OntoWeb 2002]. L'expressivité qui est liée au nombre de concepts pouvant être utilisés pour décrire les composants d'une ontologie, alors que la performance est liée à la capacité et la facilité de représentation des connaissances du domaine. D'après certains auteurs [Fensel 2001], il semblerait que plus un langage est expressif et moins il est performant, et vice versa. Il existe aussi un autre dilemme entre l'expressivité et le raisonnement dans le sens où l'expressivité d'un langage doit être parfois limitée afin d'assurer un bon service de raisonnement [OntoWeb 2002]. Mais on peut retenir le principe d'utilisation de langages expressifs dans les phases amont, pour acquérir les connaissances ontologiques où l'usage de la langue naturelle ou d'un langage de modélisation semi-informels ou semi-formels (tels que les diagrammes Entité-Association [Tardieu et al. 2002], UML [Rumbaugh et al. 2005]) est généralement le plus adapté. Alors que pour les phases en aval, l'usage de langages de représentation formels et/ou exécutables mettant en œuvre des logiques formelles (logique des descriptions, logique de premier ordre, logique d'action, logique de Horn, etc.) ou des algèbres (formalisme Z) est par contre le plus adapté [OntoWeb 2002].

Vu leurs caractéristiques intéressantes aussi bien du point de vue expressivité que du point de vue performances, nous décidons dans le cadre de nos travaux, d'utiliser le langage UML pour la représentation semi-formelle de nos ontologies et nous décidons d'utiliser le langage décidable OWL en tant que langage d'opérationnalisation de nos ontologies.

IV.3.5.4. Langage OWL

Comme nous l'avons évoqué plus haut, OWL [OWLC 2004] est un langage d'ontologie Web et il appartient à la famille en pleine croissance des recommandations W3C liées au Web sémantique, à savoir le langage XML (pour décrire la syntaxe), le langage XML schéma (pour restreindre la structure des documents XML), le langage RDF (qui fournit une sémantique simplifiée), le schéma RDF (permettant de prendre en compte la généralisation des propriétés et des classes).

Une ontologie OWL est composée d'un en-tête, d'axiomes et de faits. L'en-tête représente des méta-données. Les axiomes permettent de définir des concepts ou des relations, tandis que les faits sont simplement des individus, c'est à dire des instances dont les propriétés sont renseignées. Pour décrire des ontologies, OWL comporte trois couches ou sous-langages: OWL-Lite et OWL-DL et OWL Full. OWL-Lite est la forme

la plus légère du langage et il est considéré comme une extension de RDF(S). Le langage OWL-DL, qui inclut OWL-Lite, est considéré comme une logique descriptive très proche de la logique descriptive SHOIQ(D), et il permet de fournir des constructeurs qui enrichissent ceux offerts par OWL-Lite. OWL-Full, qui inclut OWL-DL, complète les deux langages en accroissant ainsi l'expressivité.

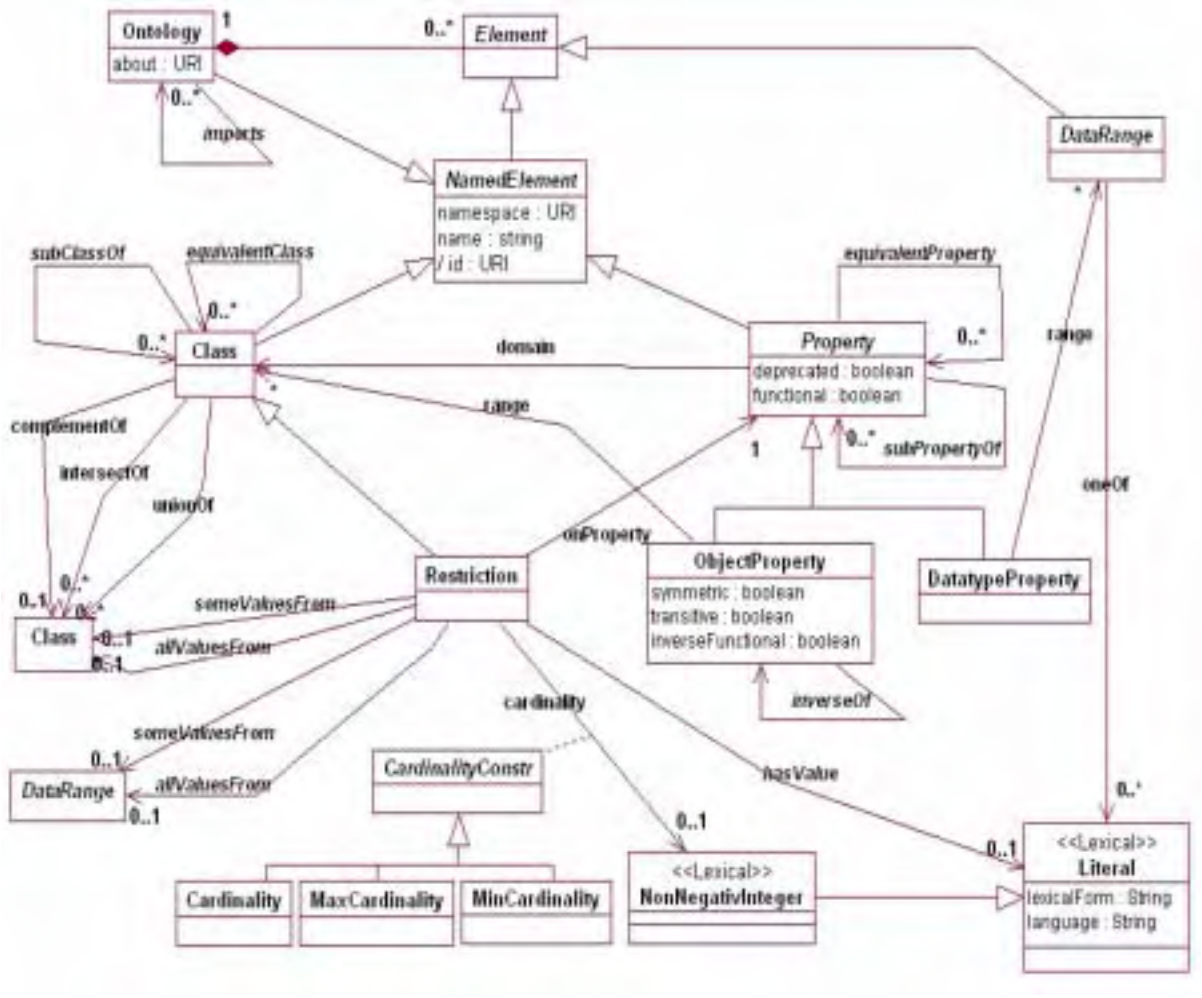

Figure IV.21. Extrait du méta modèle OWL

Le méta-modèle de la figure IV.21 illustre les principaux constructeurs utilisés par OWL-Lite. Comme on peut le remarquer, OWL-Lite réutilise les constructeurs du RDF(S) tels que Class, Property, subClassOf, subPropertyOf, domain, range, etc. Il définit également un certain nombre de constructeurs nouveaux portant sur

- l'intersection des classes (intersectionOf),
- l'(in)égalité des classes (equivalentClass) et des propriétés (equivalentProperty),
- les propriétés algébriques des propriétés (inverseOf),
- les types de données (DataTypeProperty, ObjectTypeProperty),
- les restrictions sur les rôles (allValuesFrom, someValuesFrom),
- les cardinalités (Cardinality, MinCadinality, Maxcadinality),
- et les informations d'entête (Ontology, imports), etc.

IV.3.6. Discussions

A travers de ce qui a été présenté précédemment, il ressort que la notion d'ontologie constitue l'une des approches les plus efficaces pour représenter la sémantique des systèmes d'information industriels. La maturité croissante des langages et des outils laisse penser que cette technologie peut sans doute être généralisée dans les prochaines années à tout le cycle de vie des systèmes d'information.

Cependant, des lacunes peuvent être décelées. Il s'agit en particulier de la méthodologie à mettre en œuvre pour capturer de façon efficace toute la sémantique au sein d'une grande entreprise industrielle. En effet, dans le cadre de notre projet d'intégration d'applications industrielles, nous nous sommes aperçus de la nécessité de raffiner davantage le cycle de développement des ontologies afin de pouvoir mieux assister les utilisateurs dans leur processus de développement d'ontologies industrielles. Cet aspect des choses sera traité dans les prochains chapitres.

IV.4. Architectures des ontologies pour l'intégration

Il existe plusieurs approches d'utilisation des ontologies pour l'explicitation de la sémantique. Une première approche intuitive est celle qui consiste à utiliser une ontologie unique qui permet de capturer toute la sémantique de la réalité considérée. D'autres approches consistent à utiliser, non pas une seule, mais plusieurs ontologies. Ces dernières approches sont notamment motivées par le fait qu'il est souvent impossible voire irréaliste de s'entendre sur une seule ontologie, ou à la limite sur un nombre assez réduit d'ontologies, au sein de grandes entreprises industrielles. Dans le domaine de l'intégration des données, [Wache et al. 2001] recense trois approches possibles qui sont : l'approche mono-ontologie, l'approche multi-ontologies et l'approche hybride.

IV.4.1. Approche mono-ontologie

L'approche mono-ontologie (single-ontology approach) permet de mettre en œuvre une ontologie unique et partagée entre les différents systèmes d'information, fournissant ainsi un vocabulaire partagé pour la spécification de la sémantique. Toutes les sources de données sont reliées à une ontologie globale (figure IV.22). Un exemple typique d'implémentation de cette approche est le projet SIMS [Arens et al. 1993] où un modèle indépendant de chaque source de données est décrit pour ce système en reliant les objets de chaque source au modèle du domaine global. Les approches mono-ontologie sont les approches les plus simples à mettre en œuvre lorsque les sources de données se réfèrent à des domaines similaires. Dans le cas où les sources concernent des domaines hétérogènes, il devient difficile voire impossible à concrétiser un engagement ontologique. Une autre limite importante de cette approche tient du fait que la modification des sources peut affecter l'ontologie globale.

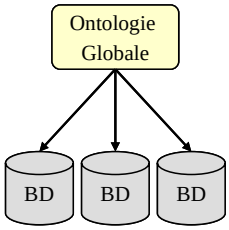

Figure IV.22. Approche mono-ontologie (adapté de [Wache et al. 2001])

IV.4.2. Approche multi-ontologies

L'approche multi-ontologies (multi-ontology approach) permet d'associer une ontologie locale à chaque Système d'Information (figure IV.23). Cette approche convient dans le cas où il devient difficile de trouver une ontologie commune résultant des grandes différences sémantiques existant entre les systèmes. Des mappings inter-ontologiques sont alors nécessaires afin d'établir une interprétation commune des données. Comme exemple d'implémentation, le projet OBSERVER [Mena et al. 2000][Mena et al. 1996] s'inscrit dans cette approche. Il associe à chaque source de données une ontologie locale. L'intérêt de cette approche est le fait que les sources peuvent être développées de façon indépendante, mais l'absence d'un vocabulaire commun conduit généralement à une difficulté extrême pour comparer différentes ontologies sources. Pour pallier cette difficulté, un formalisme de représentation additionnelle définissant le mapping inter-ontologie est utilisé. Ce dernier identifie sémantiquement les termes correspondants des différents ontologies sources.

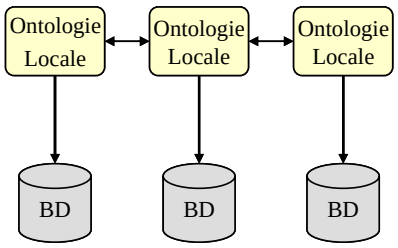

Figure IV.23. Approche multi-ontologies (adapté de [Wache et al. 2001])

IV.4.3. Approche hybride

L'approche hybride combine les deux approches précédentes en utilisant à la fois des ontologies locales et une ontologie partagée (parfois un vocabulaire commun) afin de simplifier les mappings ontologiques (figure IV.24). L'ontologie partagée fournit un vocabulaire commun et global ce qui rend les ontologies locales comparables. Cette ontologie partagée comprend les termes de base ou primitives d'un domaine. Afin de construire des termes complexes, ces termes de base sont combinés par des opérateurs, et les termes peuvent être comparés plus facilement que dans une approche multi-ontologies. Comme exemples de projets associés à cette approche, nous pouvons citer le projet COIN [Goh 1997], le projet MECOTA [Wache et al. 1999], et le projet BUSTER [Stuckenschmidt 2000][Visser et stuckenschmidt 2002] et qui utilisent respectivement la notion de contexte local, des mécanismes d'annotation, des raffinements d'une ontologie générale pour décrire localement les sources de données.

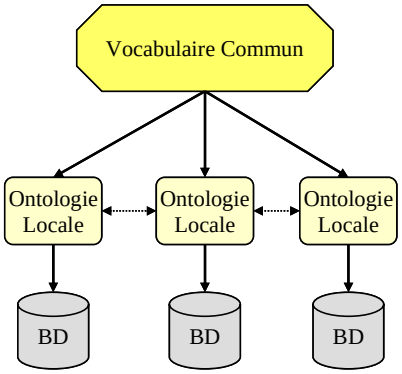

Figure IV.24. Approche hybride (adapté de [Wache et al. 2001])

IV.4.4. Discussions

Nous avons brièvement reproduit dans le tableau suivant (tableau IV.1) les principales caractéristiques de chacune de ces approches telles que présentées dans [Wache et al. 2001]. Il ressort de l'étude de ce tableau que l'approche hybride peut constituer une approche pertinente pour l'intégration des systèmes d'information industriels dans la mesure où elle peut fournir à la fois de la flexibilité et une simplicité de mise en œuvre.

	Approche mono-ontologie	Approche multi-ontologies	Approche hybride
Effort d'implémentation	Simple	Élevé	Raisonnable
Hétérogénéité sémantique	Sémantique unifiée d'un domaine	Hétérogénéité sémantique	Hétérogénéité sémantique
Impact lors de l'ajout/mise à jour d'applications	Besoin d'adaptation de l'ontologie globale	Ajouter une autre ontologie locale, mettre à jour l'ontologie locale associée à l'application mise à jour. Liaison avec les autres ontologies locales.	Ajouter une autre ontologie locale, mettre à jour l'ontologie locale associée à l'application mise à jour.
Comparaison d'ontologies	pas de comparaison d'ontologies (car il n'existe qu'une seule ontologie)	Difficile du fait qu'il n'existe pas d'ontologie partagée	Simple du fait qu'il existe une ontologie partagée

Tableau IV.1. Évaluation des approches ontologiques [Wache et al. 2001]

Du fait qu'elles comportent plus d'une ontologie, les approches à ontologies multiples et les approches hybrides peuvent être confrontées au problème d'hétérogénéité ontologique [Klein 2001]. Dans ce cas, il devient nécessaire d'intégrer et de faire interopérer des ontologies. Cet aspect est traité dans la section suivante.

IV.5. Intégration des ontologies

Le problème d'intégration des ontologies est la conséquence logique de l'utilisation des architectures se basant sur plusieurs ontologies. Plusieurs travaux existent qui étudient cet aspect et qui proposent des solutions pour résoudre les différentes hétérogénéités qui peuvent exister au niveau des ontologies. Dans cette section nous allons étudier les hétérogénéités d'ontologies et nous allons ensuite présenter les solutions typiques utilisées pour leur résolution.

IV.5.1. Hétérogénéité des ontologies

Il existe dans la littérature plusieurs classifications d'hétérogénéités [Klein 2001][Noy 2004][KnowledgeWeb 2004][Giunchiglia et Walsh 1992][Euzenat 2001][Corcho 2004][Hameed et al. 2004][Serafini et al. 04][Kalfoglou et Schorlemmer 2003-b][McGuiness et al. 2000]. On a retenu dans le cadre de nos travaux la classification proposée dans le cadre du projet Knowledge Web [KnowledgeWeb 2004] qui distingue fondamentalement quatre niveaux d'hétérogénéités qui sont

- les hétérogénéités de niveau syntaxique (ou niveau de langage);
- les hétérogénéités de niveau terminologique
- les hétérogénéités de niveau conceptuel
- et des hétérogénéités de niveau pragmatique.

IV.5.1.1. Niveau syntaxique

Le premier type d'hétérogénéité est lié aux caractéristiques des langages utilisés pour représenter les ontologies. Les langages peuvent différer dans leur syntaxe, mais plus important encore dans les concepts et aussi dans les contraintes utilisés dans les deux langages : certains concepts et/ou contraintes d'un langage ne sont pas disponibles dans un autre langage. Ce type d'hétérogénéité soulève le problème de translation ou de traduction d'ontologies [Corcho et al. 2003-b] qui constitue un cas particulier du problème d'intégration d'ontologies. Ce problème est généralement résolu de façon relativement simple par la mise en œuvre d'un processus de translation ou de traduction qui se base souvent sur des techniques de normalisation qui permettent ainsi de résoudre les différences syntaxiques pouvant exister entre deux ontologies données. Dans le cadre des travaux que nous menons, cet aspect n'est pas considéré dans la mesure où l'on impose l'utilisation d'un langage unique.

IV.5.1.2. Niveau terminologique

Le second type d'hétérogénéité concerne toutes les différences liées au processus de nomination des entités (classes, propriétés, etc.). A ce titre, il est possible d'établir les conflits typiques suivants

- Synonymie : où plusieurs mots qui désignent la même entité;
- Polysémie : où un même mot est utilisé pour dénommer différentes entités;
- Multilinguisme : où des mots de différents langages sont utilisés pour nommer des entités;
- Et des variations syntaxiques d'un même mot (abréviations, etc.)

Comme le précise [KnowledgeWeb 2004], il est important de retenir que du point de vue complexité, les hétérogénéité terminologiques ne sont pas aussi complexes que les hétérogénéités conceptuelles. Cependant, la plupart des hétérogénéités qui occurrent dans la réalité se situent à ce niveau. Ce qui rend ce dernier assez critique.

IV.5.1.3. Niveau conceptuel

Les hétérogénéités conceptuelles concernent les différences de couverture de l'univers du discours, le degré de granularité et le point de vue de l'ontologie (figure IV.25) :

- Couverture : qui désigne le fait que les ontologies couvrent différentes portions de l'univers du discours. Par exemple, on peut définir une ontologie sur la fabrication de composants électroniques qui inclut la description des opérateurs de maintenance, alors qu'on peut également avoir une autre ontologie qui ne décrit pas les opérateurs.
- Granularité : qui signifie que les entités des ontologies peuvent décrire les objets de la réalité à des degrés de détails différents. Par exemple : on peut avoir une ontologie d'équipement qui ne s'intéresse qu'aux différents type de machines dont on dispose, alors que l'on peut définir une autre ontologie qui inclut tous les équipements individuels avec leurs caractéristiques spécifiques;
- Perspective : qui signifie que les ontologies couvrent des points de vue différents. Par exemple: on peut définir deux ontologies décrivant les composants microélectroniques avec deux points de vue différents: l'un concerne l'aspect fabrication et l'autre concerne l'aspect marketing.

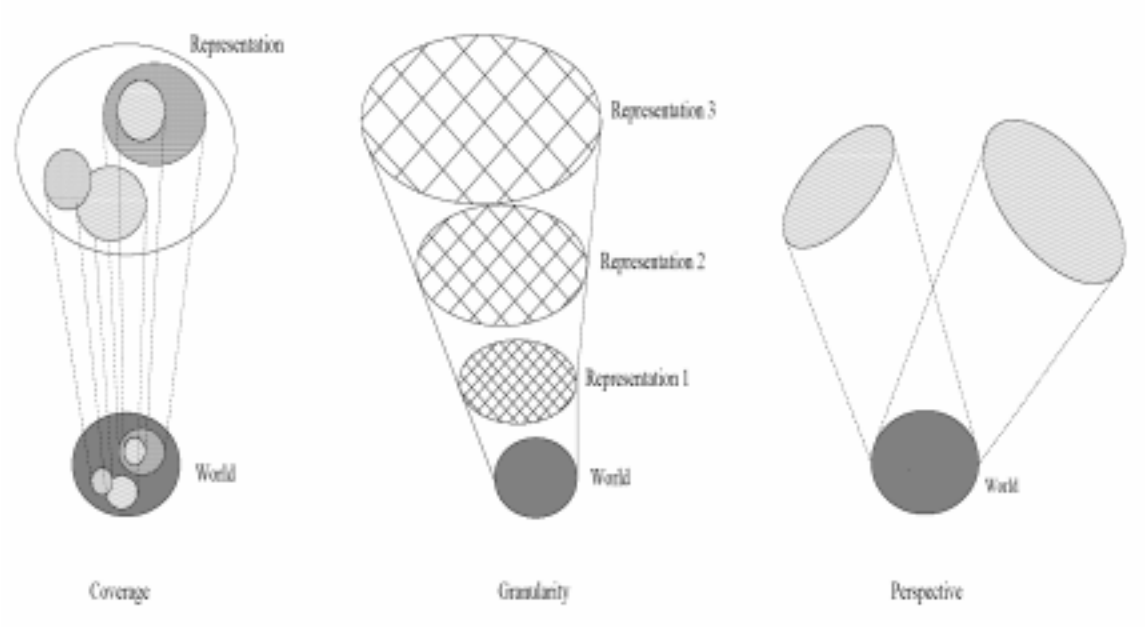

Figure IV.25. Trois dimensions d'hétérogénéités conceptuelles [KnowledgeWeb 2004]

IV.5.1.4. Niveau pragmatique

Le niveau pragmatique est le niveau le plus complexe. Il concerne les hétérogénéités d'interprétation d'une ontologie qui peuvent survenir lorsque des individus ou des communautés différentes interprètent différemment l'ontologie selon différents

contextes. Dans le cadre de nos travaux, on ne s'intéresse pas à cet aspect des choses, qui relève plutôt de l'intégration au niveau des connaissances.

IV.5.2. Processus d'intégration des ontologies

La médiation d'ontologies qui est le processus qui permet de réconcilier différentes ontologies, se base généralement sur le principe de la combinaison d'ontologies. Ce processus de combinaison est très souvent appelé intégration d'ontologies[24] qui signifie à la fois le fait que les ontologies peuvent être fusionnées en une ontologie unique nouvellement créée, et aussi le fait que les ontologies peuvent être mises en correspondance sans création de nouvelle ontologie [Klein 2001]. Dans tous les cas, les ontologies à intégrer doivent être réconciliées dans le sens où elles doivent être ramenées à un engagement ontologique mutuel.

Il est généralement admis que la démarche d'intégration d'ontologies est basée sur trois sous-processus [Kavouras 2003] : le processus d'extraction de la sémantique, le processus de comparaison et enfin le processus d'intégration proprement dit (figure IV.26).

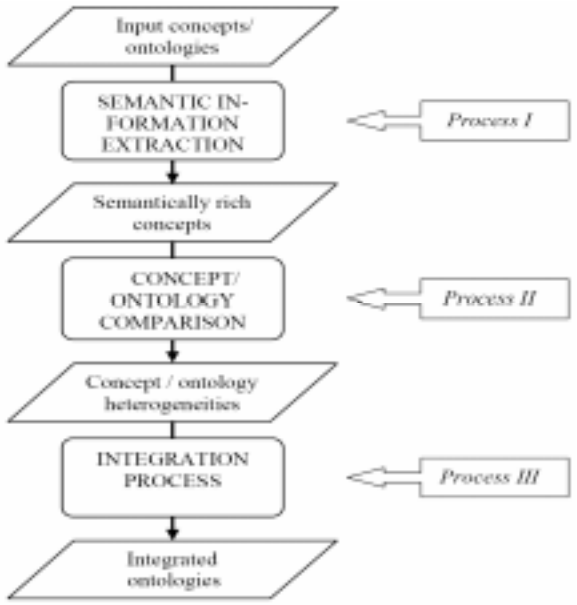

Figure IV.26. Méthodologie d'intégration d'ontologies (adapté de [Kavouras 2003])

Le processus d'extraction correspond à la modélisation de la sémantique effectuée sur la base d'ontologies. Le processus de comparaison d'ontologies permet de quantifier le degré de similarité ou d'hétérogénéité des concepts des ontologies à intégrer, et ce en se basant très souvent sur des calculs de distances [Euzenat et Valtchev 2004][Maedche et Staab 2002][Sheth et Kashyap 1993]. Et enfin, le dernier processus permet d'intégrer les ontologies concernées.

[24] On peut remarquer, au passage, que le terme d'intégration d'ontologies a été utilisé de façon abusive par la communauté d'ontologies dans la mesure où il est associé à une multitude de situations [Pinto 1999], il constitue dans notre cas le moyen pour résoudre le problème d'hétérogénéité ontologique dans le cadre général. A titre d'exemple, voici une autre définition, celle de [Sowa 1997], très connue qui considère l'intégration comme une fusion : l'intégration d'ontologies est le processus de recherche de ressemblances entre deux ontologies A et B et dérive une nouvelle ontologie C qui permet de faciliter l'interopérabilité entre les systèmes basés respectivement sur A et B. L'ontologie C peut remplacer les ontologies initiales A et B et peut aussi être utilisée comme une ontologie intermédiaire entre les systèmes exploitant les ontologies initiales. Cette définition est relativement restreinte.

IV.5.3. Approches d'intégration des ontologies

Il existe différentes approches pour intégrer des ontologies. Elles s'étendent des approches faiblement couplées jusqu'aux approches fortement couplées [Ding 2002]. Dans le cadre du projet européen INTEROP [Interop 2005], on a retenu les principales approches d'intégration suivantes : le mapping, l'alignement, la transformation et la fusion d'ontologies. Chacune de ces approches est présentée ci-dessous.

IV.5.3.1. Mapping d'ontologies

Le mapping d'ontologies repose sur la définition de relations de correspondance entre les entités de deux ontologies qui présentent une similitude (figure IV.27). Habituellement la fonction est de type 1 à 1, mais on peut définir des relations fondées sur la notion de mesure de similarité. Une caractéristique importante de cette approche est qu'elle ne modifie pas les ontologies impliquées et qu'elle produit en sortie un ensemble de correspondances.

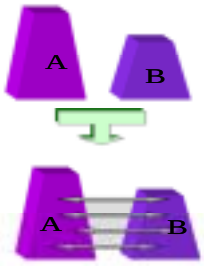

Figure IV.27. Principe du mapping d'ontologies [Interop 2005]

IV.5.3.2. Alignement d'ontologies

L'alignement d'ontologies est considéré comme le processus qui permet d'amener deux ou plusieurs ontologies hétérogènes à un "accord mutuel" en les rendant ainsi consistantes et mutuellement cohérentes. L'alignement d'ontologies nécessite la transformation des ontologies impliquées en procédant à l'élimination des entités non pertinentes et au rajout des entités manquantes (figure IV.28).

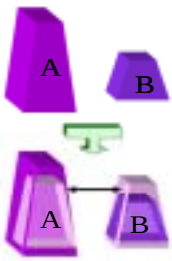

Figure IV.28. Principe de l'alignement d'ontologies [Interop 2005]

IV.5.3.3. Transformation d'ontologies

La transformation d'ontologies est un processus qui permet de changer la structure des ontologies en conservant au maximum sa sémantique (figure IV.29). Selon que la transformation se fait sans pertes ou avec perte d'informations, on parle de transformation sans pertes (lossless) et de transformation avec pertes (lossy)

Figure IV.29. Principe de la transformation d'ontologies [Interop 2005]

IV.5.3.4. Fusion d'ontologies

La fusion d'ontologies est une approche qui permet de construire à partir de deux ou plusieurs ontologies existantes une nouvelle ontologie (figure IV.30). La nouvelle ontologie intégrée s'obtient généralement en combinant les ontologies initiales et quelques concepts supplémentaires nécessaires pour la combinaison [Kavouras 2003].

Ce type d'approche est généralement utilisé, dans le cadre d'intégration de données, pour obtenir une ontologie globale qui sert d'interface pour un certain nombre d'ontologies locales [Calvanese et al. 2001]. La principale limite de cette architecture tient au fait que l'on a besoin de développer, et encore plus important, on a besoin de s'entendre sur une ontologie globale, chose qui n'est toujours pas si évidente dans les environnements fortement évolutifs.

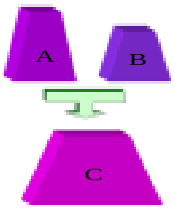

Figure IV.30. Principe de la fusion d'ontologies [Interop 2005]

IV.5.3.5. Discussions

Etant donné que l'objectif de l'intégration est d'arriver à une médiation entre les différentes ontologies, les techniques d'intégration les plus appropriées dans le cas de nos travaux sont celles qui établissent des liens entre les concepts sans nécessité de procéder à une fusion d'ontologies (intégration), autrement dit les techniques de mapping ontologique (ou de médiation d'ontologies). Cette dernière technique possède l'avantage des solutions faiblement couplées, ce qui permet d'apporter plus de flexibilité dans le processus d'intégration des systèmes d'information industriels.

Dans la suite de ce document nous allons nous restreindre principalement au mapping ontologique qui constitue à notre connaissance l'approche la plus adaptée pour les travaux que nous menons dans le cadre d'intégration dans le domaine industriel.

IV.5.4. Mapping d'ontologies

Comme il a été évoqué précédemment, nous nous focalisons dans le cadre de nos travaux sur l'approche d'intégration des ontologies basée sur des mappings. Cette

approche permet d'apporter plus de flexibilité dans les environnements aussi complexes que sont les secteurs industriels. Nous allons dans ce qui suit, étudier de plus près cette approche.

IV.5.4.1. Classification des mappings d'ontologies

L'approche mapping d'ontologies repose sur des architectures d'ontologies mettant en jeu deux ou plusieurs ontologies. Ce sont donc les approches multi-ontologies et les approches hybrides (cf. § IV.4) qui peuvent être concernées par les mappings. Selon l'architecture d'ontologie utilisée, on peut avoir :

- dans le cadre des architectures multi-ontologies, des mappings point-à-point (un-à-un ou one-to-one mapping) qui permettent de créer des mappings entre chaque paire d'ontologies. Par exemple, l'approche utilisée dans le projet OBSERVER [Mena et al. 2000] repose sur des mappings de type point-à-point. le nombre de mappings créés par l'approche multi-ontologies est de l'ordre de $o(n^2)$ où n est le nombre d'ontologies impliquées. Bien qu'elle soit relativement lourde, l'approche multi-ontologies permet de préserver la nature décentralisée des systèmes d'information à intégrer. Mais, on reproduit ainsi la complexité des architectures point-à-point.
- dans le cadre des architectures hybrides, des mappings basés sur une ontologie globale partagée (single-shared ontology). Par exemple, l'approche utilisée dans le projet MOMIS [Bergamaschi et al. 2001] repose sur des mappings de ce type. Comparée, à l'approche hybride qui ne crée que $o(n)$ mappings, où n est le nombre d'ontologies locales impliquées.

En outre, il existe une autre classification des mappings basée sur le mode de liaison des entités des ontologies impliquées dans le mapping. Aussi, on distingue principalement deux types de mappings qui sont [Bruijn et Polleres 2004] :

- Les mappings unidirectionnels : permettent de spécifier des relations entre deux ontologies uniquement dans une seule direction.
- Les mappings bidirectionnels : permettent de définir des relations entre deux ontologies dans les deux directions.

Les mappings unidirectionnels sont relativement simples à mettre en œuvre. Les mappings bidirectionnels, quant à eux, sont généralement plus flexibles, mais ils demandent souvent plus d'effort en matière de spécification que dans le cas des mappings unidirectionnels.

Notons que dans le cadre d'intégration des systèmes d'informations industriels, les mappings bidirectionnels basés sur une ontologie partagée sont à notre sens la catégorie de mappings les plus favorables. Ils permettent de définir l'approche la plus réaliste et également la plus flexible. C'est la raison principale qui nous pousse à nous intéresser dans les prochains chapitres à ce type de mappings.

IV.5.4.2. Formalisation de la notion de mapping

Formellement, il est possible de définir la notion de mapping d'ontologies comme un ensemble de correspondances au sens mathématique entre un certain nombre d'ontologies [KnowledgeWeb 2004]. Autrement dit, un mapping entre deux ou plusieurs ontologies est un ensemble de liens (nommées aussi mappings). Un mapping désigne une expression formelle qui établit une relation sémantique entre deux ontologies. Quand la relation inter-ontologique est orientée, on parle très souvent de

mapping au sens mathématique, c'est-à-dire de la notion de fonction ou de morphisme mathématiques [Kalfoglou et Schorlemmer 2003].

Du point de vue abstrait, [KnowledgeWeb 2004] définit un mapping de deux ontologies O et O' comme un opérateur $\alpha(O, O')$ qui

- à partir des deux ontologies O et O' de couverture différente, permet de réconcilier l'utilisation simultanée des deux ontologies pour la description de l'univers du discours;
- à partir des deux ontologies O et O' de granularité différente, permet d'associer aux entités de l'ontologie O les entités homologues dans O';
- à partir des deux ontologies O et O' de point de vue différent, permet de déduire le point de vue de O' associé à une entité de l'ontologie O.

Le processus de mapping consiste donc à générer une correspondance *M'* à partir de deux ontologies *O* et *O'*. Aussi, [KnowledgeWeb 2004] considère le processus de mapping comme une fonction *f* qui à partir d'une paire d'ontologies *O* et *O'* à réconcilier, un mapping initial *M*, un ensemble de paramètres *p*, un ensemble de ressources *r*, produit un nouveau mapping *M'* entre ces deux ontologies :

$$M = f(O, O', M, p, r)$$

Ce processus peut être représenté comme suit (figure IV.31) :

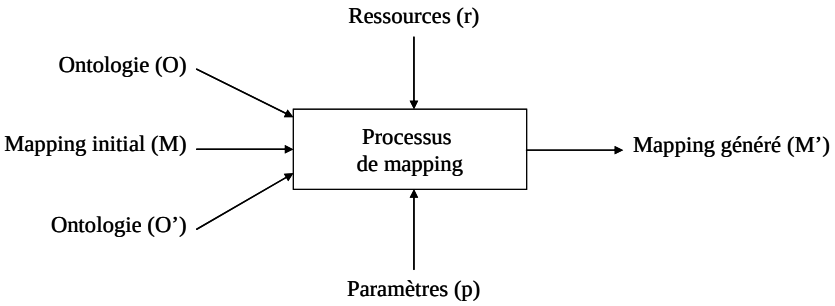

Figure IV.31. Processus générique de mapping (adapté de [KnowledgeWeb 2004])

Il est bien entendu possible de définir le mapping de plusieurs ontologies, ce dernier est alors appelé le multi-mapping d'ontologies. Et il peut être considéré comme étant une extension de la fonction précédente, en insérant comme arguments les n ontologies concernées par le mapping :

$$M' = f(O_1, O_2, \ldots, O_n, M, p, r)$$

Le mapping initial *M* peut être (s'il n'est pas vide) un mapping injecté (de façon très souvent manuelle) en entrée en vue de le mettre à jour ou le compléter. Les ressources *r* peuvent être les ressources matérielles, logicielles et humaines utilisées pour exécuter le processus de mapping. Quant aux paramètres p, ces derniers peuvent être les hypothèses effectuées comme le seuil de vraisemblance à utiliser, etc.

IV.5.4.3. Processus de mapping d'ontologies

De façon générale, le processus de mapping ontologique peut se résumer principalement aux étapes suivantes [McGuiness et al. 2000]:

- Trouver les zones de recouvrement des ontologies sources (à intégrer)
- Relier les concepts qui sont sémantiquement proches en utilisant des relations sémantiques (relations d'équivalence, de subsumption, etc.)
- Vérifier la consistance, la cohérence et la non redondance du résultat obtenu.

De façon plus détaillée, le processus de mapping comprend généralement les étapes suivantes (figure IV.32) [Sekt 2004][25] :

- L'import des ontologies : qui consiste à charger les ontologies dans l'outil de mapping en effectuant éventuellement des translations de format;
- La recherche de similarités : qui consiste à trouver et de façon semi-automatique les similarités qui peuvent exister entre les entités des deux ontologies;
- La spécification des mappings : qui consiste à spécifier les mappings et qui peut aussi se faire de façon semi-automatique en utilisant un outil tels que PROMPT.

Figure IV.32. Processus de mapping d'ontologies [Bruijn et al. 2005]

Le processus de mapping est rarement un processus complètement automatique. Comme le précise [Klein 2001], le mapping d'ontologies constitue une tâche complexe du fait qu'il nécessite la compréhension de la sémantique liée aux concepts à lier. Ceci implique que le processus de mapping ne peut être complètement automatisé. Il nécessitera sans doute toujours l'intervention humaine. C'est d'ailleurs pour cette raison que tous les outils existants ne peuvent que, dans les meilleurs des cas, suggérer à l'utilisateur des mappings potentiels. Nous devons noter que les mappings d'ontologies peuvent conduire parfois à des changements dans au moins une ontologie source. Ces changements peuvent conduire à leur tour à de nouvelles versions d'ontologies. De plus, la translation d'ontologies est parfois opérée dans le cas d'utilisation de langages ontologiques différents. Mais ce dernier point ne nous intéresse pas dans la mesure où l'on s'insère dans le cadre d'un langage de représentation unifié.

IV.5.4.4. Mesures de similarité et de distances

Le processus mapping nécessite généralement de mettre en œuvre le calcul d'une distance de similarité qui est calculée plusieurs façons (similarité syntaxique / linguistique, similarité sémantique, etc.). Mais avant de rentrer dans le détail des

[25] Deliverable D.4.2.1 (State-of-the-art survey on Ontology Merging and Aligning V1).

méthodes utilisées dans le mapping, définissons la similarité et quelques fonctions de distance qui sont souvent utilisées.

Définition 1 (similarité) : une mesure de similarité $\sigma : O \times O \rightarrow \Re$ est une fonction qui exprime la similarité entre deux entités tels que

$$\forall x, y \in O, \sigma(x, y) \geq 0 \text{ (Positivité)}$$
$$\forall x \in O, \forall y, z \in O, \sigma(x, x) \geq \sigma(y, z) \text{ (Maximalité)}$$
$$\forall x, y \in O, \sigma(x, y) = \sigma(y, x) \text{ (Symétrie)}$$

De façon duale, il est possible de définir la relation de dissimilarité

Définition 2 (dissimilarité) : une mesure de dissimilarité $\delta : O \times O \rightarrow \Re$ est une fonction qui exprime la différence entre deux objets tels que

$$\forall x, y \in O, \delta(x, y) \geq 0 \text{ (Positivité)}$$
$$\forall x \in O, \sigma(x, x) = 0 \text{ (Minimalité)}$$
$$\forall x, y \in O, \delta(x, y) = \delta(y, x) \text{ (Symétrie)}$$

Définition 3 (distance) : une distance ou métrique $\delta : O \times O \rightarrow \Re$ est une fonction de dissimilarité qui vérifie les relations suivantes

$$\forall x, y \in O, \delta(x, y) = 0 \text{ ssi } x = y$$
$$\forall x, y, z \in O, \delta(x, y) + \delta(y, z) \geq \delta(x, z) \text{ (Inégalité triangulaire)}$$

Définition 4 (ultra-métrique) : une ultra-métrique $\delta : O \times O \rightarrow \Re$ est une métrique qui vérifie

$$\forall x, y, z \in O, \delta(x, y) \leq \max(\delta(x, z), \delta(y, z)) \text{ (Inégalité ultra-métrique)}$$

De plus, on met généralement en œuvre des fonctions normalisées dans la mesure où on restreint l'ensemble image à [0, 1]. Les similarités (dissimilarités) normalisées sont respectivement notées $\underline{\sigma}$ (resp. $\underline{\delta}$). Il est également trivial de constater que $\underline{\sigma} = 1 - \underline{\delta}$.

Toutes les distances présentées précédemment sont relativement simples. Or la réalité est plus complexe, les objets ne sont pas aussi simples, ils sont souvent décrits sous forme de vecteurs pour prendre en compte la complexité des paramètres qui les caractérisent. Aussi, les métriques précédentes sont généralement étendues à n dimensions. On peut citer, et à titre d'exemple, les deux distances les plus simples, celle de Minkowski et la somme pondérée.

Définition 5 (distance de Minkowski) : Soit O un ensemble d'objets qui peuvent être analysés selon n dimensions. La distance de Minkowski δ_M entre deux objets x et y est définie en fonction de la distance traditionnelle δ (dissimilarité) comme suit :

$$\forall x, y \in O, \delta_M(x, y) = (1/p) \sum \delta(x_i, y_i)^p$$

où p est un paramètre réel. (p =2 : distance euclidienne, p= ∞ : distance de Chebichev)

Définition 6 (somme pondérée) : Soit O un ensemble d'objets qui peuvent être analysés selon n dimensions. La distance somme pondérée δ_S entre deux objets x et y est définie en fonction de la distance traditionnelle δ (dissimilarité) comme suit :

$$\forall x, y \in O, \delta_S(x, y) = \Sigma \omega_i * \delta(x_i, y_i)$$

IV.5.4.5. Découverte de mappings

Selon l'avis de plusieurs auteurs, la découverte de mappings constitue l'un des points fondamentaux de l'intégration sémantique [Kalfoglou et Schorlemmer 2003-b], [Noy 2004], [KnowledgeWeb 2004]. La découverte de mappings consiste à chercher des correspondances entre les ontologies. Cette recherche peut se faire manuellement, de façon interactive, de façon semi automatique, ou de façon automatique.

On distingue principalement deux approches de découvertes ainsi qu'une panoplie de méthodes pouvant être utilisées pour chacune de ces approches.

IV.5.4.5.1. Approches de découverte mappings

Fondamentalement, il existe deux types d'approches pour la découverte de mappings [Noy 2004] :

- la découverte basée sur une ontologie partagée;
- et la découverte basée sur des techniques heuristiques ou l'apprentissage.

La première approche repose sur l'utilisation d'une ontologie supérieure et qui est ensuite spécialisée en fonction des spécificités de chaque application. Dans ce cas, la recherche de correspondances entre deux ontologies est alors facilitée par cette ontologie supérieure partagée. La seconde approche comprend des techniques basées sur des heuristiques ou des techniques d'apprentissage qui sont utilisées dans le processus de découverte de mappings entre les ontologies. Chacune de ces approches peut exploiter un certain nombre de méthodes de mappings.

IV.5.4.5.1.1 Découverte basée sur une ontologie partagée

L'objectif de cette approche consiste à utiliser pour la découverte des mappings une ontologie supérieure partagée, le plus souvent une ontologie standard et qui peut être facilement extensible afin de prendre en charge la spécificité des applications des utilisateurs. Les exemples typiques d'ontologies utilisées dans ce cadre sont notamment :

- SUMO (Suggested Upper Merged Ontology) [Niles et Pease 2001] : est un effort du groupe de travail IEEE sur les standards d'ontologies supérieures. L'ontologie SUMO définit des concepts de haut niveau tels que Object, ContinousObject, Process, Quantity, Relation, etc.
- DOLCE [Gangemi et al. 2003] : est une ontologie formelle développée en tant qu'ontologie supérieure dans le cadre du projet européen WonderWeb. DOLCE a pour but de fournir un framework de référence commun aux ontologies WonderWeb permettant ainsi un partage d'information.
- PSL (Process Specification Language) [Gruninger 2003] : est considérée comme un standard ISO (International Organization of Standardization). Elle est une ontologie développée au sein du NIST (National Institute for Standards and Technology) dans le but de faciliter l'intégration des informations sur les processus dans l'industrie.

IV.5.4.5.1.2 Découverte basée sur des heuristiques ou l'apprentissage

Cette approche concerne la découverte de mappings ontologiques sans utilisation de vocabulaire en commun. D'une manière générale, les approches basées sur des techniques heuristiques utilisent des techniques exploitant les composants lexicaux et structurels des définitions des concepts afin de chercher des correspondances entre les concepts. Dans le cadre des approches basées ontologies (on se restreint au domaine ontologique), l'approche basée sur des heuristiques exploite la sémantique véhiculée par les caractéristiques des ontologies à mapper et qui peuvent être [Noy 2004] :

- noms des concepts ou la description des concepts en langue naturelle
- la taxonomie des concepts (relations is-a)
- les autres relations (lien part-of, instance-of, etc.)
- définitions des propriétés (contraintes: domaines, rangs, restrictions)
- les instances des concepts
- la description des classes (outils basés DL).

IV.5.4.5.2. Méthodes de découverte de mappings

Il existe dans la littérature une multitude de méthodes de découverte de mappings [Sekt 2004][KnowledgeWeb 2004]. [KnowledgeWeb 2004] propose une classification des principales méthodes. La figure IV.33 représente un extrait des principales méthodes utilisées, permet de catégoriser ces méthodes selon deux perspectives : le type de la technique de mapping (lecture descendante) et le type des objets manipulés (lecture ascendante).

La classification selon le type de technique de mapping utilisée permet de distinguer deux grandes catégories de méthodes : les méthodes individuelles et les méthodes combinées.

- Les méthodes individuelles reposent sur l'utilisation d'une seule technique, et on distingue les méthodes basées sur des instances, et les méthodes basées sur des schémas.
    - les méthodes basées sur des instances: ces méthodes exploitent les instances des bases de données pour définir des mappings en se basant fondamentalement sur des similarités linguistiques;
    - les méthodes basées sur des schémas : ces méthodes exploitent les schémas de bases de données pour définir des mappings en utilisant principalement des techniques heuristiques ou des techniques formelles pour calculer le rapprochement entre les éléments des schémas.
- Les méthodes combinées permettent d'associer plusieurs techniques en vue de définir les mappings. On distingue les méthodes hybrides et les méthodes composites.
    - Les méthodes hybrides sont des méthodes basées sur des techniques qui sont considérées comme une combinaison de deux ou plusieurs approches individuelles;
    - Les méthodes composites sont celles qui sont basées sur un assemblage (séquence) de deux ou plusieurs méthodes. Les méthodes hybrides ne sont pas des méthodes composites dans la mesure où elles sont élémentaires non décomposables.

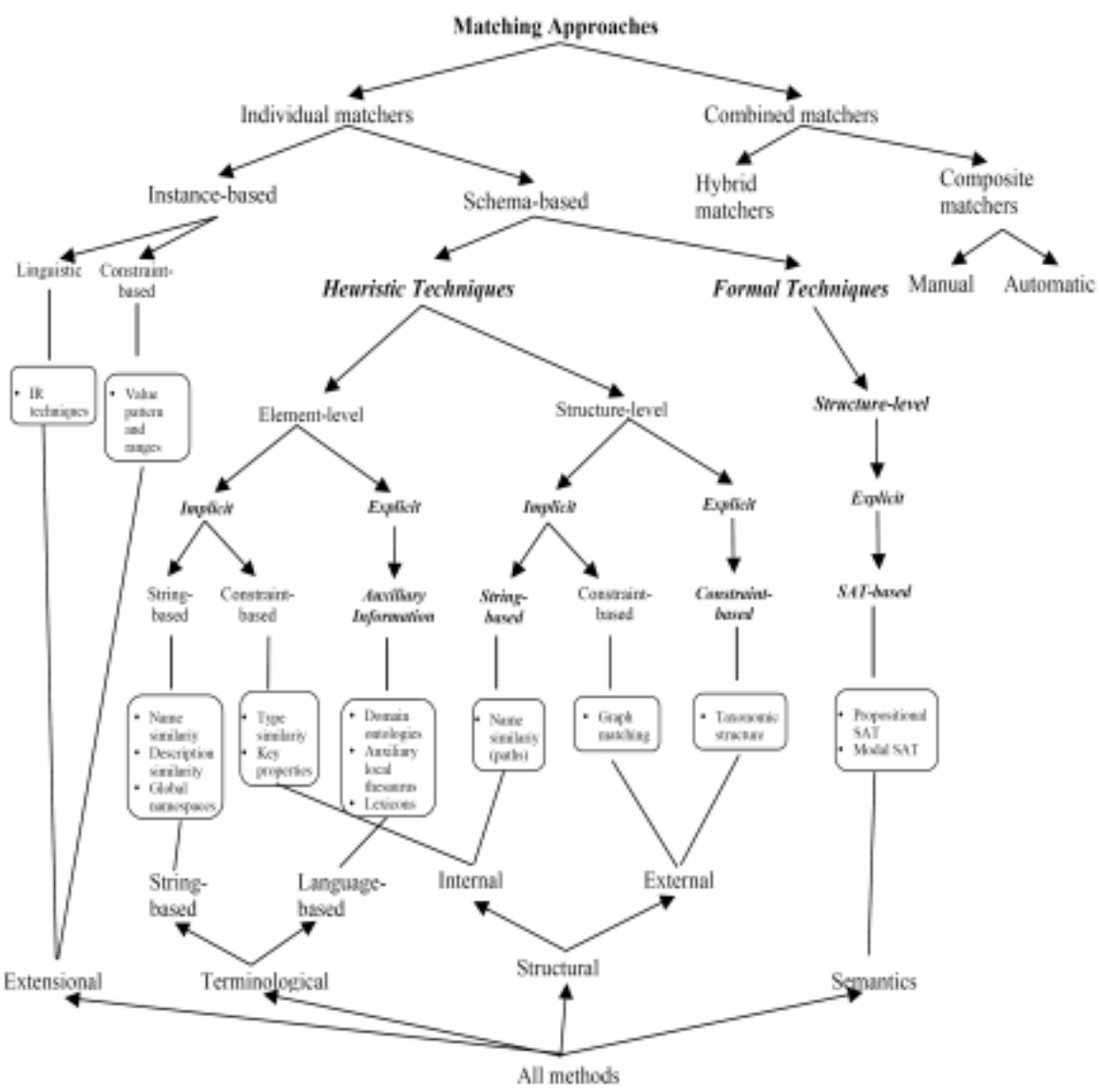

Figure IV.33. Taxonomie des principales méthodes de mapping d'ontologies [KnowledgeWeb 2004]

La seconde classification permet de décomposer les méthodes de mappings en : méthodes terminologiques, structurelles, extensionnelles, et sémantiques.

- Les méthodes terminologiques reposent principalement sur des comparaisons de texte (chaînes de caractères). Elles peuvent s'appliquer aux noms, aux commentaires, aux propriétés des concepts afin de découvrir les concepts similaires. Parmi cette catégorie, on distingue des méthodes basées sur des textes, ou alors des méthodes basées sur des langages
- Les méthodes structurelles se focalisent sur la comparaison des structures des concepts à mapper. En fonction de la nature interne ou externe des structures à comparer, on peut alors distinguer :
  - o des méthodes structurelles internes : qui utilise la comparaison de structure interne d'un concept (par exemple la comparaison des attributs, des noms, et des types des attributs);
  - o des méthodes structurelles externes : qui utilise la comparaison externe en mettant par exemple en jeu la position des concepts dans leur hiérarchie, le voisinage des concepts, etc.

- Les méthodes extensionnelles se focalisent sur la comparaison des instances des concepts des ontologies à mapper. En fonction de l'intersection des extensions de deux concepts, on peut distinguer
  - o des méthodes de comparaison d'extensions communes : qui considèrent que deux concepts sont similaires si leur intersection se réduit à l'un des concepts (en termes d'instances);
  - o des méthodes de comparaison d'extensions basée sur la similarité : qui reposent sur des techniques de calcul de similarité entre les instances des deux concepts à mapper;
  - o des méthodes de comparaison basée sur l'appariement : qui considèrent que les éléments à comparer sont ceux qui présentent des similarités.
- Les méthodes sémantiques sont très souvent des méthodes qui se basent sur des modèles théoriques et se focalisent sur des techniques déductives exploitant très souvent la logique de description (test de subsumption), SAT - SATisfiabilité propositionnelle (calcul propositionnel) ou encore SAT modale (calcul des prédicats).
- En plus de ces méthodes, il y a lieu d'ajouter les méthodes d'apprentissage. Ces méthodes exploitent les techniques développées dans le domaine d'apprentissage comme l'analyse formelle des concepts (Formal Concepts Analysis) [Stumme et Madche 2001], apprentissage Bayesien [Berlin et Motro 2002] ou les réseaux de neurones [Li et Clifton 1994].

IV.5.4.6. Outils de mapping

Les différents types de méthodes précédemment présentés ne sont pas exclusifs dans la mesure où plusieurs d'entre eux peuvent être mis en œuvre simultanément. Le tableau suivant (tableau IV.2) permet de résumer le croisement des principaux outils de mappings avec les principales méthodes présentées.

Les colonnes de ce tableau correspondent aux types de méthodes de mappings précédemment présentés :

- T : Terminologique
- TS : Terminologique basée Texte,
- TL : Terminologique basée Langage (TL));
- I : Structurelle Interne;
- S : Structurelle Externe
- ST : Structurelle Terminologique,
- SC : Structurel Cyclique),
- E: Extensionnelle,
- M: Sémantique (basée Modèle),
- U : Interaction avec l'Utilisateur.

System	T	TS	TL	I	S	ST	SC	E	M	U
Multikat	x		x	x	x	x				
FCA-Merge						x		x		
IF-map						x		x		
APrompt	x			x	x	x				x
Cupid	x	x	x	x	x	x				
QOM	x	x		x	x	x	x	x		
OLA	x	x		x	x	x	x	x		
Rondo		x			x		x			x
T-tree					x			x		
S-match	x	x	x			x			x	
Buster				x		x			x	
Glue								x		
Chimerae	x	x	x						x	x
Artemis	x		x		x					
Coma	x	x			x					x
Asco	x	x	x			x				
MoA			x							x
Dogma	x	x	x							
ArtGen			x							
Bibster	x	x		x		x				
KILT									x	

Tableau IV.2. Panorama des principaux outils de mapping (adapté de [KnowledgeWeb 2004])

Nous avons choisi seulement de présenter succinctement quelques exemples d'outils qui nous paraissent les plus représentatifs, à savoir : PROMPT, FCA-Merge, IF-Map, et GLUE.

Le système PROMPT [Noy et Musen 2003] est développé à l'origine afin de supporter la fusion d'ontologies. Il permet de stocker les mappings entre des ontologies sources créés par le système et par l'utilisateur. La découverte de mappings repose sur des caractéristiques lexicales (méthodes terminologiques) et structurelles (méthodes structurelles), ainsi que les suggestions de l'utilisateur durant le processus de fusion en vue d'établir les mappings. Un autre algorithme utilisé dans PROMPT est ANCHOR-PROMPT qui traite l'ontologie comme un graphe dont les classes sont des nœuds et les slots sont des liens. L'algorithme analyse les chemins dans le sous-graphe limité par les ancres et détermine les classes qui apparaissent fréquemment dans des positions similaires et dans des chemins similaires.

L'outil FCA-Merge (Formal Concept Analysis - Merge) [Stumme et Madche 2001] permet de comparer deux ontologies qui possèdent un ensemble d'instances partagées ou un ensemble de documents annotés. Basé sur des techniques formelles d'analyse des concepts (et aussi des techniques extensionnelles et structurelles), cet outil permet de prendre en charge les équivalences ainsi que les liens de spécialisation. Les résultats de cette méthode peuvent être utilisés par un cogniticien afin d'effectuer la fusion d'ontologies.

IF-Map [Kalfoglou et Schorlemmer 2003] permet d'identifier automatiquement les mappings en se basant sur la théorie des flux d'information en exploitant des techniques structurelles et extensionnelles. IF-Map permet de générer à partir de deux ontologies un isomorphisme logique (mapping ontologique), et qui sont ensuite traduits en mappings en utilisant la théorie des flux.

GLUE [Doan et al. 2002] est un exemple de système qui emploie des techniques d'apprentissage afin de découvrir les mappings. GLUE utilise de multiples apprenants permettant d'exploiter l'information dans les instances et la taxonomie de l'ontologie. Il utilise ensuite un modèle probabiliste afin de combiner les résultats des différents apprenants.

IV.5.4.7. Représentation des mappings

Il existe différentes manières de représenter les mappings. [Noy 2004] propose la classification suivante :

- représentation déclarative des mappings (en tant qu'instances dans une ontologie);
- définition d'axiomes de transformation en utilisant la logique du premier ordre;
- utilisation de vues pour décrire des mappings entre une ontologie globale et une ontologie locale.

La représentation déclarative permet de représenter les mappings en tant qu'instances dans une ontologie de mappings. Le mapping ontologique de Crubézy [Crubézy et Musen 2003] et l'ontologie de mapping sémantique du framework MAFRA [Maedche et al. 2002] en sont deux exemples qui exploitent cette technique. Un mapping entre deux ontologies constitue un ensemble d'instances de classes dans l'ontologie de mappings.

Le système OntoMerge [Dou et al. 2003] met en oeuvre un mécanisme de représentation des mappings en exploitant la technique des axiomes de transformation qui permet de mettre en correspondance les classes et les propriétés des deux ontologies sources à mapper. Dans ce système, le vocabulaire des deux ontologies se trouve dans deux namespaces XML et les axiomes permettent de représenter les règles de transformation des concepts d'une ontologie dans une autre ontologie. Les deux ontologies sont ensuite traitées comme une seule ontologie par un moteur d'inférence.

Enfin, nous utilisons la notion de vue, de façon similaire à ce qui se fait au niveau de l'intégration de données, afin de décrire des mappings ontologiques, et ce avec les deux variantes global-as-view (GAV) et local-as-view (LAV). Le framework OIS [Calvanese et al. 2001] qui constitue un exemple typique de cette approche de représentation, exploite une ontologie globale pour accéder aux ontologies locales. L'ontologie globale et les ontologies locales sont toutes décrites en utilisant la logique de description. Les mappings sont définis comme des vues sur l'ontologie globale ou sur les ontologies locales. Un prédicat d'une ontologie est défini comme une requête sur les prédicats d'une autre ontologie.

IV.5.4.8. Raisonnement avec des mappings

La définition de mappings ontologiques ne constitue pas une fin en soi. Ces mappings doivent être exploités pour diverses tâches d'intégration comme la transformation de données, le traitement des requêtes, ou la composition de services web, en ne citant que quelques unes. Ceci conduit alors à utiliser les mappings dans des raisonnements.

Le système OntoMerge [Dou et al. 2003], que nous avons introduit précédemment, utilise un raisonnement pour exécuter les tâches liées à la translation d'ontologies. La première tâche est la translation d'instances d'une ontologie en instances d'une autre ontologie conformément au mapping entre les deux ontologies. L'exécution de cette tâche nécessite la création d'une ontologie intégrée incluant l'ontologie source, l'ontologie cible, et les mappings. C'est sur la base de cette ontologie intégrée que le système va effectuer des inférences, et exécute une projection afin de ne retenir que les nouvelles conclusions atteintes et qui référencent exclusivement le vocabulaire de l'ontologie cible. La seconde tâche exécutée par OntoMerge concerne la génération d'extensions (taxonomie).

Les tâches d'intégration dans le cadre du système à base de mapping ontologique de [Crubézy et Musen 2003] introduit dans la section précédente sont au nombre de trois.

La première tâche consiste à translater, en utilisant un interpréteur de mappings et qui exploite l'ontologie de mappings, les données de l'ontologie source vers l'ontologie cible. La seconde tâche consiste à utiliser l'outil PROMPT et fusionne les ontologies source et cible.

Dans le cadre du framework OIS [Calvanese et al. 2001], des moteurs d'inférence basés sur la logique descriptive sont utilisés du fait que les ontologies ont été décrites dans cette logique. Ce framework adresse la tâche générale de traitement d'une requête posée en termes d'une ontologie globale en utilisant les données des ontologies locales.

IV.5.5. Discussions

Comme nous venons de le voir, le mapping d'ontologies constitue à notre sens le meilleur moyen pour mettre en œuvre l'intégration des ontologies. Cette approche constitue le moyen le plus flexible pour lier des domaines sémantiques. Les mappings constituent dans nos travaux un des éléments importants permettant de relier des domaines sémantiques hétérogènes dans les milieux industriels.

L'analyse des différentes approches, outils ainsi que des formalismes de représentations des mappings nous a poussés à nous intéresser plus particulièrement dans le cadre de nos travaux aux mappings semi-automatiques. Pour cela, nous avons décidé d'utiliser l'outil PROMPT qui permet de représenter les mappings sous forme déclarative. Ce qui leur permet d'être utilisés par un moteur d'inférence. Nous allons présenter dans les chapitres suivants une architecture sémantique spécifique définissant une typologie spécifique de mappings appropriée aux situations complexes de grandes entreprises industrielles.

IV.6. Intégration sémantique des applications

Comme il a été déjà évoqué, les ontologies sont de plus en plus sollicitées dans le domaine d'intégration des systèmes d'information [Bruijn 2003]. Dans la section précédente, nous avons étudié l'aspect intégration des ontologies. A présent, nous allons étudier l'intégration des systèmes d'informations à travers l'utilisation des ontologies.

Comme nous l'avons introduit au chapitre II, l'intégration des applications d'entreprise peut intervenir au niveau de plusieurs couches et principalement : au niveau des données, des traitements ou encore au niveau du processus. Cela demeure également valable dans le cadre des approches d'intégration sémantique, et en particulier dans le cas des approches d'intégration basées sur les ontologies.

Quelle que soit la couche retenue, il est possible de caractériser les approches d'intégration à travers trois activités principales, qui sont à notre connaissance les plus pertinentes, et qui sont comme suit [Interop 2005][26] :

- Description : est l'activité qui permet de décrire sémantiquement les aspects du système d'information;
- Découverte : est l'activité qui permet de rechercher les informations et les fonctionnalités qui peuvent être fournies par le système;

[26] Dans le déliverable D 8.1, on rajoute une activité supplémentaire qui est l'activité exécution.

- Composition : est l'activité qui permet de comparer et de combiner des informations, des fonctionnalités et des processus.

IV.6.1. Intégration sémantique par les données

L'intégration sémantique des applications au niveau des données consiste à définir une ou plusieurs ontologies permettant de décrire la sémantique du domaine et utiliser ces ontologies dans des mécanismes de conciliation et d'unification de la sémantique afin de résoudre les conflits sémantiques liés aux données échangées et/ou manipulées par les applications. De telles approches sont souvent appelées par la communauté des bases de données approches de médiation de contexte [Benslimane et al. 1999]. La résolution de conflits se base généralement sur des mécanismes de classification et de comparaison d'entités d'ontologie(s).

IV.6.1.1. Description

L'objectif principal de la description est de relier les ontologies aux données du système d'information. A cet effet, les approches génériques utilisées sont principalement [Wache et al. 2001] :

- La ressemblance structurelle : elle consiste à créer des mappings entre le schéma de la base de données et l'ontologie. Par exemple, c'est l'approche utilisée dans les systèmes SIMS [Arens et al. 1993] et TSIMMIS [Chawathe et al. 1994];
- La définition des termes : elle consiste à utiliser une ontologie pour ensuite définir les termes contenus une base de données. Par exemple, le système BUSTER est basé sur ce type d'approche;
- L'enrichissement structurel : elle est considérée comme la combinaison des approches précédentes dans le sens où elle permet d'enrichir les éléments structurels de la base de données tout en utilisant des définitions de termes. Par exemple, le système OBSERVER [Mena el al. 1996] et KRAFT [Kraft 2004] sont basés sur ce type d'approche de description sémantique.
- La méta-annotation : elle consiste à rajouter des annotations sémantiques aux bases de données. Par exemple, le système OntoBroker est basé sur l'utilisation des annotations sémantiques.

Un autre point important de la description est l'architecture d'ontologies adoptée et qui a été traité au paragraphe § IV.4. Certaines approches adoptent des architectures mono-ontologie (tel est le cas de SIMS [Arens et al. 1993]) alors que d'autres adoptent des architectures multi-ontologies (par exemple, OBSERVER [Mena et al. 2000]), voire des architectures hybrides (par exemple, le système MECOTA [Wache et al. 1999] et le projet BUSTER [Stuckenschmidt 2000]).

IV.6.1.2. Découverte.

La découverte d'information consiste généralement à rechercher des liens entre les informations contenues dans différents systèmes. Le principe général repose sur l'utilisation des mesures de similarité qui peuvent exister entre les termes des ontologies qui formalisent la sémantique des informations. Il existe une multitude d'approches et d'algorithmes qui peuvent être mis en œuvre pour effectuer la comparaison et la mesure de similarité. Des exemples d'approches et de méthodes ont été déjà introduit dans le cadre de l'étude des approches de découverte de mappings (cf. § IV.5.4.5.2).

Il est fondamental de noter que généralement, des langages de requêtes peuvent être mis en œuvre par les applications clientes pour formuler leur requête et des les exécuter. La formulation des requêtes se fait au moyen des ontologies définies qui jouent alors le rôle de modèle canonique. Des langages RDQL (RDF Query Language) ou OQL (OWL Query Language) basés respectivement sur RDF et OWL sont souvent utilisés pour formuler ces requêtes.

IV.6.1.3. Composition

Une fois que des similarités entre informations sont trouvées, des mécanismes de composition sont ensuite réalisés. Ils permettent de combiner ces informations pour former des éléments plus complexes. Cette activité est très souvent appelée intégration de données par la communauté des bases de données.

IV.6.2. Intégration par les traitements

En ce qui concerne l'intégration sémantique des applications au niveau des traitements, il existe principalement les travaux basés sur les services Web sémantiques [McIlraith et al. 2001][Bussler 2003][Kellert et Toumani 2003]. La notion de service Web sémantique a pour objectif de combler certaines lacunes des Services Web traditionnels liées à la prise en compte de l'aspect sémantique. Les initiatives les plus pertinentes autour du concept de Service Web sémantique sont principalement : OWL-S [OWLSC 2004], WSMF [Fensel et Bussler 2002][WSMF 2005] WSMO [Bruijn et al. 2005][WSMO 2005], METEOR-S [Meteor-s 2005][Cardoso et Sheth 2004] et IRS-II [Motta et al. 2003].

OWL-S fournit une ontologie générique de services permettant de décrire les capacités et les propriétés des Services Web. WSMF est une initiative qui fournit un cadre générique pour supporter le concept de Service Web sémantique. WSMO est une initiative qui raffine WSMF. METEOR-S est une initiative basée sur l'extension de certains standards des Services Web tels que WSDL et UDDI. IRS-II est une initiative basée sur les tâches de résolution de problèmes.

IV.6.2.1. Description

La description des services porte principalement sur les capacités offertes par le service, la qualité de service ainsi que la représentation du processus du service. Deux formes basiques[27] de description de la capacité d'un service sont généralement mises en oeuvre. La première consiste à utiliser une description explicite en se basant sur une ontologie de tâches, où chaque tâche du service est décrite par un concept de l'ontologie, et où chaque capacité d'un service est modélisée comme un ensemble de tâches. La seconde approche est implicite et se base sur les états de la transformation, c'est à dire en se basant sur la formalisation des inputs, des outputs, des pré-conditions et des post-conditions. L'approche explicite est plus simple et plus naturelle, mais peut conduire à des ontologies volumineuses. La seconde approche est par contre moins intuitive, mais elle est plus compacte. Un exemple de technologie industrielle qui repose sur l'approche implicite de représentation est l'ontologie OWL-S, ou encore le formalisme WSMO.

[27] D'autres approches existent pour l'enrichissement sémantique des services web. Elles reposent sur l'enrichissement ou la surcharge des documents WSDL. Il s'agit en particulier de l'approche de [Ogbuji 2000][Sivashanmugam et al. 2003], [Peer 2002], etc.

IV.6.2.2. Découverte

L'objectif de la découverte est de pouvoir rechercher dynamiquement et automatiquement des services pour un besoin donné. Par exemple, on veut chercher un service permettant d'effectuer un type particulier d'analyse sur un échantillon de données.

Traditionnellement, le processus de découverte de services comprend généralement les étapes suivantes [Cardoso et al. 2002] :

- Définir un service cible (requête) en utilisant soit une description implicite ou explicite du service souhaité;
- Lier les éléments de la requête (input, output, etc.) à des ontologies;
- Identifier les services objets potentiels qui satisfont le service cible;
- Calculer les distances de chaque service objet au service cible.

Plusieurs approches proposées dans la littérature tentent d'améliorer le processus de découverte en procédant à l'enrichissement de la description du service [Ouggi 2000][Meteor-s 2005][Paolucci et al. 2002] ou à la pondération des résultats [Noia et al. 2003][Stojanovic et al. 2003].

IV.6.2.3. Composition

La composition des services (et des processus) a pour objet de combiner des services pour former des services plus complexes. Parmi les contributions à ce sujet, nous pouvons citer celles de [Rao et Su 2004][Casati et al. 2000][Papazoglou et Heuvel 2006][Medjahed 2004][Sirin et al. 2002][Lammermann 2003][Yang et Papazoglou 2002][Casati et al. 2001] qui traitent de la problématique d'assemblage de services dans le contexte du B2B.

IV.6.3. Intégration par les processus

L'intégration des applications par le processus concerne aussi bien la problématique d'intégration inter-processus que la problématique d'orchestration par un processus des applications d'entreprise. Nous ne nous focalisons dans le cadre de nos travaux que sur l'aspect orchestration des applications et qui peut être pris en charge par la notion de composition de services présentée ci-dessus.

IV.6.3.1. Description

Plusieurs formalismes peuvent être utilisés pour décrire sémantiquement les processus. Le premier niveau de description peut concerner les différents langages de modélisation de processus qui peuvent être mis en œuvre tels que BPML, BPEL4WS, etc. Ces langages offrent un certain nombre de constructeurs permettant de définir des processus. Ce niveau de description est généralement faible dans la mesure où il est plus syntaxique que sémantique. Pour cette raison, d'autres auteurs ont proposé des ontologies plus complexes pour la description des processus. L'exemple majeur de telles ontologies est l'ontologie proposée par l'institut NIST[28] dans le cadre du langage PSL (Process Specification Language) [Nist 2005].

[28] National Institute of Standards and Technology.

IV.6.3.2. Découverte

La découverte de processus est un domaine qui est peu traité. Le peu d'approches qui s'intéressent au sujet sont notamment l'approche ebXML [ebXML 2001] et l'approche ECIMF [ECIMF 2005] qui s'intéressent à la problématique d'intégration de processus dans le contexte B2B. Mais très souvent le problème de découverte est appréhendé sous l'angle de découverte de services composites (§ IV.6. 2.2).

IV.6.3.3. Composition

La composition de processus est un domaine de recherche récent, et il n'existe pas beaucoup de travaux traitant de cette problématique. De même que pour la découverte, ebXML est considéré comme étant l'approche majeure qui permet de définir des compositions de processus à travers le ebXML-BPSS (Business Process Specification Schema). BPSS permet de définir des constructeurs permettant de mettre en œuvre des scénarios de collaboration B2B.

IV.6.4. Discussions

L'intégration sémantique des applications d'entreprise est un domaine de recherche très actif. Il peut concerner, comme nous venons de le voir, aussi bien l'intégration par les données, par les traitements que par les processus. Une analyse plus fine permet de déduire que l'orientation services peut constituer un élément fédérateur pour l'ensemble de ces approches. En effet, l'orientation service est un paradigme qui à notre sens permet l'unification dans les approches d'intégration des systèmes d'information. Ceci nous a donc conduit à nous focaliser principalement sur les services web sémantiques, qui à notre sens constituent l'un des moyens les plus efficaces pour mettre en œuvre l'intégration des systèmes dans le contexte industriel. Pour plus de précision sur cette approche, nous avons décidé de consacrer totalement la prochaine section à l'étude des services sémantiques.

IV.7. Intégration sémantique des services

Dans la section précédente, nous avons présenté un panorama des principales catégories d'approches qui peuvent être mises en œuvre dans le cadre d'intégration sémantique des systèmes d'information industriels, et nous avons retenu l'approche orientée services sémantiques comme l'une des approches les plus efficaces qui est basée à la fois sur le dynamisme et la sémantique (figure IV.34). Nous allons dans cette section, détailler les approches orientées services sémantiques. Et nous allons en particulier étudier les approches les plus représentatives des services sémantiques et qui sont à notre sens : OWL-S, WSMF, WSMO, METEOR-S et IRS-II. Cette étude est basée plus particulièrement sur [Cardoso 2006][KnowledgeWeb 2004][Cabral et al. 2004][Bussler 2004][Bussler 2003][Bobillo et al. 2005].

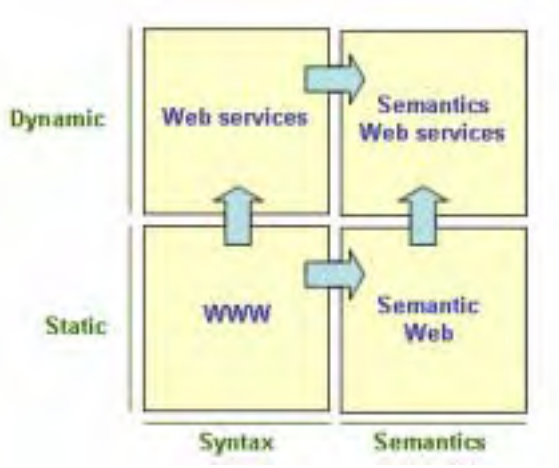

Figure IV.34. La nature des services web sémantiques [Cardoso 2006]

IV.7.1. OWL-S

OWL-S, anciennement DAML-S est une ontologie pour la description sémantique des services web qui a été développée dans le cadre du projet DAML. OWL-S se base sur le langage standard OWL [OWLSC 2004] qui permet de représenter des ontologies de façon standardisées sur le web. L'objectif de OWL-S est de formaliser de façon non ambiguë les web services de manière à ce qu'un agent logiciel puisse exploiter automatiquement les informations concernant ces services. Les bénéfices de l'emploi d'une ontologie de services peuvent être multiples, mais le plus intuitif concerne sans doute l'amélioration de la recherche de services et également leur composition.

Comme illustré en figure IV.35, OWL-S distingue trois éléments principaux qui interviennent dans la description d'un service :

- ServiceProfile : exprime ce que le service propose ainsi que les prérequis que son emploi puisse imposer;
- ServiceModel : exprime le fonctionnement du service;
- ServiceGrounding : exprime comment le service peut être utilisé.

Figure IV.35. Ontologie OWL-S [OWLSC 2004]

Dans l'approche OWL-S, la partie profil contient l'information la plus utile pour la découverte et la composition de services du fait qu'elle est utilisée à la fois par les fournisseurs de services pour la publication et par les clients pour l'interrogation. L'exemple de la figure IV.36 qui est un cas de ServiceProfile, représente les inputs, les

outputs, les pré-conditions ainsi que les effets du service de réservation de la société congo.com[29].

```
...
<profile:hasInput
  rdf:resource=
    "http://www.daml.org/services...
                #ExpressCongoBuySignInInfo" />

<profile:hasOutput
  rdf:resource=
    "http://www.daml.org/services...
              #ExpressCongoOrderShippedOut" />

<profile:hasPrecondition
  rdf:resource=
  "http://www.daml.org/services...
                              #AcctExists" />
...


<profile:hasEffect
  rdf:resource=
  "http://www.daml.org/services...
              #ExpressCongoOrderShippedEff"/>
...
```

Figure IV.36. Exemple de ServiceProfile [Cordoso 2006]

IV.7.2. WSMF

WSMF [Fensel et Bussler 2002] est un framework complet de description de services proposé dans le cadre du projet européen SWWS[30]. Il constitue une extension du langage UPML (Unified Problem-Solving Method Development Language) [Fensel et al. 2003] et il est centré autour de deux principes complémentaires qui sont :

- Un découplage fort des différents composants servant à réaliser une application de e-commerce
- Un service de médiation fort qui autorise quiconque à communiquer avec tout le monde, de manière évolutive

L'architecture de WSMF est constituée de quatre éléments principaux (figure IV.37) : les objectifs, les ontologies, les descriptions des services web et les médiateurs.

Figure IV.37. Architecture WSMF [Fensel et Bussler 2002]

[29] http://daml.semanticweb.org/services/owl-s/1.0/
[30] http://swws.semanticweb.org

Les objectifs (décrits dans le répertoire d'objectifs) permettent de définir les problèmes qui doivent être résolus par les web services. Ils incluent des pré-conditions et des post-conditions. Les ontologies fournissent la terminologie et la sémantique utilisée par les autres éléments. Les descriptions des services web définissent les différents aspects liés aux services web (input, output, opérations, etc.). Les médiateurs permettent de résoudre les problèmes liés à l'intégration entre les services web.

Nous devons noter que WSMF est un modèle conceptuel et ne définit pas de sémantique concrète particulière pour la représentation des services. Selon la spécification publique de WSMF, ceci peut être réalisé à travers l'utilisation de WSFL ou de OWL-S, et de PSL (Process Specification Language) [KnowledgeWeb 2004].

IV.7.3. WSMO

WSMO (*Web Service Modeling Ontology*) [Bussler et al. 2004] est une extension et un raffinement WSMF. C'est un langage formel et une ontologie qui permet de décrire des aspects variés des Web services sémantiques. La spécification de WSMO repose sur un modèle stratifié qui inclut la définition de trois différentes ontologies : WSMO Lite (ontologie de base), WSMO Standard (ontologie intermédiaire) et WSMO Full (qui contient tous les concepts définis dans WSMO).

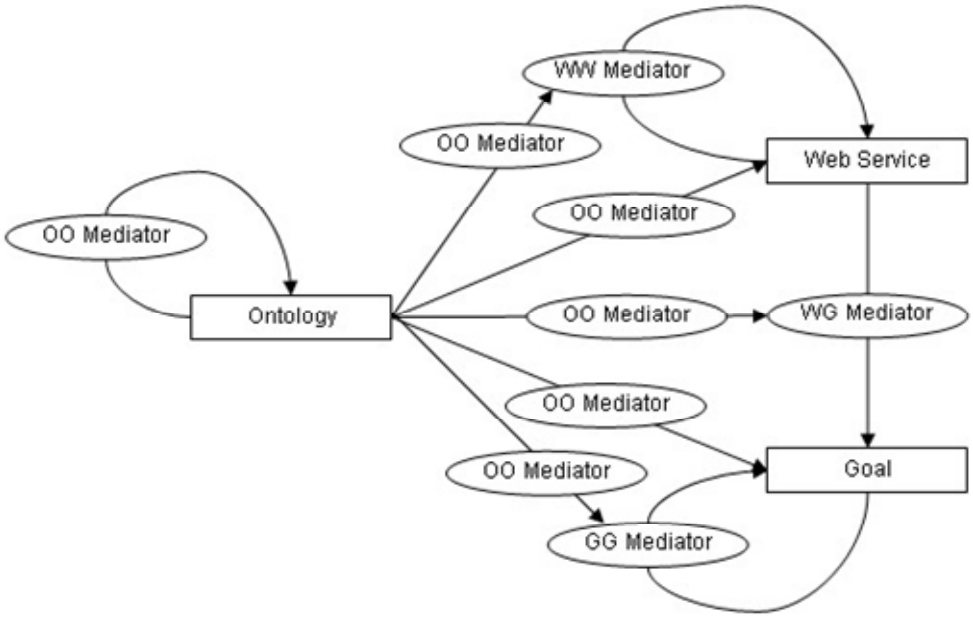

Figure IV.38. Architecture WSMO [WSMO 2005]

L'architecture de WSMO repose sur l'utilisation de médiateurs (figure IV.38). On distingue deux clases de médiateurs: des raffineurs (refiners) et des ponts (bridges). Les raffineurs sont utilisés pour générer de nouveaux composants comme des raffinements de ceux qui existent. On distingue des raffineurs d'objectifs (ggMediator) et les raffineurs d'ontologies (ooMediator). Les raffineurs d'objectifs (*goal refiners*) qui sont utilisés pour raffiner un but source dans un but cible et des raffineurs d'ontologies (*ontology refiners*) qui sont utilisés pour résoudre les éventuels problèmes lors de l'importation d'ontologies. Les ponts, seconde classe de médiateurs, permettent de faire interopérer deux composants donnés. On distingue des ponts entre services et objectifs (wgMediator) et des ponts entre services (wwMediator). Les ponts entre services et

objectifs (*service-to-goal bridges*) résolvent les problèmes entre services et objectifs, tandis que les ponts entre services (*service-to-service bridges*) effectuent le lien entre deux services Web.

Dans l'approche WSMO, on distingue deux composants principaux qui sont WSML (Web service Modeling Language) [Bruijn et al. 2005] et WSMX (Web Service Execution Environment) [Bussler et al. 2004].

WSML fournit un langage formel pour la description des éléments définis dans l'architecture WSMO. WSML est basé sur différents langages logiques qui sont la logique de description, la logique de premier ordre et la logique de programmation. Ce langage est structuré en cinq sous ensembles : WSML-Core, WSML-DL, WSML-Light, WSML-Rule et WSML-Full. Chacun de ces sous-ensembles fournit un niveau d'expressivité croissante.

WSMX est un environnement d'exécution qui permet d'offrir un certain nombre de modules utilisables en run-time permettant de prendre en charge la découverte, la sélection, la médiation et l'invocation des services sémantiques.

La figure IV.39, tirée de [Cordoso 2006] représente un exemple de description sémantique d'un service. Il s'agit du même que précédemment, c'est à dire du cas d'un service de réservation de billets d'avion.

```
capability BookTicketCapability
  sharedVariables                    {?creditCard,
?initialBalance,   ?trip,   ?reservationHolder,
?ticket}
  precondition
    definedBy
      ?reservationRequest[
        reservationItem hasValue ?trip,
        reservationHolder              hasValue
?reservationHolder
      ] memberOf tr#reservationRequest and
      ?trip memberOf tr#tripFromAustria and
      ?creditCard[balance              hasValue
?initialBalance
            ] memberOf po#creditCard.
  assumption
    definedBy
      po#validCreditCard(?creditCard)and
        (?creditCard[type              hasValue
"PlasticBuy"] or
        ?creditCard[type               hasValue
"GoldenCard"]).
...
  posteondition
...
  effect
...
```

Figure IV.39. Un exemple de description avec WSML [Cardoso 2006]

IV.7.4. METEOR-S

METEOR-S (METEOR for Semantic Web Services) est une initiative du laboratoire LSDIS de Géorgie et qui a pour objectif de fournir des services web sémantiques dans le cadre du projet METEOR (Managing End-To-End OpeRations). Il a précisément pour but d'intégrer les standards des services Web (BPEL4WS, WSDL et UDDI) avec les

technologies du Web sémantique pour l'annotation, la découverte, la composition, la qualité de service et l'exécution de services Web.

Le projet METEOR-S s'est développé selon trois phases principales qui ont permis d'introduire les trois concepts importants de cette initiative :

- mise en place de l'infrastructure : il consiste à installer une infrastructure de découverte sémantique définie au dessus d'un registre UDDI et qui est appelée MWSDI (*METEOR-S Web Service Discovery Infrastructure*);
- annotation sémantique : il définit l'outil MWSAF (METEOR-S Web Service Annotation Framework) et qui permet d'enrichir sémantiquement les services web en utilisant une extension enrichie de WSDL appelée WSDL-S (WSDL Semantics). Le rôle de WSDL-S [WSDL-S 2005] est d'ajouter un niveau sémantique aux services web à travers l'enrichissement des fichiers WSDL mais également du registre enrichi UDDI;
- la composition et l'exécution de services : il a pour rôle d'assembler des services pour composer des services complexes en se basant sur BPEL4WS. L'outil se chargeant de cette tâche s'appelle MWSCF (METEOR-S *Web Service Composition Framework*).

La figure IV.40 illustre l'architecture de METEOR-S qui comporte deux modules: le module front-end et le module back-end. Le module front-end est utilisé pour créer des services web sémantiques en procédant à l'annotation du code source avec des ontologies. La source ainsi enrichie est convertie en descriptions sémantiques en utilisant soit des fichiers WSDL annotés soit des fichiers WSDL-S. Ces derniers sont ensuite publiés dans un registre UDDI enrichi. Le module de back-end offre principalement des fonctionnalités liées à la découverte de services, à la composition de services et à l'exécution et l'orchestration des services composés.

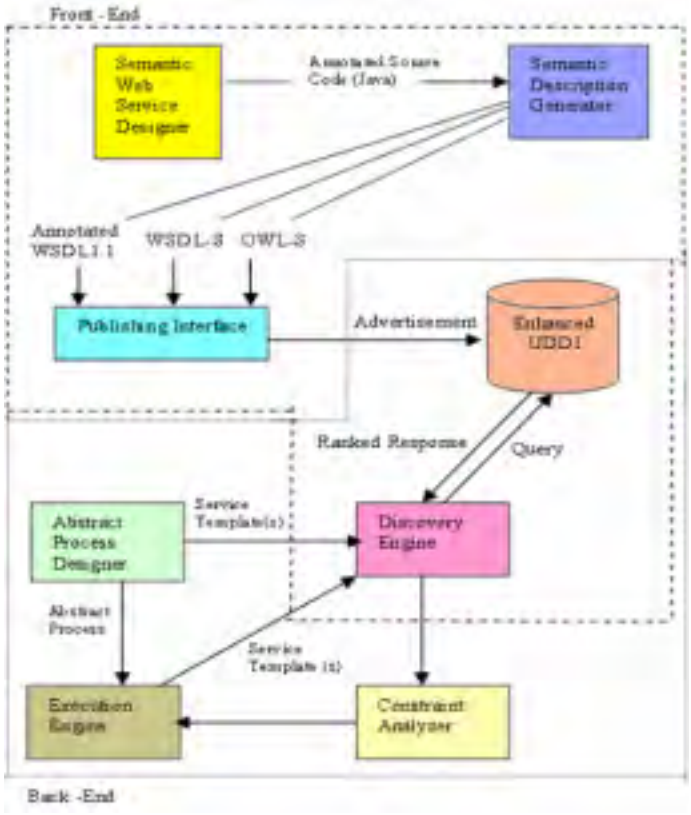

Figure IV.40. Architecture de METEOR-S [Meteor-s 2005]

Le principe d'annotation des services web peut être mieux illustré par la figure suivante (figure IV.41) qui permet de représenter un exemple de fichier WSDL-S. Dans ce fichier, des annotations ont été rajoutées afin de définir des liens avec des ontologies permettant d'expliciter la sémantique des éléments du service web.

```
<?xml version="1.0" encoding="UTF-8"?>
<definitions
      name = "BatterySupplier"
      targetNamespace = "http://lsdis.cs.uga.edu/meteor/BatterySupplier.wsdl20"
      xmlns = "http://www.w3.org/2004/03/wsdl"
      xmlns:tns = "http://lsdis.cs.uga.edu/BatterySupplier.wsdl20"
      xmlns:rosetta = " http://lsdis.cs.uga.edu/projects/meteor-s/wsdl-s/pips.owl "
      xmlns:mep=http://www.w3. rosetta:PurchaseOrderStatusResponse org/TR/wsdl20-patterns>

   <interface name = "BatterySupplierInterface" description = "Computer PowerSupply Battery Buy Quote Order
      Status "
               domain="naics:Computer and Electronic Product Manufacturing" >

      <operation name = "getQuote"  pattern = "mep:in-out"  action = "rosetta:#RequestQuote" >
       <input messageLabel = "qRequest" element = "rosetta:#QuoteRequest" />
            <output messageLabel = "quote" element  = "rosetta:#QuoteConfirmation"  />
       </operation>

       <operation name = "placeOrder"  pattern = "mep:in-out"  action = "rosetta:#RequestPurchaseOrder" >
         <input messageLabel = "order" element = "rosetta:#PurchaseOrderRequest" />
              <output messageLabel = "orderConfirmation" element  = "rosetta:#PurchaseOrderConfirmation" />
       </operation>
   </interface>
</definitions>
```

Figure IV.41. Exemple de description WSDL-S [Meteor-s 2005]

IV.7.5. IRS-II

IRS-II (Internet Reasoning Service), est une structure et une implémentation pour les Web services sémantiques, supportée par le projet AKT (Advanced Knowledge Technologies dans le cadre d'une collaboration interdisciplinaire de recherche regroupant plusieurs universités européennes dont l'université d'Aberdeen, d'Edinburgh etc. IRS-II est conçu comme un moyen de résolution de problèmes en ligne réutilisables [Motta et al. 2003]. Le but principal d'IRS-II est de supporter la découverte et la récupération de services de librairies distribuées sur Internet et de leur configuration semi-automatique, dans le but de réaliser des tâches spécifiques en fonction des exigences des utilisateurs. IRS-II tente d'apporter l'adaptabilité et une médiation flexible entre problèmes et services [Crubézy et Musen 2003].

IRS-II possède deux caractéristiques majeures. Premièrement, IRS-II est construit sur un modèle de connaissance, et deuxièmement IRS-II est basé sur le langage UPML (Unified Problem-Solving Method-Development Language) pour l'annotation des services. L'architecture de IRS-II est illustrée en figure IV.42. Les principaux composants sont le serveur IRS (IRS Server), le fournisseur IRS (IRS Publisher) et le client IRS (IRS Cleint) et qui communiquent à travers le protocole SOAP.

Le serveur ISR comporte les descriptions des services sémantiques à deux niveaux différents. Un niveau de description des connaissances permettant la description des tâches (Task models), du domaine (Domain models) et des méthodes de raisonnement (PSMs - Problem Solving Methods) et ce en se basant sur le framework UPML.

Le fournisseur IRS joue deux rôles principaux dans le framework IRS-II. Premièrement, il permet de lier les descriptions sémantiques aux services web. Deuxièmement, il permet de générer des adaptateurs standards (Java Web Service) qui permettent ainsi d'encapsuler l'hétérogénéité due aux langages utilisés (Lisp, Java, etc.).

Le client IRS est toute application permettant d'interroger le framework pour découvrir des tâches qui correspondent à un problème particulier et ensuite d'exécuter les services web associés à la résolution de ce problème.

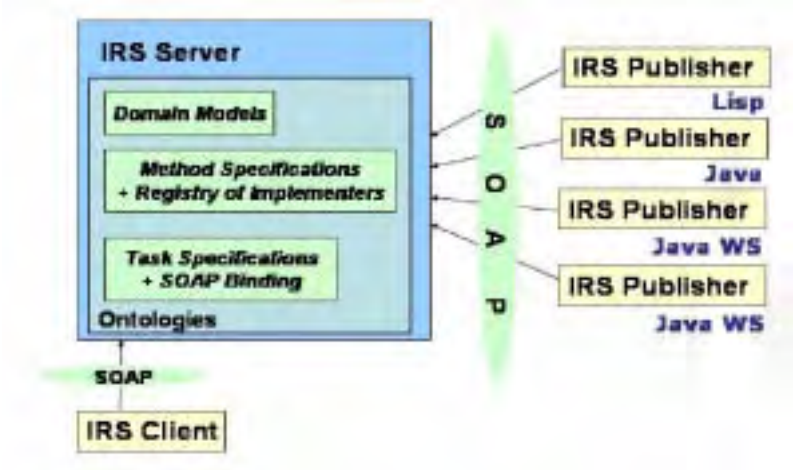

Figure IV.42. Architecture IRS-II [Motta et al. 2003]

Nous avons choisi de reproduire ci-dessous (figure IV.43) un exemple présenté dans [Motta et al. 2003] et qui décrit une méthode de résolution de problèmes (PSM) qui permet de fournir le taux de change bancaire en fonction des monnaies délivrées en entrée.

```
(def-class MM_Bank_exchange_rate_provider (primitive-method)
 ?psm
 ((has-input-role
   :value has-source-currency
   :value has-target-currency)
  (has-output-role
   :value has-exchange-rate)
  (has-source-currency :type european_currency :cardinality 1)
  (has-target-currency :type european_currency :cardinality 1)
  (has-exchange-rate :type positive_number)
  (has-precondition
   :value (kappa (?psm) (stock-available
                         (role-value ?psm has-target-currency))
  (has-postcondition
   :value (kappa (?psm ?sol)
           (= ?sol (the-European-Central-Bank-exchange-rate
                    (role-value ?psm has-source-currency)
                    (role-value ?psm has-target-currency)))))))
```

Figure IV.43. Représentation d'une PSM dans IRS-II [Motta et al. 2003]

IV.7.6. Discussions

Nous avons choisi de résumer, dans le tableau IV.2, les principales caractéristiques des principales approches de services sémantiques présentées précédemment. Les

caractéristiques de ces initiatives sont synthétisées à l'aide de sept critères que nous avons choisi de retenir qui sont, à notre sens, les plus pertinents. Il s'agit, en l'occurrence, de deux critères génériques déjà retenus dans le chapitre III pour l'analyse des approches syntaxiques auxquels nous avons décidé de rajouter quatre autres critères spécifiques aux approches sémantiques. Les deux critères génériques retenus sont comme suit :

- critère *point de vue* qui permet de préciser le niveau d'abstraction (conceptuel, technique) associé à l'approche considérée;
- et le critère *ouverture* qui permet de représenter le degré de standardisation permis par l'approche

Les trois critères supplémentaires spécifiques que nous avons choisis pour l'analyse des approches sémantiques sont :

- le critère *description sémantique* qui permet de préciser la manière avec laquelle les services sont décrits sémantiquement par l'approche étudiée;
- le critère *médiation de services* qui permet de préciser la prise en compte des mécanismes de médiation par l'approche étudiée;
- et le critère *maturité* qui désigne en fait le degré de maturité de l'approche vis à vis de la problématique d'intégration des applications industrielles.

Critères d'analyse	Techniques d'intégration sémantique				
	OWL-S	WSMF	WSMO	METEOR-S	IRS-II
Point de vue (niveau d'abstraction)	conceptuel (basée sur une ontologie générique de services)	conceptuel (basée sur un cadre conceptuel)	conceptuel (basée sur le cadre WSMF)	technique (basée sur l'enrichissement sémantique de WSDL)	conceptuel (basée sur des ontologies tâches et PSM)
Ouverture (standardisation)	très forte (basée sur WSDL, OWL)	faible (basée sur UPML)	faible (basée sur WSML)	forte (basée sur WSDL, mais WSDL-S n'est pas standard)	faible (basée sur UPML et OCML)
Description sémantique	Ontologie générique OWL-S	pas de description mais recommandation d'utilisation de OWL-S- et de PSL	Ontologie WSMO	Annotation des fichiers WSDL	Ontologie PSM et ontologie Tâche
Médiation de services	non définie	non définie	définie mais en cours de développement	non définie	non définie
Maturité relative vis à vis de la problématique d'intégration en industrie	forte	très faible	moyenne	moyenne	moyenne

Tableau IV.3. Récapitulatif des principales approches de services sémantiques

L'analyse du tableau IV.1 permet de retenir un certain nombre de points importants qui sont :

- OWL-S constitue une ontologie générique de services permettant principalement de décrire les services web dans un cadre général. Une des caractéristiques majeure de OWL-S est le fait qu'elle soit compatible avec les standards industriels notamment WSDL, et ce grâce au ServiceGrounding qui permet de définir des mappings entre OWL-S et WSDL. Certains travaux existent quant à la découverte et la composition de services en utilisant OWL-S.
- WSMF est un cadre générique pour la réalisation des services sémantiques. Il constitue donc un framework qui permet d'offrir beaucoup plus un cadre méthodologique qu'une infrastructure opérationnelle.
- WSMO constitue une approche basée sur des fondements conceptuels issus de WSMF pour la réalisation des services sémantiques. Il est beaucoup plus orienté utilisateur dans le sens elle permet d'offrir un langage très proche des utilisateurs. Mais elle demeure toutefois immature du fait qu'il s'agit d'une initiative en cours de développement. De plus, l'une des limites les plus importantes est le fait que cette approche ignore les standards industriels existants, ce qui suppose que des mappings doivent être définis afin de garantir l'intégration avec les standards existants au sein des entreprises.
- METEOR-S constitue une approche basée sur l'extension de standards industriels. Le principe d'enrichissement sémantique est basé principalement sur l'utilisation des fichiers WSDL-S qui sont le résultat de l'annotation sémantique de fichiers WSDL. Bien que cette approche repose sur des standards, il n'en demeure pas moins qu'elle manque d'abstraction dans le sens où elle constitue une approche plus proche du niveau d'implémentation que du niveau conceptuel.
- IRS-II est une approche qui permet de réaliser les services sémantiques à travers le framework UPML. Ceci constitue à la fois son point fort et son point faible. De part sa formalisation UPML est considérée comme un point fort de l'approche IRS-II. Mais il constitue aussi l'un des griefs les plus cités à l'encontre de cette approche du fait qu'elle est fortement liée à ce framework (UPML) négligeant ainsi d'autres préoccupations importantes liées aux services sémantiques. Il s'agit en particulier des besoins, qui nous intéressent beaucoup dans le cadre de nos travaux, à savoir l'intégration des services.

En résumé, on peut retenir que OWL-S est une approche générique, basée sur le standards OWL et WSDL. WSMF constitue beaucoup plus un cadre méthodologique. WSMO est une approche basée sur un langage propriétaire combinant plusieurs logiques. METEORS-S est une approche pragmatique mais qui manque d'abstraction et IRS-II est fortement liée au framework UPML. De plus, nous pouvons remarquer la compatibilité entre OWL-S et METEOR-S qui peuvent ainsi être considérées comme des approches qui peuvent être combinées, alors que ces deux approches sont plutôt contradictoires avec WSMO et avec IRS-II dans la mesure où ces approches utilisent des langages et des outils incompatibles avec les standards industriels.

Par rapport à nos travaux, qui rappelons-le, portent sur l'intégration des systèmes d'information industriels, il s'avère qu'aucune de ces approches n'est complètement mature pour prendre en charge totalement et efficacement l'ensemble du processus l'intégration des applications industrielles. Toutes ces approches souffrent plus ou moins du manque de méthodologies, du manque d'architectures flexibles pouvant décrire efficacement la sémantique liée aux applications, et aussi du manque de maturité ou parfois même de l'absence de mécanismes de description et de médiation efficaces

pouvant être utilisés dans le domaine industriel. Cependant, au vu de tous les éléments présentés dans cette section, il nous paraît que l'approche OWL-S est celle qui présente le plus de maturité, de généricité et de standardisation. C'est la raison fondamentale qui nous a amené à nous baser sur cette approche (OWL-S) comme fondement de base pour les travaux que nous menons dans le cadre de l'intégration des applications industrielles.

IV.8. Conclusion

Dans ce chapitre, nous avons présenté les principales notions portant sur l'intégration sémantique des systèmes d'information et qui peuvent être mises en œuvre dans le domaine industriel. Nous avons étudié en particulier la notion d'ontologie comme l'une des technologies les plus pertinentes pour expliciter la sémantique des applications. Nous avons alors étudié la structure, la construction, la représentation ainsi que les architectures typiques des ontologies. Devant la diversité des ontologies utilisées dans certains systèmes, nous avons montré la nécessité de les intégrer. Aussi, nous avons présenté les principales approches d'intégration des ontologies et nous avons retenu la technique du mapping comme étant l'approche la plus flexible permettant d'interconnecter des ontologies hétérogènes.

Nous avons également présenté les utilisations typiques des ontologies dans des scénarios d'intégration, et nous avons retenu les services sémantiques comme l'une des approches les plus efficaces qui va permettre l'intégration des systèmes d'information industriels. Nous avons ensuite étudié les principales approches de services sémantiques, et nous avons retenu OWL-S, grâce à son degré de maturité vis à vis de la problématique d'intégration dans les domaines industriels, comme fondement de base des travaux que nous menons.

L'analyse des techniques actuelles d'intégration sémantiques permet de révéler à notre sens quatre points essentiels. Le premier point porte sur l'inadéquation des architectures actuelles des ontologies à capturer de façon flexible et efficace la sémantique des applications industrielles. Le second point porte sur le manque de méthodologie à mettre en œuvre pour définir les ontologies et les services sémantiques. Le troisième point concerne la limite des approches actuelles en matière de description et de médiation de services dans le contexte intra-entreprise (et même extra-entreprise qui constitue une perspective logique de nos travaux). Enfin, le dernier point porte sur la complexité des techniques sémantiques d'où la nécessité, en vue de leur adoption, de mettre en œuvre des outils appropriés permettant de simplifier toute la complexité inhérente à l'utilisation de ces technologies. Cette thèse s'inscrit dans l'objectif de répondre à ces problématiques posées en particulier dans le contexte industriel. Le chapitre suivant va permettre de présenter les grands principes de notre proposition.

PARTIE 2

UNE APPROCHE D'INTEGRATION FLEXIBLE BASEE SUR LES SERVICES SEMANTIQUES

Chapitre V

VERS UNE APPROCHE D'INTEGRATION FLEXIBLE BASEE SUR LES SERVICES SEMANTIQUES

V.1. Introduction

Les trois chapitres précédents ont permis d'étudier les principales approches pour l'intégration d'applications qui peuvent être mises en œuvre dans le domaine industriel, et en particulier dans le domaine de la microélectronique. Ils ont notamment permis de mettre l'accent sur la distinction entre les approches syntaxiques et les approches sémantiques. Comme nous l'avons vu, ces dernières constituent le moyen le plus flexible pour intégrer des applications d'entreprise. L'analyse des solutions d'intégration sémantique actuelles, et en particulier des approches basées sur les services sémantiques, a montré un certain nombre de limites qui concernent principalement les architectures des ontologies actuelles, le manque de méthodologie pour définir les ontologies et les services sémantiques, le manque de maturité des mécanismes de description et de médiation de services dans le contexte intra-entreprise et le manque d'outils permettant de prendre en compte la complexité inhérente à l'utilisation des technologies aussi variées que diverses permettant de supporter des architectures de services et des architectures sémantiques.

Partant de ce constat, nous avons investigué le domaine d'intégration des services à la recherche d'une méthodologie et d'une solution flexible d'intégration. Nous allons dans ce chapitre exposer les grandes axes de notre proposition pour l'intégration des applications microélectroniques. La section V.2 effectuera un rappel de notre problématique. La section V.3 présentera les principes fondamentaux de notre modèle d'intégration. Les sections V.4 et V.5 discuteront respectivement du modèle et de l'architecture globale de notre cadre d'intégration. Elles introduiront trois couches permettant de définir respectivement les trois notions majeures de notre modèle : le service fondamental, l'ontologie d'entreprise et le service d'intégration. Enfin, la section V.6 exposera une démarche méthodologique globale pour construire les trois couches de notre proposition.

V.2. Rappel de la problématique

Rappelons que nos travaux portent sur l'intégration d'applications industrielles et s'inscrivent en particulier dans le contexte spécifique du projet industriel MiMISI (Microelectronics Manufacturing Information System Integration) qui concerne l'intégration intra-entreprise d'applications industrielles de la société STMicroelectronics, une grande entreprise dans le secteur de la microélectronique. Rappelons aussi que ce secteur est caractérisé par de fréquentes évolutions liées en grande partie aux changements fréquents de la technologie et de la stratégie déployée au sein de l'entreprise. Ceci nécessite des solutions flexibles d'intégration.

La piste qui nous parait à notre sens la plus pertinente pour mettre en œuvre des solutions flexibles en matière d'intégration d'applications microélectroniques est celle basée sur les services sémantiques (et les services web sémantiques). Ces derniers, comme nous les avons précédemment décrits (chapitre IV), sont considérés comme le résultat de la fusion des architectures orientées services (SOA) et des ontologies. Ils constituent une approche assez prometteuse pour l'intégration d'application d'entreprise. Bien que les services sémantiques soient actuellement peu matures, il n'en demeure pas moins que leur intérêt pour des scénarios d'intégration intra-entreprise est indiscutable [Haller et al. 2005].

Dans l'état actuel des choses, les services sémantiques souffrent d'un certain nombre de limites et de lacunes. Il s'agit fondamentalement du manque de méthodologies à mettre en œuvre pour définir les services sémantiques et aussi du manque de mécanismes efficaces pour la description et la médiation de services dans le contexte intra-entreprise. En particulier, nous avons besoin de spécifier, par rapport à l'état de l'art actuel des services sémantiques, le contexte particulier de la problématique d'intégration interne aux grandes entreprises, et en particulier à la société STMicroelectronics qui constitue notre cadre industriel.

L'analyse de cette problématique d'intégration intra-entreprise dans le domaine industriel nous a conduit à la décomposer en quatre sous-problématiques qui forment notre problématique de recherche et qui sont (figure V.1) :

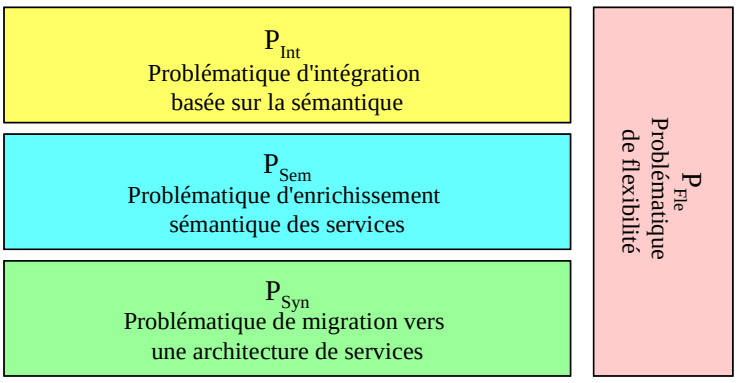

Figure V.1. Les quatre sous-problématiques traitées

- la problématique de migration vers une architecture de services (P_{Syn}), également appelée problématique de 'servicisation'[31] et qui a pour but de construire des

[31] Nous empruntons les anglicismes 'serviciser' et 'servicisation' pour désigner respectivement l'action et le processus de migration vers une architecture de services d'entreprise.

services à l'intérieur de l'entreprise;

- la problématique d'enrichissement sémantique des services (P_{Sem}), également appelée problématique de 'sémantification'[32] et qui a pour but de construire les ontologies de l'entreprise;
- la problématique d'intégration basée sur la sémantique (P_{Int}), également appelée problématique d'intégration et qui a pour but de proposer des mécanismes d'intégration basés sur la sémantique;
- et la problématique de flexibilité (P_{Fle}) qui a pour but de définir des principes permettant de garantir la flexibilité des éléments issus des trois problématiques ci-dessus.

Rappelons que ces problématiques s'inscrivent dans la continuité des problématiques déjà traitées dans les travaux de [Chapron 2006] qui par ailleurs s'intéresse à la problématique de pilotage de l'évolution des systèmes d'information microélectroniques. Notons également que les problématiques, que nous avons identifiées, s'intéressent à un aspect bien précis de l'intégration intra-entreprise. Elles sont mutuellement interdépendantes dans le sens où les notions issues d'une problématique peuvent être réutilisées dans une autre problématique. Les trois premières problématiques sont complémentaires tandis que la quatrième leur est orthogonale dans le sens où elle est transversale. Le détail de ces problématiques est donné ci-après.

V.2.1. Problématique de migration vers une architecture de services

La principale limite des architectures orientées services est le manque de méthodologies permettant de les construire de façon efficace. Le niveau d'abstraction des architectures orientées services correspond généralement au niveau logique des méthodes de conception de systèmes d'information. Ce niveau se situe, au-dessus du modèle de conception d'applications traditionnelles (par exemple, modèle objet d'OMT ou modèle logique de Merise). Les développements d'applications orientées services (SODA – Service-Oriented Development of Applications) ne font, par exemple, que définir les lignes directrices qui permettent de guider les architectes vers un modèle orienté services, mais ils ne délivrent pas de méthodologie complète et précise pour atteindre leur but de façon efficiente. Aussi, la proposition d'une méthodologie adaptée aux entreprises industrielles larges et évolutives est plus qu'une nécessité.

Conséquence du manque de méthodologie de construction de l'architecture orientée services, une difficulté majeure qui est généralement rencontrée est la définition de la notion de service. Qu'est ce qu'un un service pour l'entreprise ? Quel niveau de granularité[33] faut-il retenir pour définir correctement les services afin de permettre une gestion simplifiée et aussi une meilleure réutilisation? Quel degré de généricité[34] faut il retenir ? etc. Autant de questions qui méritent des réponses précises. Aussi, notre première sous-problématique (P_{Syn}) consiste à définir une méthodologie de migration du système d'information d'entreprise existant vers une architecture orientée services.

[32] Nous empruntons les anglicismes 'sémantifier' et 'sémantification' pour désigner respectivement l'action et le processus d'enrichissement sémantique des services d'entreprise.

[33] Il est généralement admis que les critères de réutilisation et de gestion des services sont des critères antagonistes dans la mesure où d'une part, plus les services sont de grosse granularité, plus ils sont gérables mais moins réutilisables, et d'autre part, plus ils sont de fine granularité et plus ils sont plus réutilisables mais difficilement gérables.

[34] La notion de généricité d'un service permet de définir le degré de réutilisation idéal d'un service.

V.2.2. Problématique d'enrichissement sémantique

L'analyse des solutions sémantiques pour l'intégration des applications industrielles, et en particulier de l'approche OWL-S qui est à notre sens la plus mature, montre qu'elles souffrent d'un certain nombre de limites dues principalement au manque de méthodologie et/ou la non prise en charge de certaines spécificités liées à l'intégration intra-entreprise. En effet, les solutions d'enrichissement actuelles ne proposent pas d'architecture adaptée, ni de démarche spécifique pour capturer de façon efficiente et efficace la sémantique de l'entreprise. De plus, la spécificité du projet intra-entreprise implique la nécessité de prendre en compte un certain nombre d'aspects qui n'étaient pas nécessaires dans le cadre d'intégration inter-entreprise.

L'ensemble de ces points constitue notre seconde sous-problématique (P_{Sem}) qui concerne, rappelons-le, l'enrichissement sémantique des architectures de services d'entreprise. Cette problématique consiste à définir de façon flexible l'architecture sémantique de l'entreprise, nécessaire pour mener à bien le projet d'intégration d'applications industrielles. Il s'agit en particulier de proposer d'une part la manière avec laquelle la sémantique associée aux applications doit être capturée, et d'autre part les extensions nécessaires à apporter à OWL-S pour décrire correctement la sémantique des services d'entreprise.

V.2.3. Problématique d'intégration dirigée par la sémantique

Une fois que l'enrichissement sémantique des services d'entreprise est effectué, il reste à définir des mécanismes adaptés et basés sur la sémantique permettant de mettre en œuvre l'intégration des applications industrielles. Ceci constitue notre troisième sous-problématique (P_{Int}). Il s'agit en particulier d'étendre certains mécanismes existants comme la publication et la découverte de services OWL-S, et de proposer une nouvelle approche de médiation de services basée sur la sémantique qui devrait être définie lors de la résolution de la sous-problématique précédente (cf. § V.2.2).

Il est fondamental de préciser que l'une des spécificités des travaux que nous menons repose sur le critère de flexibilité. En effet, tout au long de nos travaux nous avons adopté un comportement qui privilégie la flexibilité dans la résolution du problème d'intégration. Cette préoccupation constitue la sous-problématique de flexibilité qui est traitée ci-après.

V.2.4. Problématique de flexibilité

Face aux évolutions fréquentes qui caractérisent les systèmes d'information microélectroniques, il devient nécessaire de structurer le système d'information de manière à favoriser et à faciliter ces évolutions. La problématique de flexibilité (P_{Fle}) est une problématique orthogonale, si bien qu'elle peut concerner les trois sous-problématiques précédentes.

Le terme flexibilité peut être défini de façon générale comme l'aptitude d'un système à changer facilement pour pouvoir s'adapter aux circonstances. Dans le contexte des systèmes d'information, [Chelli 2003] définit la notion de flexibilité comme étant la déclinaison informatique de la notion d'agilité des entreprises. Il distingue quatre natures interdépendantes de flexibilité qu'il convient de considérer dans tout processus visant à améliorer l'agilité de l'entreprise :

- La *flexibilité technologique* : est celle qui est privilégiée par les techniciens IT et qui est délivrée à travers les langages, les outils, les méthodes et les technologies de l'information mises en œuvre pour développer le système d'information;
- La *flexibilité structurelle* : est celle qui favorise l'émergence de normes et de standards d'échanges et qui est délivrée à travers un équipement préalable des composants du système d'information en vue de faciliter l'intégration de nouvelles fonctionnalités;
- La *flexibilité potentielle* : est celle qui permet de libérer les ressources du système d'information entre les cycles de développement et qui est délivrée à travers des services, des fonctions ou des règles de gestion développés mais non mis en service immédiatement;
- La *flexibilité topographique* : est celle qui est délivrée à travers l'urbanisation du système d'information et qui permet de circonscrire l'impact des évolutions de la réalité opérationnelle et réduire ainsi de manière très sensible, la durée et la complexité des cycles de développement.

La figure V.2 permet de récapituler la notion de flexibilité à travers une hiérarchie des natures de flexibilité cohérente avec les grandes étapes d'évolution de la vision informatique au sein des entreprises. On peut par ailleurs retenir le fait que l'accroissement de la contribution du système d'information à l'agilité de l'entreprise s'accompagne d'un renforcement de sa synergie avec l'organisation (d'où la notion de modèle synergique préconisé et proposé par l'auteur).

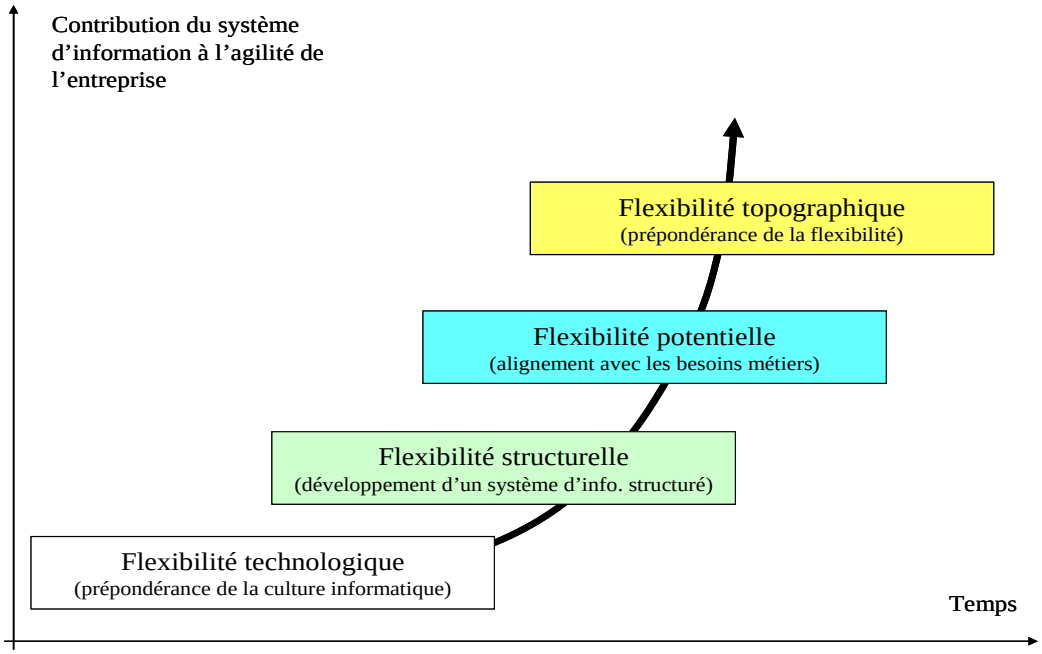

Figure V.2. Natures de flexibilité (Adapté de [Chelli 2003])

La problématique de flexibilité est orthogonale aux trois premières problématiques dans le sens où la flexibilité est traitée au niveau de chacune des problématiques. Aussi, les différentes natures de flexibilité que nous avons présentées précédemment peuvent se retrouver partiellement ou totalement au niveau de chacune des trois problématiques.

Les architectures orientées services, qui font l'objet de la problématique (P_{Syn}), ont été retenues parce qu'elles sont considérées comme étant les plus flexibles, et donc à même de rendre les systèmes d'information plus modulaires et faiblement couplés. La

flexibilité des SOA est généralement liée à la manière d'appeler un service qui est indépendante de se logique d'implémentation. Ainsi une application peut plus facilement être remplacée par une autre application nouvelle, mise en œuvre sans impacter tout l'existant. D'un point de vue technique, la démarche SOA permet de rationaliser l'intégration des applications et les développement des futures applications. En effet, les SOA apportent de par leur flexibilité une manière plus légère d'intégrer les applications d'entreprise. De plus, les SOA permettent de favoriser la mutualisation et la réutilisation par variantes de services. D'un point de vue métier, la démarche SOA permet une organisation ou une réorganisation efficaces du système d'information. En effet, les SOA apportent une rationalisation des éléments fonctionnels, la suppression des redondances d'appels de services et une catégorisation métier des services. Vis à vis des différentes natures de flexibilité, le choix de l'utilisation des standards dans les SOA constitue un moyen pour mettre en oeuvre la flexibilité technologique et structurelle, la notion de référentiel de services est un moyen qui permet de réaliser la flexibilité potentielle, et enfin, le choix du principe d'urbanisation qui a été retenu tout au long de notre démarche comme un moyen pour réaliser la flexibilité topographique au sein de la problématique (P_{Syn}).

Les ontologies, qui sont l'objet de la problématique (P_{Sem}) constituent un autre niveau de flexibilité. Elles permettent de décrire conceptuellement (avec une vision très rapprochée du métier) les services d'entreprise. En effet, les ontologies apportent de la flexibilité dans la mesure où elles peuvent nous affranchir de la complexité des technologies informatiques. Vis à vis des différentes natures de flexibilité, le choix de l'utilisation des standards pour implémenter les ontologies constitue un moyen pour mettre en oeuvre la flexibilité technologique et structurelle, la notion de référentiel d'ontologies est un moyen qui permet de réaliser la flexibilité potentielle, et enfin, le choix du principe d'urbanisation des ontologies est un moyen pour réaliser la flexibilité topographique au sein de la problématique (P_{Sem}).

Enfin, en ce qui concerne la troisième sous-problématique (P_{Int}), l'intégration basée sur la sémantique, cette dernière doit être également flexible dans le sens où elle doit prendre en charge l'évolution du système. Un exemple typique de flexibilité est de permettre par exemple de développer les services de médiation qui respecte l'orientation services. Un autre exemple de flexibilité consiste à permettre la cohabitation aussi bien de solutions syntaxiques que de solutions sémantiques. Vis à vis des différentes natures de flexibilité, le choix de l'utilisation des standards pour implémenter les services d'intégration constitue un moyen pour mettre en oeuvre la flexibilité technologique et structurelle, la notion de référentiel d'intégration est un moyen qui permet de réaliser la flexibilité potentielle, et enfin, le choix du principe d'urbanisation des services d'intégration est un moyen pour réaliser la flexibilité topographique au sein de la problématique (P_{Int}).

V.3. Principes fondamentaux

Cette section va décrire les éléments importants de notre approche que nous appelons ODSOI (**O**ntology-**D**riven **S**ervice-**O**riented **I**ntegration) où certains aspects ont été déjà exposés dans [Izza et al. 2004][Izza et al. 2005-a][Izza et al. 2006-a] et qui a pour objectif de prendre en charge la problématique liée à l'intégration sémantique d'applications industrielles, et en particulier dans le domaine microélectronique.

Tout d'abord, insistons sur le fait que ODSOI constitue une approche pour l'intégration

intra-entreprise. Ceci signifie que notre approche traite du problème d'hétérogénéité en fournissant des mécanismes de médiation basée sur le concept d'ontologie. De plus, notre approche est orientée services dans la mesure où elle se base sur des architectures orientées services et plus précisément sur des Services Web. Le choix des Services Web est principalement motivé par le fait qu'ils constituent une technologie flexible basée sur des standards industriels, tandis que le choix des ontologies peut être justifié par le fait qu'elles constituent la technologie la plus prometteuse pour la prise en compte de l'aspect sémantique des Services Web.

Plusieurs principes fondamentaux ont été retenus au sein du projet MiMISI, et qui sont :

- l'ouverture;
- l'unification;
- et l'urbanisation.

Chacun de ces principes est fondamental dans la mesure où il permet de répondre à un aspect précis de la flexibilité de notre approche d'intégration d'applications industrielles.

V.3.1. Principe d'ouverture

Plus qu'un principe, l'ouverture est avant tout une contrainte imposée par l'industriel qui désire ainsi une solution basée sur des standards industriels. Ceci justifie les multiples choix effectués dans le chapitre précédent, et en particulier le choix opérés en matière de description syntaxique des services (WSDL), de description sémantique générique des services (OWL-S) et de représentation des ontologies (OWL).

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

V.3.2. Principe d'unification

L'unification peut être définie comme étant le processus qui permet de représenter de façon uniforme les différents composants d'entreprise qui peuvent être concernés par le projet d'intégration. Il s'agit de proposer un formalisme flexible permettant de supporter à la fois l'intégration des données, des applications et des processus. Le paradigme qui est à notre sens le plus pertinent pour répondre à ce principe est l'orientation services. En effet, il est possible de considérer les multiples composants d'un système d'information comme des services d'entreprise.

Une conséquence importante de l'application du principe est la définition de la notion de service de données. Ce dernier est défini comme étant un moyen qui permet d'encapsuler sous forme de service une source de données ou une vue d'une source de données. Ce type de service permet en particulier de définir une structure qui favorise le découplage entre les données et les traitements du système d'information.

V.3.3. Principe d'urbanisation

Un autre principe majeur de notre approche est l'urbanisation. Elle consiste à proposer une solution structurée de manière à favoriser la flexibilité, notamment la flexibilité topographique. A l'origine la métaphore de l'urbanisme a été appliquée aux systèmes

d'information afin de les rendre plus aptes à servir la stratégie de l'entreprise et à anticiper les changements dans l'environnement de l'entreprise.

Notre approche d'urbanisation s'inscrit dans le prolongement de l'urbanisation "classique" des systèmes d'information. Elle consiste plus précisément à étendre les concepts d'urbanisme aux différentes couches de notre approche. Aussi, on propose une urbanisation orientée service des systèmes d'information et qui constitue un prolongement logique de l'urbanisme appliquée aux SOA. On propose également une urbanisation sémantique des systèmes d'information qui constitue un autre prolongement de la notion d'urbanisation appliquée à la sémantique de l'entreprise. Enfin, nous proposons une urbanisation de l'intégration qui est aussi un prolongement de la notion d'urbanisme aux infrastructures d'intégration du système d'information.

V.4. Modèle d'intégration en couches hiérarchisées

Le concept de modélisation en couches hiérarchisées (layering model) n'est pas nouveau. Il est introduit afin de définir une approche flexible pour le problème d'intégration. Ce dernier est généralement basé sur la notion d'engagement qui constitue le but de tout processus de standardisation. L'engagement consiste à s'accorder sur un ensemble d'éléments partagés constituant le standard. Cependant, le processus de standardisation est souvent coûteux en moyens et en temps. Ceci a donné naissance à des approches stratifiées, où les standards sont utilisés à différents niveaux. [Spring 1996] suggère qu'en développant les systèmes de façon modulaire grâce aux couches hiérarchisées, l'impact des changements au niveau d'un système peut être isolé des autres. Les exemples les plus connus viennent surtout du domaine des télécommunications, comme par exemple le modèle de référence OSI et celui du protocole d'Internet TCP/IP.

Dans le contexte de web sémantique, [Melnik et Decker 2000] suggèrent une approche hiérarchisée pour le problème d'intégration des données connu sous le nom d'Information Model Interoperability (IMI) permettant de réduire la complexité du web en identifiant une série de couches : *Syntaxique*, *Objets* et *Sémantique*. Notre approche s'inspire de ce modèle et suggère un modèle hiérarchique pour l'intégration sémantique des applications industrielles que nous appelons ODSOIM (ODSOI Model).

Ce modèle est, comme illustré en figure V.3, basé sur trois couches hiérarchisées qui sont :

- la *couche syntaxique* qui répond au problème de représentation syntaxique;
- la *couche sémantique* qui répond au problème de représentation sémantique;
- et finalement la *couche intégration* qui concerne l'intégration des applications industrielles.

Le but principal de la *couche syntaxique* est de fournir un moyen pour la description des services et la sérialisation des données et des messages entre les applications industrielles. Cette couche reçoit des informations de la couche sémantique et les transforme ensuite en données sérialisées au format XML qui peuvent ensuite être envoyées sur le réseau. Les données qui sont reçues par la couche syntaxique cible sont ensuite envoyées à la couche sémantique cible en vue d'être utilisées par l'application cible.

La *couche sémantique* a pour but de détecter et de résoudre les conflits d'ordre sémantique qui peuvent survenir lors des échanges d'information ou le partage de

fonctionnalités entre les applications industrielles. Cette couche reçoit les informations de la couche intégration et les propage en tant que données au niveau de la couche syntaxique. Ceci constitue le scénario typique qui s'opère du côté source. Du côté application cible, la couche sémantique reçoit des données de la couche syntaxique et les transmet après interprétation en tant qu'information à la couche intégration. La couche sémantique est concrètement réalisée, comme nous allons le découvrir par la suite, dans le monde du web sémantique à travers l'utilisation de la couche metadata et de la couche ontologie et qui sont généralement réalisées grâce aux couches RDF(S) et OWL [Berners-Lee et al. 2001].

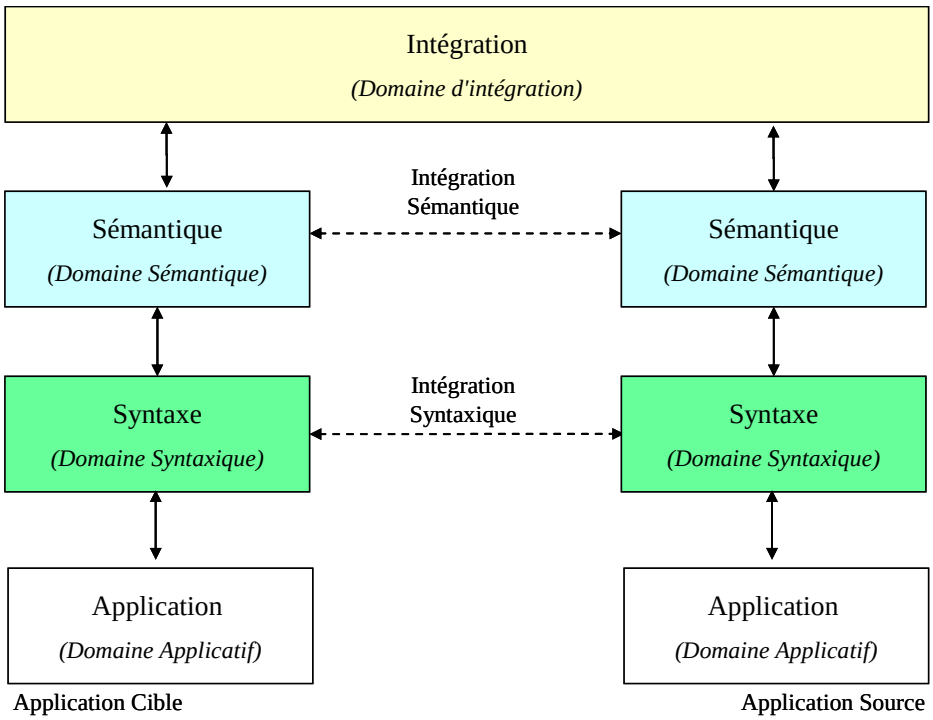

Figure V.3. Modèle d'intégration (ODSOIM) en couches des applications industrielles

La *couche intégration* est une couche conceptuelle qui permet de manipuler des applications (services) sémantiques. La notion d'application (service) sémantique est considérée comme une application (service) sémantiquement enrichi(e). Cette couche permet à des applications hétérogènes d'interopérer et de coopérer sémantiquement les unes avec les autres. Elle permet une interaction entre les applications et aussi entre les applications et les utilisateurs. A titre d'exemple, cette couche permet l'invocation des applications et l'expression des requêtes de données et de traitements, selon les besoins de l'utilisateur.

La généralisation du modèle d'intégration en couches précédemment décrit aux différents composants de l'entreprise qui sont susceptibles d'être concernés par l'intégration donne naissance à la représentation ci-dessous (figure V.4. On peut distinguer quatre types de composants de base d'entreprise : les sources de données (ou composants de données), les applications (ou composants applicatifs) les fonctions (ou composants fonctionnels) et les processus (ou composants métier).

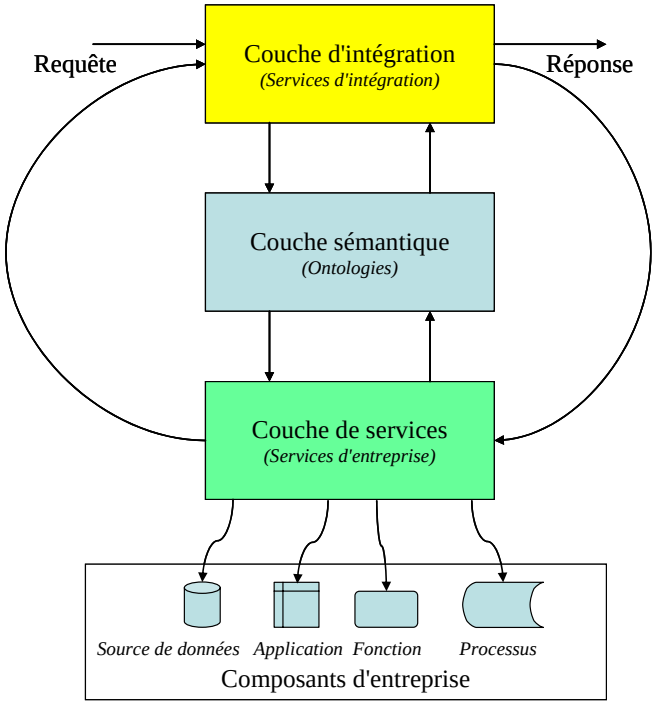

Figure V.4. Modèle étendu d'intégration des applications industrielles

Il est possible de représenter le principe de notre modèle d'intégration à travers le méta-modèle simplifié (ODSOIM2 - ODSOI Meta-Model) ci-dessous (figure V.5).

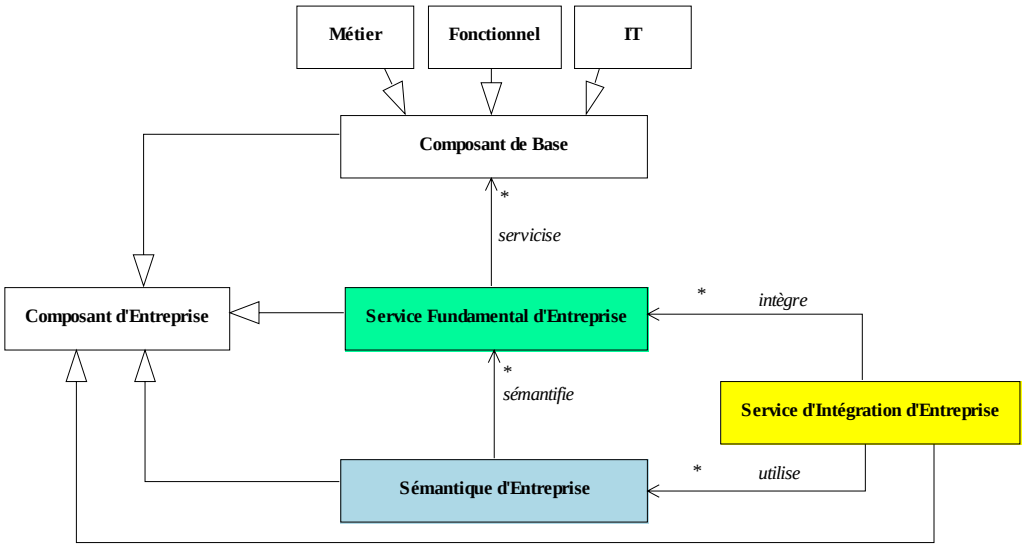

Figure V.5. Méta-modèle simplifié (ODSOIM2) pour l'intégration des applications industrielles[35]

Comme on peut le remarquer, notre approche enrichit les composants de base de l'entreprise à travers la définition de trois notions fondamentales qui sont nécessaires pour mener à bien le projet d'intégration. Il s'agit de la notion :

[35] Signalons que nous nous basons tout au long de ce chapitre et des chapitres suivants sur le formalisme UML pour représenter nos différents modèles et méta-modèles. Aussi, le sens des symboles graphiques utilisés dans ces modèles est celui préconisé dans UML 2.0. En particulier, les flèches désignent la relation de spécialisation/généralisation entre classes, les arcs possédant un losange à l'extrémité désignent des relations d'agrégation entre classes, et les arcs simples désignent les autres types de relations entre classes.

- de service fondamental d'entreprise qui permet de "serviciser" (exposer sous forme de services) les composants de base, permettant de créer ainsi une architecture de services,
- de sémantique d'entreprise qui permet de "sémantifier" (enrichir sémantiquement) les services fondamentaux;
- et de service d'intégration d'entreprise qui permet de définir des mécanismes d'intégration des services fondamentaux enrichis sémantiquement.

V.5. Architecture ODSOI

L'architecture que nous proposons est appelée ODSOIA (ODSOI Architecture). Cette dernière étend le concept d'architecture orientée services (qui constitue la couche service d'entreprise) par une couche sémantique (ou couche d'ontologie d'entreprise) et une couche d'intégration (couche d'intégration d'entreprise). Comme il est précisé en figure V.6 qui est une représentation plus synthétique du méta-modèle précédent, ces différentes couches sont interdépendantes dans le sens où elles entretiennent des interactions entre elles. Dans ce qui suit, nous nous focalisons principalement sur le cadre général et nous détaillerons ensuite chacune des couches dans les chapitres suivants.

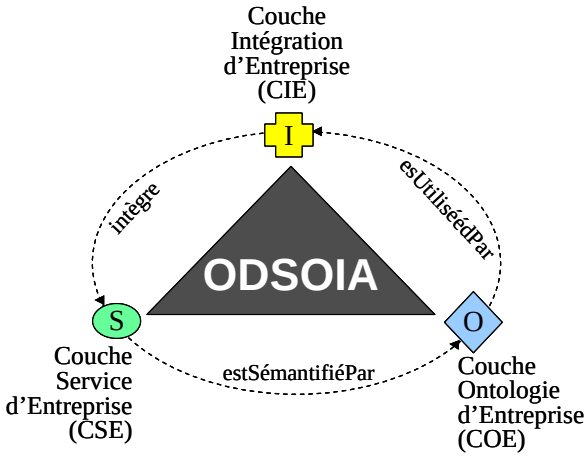

Figure V.6. Vision générale du cadre d'intégration ODSOI

L'architecture ODSOIA que nous proposons fournit un cadre unifié pour intégrer les applications industrielles. Les principales caractéristiques de cette architecture peuvent être illustrées en figure V.7 qui montre une architecture spécifique orientée services permettant d'adresser le problème d'intégration d'applications dans le contexte MiMISI. Dans notre architecture d'intégration, deux types majeurs de services, que nous appelons *services fondamentaux*, ont été identifiés : Service-IT et Service-Métier. Ces deux types de services permettent de définir respectivement des services de niveau informatique et des services de niveau métier.

Comme illustré en figure V.7, les services fondamentaux sont reliés entre eux par l'intermédiaire du bus sémantique de services d'entreprise (BSSE ou ODESB Ontology-Driven Enterprise Services Bus) qui étend le traditionnel bus de services (ESB - Enterprise Service Bus) [Chappell 2005]. Comme on peut le constater, en plus des services fondamentaux précédemment cités, nous disposons également d'un certain nombre de services d'intégration, ainsi que d'un ensemble de référentiels (ou registres)

d'intégration. Les référentiels permettent de stocker toutes les informations nécessaires à l'intégration et incluent :

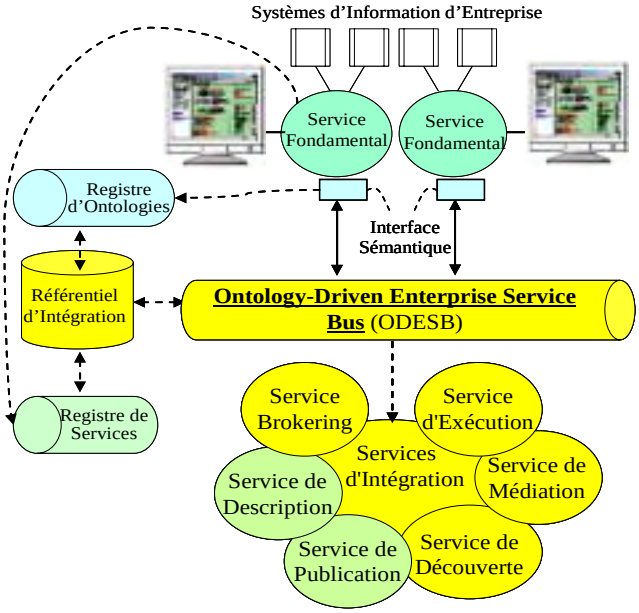

Figure V.7. Vue globale de l'architecture ODSOIA

- des référentiels de services : qui constituent l'annuaire des services fondamentaux;
- et des référentiels d'ontologies : qui contiennent la sémantique des services fondamentaux nécessaire pour la l'intégration sémantique.

En ce qui concerne les services d'intégration, ces derniers ont pour but de fournir des mécanismes nécessaires pour l'intégration des services fondamentaux et qui sont:

- service broker qui est un service dédié permettant la gestion du processus d'intégration,
- service de description permettant la description sémantique des services fondamentaux,
- service de publication permettant la publication sémantique des services fondamentaux,
- service de découverte permettant la découverte sémantique des services fondamentaux,
- service de médiation permettant la résolution des hétérogénéités sémantiques liées aux services fondamentaux,
- et service d'exécution permettant la supervision de l'exécution des services fondamentaux.

Comme on peut le constater sur la figure V.7, les services de description et les services de publication portent une couleur différente de celle des autres services d'intégration. La raison est simplement due au fait que ces deux services d'intégration interviennent lors du design-time (lors de la conception du scénario d'intégration) et non pas du run-time (lors de l'exécution du scénario d'intégration).

Il est possible d'associer les éléments de notre architecture aux différentes couches du cadre d'intégration comme le montre la coupe transversale du bus de services illustrée en figure V.8. Cette coupe transversale permet notamment de représenter d'une façon plus pragmatique le modèle ODSOIM. Elle permet plus précisément de révéler un empilement de plusieurs couches concentriques dont certaines (les couches les plus

internes de la couche services) existent actuellement et sont supportées par des standards liées aux Services Web (HTTP, SOAP, WSDL, UDDI, SAML, etc.).

On peut remarquer au passage que la couche services inclut une sous-couche urbanisation qui permet de mettre en oeuvre une urbanisation orientée services du système d'information. On peut également constater que la couche ontologique inclut à son tour deux sous-couches qui sont la couche description et la couche d'urbanisation d'ontologies, et que la couche d'intégration inclut tous les services d'intégration précédemment évoqués. Le détail de chacune des trois couches est présenté dans les chapitres suivants.

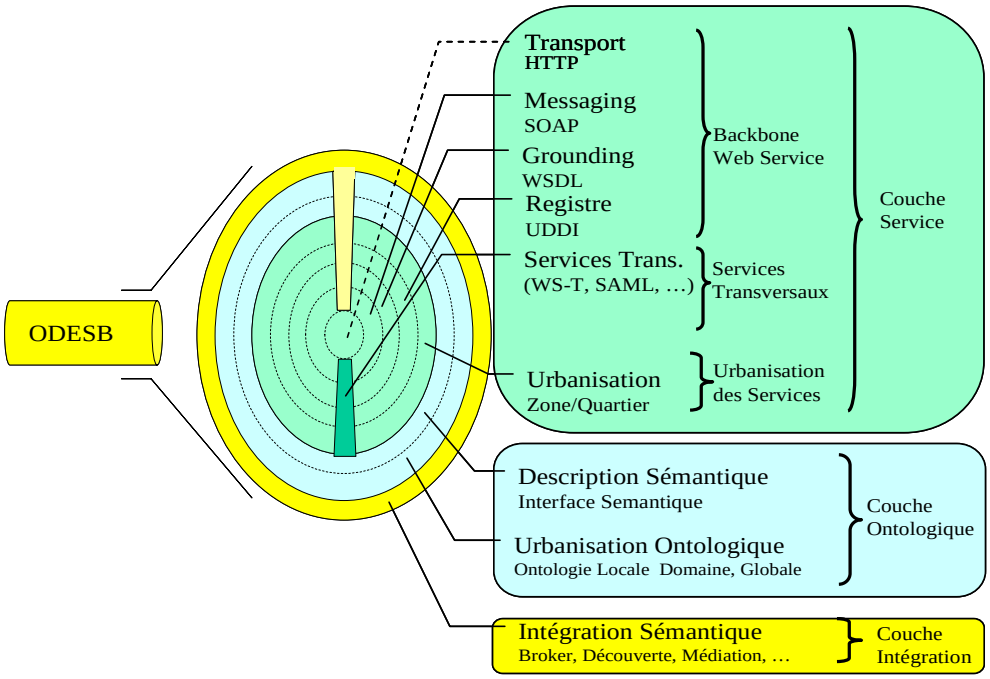

Figure V.8. Coupe transversale du bus ODESB

V.6. Démarche méthodologique globale

Nous avons décidé de résumer les principales étapes nécessaires à la mise en œuvre de l'intégration au sein d'une entreprise industrielle par le schéma de la figure V.9 qui illustre une démarche méthodologique globale permettant de définir les trois couches décrites dans les sections précédentes. Les principales étapes de cette démarche sont :

- *Servicisation* : qui inclut principalement la définition des services fondamentaux et leur urbanisation;
- *Sémantification* : qui permet de décrire la sémantique des services et de les urbaniser;
- *Intégration* : qui permet principalement de développer les services d'intégration et de les déployer.

Il est fondamental de noter que, même si la démarche est représentée schématiquement de façon linéaire, cette dernière demeure essentiellement itérative et incrémentale. Ce caractère itératif et incrémental de la démarche est fondamental car il permet d'apporter de la flexibilité pour le projet d'intégration qui, rappelons le, en a besoin du fait qu'il s'inscrit dans la durée.

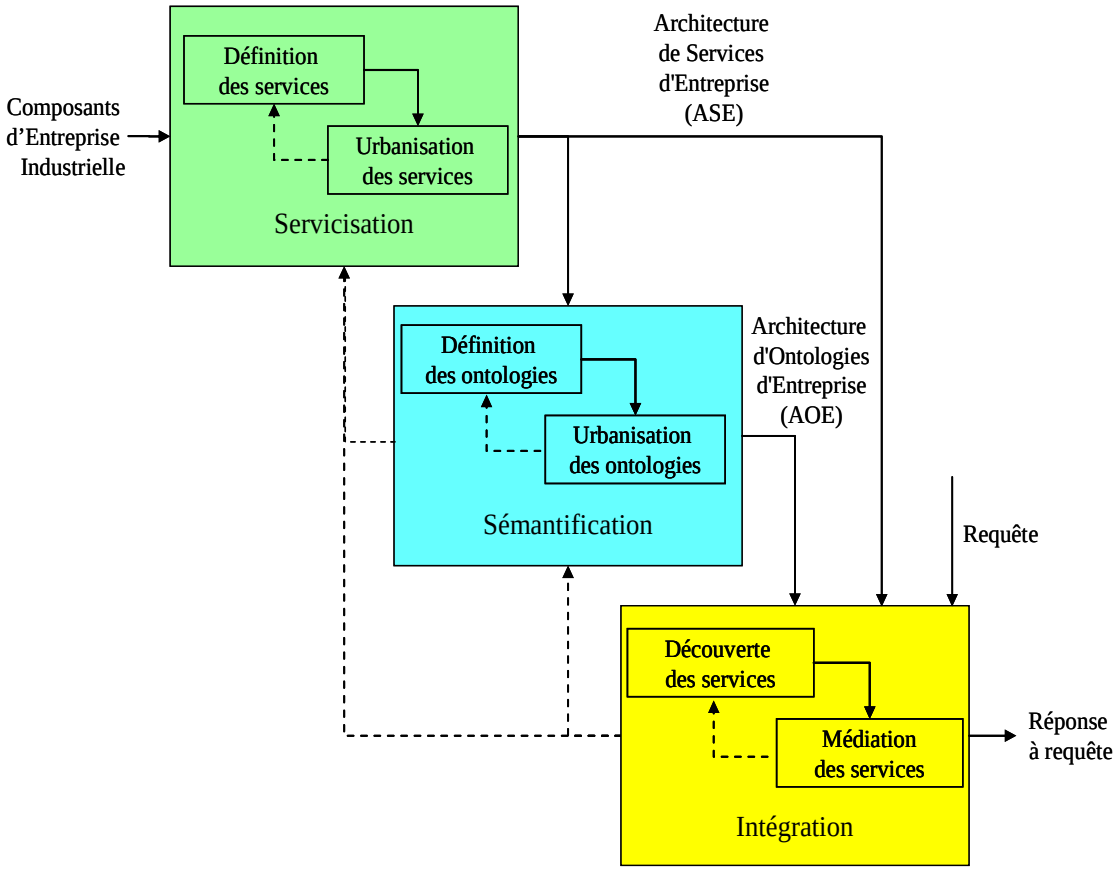

Figure V.9. Démarche méthodologique globale

V.7. Conclusion

Ce chapitre a permis de présenter une vision globale des travaux que nous menons. Plus particulièrement, il a exposé la problématique de recherche qui se décline en quatre sous-problématiques complémentaires qui sont respectivement la problématique de migration vers une architecture de services, la problématique d'enrichissement sémantique des services, la problématique de développement de mécanismes d'intégration basés sur la sémantique et enfin la problématique de flexibilité qui est orthogonale aux autres et qui pose de façon implicite la problématique d'agilité de l'entreprise.

Nous avons également exposé les principes fondamentaux de notre approche. Nous avons retenu trois principes majeurs qui sont l'ouverture, l'unification et l'urbanisation. En tenant compte de ces principes, nous avons alors proposé un modèle d'intégration en couches hiérarchisées. La motivation d'un tel modèle tient du fait que le modèle est plus flexible car un changement au niveau d'une couche impacte moins les autres couches. Nous avons ensuite proposé une architecture globale qui permet de donner quelques détails du modèle proposé. Finalement, nous avons proposé une démarche méthodologique globale qui s'articule autour de trois phases de base qui sont la servicisation, la sémantification et l'intégration. Les trois chapitres à venir ont respectivement pour objet de détailler chacune de ces phases.

Chapitre VI

CONSTRUCTION DE L'ARCHITECTURE DE SERVICES D'ENTREPRISE

VI.1. Introduction

Le chapitre précédent a permis de présenter la problématique de recherche (scientifique et industrielle) et a permis notamment de poser un certain nombre de questions dont les réponses forment les principales contributions de cette thèse. La première question posée était la problématique P_{Syn} (figure VI.1) qui consistait à savoir comment 'serviciser'[36] ou migrer vers une architecture de services (cf. § V.2.1), c'est à dire, comment modéliser et mettre en place une architecture orientée services qui est appropriée dans le cadre de grandes entreprises réactives du domaine industriel. C'est dans le but de répondre à une telle question que nous nous proposons dans ce chapitre de présenter notre approche permettant de définir la première couche (figure VI.2[37]) de notre modèle qui est appelée l'architecture des services d'entreprise (ASE).

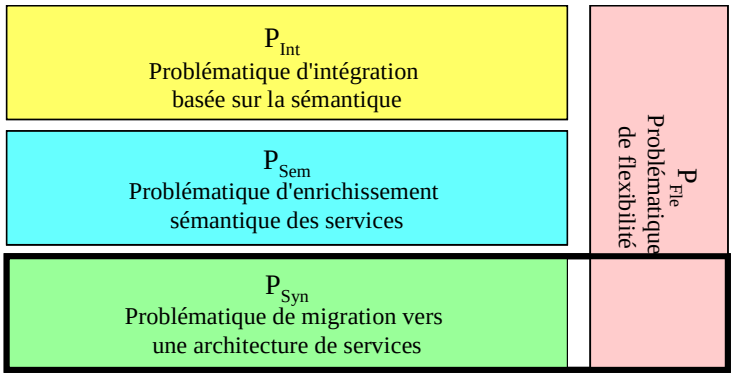

Figure VI.1. La problématique P_{Syn}

[36] Nous empruntons les anglicismes 'serviciser' et 'servicisation' pour désigner respectivement l'action et le processus de migration vers une architecture de services d'entreprise.
[37] Cette figure découle directement de la figure V.8.

La démarche que nous avons adoptée pour notre approche consistait à enrichir ce qui est fait en matière d'architectures de services classiques et à l'adapter aux grandes entreprises industrielles en quête de flexibilité et d'agilité.

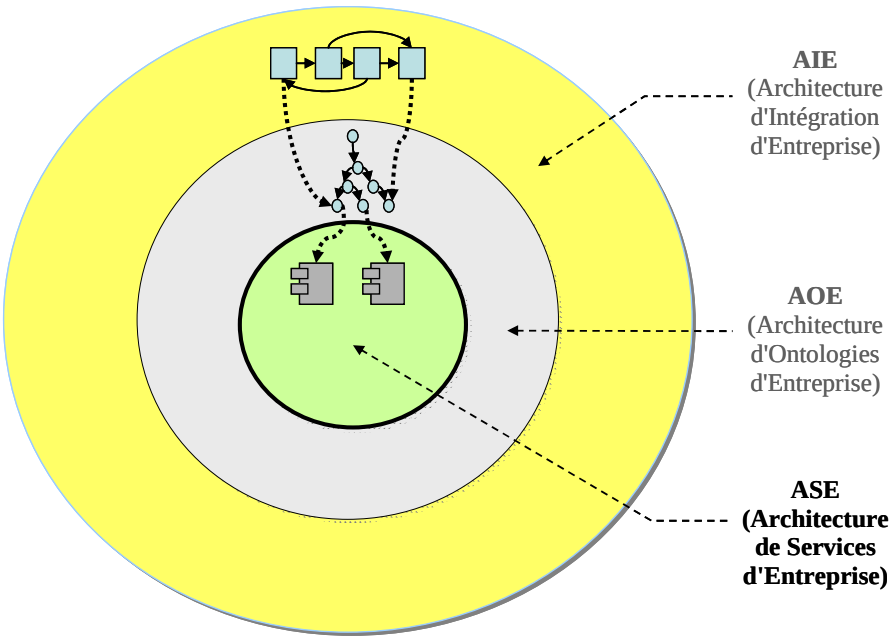

Figure VI.2. L'architecture de services d'entreprise (ASE)

Dans ce qui suit, nous allons exposer dans la section VI.2 les principes fondamentaux qui régissent la construction de notre architecture de services. A l'issue de ces principes, nous allons dans la section VI.3 proposer une modélisation des services d'entreprise, puis dans la section VI.4 proposer un cadre méthodologique permettant de construire l'architecture de services d'entreprise. Et enfin, dans la section VI.5, nous allons proposer une urbanisation orientée services qui permet de structurer les services d'entreprise en îlots afin de faciliter et favoriser à la fois l'intégration et la réutilisation des services d'entreprise.

VI.2. Principes majeurs

La construction d'une SOA à l'intérieur d'une entreprise doit répondre à plusieurs préoccupations qui portent sur les aspects modélisation et démarche orientées services. Notre approche architecturale repose principalement sur les points suivants (figure VI.3):

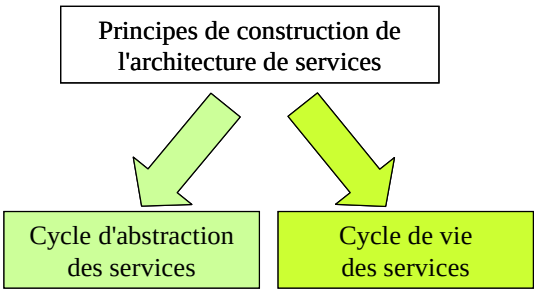

Figure VI.3. Principes majeurs de construction de l'architecture de services

- Un *cycle d'abstraction* permettant de définir une modélisation qui met en exergue une typologie de services de différents niveaux d'abstraction;
- Un *cycle de vie* des services qui propose une démarche de construction de l'architecture de services.

Le cycle d'abstraction se base sur la définition de deux niveaux d'abstraction de services qui sont le niveau technique (ou informatique) et le niveau métier. Ces deux niveaux sont fondamentaux pour pouvoir distinguer entre les services plus abstraits (métiers) et les services plus techniques (informatiques), séparant ainsi les préoccupations d'ordre informatique des préoccupations d'ordre métier. Cette double préoccupation, qui permet notamment un meilleur alignement informatique/métier, donne naissance à un double aspect pour notre architecture orientée services: un aspect purement technique (architecture de services informatiques ou services IT) et un aspect purement métier (architecture de services métier).

Le cycle de vie des services, quant à lui, permet de proposer une méthodologie spécifique qui est une extension des méthodologies actuelles en matière de construction d'architectures de services.

Il est fondamental de préciser que l'une des spécificités majeures du cycle de vie des services est la prise en compte de l'aspect urbanisation des services. Cette urbanisation a pour objet de structurer l'architecture de services en îlots de façon à simplifier la réutilisation et l'intégration des services.

Par rapport au modèle de maturité des SOA, nos travaux concernent essentiellement la couche 1, 2 et la couche 3a du modèle CMMI (figure VI.4).

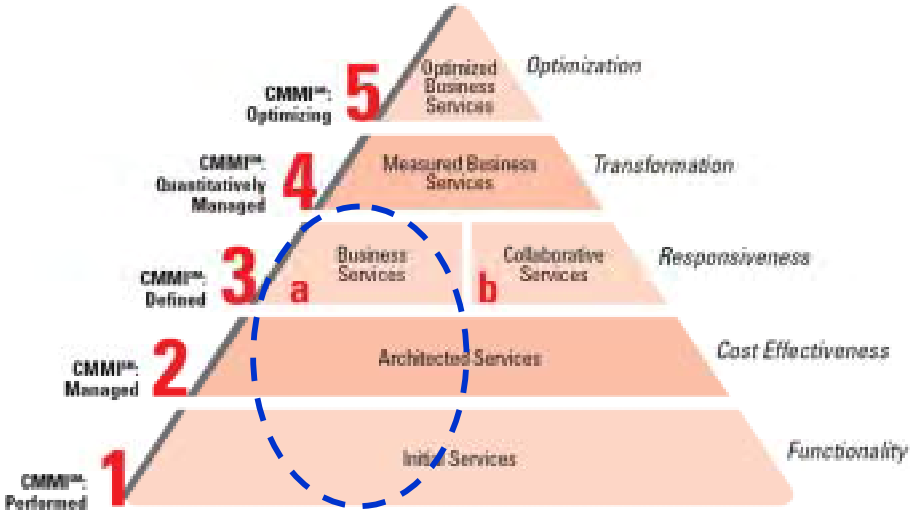

Figure VI.4. Le modèle de maturité associé aux SOA [Sonic el al. 2005]

VI.3. Modélisation des services d'entreprise

Plusieurs approches de modélisation de services ont été proposées aussi bien dans la communauté scientifique que dans l'industrie. Cependant, la majorité de ces approches ne sont pas flexibles du fait qu'elles sont fortement liées aux technologies et aux outils utilisés. Nous allons dans ce chapitre proposer un modèle de services permettant d'apporter une plus grande flexibilité et une intégration pour les systèmes d'information industriels. Nous allons en particulier mettre en exergue utilisation d'une double vision

de la modélisation : une vision métier des services et une vision technique des services. La séparation de ces visions reste fondamentale pour des raisons de flexibilité et d'évolutivité du système d'information et donc des entreprises.

VI.3.1. Spécialisation du modèle SOA de base

L'architecture de services (ASE) que nous proposons repose sur le modèle SOA de base que nous spécialisons. Le modèle d'architecture générique (ou de base) permet de définir un modèle d'interaction entre les principales entités qui interviennent dans une architecture SOA (cf. § III.8). Dans ce modèle, on peut distinguer les entités suivantes: le service (et sa description), le client de services, le fournisseur de services et le référentiel (registre) de services. Le méta-modèle permettant de décrire l'interaction qui existe entre ces entités est montré en figure VI.5.

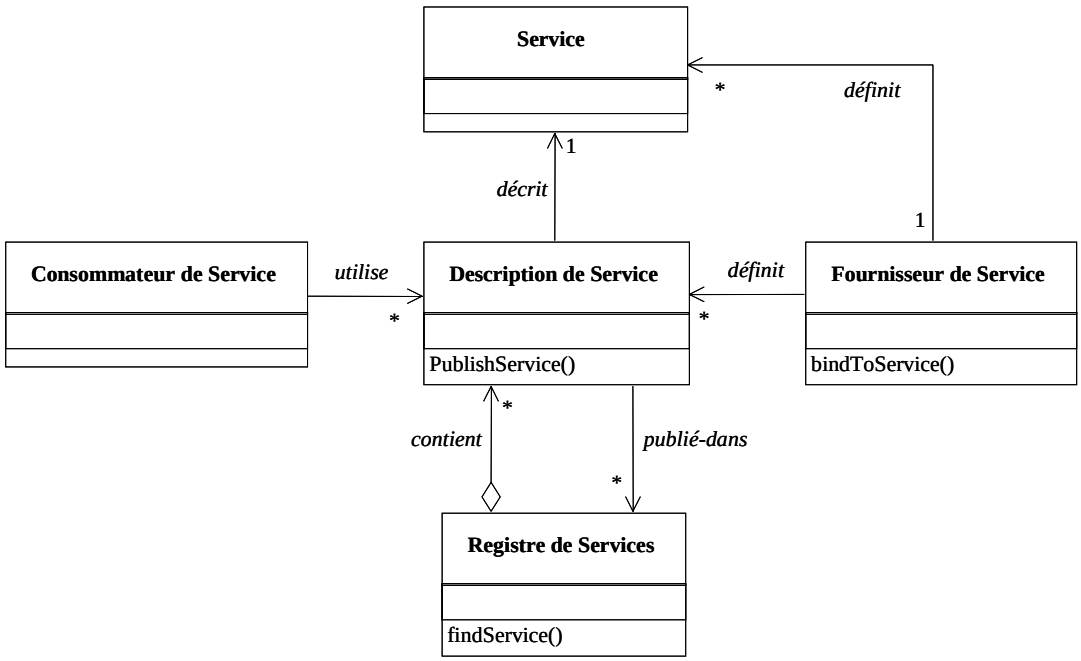

Figure VI.5.Méta-modèle SOA de base[38]

La spécialisation de cette architecture de base à notre cas permet de définir respectivement la notion de services d'entreprise, la notion de consommateur de service d'entreprise, la notion de fournisseur de service d'entreprise et la notion de référentiel de service d'entreprise. Ces notions sont décrites comme suit :

- *Service d'entreprise :* Il constitue l'entité de base de tout modèle orienté service et donc de notre architecture. Il décrit les services localisés à l'intérieur de l'entreprise. Un aspect fondamental des services d'entreprise est qu'ils sont caractérisés par un contrat de service d'entreprise.

[38] Signalons que nous nous basons tout au long de ce chapitre sur le formalisme UML pour représenter nos différents modèles et méta-modèles. Aussi, le sens des symboles graphiques utilisés dans ces modèles est celui préconisé dans UML 2.0. En particulier, les flèches avec triangles (à l'extrémité) désignent la relation de spécialisation/généralisation entre classes, les arcs possédant un losange à l'extrémité désignent des relations d'agrégation entre classes, et les arcs ou flèches simples désignent les autres types de relations entre classes.

- *Client de service d'entreprise :* Il constitue l'entité de l'entreprise qui recherche des services puis les exécute. Le client est typiquement un service d'entreprise qui désire une fonctionnalité distante. Du point de vue organisationnel, un client de service d'entreprise est une unité organisationnelle (site, département, acteur, etc.) qui effectue des requêtes de services.
- *Fournisseur de service d'entreprise :* Il constitue l'entité d'entreprise qui fournit les services et qui accepte d'exécuter les requêtes des clients. Il fournit aussi bien le contrat de service que son implémentation. Typiquement, un fournisseur de service est un service qui permet de répondre aux sollicitations du client. Du point de vue organisationnel, le fournisseur de service d'entreprise est toute unité organisationnelle qui est habilitée à définir et à exposer des services d'entreprise.
- *Registre (référentiel) de services d'entreprise :* Il constitue l'entité d'entreprise qui stocke et publie les descriptions des services élaborées par les fournisseurs de services d'entreprise et rendues disponibles pour les clients de services d'entreprise.

VI.3.2. Notion de service et de service d'entreprise

La première question que soulève l'architecture de services que nous voulons mettre en place est la définition même de la notion de "service ". Cette définition n'est pas si évidente si l'on en juge par les multiples définitions parfois contradictoires que nous pouvons rencontrer dans la littérature informatique et industrielle. Notre point de vue consiste à étendre et à compléter la vision classique de la SOA.

Rappelons que l'idée maîtresse de l'architecture orientée services est que tout élément du système d'information doit devenir un service. Cette architecture consiste fondamentalement en une collection de services qui interagissent et communiquent entre eux. Le service est un composant clef de l'architecture orientée services. Il est clairement identifiable, consiste en un ensemble de tâches bien définies, est documenté, fiable, autonome et localisable :

- *clairement identifiable :* identifier un service signifie être en mesure d'en connaître l'existence, indépendamment de sa localisation, de son mode de fonctionnement, de son mode d'appel, etc.
- *réalisant un ensemble de tâches parfaitement définies :* il s'agit de comprendre le fonctionnement global du service, c'est-à-dire de connaître la liste des tâches qu'il est capable de traiter, ses conditions opérationnelles d'exécution, les exceptions qu'il peut provoquer, etc.
- *documenté* : la documentation doit décrire clairement comment faire appel au service. Elle doit spécifier clairement (et si possible de manière unifiée, voire standardisée) les méthodes d'appels, les fonctions proposées et la signature du service.
- *fiable dans un contexte donné :* la fiabilité d'un service dépend naturellement de son contexte d'utilisation. Ceux-ci doivent être clairement identifiés et les conditions de fonctionnement du service explicitement décrites.
- *autonome et indépendant vis-à-vis d'autres services* : les services s'exécutent sans se soucier de l'état dans lequel peuvent se trouver d'autres services auxquels il fait éventuellement appel.
- *localisable :* le service possède une URL est il est accessible sur le réseau.

Pour les besoins de notre architecture de services d'entreprise (ASE), nous avons défini le service d'entreprise comme un cas particulier de service qui est localisable au sein de l'entreprise, Il correspond à toute entité de l'entreprise capable de répondre à des sollicitations en effectuant un certain nombre d'actions. Un service d'entreprise se doit d'assurer stabilité et assurance dans l'action qui lui est demandée. Il assure par ce biais un contrat d'interface et une qualité de service. Ces deux aspects forment le contrat de service qui représente ainsi le principal aspect descriptif permettant de le caractériser. La qualité de service porte généralement sur la disponibilité (taux de disponibilité), la réactivité (temps de réponse) et la robustesse (fonctionnement en mode dégradé).

VI.3.3. Typologie de services d'entreprise

La classification que nous proposons des services d'entreprise porte sur trois caractéristiques majeures d'un service qui sont : sa nature (degré d'abstraction), sa granularité (degré de composition) et sa visibilité (profil d'utilisation).

VI.3.3.1. Classification selon la nature

De par leur nature (ou degré d'abstraction), nous classifions les services d'entreprise en deux catégories majeures: les *services métier* et les *services informatiques* (ou *services IT*) [Izza et al. 2006-a].

VI.3.3.1.1. Services métier

Les *services métier* correspondent à des fonctionnalités métier. On peut distinguer des services fonctionnels et des services processus.

Les *services fonctionnels* sont de fine granularité comme par exemple des services offrant des fonctionnalités liées à la gestion d'un lot de puces microélectroniques, ou encore au calcul du taux de rebut de fabrication. Ce sont des services qui exposent la notion de fonction du système d'information, entité réutilisable et invocable du système d'information et qui permettent d'implémenter la notion de tâche ou d'activité métier.

Les *services processus* sont des services qui peuvent encapsuler des activités métier, voire des processus métier comme par exemple le processus de planification d'une tâche de maintenance préventive sur les équipements de fabrication de puces microélectroniques. Techniquement parlant, ces services encapsulent la logique du workflow de l'activité ou du processus métier qu'ils exposent. Typiquement les services processus sont composés de services fonctionnels qui à leur tour sont composés de services de plus bas niveau, - les services informatiques.

VI.3.3.1.2. Services informatiques

Les *services informatiques* (ou *services IT*) regroupent les fonctionnalités offertes par la partie informatisée du système d'information et qui sont utilisées par les services métier décrits précédemment. On distingue deux types de services informatiques qui sont les *services applicatifs* et les *services techniques*.

Les *services applicatifs* permettent d'encapsuler les applications informatiques de l'entreprise comme par exemple le service qui encapsule l'application informatique permettant de gérer les emplois du temps des techniciens qui interviennent dans les tâches de maintenance préventive sur les équipements. Les services applicatifs sont

décomposés à leur tour en *services logiciels* et *services de données*. Les *services logiciels* permettent d'exposer uniquement la partie logique applicative de l'application informatique sous forme d'un service, tandis que les *services de données* encapsulent des sources de données en permettant de gérer l'accès aux données.

Les *services techniques* regroupent les services de plus bas niveau permettant de gérer l'infrastructure du système d'information et l'intégration des services applicatifs. On distingue deux types de services techniques : les *services d'infrastructure* et les *services d'intégration*. Les *services d'infrastructure* permettent de gérer l'infrastructure du système d'information comme par exemple les services liés à la gestion du réseau et de la sécurité. Les *services d'intégration* sont des services plus complexes qui permettent d'intégrer les services applicatifs et/ou services métiers.

Par ailleurs, il est plus commode de séparer les services d'intégration des autres services que nous appelons pour plus de précision les services fondamentaux.

La figure VI.6, qui complète le méta-modèle simplifié présenté en figure V.5, permet de récapituler les principales catégories des services d'entreprise que nous retenons en fonction de la nature de ces derniers.

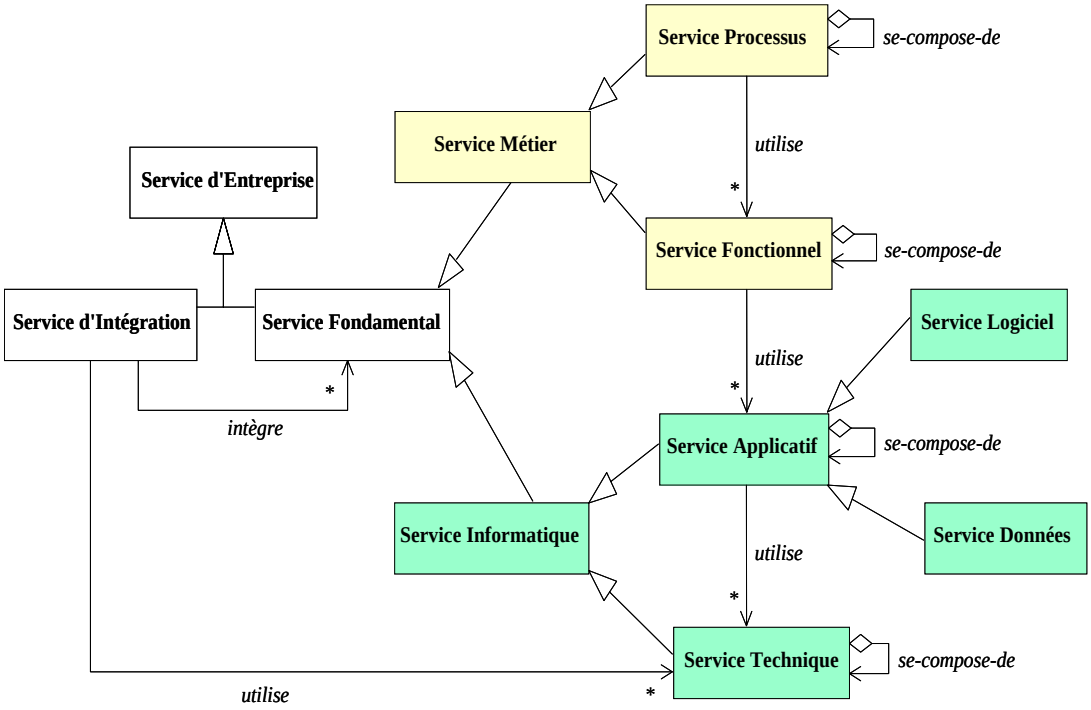

Figure VI.6. Typologie des services d'entreprise selon la nature

VI.3.3.2. Classification selon le degré de granularité

Les services d'entreprise peuvent également être classifiés selon leur degré de granularité. Aussi, il existe des services de fine granularité et des services de forte granularité. En règle générale, plus on monte dans l'échelle d'abstraction et plus les services sont des services de grosse granularité. Ce grossissement du degré de granularité en fonction du degré d'abstraction s'explique par la relation "utilise" de la figure VI.6. De plus, au sein même d'une même catégorie de services, il est possible d'avoir différents degrés de granularité des services grâce à la relation *"se-compose-de"*. Cette double catégorisation est schématisée plus clairement dans la figure VI.7 où le

niveau de gris utilisé évoque le degré de complexité lié à l'utilisation de la technique informatique.

Comme on peut le constater sur la figure VI.7, l'utilisation typique des services d'entreprise va dans le sens de la première bissectrice du repère. En effet, plus on descend dans les niveaux d'abstraction et plus simples sont les services définis. En revanche, plus on monte dans le niveau d'abstraction, et plus les services utilisés sont de complexes.

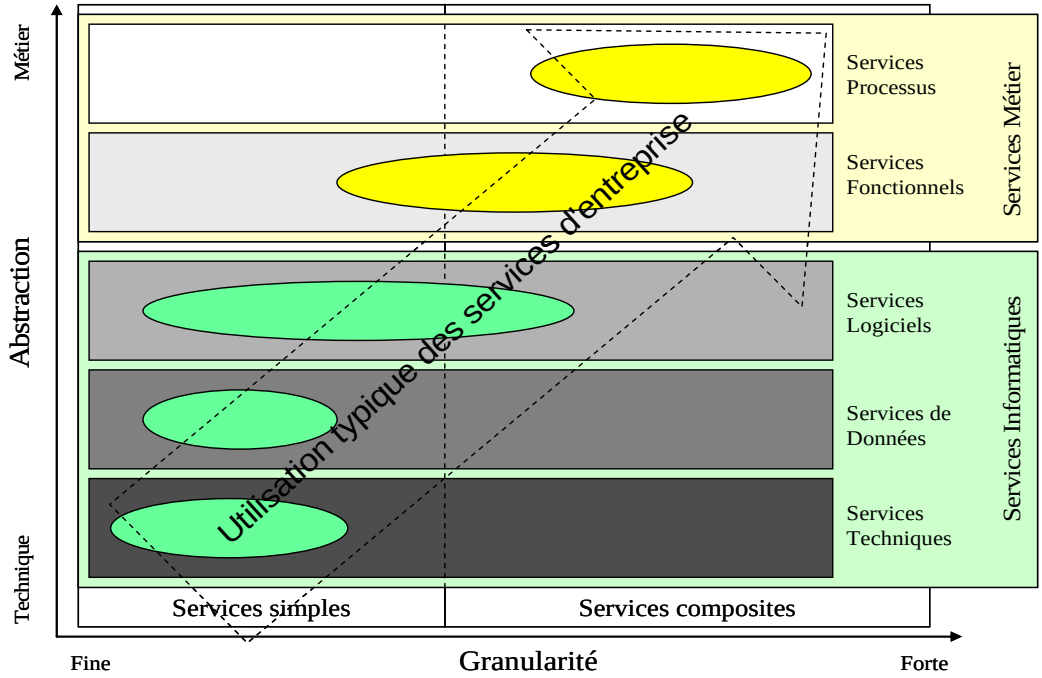

Figure VI.7.Principe de classification des services d'entreprise selon la granularité

VI.3.3.3. Classification selon la visibilité

Un autre aspect important pour la classification des services d'entreprise est la visibilité. Nous proposons deux catégories de visibilité et dans chaque catégorie nous définissons différents niveaux de visibilité (figure VI.8).

Cette notion de visibilité sous-entend que les services sont structurés en clusters. La problématique de clustérisation des services est traitée un peu plus loin dans ce chapitre (§ VI.5). Les catégories de visibilité que nous avons définies sont :

- Visibilité privée : un service donné est dit de visibilité privée si et seulement si uniquement les services d'entreprise (intra-entreprise) peuvent y accéder. Nous distinguons deux types de visibilité privée :
    - Visibilité locale : un service donné est dit localement visible si et seulement si uniquement les services locaux (appartenant au même cluster que le service considéré) peuvent y accéder.
    - Visibilité globale : un service est dit globalement visible s'il peut être accédé par tout autre service d'entreprise.
- Visibilité publique : un service est dit publiquement visible s'il est disponible à partir de l'extérieur de l'entreprise.

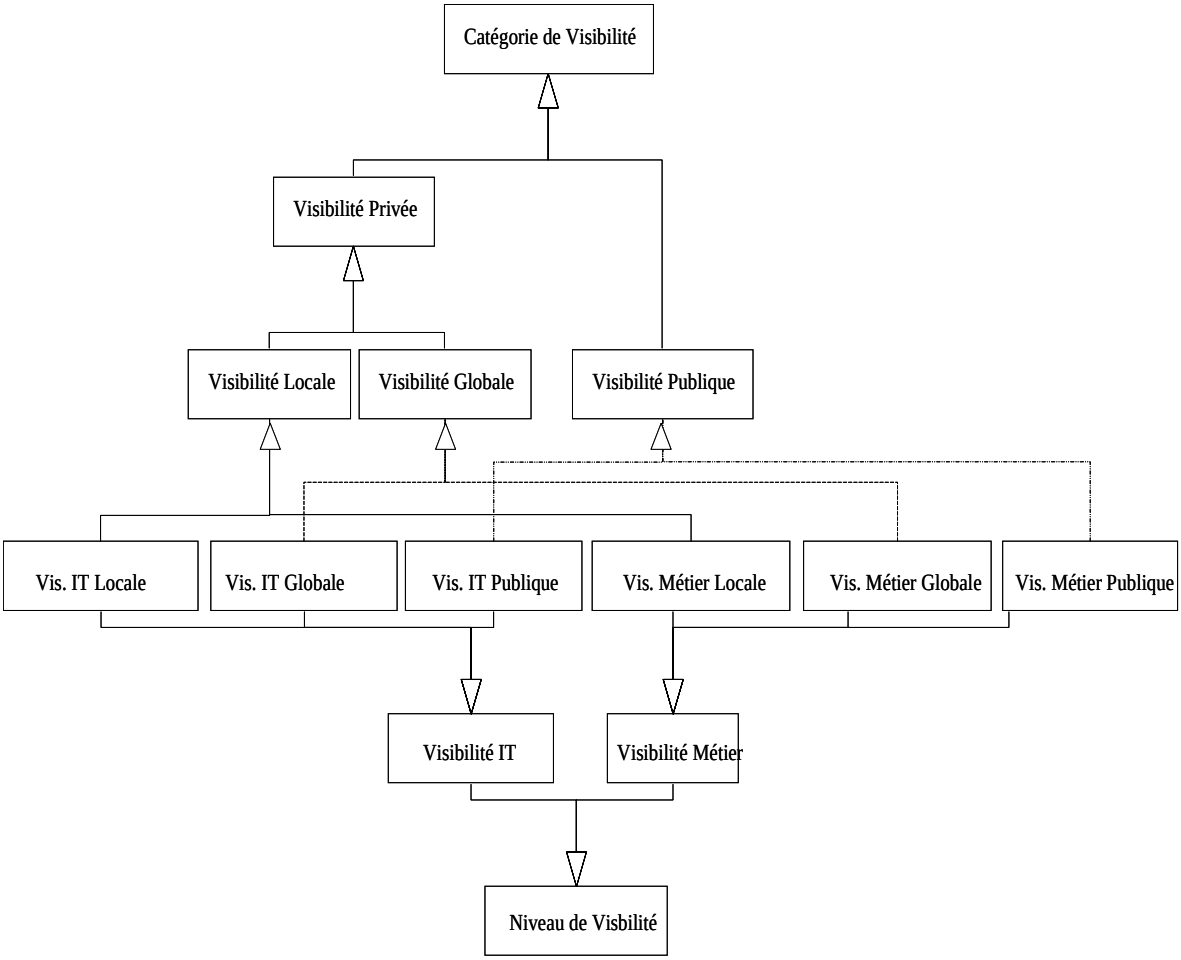

Figure VI.8. Visibilité des services d'entreprise

En outre, nous définissons pour chacune des catégories précédentes, deux niveaux principaux de visibilité qui découlent de la nature même des services. Ces niveaux sont :

- visibilité métier (ou visibilité de maîtrise d'ouvrage) : un service est de visibilité métier lorsqu'il est visible par les gens de la maîtrise d'ouvrage (les gens métiers). Ce sont donc les services métiers qui sont concernés par ce type de visibilité.
- visibilité IT (visibilité de maîtrise d'œuvre) : un service est de visibilité informatique lorsqu'il est visible par les gens de la maîtrise d'œuvre (les gens IT). Ce sont donc les services informatiques qui sont concernés par ce type de visibilité.

La notion de niveau de visibilité (métier versus informatique) est fondamentale. Elle permet de décomposer l'architecture de services d'entreprise en deux sous architectures faiblement couplées qui sont : la SOA IT et la SOA métier. Le principe de cette dichotomie est illustré en figure VI.9 qui montre d'une part la répartition des différents types de services à chacune des deux architectures et d'autre part les enjeux majeurs de chacune de ces architectures qui sont : l'enjeu de réutilisation pour la SOA IT et l'enjeu d'intégration pour la SOA métier.

De plus, la figure VI.9 montre les différents profils d'utilisateurs pouvant intervenir dans notre architecture de services. On distingue plus précisément :

- Le profil de fournisseur de services qui se décline en deux sous-profils : le profil maîtrise d'ouvrage (MOA) du fournisseur et le profil maîtrise d'oeuvre (MOE) du fournisseur de services. Le premier profil concerne les gens métiers qui interviennent dans la construction et l'utilisation du service alors que le second

profil concerne les gens IT (développeur et intégrateur) qui interviennent dans le développement et dans l'intégration du service.

- Le profil de consommateur de services qui se décline à son tour en deux sous-profils : le profil maîtrise d'ouvrage (MOA) du consommateur et le profil maîtrise d'oeuvre (MOE) du consommateur de services. Le premier profil concerne les gens métiers qui interviennent dans l'utilisation du service alors que le second profil concerne les gens IT (parfois des développeurs et surtout des intégrateurs) qui interviennent dans le processus d'intégration du service.

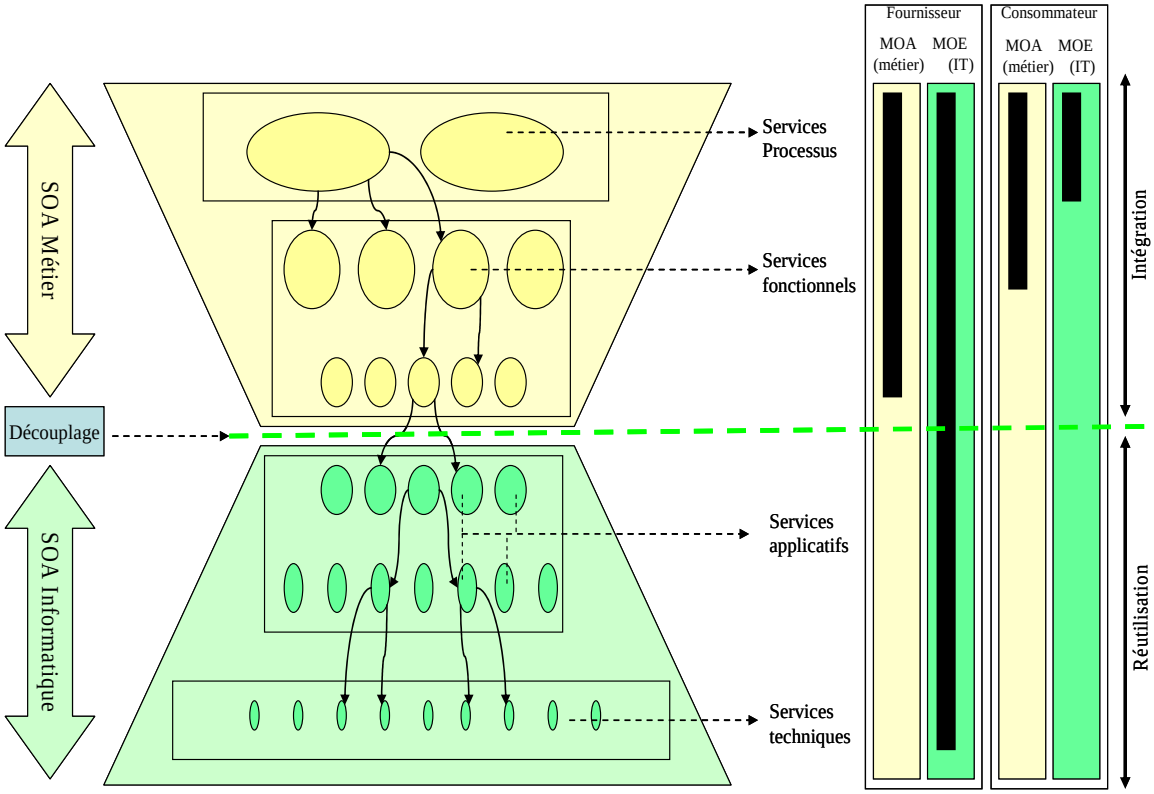

Figure VI.9. Dichotomie SOA métier – SOA IT

VI.4. Démarche de construction de la SOA

Du fait de l'existence d'une architecture double (SOA IT et SOA métier), il résulte une double problématique de construction de l'architecture de services : la problématique de construction de la SOA IT, et la problématique de construction de la SOA métier.

L'enjeu majeur de la SOA IT est la réutilisation de composants informatiques (composants applicatifs, composants techniques). De nos jours, la construction de la SOA IT est moins problématique. En effet, il existe une multitude de méthodes et d'outils relativement matures qui permettent de définir des services informatiques. La prolifération de méthodes peut s'expliquer par le fait qu'elles sont généralement dépendantes des outils de marché. Des exemples de tels méthodes et outils pouvant être utilisés dans le cadre de construction de la SOA IT sont : SAP NetWeaver [SAP 2005], Apache Beehive [Apache 2005], Lightweight Enterprise Integration Framework [RogueWave 2005], IBM WebSphere [IBM 2005], BEA WebLogic [BEA 2005], etc..

Dans le cadre de nos travaux, nous ne nous intéressons pas à la construction de la SOA IT. Plus précisément nous n'allons pas nous intéresser au développement d'applications orientées services qui relève plutôt de la discipline du génie logiciel. Nous allons nous focaliser plutôt sur la construction de la SOA métier qui est essentielle vis-à-vis de notre problématique d'intégration flexible des applications industrielles.

VI.4.1. Approche de construction de la SOA métier

Il existe fondamentalement deux approches pour la construction de logiciels qui sont l'approche descendante (bottom-up) et l'approche ascendante (top-down). L'approche descendante est généralement plus complexe mais complète, tandis que l'approche ascendante est plus pragmatique mais plus restreinte.

L'approche que nous proposons pour construire la SOA métier est une approche hybride dans le sens où elle est à la fois ascendante et descendante. De plus, notre approche est itérative et incrémentale dans la mesure où elle construit les services métier de façon progressive.

Un principe important de notre approche consiste à construire à partir des services applicatifs (résultat de la construction de la SOA IT) des services fonctionnels, comme un assemblage cohérent d'un certain nombre de services applicatifs. Ces services peuvent ensuite être composés à leur tour pour définir des services fonctionnels plus complexes ou des services processus.

Un autre principe important de notre approche de construction de la SOA métier est le principe d'urbanisation des services. Ce principe permet de structurer les services en clusters. Un cluster de services permet d'organiser les services d'entreprise et de les regrouper en fonction de leur contexte fonctionnel. Chaque cluster est considéré comme un macro-service possédant toutes les caractéristiques d'un service telles que l'unicité, l'autonomie, le contrat de service et la responsabilité sur les données (chaque donnée est sous la responsabilité d'un seul service). Chaque service n'appartient qu'à un et un seul cluster et les règles de clustérisation sont celles issues de l'urbanisation des systèmes d'information et qui portent sur le faible couplage et la forte cohésion.

VI.4.2. Démarche globale de construction de la SOA métier

La démarche que nous proposons pour la construction de la SOA métier comporte principalement quatre phases (figure VI.10) :

- *Exposer en services informatiques* : elle consiste à exposer sous forme de services les composants applicatifs et techniques de l'entreprise. Le résultat de cette phase est une SOA IT.
- *Définir les services métier* : elle permet de construire à partir de la SOA IT, construite précédemment, les services métiers qui composent la SOA métier. Le résultat de cette phase est donc une SOA métier. Il est important de préciser que cette phase nécessite l'approbation de la maîtrise d'ouvrage pour pouvoir faire "migrer[39]" un service applicatif ou un assemblage de services applicatifs vers un ou plusieurs services métier.

[39] Le terme "migrer" désigne ici le fait de faire transiter un composant existant vers un élément de l'architecture de services.

- *Urbaniser la SOA métier* : elle consiste à structurer les services métiers en blocs cohérents permettant de favoriser l'intégration. Le résultat de cette phase est un ensemble de clusters issus de l'urbanisation de la SOA métier.
- *Publier les services métier* : elle consiste à publier dans un référentiel approprié et standardisé les services métiers que la maîtrise d'ouvrage juge pertinents. Le résultat de cette phase est donc la publication des services métiers.

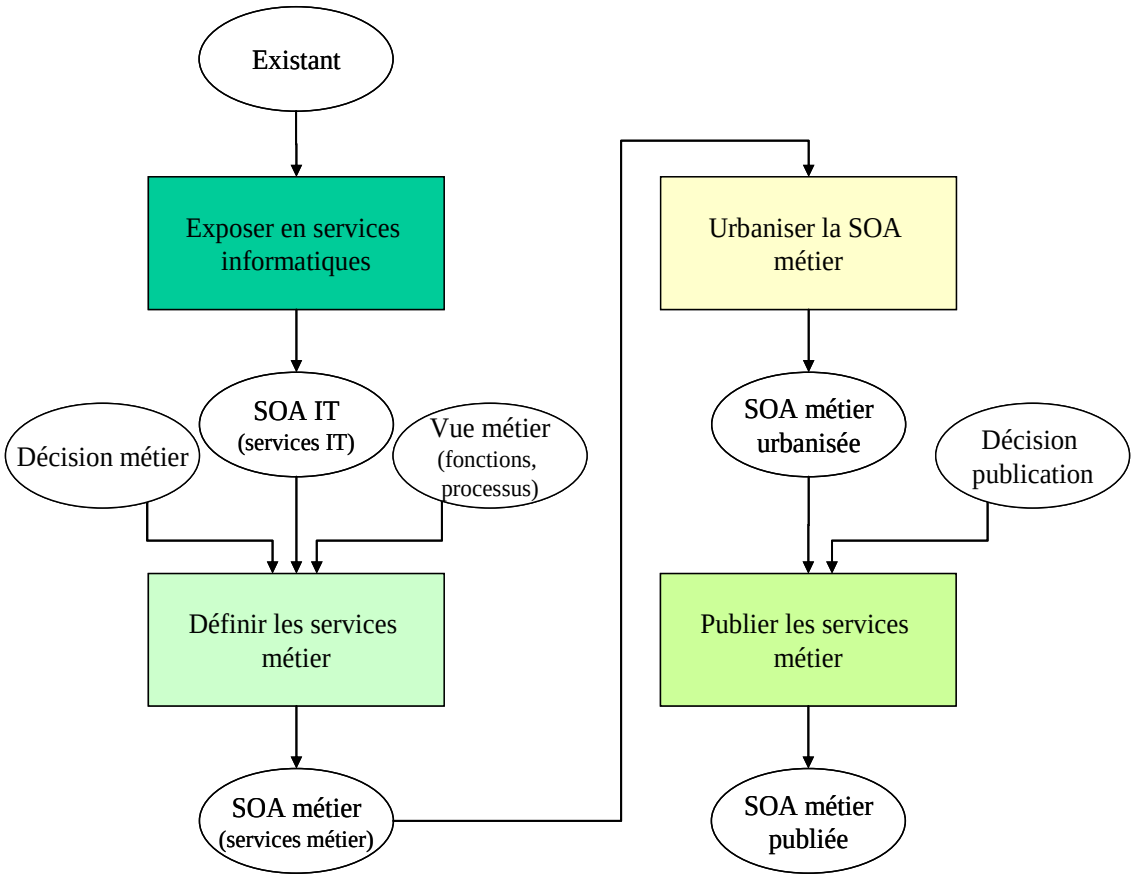

Figure VI.10. Démarche globale de construction de la SOA métier

VI.4.3. Exposer les composants en services informatiques

Le principe d'exposition des composants informatiques consiste à créer des services informatiques qui encapsulent les composants informatiques de l'entreprise. En d'autres termes, ceci revient à créer des liens entre les services et les composants existants. La figure VI.11, qui complète les figures V.5 et VI.6, permet d'illustrer le type de lien qui est généralement (1..*) – (0..*), ce qui signifie qu'un composant informatique peut être exposé en un ou plusieurs services et qu'un service donné peut correspondre à un ou plusieurs composants informatiques.

Ce scénario d'exposition est général et concerne le cas d'intégration de composants existants. Cependant, ce scénario demeure valide dans le cadre de développement orienté services. Dans ce cas, les applications sont développées avec un esprit orienté services. Les composants issus du développement peuvent donner naissance à des services informatiques durant ou après le projet de développement en fonction de leur degré de réutilisation. Une fois la décision de servicisation retenue, on procède à la

publication du service informatique qui va permettre de rendre accessible le service par les équipes IT (ou les équipes d'urbanisme).

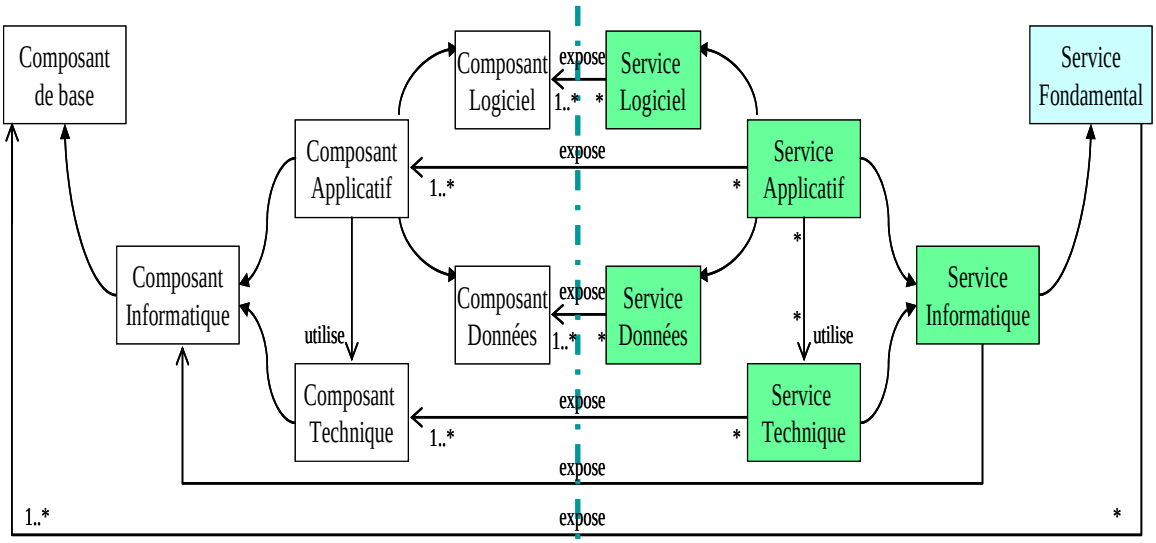

Figure VI.11. Principe d'exposition en services informatiques

Notons que les services informatiques sont de fine granularité (services unitaires). Ce sont les services qui sont à la base de la politique de réutilisation au sein de l'entreprise. Comme présenté précédemment, nous distinguons des services techniques et des services applicatifs. Ces derniers comportent à leur tour des services logiciels et des services de données. La dichotomie services logiciels/services données tient au fait que nous voulons mettre en oeuvre une dichotomie données / logiciels afin d'apporter plus de flexibilité à l'architecture de services.

Sans trop entrer dans le détail de l'exposition en services informatiques, rappelons que les deux "patterns" technologiques les plus utilisés pour l'exposition sont le "proxy" et la façade. Le "proxy" permet de définir des services unitaires en permettant l'accès à des composants distants (objets, modules, …), tandis que la façade définit des services composites par assemblage de plusieurs services unitaires.

VI.4.4. Définir les services métier

Le principe de définition des services métier consiste à créer des services qui ont un sens pour les gens métier. Cela requiert des assemblages de services (services informatiques et/ou de services métier) pour prendre en compte les besoins fonctionnels ou métier.

Les services fonctionnels sont considérés comme des assemblages de services informatiques et/ou de services fonctionnels qui permettent d'exposer une partie de fonction, une fonction ou un groupe de fonctions. Formellement, ceci se traduit par un lien de type (1..*) – (0..*), ce qui signifie qu'un composant fonctionnel peut être exposé en un ou plusieurs services et qu'un service fonctionnel donné peut correspondre à une ou plusieurs fonctions du système d'information. La figure VI.12, qui complète les figures V.5, VI.6 et VI.11, illustre le principe de définition des services fonctionnels.

Les services fonctionnels sont des services qui peuvent être de fine ou de moyenne granularité, mais rarement de grosse granularité. Ce sont donc soient des services

unitaires soient des services composites intermédiaires. Ce qui caractérise fondamentalement ces services est le fait qu'ils sont à la base de l'intégration des systèmes d'information.

Nous devons noter que la définition des services fonctionnels est une tâche complexe qui met en jeu à la fois la maîtrise d'ouvrage et la maîtrise d'oeuvre. En effet, afin de définir des services fonctionnels, la maîtrise d'oeuvre se base sur les services IT définis précédemment, et sur les décisions métier qui émanent de la maîtrise d'ouvrage et qui portent sur l'intérêt de définir un service fonctionnel donné. De plus, nous devons noter que la tâche de définition des services fonctionnels se base sur les fonctions existantes du système d'information qui servent pour effectuer des réajustements afin de mieux définir des services réutilisables.

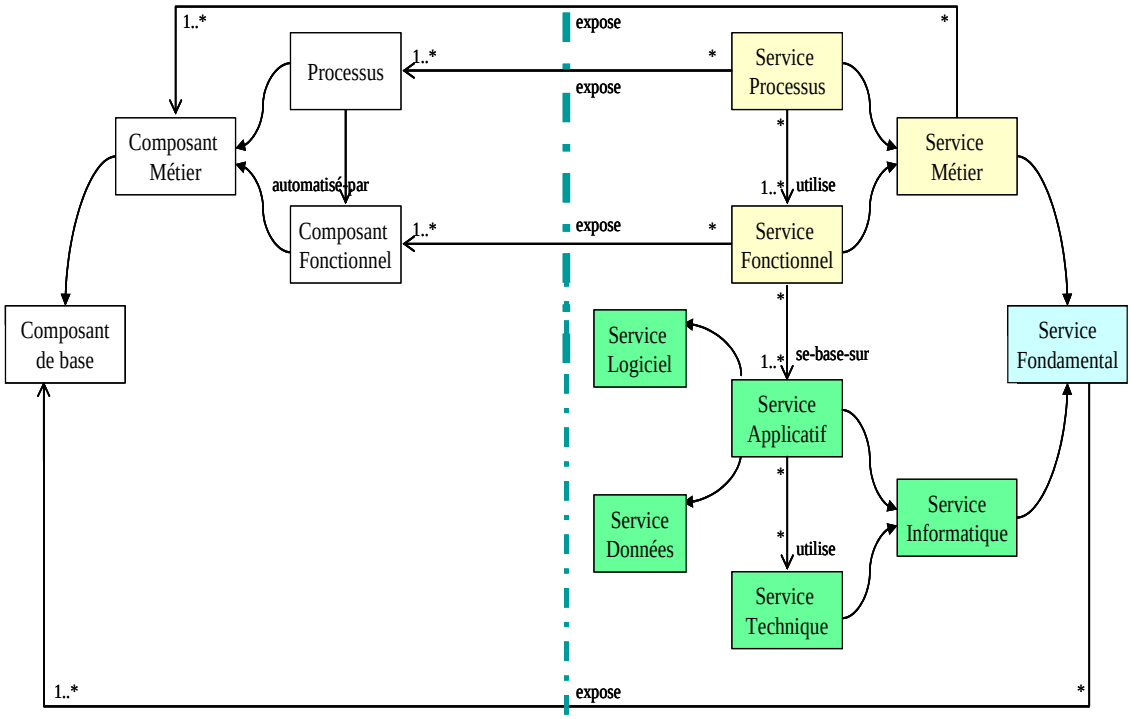

Figure VI.12. Principe de définition des services métier

En ce qui concerne les services processus, ils sont considérés comme des assemblages de services fonctionnels qui permettent d'exposer une partie d'un processus métier, un processus métier ou carrément un regroupement de processus métier. Formellement, ceci se traduit, comme dans le cas précédent, par un lien de type (1..*) – (0..*), ce qui signifie qu'un processus métier peut être exposé en un ou plusieurs services processus et qu'un service processus donné peut correspondre à un ou plusieurs processus métier (figure VI.12).

En principe, nous définissons un service processus chaque fois que nous désirons faire de l'orchestration de services fonctionnels. Les services processus sont des services de grosse granularité (services composites). Ce sont des services qui sont à la base de l'orchestration de services fonctionnels. De même que pour les services fonctionnels, la tâche de construction des services processus nécessite à la fois l'intervention de la maîtrise d'ouvrage (décision métier) et la maîtrise d'œuvre. De plus, des réajustements sont généralement effectués afin que les services définis puissent être alignés sur les processus d'entreprise.

VI.4.5. Urbaniser les services métier

Afin de mieux aider l'entreprise dans son processus de mise en œuvre d'une architecture de services, notre approche de construction se base sur le concept d'urbanisme orienté services qui constitue une extension de l'urbanisme classique des systèmes d'information [Sassoon 1998][Longépé 2001] et qui permet de mieux structurer les services métier afin de faciliter et de favoriser le processus d'intégration des services métiers.

Le principe de l'urbanisation orientée services que nous proposons consiste à regrouper des services semblables et/ou fortement couplés au sein d'un cluster (ou un macro-service) afin de faciliter l'intégration intra-cluster (intra-macro-service). Nous appelons les clusters (macro-services) ainsi construits des quartiers de services. Un second niveau de clustérisation est également utilisé pour regrouper les quartiers de services qui présentent des similitudes et une forte cohérence en clusters de niveau supérieur afin de faciliter l'intégration inter-quartiers. Nous appelons les clusters de quartiers des zones de services. La figure VI.13 récapitule le principe général de clustérisation des services métier. La section VI.5 présente de façon détaillée la démarche d'urbanisation des services métier.

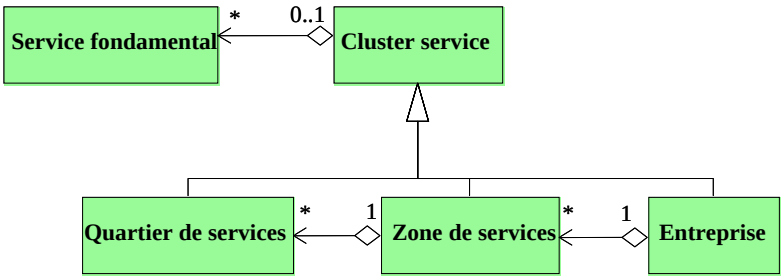

Figure VI.13. Clustérisation des services fonctionnels

Afin de mieux illustrer le principe d'utilisation de la clustérisation dans la résolution du problème d'intégration des systèmes d'information industriels, nous avons choisi de reprendre le scénario typique d'intégration inter-clusters (figure VI.14). Dans ce scénario, chaque cluster est considéré comme un fournisseur de services dans la mesure où il expose un certain nombre de services métier. D'autres clusters peuvent jouer le rôle de consommateur dans le sens où ils peuvent comporter des services métier qui peuvent réutiliser d'autres services exposés par d'autres clusters. Lorsque nous désirons intégrer deux services quelconques issus de deux clusters différents, nous définissons une situation d'intégration inter-clusters. Il faut alors mettre en place un mécanisme d'intégration entre le cluster consommateur et le cluster fournisseur.

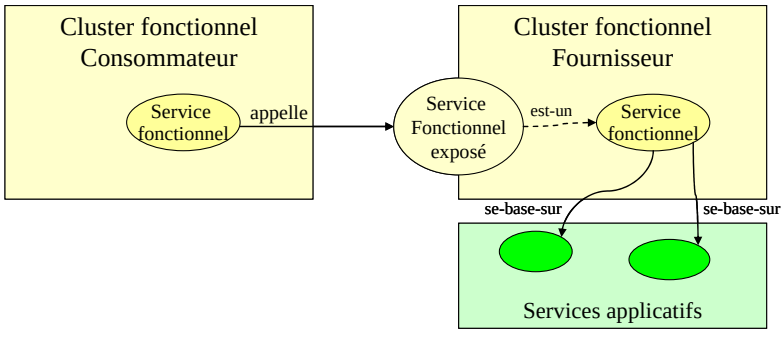

Figure VI.14. Scénario d'intégration inter-clusters

VI.4.6. Publier les services fondamentaux

La dernière étape de notre démarche méthodologique de construction de l'architecture de services est la publication de services fondamentaux. Cette publication repose sur l'utilisation d'un référentiel de services permettant d'exposer les services disponibles aux différentes unités organisationnelles de l'entreprise. Ce référentiel joue le rôle de tiers d'intermédiation. Le principe d'utilisation de technologies standardisée telle que UDDI est fondamental pour donner plus d'ouverture à la solution.

Plusieurs stratégies de publication peuvent être mises en œuvre. Il s'agit principalement de la publication centralisée (un UDDI pour toute l'entreprise), de la publication distribuée (un UDDI pour chaque cluster) et de la publication câblée (utilisation d'une logique câblée pour publier des services). Dans le cadre de notre approche, nous avons retenu une double vision : une vision logique et une vision technique. La vision technique correspond aux différents types de déploiements d'UDDI comme mentionnés précédemment, alors que la vision logique permet de définir des référentiels logiques indépendamment de toute implantation physique. Ceci permet notamment de créer des solution plus flexibles, où les changements se traduisent par le changement au niveau des liens entre les référentiels logiques et les référentiels physiques.

Logiquement, nous définissons les référentiels de services suivants (figure VI.15) :

- *Référentiel local* : qui est un référentiel associé à un quartier de services et qui contient toutes les descriptions des services d'un quartier;
- *Référentiel de domaine* : qui est un référentiel associé à un domaine et qui contient tous les descriptions des services exposés par le domaine ainsi que la liste de tous les référentiels locaux;
- *Référentiel global* : qui est un référentiel associé à l'entreprise et qui contient toutes les descriptions des services exposés par l'entreprise ainsi que la liste de tous les référentiels de domaine.

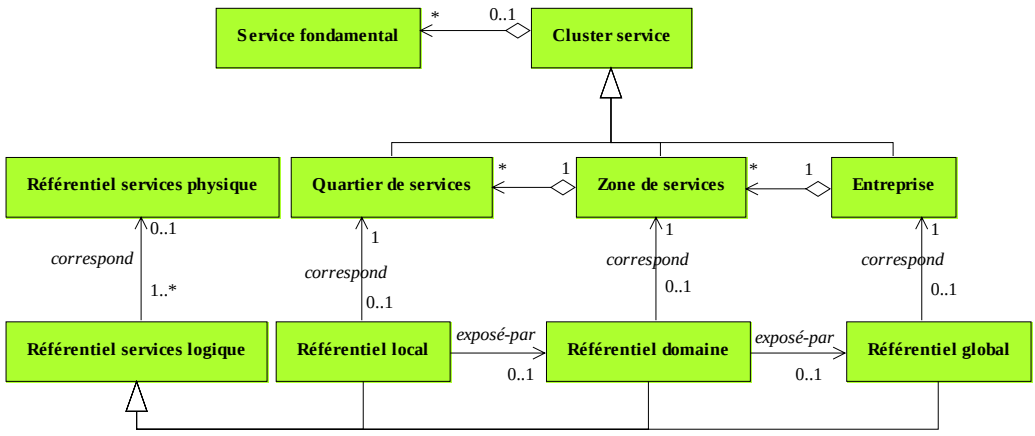

Figure VI.15. Principe de définition des référentiels logiques et physiques

Notons également que cette répartition logique/physique est techniquement faisable grâce à la version UDDI 3.0 qui est orientée beaucoup plus vers de grandes entreprises, avec notamment la notion de *service privé* et de *service public*, et également avec la notion d'*annuaires distribués.* Rappelons que la notion d'annuaires distribués permet de définir des référentiels complexes, en fait des réseaux de référentiels, où les référentiels

individuels (qu'ils soient des référentiels publics ou des référentiels privés) peuvent être connectés entre eux.

Le processus de publication syntaxique est pris en charge par le service de publication (Publication Service). Ce dernier est un service d'intégration particulier qui s'active dès réception d'une requête de type :

$$request(\#Publish, PublicationType, Publication)$$

où *PublicationType* désigne le type de publication (dans ce cas publication syntaxique de service), et où *Publication* désigne les caractéristiques sous forme de fichier XML à publier. Nous allons étudier dans le chapitre suivant une autre fonctionnalité de ce service de publication qui est la publication sémantique.

La publication syntaxique d'un service fondamental est effectuée en sauvegardant certaines informations des descriptions syntaxiques WSDL au sein des référentiels privés UDDI de services. Cette publication est effectuée en utilisant l'API standard *PublishService* (figure VI.16).

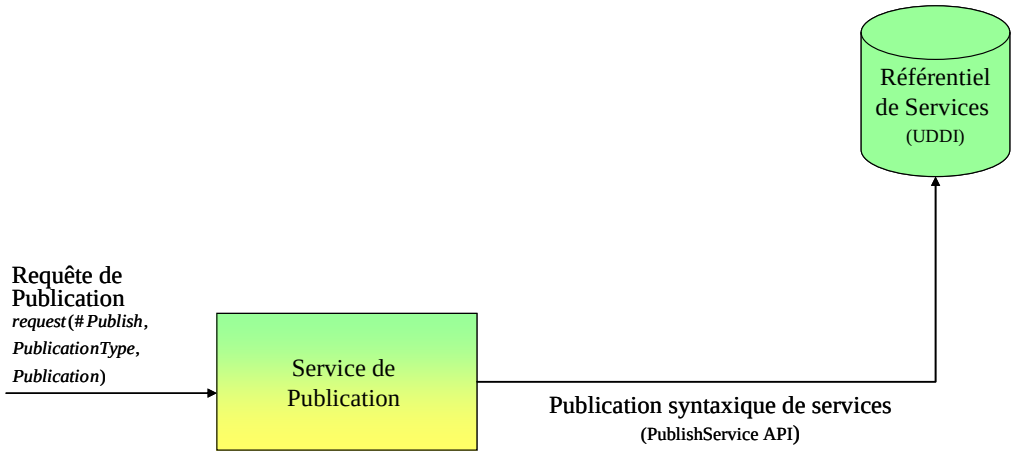

Figure VI.16. Principe de fonctionnement du service de publication

VI.5. Urbanisation orientée service des SII

VI.5.1. Nature du problème d'urbanisation orientée services

L'urbanisation orientée services consiste à structurer le système d'information de façon à favoriser et à simplifier l'intégration des services (applications). Comme nous l'avons déjà évoqué, il s'agit fondamentalement de regrouper les services similaires afin de simplifier l'architecture d'intégration que nous voulons mettre en place, c'est à dire, simplifier les échanges inter-clusters.

La clustérisation des services désigne l'opération qui permet de définir un regroupement ou une classification des services. Il faut noter que la clustérisation des services dans notre cas est assez problématique. En effet, nous voulons regrouper des services similaires pour faire en sorte que la construction de l'architecture sémantique soit aisée. Or pour que la clustérisation des services soit efficace, elle doit reposer sur cette même sémantique. Mais comme la sémantique des services ne peut être construite efficacement qu'à l'issue de la clustérisation, il devient donc assez problématique de

construire une telle clustérisation. Aussi, nous avons décidé de construire la clustérisation en tenant compte principalement des considérations syntaxiques qui sont actuellement disponibles au niveau de la couche services, et en effectuant ensuite des réajustements semi-automatiques afin de garder la cohérence tout au long du processus de clustérisation.

Une analyse fine du problème de la clustérisation des services nous a amené à décomposer le processus de clustérisation en deux phases majeures, comme peut le montrer la figure VI.17, et qui sont : le calcul de similarité des services et la détermination des clusters des services.

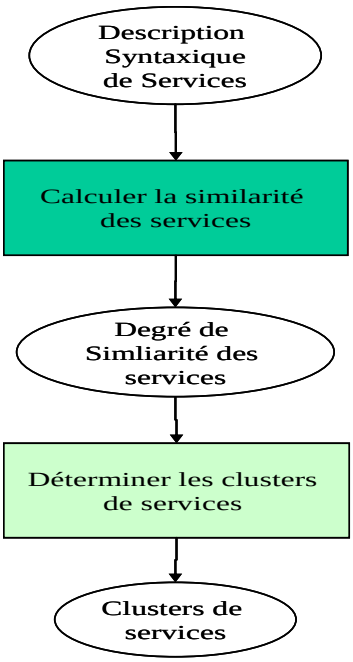

Figure VI.17. Démarche de clustérisation

La clustérisation des services est moins problématique que le calcul de similarité. Ce dernier est complexe du fait qu'il nécessite de mettre en place des approches semi-automatiques de comparaison de services. Le caractère semi-automatique de l'approche de calcul de similarité est important du fait qu'elle peut prendre en compte les multiples interactions humaines afin de corriger et de rectifier certaines mesures de similarité. Du fait que le calcul de similarité est la phase la plus critique, il est donc possible de catégoriser les différentes approches, qui nous paraissent les plus pertinentes, que l'on peut mettre en œuvre pour urbaniser les services en fonction principalement du type d'approche utilisé pour calculer la similarité des services. Nous distinguons les trois approches suivantes :

- Approche 1 - calcul manuel de la similarité : elle consiste à mettre en œuvre une approche manuelle pour définir la similarité des service en se basant sur l'appréciation que l'utilisateur donne au degré d'affinité des paramètres de services. Ces degrés d'affinités seront ensuite utilisés pour définir automatiquement des clusters de services. Cette approche est très réaliste et très simple à mettre en œuvre, mais elle peut présenter l'inconvénient de ne pas attirer l'attention de l'utilisateur sur certaines affinités implicites du fait que la première phase se fait de façon manuelle.

- Approche 2 - calcul semi-automatique de la similarité en se basant sur la syntaxe : elle consiste à mettre en œuvre des approches basées sur la syntaxe et en particulier les approches lexicales. Ces approches vont permettre de calculer la similarité syntaxique entre les services en se basant sur les techniques d'indexation de documents. Une fois que les degrés de similarité sont calculés, l'utilisateur intervient pour les valider et/ou les corriger. A l'issue de cette étape, la clustérisation est lancée de façon similaire que dans l'approche précédente.
- Approche 3 - calcul semi-automatique de la similarité en se basant sur la sémantique : cette approche consiste à mettre en œuvre un mécanisme d'extraction de la sémantique des services en faisant traduire les fichiers de description WSDL en fichiers OWL-S et en procédant à leur enrichissement pour qu'ils puissent servir ensuite de base au calcul de similarité entre ces ontologies de services. Cette approche est intéressante, mais demande des efforts importants notamment dans la l'enrichissement des fichiers OWL-S du fait que les ontologies de domaine ne sont pas encore construites. Ce qui écarte l'utilisation de ce type d'approche dans le cadre de nos travaux.

Notre approche d'urbanisation se base sur l'approche 2, c'est à dire sur le calcul semi-automatique de la similarité de services en se basant sur la syntaxe. Ce choix est motivé par le fait que jusqu'à présent nous ne possédons que de descriptions syntaxiques des services d'entreprise, et que notre approche d'urbanisation va constituer une base fondamentale pour la construction de la couche sémantique. Dans ce qui suit, nous allons présenter notre technique de calcul de similarité et aussi l'approche que nous avons retenue pour la clustérisation.

VI.5.2. Similarité des services

Notre technique de calcul de similarité exploite les informations disponibles sur les services (disponibles dans les fichiers WSDL et dans le référentiel de notre framework) et se base sur l'heuristique suivante : *Deux services sont similaires s'ils présentent* (figure VI.18)*:*

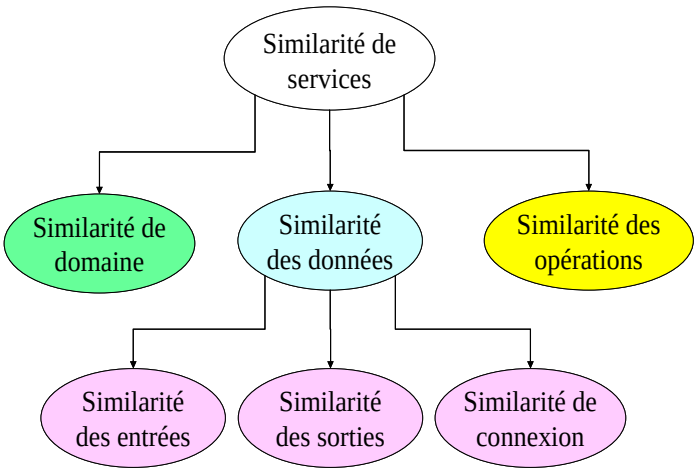

Figure VI.18. Similarité des services

- des données en entrée similaires;
- des données en sortie similaires;

- un enchaînement potentiel de services (les données en entrée de l'un sont similaires aux données en sortie de l'autre service);
- des opérations similaires,
- et des domaines fonctionnels similaires.

Plus formellement, nous définissons la similarité de deux services S_k et S_l comme étant la somme pondérée des similarités des caractéristiques des deux services :

$$\delta(S_k, S_l) = \omega_e.\delta_e(S_k, S_l) + \omega_s.\delta_s(S_k, S_l) + \omega_c.\delta_c(S_k, S_l) + \omega_o.\delta_o(S_k, S_l) + \omega_d.\delta_d(S_k, S_l)$$

$$= \sum_{x \in \{e,s,c,o,d\}} \omega_x \cdot \delta_x(S_k, S_l)$$

où $\delta_e(S_k, S_l)$ désigne la similarité des données en entrée des deux services,

$\delta_s(S_k, S_l)$ désigne la similarité des données en sortie des deux services,

$\delta_c(S_k, S_l)$ désigne le degré de connexion potentielle entre les deux services, c-à-d: la similarité des données en sortie de l'un des services avec les données en entrée de l'autre service,

$\delta_o(S_k, S_l)$ désigne la similarité des opérations des deux services,

$\delta_d(S_k, S_l)$ désigne la similarité du domaine fonctionnel des deux services,

et ω_e, ω_s, ω_c, ω_o et ω_d sont des poids dans l'intervalle [0, 1] tels que $\omega_e + \omega_s + \omega_c + \omega_o + \omega_d = 1$.

Nous devons noter que chacune des similarités ($\delta_e(S_k, S_l)$, $\delta_s(S_k, S_l)$, $\delta_c(S_k, S_l)$, $\delta_o(S_k, S_l)$ et $\delta_d(S_k, S_l)$) est normalisée dans le sens où elles prennent leurs valeurs dans l'intervalle [0, 1]. Comme conséquence de cette construction, la similarité globale $\delta(S_k, S_l)$ de deux services S_k et S_l et aussi normalisée. De plus, toutes les fonctions de similarité (δ et δ_x, $x \in \{e,s,c,o,d\}$) respecte les propriétés génériques d'une fonction de similarité, c'est à dire :

- Normalité : $\forall (S_k, S_l) \in \Im^2 : \delta(S_k, S_l) \in [0,1]$ ($\Im$ étant l'ensemble des services)
- Maximalité : $\forall (S_k, S_l, S_m) \in \Im^3 : \delta(S_k, S_k) \geq Sim(S_l, S_m)]$
- Symétrie : $\forall (S_k, S_l) \in \Im^2 : \delta(S_k, S_l) = \delta(S_l, S_k)$

L'évaluation des similarités partielles $\delta_e(S_k, S_l)$, $\delta_s(S_k, S_l)$, $\delta_c(S_k, S_l)$, $\delta_o(S_k, S_l)$ et $\delta_d(S_k, S_l)$ que nous proposons s'effectuent selon les règles suivantes :

$$\delta_e(S_k, S_l) = \psi(Entrée(S_k), Entrée(S_l))$$

$$\delta_s(S_k, S_l) = \psi(Sortie(S_k), Sortie(S_l))$$

$$\delta_c(S_k, S_l) = \psi(Entrée(S_k), Sortie(S_l))$$

$$\delta_o(S_k, S_l) = \psi(Opération(S_k), Opération(S_l))$$

$$\delta_d(S_k, S_l) = 1 \text{ si } Domaine(S_k) = Domaine(S_l), 0 \text{ sinon.}$$

où *Entrée*(S_x) : ensemble des entrées du service S_x;

Sortie(S_x) : ensemble des sorties du service S_x;

Opération(S_x) : ensemble des opérations du service S_x;

Domaine(S_x) : domaine fonctionnel du service S_x;

$\psi(A, B)$: désigne la fonction de similarité des deux ensembles de termes A et B.

VI.5.2.1. Calcul des similarités partielles

Nous avons défini la fonction ψ qui estime le degré ressemblance de deux ensembles en nous basant sur des techniques de la recherche documentaire. Il existe dans la littérature plusieurs techniques qui peuvent être utilisées pour ce calcul. Les mesures de similarité les plus populaires sont : les mesures lexicales (la distance de Hamming, la similarité de sous chaînes, la distance N-gram) [KnowledgWeb 2004], la technique TF-IDF [KnowledgeWeb 2004], le modèle LSI (Latent Semantic Indexing) [Derwester 1990] et le modèle probabiliste [Robertson et Jones 1976][Risjbergen, 1979], etc. Dans le cas de nos travaux, nous avons retenu la méthode TF-IDF comme fondement de base pour calculer les similarités partielles (la fonction ψ). Le choix de cette technique est essentiellement motivé par sa simplicité de mise en œuvre. Dans notre cas, les documents sont des ensembles de termes E_1, …, E_n qui correspondent aux caractéristiques des services dont nous voulons calculer la similarité. Par exemple, quand nous allons calculer la similarité $\delta_e(S_k, S_l)$ des entrées de services, nous allons nous intéresser aux documents constitués par les entrées des services à comparer, comme des ensembles de termes lexicaux. Les documents à considérer sont donc dans ce cas : $E_1, \ldots, E_n$ où E_k = *Entrée*(S_k). Nous appliquons le même raisonnement quand il s'agit de comparer les autres caractéristiques des services: les sorties, la connexion et les opérations des services.

La formule de calcul de la fonction similarité syntaxique ψ est définie comme suit :

$$\Psi(E_k, E_l) = \cos(x_k, x_l) = \frac{x_k^T x_l}{\sqrt{x_k^T x_k}\sqrt{x_l^T x_l}}$$

où $x_k = (x_{k1}, \ldots, x_{km})$, $x_k = (x_{l1}, \ldots, x_{lm})$ sont deux vecteurs de poids TF-IDF correspondants aux deux documents E_k et E_l,

m est le nombre total de termes dans la collection $\{E_1, \ldots, E_n\}$,

et x_{ij} désigne la mesure TF-IDF d'un terme w_j dans le document E_i et qui est calculée selon la formule traditionnelle de la technique TF-IDF :

$$x_{ij} = TF_{ij}.IDF_j = \frac{n_{ij}}{\|E_i\|}\log(\frac{n}{n_j})$$

où $n_{i,j}$ dénote le nombre d'occurrences du terme w_j dans le document E_i,

n_j représente le nombre de documents qui contiennent le terme w_j,

et $||E_i||$ désigne le nombre total de termes dans le document E_i.

Afin de mieux comprendre le comportement de la fonction $x_{i,j}$, nous avons décidé de tracer la courbe d'équation $z = f(x, y) = x\log(y)$, avec $x = \frac{n_{ij}}{\|E_i\|}$ et $y = \frac{n}{n_j}$. Les deux tracés de la figure VI.19 permettent ainsi d'illustrer selon deux angles de vue différents le comportement de cette courbe.

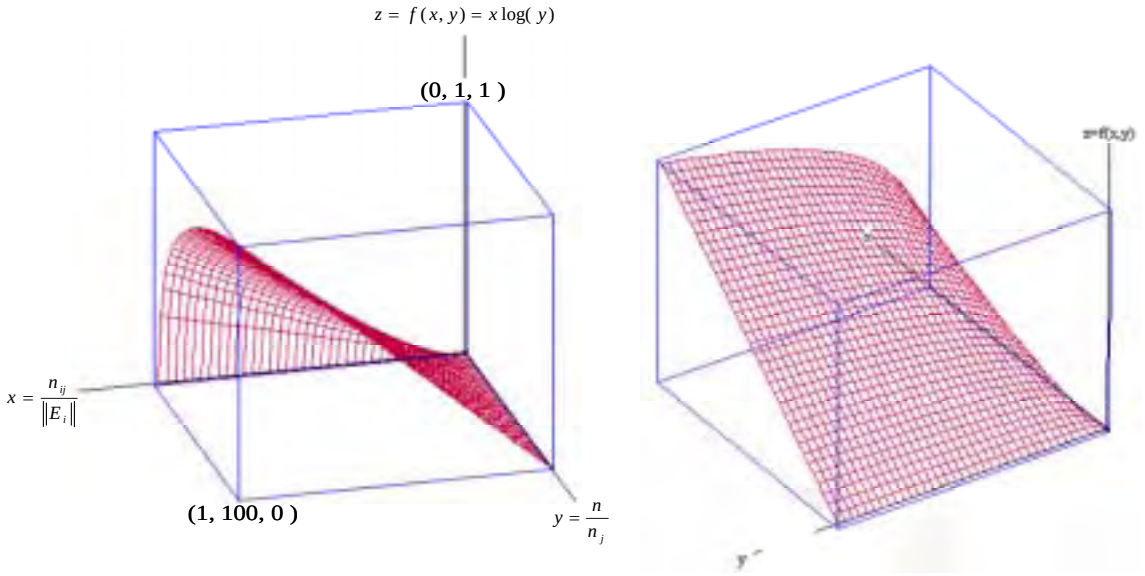

Figure VI.19. Courbes de la fonction $x_{i,j}$

Comme on peut le constater, $x_{i,j}$ possède un comportement tel que nous le désirons. Plus x et y tendent vers leurs bornes inférieures respectives (ce qui veut dire que le terme en question est peu important) et plus la fonction calculée tende vers zéro (ce qui valide alors le comportement de la fonction du fait que le poids calculé du terme s'approche de zéro). De plus, lorsque les variables x et y tendent vers leurs bornes maximales, la fonction s'approche de 1 (ce qui valide aussi le comportement de la fonction dans le cas où le terme en question possède un poids important).

VI.5.3. Détermination des clusters de services

Une fois que les degrés de similarité de services ont été calculés, ces derniers sont exploités pour déterminer les clusters de services. Pour cela, nous exploitons les techniques de classification automatique. Il existe dans la littérature deux types de méthodes de classification : l'approche supervisée et l'approche non supervisée.

Dans la première approche (la classification supervisée), on connaît au préalable les classes possibles et on dispose d'un ensemble d'objets classés qui servent d'apprentissage. Cet ensemble est ensuite utilisé comme référence pour associer à tout nouvel objet sa classe la plus appropriée, par apprentissage. Il s'agit en particulièrement de l'approche utilisée pour catégoriser les services en leur associant automatiquement des concepts d'une certaine taxonomie définie au préalable. Les travaux de cette catégorie qui nous paraissent les plus pertinents sont ceux de [Heib et Kushmerick 2003][Corella et Castells 2006][Bruno et al. 2005].

Dans la seconde approche (la classification non supervisée ou clustérisation), les classes possibles ne sont pas connues a priori. Le but est de regrouper au sein d'un même cluster

les objets considérés comme similaires. C'est ce type d'approche qui nous intéressent plus particulièrement dans le cadre de l'urbanisation de services qui, rappelons le a pour but de construire des clusters de services en respectant les principes de cohésion forte (qui consiste à maximiser la similarité des services à l'intérieur d'un cluster) et le couplage faible (qui consiste à minimiser la similarité des objets entre les clusters).

Il existe de nombreuses méthodes de clustérisation. On distingue fondamentalement deux catégories de méthodes : les méthodes de clustérisation hiérarchique et les méthodes de clustérisation non-hiérarchiques [Arabie et al. 1996].

Les méthodes hiérarchiques ont pour but de former une hiérarchie de clusters où chaque cluster regroupe des objets les plus semblables, et ce en procédant par fusion de petits regroupements en de plus grands regroupements d'objets. Le principe majeur de ce type de classification tient au fait qu'elle procède séquentiellement en regroupant les observations les plus "semblables" en premier lieu.

Les méthodes non-hiérarchiques (partitionnement) sont celles qui ne fournissent pas de hiérarchies mais simplement des partitions d'objets, chaque partition représentant alors un cluster. Le principe du partitionnement consiste à diviser simultanément un ensemble d'objets en sous-ensembles disjoints.

Sans trop entrer dans le détail des méthodes de clustérisation, nous pouvons simplement signaler le fait qu'elles peuvent être ascendantes ou descendantes[40], déterministes ou stochastiques[41], incrémentales ou non incrémentales[42], précises ou floues[43], monothétiques ou polythétiques[44] [Candillier 2004].

Dans le cadre de nos travaux, nous voulons nous intéresser uniquement au regroupement d'une collection de services en un certain nombre de clusters non connus a priori. Ceci nous conduit au choix de la méthode ascendante. De plus, comme nos objectifs portent fondamentalement non pas sur des aspects de performance mais plutôt sur des aspects de faisabilité (efficacité), nous avons donc décidé d'utiliser les méthodes les plus simples c'est à dire les méthodes précises, monothétiques, déterministes et non incrémentales.

Nous avons pour cela utilisé une approche de clustérisation ascendante hiérarchique (CAH) qui est une méthode de classification qui présente les avantages suivants :

- elle fonctionne à partir des similarités entre les services que l'on veut regrouper.
- elle permet de visualiser les résultats sous forme de dendrogramme, qui permet de mettre en évidence le regroupement progressif des services. On peut alors se faire une idée du nombre adéquat de classes dans lesquelles les services peuvent être regroupées.

La classification ascendante hiérarchique (CAH) est une méthode de classification itérative dont le principe est simple (figure VI.20) : On commence tout d'abord par calculer la similarité entre les N services. Puis on regroupe les deux services dont le

[40] Une méthode est ascendante si elle démarre avec autant de clusters que d'objets, puis procède à la concaténation de ces clusters, tandis qu'une méthode descendante démarre avec un seul cluster qui comporte tous les objets qu'elle va affiner.

[41] Une méthode est déterministe si avec les mêmes objets en entrée, elle exécute toujours la même suite d'opération et fournira toujours les mêmes résultats. A l'inverse, une méthode stochastique peut fournir des résultats différents en raison de l'exécution d'opérations aléatoires.

[42] Une méthode incrémentale est exécutée de façon continue et traite les données au fur et à mesure de leur arrivée, tandis qu'une méthode non-incrémentale s'exécute de façon ponctuelle en considérant que toutes les données sont disponibles.

[43] Une méthode est précise si elle associe à chaque objet un cluster unique, tandis qu'une méthode floue associe à chaque objet un degré d'appartenance à chaque cluster.

[44] Une méthode monothétique utilise séquentiellement les caractéristiques des objets lors de la clustérisation, tandis qu'une méthode polythétique utilise simultanément les caractéristiques des objets dans le processus de clustérisation.

regroupement minimise un critère d'agrégation donné, créant ainsi un cluster comprenant ces deux services. On calcule ensuite la similarité entre cette classe et les N-2 autres services en utilisant le critère d'agrégation. Puis on regroupe les deux services ou clusters de services dont le regroupement minimise le critère d'agrégation. On continue ainsi jusqu'à ce que tous les services soient regroupés. De façon plus formelle, le principe est illustré par l'algorithme suivant :

Algorithme CAH

Entrées: Une matrice de similarité de n objets à clustériser (ici des services): S_1, …, S_n.

Sorties: Un ensemble de clusters (dendrogramme)

Initialisation: Chaque service constitue un cluster.

Une mesure de distance *D* est calculée entre les clusters.

Tant que ((nombre de clusters) > 1)

- regrouper les deux clusters les plus proches au sens de la distance *D*,
- calculer les distances entre le nouveau cluster et les autres clusters.

Fin Tant que

Figure VI.20. Principe de la classification ascendante hiérarchique

Ces regroupements successifs produisent un arbre binaire de classification (dendrogramme), dont la racine correspond au cluster regroupant l'ensemble des services. Ce dendrogramme représente une hiérarchie de partitions. On peut alors choisir une partition en tronquant l'arbre à un niveau donné, le niveau dépendant soit des contraintes de l'utilisateur (l'utilisateur sait combien de clusters il veut obtenir), soit de critères plus objectifs, etc.

Pour calculer la similarité (ou dissimilarité) entre deux clusters de services A et B, différentes stratégies sont possibles. Les méthodes les plus courantes sont comme suit :

- **Lien simple (lien minimum)**

la dissimilarité entre A et B est la dissimilarité (distance) entre l'objet de A et l'objet de B les plus ressemblants. Plus formellement, la distance *D(h, h')* entre deux clusters h et h' est comme suit :

$$D(h,h') = \min\left[d(\mathbf{x},\mathbf{y})/\mathbf{x} \in h \; et \; \mathbf{y} \in h'\right]$$

L'agrégation par le lien simple a tendance à contracter l'espace des données et à écraser les niveaux des paliers du dendrogramme. Comme la dissimilarité entre deux éléments de A et de B suffit à relier A et B, ce critère peut conduire à relier des classes très allongées (effet de chaînage) alors qu'elles ne sont pas homogènes.

- **Lien complet**

La dissimilarité entre A et B est la plus grande dissimilarité entre un objet de A et un objet de B. Plus formellement cette mesure se calcule par la formule suivante :

$$D(h,h') = \max\left[d(\mathbf{x},\mathbf{y})/\mathbf{x} \in h \; et \; \mathbf{y} \in h'\right]$$

L'agrégation par le lien complet a tendance à dilater l'espace des données et produit des classes compactes.

- **Lien moyen**

La dissimilarité entre A et B est la moyenne des dissimilarités entre les objets de A et les objets de B. Plus formellement cette mesure se calcule par la formule suivante :

$$D(h,h') = \frac{\sum_{i=1}^{n_h}\sum_{j=1}^{n_{h'}} d(\mathbf{x}_i,\mathbf{x}_j)}{n_h \cdot n_{h'}}$$

L'agrégation selon le lien moyen est un bon compromis entre les critères précédents et respecte assez bien les propriétés de l'espace des données. Il existe deux principales variantes de ce critère :

- **Lien proportionnel**

  La dissimilarité moyenne entre les objets de A et de B est calculée comme une somme de dissimilarités pondérée de telle sorte qu'un poids égal soit attribué aux deux groupes. Comme le lien moyen, ce critère respecte assez bien les propriétés de l'espace des données.

- **Lien flexible**

  Ce critère fait intervenir un paramètre bêta variant dans l'intervalle [-1,+1[qui permet de générer une famille de critères d'agrégation. Pour bêta = 0 on retrouve le lien proportionnel. Quand bêta est proche de 1, on obtient un fort effet de chaînage, mais à mesure que bêta décroît et devient négatif, on obtient une dilatation de plus en plus forte.

- **Méthode de Ward**

Ce critère permet d'agréger deux groupes de sorte que l'augmentation de l'inertie intra-classe soit la plus petite possible, afin que les classes restent homogènes. Ce critère, proposé notamment par [Ward 1963], ne peut s'utiliser que dans le cas des distances quadratiques, c'est-à-dire ici, dans le cas de la distance euclidienne et de la distance du khi². De façon formelle, si n_h et m_h désignent respectivement le nombre d'objets dans le cluster h et la masse du cluster h, ce critère se calcule comme suit:

$$D(h,h') = \frac{n_h \cdot n_{h'}}{n_h + n_{h'}} \|\mathbf{m}_h - \mathbf{m}_{h'}\|^2$$

Dans le cadre de nos travaux, tous les critères d'agrégation ont été testés. Au travers des multiples tests, le critère de Ward nous paraît le mieux adapté pour la clustérisation de nos services car il minimise l'inertie intra-classe tout en maximisant l'inertie inter-classes. La notion d'inertie mesure le degré de dispersion des éléments autour du centre du cluster, il s'agit donc de la mesure inverse de l'homogénéité.

En se basant sur ces critères d'agrégation, l'algorithme de classification permet alors de restituer le résultat de la clustérisation sous forme de dendrogramme. Le dendrogramme permet de visualiser le regroupement progressif des services. Si une troncature a été demandée, un trait en pointillé marque le niveau auquel est effectuée la troncature. La troncature permet de limiter le nombre de clusters à retenir, ou le niveau auquel le dendrogramme doit s'achever. La troncature permet de trouver la coupe du dendrogramme qui correspond à la perte maximale d'inertie suite à un regroupement de clusters.

En plus de ce diagramme central, l'algorithme peut visualiser d'autres résultats, notamment:

- Statistiques des nœuds : qui permet d'illustrer dans un tableau les informations concernant les nœuds successifs du dendrogramme. Le premier nœud a pour indice le nombre de services augmenté de 1. Ainsi, il est aisé de repérer à quel moment un

service ou un cluster de services est regroupé avec un autre service ou cluster de services au niveau d'un nouveau nœud dans le dendrogramme.

- Diagramme des niveaux : un tableau où sont affichées les statistiques des nœuds du dendrogramme.
- Barycentres des clusters : un tableau où sont affichées les coordonnées des barycentres des clusters pour les différents descripteurs.
- Distances entre les barycentres des clusters : un tableau où sont affichées les distances euclidiennes entre les barycentres des clusters pour les différents descripteurs.
- Services centraux : un tableau où sont affichées pour chaque cluster les coordonnées du service le plus proche du barycentre du cluster.
- Distances entre les services centraux : un tableau où sont affichées les distances euclidiennes entre les services centraux des clusters pour les différents descripteurs.

VI.5.4. Exemple simplifié d'urbanisation

Afin d'illustrer comment notre technique d'urbanisation fonctionne, nous allons prendre un exemple simplifié portant sur des services fictifs. L'objectif étant seulement d'expliquer le déroulement de la technique. Un exemple plus concret sera présenté au chapitre IX lors de l'expérimentation que nous avons effectuée sur une étude de cas.

Soit la collection $\mathfrak{I}$ de services d'entreprise suivante :

Service S_i	Entrées Entrée(S_i)	Sorties Sortie(S_i)	Opérations Opération(S_i)	Domaine Domaine(S_i)
S_1	e_1, e_2	s_1, s_2	o_1, o_2	d_1
S_2	e_1, e_3	s_1, s_3	o_1, o_3	d_1
S_3	e_3, e_4	s_3, s_4,	o_3, o_4	d_1
S_4	e_4, e_5	s_4, s_5	o_4, o_5	d_2
S_5	e_5,e_6, e_7	s_5,s_6, s_7	o_5, o_6, o_7	d_2

Tableau VI.1. Exemple de collection de services d'entreprise

La méthode de calcul de similarité appliquée à cette collection de services donne pour la comparaison des entrées les mesures suivantes :

Entrées E_i / Terme w_i	E_1 $\{e_1, e_2\}$	E_2 $\{e_1, e_3\}$	E_3 $\{e_3, e_4\}$	E_4 $\{e_4, e_5\}$	E_5 $\{e_5, e_6, e_7\}$	n_j
e_1	1	1	0	0	0	2
e_2	1	0	0	0	0	1
e_3	0	1	1	0	0	2
e_4	0	0	1	1	0	2
e_5	0	0	0	1	1	2
e_6	0	0	0	0	1	1
e_7	0	0	0	0	1	1

Tableau VI.2. Calcul des coefficients n_{ij} et n_j

Entrées E_i / Terme w_i	E_1 $\{e_1, e_2\}$	E_2 $\{e_1, e_3\}$	E_3 $\{e_3, e_4\}$	E_4 $\{e_4, e_5\}$	E_5 $\{e_5, e_6, e_7\}$
e_1	0.4581	0.4581	0.0000	0.0000	0.0000
e_2	0.8047	0.0000	0.0000	0.0000	0.0000
e_3	0.0000	0.4581	0.4581	0.0000	0.0000
e_4	0.0000	0.0000	0.4581	0.4581	0.0000
e_5	0.0000	0.0000	0.0000	0.4581	0.4581
e_6	0.0000	0.0000	0.0000	0.0000	0.8047
e_7	0.0000	0.0000	0.0000	0.0000	0.5365

Tableau VI.3. Calcul des coefficients x_{ij}

x_k / x_l	x_1	x_2	x_3	x_4	x_5
x_1	0.8574	0.2099	0.0000	0.0000	0.0000
x_2	0.2099	0.4197	0.2099	0.0000	0.0000
x_3	0.0000	0.2099	0.4197	0.2099	0.0000
x_4	0.0000	0.0000	0.2099	0.4197	0.2099
x_5	0.0000	0.0000	0.0000	0.2099	1.1452

Tableau VI.4. Calcul de $x_k^T x_l$

x_l / x_k	x_1	x_2	x_3	x_4	x_5
x_1	1.0000	0.3499	0.0000	0.0000	0.0000
x_2	0.3499	1.0000	0.5001	0.0000	0.0000
x_3	0.0000	0.5001	1.0000	0.5001	0.0000
x_4	0.0000	0.0000	0.5001	1.0000	0.3028
x_5	0.0000	0.0000	0.0000	0.3028	1.0000

Tableau VI.5. Calcul de $\cos(x_k, x_l) = \frac{x_k^T x_l}{\sqrt{x_k^T x_k}\sqrt{x_l^T x_l}}$

Tous les calculs précédents sont réalisés en vue d'évaluer l'expression $\Psi(E_k, E_l) = \cos(x_k, x_l)$. Nous avons effectué la même chose en ce qui concerne la similarité des sorties, des connexions, des opérations et du domaine et nous avons consigné le résultat final dans la matrice suivante[45] :

[45] Pour des besoins de simplification, nous avons choisi de prendre les poids des paramètres comme suit : $\omega_e = \omega_s = \omega_o = \omega_d = 1/4$ *et* $\omega_c = 0$.

S_k \ S_l	S_1	S_2	S_3	S_4	S_5
S_1	1.0000	0.5124	0.2500	0.0000	0.0000
S_2	0.5124	1.0000	0.6251	0.0000	0.0000
S_3	0.2500	0.6251	1.0000	0.3751	0.0000
S_4	0.0000	0.0000	0.3751	1.0000	0.4771
S_5	0.0000	0.0000	0.0000	0.4771	1.0000

Tableau VI.6. Matrice de similarité de services

Une fois que cette matrice obtenue, nous allons lui appliquer la méthode de classification hiérarchique ascendante que nous avons retenue dans la section VI.5.3, et nous obtenons enfin le dendrogramme suivant (figure VI.21) qui va permettre de servir de base pour la clustérisation de nos services.

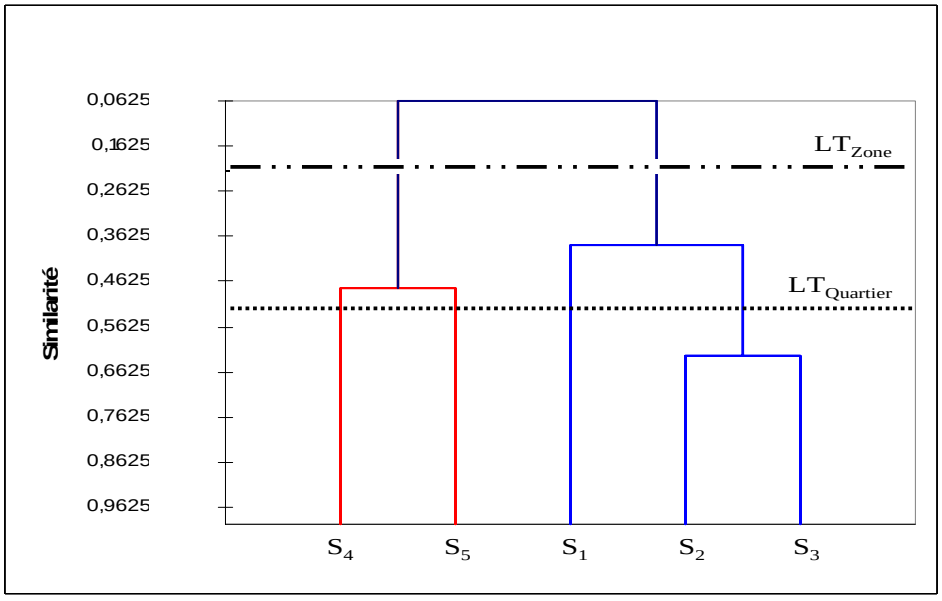

Figure VI.21. Dendrogramme[46]

Une autre représentation équivalente au dendrogramme où plutôt une interprétation du dendrogramme est la cartographie des clusters. Cette dernière est obtenue en reportant le résultat du dendrogramme précédent sur le graphe de similarité des services (figure VI.22). Nous avons ajouté à la cartographie des clusters notre interprétation qui consiste à déterminer les clusters de services (les quartiers et les zones) qui nous paraissent les plus pertinents. La détermination de ces clusters se fait par le positionnement de deux lignes de troncature. La première ligne (LT_{Zone}) permet de délimiter les quartiers tandis

[46] Signalons au passage que les calculs présentés dans ce chapitre sont réalisés par l'outil XLSTAT 2006.

que la seconde ligne ($LT_{Quartier}$) permet de délimiter les zones. Pour le moment, la détermination de ces lignes est laissée à l'apprécation de l'utilisateur humain.

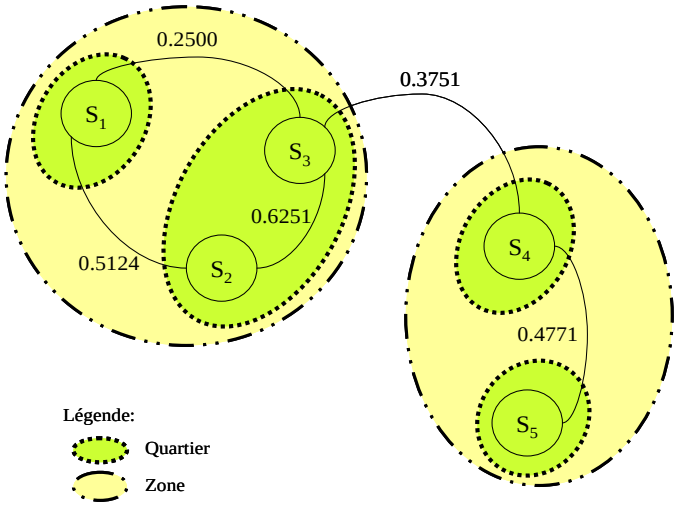

Figure VI.22. Exemple de cartographie des services d'entreprise

Le résultat de la cartographie précédente est un élément fondamental de l'urbanisation des services. Il permet de définir des blocs homogènes, cohérents et faiblement couplés de services au sein de l'entreprise. Ce découpage sera ensuite exploité dans les chapitres suivants afin de proposer une architecture sémantique appropriée pour les entreprises industrielles.

VI.6. Conclusion

Dans ce chapitre, nous avons présenté notre approche de construction de l'architecture orientée services d'entreprise. Cette architecture, qui répond à la problématique P_{Syn} (cf. § V.2.1), constitue l'une des réponses les plus efficaces pour favoriser la réutilisation et l'intégration des applications industrielles. Elle permet notamment de rationaliser et d'industrialiser les procédés de production et d'intégration d'applications, tout en permettant une meilleure évolutivité et réactivité du système d'information.

L'approche que nous avons proposée s'inscrit dans le cadre du projet global d'intégration d'entreprise avec une réflexion d'ensemble et une démarche itérative et incrémentale. Notre approche se base sur la dichotomie SOA IT et SOA métier qui permettent respectivement de répondre à deux problématiques complémentaires qui sont celle de la maîtrise d'oeuvre et celle de la maîtrise d'ouvrage. Notre démarche est composée de quatre étapes majeures qui sont (cf. § VI.4) : l'exposition en services IT, la définition de services métier, l'urbanisation et la publication des services métier.

L'urbanisation des services (cf. § VI.4.5) constitue un point majeur de la construction de l'architecture de services. Elle permet d'étendre le concept d'urbanisation (cf. § II.5.1) du système d'information aux architectures SOA (cf. § III.8) en proposant une urbanisation orientée services du système d'information d'entreprise (cf. § VI.5). Elle permet notamment de proposer une cartographie des services métier en définissant des regroupements de services similaires, ou encore des clusters, de manière à faciliter et à favoriser l'intégration intra- et inter-clusters. Ces clusters seront ensuite exploités pour définir la sémantique des applications industrielles et qui fait l'objet du chapitre suivant.

Chapitre VII

CONSTRUCTION DE L'ARCHITECTURE SEMANTIQUE D'ENTREPRISE

VII.1. Introduction

Le chapitre précédent a permis de répondre à la problématique P_{Syn} de construction de l'architecture de services d'entreprise. Il a notamment présenté une modélisation de services en se basant sur la dichotomie SOA IT et SOA métier. Il a également présenté une méthodologie de construction de l'architecture de services tout en mettant l'accent sur la notion d'urbanisation des services, notion qui est par ailleurs fondamentale dans le sens où elle a permis de jeter les bases de structuration des services en îlots sémantiques. A présent, nous allons nous intéresser dans ce chapitre à la problématique P_{Sem} (figure VII.1) qui, rappelons-le, a pour but de répondre à la question de 'sémantification'[47] ou d'enrichissement sémantique des services d'entreprise.

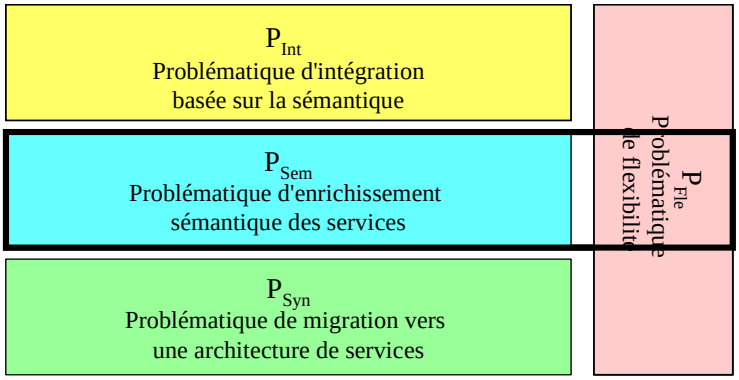

Figure VII.1. La problématique P_{Sem}

[47] Nous empruntons les anglicismes 'sémantifier' et 'sémantification' pour désigner respectivement l'action et le processus d'enrichissement sémantique des services d'entreprise.

L'objectif de ce chapitre consiste à répondre à certaines questions et limites énoncées dans les chapitres précédents, concernant la description sémantique des services dans le domaine industriel. Rappelons que la description sémantique des services se base sur la notion de modèle sémantique d'entreprise, ou la notion d'ontologie, permettant la spécification formelle des services au sein de l'entreprise. Cette ontologie sert d'élément pivot entre la couche de services et la couche d'intégration (figure VII.2[48]).

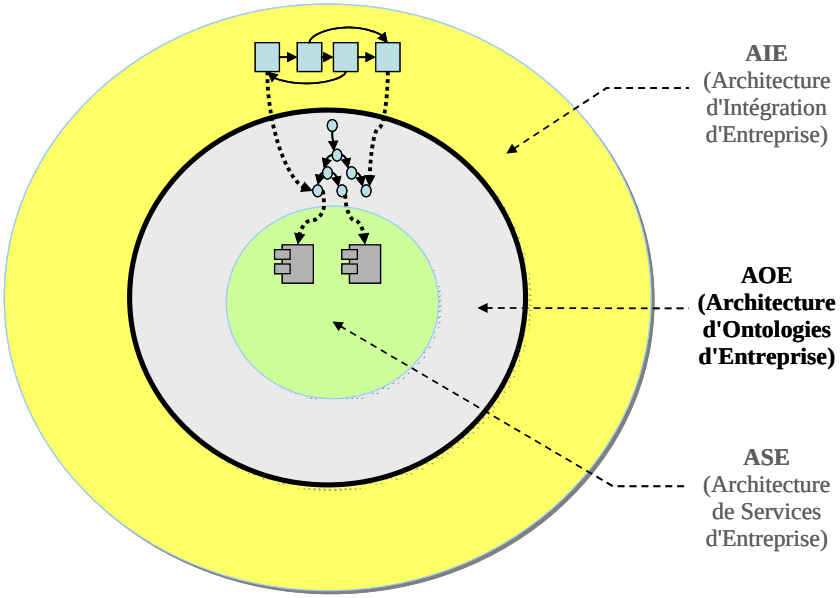

Figure VII.2. Positionnement de la couche sémantique (AOE)

L'objectif de ce chapitre consiste à décrire de façon formelle ce modèle sémantique qui contient l'ensemble des éléments nécessaires à la description et à l'intégration des services d'entreprise. Il se base en particulier sur l'enrichissement de l'ontologie générique de services OWL-S qui constitue à notre sens l'initiative la plus mature pour la mise en œuvre de services sémantiques. La section VII.2 exposera les principes majeurs permettant de guider la construction de l'architecture sémantique d'entreprise. La section VII.3 présentera de façon générale le modèle sémantique qui sera ensuite détaillé en section VII.4. La section VII.5 présentera notre principe d'urbanisation de la sémantique et enfin la section VII.6 présentera notre démarche méthodologique permettant la construction de l'architecture sémantique de l'entreprise.

VII.2. Principes majeurs

Comme nous l'avons signalé auparavant dans le chapitre IV, le moyen le plus approprié pour mettre en oeuvre une intégration flexible des applications industrielles doit reposer sur l'utilisation d'un modèle sémantique en tant que fondement de base de cette intégration. Ce modèle a pour but d'intégrer les applications en permettant un découplage à la fois entre le niveau métier et le niveau technique, et un découplage entre les différents types de services d'entreprise. Ce découplage étant fondamental afin de répondre au critère de flexibilité du système d'information, et donc en matière d'agilité de l'entreprise.

[48] Cette figure découle directement de la figure V.8.

De même que pour la construction de l'architecture de services, la construction de l'architecture sémantique ou ontologique (AOE) à l'intérieur d'une entreprise doit répondre à plusieurs préoccupations qui portent sur les aspects de modélisation et de démarche de construction de la couche ontologique. Notre approche sémantique repose principalement sur les deux points majeurs qui sont :

- Un *cycle d'abstraction* permettant de définir une modélisation des ontologies nécessaires à mettre en œuvre qui met en exergue une typologie d'ontologies de différents niveaux d'abstraction;
- Un *cycle de vie* des ontologies qui permet de proposer une démarche méthodologique en matière de développement de ces ontologies permettant ainsi de guider l'utilisateur dans le processus d'enrichissement sémantique des services définis dans le chapitre précédent.

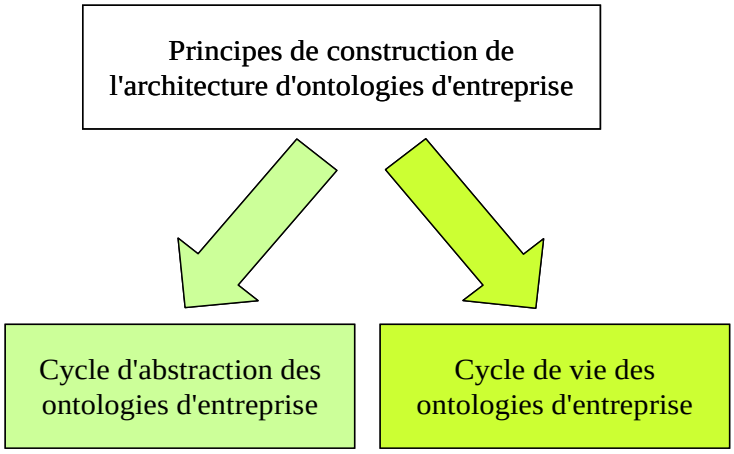

Figure VII.3. Principes majeurs de construction de l'architecture d'ontologies

Le cycle d'abstraction se base sur la définition de deux types d'ontologies qui sont les ontologies de service et des ontologies fondamentales. Ces deux niveaux d'ontologies sont importants pour pouvoir distinguer entre les ontologies d'annotation qui sont plus génériques et les ontologies fondamentales qui sont plus spécifiques à l'entreprise. Cette dichotomie permet de séparer ainsi les préoccupations d'ordre technique et des préoccupations d'ordre métier. Cette double préoccupation donne naissance à un double aspect pour notre architecture sémantique : un aspect purement technique (ontologies d'annotation de services ou ontologie de services) et un aspect relativement plus conceptuel et plus significatif pour l'entreprise (ontologies fondamentales).

D'autres aspects du cycle d'abstraction sont pris en compte et peuvent se résumer ainsi :

- Une formalisation des ontologies : En effet, notre modèle sémantique doit être suffisamment formalisé afin de permettre un traitement automatique par le biais de moteurs d'inférence. Bien qu'il soit une exigence d'ordre technique, il n'en demeure pas moins qu'il constitue un principe fondamental car il permet de définir l'usage qui sera fait de ce modèle.
- Une couverture appropriée du modèle : Le modèle sémantique à définir doit couvrir les différents aspects liés au projet d'intégration de systèmes d'information industriels. Ce principe implique la prise en compte d'une multitude de points de vue :

- le point de vue de l'architecte/urbaniste qui sera amené à utiliser le modèle sémantique afin de pouvoir cataloguer et gérer les différents composants du système d'information selon une approche urbanistique. Ceci nécessite en particulier de connaître les différents composants du système d'information existant (composants techniques, applicatifs, fonctionnels et métier) ainsi que ceux qui sont ou qui seront exposés en services d'entreprise;
- le point de vue de l'intégrateur qui sera amené à utiliser le modèle sémantique pour des fins d'interconnexion de composants du système d'information. Il s'agit en particulier de connaître la composition des services et les liaisons qui peuvent exister entre eux;
- et aussi le point de vue du développeur qui sera amené à utiliser le modèle pour la maintenance et/ou le développement des applications. Il s'agit en particulier de pouvoir retrouver la signification des multiples éléments des applications existantes pour pouvoir les maintenir. Il peut également s'agir de guider l'utilisateur dans le processus de développement des applications en lui offrant un moyen lui permettant de découvrir des modules existants à réutiliser.

Le cycle de vie des ontologies, quant à lui, permet de proposer une méthodologie spécifique qui est une extension des méthodologies actuelles en matière de construction d'ontologies d'entreprise. Il est fondamental de préciser que l'une des contributions majeures de notre cycle de vie des ontologies est la prise en compte de l'aspect urbanisation sémantique. Cette urbanisation, qui constitue le prolongement de l'urbanisation des services d'entreprise introduit au chapitre précédent, a pour objet de structurer l'architecture des ontologies en îlots de façon à simplifier la réutilisation et l'intégration des ontologies d'entreprise. Une autre contribution importante est la complétude. Cette dernière tient au fait que notre approche a pour ambition de proposer une démarche relativement exhaustive permettant de couvrir l'ensemble du cycle de vie de sémantification des services d'entreprise.

Il est possible de récapituler les grands axes de ce chapitre à travers la méta-ontologie générique de sémantification des services d'entreprise, illustrée en figure VII.4.

Cette méta-ontologie générique stipule que la description des services d'entreprise est une spécialisation de l'ontologie OWL-S et fait intervenir un certain nombre d'ontologies supplémentaires appelées ontologies fondamentales qui permettent de décrire certaines sémantiques du système d'information d'entreprise. Ces dernières ontologies sont structurées en plusieurs niveaux (local, domaine et global) définissant ainsi des clusters sémantiques qui correspondent aux clusters de services définis au chapitre précédent. Les ontologies fondamentales sont reliées entre elles par des mappings sémantiques. De plus des mappings syntaxiques (ou plus exactement des mappings syntaxe-sémantique) sont définis et permettent de lier les ontologies locales aux descriptions syntaxiques des services d'entreprise.

Les différents éléments introduits dans cette méta-ontologie générique sont essentiels afin de construire correctement une AOE (architecture d'ontologies d'entreprise) flexible permettant de décrire la sémantique des services d'entreprise. Ils constituent les points majeurs qui seront traités tout au long de ce chapitre.

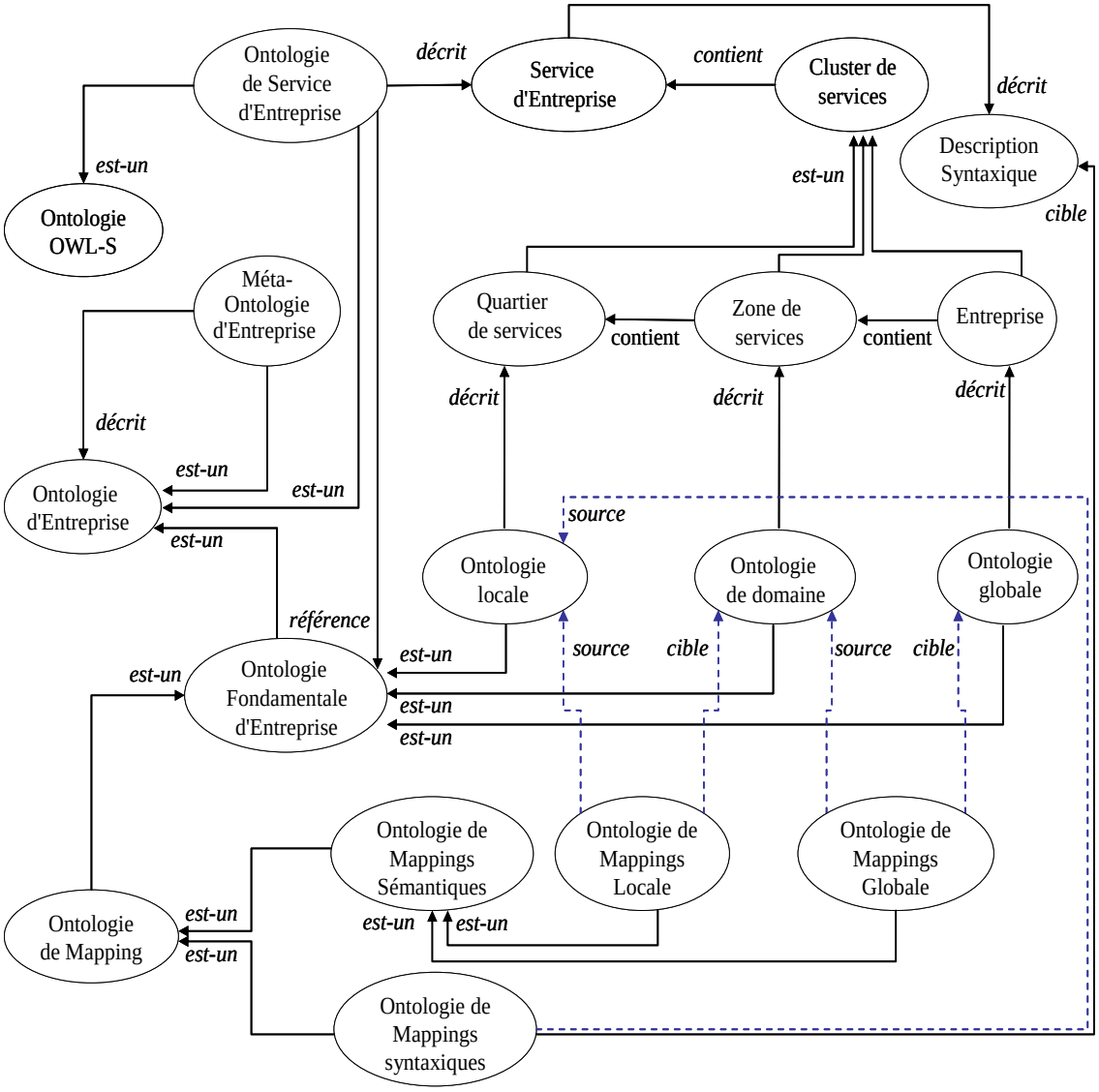

Figure VII.4. Méta-ontologie générique de sémantification des services d'entreprise[49]

VII.3. Description générale de l'AOE

Le modèle que nous proposons s'intitule *AOE* (Architecture d'Ontologies d'Entreprise ou modèle sémantique d'entreprise**)** et repose principalement sur un ensemble de concepts qui permettent de décrire sémantiquement des systèmes d'information d'entreprise, notamment les services d'entreprise issus de l'architecture présentée dans le chapitre précédent. L'élément central de l'AOE est la notion d'ontologie. Cette dernière est principalement considérée comme un ensemble de concepts permettant d'expliciter formellement un aspect du système d'information industriel. Le choix des concepts, c'est à dire des ontologies, ainsi que le choix de la plupart des règles (contraintes) qui accompagnent ces ontologies résulte d'une double approche empirique et théorique. Nous allons dans cette section fournir une vision globale de l'AOE.

[49] Signalons que nous nous basons tout au long de ce chapitre sur un formalisme compatible UML pour représenter nos différentes ontologies. Aussi, le sens des symboles graphiques utilisés dans ces modèles est compatible (à quelques graphiques près) avec celui préconisé dans UML 2.0. Les ovales désignent des classes, les flèches typées désignent les relations (au sens large) entre classes.

VII.3.1. Structure du modèle sémantique

Le modèle sémantique d'entreprise est structuré en un certain nombre d'ontologies (figure VII.5) qui servent à capturer la sémantique nécessaire pour l'intégration des systèmes d'information industriels :

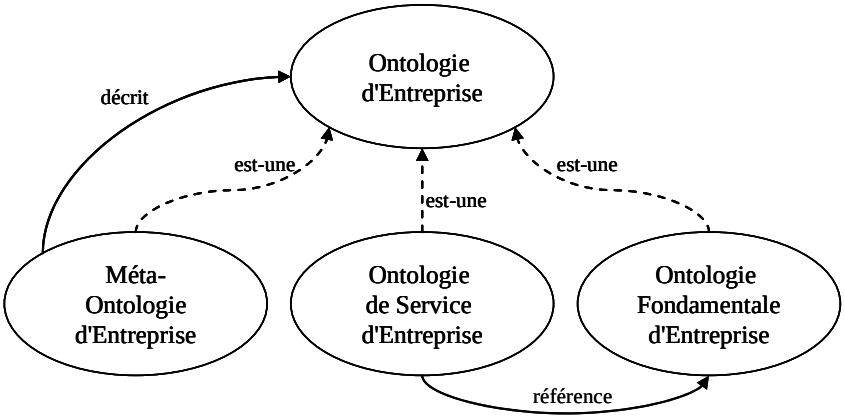

Figure VII.5. Description générale du modèle sémantique d'entreprise

- l'ontologie de service d'entreprise : elle permet de décrire sémantiquement les services fondamentaux;
- l'ontologie fondamentale d'entreprise : elle est référencée par la première et permet de capturer la sémantique spécifique associée aux applications industrielles de l'entreprise;
- et la méta-ontologie d'entreprise : elle est utilisée pour décrire les différentes ontologies utilisées.

Ces différentes ontologies, appelées ontologies d'entreprise, sont introduites dans les paragraphes qui suivent.

VII.3.2. Ontologie d'entreprise

Nous appelons ontologie d'entreprise toute ontologie ou modèle sémantique permettant de décrire la sémantique d'un aspect donné du système d'information d'entreprise. Les aspects qui peuvent être décrits peuvent être les données de l'entreprise, les applications, les fonctions du système d'information industriel, les processus métier ou encore les services issus de la structuration du système d'information en architecture orientée services. De plus, chacune de ces ontologies peut aussi être décrite en utilisant des méta-ontologies. Ces dernières sont importantes du fait qu'elles permettent une meilleure gestion de la sémantique de l'entreprise.

Il est possible de mieux apprécier la complexité du modèle sémantique en termes de nombre d'ontologies à travers le tableau suivant (tableau VII.1), qui synthétise les principales ontologie de notre approche permettant ainsi d'offrir une vision plus globale du modèle sémantique proposé.

<table>
<tr><th colspan="2">Ontologie d'entreprise</th><th colspan="2">Vue d'entreprise</th></tr>
<tr><td colspan="2">Méta-ontologie d'entreprise</td><td colspan="2">Vue sémantique</td></tr>
<tr><td colspan="2">Ontologie de service d'entreprise</td><td colspan="2">Vue services</td></tr>
<tr><td rowspan="5">Ontologie fondamentale d'entreprise</td><td>Ontologie métier d'entreprise</td><td>Vue métier</td><td rowspan="5">Vue fondamentale d'entreprise

(composants de base)</td></tr>
<tr><td>Ontologie fonctionnelle d'entreprise</td><td>Vue fonctionnelle</td></tr>
<tr><td>Ontologie logicielle d'entreprise</td><td>Vue logicielle</td></tr>
<tr><td>Ontologie de données d'entreprise</td><td>Vue données</td></tr>
<tr><td>Ontologie technique d'entreprise</td><td>Vue technique</td></tr>
</table>

Tableau VII.1. Correspondance ontologies et vues d'entreprise

Toutes ces ontologies permettent de décrire certains aspects des Systèmes d'Information d'entreprise liés à la problématique d'intégration. Elles permettent de prendre en compte l'ensemble des concepts et liaisons entre les concepts nécessaires à la représentation des systèmes d'information industriels.

La reformulation de la figure VII.5, présentant les différents niveaux de perception des concepts, en tenant compte de l'introduction des différents niveaux de modélisation devient alors la figure VII.6 qui présente cette vision hiérarchisée.

Au niveau 0, la modélisation étant principalement appréhendée avec la notion de concept d'ontologie qui permet de décrire la sémantique des vues. Au niveau 1, niveau de la méta-modélisation, nous retrouvons l'ensemble des méta-ontologies qui permettent de décrire l'aspect sémantique de l'entreprise. Le niveau 2 correspondant au niveau de la modélisation, et permet de définir les ontologies d'entreprise qui offrent la possibilité de décrire formellement le système d'information industriel (ontologie de service et ontologies fondamentales). Finalement, au dernier niveau (niveau de l'instanciation), nous trouvons les différentes instances des ontologies d'entreprise du niveau précédent.

Nous allons, dans ce qui suit, donner une description détaillée des différentes ontologies que nous venons d'évoquer et qui sont présentes au niveau 1 et 2 d'utilisation des ontologies.

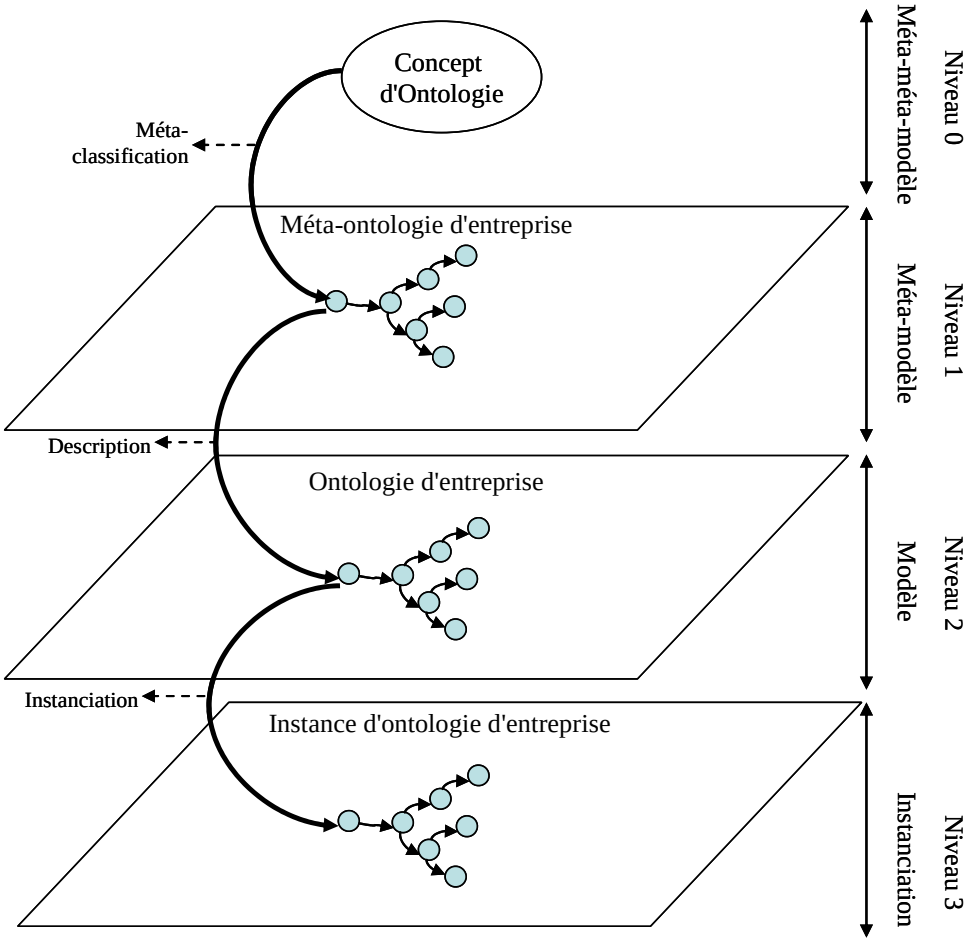

Figure VII.6. Les différents niveaux d'utilisation des ontologies

VII.4. Description détaillée de l'AOE

La section précédente a permis de présenter une vision globale de l'AOE. A présent, cette section se propose de présenter d'une manière détaillée les différentes ontologies qui composent l'AOE et qui rappelons-le sont : l'ontologie de service d'entreprise, l'ontologie fondamentale et la méta-ontologie d'entreprise. Les deux premières ontologies appartiennent au niveau 2, à savoir le niveau de modélisation, alors que la dernière ontologie appartient au niveau 1, à savoir le niveau de méta-modélisation.

VII.4.1. Ontologie de service d'entreprise

L'ontologie de services d'entreprise, dont certains aspects ont été déjà exposés dans [Izza et al. 2005-a][Izza et al. 2005-b][Izza et al. 2006-a][Izza et al. 2006-c], permet de décrire sémantiquement les services d'entreprise définis au chapitre VI. Cette ontologie est définie comme une extension de l'ontologie standard OWL-S [OWL-SC 2004] utilisée à l'origine pour annoter les services sur le web. Le principe de l'ontologie de service est illustré en figure VII.7 où l'on peut remarquer qu'elle permet d'enrichir la sémantique générique de OWL-S par une sémantique spécifique nécessaire pour décrire les spécificités des services d'entreprise telles que évoquées dans le chapitre VI.

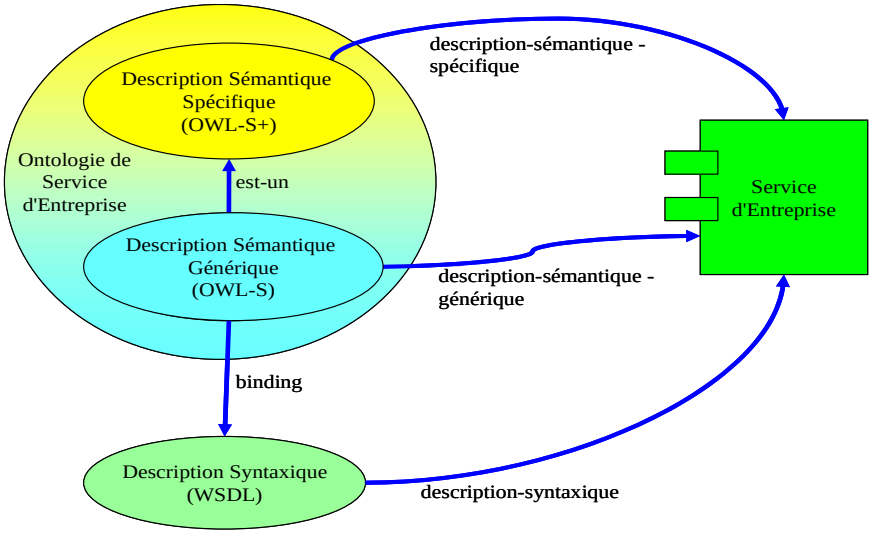

Figure VII.7. Principe d'utilisation de l'ontologie de service d'entreprise

La figure VII.8 qui est un raffinement de la figure précédente permet de présenter les principaux concepts de l'ontologie de service d'entreprise. Comme on peut le constater, l'approche de description en couches hiérarchisées a été retenue et l'on peut associer

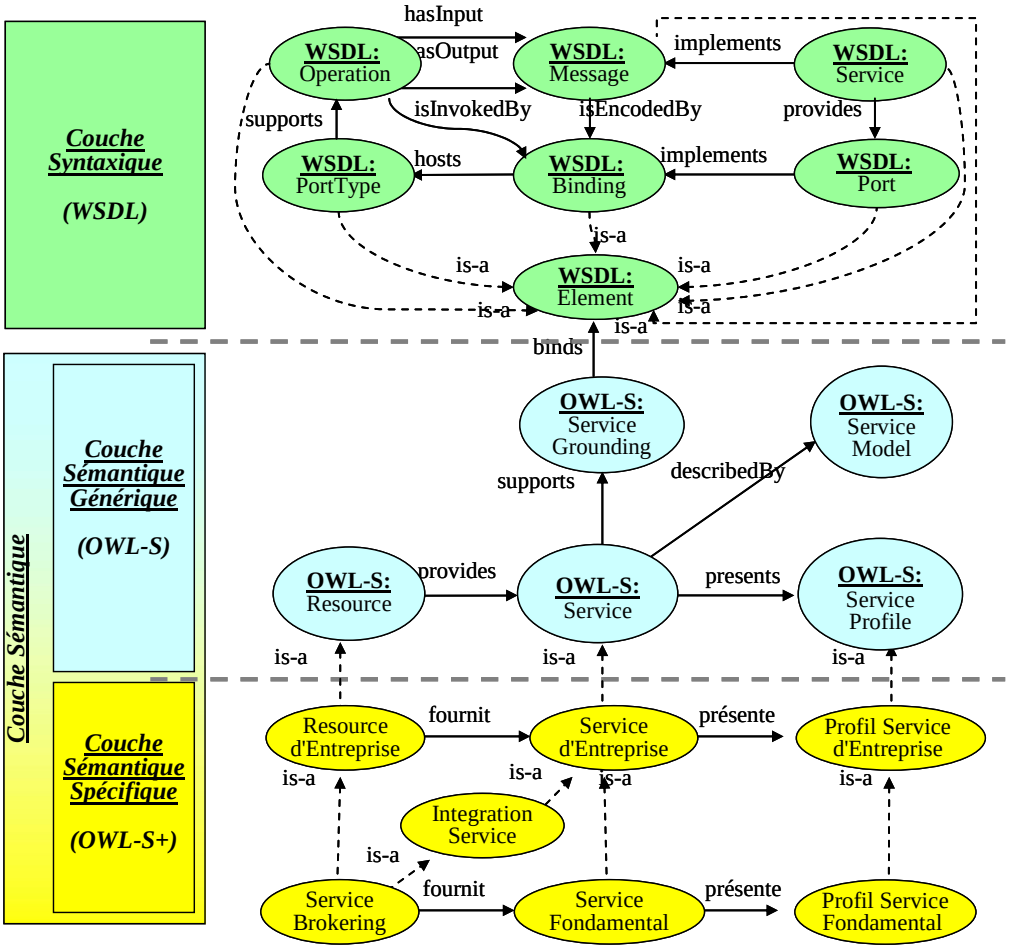

Figure VII.8. Ontologie de description de service d'entreprise [Izza et al. 2006-a]

- à la description syntaxique des services, l'élément "*WSDL: Description Syntaxique*",
- à la description sémantique des caractéristiques génériques des services, les concepts OWL-S que nous avons réutilisés, notamment "*OWL-S: Service*", "*OWL-S: Service Profile*", "*OWL-S: Service Grounding*", "*OWL-S: Resource*",
- et à la description spécifique des services d'entreprise, des concepts spécifiques (OWL-S+) qui sont notamment le concept "*Profil de Service d'Entreprise*" et le concept "*Profil de Service Fondamental*".

Il est important de préciser que les deux ontologies associées aux deux derniers niveaux sont connectées entre elles du fait que OWL-S+ constitue une extension de OWL-S. De plus, OWL-S est connectée à la couche syntaxique (WSDL: Description Syntaxique de Service) par l'intermédiaire du lien "binds" liant le grounding (l'ensemble des caractéristiques techniques permettant d'invoquer le service) de service aux éléments WSDL du service d'entreprise.

VII.4.1.1. Ontologie de description générique de services (OWL-S)

OWL-S [OWLS-C 2004], anciennement DAML-S, est une ontologie OWL qui permet de décrire les services Web. Précisément, cette ontologie permet de décrire les propriétés d'un Service Web ainsi que son ou ses opérations rendues disponibles. Les concepts principaux de l'ontologie OWL-S sont illustrés par la figure VII.9 qui ne reprend de la figure VII.8 que les concepts associés à la sémantique générique des services. Au coeur de cette ontologie, on trouve le concept *OWL-S: Service*. Ce dernier reprend les propriétés générales d'un Service. Il présente un profil (*OWL-S: Service Profile*), est décrit par un modèle (*OWL-S: Service Model*) et supporte un grounding (*OWL-S: Service Grounding*). Le profil (*OWL-S: Service Profile*) explique ce que fait le service en termes de fonctionnalités et de la nature des messages en entrée et en sortie. Le modèle (*OWL-S: Service Model*) définit le fonctionnement du Service, le grounding (*OWLS: Service Grounding*) fournit les informations nécessaires à l'invocation du service et le concept ressource (*OWL-S: Resource*) donne des informations relatives aux ressources utilisées par le Service. Ces différents concepts sont exposés dans ce qui suit.

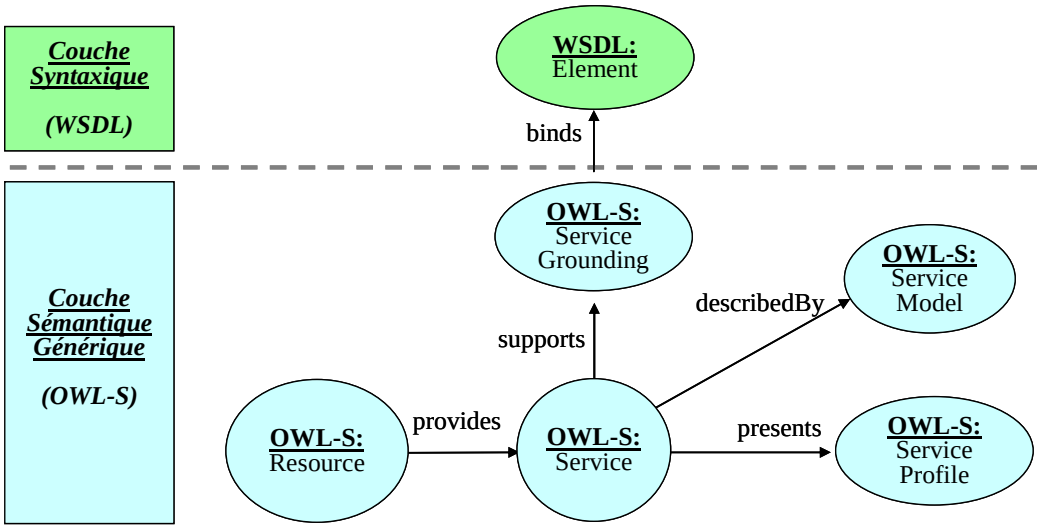

Figure VII.9.Ontologie OWL-S

VII.4.1.1.1. Le profil de service (OWL-S: Service Profile)

Le profil de service (*OWL-S: Service Profile*) décrit le service en fonction de ce qu'il fait afin de permettre à un client de voir dans quelle mesure le service proposé peut lui convenir. Les caractéristiques affichées dans le profil peuvent être décomposée en trois grands aspects (figure VII.10) :

- des caractéristiques non fonctionnelles qui permettent la description du service et de son fournisseur à travers notamment les propriétés de nom de service, de texte descriptif ou encore les propriétés concernant le fournisseur telles que les informations de contact, etc.
- des caractéristiques fonctionnelles qui permettent la description du comportement du service en termes de transformation d'information à l'aide notamment des entrées du service (inputs) des sorties du service (outputs), des pré-conditions nécessaires au bon déroulement du service et des d'effets du service.
- des caractéristiques additionnelles qui permettent la classification du service et la description de sa qualité de service (comme le temps de réponse, le coût du service, etc.).

Généralement le profil de service sert de support à la découverte de services et à leur sélection. Cette activité qui est très souvent appelée "matching", consiste à partir d'un profil hypothétique du service désiré par un client, à fournir le service du fournisseur qui propose un profil qui correspond le plus à la requête du client. Elle suppose l'intervention d'une ressource logicielle intermédiaire de type annuaire de services qui implémente cet algorithme de "matching".

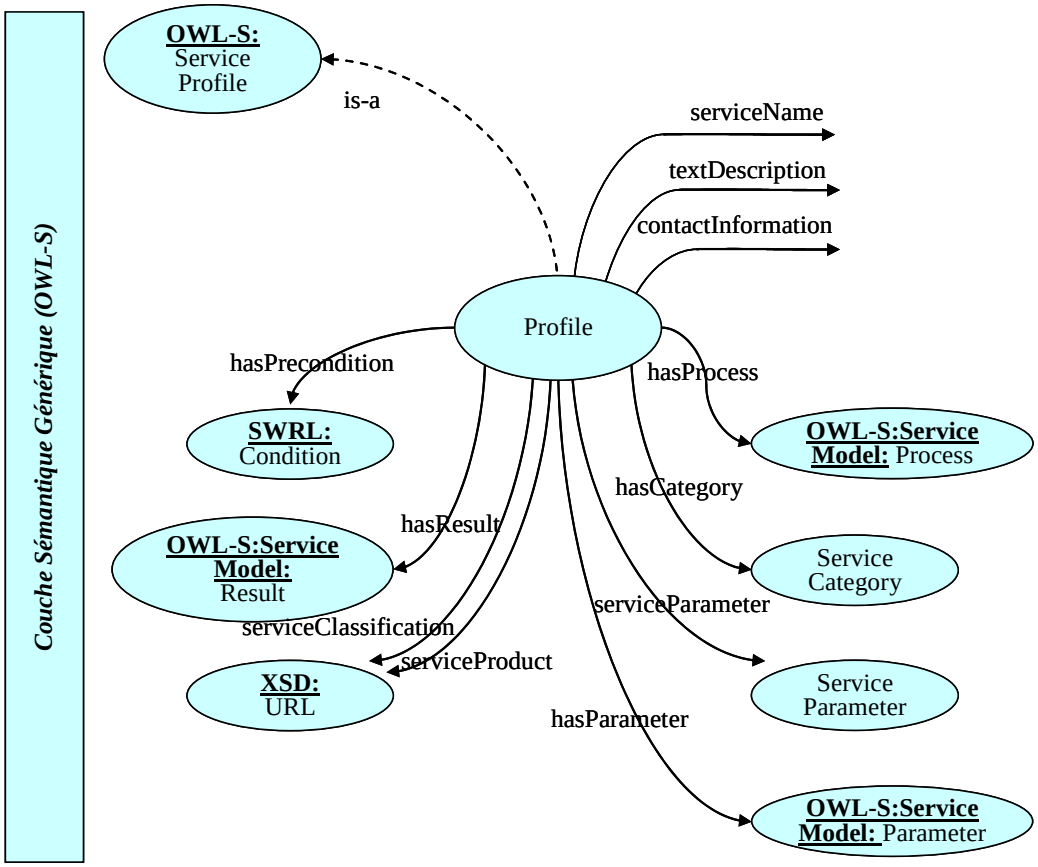

Figure VII.10. Ontologie de profil de service

Vis à vis de nos travaux, cette ontologie est pertinente mais elle demeure insuffisante du

fait qu'elle ne prend pas en compte un certain nombre de caractéristiques des services d'entreprise, notamment toutes les relations qui lient les services au back-office. A cet effet, nous proposerons dans la section VII.4.1.2 un enrichissement de cette ontologie de profil.

VII.4.1.1.2. Le modèle de service (OWL-S: Service Model)

Le modèle de service (*OWL-S: Service Model*) sert à expliquer comment le service fonctionne. A cet effet, un service est modélisé comme un processus grâce au concept de modèle de processus (*OWL-S: Process Model*). Ce dernier permet de décrire le processus du service qui est décrit par l'ontologie de processus (figure VII.11).

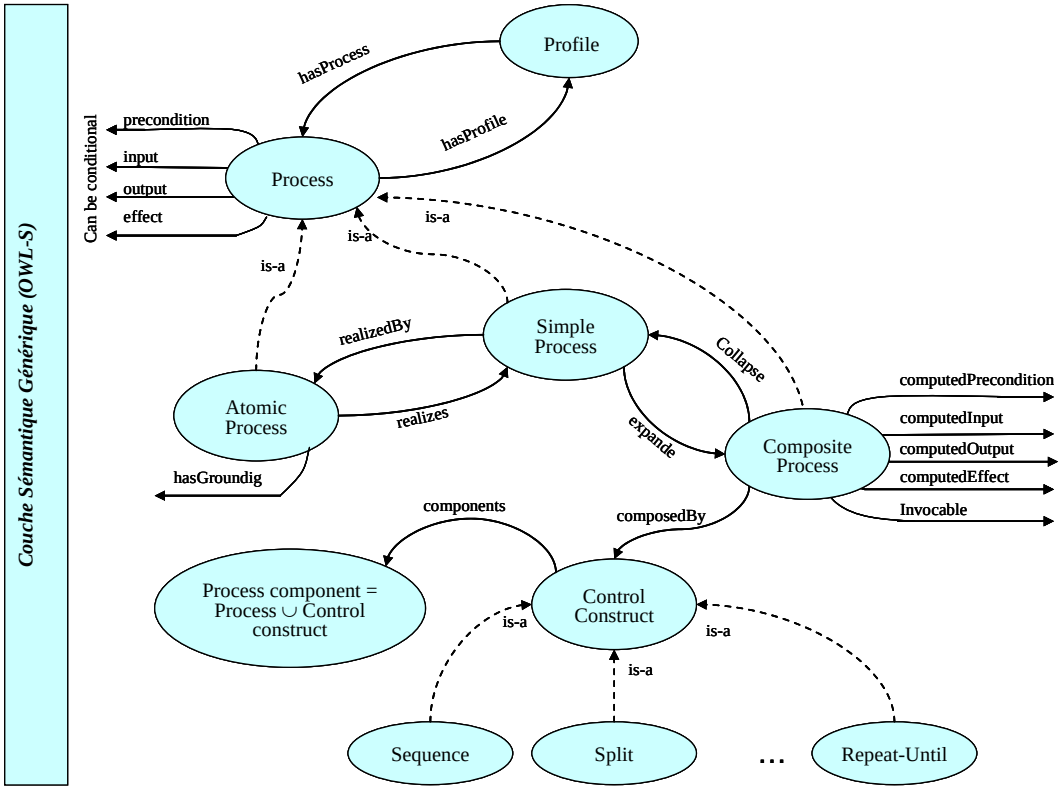

Figure VII.11. Ontologie de modèle de service

L'ontologie de processus de service permet de décrire le processus en termes d'entrées ("inputs"), de sorties ("outputs"), de pré-conditions ("preconditions") et d'effets sur le "Monde" ("effects"). Cette ontologie définit trois sous-types de processus :

- le processus atomique qui peut directement être invoqué par le client. Ce processus est élémentaire dans le sens où il n'est pas décomposable en sous-processus. Le client voit l'exécution du processus atomique comme une opération primitive, c'est-à-dire que son exécution se déroule en une seule étape.
- le processus simple qui est un processus qui ne peut pas être directement invoqué par le client mais qui est défini afin de permettre un certain niveau d'abstraction. Ce processus est également perçu par le client comme étant constitué d'une étape unique.
- le processus composite qui un processus composé d'autres processus selon certaines séquences d'enchaînement telles que la séquence conditionnelle (IF-THEN-ELSE), la répartition (SPLIT), les boucles (REPEAT-UNTIL), etc.

Le "ServiceModel" permet de réaliser plusieurs tâches importantes sur les services. En effet, après la découverte d'un service grâce à son profil (*OWL-S: Service Profile*), le modèle (*OWL-S: ServiceModel*) permet d'effectuer une analyse plus détaillée par un agent de service afin de déterminer si le processus répond véritablement aux attentes de cet agent. De plus, le modèle sert de support à la composition de services complexes. Il permet aussi de coordonner les différentes activités des services. Enfin, il offre la possibilité de surveiller l'état d'avancement d'un service.

Dans le cadre des travaux que nous menons, cette ontologie est considérée comme un élément pertinent de la description des services d'entreprise. Sa structure trop technique calquée sur WSDL la rend toutefois moins conceptuelle. A ce titre, nous décidons de l'enrichir en établissant des liens entre le concept "processus atomique" et certains concepts de nos ontologies fondamentales (en particulier l'ontologie fonctionnelle). Cette liaison permet d'améliorer la sémantique associée aux opérations de services. Comme la contribution apportée à cette ontologie est relativement négligeable, nous n'allons pas la détailler dans ce chapitre, mais nous reviendrons au chapitre VIII afin d'expliquer comment cette liaison est exploitée pour améliorer la recherche de services.

VII.4.1.1.3. Le grounding de service (OWL-S: Service Grounding)

Le grounding (*OWL-S: Service Grounding*) établit une correspondance entre une spécification abstraite définie en OWL-S par les "IOPE's" ("Inputs Outputs Preconditions Effects") pour un modèle et une spécification concrète. Cette mise en correspondance permet d'accéder au service. Pour cela, plusieurs aspects doivent être présents dans une description d'un "grounding", notamment le choix du protocole à utiliser pour accéder au service, le format de messages, la façon de les sérialiser, les mécanismes de transport à utiliser et quel mode d'adressage à employer. OWL-S se base généralement à cet effet sur WSDL qui fournit une spécification de plus bas niveau (syntaxique). La correspondance entre OWL-S et WSDL est nécessaire pour pouvoir invoquer les services OWL-S. De façon générale le principe de correspondance OWL-S /WSDL se base sur la définition de liens entre certains concepts de l'ontologie OWL-S et certains éléments de WSDL. Cette correspondance OWL-S/WSDL est réalisée grâce au concept *WSDL Grounding* de l'ontologie de grounding de service (figure VII.12) qui permet ainsi d'associer quand cela est possible d'une part un processus atomique de OWL-S à une opération de WSDL par l'intermédiaire de la propriété *wsdlOperation*, et d'autre part les entrées et les sorties de OWL-S en entrées et sorties de WSDL par l'intermédiaire des propriétés *wsdlInputMessage, wsdlInputMessagePart, wsdlOutputMessage,* et *wsdlOutputMessagePart*.

Bien que très complexe à manipuler et à mettre en œuvre, nous considérons que cette ontologie de grounding importante dans le cadre de nos travaux. Cependant, vu son caractère complexe et fastidieux, le remplissage de cette ontologie ne doit en aucun cas faire l'objet de tâches exclusivement manuelles mais doit en revanche être supportée par des outils semi-automatiques afin d'aider et d'assister l'utilisateur. De plus, nous estimons que la formalisation ainsi que la démarche de construction de cette ontologie peut être largement améliorée. Nous proposerons dans la section VII.6 une contribution à ce sujet à travers la notion d'ontologie de mappings syntaxiques.

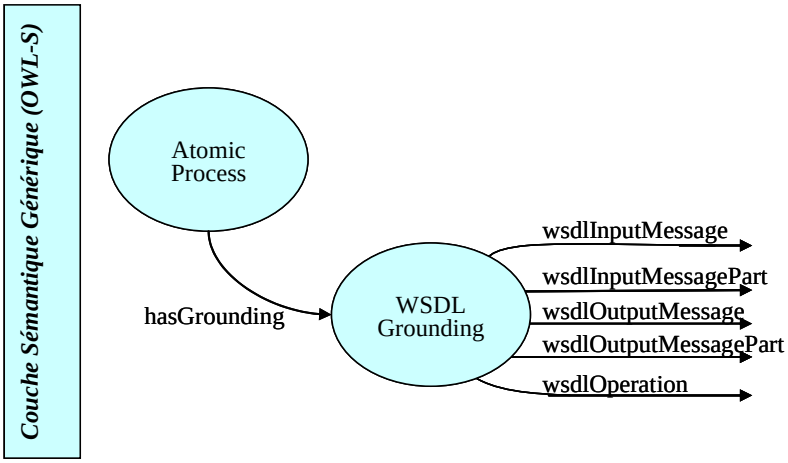

Figure VII.12. Ontologie de grounding de service

VII.4.1.1.4. Les ressources (OWL-S: Resource)

Les services ont souvent besoin de ressources pour pouvoir s'exécuter. Ces ressources sont variées et nombreuses. Il est donc intéressant de les définir dans une ontologie. L'ontologie de ressource (figure VII.13) telle que définie dans OWL-S possède un niveau d'abstraction assez élevé pour couvrir différentes ressources telles que les ressources temporelles, physiques, etc. On peut distinguer deux grandes caractéristiques des ressources en fonction du type d'allocation (AllocationType) et du type de capacité (CapacityType). Le type d'allocation permet de définir les ressources qui sont consommables (ConsumableAllocation) et celles qui restent réutilisables (ReusableAllocation). Les premières disparaissent avec l'exécution du service, tandis que les secondes sont à nouveau disponibles après l'exécution du service. Remarquons que les ressources consommables peuvent être, après utilisation, réapprovisionnées.

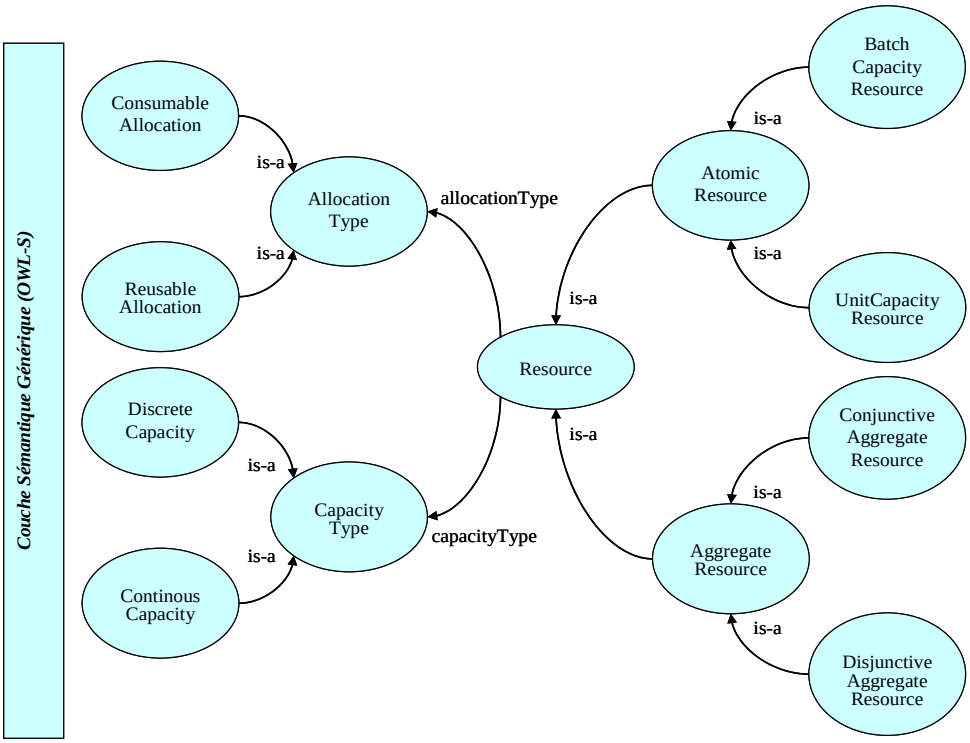

Figure VII.13. Ontologie de ressource de service

Le type de capacité (CapacityType) permet de définir la mesure de la quantité d'une ressource et qui peut être de deux sortes. La première est discrète (DiscreteCapacity), c'est-à-dire qu'une ressource de ce type est consommée en unités entières. La seconde est continue (ContinuousCapacity). Les ressources proposées par l'ontologie de ressources peuvent être atomiques ou agrégées. Dans le premier cas, nous avons une ressource unique qui est allouée pour l'utilisation du service. Dans le second cas, un ensemble ou un sous-ensemble de ressources doit être alloué afin d'exécuter le service.

L'ontologie de ressource de service telle que définie précédemment s'avère peu pertinente vis à vis des travaux que nous menons. Nous avons besoin de spécifier davantage la notion de ressource dans le cadre du projet MiMISI. En particulier, nous devons spécifier la notion de ressource organisationnelle et la notion de ressource technique, etc.

VII.4.1.2. Ontologie de description spécifique de service d'entreprise

Dans le but de mieux représenter les services d'entreprise dans le cadre de systèmes d'information industriels en général et dans le cadre spécifique du projet MiMISI, nous avons défini un certain nombre d'extensions pour l'ontologie générique OWL-S. Ces extensions sont récapitulées dans la couche sémantique spécifique des services d'entreprise (OWL-S+) de la figure VII.14 (qui ne reprend de la figure VII.8 que la partie associée à OWL-S+). Cette couche spécifique définit principalement trois concepts majeurs qui sont :

- le service fondamental (Fundamental Service);
- le service d'intégration (Integration Service);
- et le profil de service fondamental (Fundamental Service Profile).

Chacun de ces concepts est définie dans une ontologie spécifique qui est décrite ci-dessous. Signalons que l'ontologie spécifique de service d'entreprise repose sur la classification des services d'entreprise (tels que décrits dans le chapitre VI) ainsi que sur leurs caractéristiques particulières. Comme illustré sur la figure VII.14, les services d'entreprise sont répartis en services fondamentaux (Fundamental Service) et en services d'intégration (Integration Service). Les services fondamentaux permettent d'exposer sous forme de services les composants du système d'information industriel tandis que les services d'intégration constituent les moyens permettant d'intégrer les services fondamentaux. Un cas particulier important de services d'intégration est le Service Brokering (Brokering Service). Ce dernier est une ressource d'entreprise qui forme un nœud d'intégration et qui fournit des services fondamentaux aux acteurs de l'entreprise.

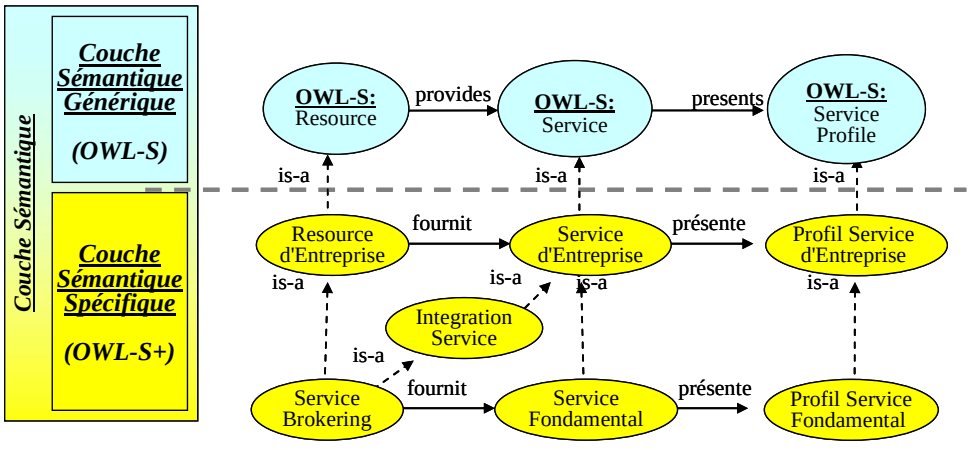

Figure VII.14. Ontologie de service d'entreprise (OWL-S+) [Izza et al. 2006-a]

VII.4.1.2.1. Ontologie des services fondamentaux

Les services fondamentaux constituent les principaux services d'entreprise qui exposent de façon découplée sous forme de services les composants du système d'information industriel. Dans notre modèle, nous avons retenus les quatre principaux types de services fondamentaux introduits au chapitre VI et qui sont (figure VII.15) : les services processus, les services fonctionnels, les services applicatifs, et les services techniques.

Les différents types de services sont reliés entre eux par deux types principaux de relations : des relations horizontales et des relations verticales qui sont définies comme suit :

- Les relations horizontales sont des relations définies au sein d'une même catégorie de services. Par exemple les services fonctionnels peuvent être liées entre eux et ces liaisons sont horizontales. Nous distinguons deux types de relations horizontales : la relation 'is-a" (est-un) et la relation "isPartOf" (est-partie-de).
    - La relation "is-a" est une relation qui spécifie qu'un service d'une certaine catégorie est une spécialisation d'un autre service de la même catégorie. Ce type de relation est la base de la construction de la taxonomie des services fondamentaux.
    - La relation "isPartOf" est une relation qui spécifie qu'un service d'une certaine catégorie est un assemblage de deux ou plusieurs autres services de la même catégorie. Ce type de relation est la base de la construction de la méronymie des services fondamentaux.
- Les relations verticales sont des relations définies entre services de différentes catégories. Par exemple les services fonctionnels peuvent être liés à des services applicatifs et ces liaisons sont des relations verticales. Ce type de relation est modélisé sur la figure grâce au lien "uses" qui permet ainsi de définir des couplages faibles entre les différentes catégories de services d'entreprise.

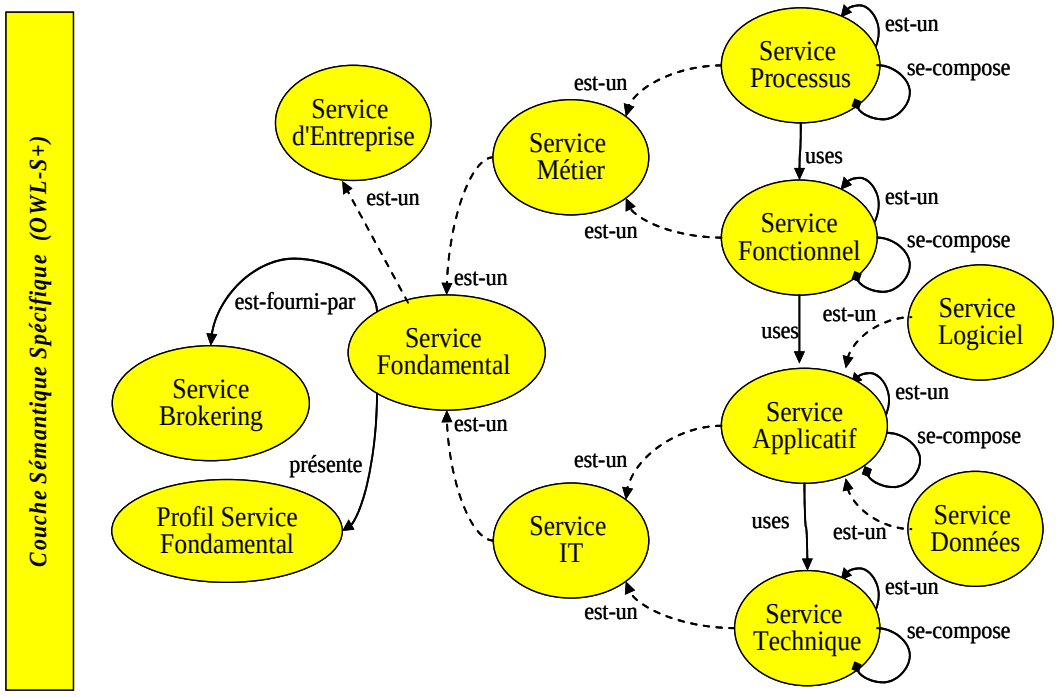

Figure VII.15. Ontologie de service fondamental d'entreprise [Izza et al. 2006-a]

Nous devons noter que l'ontologie des services fondamentaux est d'une grande importance. Elle permet de structurer l'ensemble des services fondamentaux grâce à la taxonomie et à la méronymie définies précédemment. Dans la suite, on fait très souvent référence à cette ontologie en utilisant le terme générique d'ontologie de service d'entreprise[50] (OSE).

Un autre aspect essentiel lié aux services fondamentaux est leur clustérisation. Comme nous l'avons présenté dans le chapitre VI, les services fondamentaux sont urbanisés dans la mesure où ils sont structurés en clusters homogènes et faiblement couplés. A ce titre, l'ontologie de clusters qui est définie en figure VII.16 permet de modéliser la clustérisation des services d'entreprise. Cette ontologie de cluster est automatiquement générée à partir des données produites par le processus de clustérisation et qui sont disponibles dans l'architecture de services d'entreprise (ASE).

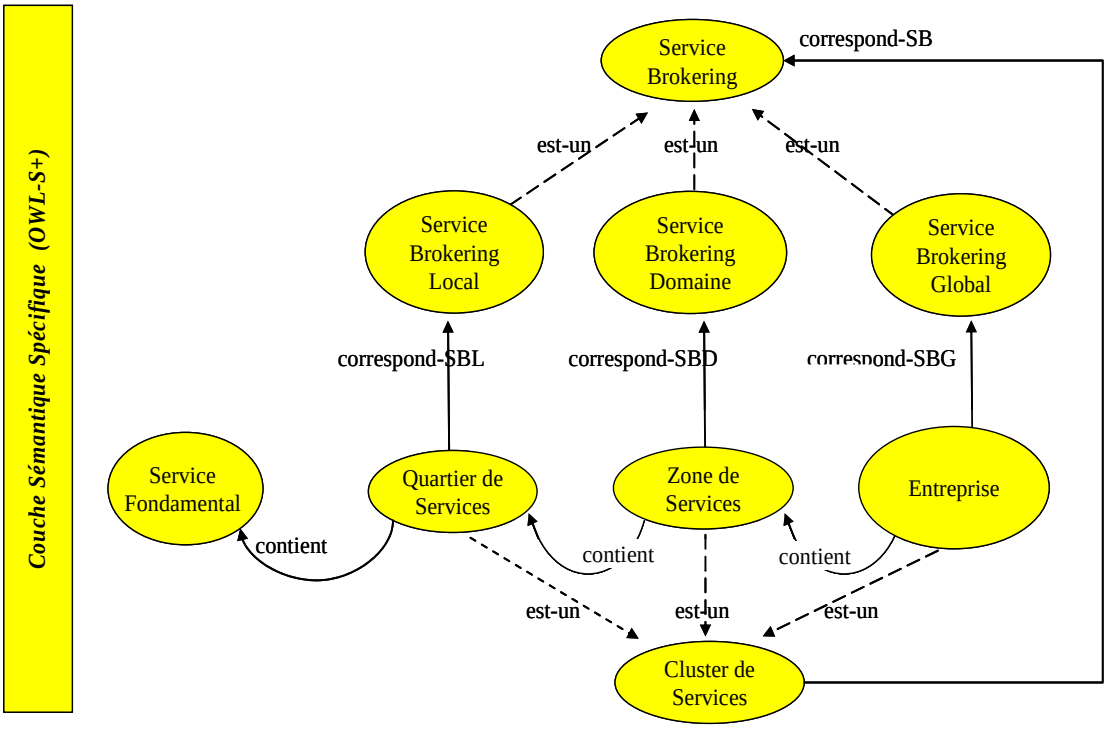

Figure VII.16. Ontologie de cluster de service [Izza et al. 2006-a]

On distingue deux types principaux de clusters : les quartiers et les zones. Un quartier de services (Service District) est un regroupement de services alors qu'une zone de service (Service Area) est un regroupement de quartiers. A chaque cluster est associé un nœud d'intégration implémenté sous forme d'un service d'intégration particulier, le Service Brokering. Ce dernier est décrit dans l'ontologie de service d'intégration.

Un autre aspect important qui caractérise les services fondamentaux et qui a fait l'objet d'une étude au chapitre VI est la notion de visibilité. En effet, à chaque service est associé une certaine visibilité qui est décrite grâce à l'ontologie de visibilité de services d'entreprise (figure VII.17).

Comme peut le montrer la figure VII.17, l'ontologie de visibilité des services est structurée en treillis. Au plus haut niveau on trouve la visibilité totale des services

[50] De façon plus rigoureuse, l'ontologie de service d'entreprise (OSE) est composée de trois ontologies : l'ontologie des services fondamentaux, l'ontologie des services d'intégration et l'ontologie de cluster de services.

(Toute Visibilité) et au plus bas niveau on trouve la non visibilité des services (Aucune Visibilité). Aux niveaux intermédiaires, on peut trouver les différentes visibilités possibles (visibilité IT locale, …, visibilité métier publique).

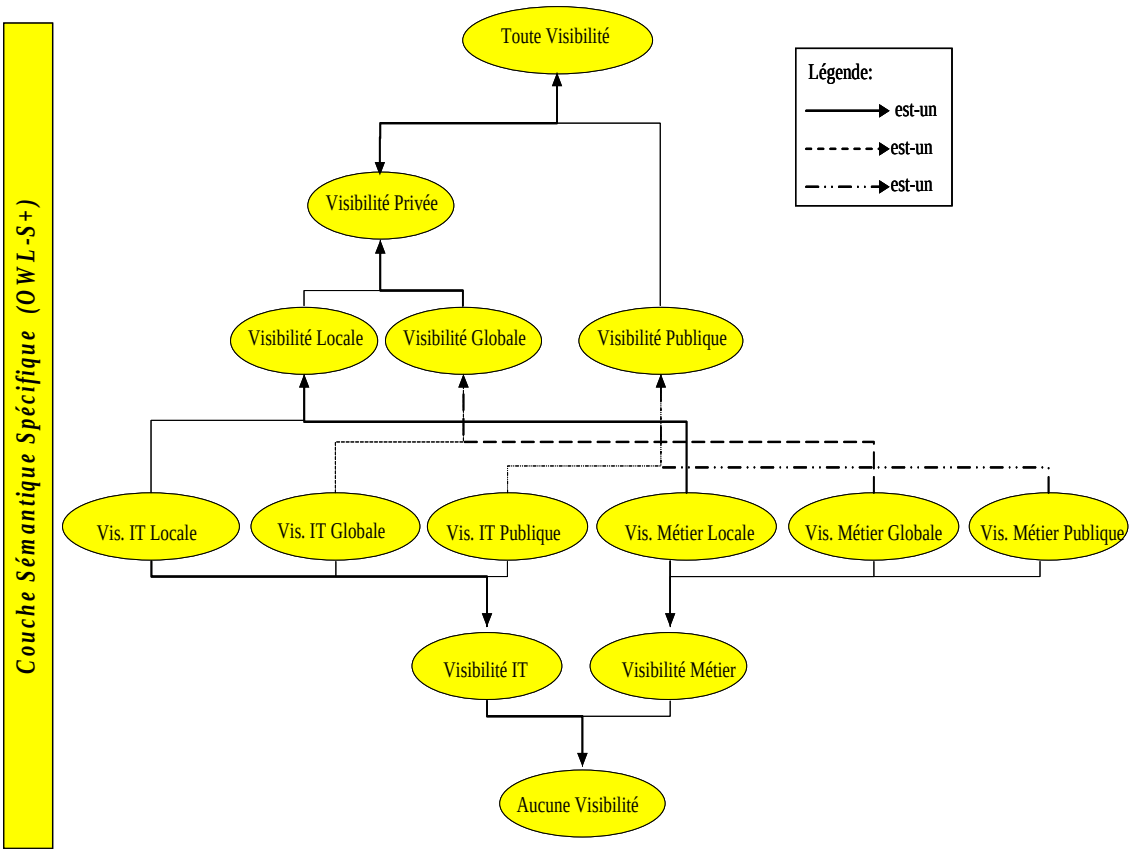

Figure VII.17. Ontologie de visibilité des services d'entreprise

VII.4.1.2.2. Ontologie de service d'intégration

L'ontologie de service d'intégration permet de définir les services d'intégration associés aux différents clusters de services. Le concept central de cette ontologie est celui du Service Brokering (Brokering Service). Un Service Brokering est un service dédié en charge du processus d'intégration permettant la gestion et l'orchestration des autres services d'intégration (tels que le service de découverte, le service de médiation, …) dans le but d'effectuer la découverte, la médiation pour intégrer les services fondamentaux. Le Service Brokering est considéré comme un tiers intermédiaire entre le client et le fournisseur de services. De cette manière, nos services fondamentaux sont découplés.

En outre, notre approche utilise plusieurs services brokering et qui sont structurés de façon hiérarchique en services brokering locaux, de domaine et global. Le service brokering local est utilisé pour intégrer localement les services fondamentaux au sein d'un quartier de services. Le service brokering de domaine permet d'intégrer les services brokering locaux au sein d'une même zone. Le service brokering global permet d'intégrer les services brokering de domaine existant au sein de l'entreprise. La figure VII.18 illustre les principaux concepts de l'ontologie de service brokering.

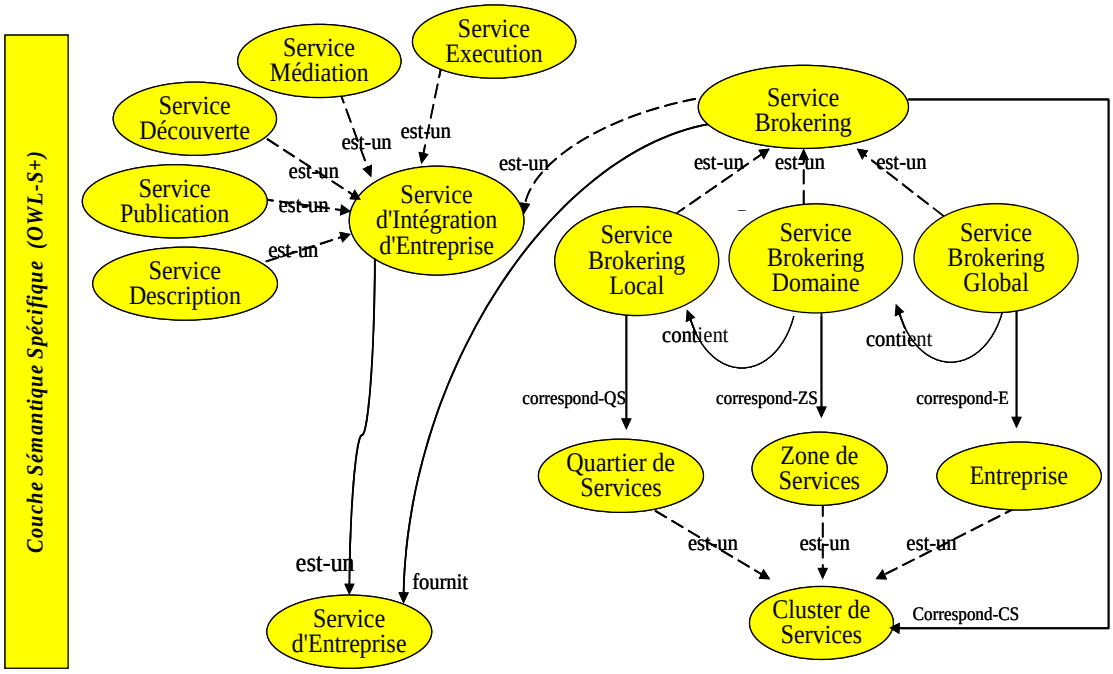

Figure VII.18. Ontologie de service brokering [Izza et al. 2006-a]

VII.4.1.2.3. Ontologie de Profil de service fondamental

La sémantique spécifique des services fondamentaux est publiée grâce au concept de profil de service d'entreprise et de façon plus détaillée par le concept profil de service fondamental (figure VII.19). Le concept de profil de service d'entreprise spécialise le concept *Profile* de l'ontologie générique Profile d'OWL-S. Dans notre approche, toutes les propriétés de profil sont utilisées et notamment les propriétés serviceClassification, serviceCategory et serviceParameter qui sont spécialisées de la suivante manière. La propriété serviceClassification est spécialisée afin de décrire la classification de services fondamentaux (c.f. § VII.4.1.2.1). La propriété serviceCategory est utilisée de façon optionnelle afin d'annoter les services par une taxonomie standard telle que NAICS ou UN/SPSC qui sont importantes pour des considérations d'intégration B2B (qui constitue une des perspectives de nos travaux). La propriété serviceParameter est spécialisée afin de décrire les caractéristiques portant sur la qualité de services et également d'autres caractéristiques importantes pour l'intégration intra-entreprise et qui sont :

- le *cluster* de services, qui permet de spécifier le cluster auquel appartient le service.
- la *vue* d'entreprise, qui permet de lier le service au composant du système d'information industriel qu'il expose. Cette notion de vue d'entreprise est importante et sera décrite un peu plus loin au sein de l'ontologie fondamentale d'entreprise.
- ou encore la *visibilité* du service, qui nous renseigne ainsi sur le caractère public ou privé du service considéré.

Notons que les deux premières propriétés sont spécifiques au profil de service fondamental alors que la propriété visibilité est une propriété de profil de service d'entreprise dans la mesure où elle peut concerner tous les services d'entreprise.

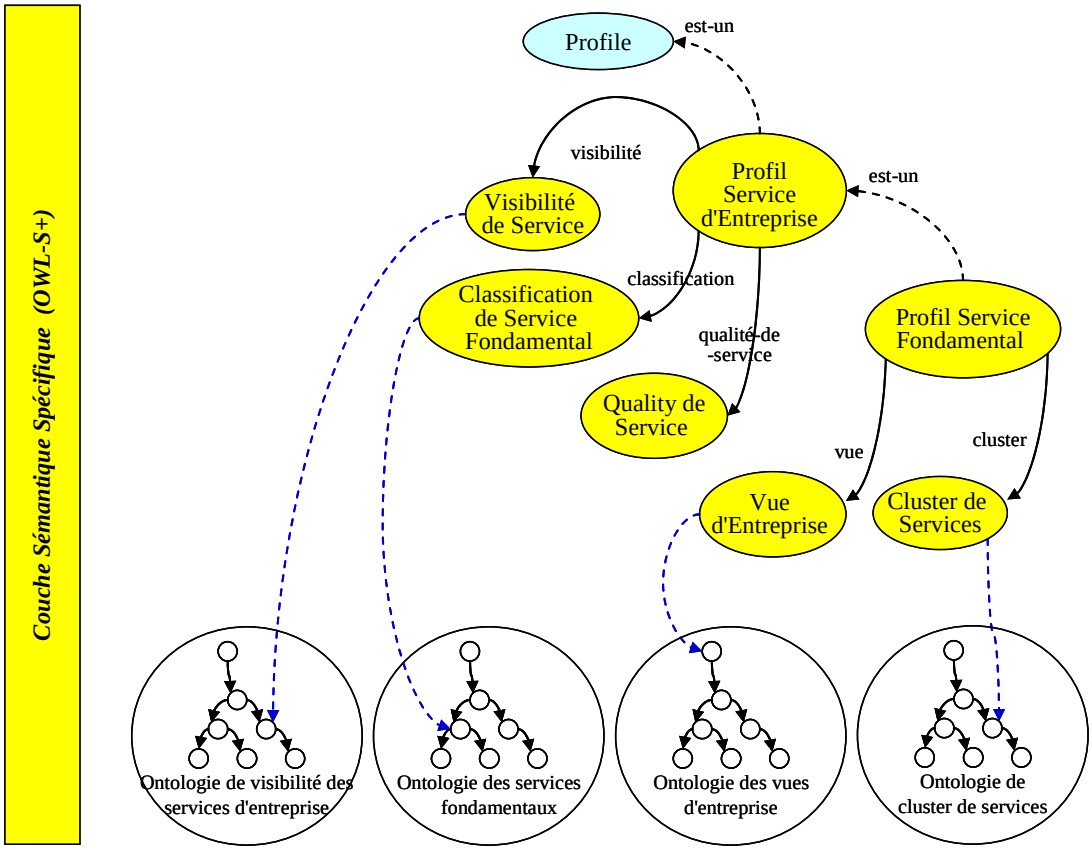

Figure VII.19. Ontologie de profil de service fondamental [Izza et al. 2006-a]

VII.4.2. Ontologie fondamentale d'entreprise

L'ontologie fondamentale d'entreprise (ou encore l'ontologie de base d'entreprise - OBE) permet de décrire la sémantique spécifique du système d'information d'entreprise. Comme peut le montrer la figure VII.20, l'ontologie de base de l'entreprise repose principalement sur le concept de vue d'entreprise. Une vue d'entreprise est un aspect que peut présenter le système d'information d'entreprise. On distingue trois vues complémentaires qui sont : la vue métier, la vue fonctionnelle et la vue informatique. Ces différentes vues représentent des concepts génériques qui permettent de décrire respectivement le métier, les fonctions et l'aspect informatique de l'entreprise. Chacune de ces vues est définie par une ontologie : l'ontologie métier (qui décrit les vues métiers, c-à-d les processus métier), l'ontologie fonctionnelle (qui décrit les vues fonctionnelles, c-à-d les fonctionnalités offertes par les applications industrielles), et l'ontologie informatique (qui décrit les vues informatiques, c-à-d les données, les logiciels et les aspects techniques des applications industrielles).

Les ontologies informatiques et notamment les ontologies de données sont les ontologies de base. Elles permettent de décrire la sémantique des applications et notamment des données manipulées par les applications industrielles. Elles sont nécessaires dans tous les cas, peu importe si on dispose d'ontologies fonctionnelles et/ou métier. Les ontologies fonctionnelles permettent de favoriser la mise en œuvre de mécanismes de réutilisation et d'invocation en définissant la sémantique liée aux fonctionnalités fournies par les applications industrielles. Les ontologies de processus permettent de définir la sémantique autour de l'exécution et de l'orchestration des processus.

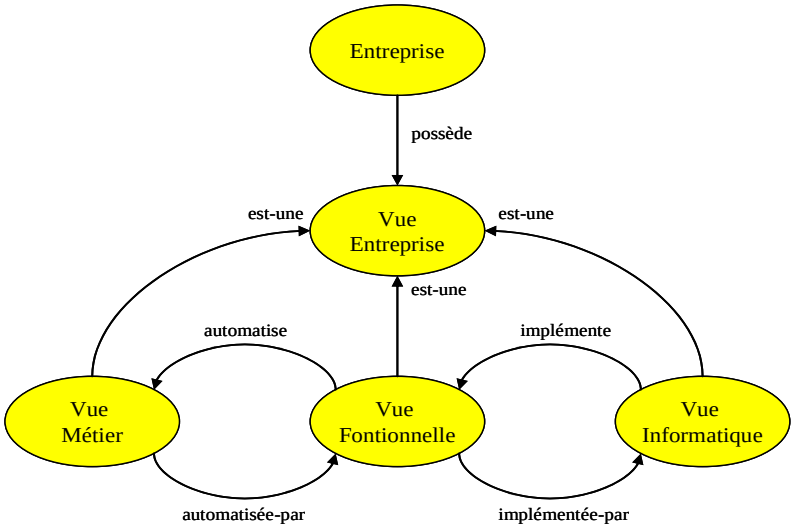

Figure VII.20. Ontologie fondamentale de l'entreprise (OBE)

Bien entendu, ces différentes vues sont interdépendantes dans le sens où elles entretiennent entre elles des interactions. Ainsi, un élément de la vue métier peut être automatisé par un ou plusieurs éléments de la vue fonctionnelle qui peuvent à leur tour être implémentés par un ou plusieurs éléments de la vue informatique. Notons que toutes ces ontologies ont été construites avec un esprit de flexibilité dans la mesure où l'on adopte des contraintes assez larges (peu strictes) permettant ainsi de rendre certains éléments facultatifs et procurant ainsi de la flexibilité à la modélisation. Ces différentes vues (ontologies) sont décrites ci-dessous.

VII.4.2.1. Ontologie métier d'entreprise

L'ontologie métier d'entreprise (OME), ou plus simplement ontologie métier (figure VII.21), permet de définir la vue métier décrivant ainsi la sémantique liée à la coordination des processus et à l'orchestration des applications. Elle est nécessaire pour prendre en compte les processus métier et les workflow d'entreprise.

Le concept principal de cette ontologie est celui du *Processus métier* (figure VII.21). Un processus métier est un enchaînement d'*Activités Métier* qui vise à atteindre un objectif établi. Une activité métier constitue l'unité de décomposition fonctionnelle du processus métier et elle peut à son tour être décomposée en *Tâches Métier*. Chacune des tâches métier peut manipuler un certain nombre d'*Evénements Métier*. Chaque concept métier, également appelé *Vue Métier*, est lié à un concept externe *Vue Fonctionnelle* appartenant à l'ontologie fonctionnelle d'entreprise (OFE). Ce lien permet de mettre en évidence l'automatisation des aspects métiers par l'intermédiaire des aspects fonctionnels du système d'information industriel. Plus précisément, cette liaison permet de définir la correspondance métier-fonction et elle traduit la relation d'utilisation des éléments fonctionnels par les éléments métiers. Plus particulièrement, les tâches métier et les événements métiers sont respectivement liés aux éléments fonction et flux informationnel de la vue fonctionnelle. De plus, nous devons noter que certaines vues métier peuvent être exposées en services processus. Ces derniers sont décrits dans l'ontologie de service d'entreprise (OSE[51]) décrite à la section § VI.4.1.2.

[51] Rappelons que l'ontologie de service d'entreprise (OSE) est composée de trois ontologies : l'ontologie des services fondamentaux, l'ontologie des services d'intégration et l'ontologie de cluster de services.

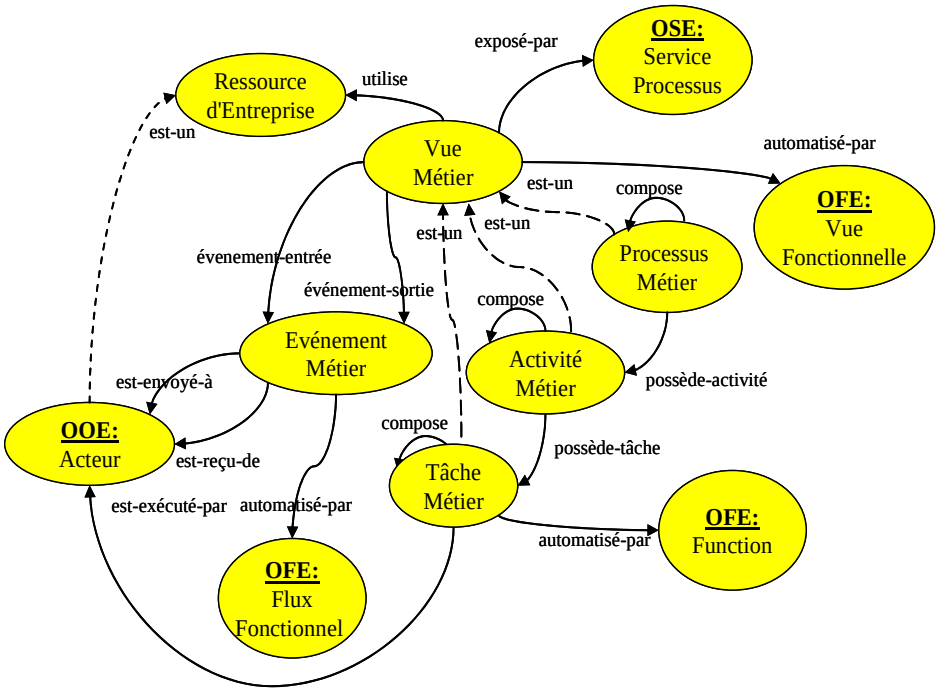

Figure VII.21. Ontologie métier d'entreprise (OME)

Par ailleurs, il est important de préciser que les tâches métiers sont exécutées par des acteurs de l'entreprise qui sont considérés comme un cas particulier de ressource d'entreprise. Ces acteurs sont décrits dans l'ontologie organisationnelle d'entreprise (OOE) qui est présentée dans la figure ci-après (figure VII.22).

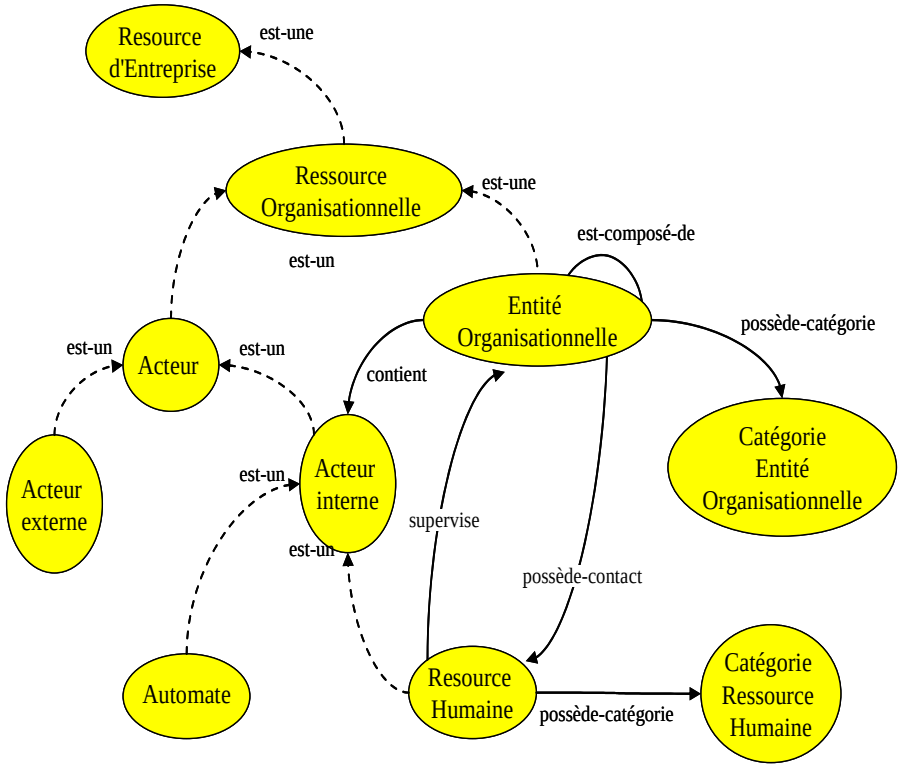

Figure VII.22. Ontologie organisationnelle d'entreprise (OOE)

Comme on peut le constater sur la figure VII.22, un acteur peut être un humain (ressource humaine) ou un automate qui opère des tâches métiers. Chaque acteur appartient à une entité organisationnelle. Chaque entité organisationnelle est une ressource organisationnelle au même titre que les acteurs mais de plus grosse granularité dans la mesure où elles sont considérées plutôt comme un regroupement de ressources organisationnelles (d'autres entités organisationnelles ou d'acteurs). Chaque entité organisationnelle est supervisée et/ou représentée par une ressource humaine. Les ressources organisationnelles sont classées selon une catégorie : les entités organisationnelles sont classées selon une catégorie d'entité organisationnelle (site, division, département, poste de travail, …), tandis que les ressources humaines sont classées selon une catégorie de ressource humaine (directeur, cadre supérieur, cadre, agent d'exécution, …).

VII.4.2.2. Ontologie fonctionnelle d'entreprise

L'ontologie fonctionnelle d'entreprise (OFE), ou plus simplement ontologie fonctionnelle, permet de définir la sémantique associée aux différentes fonctions automatisables du système d'information. Cette sémantique est nécessaire pour décrire de façon stable et de façon conceptuelle le système d'information industriel permettant ainsi un meilleur découplage entre le métier et l'informatique et également une meilleure pratique en matière de réutilisation, de publication et de découverte des fonctionnalités du système d'information.

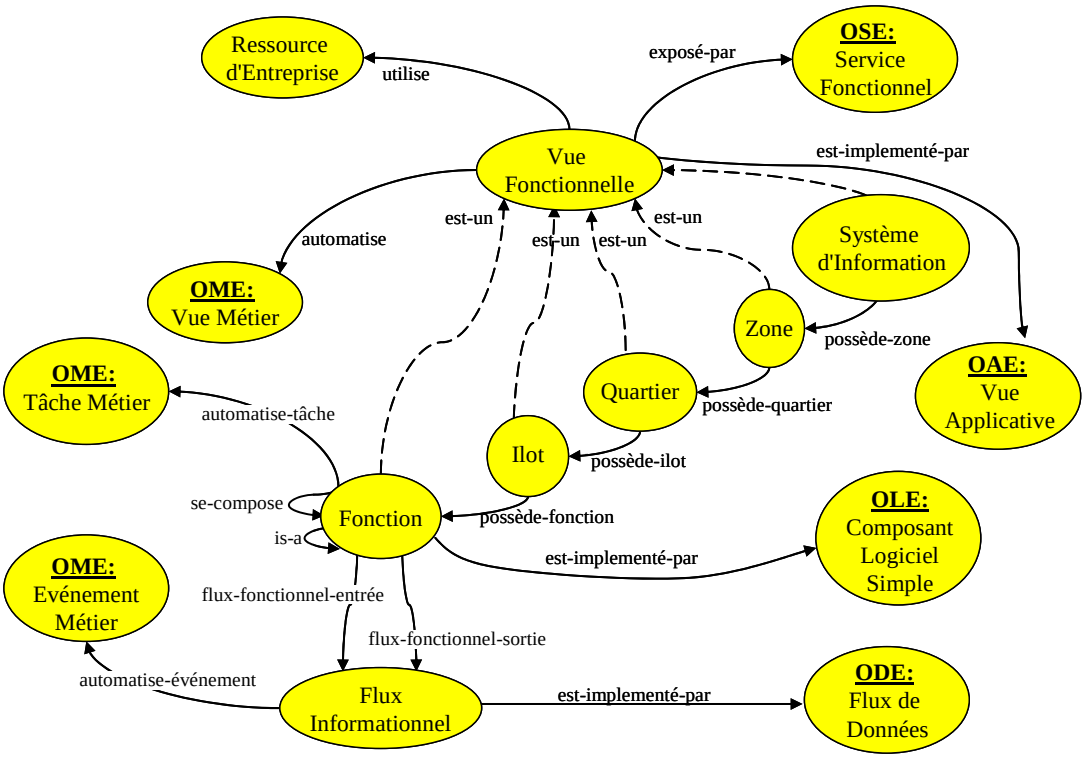

Figure VII.23. Ontologie fonctionnelle d'entreprise (OFE)

Comme illustré sur la figure VII.23, le principal concept de cette ontologie est celui de *Fonction*. Une fonction est un sous-ensemble cohérent du système d'information. Le concept de fonction est important car il constitue un élément automatisable et réutilisable du système d'information. Une fonction utilise des flux informationnels en

entrée et en produit des flux informationnels en sortie. Dans le but de rendre le système urbanisé, les fonctions sont regroupées en îlots fonctionnels qui sont à leur tour regroupés en quartiers fonctionnels qui sont également à leur tour regroupés en zones fonctionnelles. Cette décomposition fonctionnelle du système d'information est issue des principes de l'urbanisation du système d'information tels que ceux issus des travaux de (Longépé, 2001) et de (Sassoon, 1998).

Certains concepts de cette ontologie sont reliés à d'autres concepts externes issus principalement de l'ontologie métier (OME) et de l'ontologie applicative (OAE). C'est le cas des concepts *Vue Fonctionnelle*, *Fonction* et *Flux Informationnel* qui sont liés d'une part aux concepts externes *Vue Métier*, *Tâche Métier* et *Evénement Métier* de l'ontologie métier d'entreprise (OME), et d'autre part aux concepts externes *Vue Applicative*, *Composant Applicatif* et *Flux Applicatif* de l'ontologie d'application d'entreprise (OAE).

Par ailleurs, il est important de constater que certaines vues fonctionnelles peuvent s'exposer en services fonctionnels, concept externe qui est déjà défini dans l'ontologie de service d'entreprise (OSE).

VII.4.2.3. Ontologie informatique d'entreprise

L'ontologie informatique permet de décrire le système informatique de l'entreprise. Ce dernier peut être décrit selon deux points de vue qui sont la vue applicative et la vue technique (figure VII.24). La vue applicative utilise la vue technique dans la mesure où elle est construite sur la vue technique. Ces deux vues sont respectivement décrites par l'ontologie d'application d'entreprise (OAE) et l'ontologie technique d'entreprise (OTE) qui sont présentées dans ce qui suit.

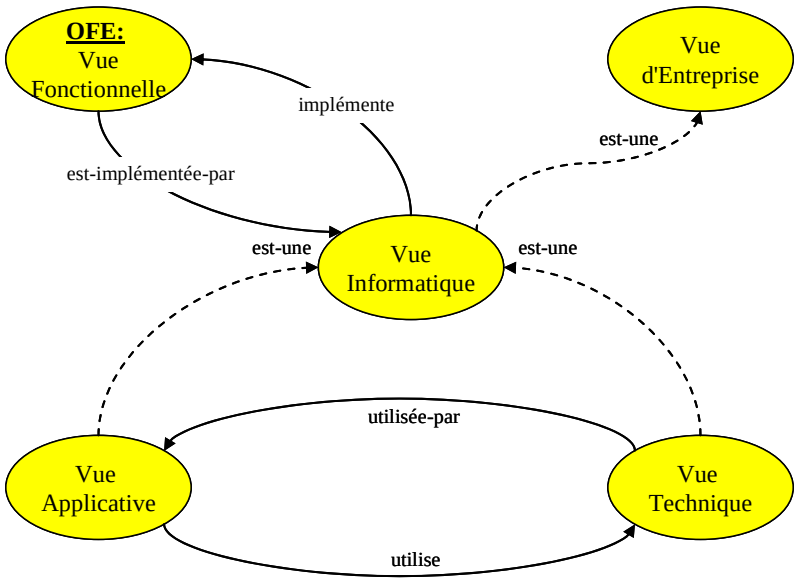

Figure VII.24. Ontologie informatique d'entreprise (OIE)

VII.4.2.4. Ontologie d'application d'entreprise

L'ontologie d'application d'entreprise (OAE) permet de décrire les applications du système informatique de l'entreprise. Cette ontologie est nécessaire pour cataloguer et rendre accessibles les différentes applications d'entreprise. Cette ontologie se base fondamentalement sur le concept de *vue applicative* qui constitue un concept générique

permettant de décrire les multiples aspects associés aux applications d'entreprise.

Comme le montre la figure VII.25, on peut distinguer deux types de vues applicatives qui sont la *vue logicielle* et la *vue données*. La *vue logicielle* permet de décrire les systèmes logiciels alors que la *vue données* permet de décrire le patrimoine informationnel de la *vue applicative*. La *vue logicielle* est décrite par l'ontologie logicielle d'entreprise (OLE), alors que la *vue données* est décrite par l'ontologie de données d'entreprise (ODE).

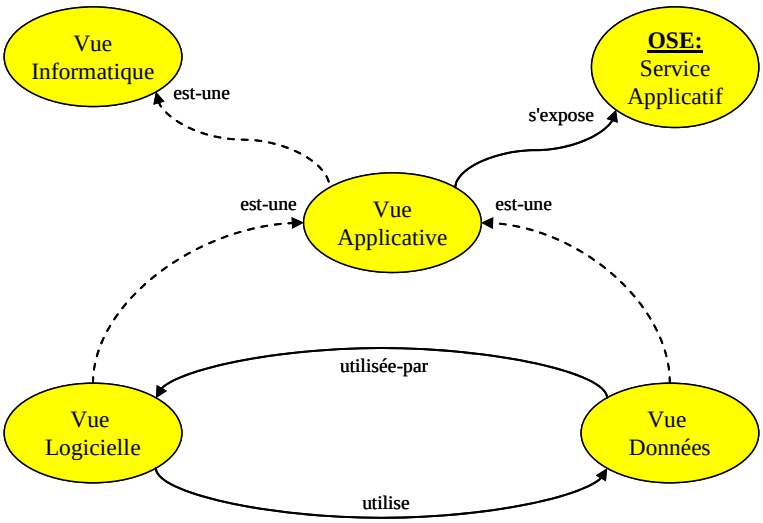

Figure VII.25. Ontologie d'application d'entreprise (OAE)

L'ontologie logicielle d'entreprise (OLE) permet de décrire les différents systèmes logiciels du système informatique et qui sont impliqués dans le projet d'intégration (figure VII.26).

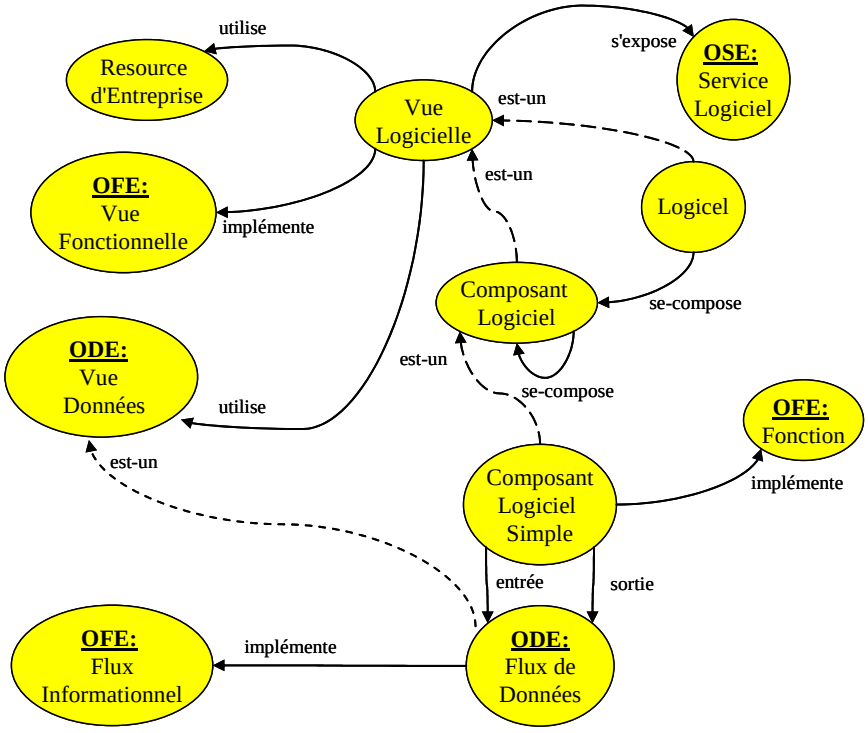

Figure VII.26. Ontologie logicielle d'entreprise (OLE)

Comme on peut le constater, le concept principal de cette ontologie est la vue logicielle et qui peut se spécialiser en logiciel et composant logiciel. Le logiciel se compose d'un certain nombre de composants logiciels et qui peuvent à leur tour se décomposer en composants plus élémentaires. Les composants logiciels élémentaires permettent d'implémenter une fonction du système d'information et possèdent des entrées et des sorties. De même que pour les fonctions de l'ontologie fonctionnelle, certains éléments de la vue logicielle peuvent s'exposer en services logiciels.

L'ontologie de données d'entreprise (ODE) permet de décrire l'ensemble des données manipulées par les composants applicatifs (et aussi par les services d'entreprise) décrits précédemment. La figure VII.27 en présente un extrait des principaux concepts de cette ontologie.

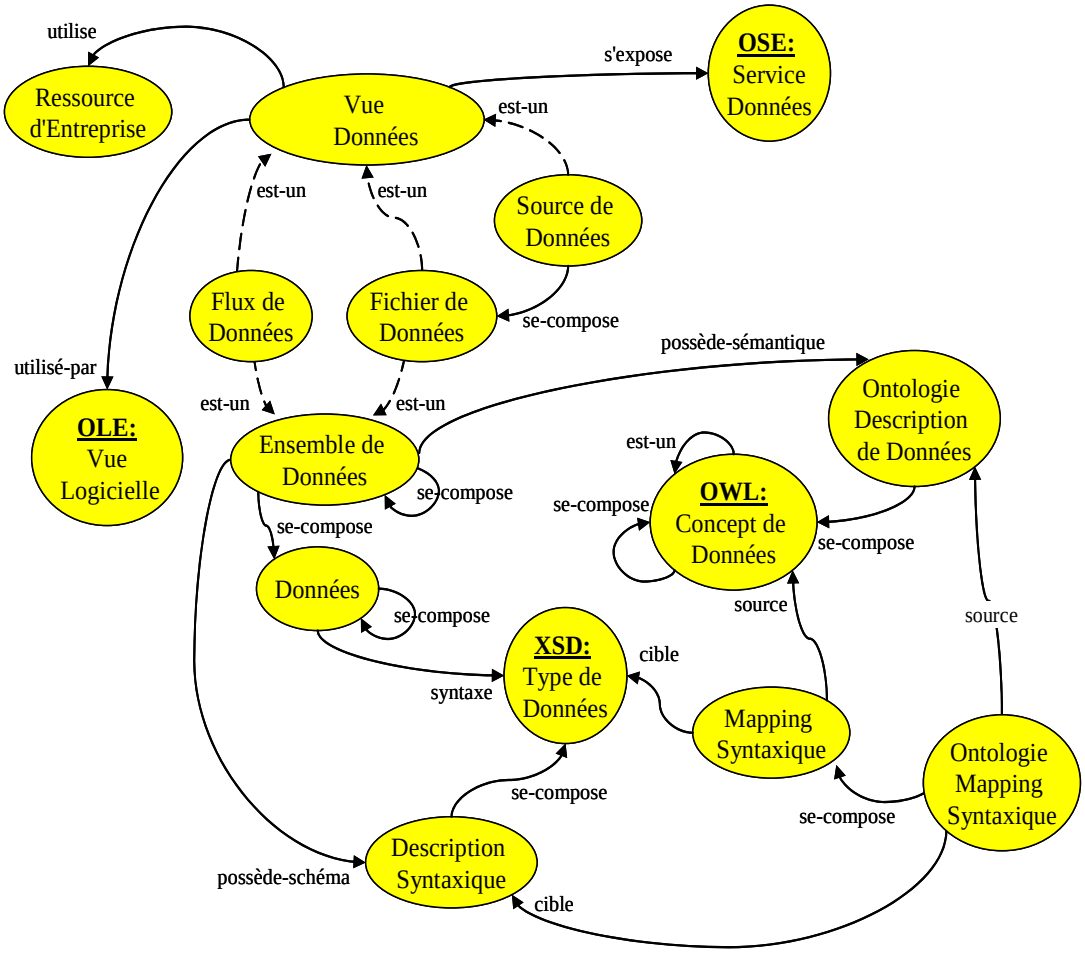

Figure VII.27. Ontologie de données d'entreprise (ODE)

Le concept principal de cette ontologie est la *vue données* qui constitue un concept générique permettant de représenter les différents éléments informationnels à savoir les *sources de données*, les *fichiers de données* ainsi que les *flux de données* (figure VII.28). Les fichiers et les flux de données sont considérés comme un *ensemble de données* qui peuvent à leur tour être composés en ensembles plus complexes. A ce titre ils comportent un certain nombre de données qui sont décrites du point de vue syntaxique et sémantique. De même que pour les autres ontologies, certains éléments de la *vue données* peuvent être exposés sous forme de *service de données*.

Nous devons signaler que la sémantique des données est principalement formalisée par la notion de *concept de données* (OWL: Concept de données). Un ensemble de ces

concepts forment la notion d'*ontologie spécifique de données*. Cette dernière sert d'ontologie de base pour décrire notamment la sémantique des entrées et des sorties des services d'entreprise. Des liaisons entre la sémantique et la syntaxe des données peuvent être définies grâce à des *mappings syntaxiques* (ou mappings syntaxe-sémantique) et qui forment la notion d'*ontologie de mapping syntaxique*.

VII.4.2.5. Ontologie technique d'entreprise

L'ontologie technique d'entreprise (OTE) permet de décrire l'aspect technique du système informatique de l'entreprise. Cette ontologie est nécessaire pour cataloguer et rendre accessibles les différentes technologies utilisées au sein de l'entreprise. Cette ontologie se base fondamentalement sur le concept de *vue technique* qui constitue un concept générique permettant de décrire les multiples aspects associés aux technologies informatiques de l'entreprise.

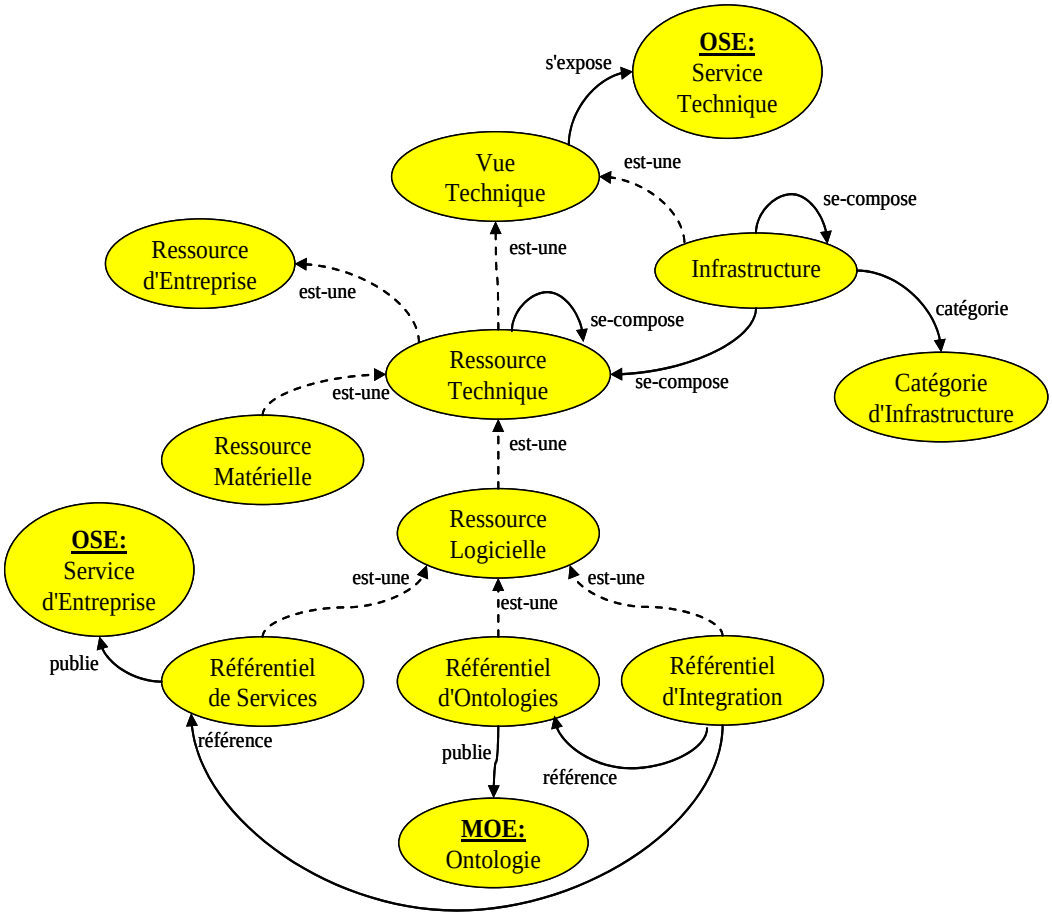

Figure VII.28. Ontologie technique d'entreprise (OTE)

Comme le montre la figure VII.29, on peut distinguer deux types de vues techniques qui sont l'infrastructure technique et la ressource technique. L'infrastructure technique permet de décrire les ressources techniques du système informatique alors que les ressources techniques permettent de décrire les différentes ressources matérielles et logicielles (de base) qui constituent l'infrastructure technique. Les ressources techniques logicielles sont principalement classées en trois types complémentaires qui sont le référentiel de services, le référentiel d'ontologies et le référentiel d'intégration. Le référentiel de services est un registre dans lequel les services d'entreprise sont publiés.

Le référentiel d'ontologies est un registre dans lequel les ontologies d'entreprise sont publiées. Le référentiel d'intégration est un registre qui stocke des données pertinentes pour l'intégration des applications d'entreprise. Ce dernier référentiel peut référencer aussi bien le référentiel de services que le référentiel d'ontologies.

Nous signalons au passage que les ressources techniques sont considérées comme un cas particulier de ressources d'entreprise, qui rappelons-le, peuvent comporter d'autres types de ressources comme les ressources organisationnelles, etc.

VII.4.3. Méta-ontologie d'entreprise

La méta-ontologie d'entreprise (MOE) permet de décrire les différentes ontologies de notre approche. Cette ontologie est nécessaire pour cataloguer et rendre accessibles les différentes ontologies utilisées au sein de l'entreprise. Cette ontologie se base fondamentalement sur le concept d'*Ontologie* qui constitue un concept générique permettant de décrire les différentes ontologies d'entreprise utilisées pour formaliser les multiples aspects du système d'information industriel.

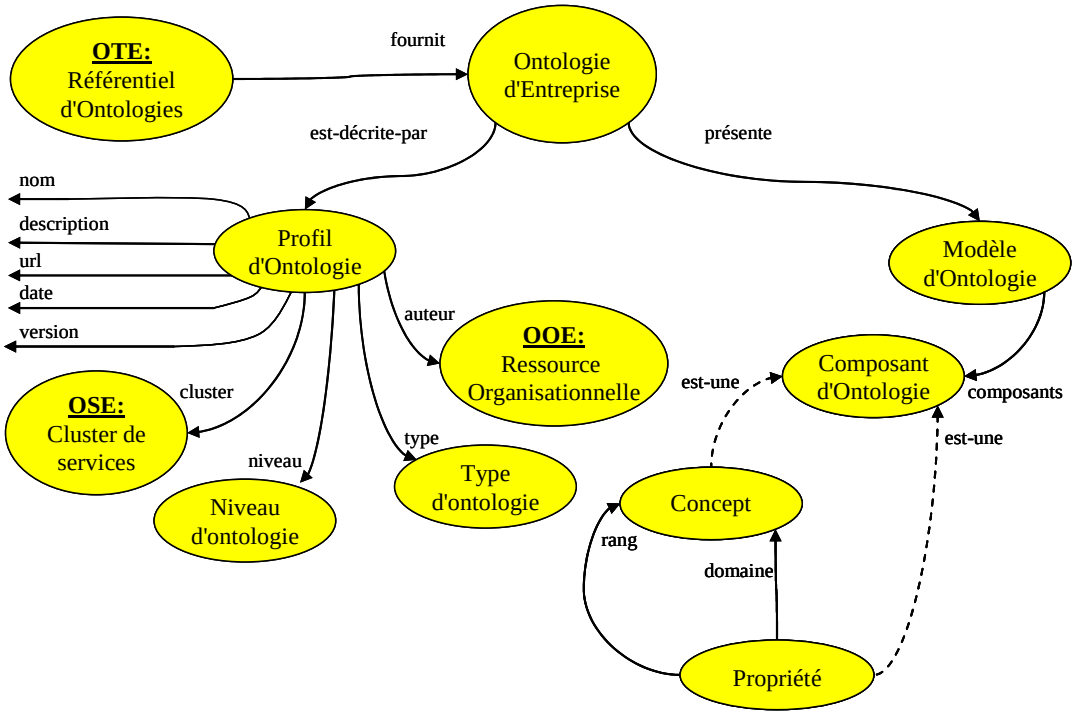

Figure VII.29. Méta-ontologie d'entreprise (MOE)

Comme on peut le constater sur la figure VII.29, chaque ontologie est décrite par un profil et un modèle. Le profil comprend un certain nombre de propriétés permettant d'afficher les caractéristiques portant sur le nom, la description; l'url, l'auteur, la date de création, la version, le type (ontologie de service, ontologie fondamentale, ontologie de mapping, etc.), le niveau (ontologie locale, globale, de domaine) et le cluster de l'ontologie. Le modèle permet de spécifier le contenu de l'ontologie en termes de concepts et de propriétés qui sont définis. De plus, une ontologie est associée à un référentiel qui la publie pour la rendre accessible aux autres agents susceptibles de la découvrir et l'utiliser pour leurs tâches d'intégration.

VII.5. Urbanisation sémantique

La section précédente a permis de présenter l'ensemble des ontologies nécessaires à mettre en œuvre pour décrire efficacement la sémantique associée aux services d'entreprise. A présent nous allons nous intéresser à l'aspect architectural ou structurel de nos ontologies. Nos ontologies fondamentales doivent être correctement structurées afin d'apporter le maximum de flexibilité et d'efficacité dans le processus de construction et de gestion des ontologies dans le cadre de grandes entreprises industrielles. De plus, nous avons besoin d'une approche de modularisation qui va permettre d'augmenter les performances des outils de gestion de la sémantique qui présentent par ailleurs certaines limites en fonction du volume des ontologies à charger. Pour cela, nous proposons une approche de structuration appelée urbanisation ontologique (ou urbanisation sémantique). Cette dernière complète l'urbanisation orientée services introduite au chapitre VI.

L'urbanisation ontologique que nous proposons et dont certains aspects ont été exposés dans [Izza et al. 2005-a][Izza et al. 2005-d][Izza et al. 2006-c], est en quelque sorte une extension de l'approche hybride présentée dans le chapitre IV. Elle est motivée par le fait qu'aucune des architectures ontologiques actuellement existantes (ontologie unique; ontologies multiples et indépendantes; et approche hybride [Wache et al. 2001] n'est suffisamment appropriée pour décrire et structurer correctement la sémantique des systèmes d'information dans un cas complexe d'une entreprise grande et dynamique tel est le cas dans le projet MiMISI. Dans notre approche, nous utilisons une architecture d'ontologies à trois niveaux (figure VII.30): niveau local, niveau de domaine et niveau global.

Le niveau local définit des ontologies locales qui permettent de fournir la sémantique pour la description des services fondamentaux. Les ontologies locales définissent la notion de contexte de service. Elles contiennent précisément la sémantique partagée au sein d'un quartier de services. Du fait que nous disposons d'une multitude de quartiers de services, et du fait que les services concernés sont généralement des entités évolutives (soumises à de fréquents changements), nous disposons par conséquent d'une multitude d'ontologies locales qui sont aussi évolutives dans le sens où elles peuvent être soumises à de fréquents changements.

Les ontologies de domaine peuvent être considérées comme la généralisation d'un certain nombre d'ontologies locales qui appartiennent à un même domaine de connaissance (Manufacturing, Ressources Humaines, Système central, etc.) formalisé par la notion de zone de services. Ces ontologies sont considérées comme la sémantique partagée au sein d'une zone de services. Elles peuvent servir pour la réconciliation des ontologies locales impliquées lors des échanges entre services fondamentaux. Du fait que ces ontologies concernent fondamentalement les domaines de connaissances de l'entreprise, et du fait que ces derniers sont généralement limités en nombre et relativement stables, nous disposons par conséquent de quelques ontologies de domaine qui sont relativement stables dans le sens où elles sont peu évolutives.

Enfin, l'ontologie globale[52] est considérée comme une généralisation des ontologies de domaine. Elle constitue la racine de la hiérarchie d'ontologies et peut servir pour la réconciliation des ontologies de domaine et aussi pour l'intégration externe. Elle est

[52] Nous devons noter qu'il nous arrive parfois de parler d'ontologies globales au pluriel pour designer le fait que notre ontologie globale peut être décomposée en trois sous-ontologies globales : ontologie de données globale, ontologie fonctionnelle globale et ontologie métier globale.

considérée comme la sémantique partagée au sein de l'entreprise. Du fait que cette ontologie est générique, elle est par conséquent unique et de plus elle est rarement évolutive.

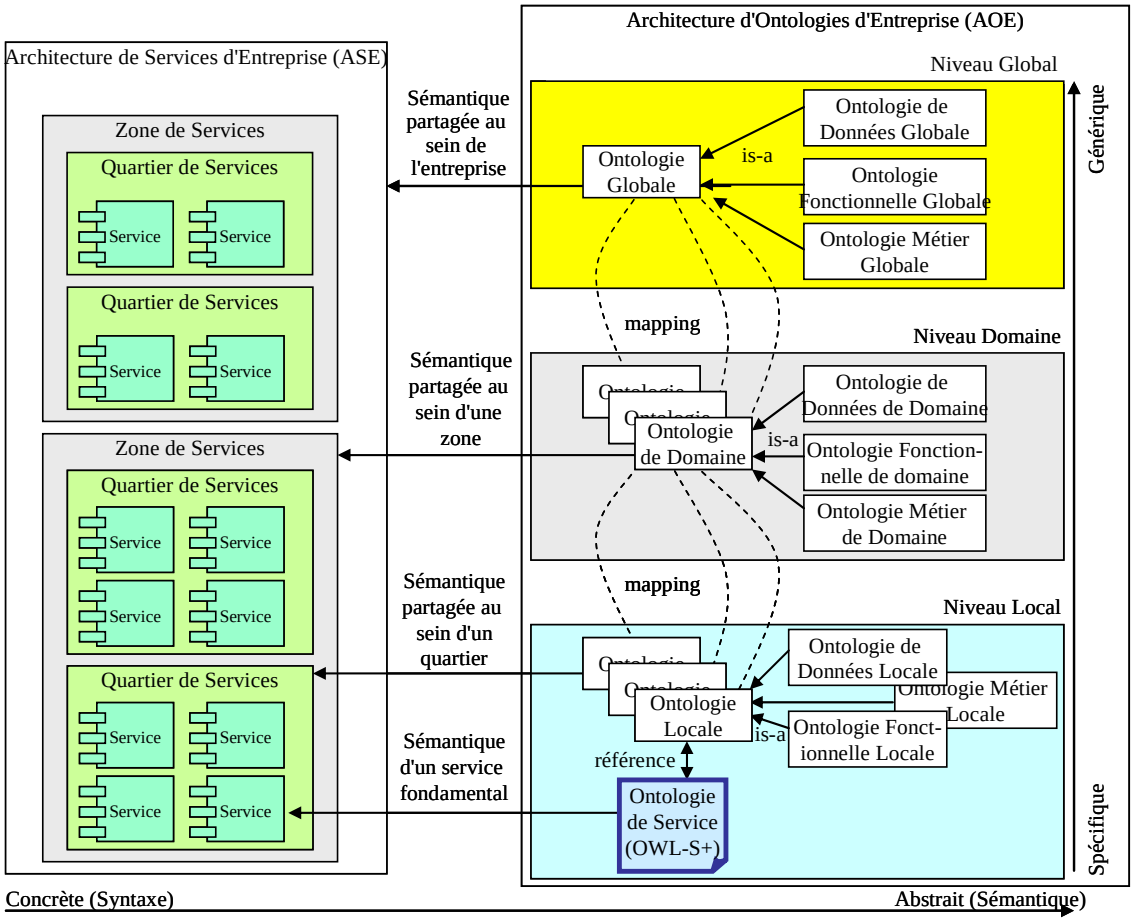

Figure VII.30. Principe d'urbanisation des ontologies (adapté de [Izza et al. 2006-c])

De façon plus formelle, la figure VII.31 illustre le principe de structuration des ontologies fondamentales. Comme on peut le constater, les trois niveaux de structuration des ontologies sont associés aux trois niveaux de structuration des services, et ces trois niveaux sont liés entre par des relations de mappings (maps-To). Les relations maps-To sont en réalité des liens plus complexes qui sont traités dans la section VII.6.4 qui expose la démarche de construction des mappings.

Nous devons souligner que cette urbanisation concerne la plupart des ontologies que nous avons précédemment définies. Aussi, nous disposons par exemple de plusieurs ontologies de données locales, quelques ontologies de données de domaine et une seule ontologie de données globale. Cette architecture hiérarchique concerne aussi les ontologies fonctionnelles, les ontologies métier et l'ontologie de service. Cette dernière comporte à ce titre un niveau local qui correspond à la description des services fondamentaux, un niveau de domaine qui correspond à la description des quartiers de services, et enfin un niveau global qui correspond à la description des zones de services.

En appliquant ces principes d'urbanisation, l'ontologie de données est décomposée en trois catégories: ontologie de données globale (ODG), ontologies de données de domaine (ODD) ontologies de données locales (ODL). A titre d'exemple, dans le contexte du domaine microélectronique, l'ODG inclut des concepts génériques tels que EQUIPMENT, PRODUCT, PEOPLE et TASK, tandis que les ODD et les ODL incluent

des concepts de plus en plus spécifiques tels que : OPERATOR et OPERATION dans le cas d'ODD, et TECHNICAL-OPERATOR et MAINTENANCE-OPERATION dans le cas d'ODL.

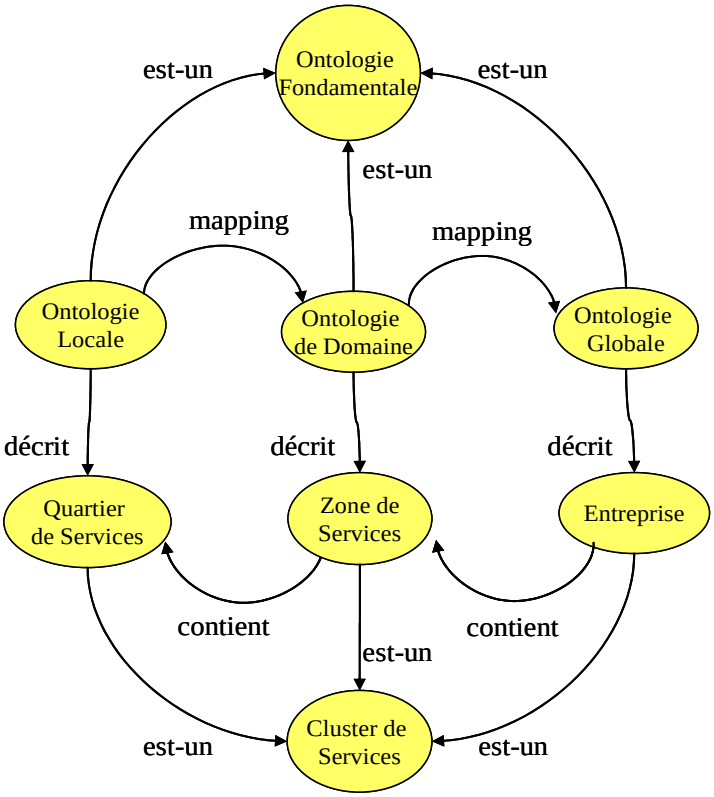

Figure VII.31. Principe de Structuration des ontologies

Contrairement aux ontologies de données qui sont basées sur des objets, les ontologies fonctionnelles sont basées sur des classifications de verbes (ou des actions). Par exemple, dans le cadre de la gestion préventive de la maintenance, on peut utiliser la fonctionnalité QUERY-TECHNICIAN-QUALIFICATION qui est un concept fonctionnel spécifique permettant d'effectuant des recherche de qualifications. Cette fonctionnalité peut être éventuellement substituée par une autre fonctionnalité équivalente telle que: SEARCH, LIST ou FIND-TECHNICIAN-QUALIFICATION qui permettent de consulter des qualifications d'opérateurs. Cet exemple réfère précisément à des concepts d'une ontologie fonctionnelle locale qui permet de fournir des concepts fonctionnels du quartier considéré. De même que pour les ontologies de données, nous avons aussi plusieurs ontologies fonctionnelles locales. De plus, notons que nous avons aussi une seule OFG (ontologie fonctionnelle globale) qui fournit des concepts fonctionnels (tels que QUERY, DELETE, CREATE, …), et plusieurs OFD (ontologie fonctionnelle de domaine) qui fournit des concepts fonctionnels plus spécifiques et stables (tels que QUERY-TECHNICIAN, CREATE-TECHNICIAN, …).

La figure VII.32 présente un exemple typique d'urbanisation de la sémantique. Comme on peut le constater, nous avons une ontologie globale contenant une ontologie globale de données et une ontologie globale de fonctions et d'un certain nombre ontologies de domaines pour capturer la sémantique respectivement du domaine des relations humaines (RH), de la fabrication et des tests. Au sein de chaque domaine, nous avons un certain nombre d'ontologies locales associées aux différents sous-domaines obtenus grâce au processus de clustérisation vu au chapitre VI.

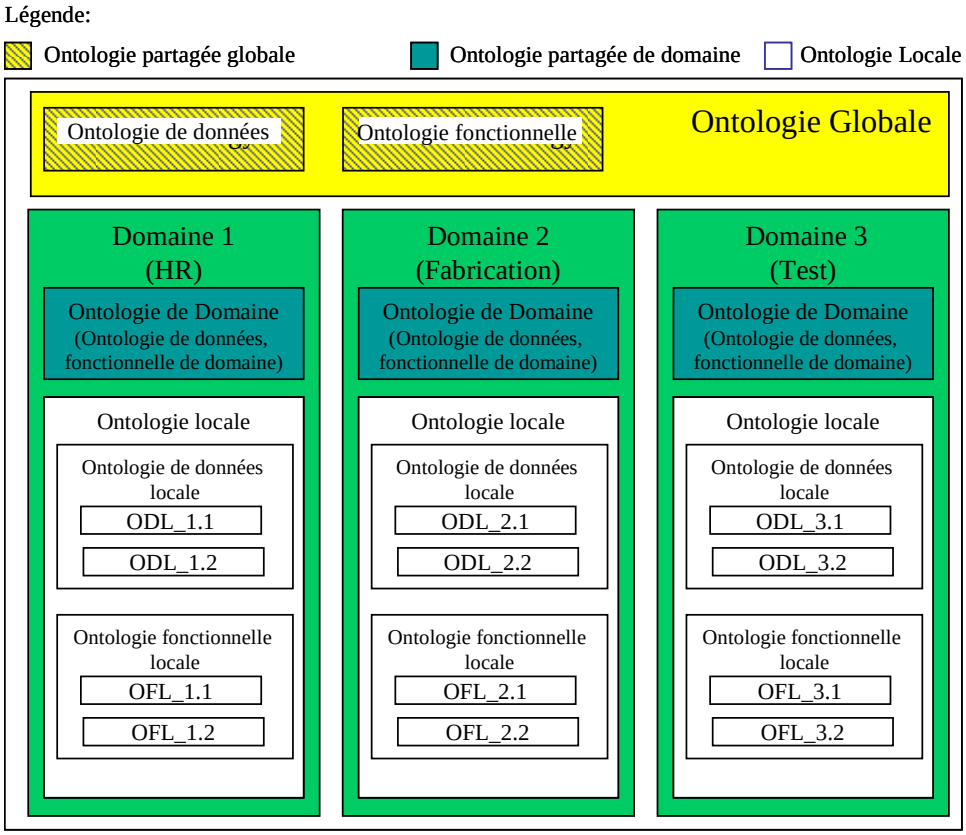

Figure VII.32. Urbanisation sémantique typique

Une caractéristique importante de notre architecture ontologique est la scalabilité. En effet, notre approche nécessite *o(n + m)* mappings d'ontologies, où *n* et *m* dénotent respectivement le nombre d'ontologies locales et le nombre d'ontologies de domaine. Une autre caractéristique importante est la flexibilité. Du fait que notre environnement cible est ouvert et distribué, nous devons alors fournir une architecture flexible afin de prendre en compte de façon efficace le changement et l'évolution des ontologies. Dans notre architecture, la sémantique est passée du niveau local au niveau global via le niveau de domaine (et vice-versa), les problèmes de consistance doivent donc être considérés de deux façons : selon une approche descendante (top-down) et selon une approche ascendante (bottom-up).

Dans une approche d'évolution descendante, nous proposons que les fournisseurs d'ontologies de domaine et de l'ontologie globale soient responsables afin de garantir l'évolution de leurs ontologies et de garantir leur consistance avec les versions précédentes. L'extension de ces ontologies (ontologies de domaine et de l'ontologie globale) avec de nouveaux concepts et de nouvelles propriétés ne cause aucune inconsistance sur les ontologies locales. Cependant, le changement de certains concepts et propriétés existantes peut engendrer certaines incohérences au niveau des ontologies locales. Mais comme les ontologies globales les ontologies de domaine sont construites selon un principe de stabilité relative avec des changements rares ou à la limite des changements limités après une construction initiale, nous pouvons alors considérer que notre architecture ontologique permet généralement de garantir la cohérence et permet ainsi l'évolution ascendante de sémantique avec le temps.

En ce qui concerne, l'évolution ascendante (c-à-d., la situation qui correspond au cas où ce sont les ontologies locales qui évoluent, suite à l'évolution des services fondamentaux), les ontologies locales peuvent alors être ajoutées, supprimées et

modifiées (comme conséquence de l'évolution des services fondamentaux) et cette évolution n'influera pas sur les ontologies de domaine et globales tant que cette évolution ne concerne pas les concepts et les propriétés qui sont mappés avec les ontologies de domaine. Dans le cas où la modification des ontologies locales induit des modifications dans les ontologies de domaine, cette situation peut entraîner des inconsistances dans ces ontologies. Dans ce cas, les ontologies de domaine et globale doivent être alors maintenues en prenant en considération et dans la mesure du possible le principe de compatibilité ascendante (consistance avec les versions précédentes). Mais dans la plupart des cas, ce scénario est généralement inhabituel du fait que les ontologies de domaine et les ontologies globales sont conçues comme des ontologies relativement stables. En conséquence, notre approche de structuration fournit un moyen flexible permettant de capturer de façon efficace la sémantique des services au sein d'entreprises complexes, et en particulier dans le cadre du projet MiMISI. Dans la section suivante, nous allons exposer la démarche méthodologique pour construire notre architecture sémantique.

VII.6. Démarche de construction du modèle sémantique

Jusqu'à présent, les ontologies présentées dans ce chapitre ne sont en fait que des squelettes d'ontologies qu'il faut compléter afin de décrire la sémantique des services d'entreprise. Devant la complexité de cette architecture, il devient alors nécessaire de proposer une démarche méthodologique pouvant guider à mieux construire l'architecture sémantique.

VII.6.1. Approche de construction du modèle sémantique

Comme nous l'avons déjà mentionné dans le chapitre IV, il existe une multitude de méthodologies permettant d'aider dans le processus de construction des ontologies. Ces méthodologies peuvent être fondamentalement classées en deux approches qui sont l'approche ascendante (par exemple, Uschold, Kactus, Cyc, etc.) et l'approche descendante (par exemple, Sensus, Grüninger & Fox, etc.). De plus, les ontologies peuvent être construite à partir du scratch (en faisant table rase) ou elles peuvent être basées sur la réutilisation de certaines ontologies existantes et/ou standard. En ce qui nous concerne, nous avons choisi de mettre en oeuvre une approche hybride qui combine à la fois les principes de construction ascendants et les principes descendants. De façon générale, la construction de nos ontologies se fait à partir du scratch, mais il nous arrive parfois de faire recours à la réutilisation totale ou partielle de certaines ontologies existantes (notamment les ontologies OWL-S, PSL, DOLCE et SUMO). Par exemple, nous avons réutilisé l'ossature de l'ontologie de service OWL-S que nous avons étendue, et nous avons réutilisé les liens de spécialisation de certains concepts d'ontologies génériques existantes comme guide pour la généralisation de certains de nos concepts, nous aidant ainsi à définir des concepts plus génériques réutilisables nécessaires à la construction notamment des ontologies de domaine et de l'ontologie globale. A titre d'exemple, nous nous somme inspirés de la notion d'activités simples et d'activités complexes de PSL pour définir notre ontologie métier. Nous nous sommes également inspirés de la notion d'entité abstraite et de la notion d'entité physique de SUMO pour mieux structurer certaines de nos ontologies fondamentales.

VII.6.2. Démarche globale de construction du modèle sémantique

La démarche que nous proposons pour la construction du modèle sémantique comporte principalement six phases (figure VII.32) :

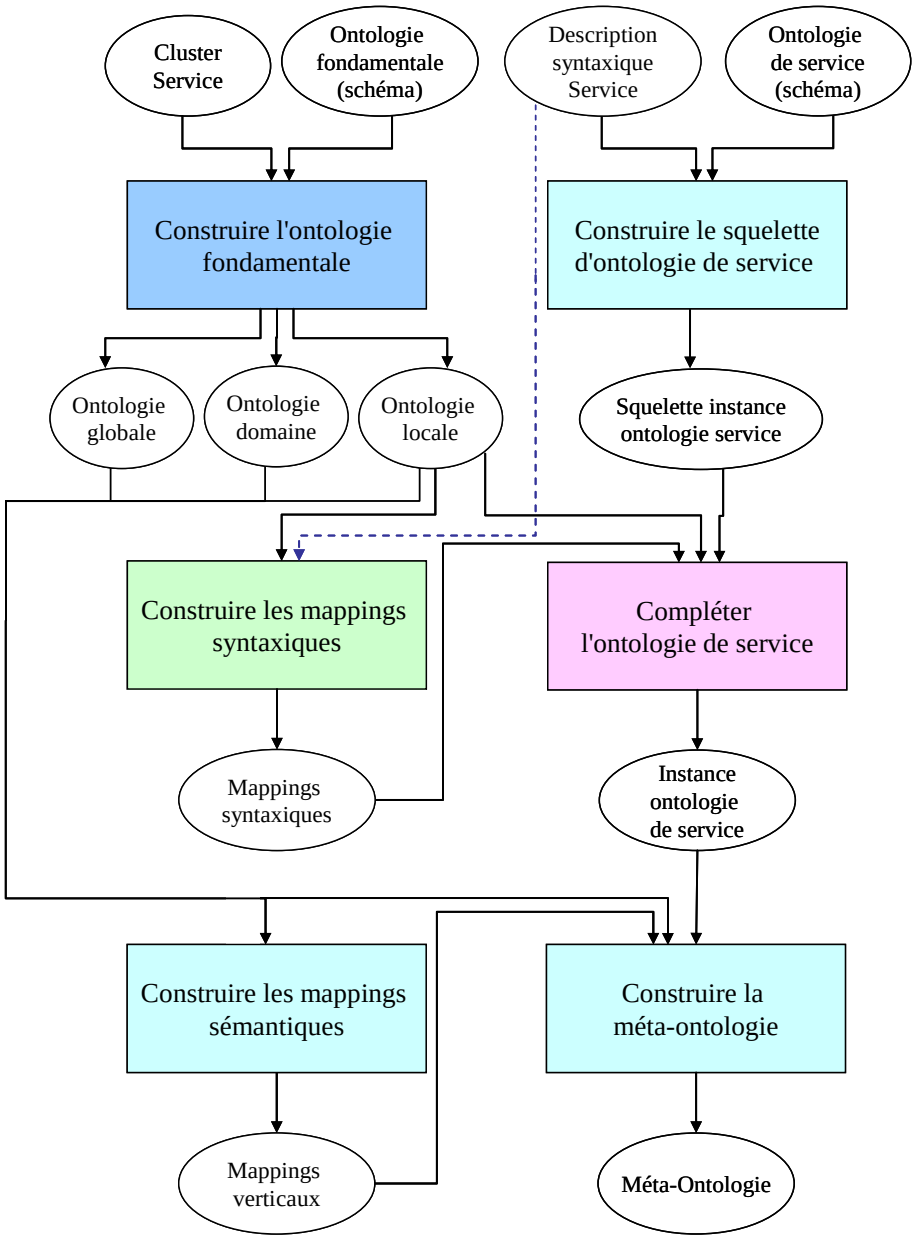

Figure VII.33. Démarche globale de construction du modèle sémantique

- *Construire l'ontologie fondamentale* : elle consiste à définir l'ontologie fondamentale permettant de décrire la sémantique spécifique du système d'information d'entreprise. Le résultat de cette phase est une ontologie fondamentale (cf. § VII.4.2) qui est généralement décomposée en plusieurs sous-ontologies fondamentales qui incluent généralement une ontologie globale, quelques ontologies de domaines et plusieurs ontologies locales.
- *Construire les mappings syntaxiques* : elle consiste à définir les différents liens qui peuvent exister entre nos différentes ontologies locales et les multiples descriptions syntaxiques des services d'entreprise permettant ainsi de définir un pont entre le

domaine sémantique et le domaine syntaxique. Le résultat de cette phase est donc une collection d'ontologies particulières (ontologies de mappings syntaxiques[53]) permettant de décrire les mappings syntaxiques pouvant exister entre les ontologies fondamentales locales et les fichiers de description syntaxique des services (cf. § VII.4.1.1.3).

- *Construire les mappings sémantiques* : elle consiste à définir les différents liens qui peuvent exister entre nos différentes ontologies définissant ainsi une intégration au niveau des ontologies. Le résultat de cette phase est donc une collection d'ontologies particulières (ontologies de mappings) permettant de décrire les mappings sémantiques (cf. § VII.5) pouvant exister entre les ontologies fondamentales.
- *Construire le squelette d'ontologie de service* : elle consiste à définir de façon automatique un squelette de l'ontologie OWL-S+ qui sera ensuite enrichi pour compléter la description sémantique des services d'entreprise. Le résultat de cette phase est donc une partie de l'ontologie de service proposée automatiquement à l'utilisateur.
- *Compléter l'ontologie de service* : elle consiste à enrichir et à compléter de façon interactive le squelette de l'ontologie OWL-S+ (cf. § VII.4.1.2) produit par la phase précédente. Il s'agit en particulier de définir les références liant l'ontologie de service aux ontologies fondamentales. Le résultat de cette phase est donc une ontologie de service complète prête être publiée au sein du registre (ou référentiel) d'ontologies d'entreprise.
- *Construire la méta-ontologie* : elle consiste à construire les différentes méta-ontologies décrivant les multiples ontologies d'entreprise. Le résultat de cette phase est principalement un ensemble de profils d'ontologies (cf. § VII.4.3) qui vont être publiés au sein du référentiel d'ontologies d'entreprise.

VII.6.3. Construire l'ontologie fondamentale

La démarche méthodologique que nous proposons pour la construction de l'ontologie fondamentale comporte principalement deux phases (figure VI.34) : La construction initiale (Initial Building) et la construction courante (Common Building).

Dans la première phase, une approche ascendante est utilisée pour construire les ontologies les plus représentatives. Cette phase inclut trois principales opérations. La première opération (Construction Initiale) construit quelques ontologies locales représentatives (ou initiales) à partir des méta-données collectées sur le système d'information existant (sources de données, applications et les processus), plus précisément sur les quartiers de services d'entreprise. Le choix des quartiers de services est effectué selon des considérations d'exhaustivité et de représentativité. Ensuite, ces ontologies locales sont factorisées dans le sens où elles sont généralisées et/ou combinées avec certains concepts issus de certaines ontologies standard (PSL, SUMO, DOLCE) [Nist 2005][SUMO 2005][Cangemi et al. 2003] dans le but de construire des ontologies de domaine plus réutilisables permettant de service de sémantique partagée au sein des zones de services. Finalement, une autre factorisation est effectuée sur les ontologies de domaine afin de définir l'ontologie globale. A ce stade, nous avons défini une hiérarchie d'ontologies en respectant les principes d'urbanisation sémantique.

[53] Il s'agit réellement d'ontologies de mappings syntaxe-sémantique. Mais pour simplifier, nous les appelons aussi: ontologies de mappings syntaxiques.

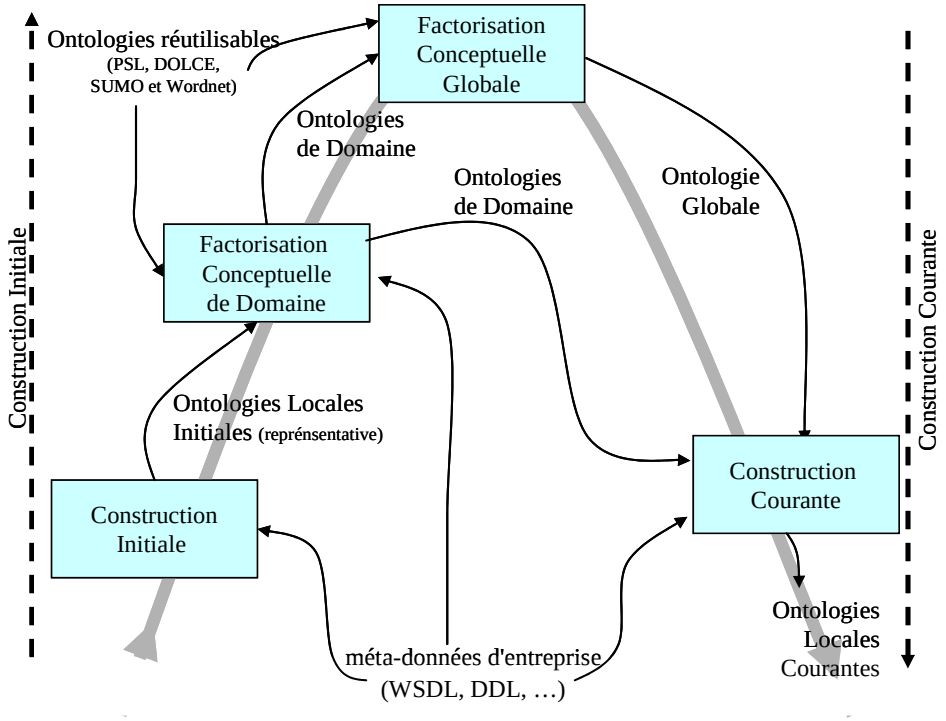

Figure VII.34. Approche de construction des ontologies fondamentales [Izza et al. 2005-d]

La seconde phase (construction courante) concerne la construction des autres (ou nouvelles) ontologies locales après la construction de l'ossature de la hiérarchie. Dans ce cas, l'approche utilisée est généralement descendante, et les nouvelles ontologies locales sont liées aux ontologies de domaines. La figure VII.34 récapitule les principes notre approche de construction des ontologies fondamentales.

En ce qui concerne le langage de représentation des ontologies, nous avons choisi d'utiliser un langage unique qui est OWL pour des raisons de standardisation, d'expressivité, de performance et l'existence de nombreux outils le supportant tels que les outils d'édition (par exemple, Protégé, OntoEdit, etc.) et les moteurs d'inférence (par exemple, Racer, Jena, etc.). En ce qui concerne le choix de sous-langage de OWL, nous avons opté pour OWL-DL, qui constitue le langage le plus connu, le plus complet et décidable parmi les sous-langages de OWL. OWL-DL (§ IV.3.5.4) est considéré comme une notation alternative et équivalente du langage de la logique de description $\mathcal{SHOIN}(\mathcal{D})$. Ceci signifie que OWL-DL permet des mécanismes de raisonnement, qui sont nécessaire pour des tâches de matching et de résolution d'hétérogénéités sémantiques. Même si OWL-DL est souvent considéré comme une solution lourde comparativement à OWL-Lite, il fournit des constructeurs additionnels qui n'existent pas dans OWL-Lite tels que les classes énumérées (owl:oneOf, owl:DataRange) ou les valeurs d'attributs (owl:hasValue) et qui sont nécessaires dans le cadre de notre projet MiMISI.

En ce qui concerne les outils utilisés, nous avons choisi Protégé en tant qu'éditeur d'ontologies fondamentales. Ce choix est dû au fait que Protégé est un outil puissant, open source et compatible avec le langage OWL retenu. De plus, comme les outils de gestion de la sémantique présentent des performances limitées en fonction de la taille des ontologies manipulées, nous avons choisi de limiter la taille de nos ontologies afin d'accroître les performances. A ce titre, nous avons choisi les tailles typiques de schéma

d'ontologie (obtenues par expérimentations sur notre cas d'étude) comme suit : quelques dizaines (idéalement entre 10 et 30) de concepts, quelques dizaines (idéalement entre 10 et 30) de relations et quelques centaines (idéalement entre 100 et 300) propriétés. Par exemple, en ce qui concerne la version 1.0 des ontologies que nous avons construites dans le cadre de l'étude de cas (chapitre IX), nous avons retenu les métriques suivantes :

- l'ontologie locale des ressources humaines présente 23 concepts et 21 relations;
- l'ontologie locale des équipements présente 16 concepts et 19 relations;
- l'ontologie locale de maintenance préventive possède 33 concepts et 39 relations.

Finalement, nous devons mentionner le fait qu'il est important de mettre en place un groupware spécifique pour le développement et la maintenance collaboratives de certaines ontologies fondamentales. Ce groupware qui doit regrouper une équipe d'experts de domaine est utile pour définir de façon correcte et efficace mais aussi pour gérer les multiples versions et l'évolution de ces ontologies. Cette dernière notion (évolution d'ontologies) peut être considérée comme un type spécifique de mapping d'ontologies. Comment gérer l'évolution et les versions d'ontologies constitue un important sujet de recherche qui n'est pas traité dans nos travaux. Nous mentions seulement quelques principes que nous avons mis en oeuvre pour assister l'utilisateur durant ce processus complexe. Nous considérons que nos ontologies sont mappées aux versions précédentes en se basant sur l'utilisation de certains constructeurs de OWL tels que : *owl:versionInfo, owl:priorVersion, owl:backwardCompatibleWith* et *owl:incompatibleWith*.

VII.6.4. Construire les mappings syntaxiques

Cette étape permet la construction des mappings syntaxiques (et plus exactement des mappings syntaxe-sémantique) nécessaires pour définir des correspondances entre les concepts des ontologies locales et les éléments de description syntaxique des services d'entreprise permettant ainsi de relier le domaine sémantique au domaine syntaxique. Rappelons que le domaine syntaxique correspond aux fichiers WSDL qui décrivent les services d'entreprise. Notons que cette phase est importante car elle constitue une des bases de la médiation sémantique que nous allons étudier au chapitre VIII.

La figure VII.35 qui constitue l'ontologie de mappings syntaxique-sémantique, illustre le principe de la construction de ces mappings. Ainsi, on peut distinguer principalement deux types de correspondances entre les ontologies locales et les fichiers XML issus de la description syntaxique WSDL des services d'entreprise:

- correspondance type-concept, permettant de définir des mappings entre les concepts d'ontologies locales et les types de schémas XML,
- correspondance attribut-propriété, permettant de définir des mappings plus fins entre les propriétés d'ontologies locales et les attributs de schémas XML.

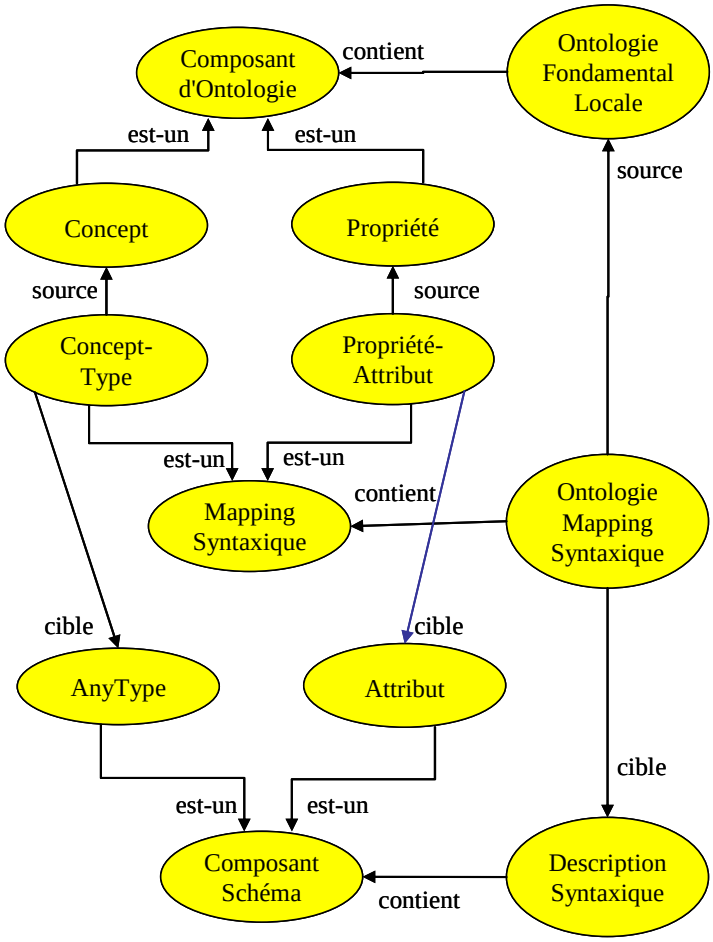

Figure VII.35. Ontologie de mappings syntaxiques

VII.6.5. Construire les mappings sémantiques

Un aspect important qui découle de l'urbanisation ontologique est l'approche de médiation préconisée. Nous distinguons deux types de médiation d'ontologies: durant le design- time et durant le run-time. En design-time, la médiation d'ontologies concerne principalement la définition des mappings (en mode intensionnel) entre nos ontologies fondamentales, tandis qu'en run-time, la médiation est principalement effectuée en mode extensionnel (médiation d'instances). On se focalise dans cette section uniquement sur l'aspect médiation en design-time (le principe de la médiation en run-time est quant à lui traité au chapitre VIII) et nous pouvons résumer cette approche de médiation par l'identification de deux types de mappings qui sont définis verticalement de deux manières: mappings locaux et mappings globaux.

La figure VII.36 qui est un extrait de l'ontologie de mappings sémantiques illustre le principe des mappings sémantiques permettant de lier sémantiquement les concepts d'ontologies fondamentales. Le même raisonnement se fait en ce qui concerne les propriétés des ontologies fondamentales.

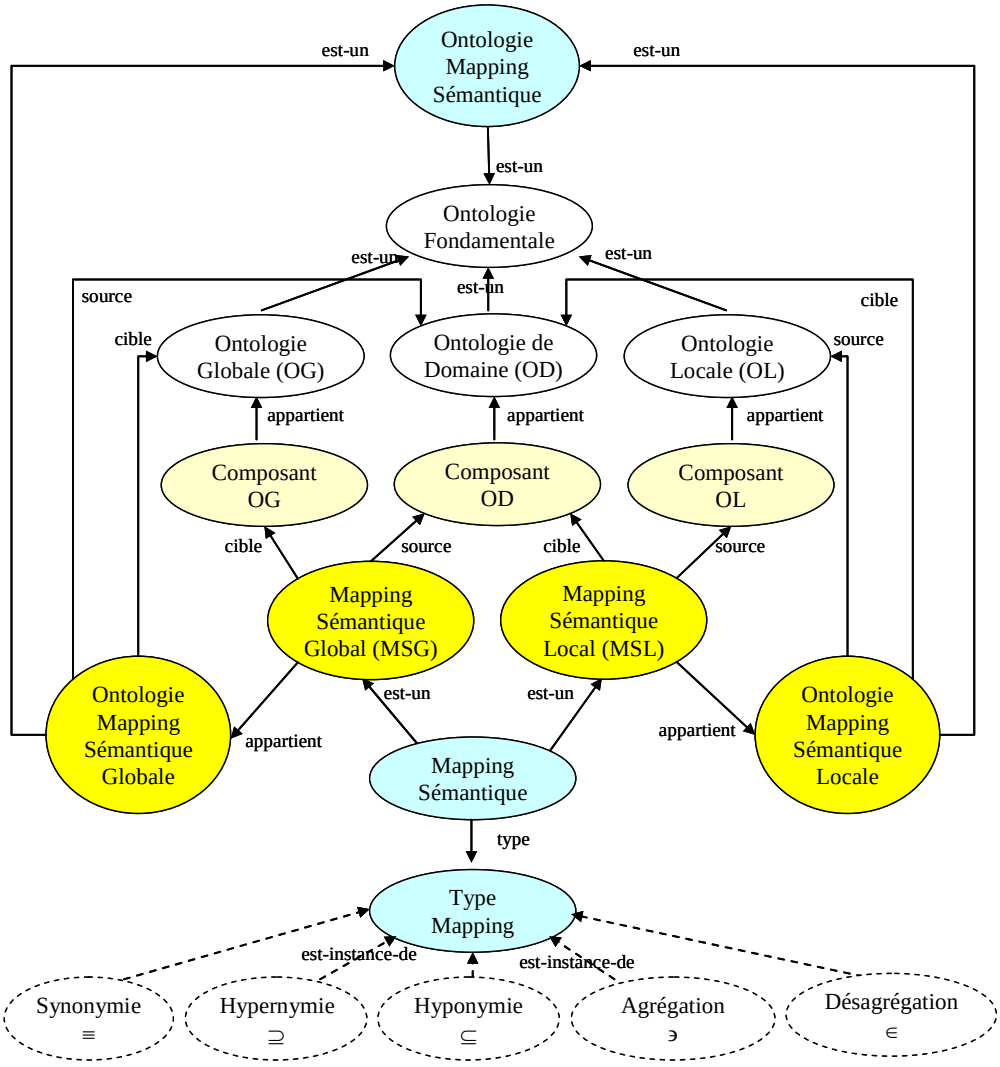

Figure VII.36. Ontologie de mappings sémantiques

Les mappings locaux sont utilisés afin de lier les ontologies locales aux ontologies de domaine, tandis que les mappings globaux permettent de lier les ontologies de domaine à l'ontologie globale. Nous appelons cette approche de médiation, médiation verticale, du fait qu'elle se conforme aux principes hiérarchique de notre approche d'urbanisation d'ontologies. En revanche, les mappings horizontaux (mappings qui sont définis entre les ontologies sœurs: entre les ontologies de même niveau, c-à-d entre les ontologies locales ou entre les ontologies de domaine) sont considérés comme le résultat d'une inférence qui exploite les mappings verticaux en vue de dériver les mappings horizontaux. Toutefois, ces derniers peuvent être calculés aussi bien en design-time qu'en run-time, selon si l'on désire de les stocker ou les recalculer le moment opportun. Dans notre approche, nous avons décidé calculer en design-time tous les mappings qui sont souvent sollicités, et nous exploitons l'orientation run-time en vue d'inférer occasionnellement les mappings correspondant aux besoins ponctuels.

VII.6.6. Construire le squelette de l'ontologie de service

L'ontologie de services OWL-S+ définie précédemment est utilisée pour décrire sémantiquement les services fondamentaux. Cette étape de construction du squelette de

l'ontologie de service consiste à définir certains éléments de l'ontologie de service qui peuvent être obtenus de façon automatique à partir des descriptions syntaxiques des services d'entreprise, et notamment à partir du fichier WSDL. Cette construction va permettre plus précisément de définir de façon automatique :

- la plupart des éléments du profil du service d'entreprise,
- la plupart des éléments du modèle de service,
- et certains éléments du grounding de service.

A cet effet, et dans le but d'assister l'utilisateur, notre approche inclut un traducteur semi-automatique "WSDL2OWL-S+" qui est une extension de celui proposé dans [Semwebcentral 2005] et qui permet d'assister l'utilisateur durant le processus de création de l'ossature de description des services. Cet outil propose une traduction automatique au format OWL-S+ à partir de la description syntaxique WSDL. Cette ossature doit ensuite être complétée, notamment en établissant les liens avec les ontologies fondamentales, ce qui fait l'objet de l'étape suivante.

VII.6.7. Compléter l'ontologie de service

Une fois que l'ossature de l'ontologie de service est construite (à l'issue de l'étape précédente), cette étape vient pour la compléter en définissant tous les aspects restés en suspens et qui nécessitent l'intervention humaine. Aussi, cette étape est essentiellement une étape interactive où l'utilisateur intervient afin de saisir ses choix de description qui vont ainsi compléter la description sémantique des services d'entreprise.

Un aspect majeur de cette étape est l'établissement des correspondances entre les inputs et les outputs avec les ontologies de données locales. Un autre aspect est l'établissement de la liaison entre l'ontologie de service et l'ontologie fonctionnelle et enfin l'établissement des liens entre les paramètres non fonctionnels (cluster, vue, visibilité, …) de OWL-S+ et les ontologies d'entreprise (OWL-E). La figure VII.37 permet d'illustrer le principe de complètement de certains aspects de l'ontologie de service.

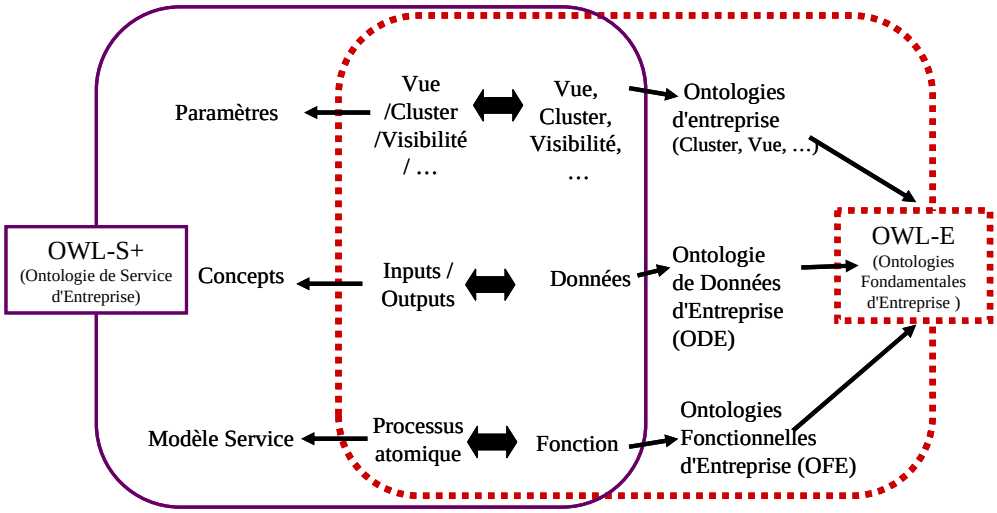

Figure VII.37. Liaisons entre l'ontologie de service OWL-S+ et les ontologies d'entreprise

Un autre aspect important de cette étape est la construction du grounding permettant de définir les mappings syntaxique-sémantique liant ainsi certains concepts de l'ontologie

fondamentale aux types de données définis dans les fichiers WSDL. Traditionnellement cette phase se fait de façon manuelle, et notre approche propose de formaliser davantage la procédure de correspondance et d'offrir une assistance permettant d'aider l'utilisateur dans le processus de construction de l'ontologie grounding de service qui est rappelons-le est une étape complexe, fastidieuse et peu formalisée.

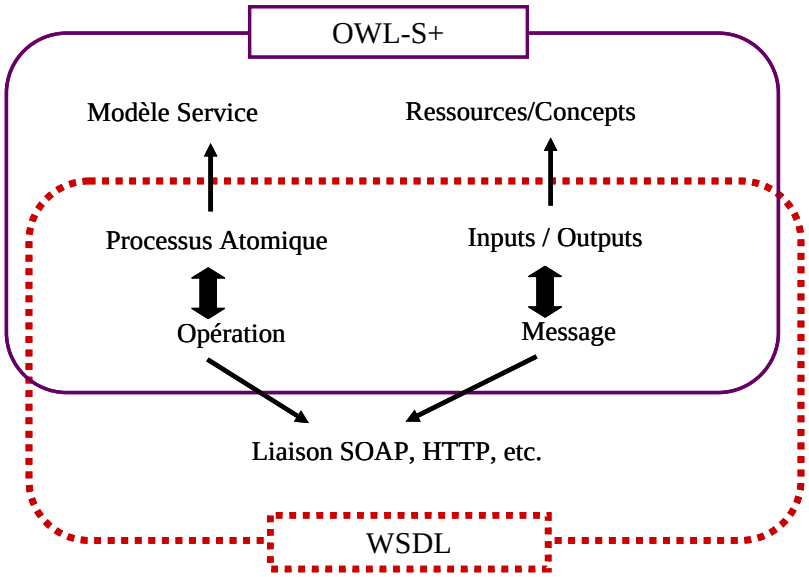

Figure VII.38. Mapping OWL-S+/WSDL

De même que pour l'étape précédente, nous avons construit un outil permettant d'assister l'utilisateur dans ce processus de complètement de la description sémantique. Cet outil qui sera décrit au chapitre IX lors de la description du prototype développé permet d'aider l'utilisateur à remplir le grounding du service.

VII.6.8. Construire la méta-ontologie

Cette dernière étape permet la construction de la méta-ontologie d'entreprise qui contient les méta-données permettant la description des ontologies d'entreprise. Cette méta-ontologie est importante car elle sert de base pour le catalogage et la découverte de ces ontologies. Un aspect majeur de la méta-ontologie est la notion de profil d'ontologie. Ce dernier contient les informations publiables sur ces ontologies. C'est à l'issue de cette étape que les ontologies vont être publiées au sein du référentiel d'ontologies. Ce mécanisme de publication est décrit au chapitre suivant.

VII.7. Conclusion

Nous avons présenté dans ce chapitre une approche de construction de l'architecture sémantique d'entreprise permettant de décrire de façon flexible les services d'entreprise tels que présentés au chapitre VI. Cette construction constitue la réponse à la problématique P_{Sem} qui posait la question de sémantification des services d'entreprise.

Notre approche se base sur l'utilisation de deux types majeurs d'ontologies d'entreprise: les ontologies de service et les ontologies fondamentales. Les ontologies de services, qui

constituent une extension de l'ontologie OWL-S, permettent de décrire sémantiquement les services d'entreprise. En revanche les ontologies fondamentales, qui sont référencées au sein des ontologies de services, permettent de décrire la sémantique spécifique à l'entreprise.

Le processus de sémantification suit une démarche comportant six étapes majeures qui sont : la construction des ontologies fondamentales, la construction des mappings syntaxiques, la construction des mappings sémantiques, la construction du squelette de l'ontologie de service, le complètement de l'ontologie de service et enfin la construction de la méta-ontologie. Un aspect majeur de notre démarche de construction de l'architecture sémantique est l'urbanisation. En effet, nos ontologies d'entreprise sont urbanisées dans le sens où elles sont regroupées en clusters sémantiques. Cette clustérisation permet de construire une architecture plus flexible où les changements au sein d'un cluster impactent dans une moindre mesure les autres clusters. De plus, cette urbanisation permet de définir une cartographie naturelle de la sémantique du système d'information.

La construction de l'architecture sémantique n'est pas une fin en soi. Elle ne constitue qu'une étape de la démarche d'intégration du système d'information d'entreprise. Cet aspect est traité au chapitre suivant qui va présenter comment la sémantique définie tout au long de ce chapitre peut être utilisée pour définir efficacement certains scénarios d'intégration d'applications industrielles.

Chapitre VIII

CONSTRUCTION DE L'ARCHITECTURE D'INTEGRATION D'ENTREPRISE

VIII.1. Introduction

Les deux chapitres précédents (chapitres VI et VII) ont permis de répondre respectivement à la problématique P_{Syn} de construction de l'architecture de services d'entreprise (cf. § V.2.1) et à la problématique P_{Sem} de construction de l'architecture sémantique (cf. § V.2.2) permettant de 'sémantifier' les services d'entreprise. A présent, nous allons nous intéresser dans ce chapitre à la problématique P_{Int} (cf. § V.2.3) qui a pour but de répondre à la question d'intégration sémantique de ces services d'entreprise (figure VIII.1).

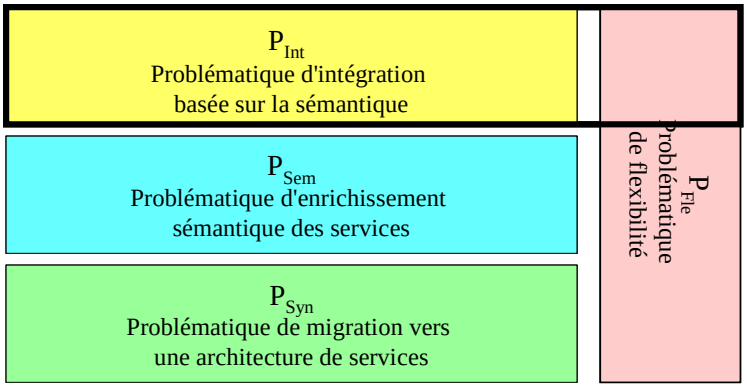

Figure VIII.1. La problématique P_{Int}

L'objectif de ce chapitre consiste à répondre à certaines questions et limites énoncées dans les chapitres IV et V, concernant l'intégration sémantique des services dans le domaine industriel. Il s'agit plus précisément de définir la couche d'intégration (ou architecture d'intégration d'entreprise), c'est à dire les différents services d'intégration permettant d'offrir les activités nécessaires à l'intégration flexible des services

fondamentaux d'entreprise. Rappelons que cette couche d'intégration se base sur la couche sémantique construite au chapitre VII afin d'intégrer les services d'entreprise identifiés au chapitre VI (figure VIII.2[54]).

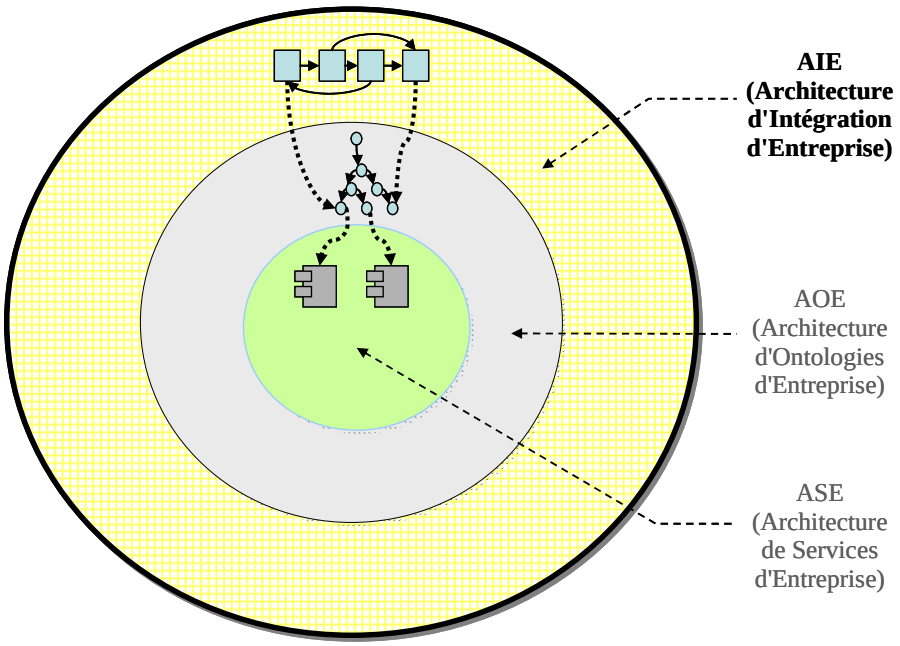

Figure VIII.2. Positionnement de la couche d'intégration (AIE)

L'intérêt principal de ce chapitre est qu'il permet de fournir un cadre global pour la mise en œuvre de l'intégration des services d'entreprise. Comme le domaine d'intégration est assez vaste, nous nous limitons dans notre étude en nous focalisant sur trois aspects fondamentaux qui constituent les principales contributions de ce chapitre et qui sont la publication, la découverte et la médiation de services d'entreprise.

Aussi, dans ce chapitre, nous présenterons une vision globale des principaux éléments de la couche d'intégration. La section VIII.2 exposera les principes majeurs permettant de guider la construction de l'architecture d'intégration d'entreprise. La section VIII.3 présentera de façon générale l'architecture d'intégration qui sera ensuite détaillée en section VIII.4. La section VIII.5 présentera le processus d'intégration et enfin la section VIII.6 décrira un scénario typique d'intégration des services d'entreprise en tenant compte du triptyque intégrateur/ développeur/utilisateur métier.

VIII.2. Principes majeurs

De même que pour la construction de l'architecture de services (cf. § VI.2) et de l'architecture sémantique (cf. § VII.2), la construction de l'architecture d'intégration (AIE) à l'intérieur d'une entreprise doit répondre à plusieurs préoccupations qui portent sur les éléments de l'architecture d'intégration et le processus permettant d'orchestrer ces éléments. Notre approche d'intégration repose principalement sur les deux points majeurs qui sont (figure VIII.3) :

- les *éléments de l'architecture d'intégration* permettant de définir les différents

[54] Cette figure découle directement de la figure V.8.

services d'intégration nécessaires à mettre en œuvre pour permettre l'intégration des services d'entreprise;

- un *processus d'intégration* qui définit le mécanisme permettant d'orchestrer les différents services d'intégration.

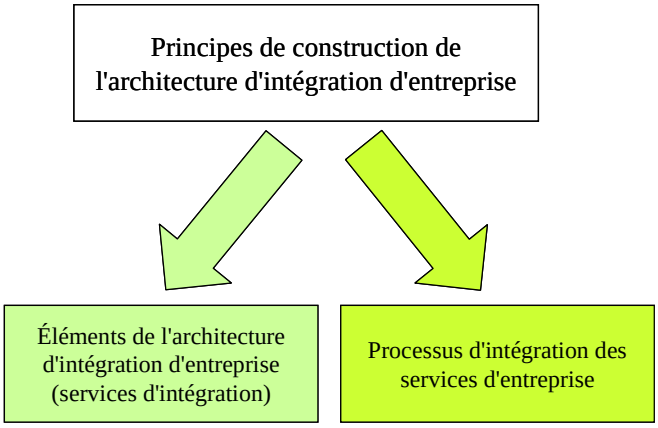

Figure VIII.3. Principes majeurs de construction de l'architecture d'intégration

VIII.3. Description générale de l'AIE

VIII.3.1. Architecture globale de l'AIE

La couche d'intégration (AIE) permet de définir tous les services d'intégration nécessaires pour réaliser l'intégration sémantique des applications. La figure VIII.4, déjà présentée au chapitre V (cf. figure V.7 et V.8), illustre une vue générale de cette architecture qui comprend cinq[55] types de services d'intégration :

- service brokering qui est un service dédié, permettant la gestion du processus d'intégration,
- service de publication permettant la publication sémantique des services fondamentaux,
- service de découverte permettant la découverte sémantique des services fondamentaux,
- service de médiation permettant la résolution des hétérogénéités sémantiques liées aux services fondamentaux,
- et service d'exécution permettant la supervision de l'exécution des services fondamentaux.

La particularité de ces services d'intégration tient au fait que la majorité d'entre eux présente des spécificités de par leur structure et leur comportement. En effet, hormis le service exécution qui demeure relativement générique qui peut être implémenté par les plateformes technologiques industrielles actuelles, tous les autres services d'intégration présentent des spécificités qui sont liées en majeure partie aux spécificités des services d'entreprise et aussi à la spécificité de la sémantique permettant de décrire ces services. De plus, la majorité des services d'intégration que nous proposons intègrent des

[55] Un autre service d'intégration (service de description) qui est destiné au design-time est volontairement omis car il a été étudié lors des chapitres précédents.

mécanismes nouveaux et/ou des mécanismes enrichis afin d'accroître l'efficacité et la flexibilité de l'intégration des services d'entreprise. Avant d'entrer dans le détail de certains services d'intégration, nous allons tout d'abord donner une vision générique du processus d'intégration, exécuté par le service brokering, et qui permet de mettre en musique l'ensemble de ces services d'intégration.

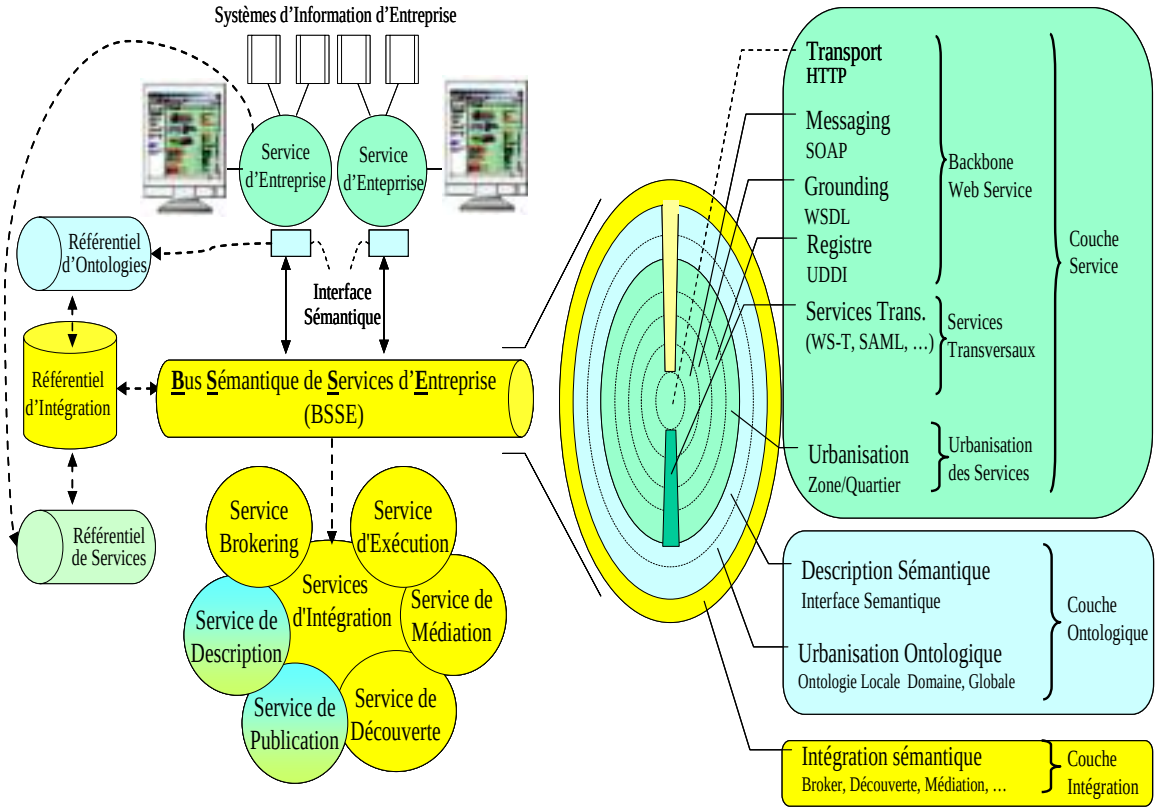

Figure VIII.4. Architecture globale d'intégration [Izza et al. 2005-c]

VIII.3.2. Processus générique d'intégration

Le processus d'intégration est principalement guidé par le service brokering (Brokering Service) qui constitue un nœud d'intégration de notre architecture d'intégration semi centralisée. Ce service a pour but d'orchestrer la plupart des autres services d'intégration tels que le service de découverte (Discovery Services), les services de médiation (Mediation Services) et les services d'exécution (Execution Services). Le scénario générique du processus d'intégration est illustré en figure VIII.5.

Comme le montre la figure VIII.5, le processus d'intégration commence à la suite de la publication sémantique (Semantic Publication) des services d'entreprise dans des référentiels appropriés par les services de publication (Publication Services). Une fois publiés, ces services peuvent être dynamiquement découverts (Semantic Discovery) grâce aux services de découverte (Discovery Services) dans le but de réaliser une tâche donnée formulée sous forme de requête qui correspond à une demande d'un utilisateur ou d'un service d'entreprise relayée par le service brokering (Brokering Services). Les services de découverte (Discovery Services) utilisent des algorithmes particuliers afin d'effectuer la recherche et la sélection de services. Ces algorithmes sont basés sur l'utilisation d'un moteur d'inférence qui exploite les concepts d'ontologies qui décrivent

la requête et les services et qui calculent le rapprochement sémantique entre la requête et les services publiés. Une fois que des services ont été sélectionnés, ils peuvent être connectés dynamiquement grâce au service de médiation (Mediation Services) qui permet de résoudre de façon dynamique et/ou de façon statique les hétérogénéités. Finalement, les services connectables peuvent être exécutés par le service d'exécution (Execution Service). Les services exécutés peuvent éventuellement invoquer le service brokering en formulant une requête, qui peut alors conduire à une autre boucle similaire du processus d'intégration.

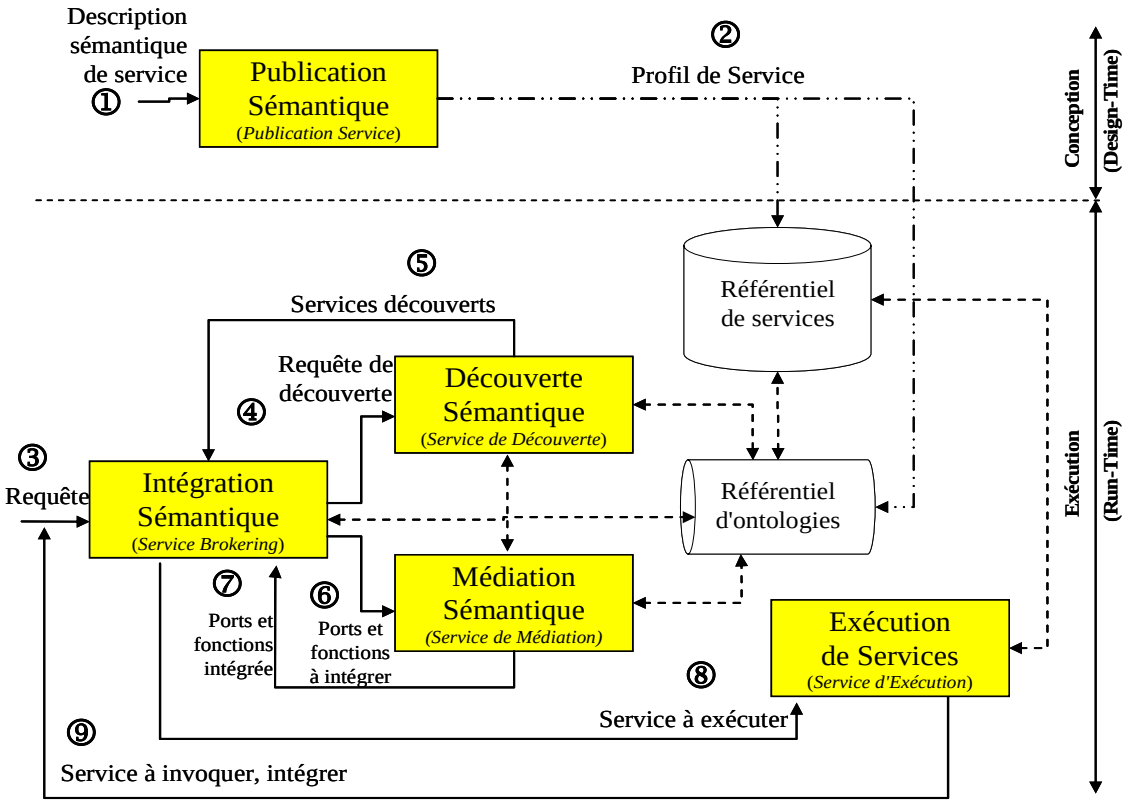

Figure VIII.5. Vision globale du processus d'intégration [Izza et al. 2005-a]

En réalité, il existe trois catégories de services de médiation qui sont :

- service de médiation de données (Data Mediation Services) qui concernent l'intégration sémantique des services d'entreprise au niveau données en permettant de résoudre les hétérogénéités liées aux données;
- service de médiation fonctionnelle (Function Mediation Services) qui concernent l'intégration sémantique des services d'entreprise au niveau fonctions en permettant de résoudre les hétérogénéités d'ordre fonctionnel;
- et enfin le service de médiation de processus (Business Mediation Services) qui concernent l'intégration sémantique des services d'entreprise au niveau du processus permettant ainsi de fournir des mécanismes pour la coordination et l'orchestration des services.

Ces trois types de services de médiation fournissent à notre architecture d'intégration les moyens pour intégrer les applications industrielles à différents niveaux. Ces trois niveaux d'intégration définissent une stratification de la couche d'intégration (cf. figure VIII.4) de manière que chaque niveau inclut ses niveaux inférieurs. La figure VIII.6 extraite de [Izza et al. 2005-a] illustre l'inclusion des trois frameworks d'intégration qui

correspondent à ces trois niveaux d'intégration :

- le framework ODSODI (Ontology-Driven Service-Oriented Data Integration) qui s'intéresse à l'intégration des applications au niveau données (cf. § VIII.3.2.1),
- le framework ODSOFI (Ontology-Driven Service-Oriented Function Integration) qui s'intéresse à l'intégration des applications au niveau fonctions (cf. § VIII.3.2.2),
- et le framework ODSOBI (Ontology-Driven Service-Oriented Business Integration) qui s'intéresse à l'intégration des applications au niveau processus (cf. § VIII.3.2.3).

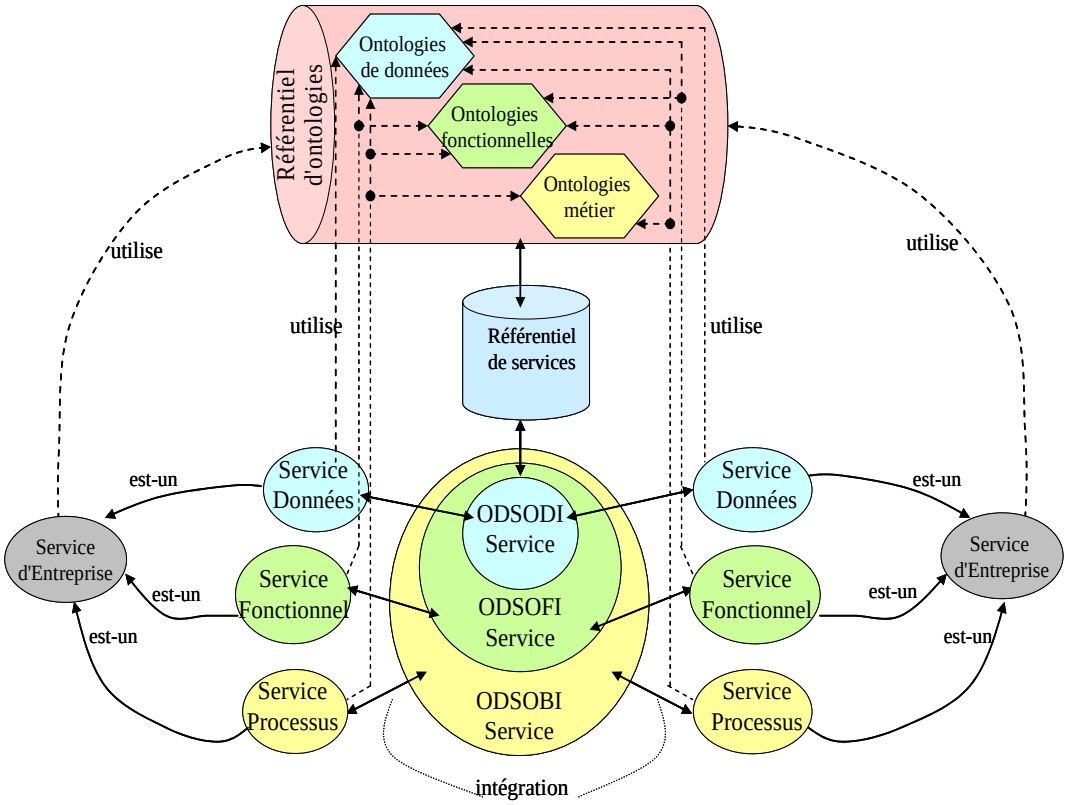

Figure VIII.6. Les trois types d'utilisations typiques du cadre ODSOI [Izza et al. 2006-d]

Quant au tableau VIII.1 (extrait de [Izza et al. 2005-c]), il résume l'articulation des trois couches de notre approche avec ces trois niveaux d'intégration précédemment cités.

Niveau	Couche Services	Couche sémantique	Couche d'intégration
Intégration au niveau données	Services données	Ontologie de données	Service de découverte, service de médiation de données
Intégration au niveau fonctions	Services fonctionnels	Ontologie fonctionnelle	Service de découverte, service de médiation de fonctions
Intégration au niveau processus	Services processus	Ontologie métier	Service de découverte, service de médiation de processus

Tableau VIII.1. Articulation entre les niveaux d'intégration et les couches de notre approche

VIII.3.2.1. Intégration au niveau données

Ce niveau, qui constitue le niveau de base de l'intégration d'applications, facilite l'intégration des applications en adressant la représentation des données dans les différentes applications permettant ainsi des échanges de données tout en conservant leur sens. Les tâches typiques à ce niveau sont principalement la découverte et la médiation sémantique de données. Ces deux tâches exploitent la sémantique définie au chapitre précédent, en particulier les ontologies de données, et fournissent respectivement des mécanismes flexibles pour effectuer la recherche et aussi la transformation des données.

VIII.3.2.2. Intégration au niveau fonctions

Ce niveau d'intégration permet aux applications et aux utilisateurs de chercher, d'invoquer toute fonctionnalité exposée d'une autre application. Les tâches typiques à ce niveau sont principalement la découverte et la médiation de fonctions. Ces deux tâches exploitent la sémantique définie au chapitre précédent, en particulier les ontologies fonctionnelles, et fournissent respectivement des mécanismes flexibles pour effectuer la recherche et aussi la substitution de fonctions.

VIII.3.2.3. Intégration au niveau processus

Ce niveau d'intégration permet aux applications et aux utilisateurs de définir des processus permettant de coordonner et d'orchestrer un ensemble de services d'entreprise. Les tâches typiques à ce niveau sont la découverte et la médiation de processus. Ces deux tâches exploitent la sémantique définie précédemment, en particulier les ontologies métier, et fournissent respectivement des mécanismes flexibles pour effectuer la recherche et la définition des règles d'intégration permettant d'exécuter des processus.

VIII.4. Description détaillée de l'AIE

Après avoir présenté une vision globale de l'architecture AIE, à présent nous allons détailler les principaux éléments, à savoir les principales activités d'intégration et les services d'intégration associés.

VIII.4.1. Intermédiation et gestion du processus d'intégration

Le service brokering constitue le service central de notre architecture d'intégration. Il permet de mettre en œuvre le processus d'intégration des services d'entreprise. Nous allons décrire, dans ce qui suit, les approches classiques d'intermédiation, notre architecture générale du service brokering, les règles d'intégration sur lesquelles il se base pour s'exécuter, la typologie des requêtes qu'il prend en charge et enfin sa logique de fonctionnement général.

VIII.4.1.1. Approches classiques d'intermédiation

Traditionnellement, il existe principalement deux approches possibles pour mettre en œuvre l'intégration de services d'entreprise. Il s'agit de l'approche peer-to-peer et de l'approche centralisée (figure VIII.7). La première approche consiste à définir des

scénarios d'intégration fortement couplés dans le sens où les services peuvent s'appeler mutuellement sans tiers d'intermédiation. La seconde approche repose sur l'utilisation d'un seul nœud d'intégration permettant de jouer le rôle de tiers d'intermédiation entre les services à intégrer. Chacune des approches possède des avantages et des inconvénients. La première est une solution distribuée mais présente une certaine complexité dans la gestion et la maintenance des services. De plus le processus d'intégration est dispersé au sein de la collection de services. La seconde approche est plus découplée mais présente l'inconvénient de tout centraliser au sein d'un nœud d'intégration. L'approche que nous proposons est hybride dans le sens où elle est semi centralisée, reprenant ainsi les avantages des deux approches.

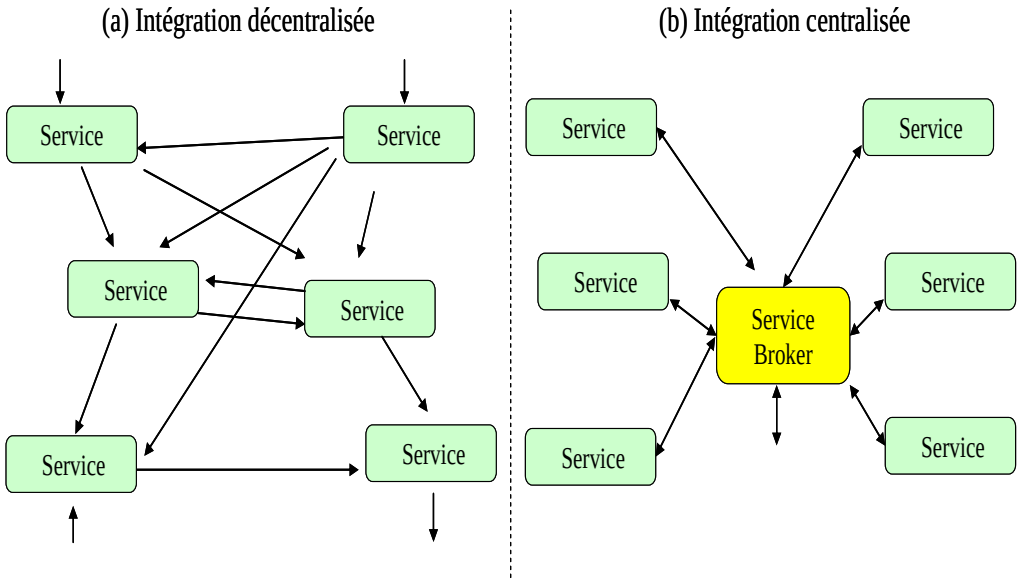

Figure VIII.7. Approches traditionnelles d'intégration de services

VIII.4.1.2. Approche ODSOI d'intermédiation

VIII.4.1.2.1. Architecture générale des services brokering

Notre approche d'intégration repose sur une multitude de nœuds d'intégration que nous appelons les services brokering. Ces derniers constituent les principaux services d'intégration en charge du processus d'intégration des services d'entreprise. Ils permettent de superviser le processus d'intégration en orchestrant un certain nombre de services d'intégration, notamment des services de découverte, de médiation et d'exécution.

Chaque service brokering expose les différents services qu'il comporte. Les services brokering sont reliés entre eux et chaque service brokering peut être exposé comme un service particulier du service brokering de plus haut niveau. Ceci conduit à une architecture hiérarchique de services brokering (figure VIII.8).

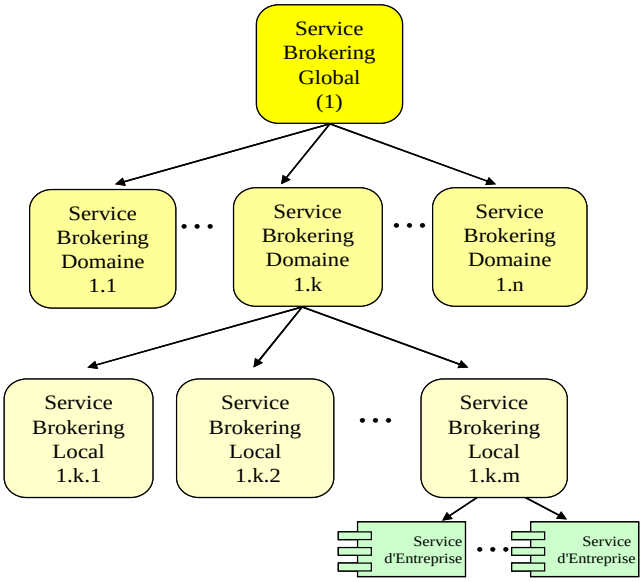

Figure VIII.8. Architecture des services brokering

Chaque service brokering possède une structure générique comprenant un référentiel d'intégration qui inclut un référentiel d'ontologies et un référentiel de services. La figure VIII.9 illustre le principe générique de structuration de ces services.

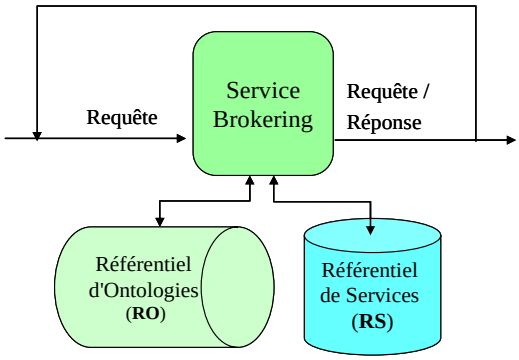

Figure VIII.9.Structure d'un service brokering

VIII.4.1.2.2. Règles d'intégration

En réalité, le service brokering se comporte comme un moteur d'inférence qui exploite un certain nombre de règles appelées: règles d'intégration. Ces règles contiennent toute la logique de fonctionnement de ces services d'intégration, logique qui peut par ailleurs servir à formaliser certains aspects du processus d'intégration d'entreprise. L'exécution d'une règle se traduit par l'exécution d'un certain nombre d'actions d'intégration qui forment le corps de la règle et qui permettent de faire transiter le service brokering d'un état cohérent vers un autre état cohérent. Le service brokering peut recevoir des stimuli qui formalisent des requêtes et qui peuvent parvenir de l'utilisateur ou des autres services d'entreprise (services fondamentaux ou services d'intégration). En réponse à un stimulus, le service brokering cherche puis exécute les règles d'intégration appropriées. Les règles exécutées sont généralement de la forme :

$$(IR_i^j) : (State_i, Condition_i^j) \mapsto (State_j, Action_i^j)$$

que nous noterons avec la forme plus compacte équivalente suivante :

$$(IR_i^j) : \frac{(State_i, Condition_i^j)}{(State_j, Action_i^j)}$$

où $State_i$, $State_j$ sont respectivement l'état source et l'état cible du service brokering,

$Condition_i^j$ est une condition booléenne permettant de spécifier la règle de transition du service brokering de l'état source à l'état cible,

$Action_i^j$ est une action d'intégration qu'il faut exécuter pour pouvoir faire passer le service brokering de l'état source à l'état cible.

Parmi les actions d'intégration que nous avons définies et qui peuvent être exécutées lors de l'exécution d'une règle d'intégration, nous pouvons citer [56] :

- l'action *#Discovery* : qui permet de lancer le service de découverte afin de chercher certains services qui correspondent à certains critères bien identifiés;
- l'action *#Mediate* : qui permet de lancer le service de médiation pour effectuer une tâche de médiation, par exemple la médiation de données ou de fonctions;
- l'action *#Connectable* : qui peut être considérée comme un cas particulier de l'action *#Mediate* et qui permet de calculer le degré de connectivité de deux services afin de savoir si les deux services peuvent être connectés l'un à l'autre;
- l'action *#Compatible* : qui peut être considérée comme un cas particulier de l'action *#Mediate* et qui permet de calculer le degré de substitution (compatibilité) de deux services afin de savoir si les deux services peuvent être mutuellement substitués pour une fonctionnalité bien identifiée;
- l'action *#Integrate* : qui permet de lancer le (éventuellement un autre) service brokering en lui transmettant une requête et qui permet éventuellement de définir des liaisons entre les différents nœuds d'intégration;
- l'action *#Invoke* : qui permet de lancer le service d'exécution pour superviser l'exécution d'un service bien identifié;
- l'action *#Restitute* : qui permet de restituer la réponse au client du service brokering qui peut être un utilisateur ou un service d'entreprise.

Un exemple typique de règle d'intégration est celle qui va permettre d'invoquer le service de découverte à la réception d'une requête de recherche de services. Cette règle peut s'écrire:

$$(IR_0^0) : \frac{(\#\,All, Type(request) = \#\,Discover)}{(\#\,All, \#\,Invoke(\#\,DiscoveryService, request)}$$

Elle stipule que dès réception d'une requête de type découverte (*#Discover*) de service, et quel que soit l'état (*#All*) du service brokering, la requête est redirigée au service de découverte (*#DiscoveryService*).

[56] Bien entendu, nous omettons de mentionner les actions de design-time telles que :
- l'action *#Describe* : qui permet de lancer le service de description afin de décrire sémantiquement un service donnée;
- l'action *#Publish* : qui permet de lancer le service de publication afin de publier syntaxiquement et/ou sémantiquement un service donnée;

VIII.4.1.2.3. Typologie des requêtes

Les requêtes qui peuvent déclencher le service brokering sont multiples. Nous avons classifié les principales requêtes selon la taxonomie de la figure VIII.10. Nous distinguons les types[57] suivants :

- Les requêtes de découverte de services qui portent sur la recherche d'un certain nombre de services en fonction d'un certain nombre de critères qui peuvent porter sur le cluster, la vue d'entreprise, la visibilité, les entrées, les sorties, etc. et qui sont généralement de la forme :

  $$request(\#Discover, \#Service, argument)$$

  où le paramètre *arguments* spécifie la requête de services.

- Les requêtes de découverte de données qui sont des cas particuliers de la requête précédente, qui portent sur la recherche d'un certain nombre de données (services de données) et qui sont généralement de la forme :

  $$request(\#Discover, \#Data, arguments)$$

  où le paramètre *arguments* spécifie les données à chercher.

- Les requêtes de découverte de fonctions qui sont des cas particuliers de la requête de découverte de services, qui portent sur la recherche d'un certain nombre de fonctions et qui sont généralement de la forme :

  $$request(\#Discover, \#Function, arguments)$$

  où *arguments* spécifie respectivement les caractéristiques de la fonction à chercher.

- Les requêtes de médiation de données qui portent sur la médiation de données et qui sont de la forme :

  $$request(\#DataMediate, service_i, service_j, port_i, port_j)$$

  où $service_i$ *et* $service_j$ sont les services concernés par la médiation et $port_i$ *et* $port_j$ sont les ports respectivement des $service_i$ *et* $service_j$ et qui sont impliqués dans la médiation.

- Les requêtes de médiation de fonctions qui portent sur la substitution de services et qui sont de la forme :

  $$request(\#FunctionMediate, service_i, service_j, function_i, function_j)$$

  où $service_i$ *et* $service_j$ sont les services concernés par la médiation fonctionnelle et $function_i$ *et* $function_j$ sont les fonctions respectivement des $service_i$ *et* $service_j$ et qui sont impliquées dans la médiation.

- Les requêtes de connectivité de services qui portent sur la possibilité de connecter

[57] Bien que le service brokering puisse être concerné par des requêtes de design-time (telles que celles portant sur la description et la publication des services par exemple), nous avons choisi de nous focaliser lors de l'étude de ce service d'intégration uniquement sur les requêtes de run-time. Notons que certaines requêtes de design-time ont été déjà étudiées dans les deux chapitres précédents.

deux services quelconques (syntaxiquement ou sémantiquement) et qui sont de la forme :

$$request(\#Connectable, service_i, service_j, port_i)$$

- Les requêtes d'invocation de services qui portent sur l'exécution d'un service donné et qui sont de la forme :

$$request(\#Invoke, service_i, arguments)$$

où $service_i$ est le service à invoquer et *arguments* constitue les paramètres d'invocation.

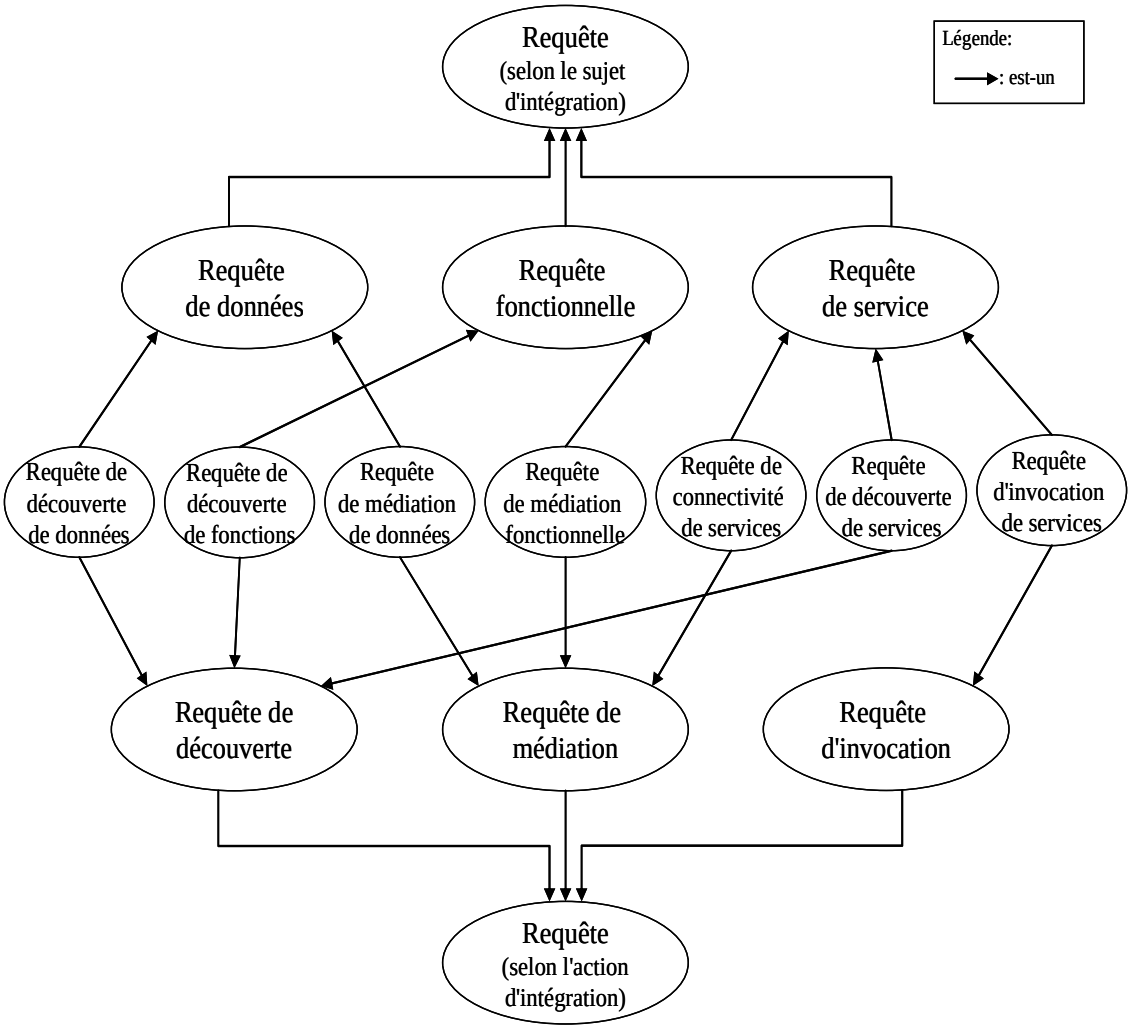

Figure VIII.10. Taxonomie des principales requêtes

VIII.4.1.2.4. Fonctionnement du service brokering

La logique de fonctionnement du service brokering peut être résumée par l'algorithme suivant :

Algorithme général du processus d'intégration

```
1.  TANT QUE vrai
2.      Lire la requête reçue de forme générale: Request(#Action, arguments)
3.      Chercher dans la base de règles du service brokering, l'ensemble (R) des règles exécutables
4.      TANT QUE R <> ∅
5.          Exécuter la règle (l'action d'intégration et les états de transition)
6.      FIN TANT QUE
7.      Restituer la réponse
8.  FIN TANT QUE
```

Figure VIII.11. Logique de fonctionnement du service brokering

Comme le montre cet algorithme, ce service s'exécute selon le mode serveur en effectuant une boucle infinie qui permet ainsi de répondre aux requêtes qu'il reçoit. A chaque fois qu'il reçoit une requête, il consulte sa base de connaissance et ensuite il déclenche la ou les règles appropriées qui correspondent au type de ces requêtes.

VIII.4.2. Publication de services d'entreprise

VIII.4.2.1. Approches classiques de publication

A l'issue de la description syntaxique et/ou sémantique des services d'entreprise, ces derniers doivent être publiés afin qu'ils puissent être éventuellement découverts et utilisés par des acteurs internes et/ou externes à l'entreprise. Traditionnellement W3C [W3C 2006] identifie deux principales approches de publication qui sont :

- l'approche basée sur un référentiel unique qui permet de centraliser l'ensemble des descriptions au sein d'un même référentiel,
- l'approche peer-to-peer qui se base sur l'utilisation d'un ensemble de référentiels permettant ainsi de distribuer les descriptions sur un ensemble de nœuds.

VIII.4.2.2. Approche ODSOI de publication

Notre approche est considérée comme une combinaison de ces deux approches dans le sens où nous proposons une approche hiérarchique de publication, qui se base sur une hiérarchie de nœuds d'intégration (services brokering). De plus, nous proposons une approche flexible basée sur l'utilisation de deux niveaux de référentiels : niveau logique et un niveau physique (cf. § VI.4.6), où chaque référentiel de niveau physique peut implémenter un ou plusieurs référentiels de niveau logique. Cet ensemble de référentiels physiques constitue le référentiel d'intégration de la figure VIII.4.

La publication des services fondamentaux se base sur les descriptions syntaxiques et/ou descriptions sémantiques définies respectivement aux chapitres VI et VII. Notre approche de publication se base sur une extension sémantique des référentiels UDDI privés. Notre approche diffère de celle proposée par la coalition OWL-S qui se base uniquement sur des descriptions sémantiques. Notre approche de publication est stratifiée et organisée en couches afin d'apporter le maximum de flexibilité. Dans notre cas, les services d'entreprise sont décrits et publiés syntaxiquement dans des référentiels

de services avant d'être éventuellement décrits puis publiés sémantiquement dans des référentiels sémantiques. Cette technique qui est plus réaliste apporte plus de flexibilité en permettant ainsi à certains services d'entreprise d'exister sans description sémantique. Nous appelons ce principe qui recommande l'utilisation à la fois des descriptions syntaxiques et/ou sémantiques le principe de *cohabitation syntaxique-sémantique*. Ce dernier est utile si l'on considère que le processus de description sémantique des services est long et nécessite un cycle incrémental et itératif durant lequel des services peuvent exister avec différents de degrés de maturité (degré 1: service syntaxique, degré 2: service sémantique) sans nécessairement disposer de description sémantique.

Logiquement, chaque service brokering dispose d'un référentiel d'intégration comprenant deux sous-référentiels : un référentiel syntaxique (ou référentiel de services) (cf. § VI.4.6) et un référentiel sémantique (permettant de stocker les ontologies d'entreprise (cf. § VII.3.2), notamment l'ontologie OWL-S+ (cf. § VII.4.1)). Les services brokering locaux disposent d'un référentiel local qui sert à publier les descriptions des services fondamentaux qui sont nécessaires pour une utilisation locale. Les services brokering locaux sont publiés par les services brokering de domaine (Domain Brokering Services). Ces derniers sont également publiés par le service brokering global (Global Brokering Service).

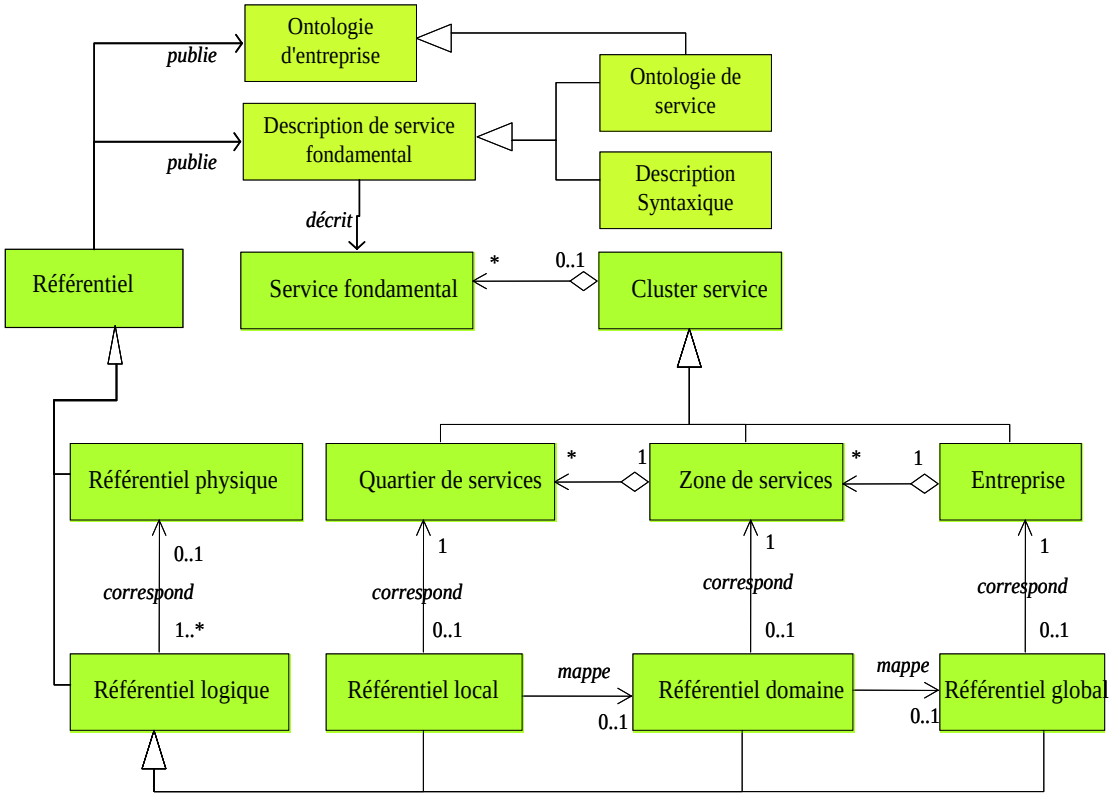

Figure VIII.12. Référentiels logiques et référentiels physiques

Le processus de publication est pris en charge par le service de publication (Publication Service). Ce dernier est un service particulier qui s'active dès réception d'une requête de type :

$$request(\#Publish, PublicationType, Publication)$$

où *PublicationType* désigne le type de publication (publication syntaxique de service, publication sémantique de service, publication d'ontologie fondamentale), et où

Publication désigne les caractéristiques (sous forme de fichier XML ou OWL) à publier.

Le service de publication étend la publication traditionnelle en permettant la publication de certaines ontologies grâce aux deux fonctionnalités (API) que nous avons développées et qui s'appellent *PublishFundamentalService* et *PublishFundamentalOntology* et qui permettent respectivement de publier une description sémantique d'un service ou de publier une ontologie fondamentale.

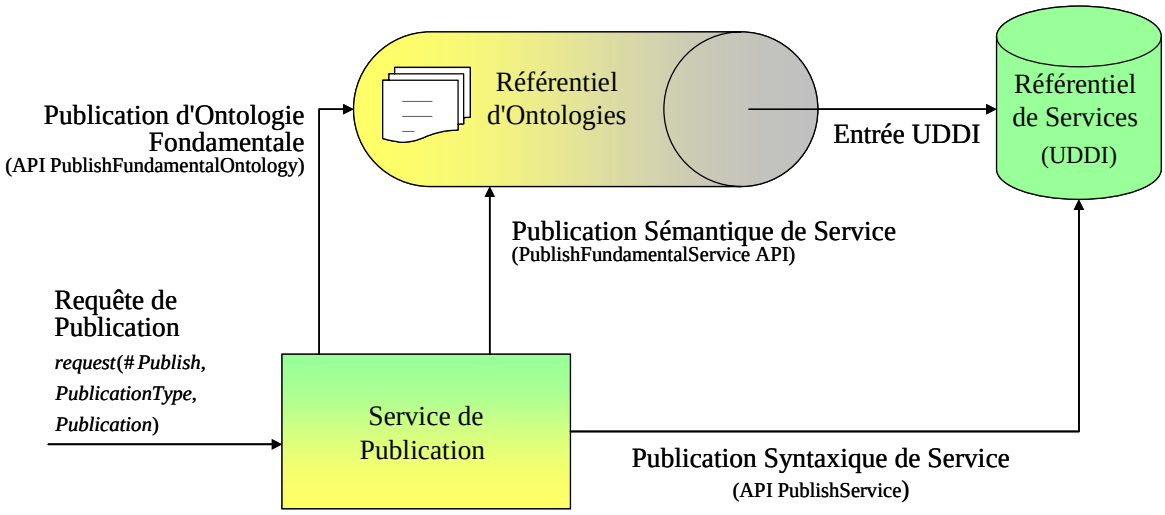

Figure VIII.13. Principe de fonctionnement du service de publication [Izza et al. 2006-a]

La publication d'un service fondamental est effectuée comme suit: les descriptions syntaxiques WSDL sont publiées dans un référentiel privé UDDI de services en utilisant les API standard *PublishService* (cf. § VI.4.6) puis publie les descriptions sémantiques (OWL-S+) qui constituent des descriptions sémantiques complétées par des références à l'UDDI et qui sont publiées dans un référentiel d'ontologies en utilisant l'API *PublishFundamentalService*. La figure VIII.13 récapitule le principe de fonctionnement du service de publication.

VIII.4.3. Découverte de services d'entreprise

VIII.4.3.1. Approches classiques de découverte

Une fois que les services fondamentaux sont publiés à l'aide du service de publication précédemment décrit, des requêtes de découverte peuvent alors être formulées afin de chercher des services d'entreprise selon certains critères bien définis.

Différentes approches ont été proposées dans la littérature pour réaliser la découverte de services [Bernstein et Klein 2002][Chakraborty et al. 2001][Gonzàlez-Castillo et al. 2001] [Paolucci et al. 2002][Benatallah et al. 2003]. Toutes ces approches se basent sur une découverte approximative de services dès lors qu'il n'est pas réaliste de localiser le service qui correspond exactement aux besoins spécifiés. Ces approches diffèrent principalement par le langage de description utilisé (OWL-S, logique de description, etc.) et/ou par l'algorithme de découverte mis en œuvre en vue de comparer la requête et les services (matchmaking [Paolucci et al. 2002], test de subsumption [Gonzàlez-Castillo et al. 2001], mécanisme de réécriture [Benatallah et al. 2003], etc.).

Dans l'approche OWL-S sur laquelle nous nous focalisons dans nos travaux, la

découverte de services se base sur l'utilisation de quatre paramètres qui sont : les inputs, les outputs, les pré-conditions et les effets (IOPEs) [KnowledgeWeb, 2004]. Les travaux les plus importants portant sur les algorithmes de découverte sont principalement ceux décrits dans [Paolucci et al. 2002] et [Li et Horrocks 2004] et qui permettent de calculer dans quelle mesure un service peut répondre à une requête donnée d'un client. Mais ces algorithmes demeurent limités dans le sens où ils n'utilisent principalement que les inputs et les outputs ignorant très souvent les autres paramètres, notamment les pré-conditions et les effets.

Afin de mieux illustrer le principe de fonctionnement de ces algorithmes, nous allons décrire en particulier celui de [Paolucci et al. 2002] qui constitue l'approche la plus importante utilisée actuellement dans la découverte de services OWL-S. Cet algorithme se base sur les propositions du système LARKS [Sycara et al. 2002] et utilise la relation de subsomption du langage OWL. L'idée de base est de chercher l'ensemble des annonces utilisables par le client. Pour ce faire, l'algorithme utilise la signature de l'annonce qu'il compare à la signature de la requête de telle sorte que :

$$(In_{provider} \subseteq In_{requester}) \wedge (Out_{requester} \subseteq Out_{provider})$$

où $\subseteq$ désigne l'opérateur de subsomption, $In_{provider}$ constitue l'ensemble des paramètres d'input du service offert, $In_{requester}$ constitue l'ensemble des paramètres d'input du service demandé, $Out_{provider}$ constitue l'ensemble des paramètres d'output du service offert, $Out_{requester}$ constitue l'ensemble des paramètres d'output du service demandé.

L'algorithme de comparaison repose donc sur la comparaison des outputs et des inputs des services requête (requêtes) et des services publiés (annonces). Tout d'abord il compare les outputs de la requête avec tous les profils de services existants $(Out_{requester} \subseteq Out_{provider})$. Une correspondance est reconnue si et seulement si pour chaque paramètre de la requête, il existe une correspondance dans le profil de service. Sinon la comparaison échoue. Pour les inputs l'ordre de prise en compte est inversé $(In_{provider} \subseteq In_{requester})$. Afin d'effectuer une découverte flexible, quatre degrés de correspondance entre les concepts ont été définis. Ils sont décrits ci-dessous, selon le degré décroissant, dans le cas de la comparaison des outputs :

- EXACT : le résultat de la comparaison est EXACT dans deux cas :
  - si l'output de la requête est équivalent à l'output de l'annonce,
  - si l'output de la requête est une sous-classe immédiate de l'output de l'annonce.
- PLUG-IN : le résultat de la comparaison est PLUG-IN dans le cas où l'output de l'annonce subsume l'output de la requête,
- SUBSUMES : le résultat de la comparaison est SUBSUMES dans le cas où l'output de la requête subsume l'output de l'annonce,
- FAIL : le résultat de la comparaison est FAIL dans le cas où il n'existe pas de relation de subsomption entre l'output de la requête et l'output de l'annonce.

Comme on peut le constater, cet algorithme ne prend en compte que la signature des services : ses inputs et ses outputs. Or la signature ne semble pas être un élément suffisant pour identifier un service. En effet, deux services peuvent présenter la même signature sans être similaires. Des éléments tels que le contexte, les pré-conditions, les effets permettent de distinguer davantage les services. C'est dans ce but par exemple que l'algorithme de base précédent a été amélioré de le cadre des travaux de [Srinivasan et

al. 2006] afin de prendre en compte davantage de paramètres supplémentaires tels que la classification du produit de service et la classification de service.

Dans nos travaux, nous nous basons principalement sur l'algorithme de [Paolucci et al. 2002] et nous proposons une extension afin de prendre en compte nos spécificités, notamment les différents éléments ajoutés ainsi que la prise en compte de mappings sémantique qui font carence dans l'approche actuelle de OWL-S. Pour cela, nous proposons une approche plus complète pour la découverte de services d'entreprise. Cet algorithme qui constitue le cœur du service de découverte est décrit dans ce qui suit.

VIII.4.3.2. Approche ODSOI de découverte

VIII.4.3.2.1. Algorithme de comparaison

L'algorithme de comparaison que nous proposons et qui constitue le cœur du service de découverte est principalement caractérisé par le fait qu'il exploite en plus des aspects traditionnels (les inputs et les outputs d'un service) un certain nombre d'autres aspects qui sont importants dans le cadre de notre approche d'intégration, et qui sont :

- le *contexte de service* : la requête et l'annonce doivent partager le même domaine. Aussi le filtre portant sur le contexte calcule les distances entre les deux services en se basant sur les paramètres
    - de classification de service,
    - de cluster de service,
    - et aussi la notion de vue d'entreprise de service

- les *contraintes de service* : la requête et l'annonce doivent présenter des contraintes compatibles qui portent sur les pré-conditions et les résultats de la requête et de l'annonce et qui s'expriment généralement[58] sous la forme : $(precond_{requester} \Rightarrow precond_{provider}) \wedge (postcond_{provider} \Rightarrow postcond_{requester})$ où $precond_{requester}$ et $postcond_{requester}$ sont les pré-conditions et les post-conditions de la requête, et $precond_{provider}$ et $postcond_{provider}$ sont les pré-conditions et les post-conditions de l'annonce.

- les paramètres *non fonctionnels* tels que la qualité de service et la visibilité de service :
    - la *qualité du service* : la requête et l'annonce doivent présenter des niveaux de qualité de service comparables : $QoS_{requester} \supseteq QoS_{provider}$ où $QoS_{requester}$ et $QoS_{provider}$ représentent respectivement la qualité de service de la requête et celle de l'annonce.
    - la *visibilité de service* : le profil du client ayant effectué la requête doit être compatible avec le type de visibilité de l'annonce : $visibility_{requester} \supseteq visibility_{provider}$ où $visibility_{requester}$ et $visibility_{provider}$ représentent respectivement la qualité de service de la requête et celle de l'annonce.

[58] Notons que l'implication logique n'est pas décidable pour la logique de premier ordre, aussi on se base généralement sur une relation plus faible que l'implication logique.

Notre algorithme de comparaison (matching) est donc décomposé en plusieurs étapes (figure VIII.14):

- (i) la comparaison du contexte (clusters, vues et type)
- (ii) la comparaison des éléments de la signature (les inputs et les outputs)
- (iii) la comparaison des contraintes de fonctionnement du service (les pré-conditions et les post-conditions),
- (iv) et la comparaison des éléments non fonctionnels (la visibilité et la qualité).

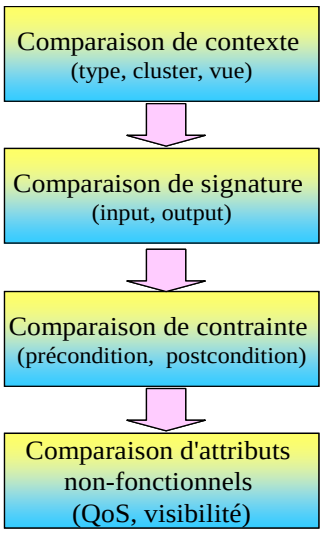

Figure VIII.14. Principe de l'algorithme de comparaison

Dans notre approche, tous les aspects de OWL-S+ qui interviennent dans l'algorithme de comparaison sont des éléments d'ontologies (issues de l'architecture sémantique du chapitre VII) et sont gérés grâce à des mécanismes de raisonnement basés sur la logique de description (OWL-DL).

Notre approche se base principalement sur le calcul d'un indice de similarité qui mesure le degré de satisfaction évalué grâce à la similarité des concepts mesurée à l'aide des mécanismes d'inférence. En parvenant à calculer cet indice de similarité entre deux services ou entre l'annonce et la requête nous pourrons déterminer un tri des réponses possibles.

Cet indice de satisfaction se base sur des indices plus élémentaires portant principalement sur l'inférence entre deux concepts d'ontologies A et B et qui peut revêtir l'une des valeurs suivantes :

- *équivalence* : elle est notée par $\equiv$, et l'expression A $\equiv$ B signifie que les deux concepts A et B ne représentent qu'un seul et même concept: $(A \equiv B) \Leftrightarrow ((A \subseteq B) \wedge (B \subseteq A))$,

- *généralisation* ou subsomption, elle est notée par $\supseteq$, et l'expression $(A \supseteq B)$ signifie que le concept A est plus général que le concept B,

- *spécialisation* ou subsomption inverse : elle est notée par $\subseteq$, et l'expression $(A \subseteq B)$ signifie que le concept B est plus général que le concept A,

- *intersection* : nous la notons par ∞ et l'expression $A \infty B$ signifie que les deux concepts se recouvrent[59] : $(A \infty B) \Leftrightarrow (\neg(A \wedge B) \subseteq \perp)$,

- *disjonction* : nous la notons par $\neq$ et l'expression $A \neq B$ signifie qu'il n'existe aucune classification entre les deux concepts A et B : $(A \neq B) \Leftrightarrow ((A \wedge B) \subseteq \perp)$.

En considérant les différentes valeurs d'inférence entre les éléments de services à comparer, il est possible de mieux illustrer les transitions entre les étapes de l'algorithme de comparaison présenté précédemment (figure VIII.14) en considérant les conditions effectuées sur les résultats d'inférence obtenus à chaque étape de comparaison (figure VIII.15).

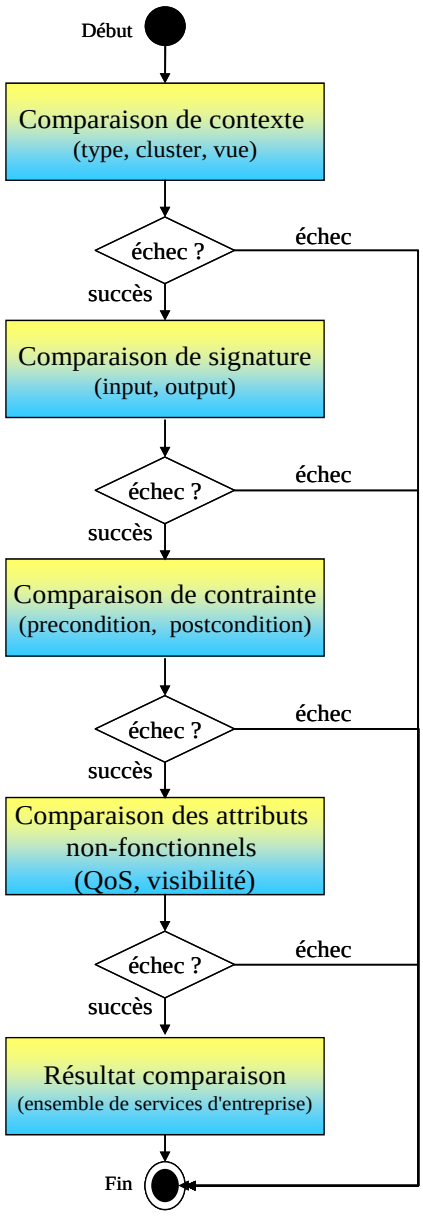

Figure VIII.15. Logique générale de fonctionnement de l'algorithme de comparaison

[59] Nous précisons que le symbole anti-truc (⊥) désigne le concept BOTTOM de la logique de description [Napoli 1997]. Il s'agit du concept le plus spécifique et dont l'extension est vide. Le concept opposé est le concept TOP (⊤) qui est le concept le plus général et qui inclut tous les individus possibles.

Ainsi, et comme le montre la figure VIII.15, l'algorithme de comparaison se termine dès qu'il rencontre un échec (failure) au cours de l'une de ses étapes. Ce choix de conception, qui est par ailleurs paramétrable, est important car il nous paraît illusoire de continuer l'algorithme de comparaison dès qu'un certain seuil (seuil d'échec) est franchi pour l'un des paramètres à comparer. De plus, ce choix peut également être justifié par le fait que le modèle mathématique que nous avons mis en oeuvre pour le calcul de l'indice de similarité est basé sur le produit pondéré (weighted product), ce qui entraîne alors l'arrêt de l'algorithme dès que l'indice d'un paramètre est nul ou jugé négligeable.

VIII.4.3.2.2. Indice de similarité de services

Plus formellement, nous proposons de calculer l'indice global de similarité $sim_{global}(S,S')$ entre deux services S et S' (ou entre une requête S et une annonce S') selon la formule suivante :

$$sim_{global}(S,S') = \prod_{x \in Filter} [sim_x(S,S')]^{\lambda_x}, \text{ où } x \in \{context, signature, constraint, non\text{-}functional\}$$

$$sim_{global}(S,S') = sim_{context}^{\lambda_{context}}(S,S') \cdot sim_{signature}^{\lambda_{signature}}(S,S') \cdot sim_{constra\,int}^{\lambda_{constra\,int}}(S,S') \cdot sim_{non-functional}^{\lambda_{non-functional}}(S,S')$$

où

$sim_{context}(S,S')$ désigne l'indice de similarité entre les contextes des deux services,

$sim_{signature}(S,S')$ désigne l'indice de similarité entre les signatures des deux services,

$sim_{constra\,int}(S,S')$ désigne l'indice de similarité entre les contraintes des deux services,

$sim_{non-functional}(S,S')$ désigne l'indice de similarité entre les attributs non fonctionnels des deux services,

λ_x désigne les poids des différents indices.

Cet indice global de similarité $(sim_{global}(S,S'))$ peut être interprété comme une probabilité de ressemblance des deux services *S* et *S'*. Le produit pondéré que nous avons retenu est justifié par le fait que la similarité globale doit dépendre de la performance de chacun des indices et non de l'excellence de certains indices. Une autre justification tient du fait que la probabilité de réalisation de l'intersection d'un certain nombre d'événements élémentaires indépendants est égale au produit des probabilités de réalisation de chacun des événements élémentaires :

$$P(\bigcap_i A_i) = \prod_i P(A_i).$$

Précisons que dans notre approche, l'algorithme fonctionne de façon flexible. en effet, on choisit les poids des indices en fonction du type de la requête. Par exemple, si l'on veut uniquement effectuer une comparaison des inputs et des outputs en court-circuitant les autres étapes de comparaison, il suffit alors de prendre le poids $\lambda_{signature}$ égal à 1 et tous les autres égaux à 0. En faisant ainsi, tous les indices dont le poids est nul sont ignorés durant l'algorithme de comparaison.

En ce qui concerne l'évaluation de l'indice global de similarité, comme il est un produit d'indices plus élémentaires, il devient nécessaire d'évaluer ces derniers afin de pouvoir

calculer l'indice global. De même que pour l'indice global, le sous-indice de similarité de contexte $sim_{context}(S,S')$ est également une combinaison de sous-indices. Il se calcule en fonction de ses sous-indices de similarité (qui entrent dans la définition de la notion de contexte), et il est de la forme :

$$sim_{context}(S,S') = \prod_{x\in\{\text{type, cluster, view}\}} sim_x^{\lambda_x}(S,S')$$

De même que pour le contexte, nous proposons de calculer les autres indices de façon similaire:

$$sim_{signature}(S,S') = \prod_{x\in\{\text{input, output}\}} sim_x^{\lambda_x}(S,S')$$

$$sim_{constra\,int}(S,S') = \prod_{x\in\{\text{precondition, postcondition}\}} sim_x^{\lambda_x}(S,S')$$

$$sim_{non-functional}(S,S') = \prod_{x\in\{\text{QoS, visibility}\}} sim_x^{\lambda_x}(S,S')$$

Ainsi, de proche en proche on définit notre indice global de similarité comme une combinaison d'un certain nombre d'indices élémentaires. Nous appelons un indice élémentaire, celui qu'il n'est pas possible ou qu'il n'est pas nécessaire de décomposer pour pouvoir le calculer. Par exemple, nous considérons les deux indices $sim_{input}(S,S')$ et $sim_{output}(S,S')$ comme élémentaires dans la mesure où nous les considérons comme non décomposables. Toute la problématique de calcul de l'indice global se traduit sous forme de problématique de calcul des indices élémentaires.

VIII.4.3.2.3. Calcul des indices élémentaires

Une caractéristique fondamentale de notre approche est que la majorité des indices élémentaires de similarité que nous avons définis portent sur le calcul de similarité entre deux concepts d'ontologies, c'est à dire sur la distance entre deux concepts. Cette caractéristique est importante car elle permet d'apporter de l'unification tout en simplifiant le processus de calcul de l'indice global.

Pour le calcul de nos indices élémentaires, nous avons décidé d'exploiter une variante combinant la distance de dissimilarité topologique structurelle (structural topological dissimilarity) introduite dans [Valtchev et Euzenat 1997] et la distance cotopique (upward cotopic distance) [Mädche et Zacharias 2002].

La dissimilarité topologique structurelle est définie sur une ontologie O par :

$$\forall e,e'\in O, \delta(e,e') = \min_{c\in O}[dist(e,c) + dist(e',c)]$$

où $dist(x,y)$ désigne le nombre d'arêtes intermédiaires qui existent entre les deux éléments *x* et *y*. La fonction normalisée correspondant à cette mesure est donnée par :

$$\overline{\delta}(e,e') = \frac{\delta(e,e')}{\max_{x,y\in O}\delta(x,y)}$$

Quant à la distance cotopique, elle est définie sur une ontologie *O* présentant une hiérarchie *H* par la formule suivante :

$$\delta(c,c') = \frac{|UC(c,H) \cap UC(c'H)|}{|UC(c,H) \cup UC(c'H)|}$$

où $UC(c, H) = \{c' \in H; c \preceq c'\}$ est l'ensemble des superclasses de c.

En se basant sur ces deux mesures, nous avons décidé de définir une nouvelle mesure permettant de comparer deux concepts d'une même ontologie. La raison pour laquelle nous avons décidé de définir une nouvelle mesure est due au fait que les mesures existantes ne sont pas suffisamment appropriées pour comparer des concepts: elles ne sont pas sensibles à la profondeur des concepts à comparer. Aussi, nous avons proposé une nouvelle mesure (heuristique) qui permet de mieux calculer le degré de similarité de deux concepts C et C' d'une ontologie O de profondeur *Depth(O).* Elle se calcule selon la formule :

$$sim(C,C') = (1-(\frac{|d-d'|}{d+d'})^{\alpha}) \cdot (1-\frac{\max(d,d')-1}{1+Depth(O)}) \cdot (1-\frac{\min(d,d')-1}{1+Depth(O)})(\frac{1}{(\min(d,d'))^{\beta}})$$

où α et β sont des paramètrès qui agissent sur l'aplatissement (enfoncement de la cloche) de la fonction et qui sont définis par :

$$\alpha = 1+\max(d,d'),\ \ \beta = \log_{10}(Depth(O)),$$

et où d et d' sont respectivement des distances (augmentées[60] de 1) entre le concept C (resp. C') et le plus petit ancêtre commun C_0 :

$$d = dist(C,C_0)+1,\ \ d' = dist(C',C_0)+1$$

L'ancêtre commun C_0 des deux concepts C et C' est la solution de la fonction d'optimisation permettant de définir la dissimilarité topologique structurelle (structural topological dissimilarity) telle qu'introduite dans [Valtchev et Euzenat 1997] :

$$?c:\ \min_{c}[dist(C,c)+dist(C',c)]$$

Bien entendu, la fonction *sim* précédente est une fonction de similarité au sens mathématique du terme du fait qu'elle vérifie les conditions de positivité, de maximalité et de symétrie (cf. *définition 1* du § IV.5.4.4). De plus cette fonction est normalisée du fait qu'elle prend ses valeurs dans l'intervalle [0, 1].

Comme on peut le constater, la formule est constituée de quatre facteurs. Le choix du facteur $(1-(\frac{|d-d'|}{d+d'})^{\alpha})$ dans le calcul de la similarité de deux concepts tient au fait qu'il représente un comportement global que nous désirons. Ce facteur permet une mesure très sensible à la profondeur dans laquelle se trouvent les deux concepts. Plus les deux

[60] Cette augmentation est effectuée dans le seul but d'éviter d'avoir des dénominateurs nuls.

concepts sont proches de l'ancêtre commun et plus cette quantité est grande. A l'inverse cette quantité décroît en fonction de l'éloignement des concepts de l'ancêtre commun. Le seul inconvénient est qu'il privilégie les cas où les distances d et d' sont égales même si l'on s'éloigne considérablement de l'ancêtre commun dans le sens du plan bissecteur du repère (d, d'). Cet inconvénient est vite corrigé par le second facteur $(1-\frac{\max(d,d')-1}{1+Depth(O)})$, qui permet ainsi de décroître la courbe quand les distances d et d' s'agrandissent de façon égale. Ceci permet d'avoir un effet que nous désirons qui est celui de la cloche (loi statistique normale). Cet effet est important car il permet de faire accroître la similarité aux alentours de l'ancêtre commun et de la faire décroître quand les concepts s'éloignent de cet ancêtre. Quant au facteur $(1-\frac{\min(d,d')-1}{1+Depth(O)})$, il permet de lisser davantage la courbe, notamment tout le long du contour de la courbe où d = d'. Le dernier facteur

Afin de mieux apprécier le comportement de la fonction de similarité, nous avons préféré schématiser le tracé de la courbe d'équation sur un repère 3D:

$$sim(d,d') = (1-(\frac{|d-d'|}{d+d'})^{\alpha})\cdot(1-\frac{\max(d,d')-1}{1+Depth(O)})(1-\frac{\min(d,d')-1}{1+Depth(O)})(\frac{1}{(\min(d,d'))^{\beta}}).$$

Les figures VIII.16 à VIII.18 représentent les tracés[61] de cette courbe à différents angles de vue.

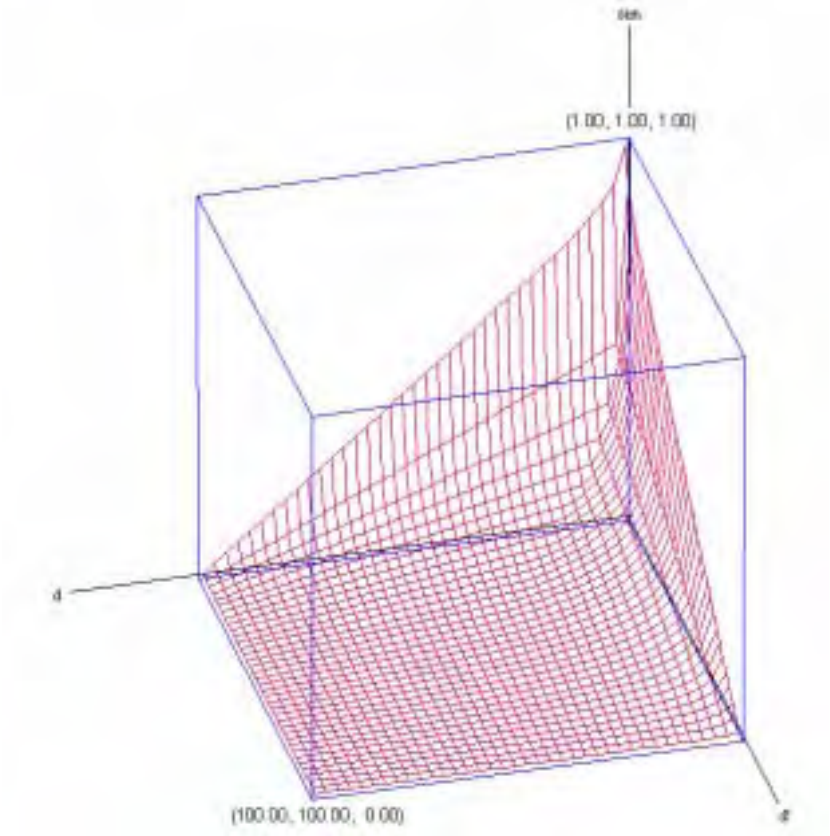

Figure VIII.16. Tracé de la courbe de la fonction de similarité (Vue 1)

[61] Les tracés ont été réalisés avec le logiciel WinPlot en prenant *Depth(O) = 100*.

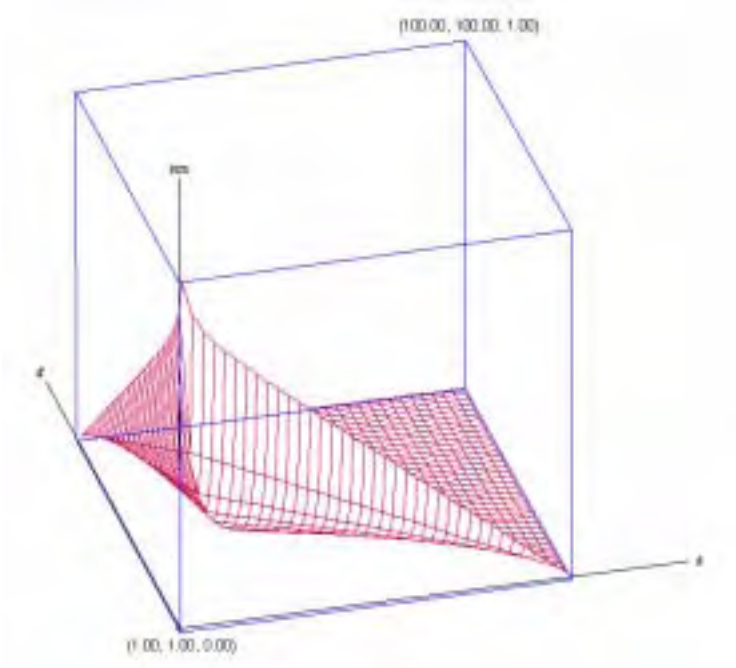

Figure VIII.17.Tracé de la courbe de la fonction de similarité (vue 2)

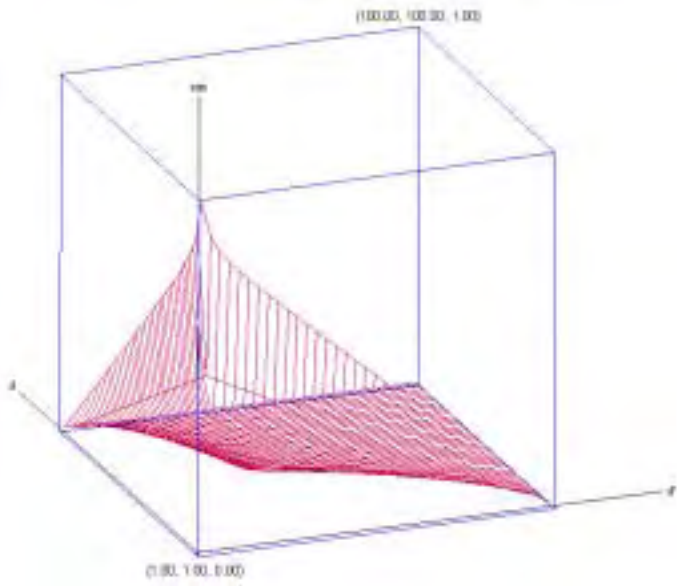

Figure VIII.18. Tracé de la courbe de la fonction de similarité (vue 3)

Comme on peut le constater, la courbe se comporte comme une cloche enfoncée:

- la similarité est forte au voisinage de l'ancêtre commun (lorsque *d* et *d'* convergent vers *1*)
- la similarité s'affaiblit au fur et à mesure que l'on s'éloigne de l'ancêtre (*d* et *d'* tendent vers *Depth (O)*).

Le calcul de quelques valeurs de la fonction de similarité, en prenant pour Depth(O) égal à 100 est consigné dans le tableau VIII.2 qui permet d'illustrer quantitativement le comportement de cette fonction de similarité.

d \ d'	1	2	3	4	5	6	7	8	9	10
1	1.0000	0.9534	0.9189	0.8948	0.8761	0.8603	0.8464	0.8338	0.8219	0.8107
2	0.9534	0.2451	0.2422	0.2392	0.2362	0.2334	0.2307	0.2280	0.2254	0.2229
3	0.9189	0.2422	0.1068	0.1057	0.1046	0.1035	0.1024	0.1013	0.1002	0.0991
4	0.8948	0.2392	0.1057	0.0588	0.0582	0.0576	0.0570	0.0564	0.0558	0.0552
5	0.8761	0.2362	0.1046	0.0582	0.0369	0.0365	0.0361	0.0358	0.0354	0.0350
6	0.8603	0.2334	0.1035	0.0576	0.0365	0.0251	0.0248	0.0246	0.0243	0.0240
7	0.8464	0.2307	0.1024	0.0570	0.0361	0.0248	0.0181	0.0179	0.0177	0.0175
8	0.8338	0.2280	0.1013	0.0564	0.0358	0.0246	0.0179	0.0135	0.0134	0.0132
9	0.8219	0.2254	0.1002	0.0558	0.0354	0.0243	0.0177	0.0134	0.0105	0.0104
10	0.8107	0.2229	0.0991	0.0552	0.0350	0.0240	0.0175	0.0132	0.0104	0.0083

Tableau VIII.2. . Exemple d'évaluation de la fonction sim

Afin mieux illustrer le comportement de notre fonction sur un exemple simple et plus parlant, nous avons choisi d'effectuer des calculs d'indices en ce qui concerne les comparaisons qualitatifs (cf. § VIII.4.2.2.1) précédemment introduits (équivalence, généralisation spécialisation, intersection et disjonction) sur un fragment d'ontologie de véhicule issu de [Srinivasan et al. 2006] (figure VIII.19).

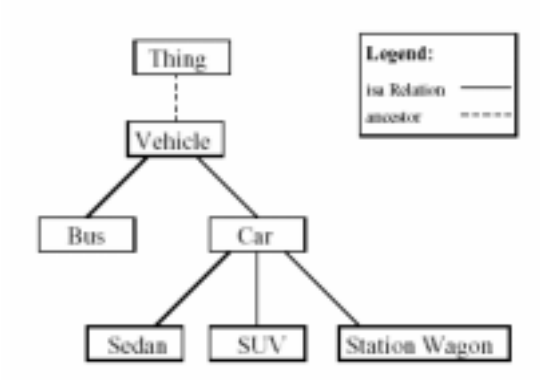

Figure VIII.19. Un fragment d'ontologie [Srinivasan et al. 2006]

Le résultat du calcul des indices obtenu, en prenant *Depth(O) = 100,* est consigné dans le tableau VIII.3.

Type de Comparaison	Concept C	Concept C'	Ancêtre le plus proche C_0	Distance $d = Dist(C,C_0)+1$	Distance $d' = Dist(C',C_0)+1$	$sim(C,C')$
Equivalence	Car	Car	Car	1	1	1.0000
Généralisation	Vehicle	Car	Vehicle	1	2	0.9534
Spécialisation	Sedan	Car	Car	2	1	0.9534
Intersection	Thing	SUV	Thing	1	4	0.8948
Echec	Plane (ce concept n'existe pas dans l'ontologie)	SUV	-	∞	∞	0.0000 (calcul avec passage aux limites)

Tableau VIII.3. Exemple d'évaluation de la fonction sim sur un fragment d'ontologie

VIII.4.3.3. Service de découverte

Dans notre approche, le processus de découverte est pris en charge par le service de découverte (Discovery Service). Ce dernier est un service d'intégration particulier qui s'active dès réception d'une requête de type :

$$request(\#Discover, Arguments)$$

où *Arguments* désigne les caractéristiques de la requête de découverte qui permet ainsi de savoir s'il s'agit d'une découverte de données, de fonctions ou de services de façon générale.

La requête de découverte permettant de formuler les besoins d'un service ou d'un utilisateur est formalisée sous forme d'une ontologie OWL-S+ (cf. § VII.4.1), conformément au principe de l'algorithme précédent (cf. § VIII.4.2.2.1). Ceci veut dire que les arguments de la requête contiennent généralement (sauf dans le cas de découverte syntaxique) un fragment d'une ontologie OWL-S+. Une fois constituée le fragment de cette ontologie est alors comparée aux éléments du référentiel d'ontologies en utilisant des moteurs d'inférence selon le principe de l'algorithme de comparaison précédent. Dans ce scénario, l'utilisation de OWL-S+ permet d'améliorer substantiellement l'algorithme de comparaison dans le sens où il permet de réduire les services filtrés et sélectionnés, et ce en exploitant plusieurs aspects de OWL-S+ tels que la qualité de service, la catégorisation des services, etc.

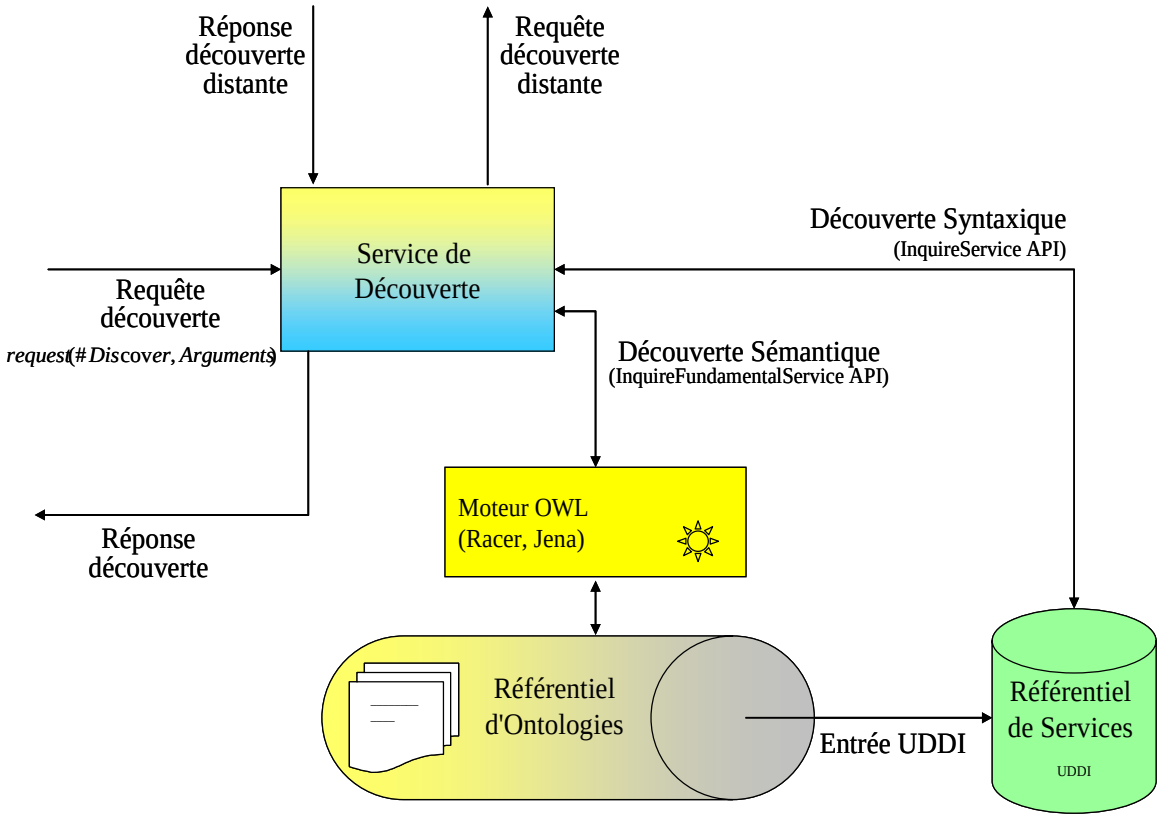

Figure VIII.20. Principe général de fonctionnement du service de découverte

La figure VIII.20 illustre le principe de fonctionnement de notre service de découverte. Comme on peut le constater, le service se base à la fois sur la découverte syntaxique et

la découverte sémantique. Cette approche permet ainsi d'apporter de la flexibilité à notre architecture AIE. En faisant ainsi, le service peut adresser à la fois les services décrits syntaxiquement et ceux décrits sémantiquement. Techniquement, la découverte syntaxique se base sur les API existantes des technologies web services (InquireService API), tandis que la découverte sémantique se base sur l'algorithme de comparaison que nous avons proposé précédemment.

A l'arrivée d'une requête, le service de découverte analyse la requête de découverte et peut alors effectuer une recherche distante dans le cas où le service demandé n'est pas dans le cluster local ou dans le cas le service demandé n'est pas disponible localement. En fonction du type de découverte, un algorithme qui peut être syntaxique ou sémantique est alors exécuté. Dans le cas de découverte sémantique, c'est l'algorithme de comparaison défini précédemment qui sera exécuté. Il détermine alors individuellement les comparaisons définies dans chacune des trois étapes (i), (ii) et (iii) (de l'algorithme de comparaison § VIII.4.2.2.1). Les différents résultats sont alors agrégés et le résultat de l'agrégation est restreint conformément à la qualité de service voulue et aussi en fonction de la visibilité associée au profil de l'émetteur de la requête.

VIII.4.4. Médiation de services d'entreprise

La médiation de services, et dans une plus large mesure la composition[62] de service, permet de créer de nouvelles fonctionnalités en offrant la possibilité de combiner des fonctionnalités offertes par d'autres services existants. Contrairement aux processus métier traditionnels qui sont exécutés de manière statique et répétitive, les services web composés s'exécutent dans un environnement dynamique. Parmi les travaux existants qui s'intéressent à la composition, nous pouvons citer en premier lieu les langages procéduraux de type BPEL4WS [IBM et al. 2005], BPML [BPMI 2003] et WSCI et qui permettent de générer l'implantation d'un service composite à partir de spécifications déclaratives de son comportement. Nous avons également les travaux de [Hull et al. 2003] et [Bultan et al. 2003] qui proposent une modélisation abstraite des services ainsi que la définition d'un cadre formel pour les services composés. D'autres travaux de la communauté web sémantique explorent des approches combinant des outils d'annotation de services et de planification de manière à pouvoir composer automatiquement des services en vue d'atteindre des fonctionnalités désirées [Narayanan et McIlraith 2002][Hendler et al. 2003]. Parmi les projets les plus matures qui s'intéressent au processus de composition, nous pouvons citer le projet ICARIS [Narayanan et McIlraith 2002][Tosic et al. 2001] et METEOR-S [Meteor-s][Cardoso et Sheth 2005]. ICARIS se base sur une approche qui utilise les patterns de composition en exploitant les technologies XML, JINI et JavaBeans, tandis que METEOR-S se base sur une approche exploitant des ontologies et plus précisément il exploite quatre types de sémantiques en vue de composer des services: la sémantique des données, la sémantique des fonctions (functional semantics), la sémantique de la qualité (QoS semantics) et la sémantique d'exécution (execution semantics).

En ce qui concerne les approches de médiation, ces dernières portent principalement sur les fonctionnalités de transformation d'événements transitant entre deux services. [Bussler 2003-c] décrit une typologie qui distingue deux types de médiation: une médiation d'événements métier (business event mediation) et une médiation de

[62] La composition implique la capacité de sélectionner, de composer et de faire interopérer des services existants. De ce fait, nous considérons la médiation (ou l'interopération) comme un aspect de la composition.

processus métiers (business process mediation). La médiation d'évenements permet de résoudre les hétérogénéités liées aux données tandis que la médiation de processus porte sur la résolution des hétérogénéités liées au comportement. De plus, [Bussler 2003-c] distingue trois formes typiques de médiation: la médiation peer (peer mediation) où le processus de médiation est pris en charge par les services impliqués, médiation centralisée (brokered mediation) où le processus de médiation est pris en charge par une tierce entité, la médiation enchaînée (chain mediation) où un système peut solliciter un autre système pour effectuer la médiation à sa place. Nous devons noter que ces différentes formes de médiation possèdent toutes leurs avantages et leurs inconvénients. Par ailleurs, le fait qu'une forme soit plus adaptée qu'une autre dépend généralement du contexte d'utilisation. Dans le cadre de nos travaux, nous exploitons une approche qui combine à la fois la médiation centralisée et la médiation enchaînée afin de résoudre les hétérogénéités liées aux services d'entreprise. Ce type d'approche est à notre sens le plus approprié dans le contexte de grandes entreprises réactives. L'idée principale est d'utiliser un certain nombre de médiateurs distribués au sein de l'entreprise afin de résoudre les incompatibilités et les hétérogénéités pouvant exister entre les services d'entreprise. De plus, notre approche se focalise sur les deux aspects de médiation à savoir la médiation de données et la médiation fonctionnelle.

VIII.4.4.1. Médiation de données

VIII.4.4.1.1. Approches classiques de médiation de données

Rappelons que traditionnellement, la médiation de données se fait par transformation syntaxique. Cette dernière opère des transformations au niveau des syntaxes en utilisant des mappings XSLT ou carrément du code ad hoc permettant d'établir des connexions entre les services (figure VIII.21).

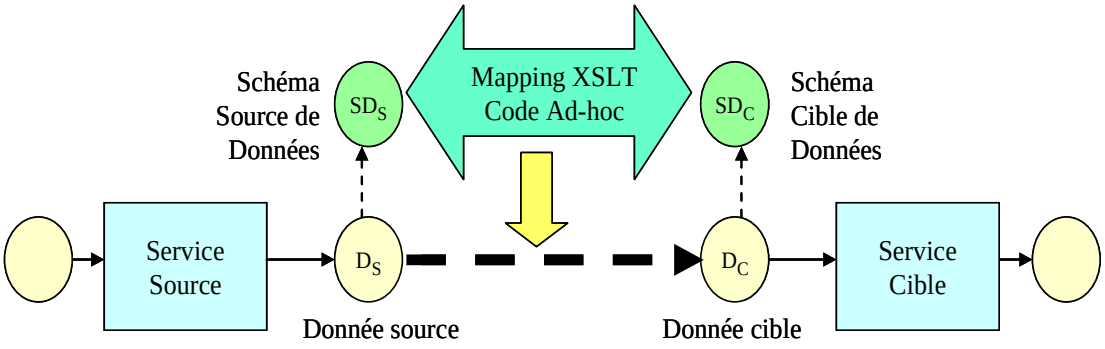

Figure VIII.21. Principe de médiation traditionnelle de données

De façon générale, les travaux de médiation de données portent principalement sur les approches d'intégration et de transformation de données [Ullman 1997][Pottinger et Bernstein 2003], dont certains travaux exploitent le concept d'ontologie [Sciore et al. 1994][Lakshmanan 2003] auquel nous nous intéressons. Il existe un certain nombre de travaux portant sur les langages d'explicitation des transformations entre schémas hétérogènes [Krishnamurthy et al. 1991][Davidson et Kosky 1997][Cluet et al. 1998]. Ces travaux se basent principalement sur une approche manuelle et il est important de pouvoir automatiser le processus de transformation afin de dériver systématiquement les médiations et/ou et de les effectuer à la volée. D'autres travaux existent et concernent la définition des correspondances nécessaires pour l'intégration et la transformation de

données [Parent et Spaccapietra 1998][Miller et al. 2001][Bowers et Delcambre 2000]. L'approche utilisée dans [Miller et al. 2001] permet aux utilisateurs de définir des correspondances entre les schémas de bases de données que le système complète par la suite. La technique utilisée dans [Pottinger et Bernstein 2003] permet de définir des correspondances simples en vue d'intégrer des bases hétérogènes. [Bowers et Delcambre 2000] propose de représenter les correspondances en se basant sur la notion de modèle super-imposé (superimposed model).

Dans le cadre de l'approche OWL-S, aucun mécanisme de médiation de données n'est défini. De plus, la notion de mapping sémantique, qui est importante du fait qu'elle permet de relier les ontologies des domaines hétérogènes, n'est pas prise en charge. Ceci constitue l'une des limites de l'approche OWL-S. L'une des rares approches sémantiques qui propose de prendre en charge cet aspect est l'approche WSMO. Cette dernière se base sur la notion de mapping d'ontologies mais l'approche n'est pas assez mature et elle est encore en cours de développement. Cette approche n'explique pas clairement la manière avec laquelle elle procède pour effectuer la médiation de données [WSMO 2005]. De plus, elle est en contradiction avec notre approche du fait que les formalismes et les outils sur lesquels elle se base sont totalement incompatibles avec l'approche OWL-S (§ IV.7.6) que nous avons choisie. Dans ce qui suit, nous allons présenter notre approche de médiation dont le principe présente certaines similitudes à celui présenté dans [Bowers et Delcambre 2000] mais qui prend en compte les spécificités issues des chapitres précédents, à savoir les notions d'ontologies d'entreprise, de mappings sémantiques et de mappings syntaxiques.

VIII.4.4.1.2. Approche ODSOI de médiation de données

VIII.4.4.1.2.1 Fondements théoriques de la médiation de données

Notre approche de médiation repose sur une formalisation basée sur la notion d'interprétation abstraite, qui est utilisée à l'origine pour formaliser la correspondance approximative entre sémantiques concrètes de programmes syntaxiquement corrects [Cousot et Cousot 2002].

Tout d'abord, signalons que notre approche de médiation exploite la notion générique d'entité sémantique. Une entité sémantique d'entreprise est définie comme étant toute partie d'un système d'information d'entreprise qui peut être un cluster, un service, une opération, une donnée, etc., et qui comporte à la fois une description syntaxique et une description sémantique. Afin d'apporter un maximum de flexibilité dans notre approche, nous acceptons qu'une entité sémantique puisse ne pas posséder de description sémantique (§ VIII.4.2.2). Une entité sémantique qui nous intéresse plus particulièrement dans le cadre de la médiation de données est le service sémantique. Nous définissons un service sémantique d'entreprise comme un cas particulier d'entité sémantique d'entreprise qui permet d'exposer certains aspects d'un système d'information (certaines fonctionnalités d'applications ou des sources de données.

De ce point de vue, le service sémantique constitue le résultat de l'intégration des concepts issus des deux architectures décrites aux chapitres précédents : architecture de service et architecture sémantique. Dans le cadre de la médiation de données, nous considérons un service sémantique comme un ensemble de ports sémantiques. Les ports sont définis comme des points de connexion au service. On distingue deux types de ports; des ports d'entrée et des ports de sortie qui servent respectivement comme entrées et sorties du service.

Définition 1: *Un service sémantique*[63] *d'entreprise S est défini comme un ensemble de ports :*

$$S = <P> = <\{P_{i,\ i = 1..p}\}>$$

Dans notre framework, chaque entité sémantique *X (les services et les ports sémantiques)* est décrite à la fois par une syntaxe et une sémantique.

Proposition 1: Toute entité sémantique est définie par une paire

$$X \cong [Syn(X), Sem(X)],$$

où Syn(X) dénote la syntaxe de X et Sem(X) dénote sa sémantique.

Le symbole $\cong$ utilisé précédemment est celui de l'opérateur d'*équivalence de descriptions* que nous avons défini afin de décrire les entités sémantiques. Il est différent de l'opérateur d'équivalence $\equiv$ qui est défini dans la section VIII.4.3 et qui est l'équivalence sémantique portant sur des concepts d'ontologies. Il s'agit en fait d'un opérateur linguistique car il transgresse certaines lois mathématiques. En effet, l'une des propriétés importante de cet opérateur d'équivalence de description ($\cong$) est que la description d'un ensemble de paires d'entités $[X_i, Y_i]$ est une paire d'ensembles de descriptions des éléments des paires d'entités :

$$\{[X_i, Y_i]\} \cong [\{X_i\}, \{Y_i\}].$$

Proposition 2 : *Deux entités sémantiques X et Y sont dites équivalentes du point de vue description et l'on note $X \cong Y$ si et seulement si X et Y possèdent des descriptions syntaxiques et sémantiques équivalentes i.e.,*

$$X \cong Y \Leftrightarrow ((Sem(X) \equiv Sem(Y)) \wedge (Syn(X) \approx Syn(Y))).$$

où les opérateurs $\equiv$ et $\approx$ sont respectivement les opérateurs d'équivalence sémantique (cf. § VIII.4.3) et d'équivalence syntaxique (qui découle de la relation d'ordre syntaxique $\preceq$ qui est introduit un peu plus loin).

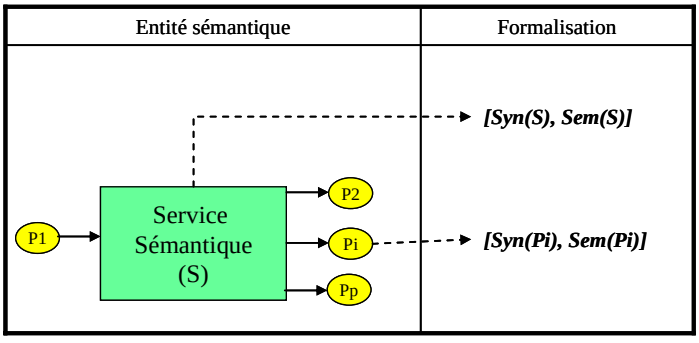

Figure VIII.22. Formalisation des entités sémantiques

Les services sémantiques et leurs différentes caractéristiques sont alors des entités

[63] De façon plus complète, un service sémantique S est défini comme un quintuplet <Co,P, F, Ct, N> (c.f. § VIII.4.3.2), où chaque composante définit une entité sémantique qui permet de représenter respectivement le contexte du service (Co), les ports (d'entrée et de sortie) du service (P), les opérations ou les fonctions du service (F), les contraintes du service (Ct) et les paramètres non-fonctionnels (N).

sémantiques décrites par la paire syntaxe-sémantique (figure VIII.22). En particulier, les ports d'entrée et de sortie d'un service sont décrits par la paire syntaxe-sémantique. Ainsi, si S représente un service sémantique et $P_{i,\ i\ =\ 1..p}$ représentent l'ensemble P des ports sémantiques du service S, nous avons alors:

$$\begin{aligned} S &= <P> \\ &= <\{P_{i,\ i\ =\ 1..p}\}> \\ &\cong <\{[Syn(P_i), Sem(P_i)]\}> \\ &\cong <[\{Syn(P_i)\},\{Sem(P_i)\}]> \\ &\cong [<\{Syn(P_i)\}>, <\{Sem(P_i)\}>] \\ &\cong [<Syn(P)>, <Sem(P)>] \\ &\cong [Syn(S), Sem(S)] \end{aligned}$$

La syntaxe des services sémantiques, ou plus précisément la syntaxe des ports sémantiques (et de façon plus générale des entités sémantiques) appartient à un domaine syntaxique D_{Syn} qui est un poset (partially ordered set)

$$po\langle D_{Syn;}\preceq\rangle$$

où D_{Syn} est partiellement ordonné par la relation standard de sous-typage $\preceq$ qui formalise la notion de compatibilité syntaxique (i.e.: $X_i \preceq X_j$ signifie que l'entité sémantique X_i est un sous-type de X_j ou que l'entité sémantique X_i est syntaxiquement compatible avec l'entité sémantique X_j).

La sémantique des ports appartient au domaine sémantique D_{Sem} qui est également un poset

$$po\langle D_{Sem;}\sqsubseteq\rangle$$

où D_{Sem} est partiellement ordonné par la relation standard de subsumption $\sqsubseteq$ utilisée en logique de description pour formaliser la compatibilité sémantique (i.e.: $X_i \sqsubseteq X_j$ signifie que l'entité sémantique X_i est sémantiquement compatible avec l'entité sémantique X_j ou que X_j subsume X_i). Notons que les domaines syntaxiques et les domaines sémantiques sont souvent accompagnés de propriétés plus fortes, comme l'ordre total ou les treillis complets.

La correspondance entre le domaine syntaxique et le domaine sémantique est donnée par une paire de mappings α (qui est l'abstraction sémantique) et δ (qui est la concrétisation syntaxique). La concrétisation syntaxique $\delta[Sem(P_i)]$ exprime le raffinement syntaxique de la représentation sémantique de P_i. Elle constitue une approximation faible (est un sur-type) de la syntaxe réelle de P_i du fait que

$$Syn(P_i) \preceq \delta[Sem(P_i)].$$

L'abstraction sémantique $\alpha[Syn(P_i)]$ exprime l'enrichissement sémantique de la représentation syntaxique de X_i. Contrairement à la concrétisation syntaxique, l'abstraction sémantique constitue une approximation plus forte (est un sous-concept ou une spécialisation) de la sémantique réelle de P_i du fait que

$$\alpha[Syn(P_i)] \sqsubseteq Sem(P_i).$$

Formellement, si tout élément Syn_i du domaine syntaxique $po\langle D_{Syn;}\preceq\rangle$ possède une meilleure approximation dans le domaine sémantique $po\langle D_{Sem;}\sqsubseteq\rangle$ donné par $\alpha[Syn_i]$,

alors la paire $\langle \alpha, \delta \rangle$ forme un treillis de Galois (Galois connection) noté $D_{Syn} \langle \alpha, \delta \rangle D_{Sem}$ qui par définition, signifie que

$$\forall X \in D_{Syn} \quad \forall Y \in D_{Sem}, \alpha[X] \sqsubseteq Y \Leftrightarrow X \preceq \delta[Y]$$

L'une des principales raisons de cette formalisation est d'utiliser les propriétés des treillis de Galois pour pouvoir dériver par construction la représentation sémantique exacte (resp. approximative) $Sem(P_i)$ à partir d'une représentation syntaxique donnée $Syn(P_i)$ en se basant sur l'enrichissement de la spécification $Syn(P_i)$:

$$\alpha[Syn(P_i)] = Sem(P_i) \text{ (resp. } \alpha[Syn(P_i)] \sqsubseteq Sem(P_i)).$$

De la même façon, on peut dériver par construction la représentation syntaxique exacte (resp. approximative) $Syn(P_i)$ à partir de la représentation sémantique $Sem(P_i)$ en effectuant un raffinement de la spécification $Sem(P_i)$:

$$Syn(P_i) = \delta[Sem(P_i)] \text{ (resp. } Syn(P_i) \preceq \delta[Sem(P_i)]).$$

La principale caractéristique du treillis de Galois défini précédemment est que α est une relation surjective (i.e.: $\forall Y \in D_{Sem}, \exists X \in D_{Syn}: \alpha[X] = Y$) si et seulement si δ est une relation injective (i.e.: $\forall Y \in D_{Sem}, \forall Y' \in D_{Sem}: \delta[Y] = \delta[Y'] \Rightarrow Y = Y'$). Comme conséquence, on doit seulement construire la relation α comme fonction surjective (resp. δ comme relation injective) dans le but de dériver automatiquement la relation adjointe δ (resp. α) comme relation injective (resp. surjective). Cette approche est motivée par le fait que dans l'univers du web sémantique et de l'entreprise sémantique[64], la réalité est plus complexe parce que δ est souvent une relation arbitraire tandis que α est moins problématique. Ce qui signifie qu'en réalité α peut facilement être (semi-) automatisée, et plus encore il existe actuellement des algorithmes pertinents qui peuvent prendre en charge certaines préoccupations de l'abstraction sémantique comme la translation XSD2OWL [Rhizomik 2005] (i.e., de XSD vers OWL) ou translation WSDL2OWL-S [SemWebcentral 2005] (i.e.., de WSDL vers OWL-S). Cependant, δ est plus problématique et elle nécessite une solution appropriée afin de lever son ambiguïté. Pour pallier cette difficulté, nous proposons de construire δ grâce à des règles de mapping tels que définies au chapitre VII (§VI.6.4) et ensuite de déduire de façon unique et systématique la relation α en utilisant la formule:

$$\forall X \in D_{Syn}: \alpha[X] = \sqcap \{Y \in D_{Sem}: X \preceq \delta[Y]\},$$

où $\sqcap$ représente l'opération d'infimum (GLB: Greatest lower bound) qui est défini pour tout $X \in D_{Sem}$, par les deux conditions suivantes :

i) $\sqcap X \sqsubseteq X$,

ii) $(\exists X' \in D_{Sem}: X' \sqsubseteq X) \Rightarrow (X' \sqsubseteq \sqcap X)$.

VIII.4.4.1.2.2 Principe de la médiation de données

Dans notre approche, le processus de médiation peut être considéré comme étant une relation composée τ (figure VII.23)

[64] L'entreprise sémantique est un concept qui a été initialement introduit dans [Gadon 2001] puis repris par plusieurs auteurs dont [Pollock 2005].

$$\tau = \delta o \varphi o \alpha$$

et qui comprend les étapes suivantes :

- l'abstraction sémantique (α) qui traduit les schémas des données source (XML) en structure plus abstraite appartenant au niveau sémantique (OWL);
- la classification sémantique (φ) qui permet de transformer sémantiquement tout concept d'ontologie d'un domaine sémantique source en un autre concept ontologique équivalent appartenant au domaine sémantique cible;
- et enfin la concrétisation syntaxique (δ) qui permet de traduire les concepts ontologiques résultants appartenant au niveau sémantique cible au format de donnés cible du service concerné (XML).

Les relations d'abstraction sémantique (α) et de concrétisation syntaxique (δ) sont telles que définies dans la section précédente (§ VIII.4.4.1.2.1). Ces relations sont basées sur les mappings syntaxiques tels que définis dans les chapitre VII (§ VII.6.4). Quant à la relation de transformation ou de classification sémantique (φ), elle permet d'établir des relations sémantiques entre les domaines sémantiques D_{Sem} et D'_{Sem} auxquels appartiennent les sémantiques des ports P_i et P_j :

$$\varphi: po\langle D_{Sem;} \sqsubset \rangle \rightarrow po\langle D'_{Sem;} \sqsubset \rangle).$$

Dans la réalité, les domaines sémantiques D_{Sem} et D'_{Sem} correspondent à la notion d'ontologies locales que nous avons définies au chapitre VII (§VII.5). L'objectif de cette transformation φ est de trouver des ports sémantiquement compatibles $Y \in D'_{Sem}$ pour chaque port $X \in D_{Sem}$ d'un service donné.

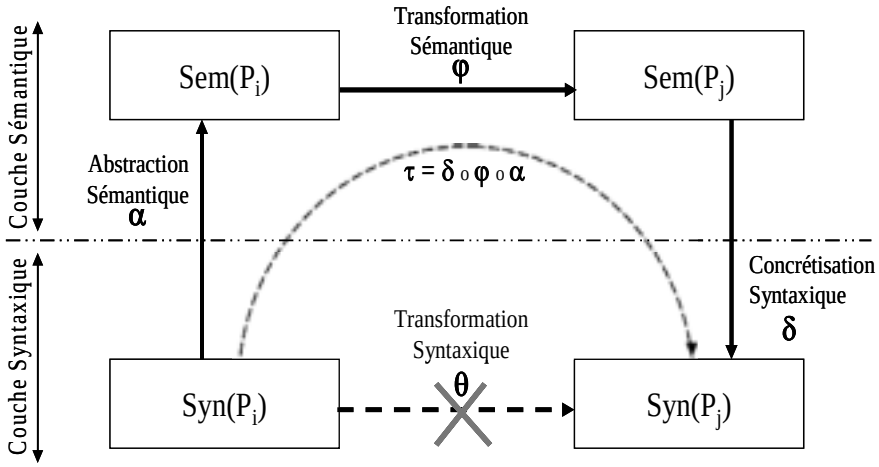

Figure VIII.23. Principe de la médiation de données [Izza et al. 2006-b]

En exploitant les relations δ, φ et α, nous pouvons alors définir une médiation sémantique flexible des données. Cette médiation peut transformer sémantiquement les données sources envoyées par un service source en données cibles équivalentes pouvant être consommées par un service cible, en permettant ainsi la connexion sémantique de ces deux services sémantiques.

Dans le but d'effectuer la médiation sémantique formalisée par la relation τ, une condition forte que nous exploitons est que $\delta o \varphi o \alpha[P_i]$ et $\theta[P_i]$ (avec θ dénote la transformation syntaxique) doivent être équivalentes (sémantiquement et

syntaxiquement équivalentes), c-à-d :

$$(Sem(\theta[Syn(P_i)]) = \varphi[Sem(P_i)]) \wedge (\theta[Syn(P_i)] = \delta[\varphi[Sem(P_i)]]).$$

Le premier terme de la conjonction $(Sem(\theta[Syn(P_i)]) = \varphi[Sem(P_i)])$ signifie que pour tout port P_i, la sémantique de la transformation syntaxique du port P_i doit être la même que la transformation sémantique de la sémantique du port P_i. Le second terme $(\theta[Syn(P_i)] = \delta[\varphi[Sem(P_i)]])$ désigne le fait que pour tout port P_i, la syntaxe de la transformation sémantique du port P_i est la même que la transformation syntaxique de la syntaxe du port P_i.

La généralisation de ce cadre générique de médiation dans le contexte de l'intégration des services d'entreprise consiste à considérer les ports comme des entités complexes d'une part, et de considérer notre approche de structuration syntaxique (§ VI.4.5) et sémantique (§ VII.5) particulières d'autre part. Ainsi, la généralisation du processus de médiation illustré en figure VIII.23 donne naissance au processus plus complet illustré en figure VIII.24.

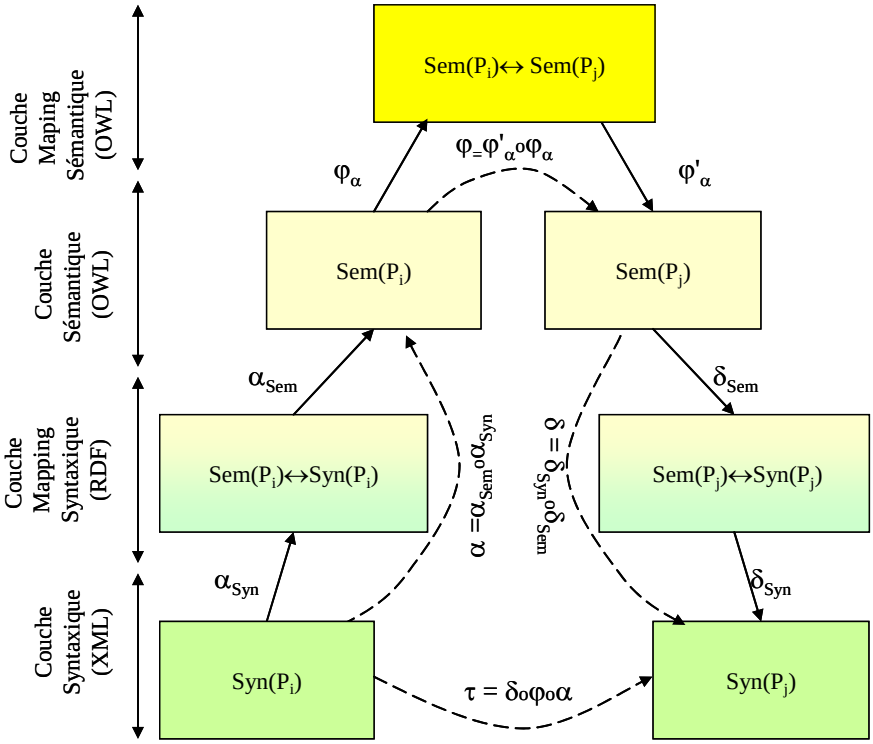

Figure VIII.24. Approche Stratifiée du Processus de Médiation de Données [Izza et al. 2006-b]

Comme on peut le constater, nous avons pris en considération les différents éléments de l'architecture syntaxique (chapitre VI) et de l'architecture sémantique (chapitre VII). Nous avons en particulier pris en compte les mappings sémantiques qui peuvent exister entre les domaines sémantiques (ontologies locales). Nous avons également pris en compte les mappings syntaxiques qui permettent de lier les syntaxes aux sémantiques.

La correspondance entre les différents domaines sémantiques est définie en utilisant les mappings sémantiques (verticaux) tels que décrits dans la section VIII.6.5. De façon plus formelle, ces mappings sont implémentés sous la forme de règles de mapping sémantique de la forme Sem ↔ Sem. En conséquence, la classification sémantique φ peut être considérée comme une relation composée de la forme

$$\varphi = \varphi'_{\alpha} \, o \, \varphi_{\alpha},$$

où φ'_{α} et φ_{α} dénotent la notion de subsomption sémantique définie dans les mappings verticaux.

De façon similaire, la correspondance entre la syntaxe et la sémantique est définie grâce à la notion de règles de mapping syntaxique. Ces règles permettent de rendre possible l'abstraction sémantique (α) et la concrétisation syntaxique (δ). Les règles de mapping syntaxique sont bidirectionnelles de la forme Syn $\leftrightarrow$ Sem, où Syn sont des instances du domaine syntaxique (XML) qui correspondent aux expressions Sem qui sont des instances du domaine sémantique (OWL). Notez que, les expressions Syn (resp. Sem) sont implémentées grâce à la notion d'espaces de noms qui permettent ainsi d'identifier de façon unique chaque élément syntaxique et sémantique.

Comme résultat d'utilisation de ces règles, nous avons

$$\alpha = \alpha_{Sem} \, o \, \alpha_{Syn}$$

et

$$\delta = \delta_{Syn} \, o \, \delta_{Sem},$$

où α_{Syn} et α_{Sem} (resp. δ_{Syn} et δ_{Sem}) sont des relations qui permettent respectivement de lier la syntaxe des ports aux règles de mapping syntaxique et de lier les règles de mapping à la sémantique des ports (resp. sont des relations qui permettent respectivement de lier la sémantique des ports aux règles de mapping syntaxique et de lier les règles de mapping à la syntaxe des ports).

Du point de vue implémentation, les relations α_{Sem}, α_{Syn}, δ_{Sem} et δ_{Syn} sont définies comme des accesseurs qui exploitent les règles de mapping syntaxique, tandis que les relations φ_{α} et φ'_{α} sont définies sous forme de méthodes invoquant les fonctionnalités offertes par les moteurs d'inférence tels que le test de subsumption et qui exploite les mappings sémantiques.

VIII.4.4.1.3. Service de médiation de données

Dans notre approche, le processus de médiation de données tel que illustré précédemment est pris en charge par le service de médiation de données (Data Mediation Service). Ce dernier est un service d'intégration particulier qui s'active dès réception d'une requête de type :

$$request(\#Mediate, Arguments)$$

où *Arguments* désigne les caractéristiques de la requête de médiation qui permet ainsi de préciser le service source, les données source, et le service cible.

Le fonctionnement du service de médiation de données est basé sur l'utilisation d'un moteur d'inférence basé sur OWL (ex., Racer [Racer 2005]) qui permet de raisonner aussi bien sur des concepts (TBox) que sur des instances (ABox) et qui permet ainsi de transformer dynamiquement les instances de données d'un contexte à un autre contexte en exploitant les ontologies (§VII.5, §VII.6.1) et les mappings précédemment définis (§VII.6.4, §VII.6.5). Ces mappings constituent la base pour la médiation : c'est en fonction des relations sémantiques définies dans ces mappings (par exemple, équivalence, subsomption, subsomption inverse, disjonction) que des instances d'une ontologie source peuvent alors être transformées (ou appariées avec) en instances d'une ontologie cible.

La figure VIII.25 illustre le principe de fonctionnement de notre service de médiation. Comme on peut le constater, le service de médiation permet de mettre en œuvre des mécanismes de médiation d'instances (de données) en se basant sur les ontologies définies auparavant et en se basant aussi sur la médiation d'ontologies définie en design-time à travers la notion de mappings verticaux (§ VII.5.3).

Comme on peut le remarquer sur la figure VIII.25, les données échangées entre les services fondamentaux sont basées sur des documents XML dont les schémas (XSD) sont supposés être mappés à nos ontologies durant les étapes amont. Le processus de médiation permet alors de transformer les données issues du service source en données exploitables par le service cible tout en transitant par la couche sémantique. Cette façon d'interconnecter les applications peut s'avérer plus flexible qu'une interconnexion plus directe s'effectuant au niveau syntaxique.

Bien entendu, le processus de médiation exploite la spécificité de l'architecture ontologique dans la mesure où l'effort de médiation peut différer d'une situation à une autre selon que les services fondamentaux partagent la même ontologie locale, ou partagent la même ontologie de domaine (tout en référençant des ontologies locales différentes) ou selon qu'ils sont obligés de réaliser la réconciliation sémantique le plus loin possible en passant par l'ontologie globale (du fait que les services concernés référencent des ontologies locales et de domaines différentes).

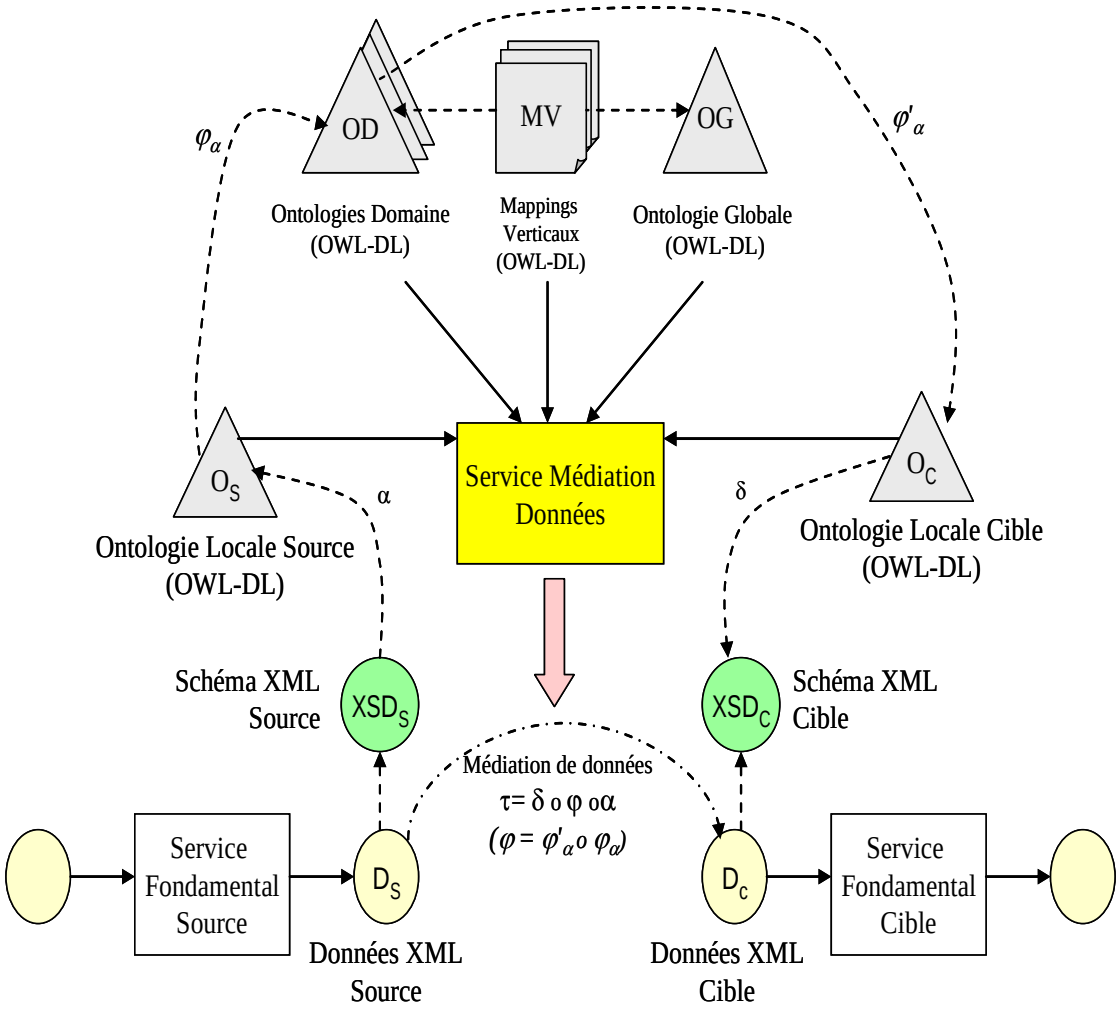

Figure VIII.25. Principe du service de médiation de données

VIII.4.4.1.4. Cohabitation syntaxe-sémantique

En réalité, les services de médiation sont plus complexes du fait qu'ils doivent prendre en charge les considérations de flexibilité durant le processus de médiation. En effet, les services de médiation doivent favoriser la coexistence à la fois des connexions sémantiques et des connexions syntaxiques. Ce type de prise en charge permet alors d'apporter de la flexibilité en matière de connexion des services. Ceci est justifié par le fait que le scénario de migration n'est jamais instantané, et en conséquence nous devons envisager l'existence à la fois de services basés sur la syntaxe et de services basés sur la sémantique.

Par conséquent, le processus de médiation opéré au sein des services de médiation est plus complexe et le processus illustré dans les sections précédentes ne représente en fait qu'un fragment du processus global de médiation. Il s'agit plus précisément de la phase "Exécution de la médiation sémantique" de l'étape 3 du processus global (figure VIII.26).

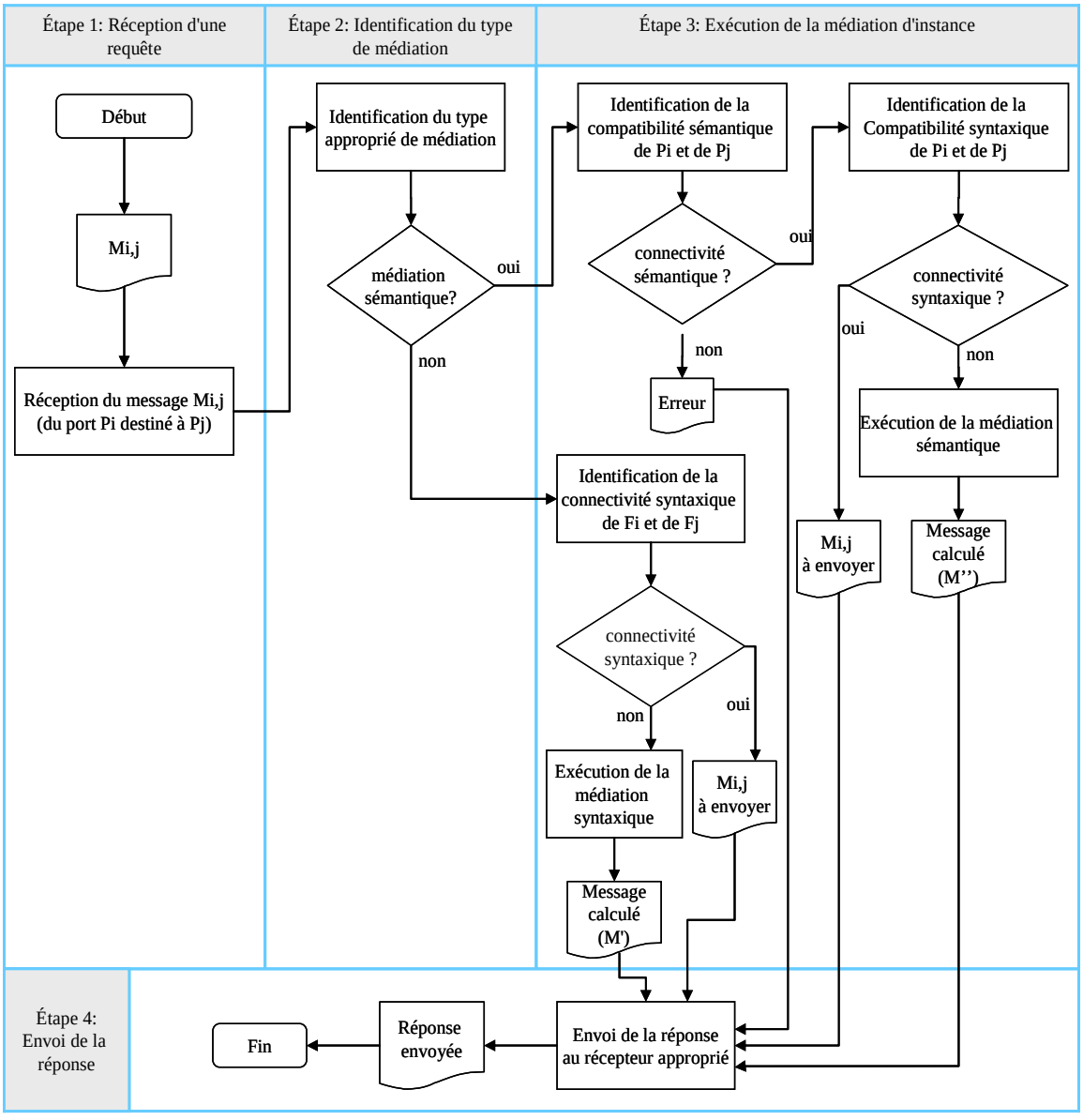

Figure VIII.26. Processus global du service de médiation de données

Comme on peut le constater sur la figure VIII.26, le processus global de médiation comporte réellement quatre étapes :

- *étape 1 - réception d'une requête* qui permet de capturer un message $M_{i,j}$ d'un port P_i d'un service source donné destiné au port P_j d'un service cible donné,
- *étape 2 - identification du type de médiation* qui permet de choisir le type de médiation (syntaxique ou sémantique) en fonction du degré de maturité (cf. § VIII.2.2.2) des services mis en jeu,
- *étape 3 - exécution de la médiation d'instance* qui permet d'effectuer la transformation du message $M_{i,j}$ en effectuant la médiation de donnée dont le type a été identifié précédemment. Deux types de médiation peuvent être exécutées:
    - *médiation syntaxique* qui exploite une transformation syntaxique classique (θ),
    - *médiation sémantique* qui exploite le principe de médiation illustré dans les deux sections précédentes ($\tau = \delta \circ \varphi \circ \alpha$),
- *étape 4 - envoi de la réponse* qui permet de transmettre la réponse au récepteur approprié.

Signalons que le processus global de médiation se base sur la notion de compatibilité de ports et de connectivité de services. De façon similaire à la compatibilité des ports, nous définissons la connectivité (qui est quelque sorte une compatibilité relativement à la connexion de deux services) de deux services par :

Définition (connectivité entre deux services): Il existe une connectivité entre deux services S_1 et S_2 et nous notons $S_1 \preceq_p S_2$ si et seulement s'il existe au moins un port de sortie de S_1 qui est compatible avec un port d'entrée de S_2. De même que pour les ports, nous définissons la connectivité syntaxique et la connectivité sémantique de deux services.

- la *connectivité syntaxique* de services : deux services S_1 et S_2 sont syntaxiquement connectables (connexion relativement aux ports) que nous notons $S_1 \preceq_p S_2$ si et seulement s'il existe au moins un port de sortie de S_1 qui est syntaxiquement compatible avec un port d'entrée de S_2, c'est à dire :

$$S_1 \preceq_p S_2 \Leftrightarrow (\exists P_i \in Sortie(S_1), \exists P_j \in Entrée(S_2) : P_i \preceq P_j)$$

  où *Sortie(S)* et *Entrée(S)* désignent respectivement l'ensemble des ports de sortie et d'entrée du service *S*.

  La connectivité syntaxique précédente est directe et il existe une connectivité indirecte entre deux services S_1 et S_2 notée par $S_1 \preceq^*_p S_2$ et qui définie par :

$$S_1 \preceq^*_p S_2 \Leftrightarrow (\exists P_i \in Sortie(S_1), \exists P_j \in Entrée(S_2), \exists \theta : P_i \mapsto P_j : \theta(P_i) \preceq P_j)$$

  où θ *désigne une transformation syntaxique de ports.*

- la *connectivité sémantique* de services : deux services S_1 et S_2 sont sémantiquement connectables (connexion relativement aux ports) que nous notons $S_1 \sqsubset_p S_2$ si et seulement s'il existe au moins un port de sortie de S_1 qui est sémantiquement compatible avec un port d'entrée de S_2, c'est à dire :

$$S_1 \sqsubset_p S_2 \Leftrightarrow (\exists P_i \in Sortie(S_1), \exists P_j \in Entrée(S_2) : P_i \sqsubset P_j).$$

VIII.4.4.2. Médiation fonctionnelle

VIII.4.4.2.1. Approches classiques de médiation fonctionnelle

La médiation fonctionnelle (function mediation) constitue un aspect qui porte sur la l'orchestration et la réutilisation des fonctions des services d'entreprise. Traditionnellement, cet aspect est pris en charge par des outils d'orchestration tels que les langages BPEL4WS et WSFL (cf. § VI.8) ou encore par certains outils d'intégration tels que les process broker (cf. § VI.7). Ces derniers permettent de résoudre certaines hétérogénéités syntaxiques liées aux fonctions. Au niveau sémantique, cet aspect constitue un domaine de recherche encore peu investigué. Il s'apparente à la notion de médiation de processus introduite dans l'approche WSMO (cf. § IV.7.3) qui s'intéresse aux processus de collaboration inter-entreprise, mais à un niveau plus élémentaire, celui des services fonctionnels. Dans ce qui suit, nous allons proposer notre approche pour la médiation fonctionnelle.

VIII.4.4.2.2. Approche ODSOI de médiation fonctionnelle

Dans notre approche, un service fonctionnel peut invoquer un ou plusieurs autres services. L'ordre d'invocation de ces services est défini dans des orchestrations qui se présentent sous forme de services fonctionnels composites ou sous forme de services processus. Ces orchestrations permettent de définir toutes les interactions qui peuvent exister entre les services fonctionnels élémentaires mis en jeu.

Si un service fonctionnel est dynamiquement découvert, il se pourrait que sa chorégraphie ne soit pas totalement compatible avec celle définie au sein de l'orchestration supportée par les services fonctionnels composites ou les services processus. Dans ce cas, on met en oeuvre la médiation fonctionnelle afin d'invoquer le service découvert ou plus précisément la fonction du service découvert. La médiation fonctionnelle consiste alors à déterminer si deux fonctions données sont connectables, d'invoquer dans ce cas précis la fonction à connecter et de déclencher le processus de substitution dans le cas contraire.

Ainsi, dans notre approche, la médiation fonctionnelle est appréhendée sous l'angle de la compatibilité et de la connectivité de fonctions de services. Nous avons déjà défini dans la section VIII.4.4.1.4 la notion de connectivité de services. Nous raffinons cette notion en l'appliquant aux fonctions des services. Nous proposons alors les définitions suivantes :

Définition : (connectivité de fonctions)

Il existe une connectivité entre deux fonctions F_1 et F_2 (issues de services distincts ou de services identiques) et nous notons $F_1 \preceq_p F_2$ si et seulement s'il existe au moins un port de sortie de F_1 qui est compatible avec un port d'entrée de F_2.

De même que pour les services, nous définissons la connectivité syntaxique et la connectivité sémantique de deux fonctions.

Définition : (connectivité syntaxique de fonctions)

Deux fonctions F_1 et F_2 sont syntaxiquement connectables (connexion relativement aux ports) et nous notons $F_1 \preceq_p F_2$ si et seulement s'il existe au moins un port de sortie de F_1 qui est syntaxiquement compatible avec un port d'entrée de F_2, c'est à dire :

$$F_1 \preceq_p F_2 \Leftrightarrow (\exists P_i \in Sortie(F_1), \exists P_j \in Entrée(F_2) : P_i \preceq P_j)$$

où *Sortie(F)* et *Entrée(F)* désignent respectivement l'ensemble des ports de sortie et d'entrée de la fonction *F*.

Définition : (connectivité sémantique de fonctions)

Deux fonctions F_1 et F_2 sont sémantiquement connectables (connexion relativement aux ports) et nous notons $F_1 \sqsubset_p F_2$ si et seulement s'il existe au moins un port de sortie de F_1 qui est sémantiquement compatible avec un port d'entrée de F_2, c'est à dire :

$$F_1 \sqsubset_p F_2 \Leftrightarrow (\exists P_i \in Sortie(F_1), \exists P_j \in Entrée(F_2) : P_i \sqsubset P_j)$$

Définition : (compatibilité syntaxique de fonctions)

Soient F_i: $E_i \rightarrow S_i$ (E_i et S_i sont respectivement les ports d'entrée et les ports de sortie de F_i) et F_j: $E_j \rightarrow S_j$ deux fonctions de deux service S_1 et S_2. Nous disons que F_i est syntaxiquement compatible avec F_j (ou F_i est un sous-type fonctionnel de F_j) et nous notons $F_i \preceq_f F_j$ si et seulement si leurs ports respectifs vérifient les conditions suivantes:

1) $Syn(E_i) \preceq Syn(E_j)$

2) $Syn(S_j) \preceq Syn(S_i)$.

Cette définition peut s'écrire de façon plus complète :

$$F_i \preceq_f F_j \Leftrightarrow (\forall (E_i, S_i) \in Entrée(F_i) \times Sortie(F_i), \forall (E_j, S_j) \in Entrée(F_j) \times Sortie(F_j) :$$
$$(Syn(E_i) \preceq Syn(E_j)) \wedge (Syn(S_j) \preceq Syn(S_i)))$$

où *Entrée(F)* et *Sortie(F)* désignent respectivement l'ensemble des entrées et des sorties de la fonction *F*.

Définition : (compatibilité sémantique de fonctions)

Soient F_i: $E_i \rightarrow S_i$ et F_j: $E_j \rightarrow S_j$ deux fonctions de deux service S_1 et S_2. Nous disons que F_i est sémantiquement compatible avec F_j (ou F_i est un sous-concept fonctionnel de F_j) et nous notons $F_i \sqsubset_f F_j$ si et seulement si leurs ports respectifs vérifient les conditions suivantes :

1) $Sem(E_i) \sqsubset Sem(E_j)$

2) $Sem(S_j) \sqsubset Sem(S_i)$

3) $Concept\_F(F_i) \sqsubset Concept\_F(F_j)$.

Cette définition peut s'écrire de façon plus formelle:

$$F_i \sqsubset_f F_j \Leftrightarrow (\forall (E_i, S_i) \in Entrée(F_i) \times Sortie(F_i), \forall (E_j, S_j) \in Entrée(F_j) \times Sortie(F_j) :$$
$$(Concept\_F(F_i) \sqsubset Concept\_F(F_j)) \wedge (Sem(E_i) \sqsubset Sem(E_j)) \wedge (Sem(S_j) \sqsubset Sem(S_i))$$

où *Concept_F(F)* désigne le concept fonctionnel associé à F et qui est défini au sein de l'ontologie fonctionnelle locale (§ VII.5).

De façon similaire, nous définissons aussi la notion de compatibilité fonctionnelle de services en nous basant également sur les opérateurs définis dans la section précédente (§VIII.4.4.1).

Définition : (compatibilité fonctionnelle de services)

Soient S_1 et S_2 deux services (distincts ou identiques). Nous disons que S_1 et S_2 sont fonctionnellement compatibles relativement aux fonctions F_i et F_j (F_i et F_j étant respectivement des fonctions des services S_1 et S_2) si et seulement si F_i est compatible avec F_j.

De même que pour les ports, nous définissons deux types de compatibilité fonctionnelle de services :

Définition : (compatibilité fonctionnelle syntaxique de services)

Soient S_1 et S_2 deux services (distincts ou identiques). Nous disons qu'il existe une compatibilité fonctionnelle syntaxique entre S_1 et S_2 relativement aux fonctions F_i et F_j et l'on note $S_1 \preceq_f S_2$ si et seulement si F_i est syntaxiquement compatible avec F_j.

$$S_1 \preceq_f S_2 \Leftrightarrow (\exists F_i \in Fonction(S_1), \exists F_j \in Fonction(S_2) : F_i \preceq_f F_j)$$

où *Fonction(S)* désigne l'ensemble des opérations ou des fonctions du service *S*,

et $\preceq_f$ est l'opérateur de compatibilité fonctionnelle syntaxique.

Définition : (compatibilité fonctionnelle sémantique de services)

Soient S_1 et S_2 deux services (distincts ou identiques). Nous disons qu'il existe une compatibilité fonctionnelle sémantique entre S_1 et S_2 relativement aux fonctions F_i et F_j et l'on note $S_1 \sqsubset_f S_2$ si et seulement si F_i est sémantiquement compatible avec F_j.

$$S_1 \sqsubset_f S_2 \Leftrightarrow (\exists F_i \in Fonction(S_1), \exists F_j \in Fonction(S_2) : F_i \sqsubset_f F_j).$$

où $\sqsubset_f$ est l'opérateur de compatibilité fonctionnelle sémantique.

C'est en exploitant les définitions précédentes que nous arrivons à résoudre les hétérogénéités d'ordre fonctionnel. C'est à dire que nous sommes en mesure de décider si deux fonctions F_i et F_j sont liées par une relation d'équivalence, une relation d'ordre (subsomption ou subsomption inverse) ou une relation de disjonction. L'intérêt de cette médiation est que nous pouvons, d'une part remplacer ou substituer toute fonction d'un service par une autre fonction d'un autre service en fonction de leur compatibilité syntaxique et sémantique, et d'autre part d'établir une connexion de ces fonctions en se basant sur leur connectivité.

Plus précisément, le processus de médiation repose sur un principe similaire à celui exposé pour la médiation de données (§ VIII.4.4.1.2.2). Il s'agit alors de favoriser la médiation au niveau sémantique si les fonctions sont sémantiquement décrites. En revanche, on doit se suffire de la médiation au niveau syntaxique et ce conformément au principe de la cohabitation syntaxe-sémantique (§VIII.4.4.1.4).

La figure VIII.27 illustre le principe de la médiation fonctionnelle qui s'effectue de façon relativement similaire à la médiation de données. Dans cette figure, chaque fonction *F* est décrite par une paire *[Syn(F), Sem(F)]*. De même que pour les données, la connectivité de deux fonctions est généralement considérée comme le résultat d'une relation composée τ_f ($\tau_f = \delta_f \circ \varphi_f \circ \alpha_f$) qui permet d'établir sémantiquement une connexion entre deux fonctions données F_i et F_j en effectuant les trois étapes suivantes :

- remonter dans la couche sémantique (grâce à la relation d'abstraction (α_f)) afin de récupérer la sémantique de la fonction F_i,

- calculer la connectivité sémantique (grâce à la relation (φ_f)) pour savoir si les deux fonctions F_i et F_j sont connectables,
- récupérer la syntaxe (grounding) de la fonction F_i pour l'invoquer en appliquant la relation concrétisation syntaxique (δ_f).

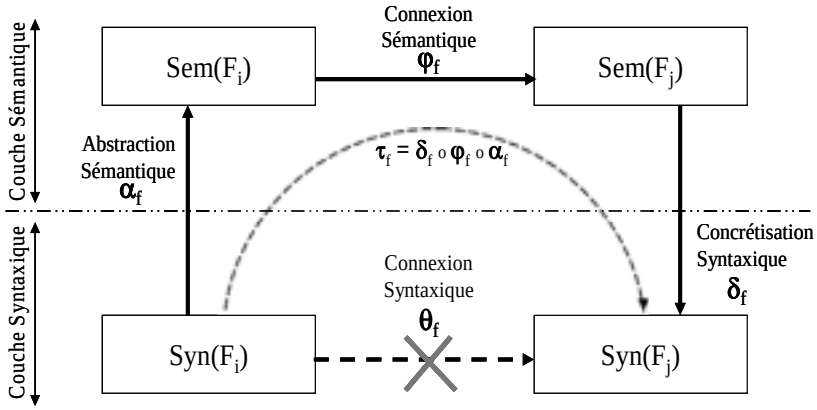

Figure VIII.27. Principe de la médiation fonctionnelle

VIII.4.4.2.3. Service de médiation fonctionnelle

Dans notre approche, la médiation fonctionnelle est prise en charge par un service de médiation particulier qui s'appelle le service de médiation fonctionnelle (Function Mediation Service). Ce dernier est un service d'intégration particulier qui s'active dès réception d'une requête de type :

$$request(\#Mediate, Arguments)$$

où *Arguments* désigne les caractéristiques de la requête de médiation fonctionnelle qui permet ainsi de préciser la fonction source et la fonction cible à intégrer.

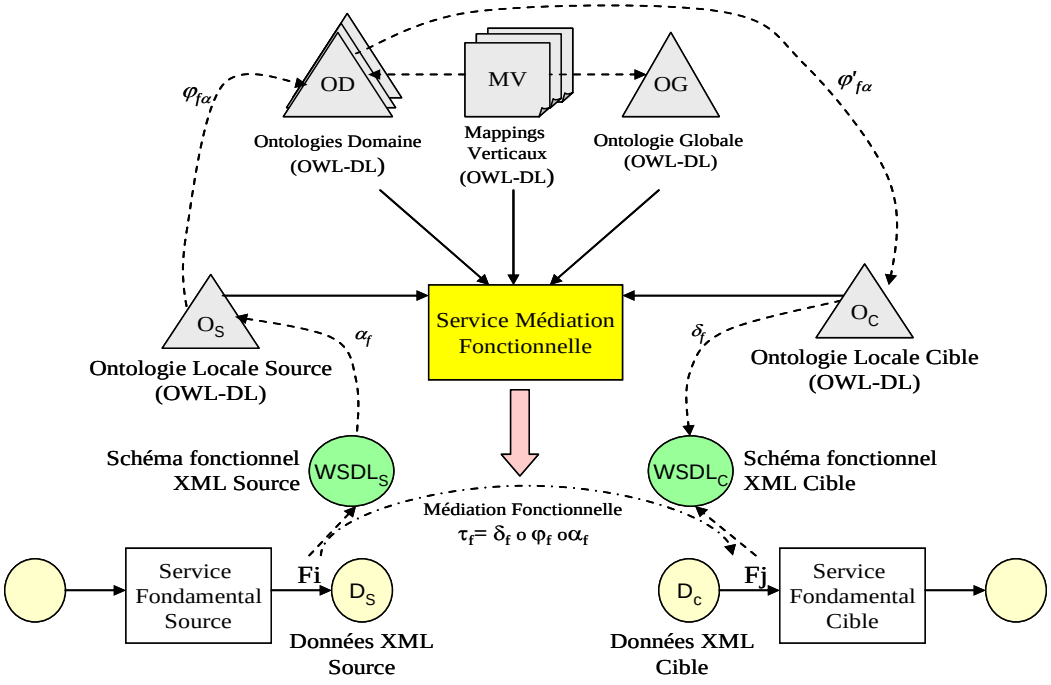

Figure VIII.28. Principe du service de médiation fonctionnelle

La figure VIII.28 illustre le principe de fonctionnement de notre service de médiation fonctionnelle. Dès qu'un service sollicite l'exécution d'une certaine fonction d'un autre service, le service de médiation intervient pour établir une connexion sémantique telle que définie précédemment. Cette médiation fonctionnelle se base sur les aspects syntaxiques et sémantiques définis dans les chapitres précédents (chapitre VI et VII). Elle se base en particulier sur les ontologies fondamentales définies auparavant ainsi que sur les mappings sémantiques et syntaxiques.

VIII.4.4.2.4. Cohabitation syntaxe-sémantique

De même que pour les données, le processus de médiation fonctionnelle est en réalité plus complexe. On lui applique aussi le principe de cohabitation syntaxe-sémantique afin d'apporter plus de flexibilité à notre approche de médiation fonctionnelle. Aussi, dans le cas de services qui ne sont que syntaxiques (décrits que syntaxiquement), on se base sur la connectivité syntaxique pour pouvoir effectuer la médiation fonctionnelle. La figure VIII.29 résume la cohabitation de ces deux approches.

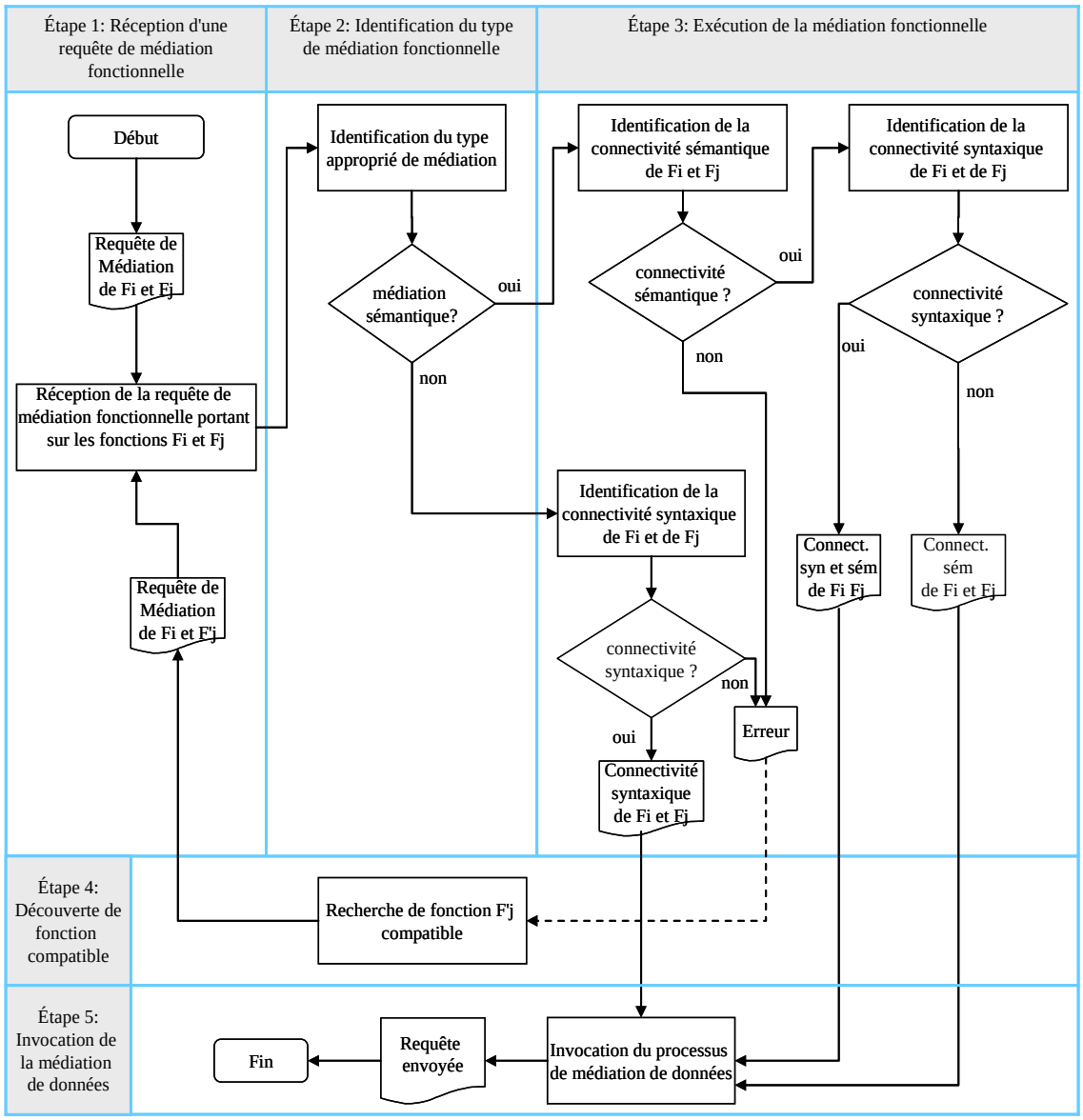

Figure VIII.29. Processus global de la médiation fonctionnelle

Comme le montre la figure VIII.29, le processus global de médiation fonctionnelle repose sur la notion de connectivité syntaxique et /ou connectivité sémantique. A une requête de médiation fonctionnelle, le processus effectue cinq étapes qui sont :

- *étape 1 - réception d'une requête de médiation fonctionnelle* qui permet de recevoir la requête de médiation portant sur les fonctions F_i et F_j;
- *étape 2 - identification du type de médiation* qui permet de choisir le type de médiation (syntaxique ou sémantique) approprié en fonction du degré de maturité des services (cf. § VIII.4.2.2) mis en jeu;
- *étape 3 - exécution de la médiation fonctionnelle* qui permet d'identifier le degré de connectivité des deux fonctions. Les deux types de connectivité précédemment introduits (connectivité syntaxique et connectivité sémantique) sont pris en charge;
- *étape 4 - découverte de fonction compatible* qui permet, dans le cas où l'étape précédente se termine par un échec, de lancer le processus de substitution afin de découvrir d'autres fonctions compatibles pour un éventuel remplacement de la fonction sollicitée;
- *étape 5 - invocation de la médiation de données* qui permet de lancer le service de médiation de données défini dans la section VIII.4.4.1.3 afin de mettre en œuvre l'invocation de la fonction connectable sollicitée.

VIII.5. Conclusion

Nous avons présenté dans ce chapitre une approche de construction de l'architecture d'intégration d'entreprise permettant de définir des services d'intégration pouvant fournir des mécanismes pour intégrer les services d'entreprise (définis au chapitre VI) en se basant sur des aspects sémantiques (construits au chapitre VII). Il a permis de présenter la façon dont les services d'intégration sont construits, en répondant ainsi à la problématique P_{Int} (cf. § V.2.3) qui posait la question d'intégration sémantique des services d'entreprise.

Notre architecture d'intégration comprend quatre principaux types de services d'intégration qui sont le service brokering, le service de publication, le service de découverte et le service de médiation (cf. § VIII.3.1). Le service brokering est un service dédié, permettant la gestion du processus global d'intégration (cf. § VIII.4.1). Le service de publication permet d'effectuer la publication sémantique (et syntaxique) des services d'entreprise (cf. § VIII.4.2). Le service de découverte permet d'effectuer des recherches sémantiques (et syntaxique) des services d'entreprise (cf. § VIII.4.3). Enfin le service de médiation permet de résoudre les hétérogénéités sémantiques (et syntaxiques) liées aux services d'entreprise (cf. § VIII.4.4).

L'architecture proposée répond et comble certaines limites concernant les technologies des services sémantiques en général et celles de l'approche OWL-S (cf. § IV.7.1) en particulier. Précisément, notre architecture a proposé un enrichissement du mécanisme de publication et de découverte de services pouvant être mis en œuvre dans le cadre d'intégration intra-entreprise. Elle a également proposé des approches de médiation de données et de fonctions qui reposent sur les ontologies de données (cf. § VII.4.2.4) et les ontologies fonctionnelles (cf. § VII.4.2.2) définies au chapitre VII. De plus, notre architecture a proposé une approche flexible de mise en œuvre du processus d'intégration (cf. § VIII.3.2) basée sur la notion de règles d'intégration (cf. § VIII.4.1.2.2). Enfin, notre architecture apporte de la flexibilité dans la mise en œuvre du

processus d'intégration du fait qu'elle repose sur le principe de cohabitation syntaxe-sémantique (cf. § VIII.4.4.1.4). Ce principe qui est fondamental au sein des entreprises industrielles, permet de prendre en compte le caractère continu et incrémental du processus de l'intégration.

Afin de mettre en pratique l'ensemble des éléments définis dans ce chapitre et aussi les éléments définis dans les deux chapitres précédents, nous nous proposons dans le chapitre suivant de réaliser un prototype d'intégration portant sur l'intégration de quelques services d'entreprise dans le milieu industriel.

PARTIE 3

PROTOTYPAGE

Chapitre IX

IMPLEMENTATION ET EXPERIMENTATION

IX.1. Introduction

Les trois chapitres précédents (chapitres VI, VII et VIII) ont permis de répondre à un certain nombre de préoccupations portant sur l'intégration des services d'entreprise. Plus précisément, le chapitre VI a présenté des réponses à la problématique P_{Syn} de construction de l'architecture de services d'entreprise (cf. § V.2.1). Le chapitre VII a apporté des réponses pour la problématique P_{Sem} de construction de l'architecture sémantique (cf. § V.2.2). Enfin, le chapitre VIII s'est intéressé à la problématique P_{Int} d'intégration sémantique des services d'entreprise (cf. § V.2.3). A présent, nous allons nous focaliser dans ce chapitre sur les aspects liés à l'implémentation de notre prototype et aussi aux diverses expérimentations effectuées dans le cadre du déploiement de ce prototype. Aussi, la section IX.2 exposera les principaux objectifs du prototype. La section IX.3 présentera les éléments principaux de l'implémentation du prototype. Enfin, la section IX.4 décrira les expérimentations que nous avons réalisées sur un cas réel.

IX.2. Objectifs principaux du prototype

L'objectif du prototype que nous désirons développer est double (figure IX.1). Tout d'abord, il s'agit de valider au travers d'une implémentation certains éléments du cadre méthodologique d'intégration que nous avons proposés. Ensuite il s'agit d'effectuer un plan d'expérimentations prouvant la pertinence de notre approche sur un cas d'étude réel.

Plus précisément, l'implémentation du prototype a pour but de valider :

- la construction de la couche de services en proposant une implémentation d'un référentiel permettant de gérer les services d'entreprise et en proposant aussi une implémentation de l'algorithme de clustérisation permettant de déterminer les clusters de services,
- la sémantification des services en implémentant certains modules permettant de guider la gestion de la sémantique des services d'entreprise,

- et les mécanismes d'intégration en implémentant certaines fonctionnalités permettant de prendre en charge la découverte et la médiation de services.

Quant à l'expérimentation, elle porte sur un scénario typique d'intégration portant sur une application dans le domaine de la maintenance préventive des équipements de production dans le secteur de la microélectronique.

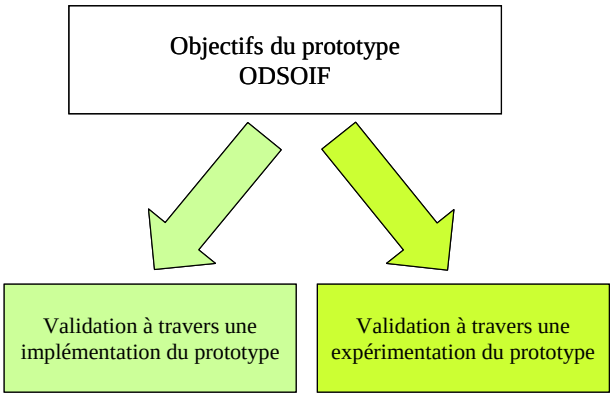

Figure IX.1. Objectifs du prototype

IX.3. Implémentation du prototype

Cette section va présenter les différents éléments qui caractérisent le développement de notre prototype. Après avoir rappelé les fonctionnalités principales du prototype, nous présenterons l'architecture technique générale du prototype et ensuite nous détaillerons l'implémentation des principaux modules développés.

IX.3.1. Fonctionnalités du prototype

Afin de répondre aux objectifs précédents décrits à la section IX.2, nous avons décidé d'implémenter au sein du prototype les fonctionnalités principales correspondant aux cas d'utilisation suivants (figure IX.2):

- *Création de projet d'intégration* qui a pour but de créer en design-time la couche de services d'entreprise (chapitre VI) et la couche sémantique (chapitre VII). Cette fonctionnalité comporte deux sous-cas d'utilisation qui sont :
    - *serviciser* qui permet de construire la couche de services et qui inclut les cas d'utilisation *:*
        - *définir le service* qui permet d'identifier le service et de le relier aux composants de base existants du système d'information (cf. § VI.4.3 et § VI.4.4),
        - *publier syntaxiquement le service* qui permet d'effectuer une publication syntaxique au sein d'un référentiel UDDI (cf. § VI.4.6),
        - *rechercher syntaxiquement le service* qui permet d'effectuer des découvertes syntaxiques de services (cf. § VIII.4.3.3),
        - et *urbaniser les services* qui permet de lancer l'algorithme de clustérisation pour définir des clusters de services (cf. § VI.4.5),

  - *sémantifier* qui permet de construire la couche sémantique ou ontologique de l'entreprise. Ce cas d'utilisation inclut les cas suivants :
    - *construire l'ontologie fondamentale* qui a pour but de guider le concepteur dans le processus de création d'ontologies fondamentales (cf. § VII.6.3),
    - *construire les mappings* qui a pour but de guider le concepteur dans le processus de création des mappings syntaxiques ou des mappings sémantiques (cf. § VII.6.4 et § VII.6.5),
    - *décrire la sémantique des services* qui a pour but de remplir les ontologies OWL-S+ qui constituent des descriptions sémantiques des services d'entreprise (cf. § VII.6.6 et § VII.6.7),
    - *construire la méta-ontologie* qui permet de créer des profils d'ontologies destinés à la publication des ontologies fondamentales (cf. § VII.6.8 et § VIII.4.2).
- *Intégrer les services* qui a pour but d'exécuter certains mécanismes d'intégration tels que définis au chapitre VIII. Ce cas d'utilisation inclut les cas suivants:
  - *publier sémantiquement le service* qui a pour but d'effectuer la publication du service au sein d'un référentiel sémantique (cf. § VIII.4.2),
  - d*écouvrir sémantiquement des services* qui a pour but de rechercher des services en se basant sur la sémantique (cf. § VIII.4.3),
  - et *effectuer la médiation sémantique de services* qui a pour but de mettre en œuvre l'intégration des services. Ce cas d'utilisation inclut la médiation de données et la médiation fonctionnelle de services (cf. § VIII.4.4).

IX.3.2. Architecture générale du prototype

L'architecture générale du prototype ODSOIF que nous voulons développer comprend les principaux composants suivants (figure IX.3) :

- **Module de conception (design-time) :** Cette partie constitue une boite à outils (toolkit) permettant de fournir certaines fonctionnalités de base servant à gérer un projet d'intégration sémantique. Ce module utilise un référentiel de développement appelé le "référentiel ODSOIF" qui constitue un référentiel de développement de projets d'intégration. Il permet de stocker tous les éléments manipulés par notre prototype. Bien que ce référentiel contienne pratiquement les mêmes informations que celles contenues dans le référentiel de services et dans le référentiel sémantique, sa logique d'utilisation est totalement différente. Il a pour but non pas de publier des services ou des ontologies comme le font les référentiels de services ou de sémantique mais plutôt de servir de base de travail pour construire des projets d'intégration qui peuvent ensuite être utilisés pour alimenter, une fois que les projets sont validés, les référentiels de services et de sémantique. Ce module correspond au cas d'utilisation "gérer le projet d'intégration". Il comporte principalement les modules ou composants suivants :
  - **Module construction de l'architecture de services (ASE):** Il permet de construire l'architecture orientée services avec ses deux sous-architectures complémentaires : la SOA IT et la SOA métier (cf. §VI.3.3) qui permettent

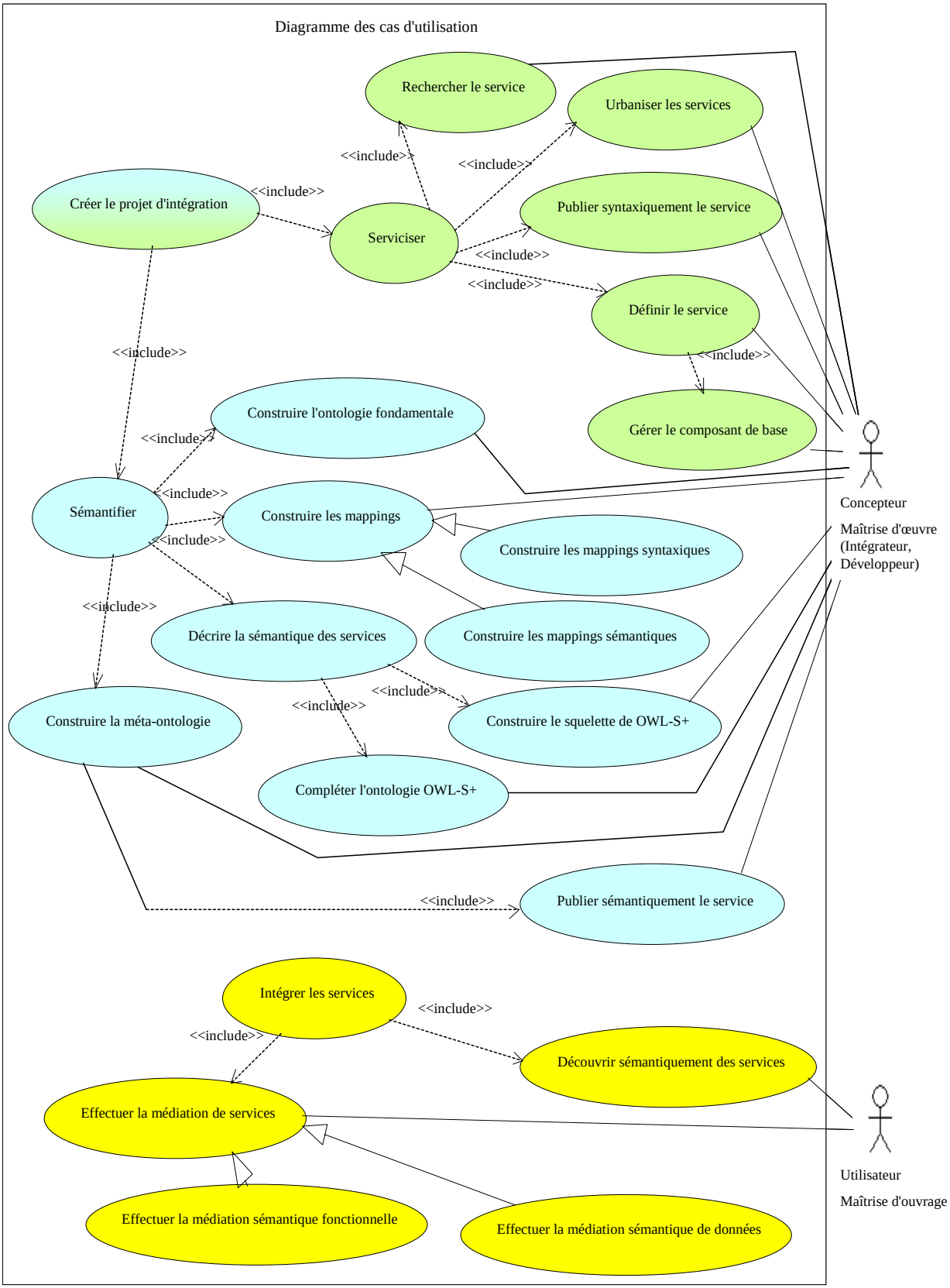

Figure IX.2. Diagramme des cas d'utilisation

d'exposer respectivement les services informatiques et les services métiers (§VVI.3.3). Les composants importants de ce module sont ceux destinés à définir des services (cf. § VI.4.3 et § VI.4.4), à les lier aux composants de base du système d'information existant, à les publier syntaxiquement (cf. § VI.4.6) et à les clustériser (cf. § VI.4.5 et VI.5). Par rapport au diagramme des cas d'utilisation présenté précédemment, ce module correspond au cas "Serviciser".

- **Module de construction de la couche d'ontologies (AOE):** Il construit la couche sémantique pour enrichir sémantiquement les services d'entreprise. Ce module correspond au cas d'utilisation 'Sémantifier' et il comporte les sous-modules suivants :
  - **Service de description sémantique** : Il permet de définir la sémantique de nos services d'entreprise en respectant rigoureusement le modèle de description OWL-S+ (cf. §VII.4.1) que nous avons défini. Ce module correspond au cas d'utilisation "Décrire la sémantique des services".
  - **Module de construction des ontologies fondamentales :** Il correspond au cas d'utilisation "Construire l'ontologie fondamentale" et qui permet de construire et de gérer les multiples ontologies fondamentales d'entreprise (cf. §VII.6.3) selon une vision appropriée basée sur la notion d'urbanisme sémantique tel que défini au chapitre VII,
  - **Module de construction des mappings** : Il permet de construire et de gérer les différents mappings qui peuvent être définis entre les différents domaines (mappings sémantiques et mappings syntaxiques) (cf. §VII.6.4 et §VII.6.5). Ce module correspond au cas d'utilisation "Construire les mappings",
  - **Service de publication sémantique** : Il correspond au cas d'utilisation "Publier sémantiquement le service" et qui permet de publier dans le référentiel sémantique la description sémantique des services (et aussi les ontologies fondamentales) (cf. § VIII.4.2),

- **Module d'exécution (run-time) :** Cette partie permet de fournir un certain nombre de modules qui implémentent les mécanismes d'intégration basés sur la sémantique tels que définis au chapitre VIII. Cette partie comporte les sous-modules suivants:
  - **Service brokering sémantique** : Il correspond au cas d'utilisation "Intégrer les services" et qui représente le module principal permettant d'orchestrer les autres modules de run-time (cf. § VIII.4.1).
  - **Service de découverte sémantique** : Il correspond au cas d'utilisation "Découvrir sémantiquement des services" et qui permet de définir des mécanismes de découverte de services d'entreprise en se basant sur les spécifications données en section VIII.4.3,

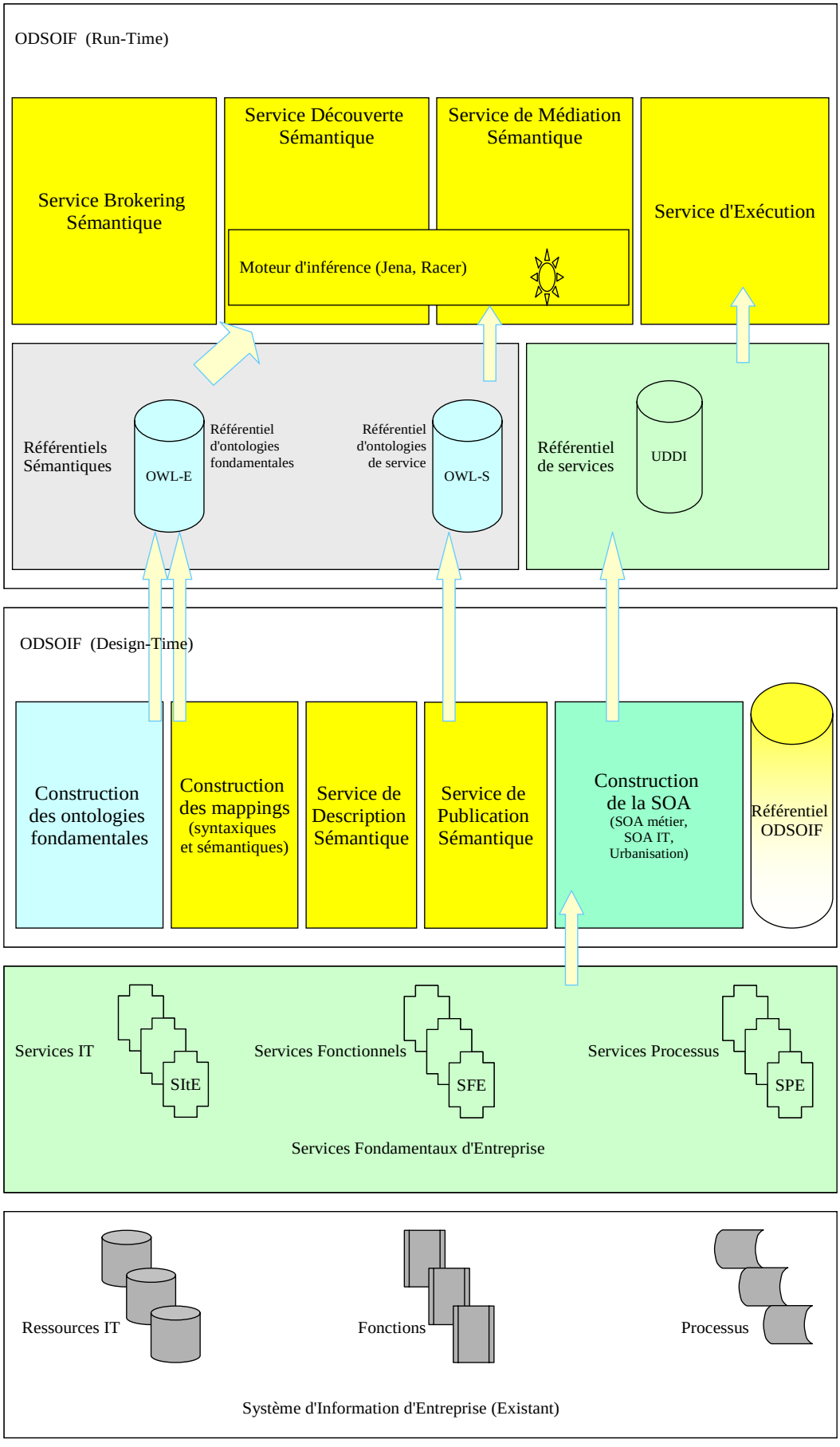

Figure IX.3. Architecture générale du prototype

- **Service de médiation sémantique** : Il correspond au cas d'utilisation "Effectuer la médiation de services" et qui permet de mettre en œuvre les mécanismes liés à l'intégration des services. Ce module comporte en réalité deux sous modules permettant de mettre en œuvre la médiation de données et la médiation fonctionnelle telles que définies en section VIII.4.4.

Dans ce qui suit, nous allons présenter l'implémentation de chacun de ces modules, mais tout d'abord, notons que les modules du prototype ODSOIF sont développés en langage Java, sous l'environnement Eclipse et utilisent un certain nombre d'API et/ou d'outils externes supplémentaires qui sont :

- le SGBD mySQL pour implémenter principalement le référentiel de services (UDDI) et le référentiel d'ontologies. Accessoirement, mySQL est utilisé pour sauvegarder certaines données temporaires, sous forme de tables SQL,
- les wrappers JWS (Java Web Services) de BEA Weblogic pour exposer certains applicatifs en services web,
- la plateforme Axis proposée par la fondation Apache pour l'exécution des web services,
- le serveur d'applications d'Apache Tomcat,
- l'API UDDI4J co-développé par IBM et HP et le registre JUDDI développé par la fondation Apache pour implémenter le registre UDDI et l'interface de publication et de découverte syntaxique,
- l'éditeur OWL-S de SRI pour visualiser ou pour créer manuellement les ontologies OWL-S+,
- l'API OWL-S de SRI pour pouvoir lire, écrire et manipuler les fichiers OWL-S+,
- l'outil et l'API Protégé pour créer, gérer et maintenir les ontologies fondamentales,
- et l'API Jena et l'outil Racer pour implémenter le moteur d'inférence permettant de déterminer le rapprochement entre les différents concepts d'ontologies.

IX.3.3. Module de conception (design-time)

Comme il a été précédemment évoqué, ce module se base sur un référentiel de développement appelé "référentiel ODSOIF". Dans sa version 1.0, ce référentiel est implémenté sous forme d'une ontologie qui intègre l'ensemble des fragments d'ontologies étudiés au chapitre VII. Les éléments de ce référentiel sont visualisés sous forme d'une arborescence, appelée arborescence du projet d'intégration (figure IX.4).

Cette fenêtre constitue une véritable tour de contrôle pour le projet d'intégration. En effet, tous les éléments de notre architecture sont visualisés au sein de cette arborescence et tous les traitements (en design-time) sont pratiquement disponibles à partir de cette arborescence. A titre d'exemple, on peut décrire, comme peut le montrer la figure précédente (figure IX.4), tous les éléments de l'architecture de service, en exécutant les traitements associés aux nœuds et qui sont disponibles par un simple clic sur le bouton gauche de la souris.

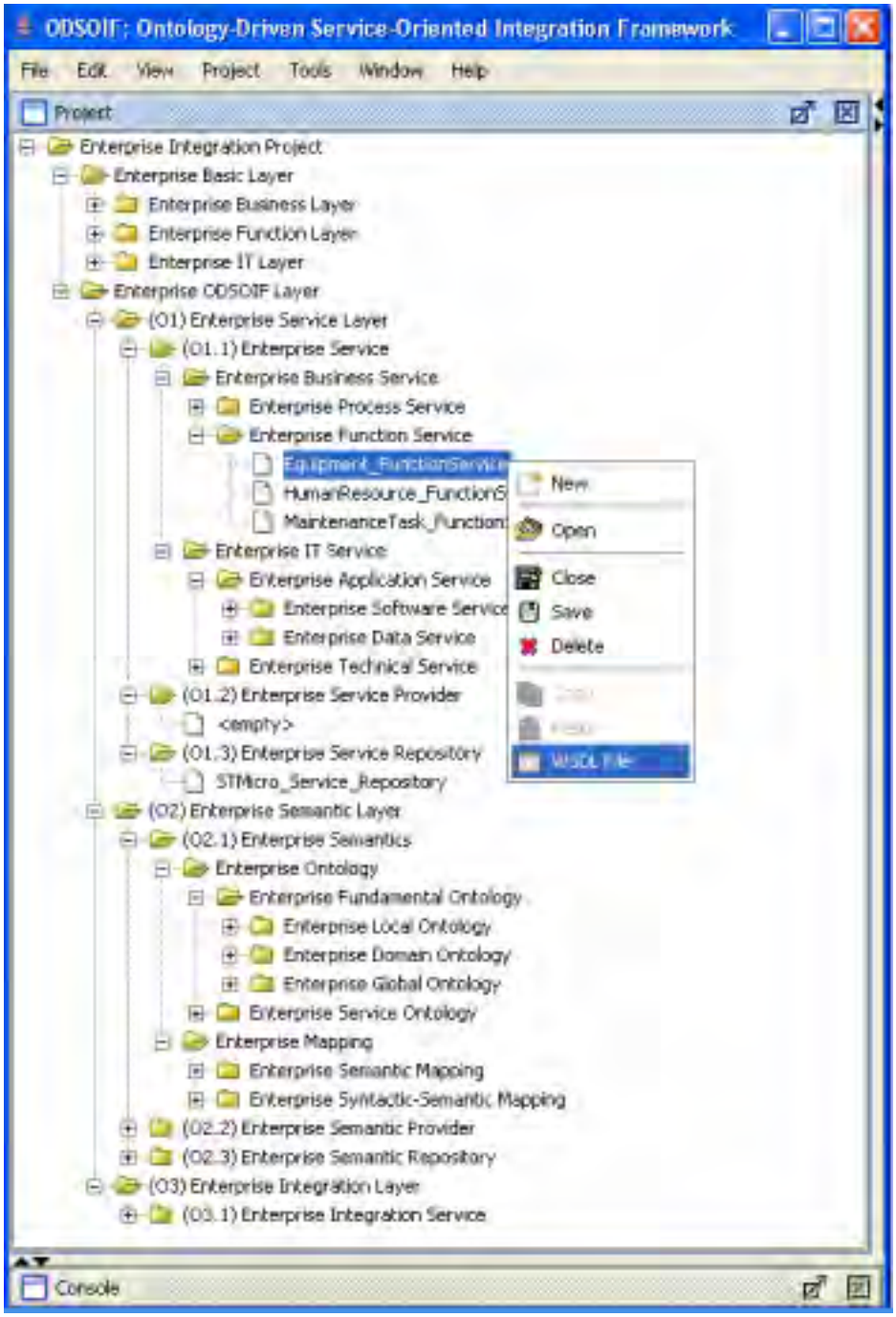

Figure IX.4. Arborescence du projet d'intégration

Afin d'illustrer la structure de notre référentiel de développement, nous fournissons en figure IX.5 un fragment de cette ontologie (référentiel).

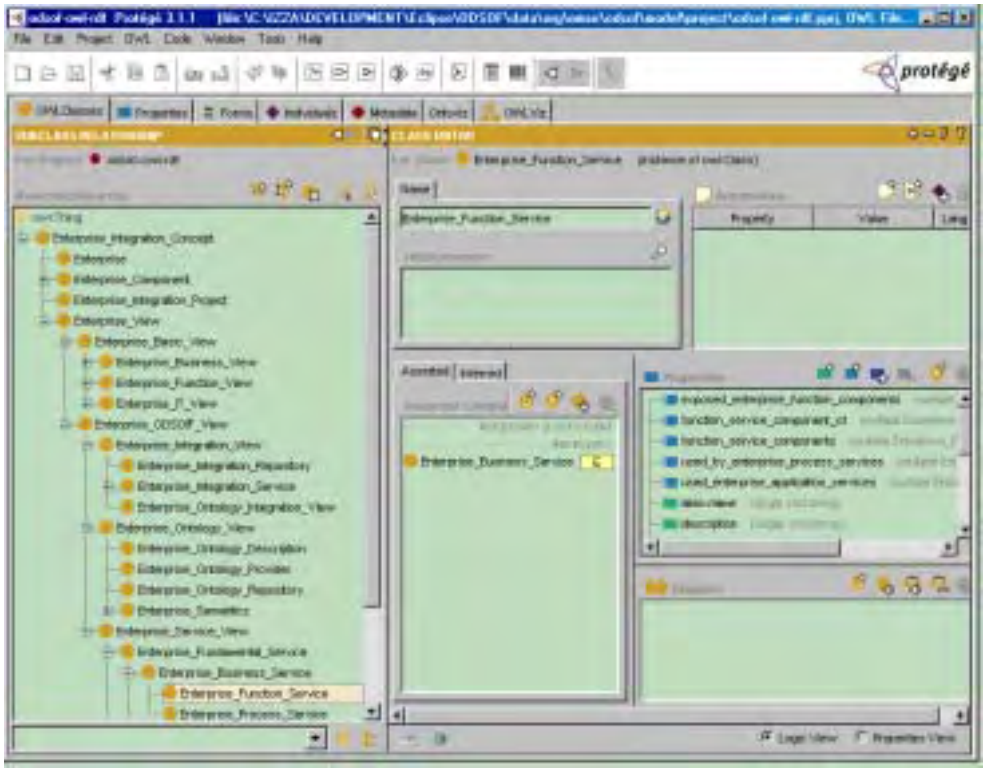

Figure IX.5. Structure du référentiel ODSOIF

Un fragment de notre référentiel est également fourni selon le format OWL en figure IX.6.

```
<?xml version="1.0"?>
<rdf:RDF
    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
    xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
    xmlns:owl="http://www.w3.org/2002/07/owl#"
    xmlns="http://www.owl-ontologies.com/odsof-normal.owl#"
  xml:base="http://www.owl-ontologies.com/odsof-normal.owl">
  <owl:Ontology rdf:about=""/>

  <owl:Class rdf:ID="Enterprise_Integration_Project">
    <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >the integration project</rdfs:comment>
    <rdfs:subClassOf>
       owl:Class rdf:ID="Enterprise_Integration_Concept"/>
    </rdfs:subClassOf>
  </owl:Class>
  <owl:Class rdf:ID="Enterprise_ODSOIF_View">
    <rdfs:subClassOf>
      <owl:Class rdf:ID="Enterprise_View"/>
    </rdfs:subClassOf>
  </owl:Class>
  <owl:Class rdf:about="#Enterprise_Service_View">
    <rdfs:subClassOf rdf:resource="#Enterprise_ODSOIF_View"/>
  </owl:Class>
  ...
</df:RDF>
```

Figure IX.6. Fragment du fichier OWL du référentiel ODSOIF

IX.3.3.1. Module de construction de l'architecture de services (ASE)

Ce module permet de définir, de publier, de rechercher et d'urbaniser les services d'entreprise. La figure IX.7 fournit l'interface générale de ce module. Comme on peut le

constater, on peut effectuer les opérations de base telles que la création, la suppression, la modification et la description (WSDL) de services d'entreprise.

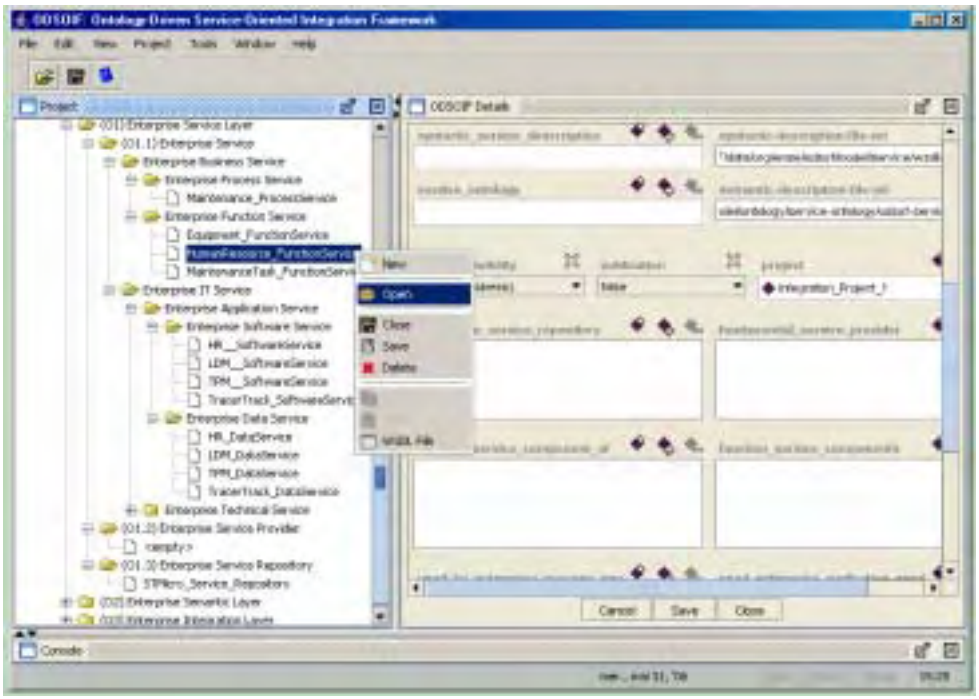

Figure IX.7. Interface de gestion des services d'entreprise

Comme l'architecture de services comporte des référentiels, nous avons implémenté certaines méthodes permettant la publication des services au sein de ces référentiels. La figure IX.8 fournit l'interface générale d'invocation des méthodes liées à la manipulation des référentiels de services, tandis que la figure IX.9 fournit un extrait de code réutilisé pour la publication syntaxique.

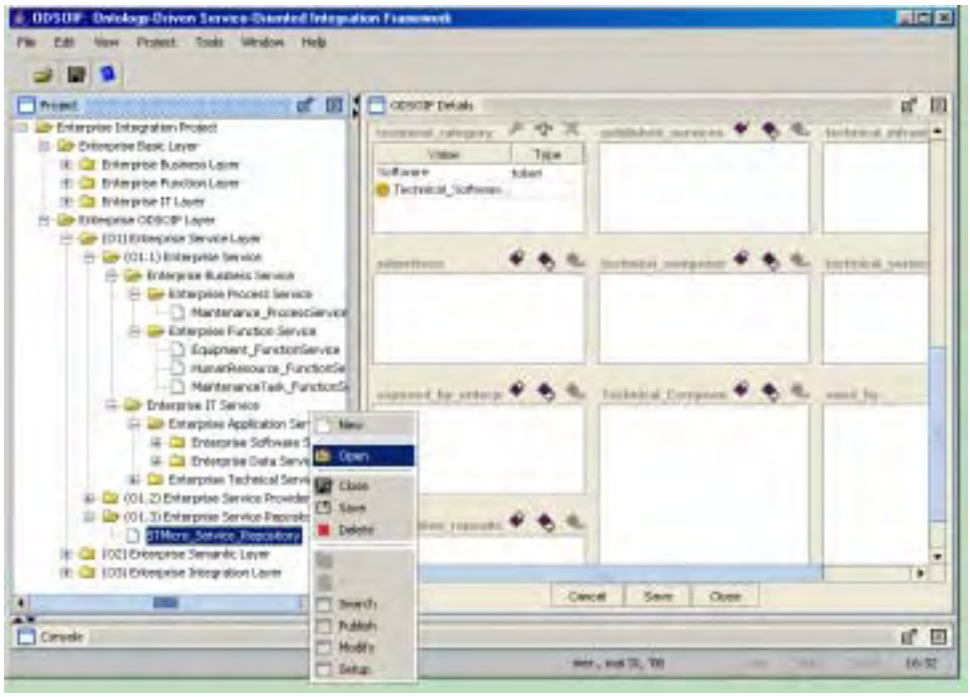

Figure IX.8. Interface de gestion des référentiels de services

```
            /**
            * Publication syntaxique des services d'entreprise
            */
public static BusinessService publishServiceImplementation  (
  String wsdlURL,  String userid,  String password,  String businessKey,
  String tModelKey,  String inquiryURL,  String publishURL)
  throws Exception  {
  Element element;
  ServiceDetail serviceDetail;
  Vector tModelList = new Vector();
  TModelInstanceInfo tModelInstanceInfo;
  TModelInstanceDetails tModelInstanceDetails = new TModelInstanceDetails();
  Vector tModelInstanceInfoList = new Vector();
  InstanceDetails instanceDetails = new InstanceDetails();
  OverviewDoc overviewDoc = new OverviewDoc();

  UDDIProxy uddiProxy = getUddiProxy(inquiryURL, publishURL);
  AuthToken authToken = uddiProxy.get_authToken(userid, password);
  Definition wsdlDefinition = WSDLReader.readWSDL(null, wsdlURL);
  Service wsdlService = ((Service[]) wsdlDefinition.getServices().values().toArray(new
Service[0]))[0];
  // Business Service
  BusinessService businessService = new BusinessService();
  businessService.setBusinessKey(businessKey);
  businessService.setName(wsdlService.getQName().getLocalPart());
  element = wsdlService.getDocumentationElement();
  businessService.setDefaultDescriptionString(DOMUtils.getChildCharacterData(element));
  // Binding Template
  BindingTemplate bindingTemplate = new BindingTemplate();
  Port wsdlPort = ((Port[]) wsdlService.getPorts().values().toArray(new Port[0]))[0];
  element = wsdlPort.getDocumentationElement();
  bindingTemplate.setDefaultDescriptionString(DOMUtils.getChildCharacterData(element));
  ExtensibilityElement ext = (ExtensibilityElement)
  wsdlPort.getExtensibilityElements().get(0);
  AccessPoint accessPoint = new AccessPoint(((SOAPAddress)ext).getLocationURI(), "http");
  bindingTemplate.setAccessPoint(accessPoint);
  tModelInstanceInfo = new TModelInstanceInfo(tModelKey);
  // overview URL
  OverviewURL overviewURL = new OverviewURL(wsdlURL);
  overviewDoc.setOverviewURL(overviewURL);
  instanceDetails.setOverviewDoc(overviewDoc);
  tModelInstanceInfo.setInstanceDetails(instanceDetails);
  tModelInstanceInfoList.addElement(tModelInstanceInfo);
  tModelInstanceDetails.setTModelInstanceInfoVector(tModelInstanceInfoList);
  // tModel -- bindingTemplate
  bindingTemplate.setTModelInstanceDetails(tModelInstanceDetails);
  BindingTemplates bindingTemplates = new BindingTemplates();
  Vector bindingTemplateVector = new Vector();
  bindingTemplateVector.addElement(bindingTemplate);
  bindingTemplates.setBindingTemplateVector(bindingTemplateVector);
  businessService.setBindingTemplates(bindingTemplates);
  CategoryBag categoryBag = new CategoryBag();
  Vector krList = new Vector();
  KeyedReference kr = new KeyedReference("Stock market trading services", "84121801");
  kr.setTModelKey("UUID:DB77450D-9FA8-45D4-A7BC-04411D14E384");
  krList.add(kr);
  categoryBag.setKeyedReferenceVector(krList);
  businessService.setCategoryBag(categoryBag);
  Vector businessServiceVector = new Vector();
  businessServiceVector.addElement(businessService);
  serviceDetail = uddiProxy.save_service(authToken.getAuthInfoString(), businessServiceVector);
  return (BusinessService) serviceDetail.getBusinessServiceVector().elementAt(0);
  }
```

Figure IX.9. Extrait de code pour l'implémentation de la publication syntaxique

Un autre point important de la construction de l'architecture de services est la gestion des composants de base du système d'information existant. En effet, les services définis sont liés à certains composants (applicatifs, sources de données, des fonctions ou des processus), et l'interface donnée en figure IX.10 permet de construire ces liens.

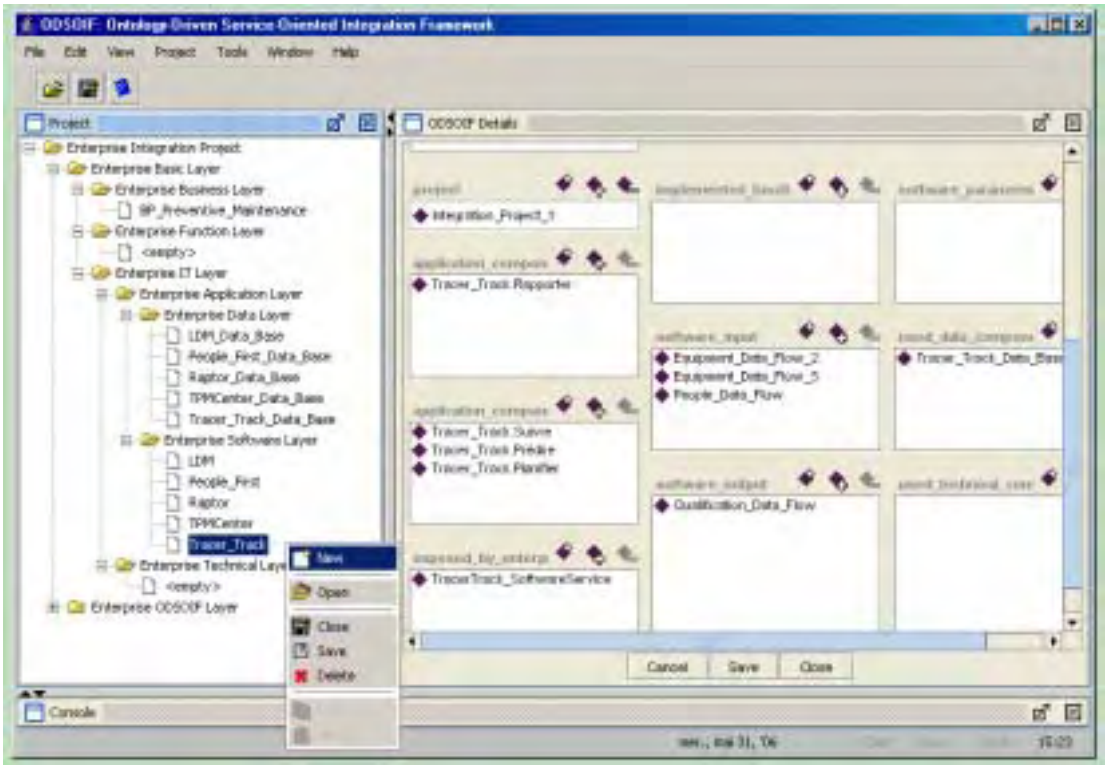

Figure IX.10. Interface de gestion des composants de base

Un dernier aspect pris en charge par le module de construction de l'architecture de services est la clustérisation des services. En effet, ce module offre la fonctionnalité de calcul des degrés de similarité entre les services en se basant sur l'algorithme défini au § VI.5. La figure IX.11 illustre l'interface utilisée pour lancer ce calcul de similarité.

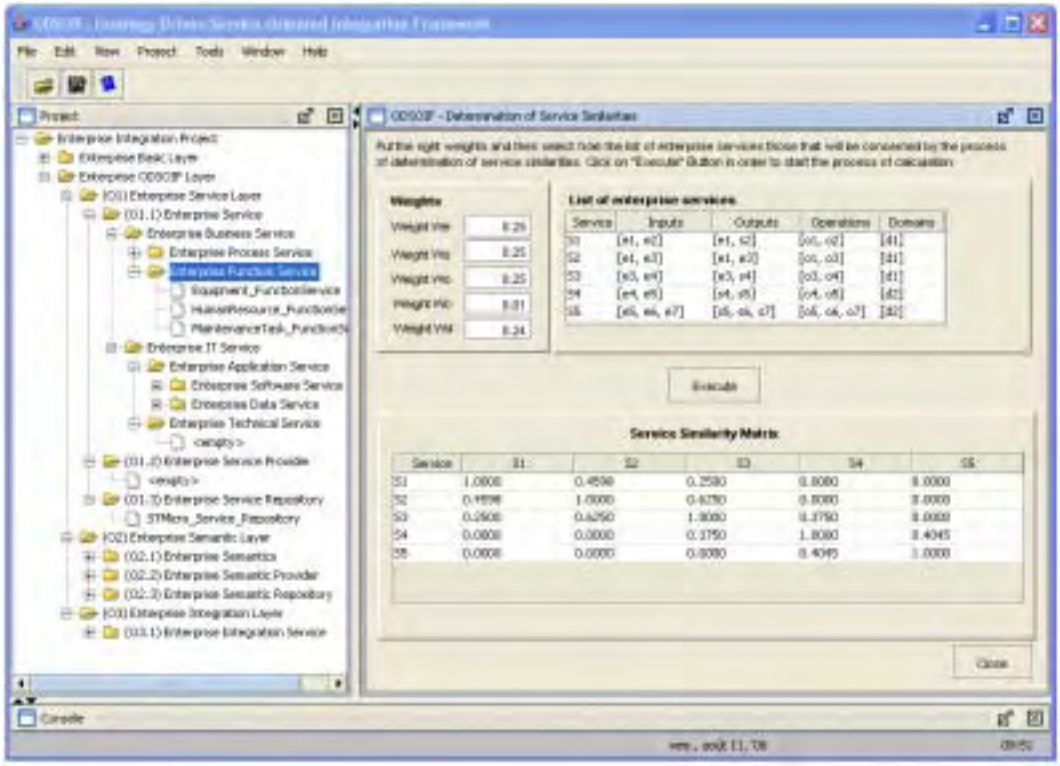

Figure IX.11. Interface pour le calcul de similarité de services

Nous devons préciser que pour l'instant le logiciel de clustérisation utilisé (XLSTAT) ne se lance pas de façon automatique. La raison est toute simple : nous n'avons pas encore développé de module permettant d'intégrer cet outil. Aussi, dans la version 1.0 du prototype, une fois que les degrés de similarité entre les services sont calculés, ces derniers sont introduits manuellement dans le logiciel XLSTAT. Toutefois, des améliorations sont prévues dans les futures versions du prototype.

IX.3.3.2. Module de construction de l'architecture d'ontologies (AOE)

IX.3.3.2.1. Module de construction des ontologies fondamentales

Ce module permet de gérer les ontologies fondamentales d'entreprise. La figure IX.12 montre l'interface de gestion de ces ontologies. Comme on peut le remarquer, on peut effectuer les opérations de base de création, de suppression et de modification d'ontologies.

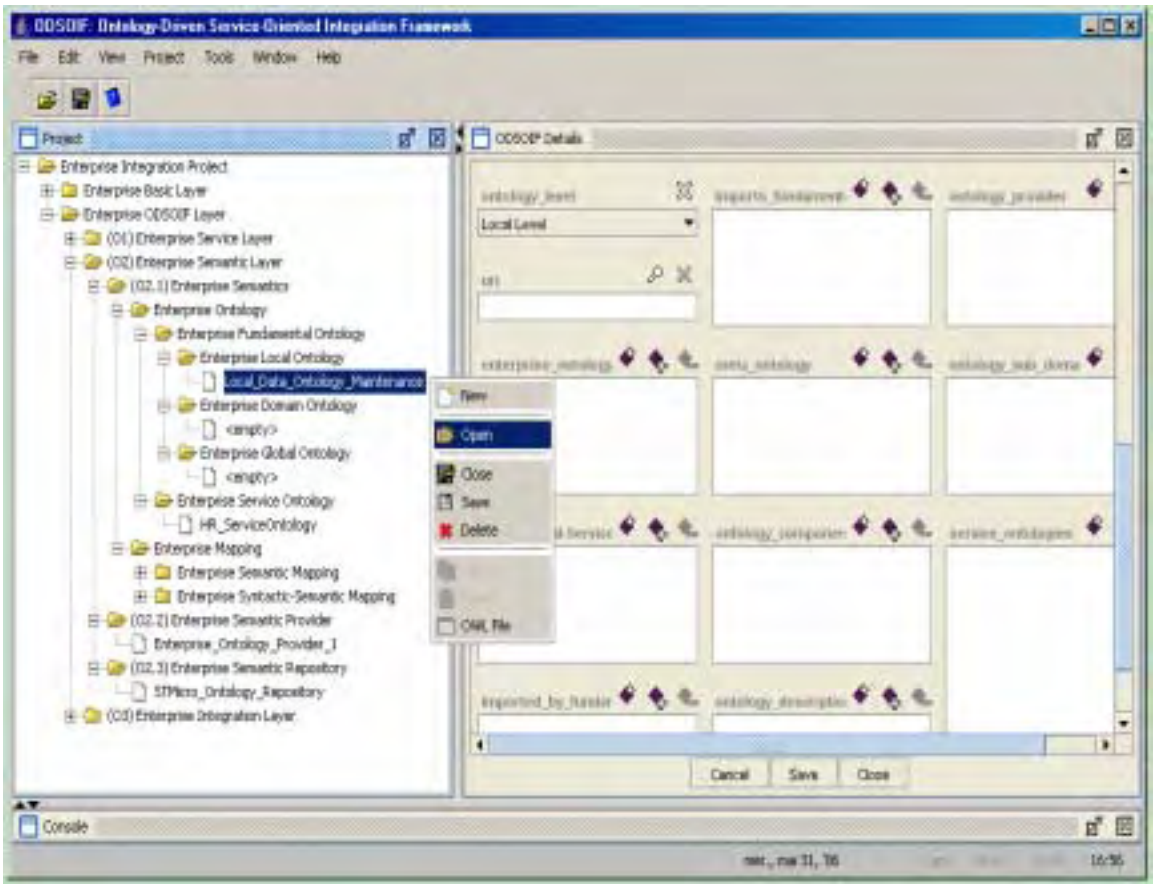

Figure IX.12. Interface du module de construction d'ontologies fondamentales

Nous devons noter que l'exécution de certaines options des menus de l'interface ODSOIF a pour effet de lancer l'éditeur d'ontologie utilisé pour la construction d'ontologies fondamentales. La figure IX.13 fournit un exemple de construction d'une ontologie fondamentale en utilisant l'éditeur d'ontologies Protégé qui constitue l'environnement technique de développement d'ontologies dans notre cas.

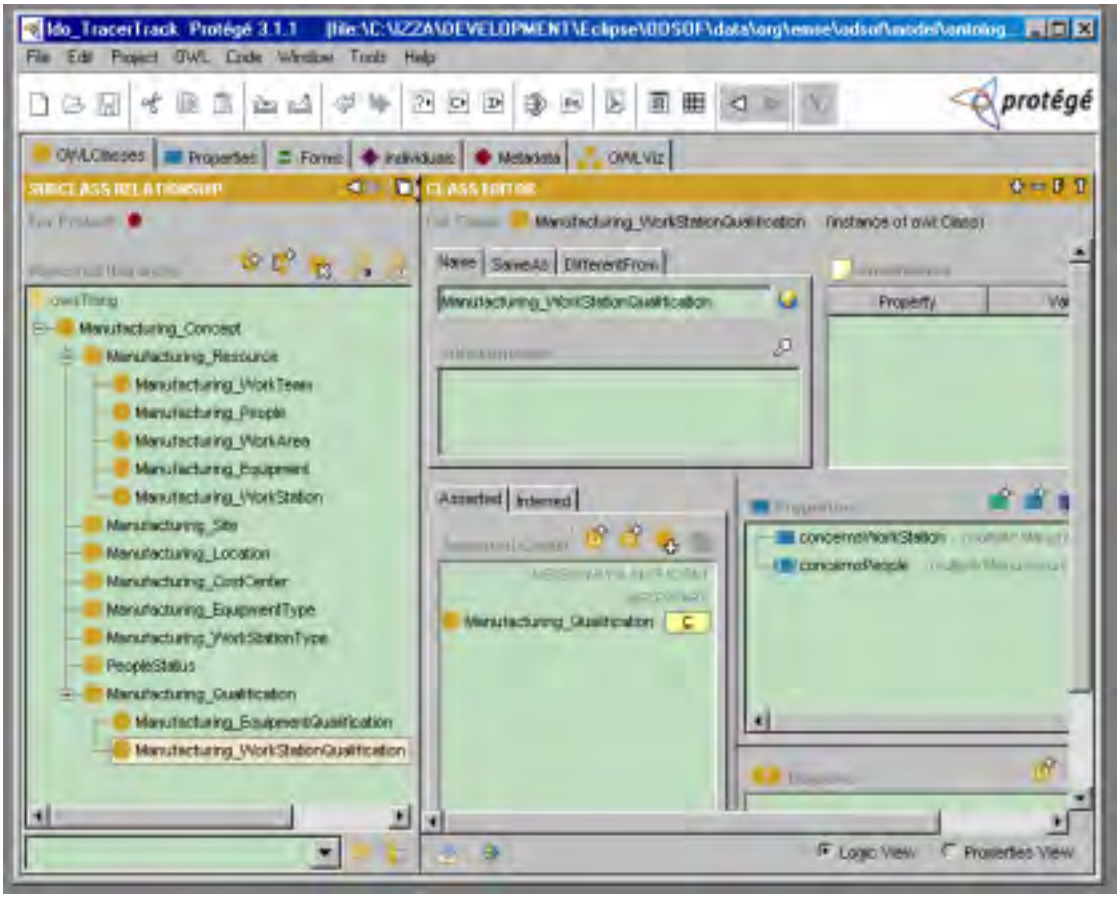

Figure IX.13. Exemple d'éditeur externe pour la construction de l'ontologie fondamentale

IX.3.3.2.2. Module de construction des mappings

Ce module permet de construire des mappings (syntaxiques et sémantiques). Pour ce faire, nous utilisons une interface spécifique permettant d'assister l'utilisateur dans le processus de création de ces mappings. La figure IX.14 représente l'interface du module de création des mappings syntaxiques.

Le fonctionnement typique de cet outil est le suivant :

- Le concepteur sélectionne un service et le système chargera automatiquement le fichier de description syntaxique et l'ontologie fondamentale locale.
- Le concepteur définit alors des mappings en sélectionnant dans le panneau du schéma XML des éléments (qui représentent les types des messages contenus dans WSDL) à mapper avec les éléments à sélectionner dans le panneau de l'ontologie (qui représentent les concepts de l'ontologie fondamentale locale).
- Le concepteur peut également définir des mappings complexes en renseignant le champ XSLT qui permet alors de sauvegarder un fragment de code XSLT (voire de XQuery) à utiliser lors de l'exécution des règles de mappings. L'intérêt de ce code est qu'il permet principalement de définir des correspondances de valeurs, par exemple la conversion d'unités de mesure (convertir des euros en dollars).
- Une fois que les mappings sont validés, le concepteur peut alors les sauvegarder.

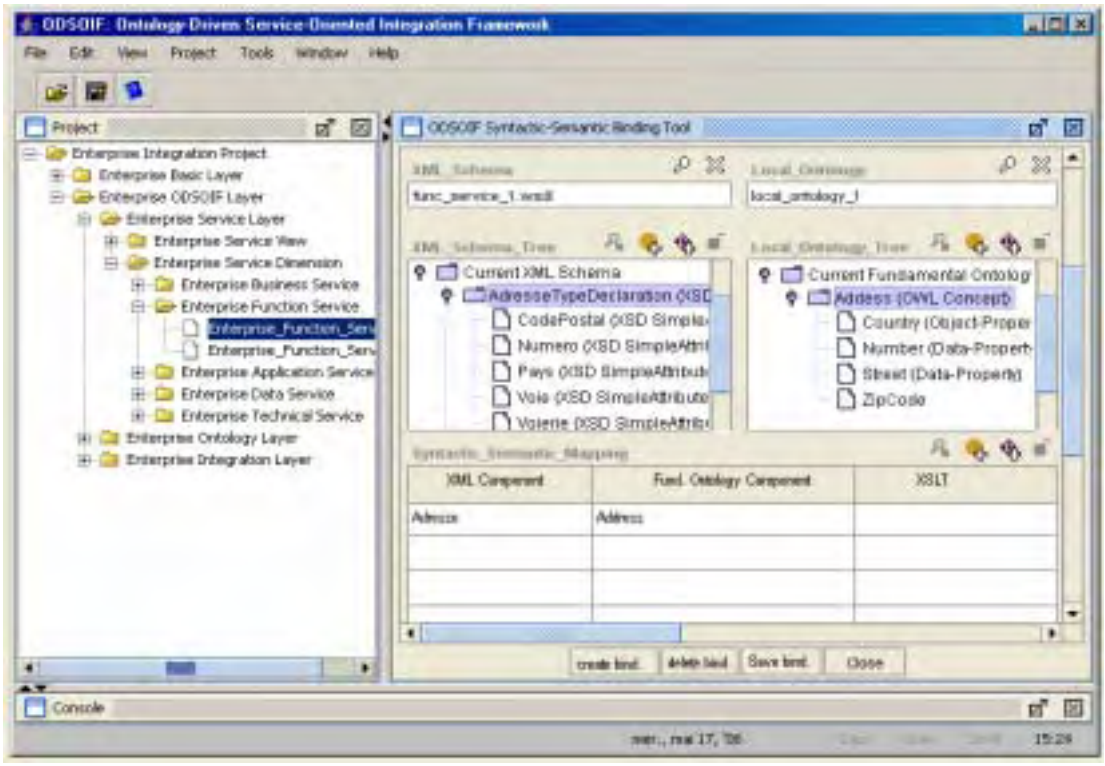

Figure IX.14. Interface de gestion des mappings syntaxe-sémantique

Une interface semblable a été implémentée pour la construction de mappings sémantiques. Comme nous pouvons le constater sur la figure IX.15, le module de construction des mappings sémantiques se base sur la création de mappings verticaux entre les ontologies locales et les ontologies de domaine ou entre les ontologies de domaine et l'ontologie globale.

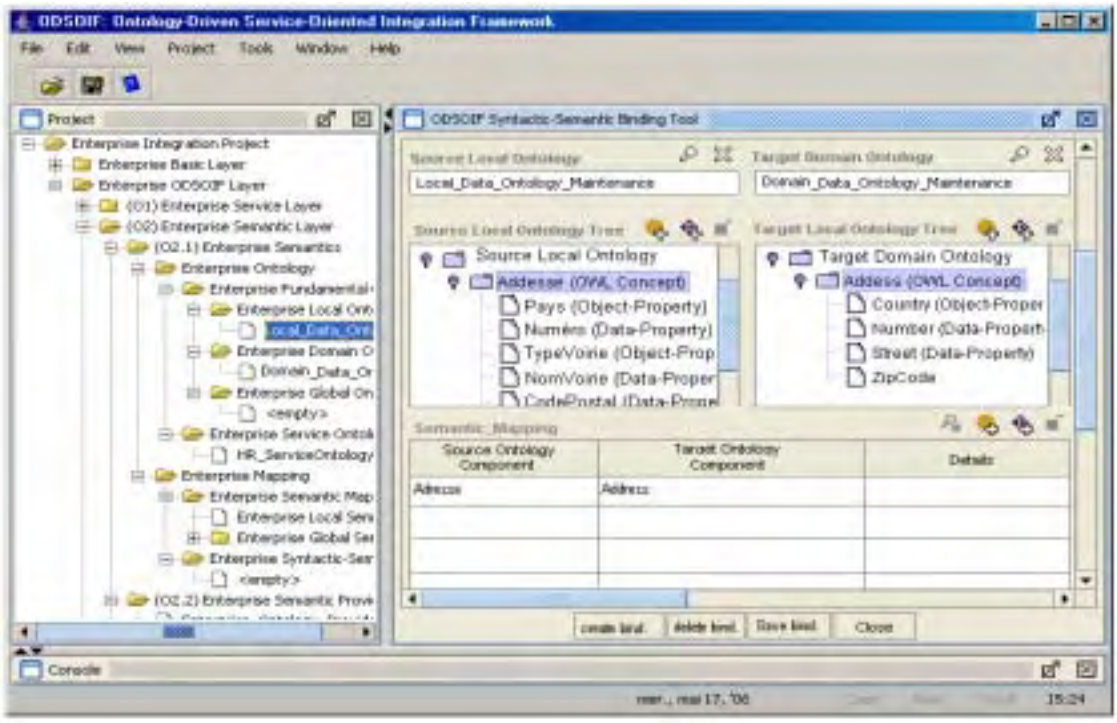

Figure IX.15. Interface de gestion des mappings sémantiques

IX.3.3.2.3. Service de description sémantique

La description sémantique des services repose sur l'ontologie OWL-S+. Rappelons que OWL-S+ constitue une extension de OWL-S. La figure IX.16 montre un fragment d'ontologie OWL-S+ contenant certaines extensions apportées à OWL-S.

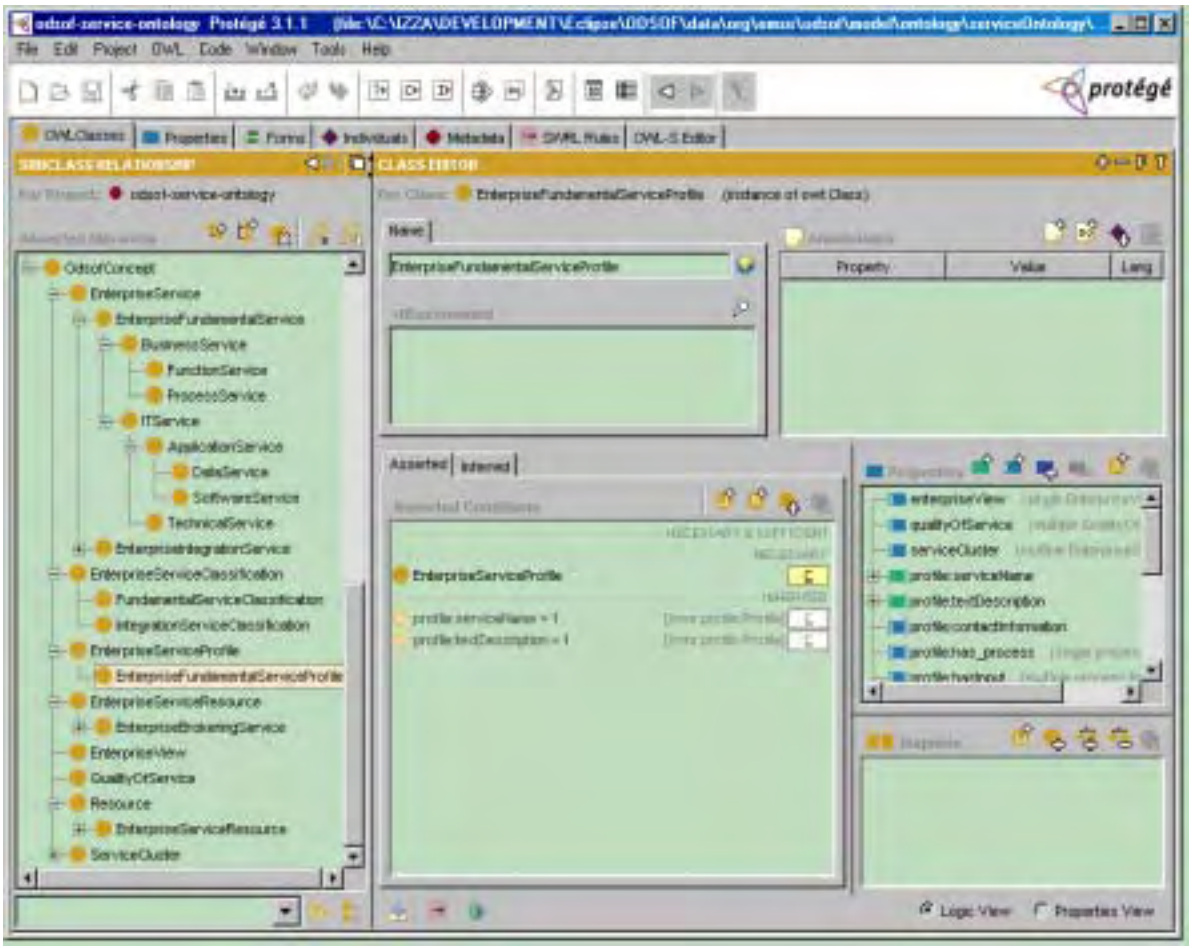

Figure IX.16. Construction de OWL-S+

Une fois que l'ontologie OWL-S+ construite (figure IX.16), nous avons implémenté un module de description sémantique de services. La figure IX.17 montre le principe de la description sémantique des services en utilisant une extension que nous avons définie à l'interface de l'éditeur OWLS de SRI. Cette interface permet de créer manuellement ou de visualiser de manière ergonomique des descriptions sémantiques.

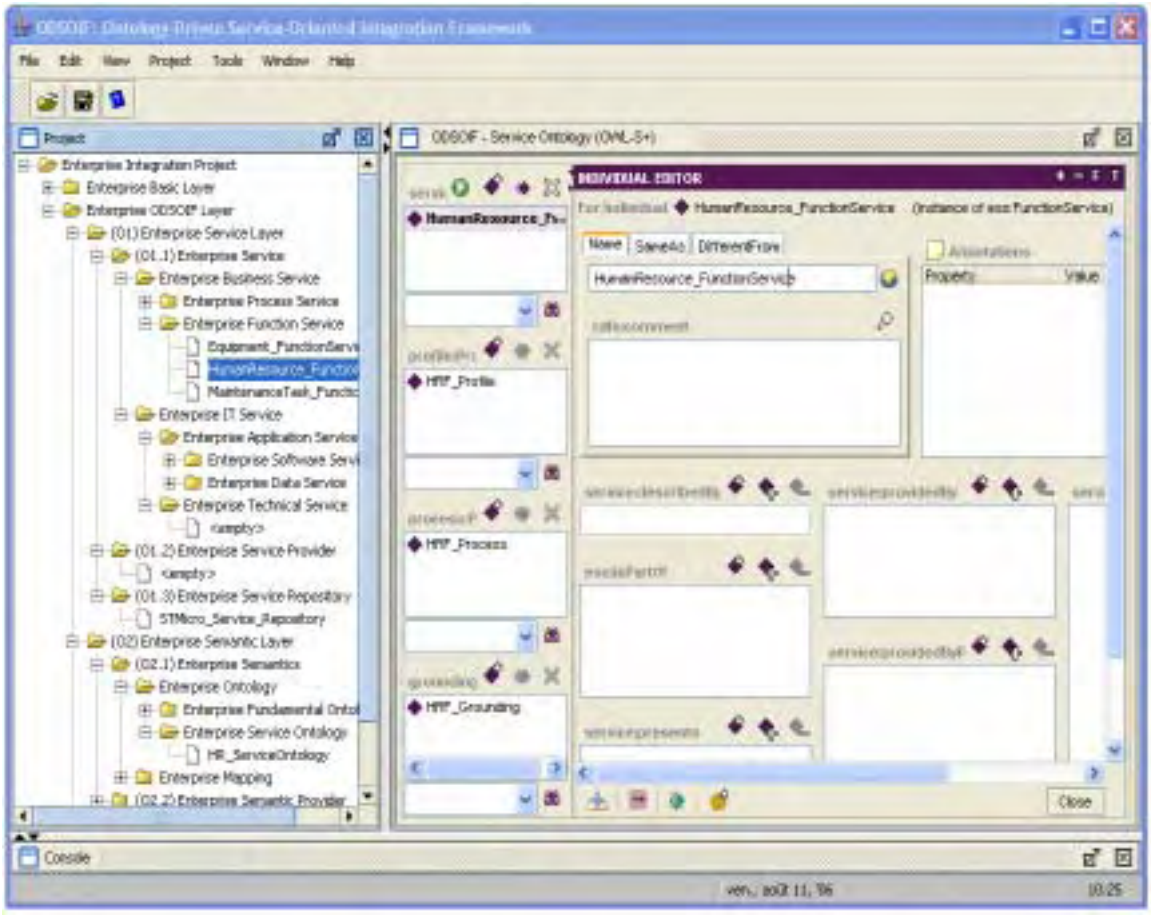

Figure IX.17. Visualisation et création manuelle de descriptions sémantiques

En utilisant cette interface (figure IX.17), le concepteur peut renseigner les différentes caractéristiques du service ou peut aussi utiliser la fonctionnalité semi-automatique "WSDL2OWL-S+" (qui est une extension de celle proposée dans [SemWebcentral 2005]) qui permet de l'assister durant le processus de description en lui proposant une traduction automatique de certains éléments de la description syntaxique WSDL au format OWL-S+. La figure IX.18 montre le principe de fonctionnement du convertisseur WSDL2OWLS+. Le système propose un squelette d'ontologie OWL-S+ qui est le résultat d'une traduction de certains éléments de WSDL au format OWL-S+. Le concepteur complète l'ontologie de services (OWL-S+) : en renseignant les autres caractéristiques de l'ontologie OWL-S+, en établissant les liens entre certains concepts de OWL-S+ avec les autres ontologies fondamentales et enfin en remplissant le grounding de OWL-S+ à partir des mappings syntaxiques précédemment définis (cf. § IX.3.3.2.2).

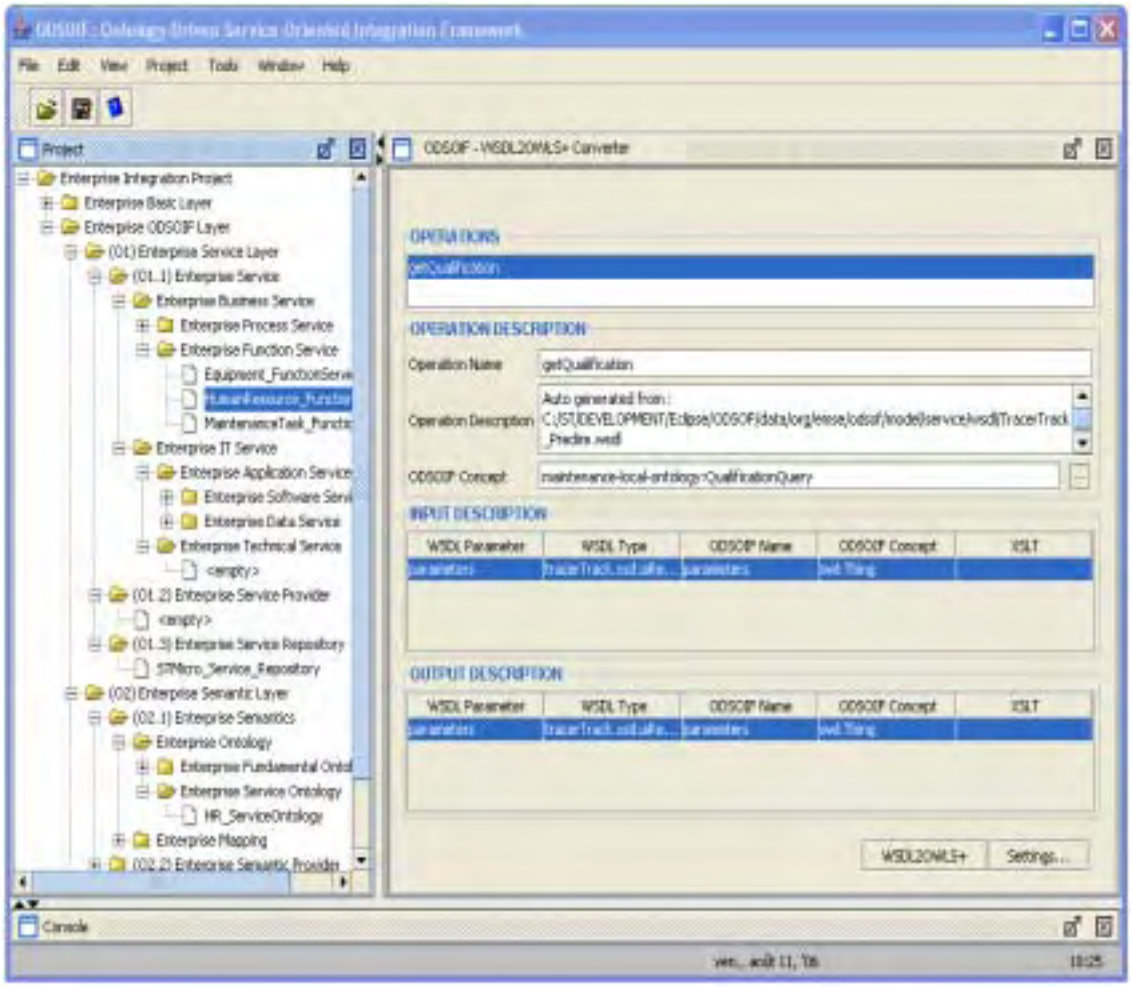

Figure IX.18. Interface du service de description sémantique des services fondamentaux

IX.3.3.2.4. Service de publication sémantique

Le module de publication sémantique est implémenté en deux sous-modules complémentaires : le module de publication des ontologies de services et le module de publication des ontologies. Ces deux modules correspondent respectivement aux deux API introduites au chapitre VIII (cf. § VIII.4.2.2) et qui s'intitulent : *PublishFundamentalService* et *PublishFundamentalOntology*. Pour cela, ces modules de publication sémantique se basent sur l'interface illustrée en figure IX.19.

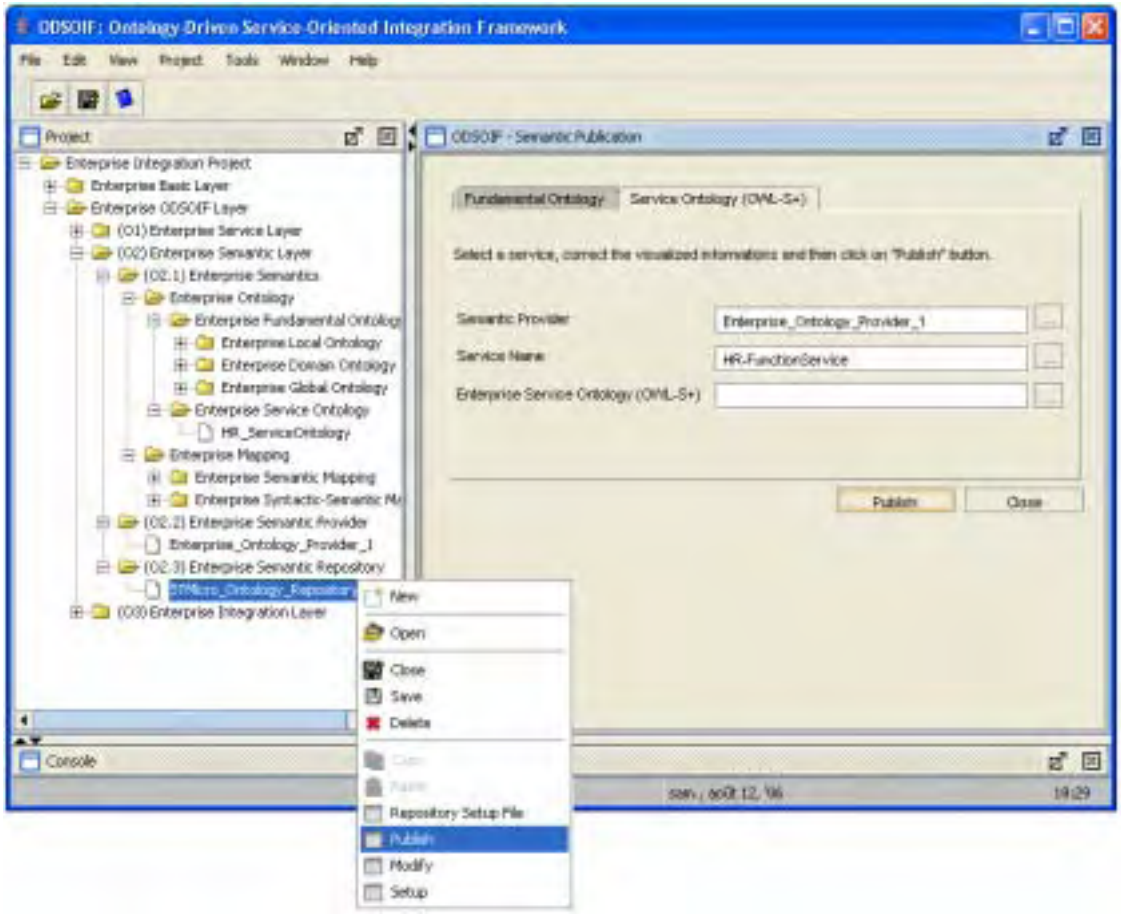

Figure IX.19. Interface pour la publication sémantique

Une fois que les champs de l'interface sont correctement saisis, le concepteur pourra cliquer sur le bouton "Publish" qui va permettre de publier l'ontologie d'entreprise (ontologie de service ou ontologie fondamentale) dans le référentiel d'ontologies.

```
            /**
            * Java beans pour la gestion du référentiel d'ontologies
            *  @version 1.0
            */
public class FundamentalOntologyVo {
    private String name; private String description; private String type;
    private String level; private String owner;
    public FundamentalOntologyVo (String name, String description, String type, String level,
                String owner) {
                this.name = name;
                this.description = description;
                this.type = type;
                this.level = level;
                this.owner = owner; …}
    public void setName(String name) {this.name = name;}
    public String getDescription() {return description;}
    public void setDescription(String description) {this.description = description;}
    public String getType() {return type; }
    public void setType(String type) {this.type = type; }
…}
//---------------------------------------------------------------------
public FundamentalServiceOntologyVo (String serviceName, String textDescription, String
   contactInformation, byte[] serviceOntology) {
   this.serviceName = serviceName;
   this.textDescription = textDescription;
   this.contactInformation = contactInformation;
   this.serviceOntology = serviceOntology;
…}
…
    private Connection  getDbConnection() throws SQLException {
    RDBMS r = null; // RDBMS est une classe permettant de gérer les connexions JDBC-mySQL
    try {
          r = new RDBMS();
        } catch (Exception e) {e.printStackTrace();}
    return (r.getDbConnection());
    }
        }
```

Figure IX.20. Fragment de code des Java Beans pour la gestion du référentiel d'ontologies

Le référentiel d'ontologies est une base de données relationnelle implémentée en utilisant le SGBDR mySQL. La structure de la base de données de ce référentiel se base sur la méta-ontologie d'entreprise (cf. § VII.4.3). Pour manipuler ce référentiel, nous exploitons certains Java Beans dont la figure IX.20 illustre un fragment de code pour la gestion de ces ontologies.

Nous illustrons ci-dessous dans la figure IX.21 un fragment de code Java qui correspond à la méthode *publishFundamentalService()* qui permet de publier une ontologie de service dans le référentiel d'ontologies. Nous illustrons aussi dans la figure IX.22 un fragment de code Java qui correspond à la méthode inverse *getServiceOntology()*[65] exploitée par le service brokering pour récupérer le contenu des ontologies de service.

```
  /**
   * Publication d'une ontologie de service dans le référentiel d'ontologies
   *  @version 1.0
   */

 public void publishFundamentalService(FundamentalServiceOntologyVo fundamentalServiceOntologyVo)
 throws SQLException {
    String sql = "INSERT INTO FundamentalServiceOntology (serviceName,
    textDescription,contactInformation, serviceOntology) VALUES (?,?,?,?)";
    Connection con = getDbConnection();
    PreparedStatement ps = null;
    try {
        ps = con.prepareStatement(sql);
        ps.setString(1, fundamentalServiceOntologyVo.getServiceName());
        ps.setString(2, fundamentalServiceOntoologyVo.getTextDescription());
        ps.setString(3, fundamentalServiceOntologyVo.getContactInformation());
        ps.setBytes(15, fundamentalServiceOntologyVo.getServiceOntology());
        ps.executeUpdate();
    } finally {
    try { ps.close(); } catch (Exception e) { }
    try { con.close(); } catch (Exception e) { }
    }
 }
```

Figure IX.21. Extrait du code de publication d'une ontologie de service

```
/**
* Récupère le fichier OWL-S:Profile depuis le réféntiel d'ontologies
*/
public InputStream getServiceOntology(String serviceName) throws SQLException {
  InputStream myServiceOntology = null;
  String sql = "SELECT * FROM FundamentalServiceOntology WHERE serviceName ="+serviceName;
  Connection con = getDbConnection();
  PreparedStatement ps = null;
  try {
       ps = con.prepareStatement(sql);
       ps.setString(1, serviceName);
       ResultSet rs = ps.executeQuery();
       myServiceOntology = new rs.getBytes("ServiceOntology");
       return myServiceOntology;
  } finally {
  try { ps.close(); } catch (Exception e) { }
  try { con.close(); } catch (Exception e) { }
  }
}
```

Figure IX.22. Extrait du code de récupération de l'ontologie de service

[65] De façon similaire, on dispose également de la méthode get*FundamentalOntology() qui récupère les ontologies fondamentales.*

IX.3.4. Module d'exécution (run-time)

IX.3.4.1. Service brokering sémantique

Dans sa version 1.0, certaines fonctionnalités du service brokering sémantique sont implémentées de façon relativement simplifiée sous forme de classes Java qui répondent aux requêtes des utilisateurs et des autres services en invoquant les modules appropriés. Pour cela, nous avons développé une interface permettant aux utilisateurs du prototype d'interagir avec ce service. La figure IX.23 qui illustre l'interface implémentée permet de montrer qu'il est possible d'invoquer à partir de ce point d'accès la majorité des requêtes : requête de découverte, requête de médiation, etc.

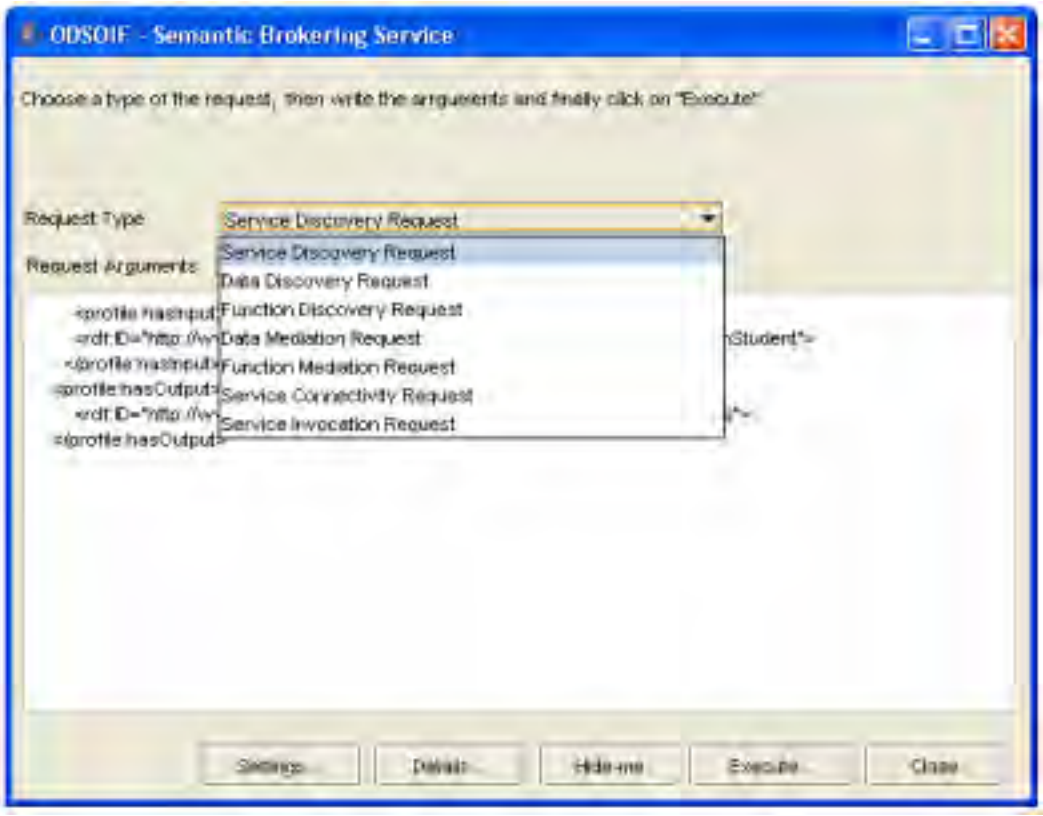

Figure IX.23. Interface du service brokering

IX.3.4.2. Service de découverte sémantique

La figure suivante illustre l'interface utilisée pour invoquer le service de découverte de services d'entreprise. Comme on peut le constater, cette interface offre dans sa version 1.0, des zones de texte permettant de saisir les éléments de la requête. De plus, on offre la possibilité pour modifier les poids (weights) qui sont utilisés pour calculer l'indice global de similarité entre la requête et les services existants (cf. § VIII.4.3.2.2).

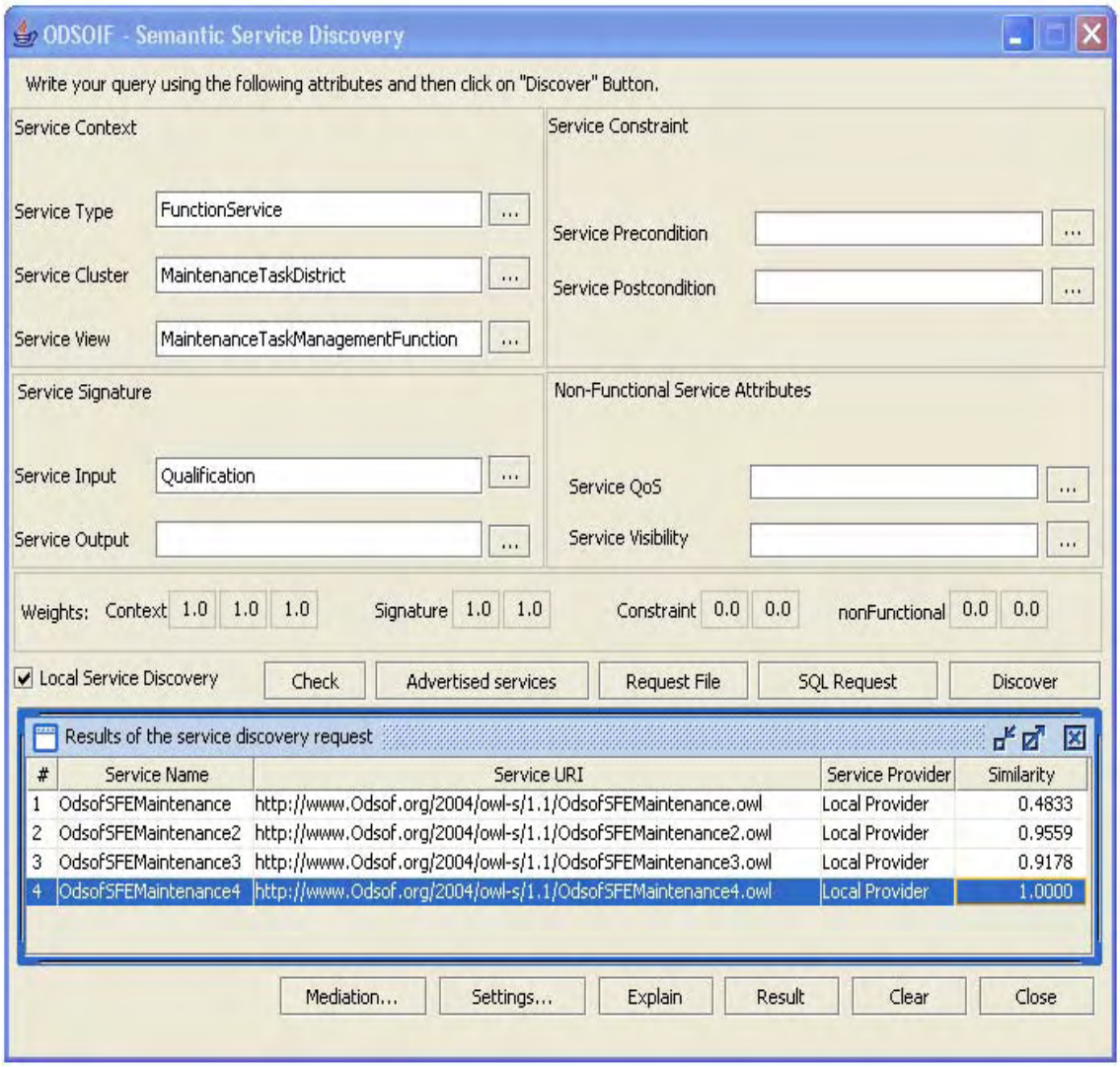

Figure IX.24. Module de découverte sémantique de services

Une fois que l'utilisateur clique sur le bouton "Discover", l'algorithme de comparaison (tel que décrit dans la section VIII.4.3.2.1) est lancé. Ce dernier exploite la comparaison de concepts pour calculer les indices élémentaires (cf. § VIII.4.3.2.3). La figure IX.25 fournit un extrait de code Java permettant de restituer les indices élémentaires et l'indice global de similarité.

```
/* méthode de calcul des indices de similarité. Le tableau 'options' transmis en argument précise les types de
matchings à effectuer : matching de types (options[0]==1), matching de cluster (options[1]==1), matching de vues
(options[2]==1),  matching d'inputs (options[3]==1),... Notez que typeThreshold, clusterThreshold, viewThreshold,…
sont des constantes qui définissent les seuils associés aux différents types de matchings. Notez aussi que typeWeight,
clusterWeight, viewWeight,…sont des poids associés aux différents types de matchings.
*/
public static double getServiceMatching(Request r, Service s, int [] options) {
   // Context matching
   double typeIndice = (options[0]==1)? matchServiceType(r, s): 1;
   if(typeIndice <= typeThreshold) return 0;
   double clusterIndice= (options[1]==1)? matchServiceCluster(r, s): 1;
   if(clusterIndice <= clusterThreshold) return 0;
   double viewIndice= (options[2]==1)? matchServiceViews(r, s): 1;
   if(viewIndice <= viewThreshold) return 0;
   double contextIndice= ((options[0]==1)|(options[1]==1)|(options[2]==1))? (Math.pow(clusterIndice, clusterWeight )*
               Math.pow(typeIndice, typeWeight) * Math.pow(viewIndice, viewWeight)): 1;
   // Signature matching
   double inputIndice= (options[3]==1)? matchServiceInputs(r, s): 1;
   if(inputIndice <= inputThreshold) return 0;
   double outputIndice= (options[4]==1)? matchServiceOutputs(r, s): 1;
   if(outputIndice <= outputThreshold) return 0;
   double signatureIndice= ((options[3]==1)|(options[4]==1))? (Math.pow(inputIndice, inputWeight )*
               Math.pow(outputIndice, outputWeight)): 1;
    ...
    // Global matching
   double globalIndice = contextIndice * signatureIndice * constraintIndice * nonFunctionalIndice;
   return globalIndice;
}
public double matchServiceType(Request r, Service s){
   ServiceType type = s.getServiceType(); OWLType typeClass  = type.getParamType();
   ServiceType requestType = s.getServiceType(); OWLType requestTypeClass  = requestType.getParamType();
   return getConceptMatching( requestTypeClass.getURI().toString(), typeClass.getURI().toString(),
               r.serviceTypeOntology());
}
...
public double matchServiceViews(Request r, Service s){
  ViewList serviceViews = s.getProfile().getViews();
  ViewList requestViews = r.getProfile().getViews();
  Iterator i = serviceViews.iterator(); double finalIndice = 0;
  while( i.hasNext() ) {
    View view = (View) i.next(); OWLType viewType = view.getParamType();
    Iterator j = requestViews.iterator();
    while( j.hasNext() ) {
      View requestView = (View)  j.next(); OWLType requestViewType = requestView.getParamType();
      double matchIndice = getConceptMatching( requestViewType.getURI().toString(), viewType.getURI().toString(),
               r.serviceViewOntology());
      finalIndice = Math.max(matchIndice, finalIndice);
    }
}
   return finalIndice;
}
public static double matchServiceInputs(Request r, Service s){
   InputList serviceInputs = s.getProfile().getInputs();InputList requestInputs = r.getProfile().getInputs();
   Iterator i = serviceInputs.iterator(); double finalIndice = 0;
   while( i.hasNext() ) {
     Input input = (Input) i.next(); OWLType inputType  = input.getParamType();
     Iterator j = requestInputs.iterator();
     while( j.hasNext() ) {
       Input requestInput = (Input)  j.next();OWLType requestInputType = requestInput.getParamType();
       double matchIndice = getConceptMatching( requestInputType.getURI().toString(), inputType.getURI().toString(),
               r.serviceDataOntology());
       finalIndice = Math.max(matchIndice, finalIndice);
     }
}
   return finalIndice;
}
...
public static double getConceptMatching(String sourceConcept, String targetConcept, String ontology) {
    double result = 0;
    int[] d = getRacerAncestorDistance(sourceConcept, targetConcept, ontology);
    result = 1 - Math.pow((Math.abs(d[0]-d[1])/(d[0]+d[1])), 1+Math.max(d[0],d[1]));
    result = result * (1 - ((Math.max(d[0],d[1])-1)/(1+depth(ontology))));
    result = result * (1-((Math.min(d[0],d[1])-1)/(1+depth(ontology))));
    result = result * Math.pow((1/(Math.min(d[0],d[1]))),Math.log10(depth(ontology)));
return result;
}
```

Figure IX.25. Extrait du code pour le calcul des indices de similarité

IX.3.4.3. Service de médiation sémantique

Dans la version 1.0, les services de médiation sont développés sous formes de services simples permettant de mettre en évidence le déroulement du processus de médiation. Les figures IX.26 et IX.27 illustrent deux extraits de code permettant d'implémenter respectivement le service de médiation de données et le service de médiation fonctionnelle.

```
/* méthodes de médiation de données
*/
public static int syntacticMaturity = 0;
public static int semanticMaturity = 1;
// Processus global de médiation
public static Message dataMediation(OutputPort sourcePort, InputPort targetPort, Message message) {
  if ( (sourcePort == null) || (targetPort == null) || (message == null))return (null);
  int serviceMaturity = sourcePort.getServiceMaturity()* targetPort.getServiceMaturity();
  if (serviceMaturity == syntacticMaturity) return (syntacticDataMediation(sourcePort, targetPort, message));
  if (serviceMaturity == semanticMaturity) return (semanticDataMediation(sourcePort, targetPort, message));
  return null;
}
// Médiation syntaxique des données
public static Message syntacticDataMediation(Port sourcePort, Port targetPort, Message message) {
  if ( (sourcePort == null) || (targetPort == null) || (message == null))return (null);
  boolean syntacticServiceConnectivity = ServiceConnectivity.getSyntacticServiceConnectivity(sourcePort,
targetPort);
  if (syntacticServiceConnectivity) {return (message);}
  else {return (TraditionalDataMediation.mediate(sourcePort, targetPort, message));}
}
// Médiation sémantique des données
public static Message semanticDataMediation(Port sourcePort, Port targetPort, Message message) {
  if ( (sourcePort == null) || (targetPort == null) || (message == null))return (null);
  boolean semanticServiceConnectivity = ServiceConnectivity.getSemanticServiceConnectivity(sourcePort,
targetPort);
  if (semanticServiceConnectivity)
  {
   boolean syntacticServiceConnectivity = ServiceConnectivity.getSyntacticServiceConnectivity(sourcePort,
targetPort);
   if (syntacticServiceConnectivity) {return (directDataMediation(sourcePort,targetPort, message));}
   else {return (ontologyDrivenDataMediation(sourcePort, targetPort, message));}
   }
  else return (null);
}
// Calcul du message à renvoyer
public static Message ontologyDrivenDataMediation(Port sourcePort, Port targetPort, Message message) {
  if ( (sourcePort == null) || (targetPort == null) || (message == null)) {return (null);}
  return OntologyDrivenDataMediation.mediate(sourcePort, targetPort, message);
}
// Passage du message
public static Message directDataMediation(Port sourcePort, Port targetPort, Message message) {
  if ( (sourcePort == null) || (targetPort == null) || (message == null)) {return (null);}
  return DirectDataMediation.mediate(sourcePort, targetPort, message);
}
// Calcul du message à renvoyer en utilisant la médiation sémantique
public static Message mediate(Port sourcePort, Port targetPort, Message message) {
  if ( (sourcePort == null) || (targetPort == null) || (message == null))
  {return (null);}
  return syntacticConcretisation( sourcePort, targetPort,
            semanticTransformation( sourcePort, targetPort,
                  semanticAbstraction(sourcePort,targetPort, message)));
}
// retourne la concrétisation syntaxique d'un port
public static Message syntacticConcretisation (Port sourcePort, Port targetPort, Message message) {
  return SyntacticConcretisation.syntacticConcretisation (sourcePort, targetPort, message);
}
// la transformation sémantique d'un port
public static Message semanticTransformation (Port sourcePort, Port targetPort, Message message) {
  return SemanticTransformation.semanticTransformation (sourcePort, targetPort, message);
}
// l'abstraction sémantique d'un port
public static Message semanticAbstraction (Port sourcePort, Port targetPort, Message message) {
  return SemanticAbstraction.semanticAbstraction (sourcePort, targetPort, message);
}
```

Figure IX.26. Extrait de code du module de médiation de données

```
/* méthodes de médiation fonctionnelle
*/

public static int _SyntacticMaturity = 0;
public static int _SemanticMaturity = 1;
public static int _ExactMatch   = 0;
public static int _SubsumeMatch = 1;
// Processus global de médiation
public static Message functionMediation(Function sourceFunction, Function targetFunction, Message message) {
  if ( (sourceFunction == null) || (targetFunction == null) || (message == null))return (null);
  int serviceMaturity = sourceFunction.getServiceMaturity()* targetFunction.getServiceMaturity();
  if (serviceMaturity == _SyntacticMaturity) return (syntacticFunctionMediation(sourceFunction, targetFunction, message));
  if (serviceMaturity == _SemanticMaturity) return (semanticFunctionMediation(sourceFunction, targetFunction, message));
  return null;
}
// Médiation syntaxique
public static Message syntacticFunctionMediation(Function sourceFunction, Function targetFunction, Message message) {
  if ( (sourceFunction == null) || (targetFunction == null) || (message == null))return (null);
  boolean syntacticConnectivity = getSyntacticConnectivity(sourceFunction, targetFunction);
  if (syntacticConnectivity) return dataMediation(sourceFunction.selectedPort(), targetFunction.selectedPort(), message);
  else
  {
  targetFunction = getCompatibleFunction(targetFunction, message);
  return functionMediation(sourceFunction, targetFunction, message);
  }
}
// Médiation sémantique
public static Message semanticFunctionMediation(Function sourceFunction, Function targetFunction, Message message) {
  if ((sourceFunction == null) || (targetFunction == null) || (message == null))return (null);
  boolean semanticConnectivity = getSemanticConnectivity(sourceFunction, targetFunction);
  if (semanticConnectivity)
  {
  boolean syntacticConnectivity = getSyntacticConnectivity(sourceFunction, targetFunction);
  if (syntacticConnectivity) return (directDataMediation(sourceFunction.selectedPort(), targetFunction.selectedPort(), message));
  else return (ontologyDrivenDataMediation(sourceFunction, targetFunction, message));
  }
  else
  targetFunction = getCompatibleFunction(targetFunction, message);
  return functionMediation(sourceFunction, targetFunction, message);
}
// Calcul de la connectivité sémantique
public static boolean getSemanticConnectivity(Function sourceFunction, Function targetFunction, Message message) {
  List sourceOutputs = sourceFunction.getOutputs(message) ;
  List targetInputs = targetFunction.getInputs(message);
  Iterator i = sourceOutputs.iterator();
  while( i.hasNext() ) {
    Mapping mapping = (Mapping) i.next();
    Port output = (Port) ((OWLIndividual) mapping.getValue("port")).castTo(Output.class);
    OWLType outputConcept = output.getParamType();
    Iterator j = targetInputs.iterator();
      while( j.hasNext()) {
        mapping = (Mapping) j.next() ;
        Input input = (Input) ((OWLIndividual) mapping.getValue("port")).castTo(Input.class);
        OWLType inputConcept = input.getParamType();
        int matchResult = getPortMatching(outputConcept, inputConcept);
        if((matchResult != _ExactMatch)||((matchResult != _SubsumeMatch)))
        return true;
      }
   }
   return false;
}
```

Figure IX.27. Extrait du code du module de médiation fonctionnelle

IX.4. Expérimentation

Cette section est consacrée à l'expérimentation sur un cas réel du prototype que nous avons décrit précédemment. Le cas porte sur l'intégration sémantique de services dans le domaine de la maintenance des équipements de production de la société

STMicroelectronics (cf. § I.2). Dans ce qui suit, nous allons tout d'abord présenter notre cas d'étude. Ensuite nous allons, à travers une expérimentation du prototype développé, mettre en pratique les principaux éléments issus de notre cadre méthodologique et qui rappelons-le sont :

- la construction de l'architecture de services d'entreprise (ASE),
- la construction de l'architecture d'ontologies d'entreprise (AOE),
- et l'utilisation de l'architecture d'intégration d'entreprise (AIE).

Bien entendu, la mise en pratique de tous ces éléments nécessite un travail important. Aussi nous nous sommes très souvent contentés de présenter les choses de façon simples voire parfois de façon simplifiée afin de rendre aisée la lecture et montrer surtout la pertinence de notre approche.

IX.4.1. Etude de cas - STMicroelectronics

Rappelons que cette expérimentation s'inscrit dans le prolongement de la décision de migration de STMicroelectronics vers les architectures orientées services. Notre étude de cas porte sur un sous-système de gestion de la maintenance préventive des équipements de fabrication des puces microélectroniques. Le schéma suivant permet d'illustrer l'architecture applicative existante et qui comprend les applications suivantes (figure IX.28) :

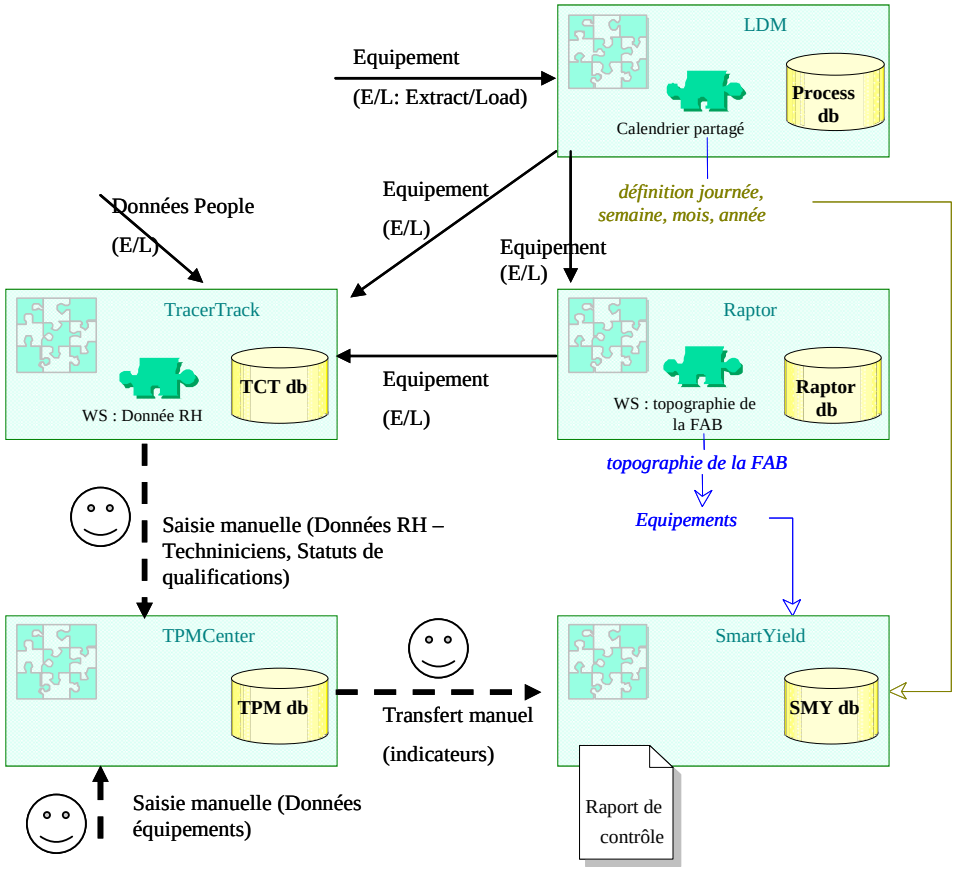

Figure IX.28. Architecture applicative du système ciblé

- TPMCenter: application qui gère la maintenance préventive des équipements,
- TracerTrac: application qui gère la certification des opérateurs sur les équipements et les

postes de travail,

- LDM: application qui centralise les données sur les équipements,
- et SmartYield application qui gère les rendements.

IX.4.1.1. TPMCenter

L'application TPMCenter (Total Preventive Maintenance management Center) permet la gestion des tâches de planification de la maintenance des équipements de fabrication. Cette application produit les plannings de maintenance préventive en utilisant les données portant sur les statuts de qualifications (issues de TracerTrack), l'état des équipements (application CAM), les compteurs équipements (nombre plaques wafers fabriqués, …), informations de planifications (STAP) et les informations ADCS[66] (Gestion de la documentation) qui permet de renseigner sur les opérations à faire durant les tâches de maintenance.

Parmi les principaux services rendus par cette application, nous pouvons citer :

- la gestion des listes (Checklists Management);
- la planification basée compteurs (Counter Based Planning);
- la gestion de la maintenance curative (Curative Maintenance Management);
- l'extraction de compteurs équipements (Equipments Counter Files Extraction);
- le chargement des compteurs équipements (Equipments Counter Files Loading);
- la planification basée sur la fréquence (Frequency Based Planning);
- la gestion de la maintenance non-standard (Non Standard Maintenance Management);
- la gestion de la maintenance préventive (Preventive Maintenance Management);
- et la génération de fichiers STAP[67] (STAP File Generation).

Il est important de préciser que cette application doit être intégrée avec d'autres applications existantes, notamment avec les applications TracerTrack et LDM. A ce jour, une intégration manuelle est mise en place afin de pouvoir intégrer cette application en effectuant la saisie manuelle des informations produites par d'autres applications. A terme, les utilisateurs souhaitent avoir accès à partir de l'application TPMCenter à certains services de TracerTrack leur fournissant ainsi l'information de façon automatique sur la qualification des techniciens.

IX.4.1.2. TracerTrack

L'application TracerTrack permet de gérer la certification des opérateurs et des techniciens sur les équipements. En d'autres termes, il s'agit du système qui permet de répondre à la question qui porte sur la qualification d'une personne donnée. Le système nous permet donc de connaître si un opérateur ou technicien donné est qualifié (a reçu la formation nécessaire) pour utiliser un équipement ou un procédé précis, et si sa qualification est en cours de validité.

Cette application gère principalement les qualifications et les statuts d'activité d'une personne. En revanche, elle nécessite en entrée des données sur les personnes issues du système de gestion des ressources humaines (PeopleFirst) et du système de gestion des

[66] ADCS est une application de gestion des documents techniques des équipements et qui sont particulièrement utilisés dans le cadre de TPMCenter dans le but de connaître les gammes opératoires en ce qui concerne les tâches de maintenance.
[67] STAP est une application qui permet de gérer les plans de production à court terme.

congés (Arhistote). Elle nécessite aussi des données sur les équipements issues du système LDM. De plus, cette application peut gérer des notifications pour les applications clientes concernant les changements qui peuvent s'opérer sur le statut des personnes.

Les principaux services offerts par cette application sont :

- la qualification d'une personne sur tel équipement (manufacturing equipment qualification);
- et la qualification d'une personne sur tel poste de travail (manufacturing workstation qualification).

On doit préciser que TracerTrack est une application importante dans le cadre de la gestion de la maintenance. Il doit jouer le rôle d'un serveur naturel pour les clients TPMCenter et SmartYield qui sont demandeurs d'informations sur les statuts de qualification des techniciens. Dans l'état actuel des choses, les réponses à ces requêtes sont rendues de façon manuelle, d'où la nécessité de l'intégration orientée services pour pouvoir réutiliser ces fonctionnalités de manière distante et de façon flexible.

IX.4.1.3. Raptor

L'application Raptor (Reliability, Availability and Productivity Tracking for Operational Reporting) permet de gérer le reporting sur les équipements. Elle constitue le référentiel des données équipements consolidées (données issues du système CAM et de l'automation). Elle utilise les données fournies par EDA[68] et restitue des rapports sur les équipements.

Les principaux services offerts par cette application sont :

- le chargement des données (data loading);
- l'agrégation des données (data aggregation);
- le calcul de certains indicateurs, notamment ceux portent sur l'efficacité des équipements de production;
- et le rafraîchissement des rapports (automatic reports refreshing).

Nous devons préciser que, comme Raptor constitue le propriétaire naturel des données consolidées sur les équipements, on peut donc le charger de mettre à disposition ces données pour les applications clientes (notamment TPMCenter). Actuellement, ces données sont disponibles mais elles sont saisies manuellement dans TPMCenter.

IX.4.1.4. LDM

L'application LDM (Local Data Management) permet de centraliser les données issues de Workstream (MES[69]) et du système APF (gestion des rebus ou des scraps) pour les mettre à disposition d'autres applications, dont notamment TracerTrack et Raptor. Ces données en entrée portent principalement sur le modèle du lot, le modèle équipement et les activités humaines. Ce système intègre d'autres données importantes pour la maintenance dont le calendrier partagé qui est issu du système MES. Toutes ces données sont rendues disponibles aux autres applications. De ce fait, LDM constitue la référence en ce qui concerne les données répliquées au niveau de TracerTrack et de Raptor.

Les principaux services offerts par cette application sont :

[68] EDA (Engineering Data Analysis) est un système qui analyse les multiples données issues des équipements en vue de chercher les causes expliquant les symptômes qui peuvent être constatés sur les données analysées.
[69] Manufacturing Execution System.

- le chargement de modèles (model loader);
- le chargement des données équipement (WIP equipment loader);
- le chargement des données lots (WIP lot loader);
- et la restitution des données (data extraction).

IX.4.1.5. SmartYield

L'application SmartYield permet la gestion du rendement des équipements de fabrication. Elle permet notamment d'effectuer l'analyse des scraps (rebus). Elle est alimentée par des données fournies par l'application APF (données scraps), par l'application EDA (données portant sur le modèle CAM) et aussi par le référentiel LDAP (données sur les utilisateurs). Une fois les données récoltées, ces dernières sont centralisées dans une source unique d'information pour les rendre accessibles par tous les utilisateurs concernés. De plus, SmartYield fournit des rapports de contrôle (contenant divers indicateurs) à destination des gestionnaires.

Les principaux services offerts par cette application sont :

- le chargement des données (load data);
- la gestion du plan d'action à l'analyse des scraps;
- la répartition des scraps en function des opérateurs et des équipements;
- et l'élaboration de rapport de contrôle (control report).

Il faut préciser que cette application doit être liée aux autres services issus des autres applications pour une meilleure gestion des scraps. Il s'agit notamment de lier ces services à ceux fournis par l'application TPMCenter. Ceci permettrait de fournir à SmartYield un récapitulatif de l'historique des maintenances préventives réalisées sur un équipement donné. Ce qui permettrait à mieux gérer les scraps.

IX.4.2. Construction de l'architecture de services d'entreprise

Comme nous l'avons déjà évoqué dans le chapitre VI, la construction de l'architecture de services d'entreprise s'inscrit dans le cadre de la modernisation du système d'information d'entreprise et constitue une tâche complexe du fait qu'elle nécessite des efforts importants et la maîtrise d'un certain nombre de technologies nouvelles et complexes. Pour cela, dans le cadre de notre expérimentation, et dans le but de simplifier le processus de ré-ingénierie nécessaire pour la 'servicisation', nous sommes parfois conduits à construire des "bouchons" qui simulent le comportement des modules applicatifs, et d'utiliser ces bouchons en lieu et place des véritables modules applicatifs qui eux ne sont pas encore totalement prêts à être déployés en tant que services applicatifs.

Notons que dans le cadre de la construction de la couche de services, nous avons principalement utilisé la plate-forme BEA-Weblogic qui permet de définir facilement des wrappers à partir de modules Java, d'un certain nombre de bases de données et des applications de certains éditeurs. La figure IX.29 illustre un exemple de génération d'un service et de sa description syntaxique (WSDL) en utilisant l'outil BEA-Workshop.

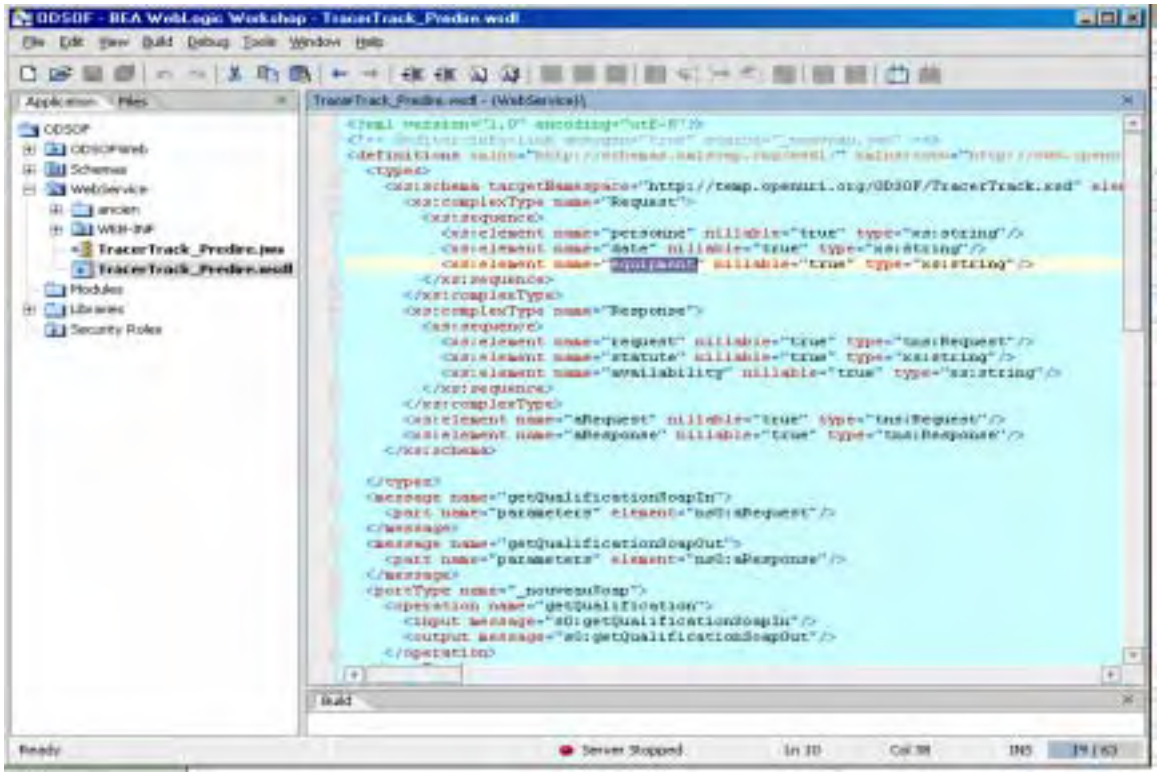

Figure IX.29. Environnement technique utilisé pour la construction des services

Dans ce qui suit nous allons dérouler, dans le cadre de notre étude de cas, les différentes étapes pour la construction de l'architecture de services (cf. § VI.4).

IX.4.2.1. Exposition des composants en services informatiques

La construction de la SOA-IT (cf. § VI.4.2) dans le cadre de notre cas d'étude a permis de définir les services applicatifs suivants :

Application	Composant applicatif	Service applicatif
TPMCenter	gestion des listes	SAE_TPMCenter_Gérer_Liste
	planification basée compteurs	SAE_TPMCenter_Planifier_Compteurs
	gestion de la maintenance curative	SAE_TPMCenter_Gérer_Maintenance_Curative
	extraction de compteurs équipements	SAE_TPMCenter_Extraire_Compteurs
	chargement des compteurs équipement	SAE_TPMCenter_Charger_Compteurs
	planification basée fréquence	SAE_TPMCenter_Planifier_Frequence
	gestion de la maintenance non-standard	SAE_TPMCenter_Gérer_Maintenance_Non_Std
	gestion de la maintenance préventive	SAE_TPMCenter_Gérer_Maintenance_Preventive
	génération de fichiers STAP	SAE_TPMCenter_Génerer_STAP

TracerTrack	Qualification sur équipement d'une personne	SAE_TracerTrack_Restituer_Qualification_Equipement
	Qualification sur poste d'une personne	SAE_TracerTrack_Restituer_Qualification_Poste
Raptor	chargement des données	SAE_Raptor_Charger
	agrégation des données	SAE_Raptor_Agréger
	rafraîchissement des rapports	SAE_Raptor_Rafraichir
LDM	chargement de modèles	SAE_LDM_Charger_Modèles
	chargement des données équipement	SAE_LDM_Charger_Equipement
	chargement des données lots	SAE_LDM_Charger_Lot
	extraction de données	SAE_LDM_Extraire
SmartYield	chargement des données	SAE_SmartYield_Charger
	gérer plan d'action	SAE_SmartYield_Planifier
	répartir les scraps	SAE_SmartYield_Répartir_Scraps
	élaboration de rapport de contrôle	SAE_SmartYield_Rapport

Tableau IX.1. Définition des services applicatifs

Rappelons que ces services applicatifs découlent directement des modules des applications existantes. Une fois définis, ces services applicatifs sont saisis dans notre référentiel ODSOIF. L'étape suivante consiste à définir les services métier.

IX.4.2.2. Définition des services métier

Sur la base des services précédents, nous avons défini un certain nombre de services fonctionnels (cf. § VI.4.3) qui constituent la SOA-métier de notre architecture. Nous avons reproduit ci-dessous les services fonctionnels issus des deux applications TpmCenter et TracerTrack.

IX.4.2.2.1. Services fonctionnels issus de TpmCenter

Sur la base des services applicatifs issus de l'application TpmCenter, nous avons pu identifier les quatre services fonctionnels suivants : *SFE_Maintenance_Planifier, SFE_Maintenance_Suivre_Réalisation, SFE_Maintenance_Rapporter* et *SFE_Maintenance_Avertir*. Seuls ces services possèdent pour l'instant un sens métier et un intérêt particulier dans le cadre de l'intégration de services de STMicroelectronics qui nous pousse ainsi à les définir et à les publier en tant que services métier.

IX.4.2.2.1.1 SFE_Maintenance_Planifier

Ce service permet de définir des plannings prévisionnels en matière de tâches de maintenance à effectuer sur les équipements. Pour produire les plannings prévisionnels de maintenance, ce service accepte en entrée les informations suivantes

- informations sur équipement
- tâches de maintenance à réaliser
- Série (enchaînement de tâches de maintenance)

IX.4.2.2.1.2 SFE_Maintenance_Suivre_Réalisation

Ce service permet à un utilisateur de suivre la réalisation des tâches de maintenance. On peut distinguer deux opérations de base qui sont la visualisation des cheklists et leur remplissage. Pour suivre la réalisation des tâches de maintenance, ce service exploite en entrée un planning prévisionnel.

IX.4.2.2.1.3 SFE_Maintenance_Rapporter

Ce service permet d'effectuer des reportings à destination d'autres services (tels que Raptor). On distingue deux types de reportings

- reportings portant sur les tâches de maintenance réalisées par un opérateur (tâches réalisées avec qualification, tâches réalisées sans qualification)
- reportings portant sur les tâches de maintenance réalisées sur un équipement (tâches réalisées à temps, tâches reportées)

IX.4.2.2.1.4 SFE_Maintenance_Avertir

Ce service permet d'envoyer des alarmes en ce qui concerne les tâches de maintenance non effectuées. Les utilisateurs alertés se chargeront alors de gérer les tâches de maintenance à leur niveau.

En résumé, il est possible de récapituler l'ensemble des services fonctionnels issus de TpmCenter par le tableau suivant qui résume ainsi leurs caractéristiques principales.

Nom du service	Description	Input	Output
SFE_Maintenance_Planifier	Définir des plannings de maintenance préventive	Equipement Tâche de maintenance Série	Planning de maintenance
SFE_Maintenance_Suivre_Réalisation	Suivre la réalisation des tâches de maintenance	Cheklist	Cheklist remplie
SFE_Maintenance_Rapporter	Etablir des rapports sur les tâches de maintenance	Durée Population Equipement Poste	Tâche réalisée (par population, par équipement, par poste)
SFE_Maintenance_Avertir	Notifier les tâches de maintenance non encore réalisées	Evénement (Date dépassée)	Tâches non réalisées

Tableau IX.2. Services fonctionnels issus de TpmCenter

IX.4.2.2.2. Services fonctionnels issus de TracerTrack

Comme pour TpmCenter, nous avons pu identifier sur la base des services applicatifs issus de TracerTrack les quatre services fonctionnels suivants : *SFE_Qualification_Planifier, SFE_Qualification_Suivre, SFE_Qualification_Predire et*

SFE_Qualification_Rapporter. Comme pour précédemment, seuls ces services possèdent pour l'instant un sens métier et un intérêt particulier dans le cadre de l'intégration de services de STMicroelectronics qui nous pousse ainsi à les définir et à les publier en tant que services métier.

IX.4.2.2.2.1 SFE_Qualification_Planifier

Ce service permet de planifier et d'organiser des formations. Il permet aussi de valider la qualification d'une personne à l'issue d'une formation. Les informations utilisées en entrée pour pouvoir alimenter ensuite la base de données sont: formation (Training), Date, Population visée (opérateurs), formateur (Trainer).

IX.4.2.2.2.2 SFE_Qualification_Suivre

Ce service permet de suivre au jour le jour les états de qualification des opérateurs et des techniciens. Il est important de préciser que certaines formations ne sont pas qualifiantes alors que d'autres le sont, et ce service ne s'intéresse qu'à celles qui sont qualifiantes. De plus, il faut signaler que les qualifications possèdent une durée de validité au delà de laquelle la personne concernée est considérée comme disqualifiée. Le principe de base de ce service repose sur le fait qu'il surveille quotidiennement les événements qualifiants (et disqualifiants) en relevant les qualifications expirées. Il avertit alors les services et/ou les utilisateurs concernés par le biais d'alarmes de disqualification (en principe de façon asynchrone). L'entrée de ce service est donc fondamentalement la date du jour (date du système), tandis que la sortie est l'alarme de disqualification.

IX.4.2.2.2.3 SFE_Qualification_Predire

Ce service permet de prédire la présence et la qualification d'une population (opérateur) à une date donnée concernant un poste ou un équipement précis. Il s'agit d'un service synchrone qui répond aux sollicitations des utilisateurs qui gèrent notamment les tâches de maintenance et qui veulent connaître le statut de qualification de leurs opérateurs. Ce service répond à des requêtes comportant la date, la personne (ou la population: groupe), l'équipement ou le poste et répond par le statut de qualification ainsi que par la date d'expiration de la qualification. De plus, des informations sur la disponibilité de la population y sont généralement ajoutées.

IX.4.2.2.2.4 SFE_Qualification_Rapporter

Ce service porte essentiellement sur deux types de rapports :

- Rapport sur le nombre d'heures de formation concernant une population donnée (personne, atelier, formateur) sur une certaine durée de temps. On parle du nombre d'heures d'OCR (Operation Certification Rate) d'une population donnée.
- Rapport sur le nombre ou le niveau de qualifications d'une population (personne, équipe).

De même que précédemment, il est possible de récapituler l'ensemble des services fonctionnels issus de TracerTrack par le tableau suivant, qui résume leurs caractéristiques principales.

Nom service	Description	Input	Output
SFE_Qualification_Planifier	Organiser les trainings (qualifiants ou pas) puis valider les qualifications.	Formation Date Population (opérateur, groupe) Formateur	Mise à jour de la base de données Track
SFE_Qualification_Suivre	Surveiller au jour le jour les événements liés à la qualification.	Date	Alarmes (fin de qualification)
SFE_Qualification_Predire	Prévoir la présence et la qualification des opérateurs à une date donnée.	Date Population (opérateur, groupe) Equipement/ poste	Statut de qualification, Disponibilité Date d'expiration
SFE_Qualification_Rapporter	Etablir des états récapitulatifs par type de formation, par formateur, etc.	Durée Type formation, Formateur	Formations filtrées

Tableau IX.3. Services fonctionnels issus de TracerTrack

IX.4.2.2.3. Référencement des services fonctionnels définis

Une fois les services fonctionnels définis, ces derniers sont référencés au sein du référentiel ODSOIF. Le référencement d'un service fonctionnel permet de décrire les différents attributs qui caractérisent le service fonctionnel. Il permet notamment de renseigner les champs sur la localisation de la description syntaxique du service (fichier WSDL). A l'issue de cette phase, on dispose d'un référentiel ODSOIF contenant la description de notre SOA métier. La figure qui suit illustre un extrait de ce référentiel.

IX.4.2.3. Urbanisation des services métiers

Une fois que le référentiel de la SOA-métier est soigneusement rempli, nous exploitons ses informations afin d'effectuer une urbanisation orientée services (cf. § VI.5). Nous utilisons alors notre module de calcul des degrés de similarité qui nous délivre des indices de similarité sur l'échantillon des services présenté en IX.4.2.2 (figure IX.31).

```
<?xml version="1.0"?>
<rdf:RDF
    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
    xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
    xmlns:owl="http://www.w3.org/2002/07/owl#"
    xmlns="http://www.owl-ontologies.com/odsof-normal.owl#"
  xml:base="http://www.owl-ontologies.com/odsof-normal.owl">
  <owl:Ontology rdf:about=""/>
        <enterprise_service_view>
          <Enterprise_Function_Service rdf:ID="SFE_Qualification_Rapporter">
            <project rdf:resource="#STMicro_Integration_Project"/>
            <publication rdf:datatype="http://www.w3.org/2001/XMLSchema#boolean"
            >false</publication>
            <syntactic-description-file-uri rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
            >C:/wsdl/qualif_rapporter.wsdl</syntactic-description-file-uri>
            <service_visibility rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
            >Public</service_visibility>
          </Enterprise_Function_Service>
        </enterprise_service_view>
        <enterprise_service_view>
          <Enterprise_Function_Service rdf:ID="SFE_Maintenance_Declencher_Alarme">
            <syntactic-description-file-uri rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
            >C:/wsdl/maint_Declencher.wsdl</syntactic-description-file-uri>
            <project rdf:resource="#STMicro_Integration_Project"/>
            <publication rdf:datatype="http://www.w3.org/2001/XMLSchema#boolean"
            >false</publication>
            <service_visibility rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
            >Private (Business)</service_visibility>
          </Enterprise_Function_Service>
        </enterprise_service_view>
        <enterprise_service_view>
          <Enterprise_Function_Service rdf:ID="SFE_Qualification_Predire">
            <service_visibility rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
            >Public</service_visibility>
            <project rdf:resource="#STMicro_Integration_Project"/>
            <syntactic-description-file-uri rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
            >C:/wsdl/qualif_Predire.wsdl</syntactic-description-file-uri>
            <publication rdf:datatype="http://www.w3.org/2001/XMLSchema#boolean"
            >false</publication>
          </Enterprise_Function_Service>
        </enterprise_service_view>
        <enterprise_service_view>
          <Enterprise_Function_Service rdf:ID="SFE_Qualification_Suivre">
            <publication rdf:datatype="http://www.w3.org/2001/XMLSchema#boolean"
            >false</publication>
            <syntactic-description-file-uri rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
            >C:/wsdl/qualif_suivre.wsdl</syntactic-description-file-uri>
            <service_visibility rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
            >Private (Business)</service_visibility>
            <description rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
            ></description>
            <project rdf:resource="#STMicro_Integration_Project"/>
          </Enterprise_Function_Service>
        </enterprise_service_view>
</rdf:RDF>
<!-- Created with Protege (with OWL Plugin 2.1, Build 284)  http://protege.stanford.edu -->
```

Figure IX.30. Extrait du référentiel correspondant à la SOA-métier

Codification des services			
Code du service	**Nom du service**	**Code du service**	**Nom du service**
S1	SFE_Maintenance_Planifier	**S5**	SFE_Qualification_Planifier
S2	SFE_Maintenance_Suivre_Réalisation	**S6**	SFE_Qualification_Suivre
S3	SFE_Maintenance_Rapporter	**S7**	SFE_Qualification_Predire
S4	SFE_Maintenance_Avertir	**S8**	SFE_Qualification_Rapporter

Matrice de similarité des services								
	S1	**S2**	**S3**	**S4**	**S5**	**S6**	**S7**	**S8**
S1	1,0000	0,5000	0,5250	0,4575	0,1000	0,0000	0,0275	0,0000
S2	0,5000	1,0000	0,4750	0,4500	0,0000	0,1000	0,0000	0,0000
S3	0,5250	0,4750	1,0000	0,5500	0,0500	0,0000	0,0750	0,1125
S4	0,4575	0,4500	0,5500	1,0000	0,0750	0,2500	0,1500	0,0250
S5	0,1000	0,0000	0,0500	0,0750	1,0000	0,2500	0,3750	0,2250
S6	0,0000	0,1000	0,0000	0,2500	0,2500	1,0000	0,1750	0,1250
S7	0,0275	0,0000	0,0750	0,1500	0,3750	0,1750	1,0000	0,1000
S8	0,0000	0,0000	0,1125	0,0250	0,2250	0,1250	0,1000	1,0000

Figure IX.31. Matrice des degrés de similarité de services

Nous injectons ensuite ces données dans l'outil XLSTAT qui effectue la clustérisation de services en appliquant l'algorithme de classification hiérarchique ascendant. Le résultat de la clustérisation sur l'échantillon précédent est consigné dans la figure IX.32.

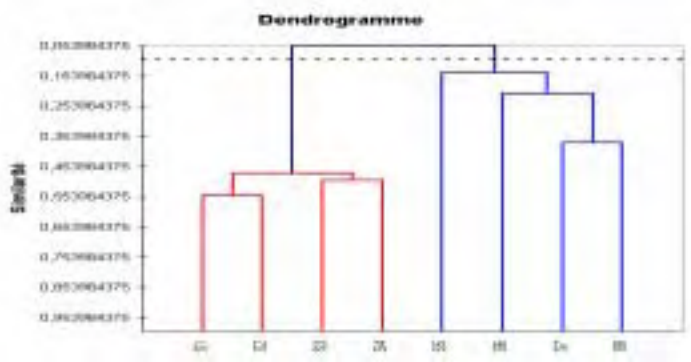

Figure IX.32. Dendrogramme obtenu

Ce résultat est ensuite soumis à l'interprétation des urbanistes et des responsables métiers pour déterminer les différents clusters de services (quartiers et zones de services). Dans notre cas, nous avons retenu la configuration de clusters suivante (figure IX.33) qui comporte une zone de services appelée *MaintenanceArea* et deux quartiers de services appelés *MaintenanceTaskDistrict* et *MaintenanceOperatorDistrict*.

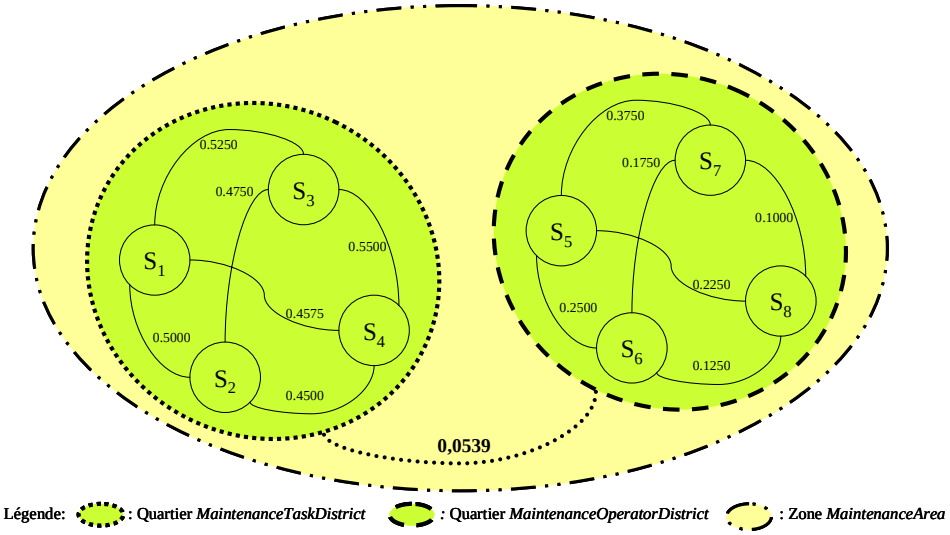

Figure IX.33. Cartographie des services d'entreprise

IX.4.2.4. Publication des services fondamentaux

Dans le cadre de notre étude, la publication syntaxique de nos services (cf. § VI.4.6) se base sur l'utilisation d'un seul référentiel UDDI. Ce dernier devra évoluer à terme pour devenir le référentiel du site Rousset[70] de l'entreprise STMicroelectronics. D'autres référentiels sont à prévoir lors de la généralisation du projet MiMISI. Nous reproduisons dans la figure qui suit un extrait du référentiel UDDI utilisé pour publier les services présentés précédemment en utilisant l'API *find_service*.

```
<serviceList generic="2.0"
  operator="STMicroelectronics"
  [truncated="false"]
  xmlns="urn:uddi-org:api_v2">
<serviceInfos>
<serviceInfo
  serviceKey="3D450068-EB79-5C09-AA00-99AF00000001"
  businessKey="01A001F5-45FB-2AF9-B78F-FE6700000001">
  <name>SFE_Maintenance_Planifier</name>
</serviceInfo>
<serviceInfo
  serviceKey="3D450068-EB79-5C09-AA00-99AF00000002"
  businessKey="01A001F5-45FB-2AF9-B78F-FE6700000001">
  <name>SFE_Maintenance_Suivre_Realisation</name>
</serviceInfo>
<serviceInfo
  serviceKey="3D450068-EB79-5C09-AA00-99AF00000003"
  businessKey="01A001F5-45FB-2AF9-B78F-FE6700000001">
  <name>SFE_Maintenance_Rapporter</name>
</serviceInfo>
<serviceInfo
  serviceKey="3D450068-EB79-5C09-AA00-99AF00000004"
  businessKey="01A001F5-45FB-2AF9-B78F-FE6700000001">
  <name>SFE_Maintenance_Avertir</name>
</serviceInfo>
</serviceInfos>
</serviceList>
```

Figure IX.34. Extrait du référentiel UDDI (API find_service)

[70] Rappelons que Rousset est l'un des sites français de la société STMicroelectronics. Il est basé dans la commune de même nom se trouvant dans le pays d'Aix en région PACA.

IX.4.3. Construction de la couche d'ontologies

IX.4.3.1. Construction des ontologies fondamentales

En se basant sur l'urbanisation des services précédente, nous avons procédé à la construction des ontologies fondamentales (cf. § VII.6.3). Pour cela nous avons construit deux ontologies locales et une ontologie de domaine (cf. § VII.5). Dans notre cas, l'ontologie globale n'a pas été construite vu que le cas d'étude est relativement restreint et ne comprend qu'une seule zone et deux quartiers.

Les figures IX.35, IX.36, IX.36 présentent des extraits des ontologies de données. Les deux premières figures concernent les ontologies de données locales tandis que la dernière figure concerne l'ontologie de données de domaine.

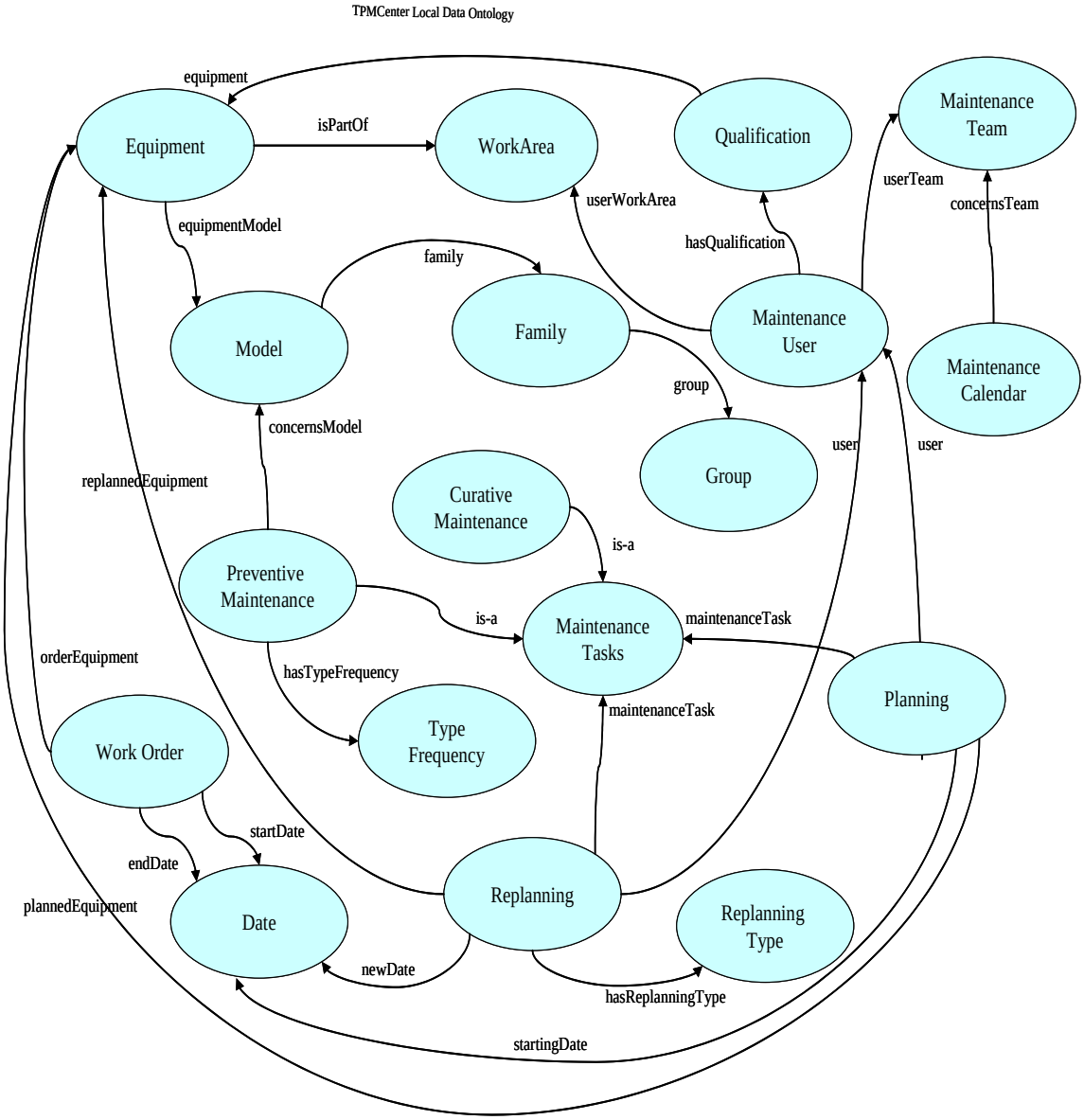

Figure IX.35. Extrait de l'ontologie de données locale (MaintenanceTaskDistrict)

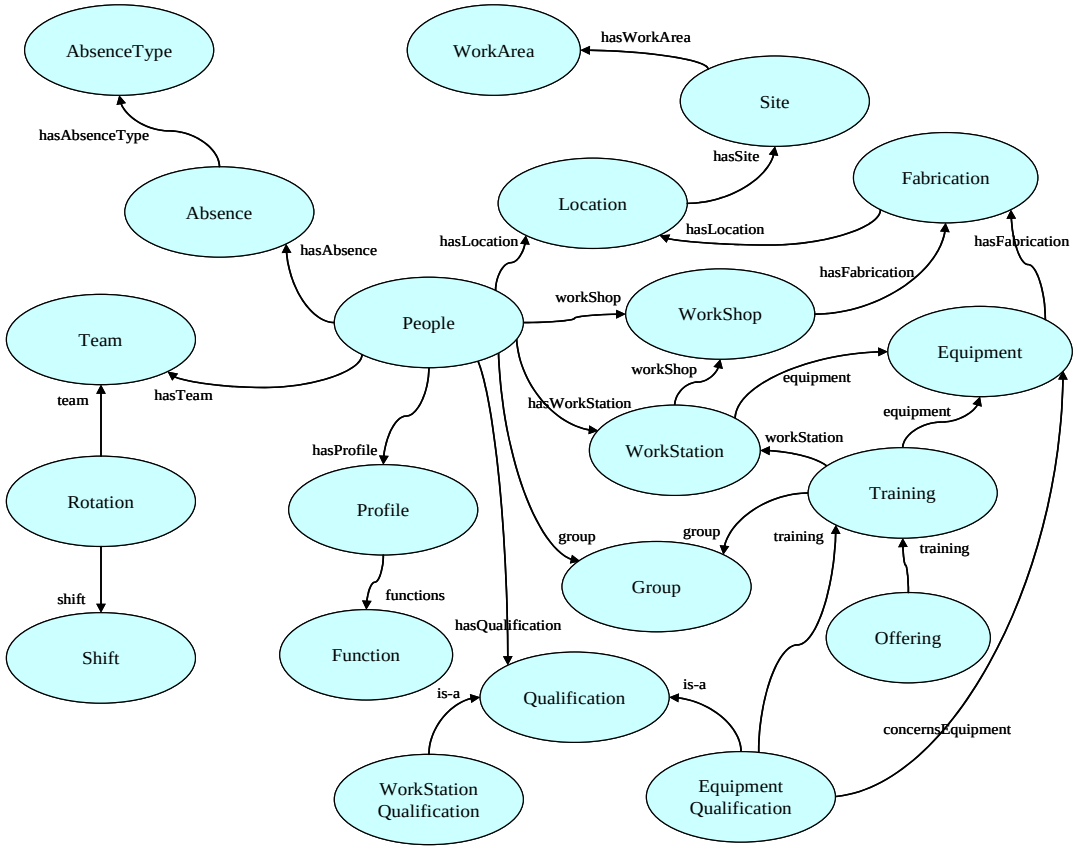

Figure IX.36. Extrait de l'ontologie de données locale (MaintenanceOperatorDistrict)

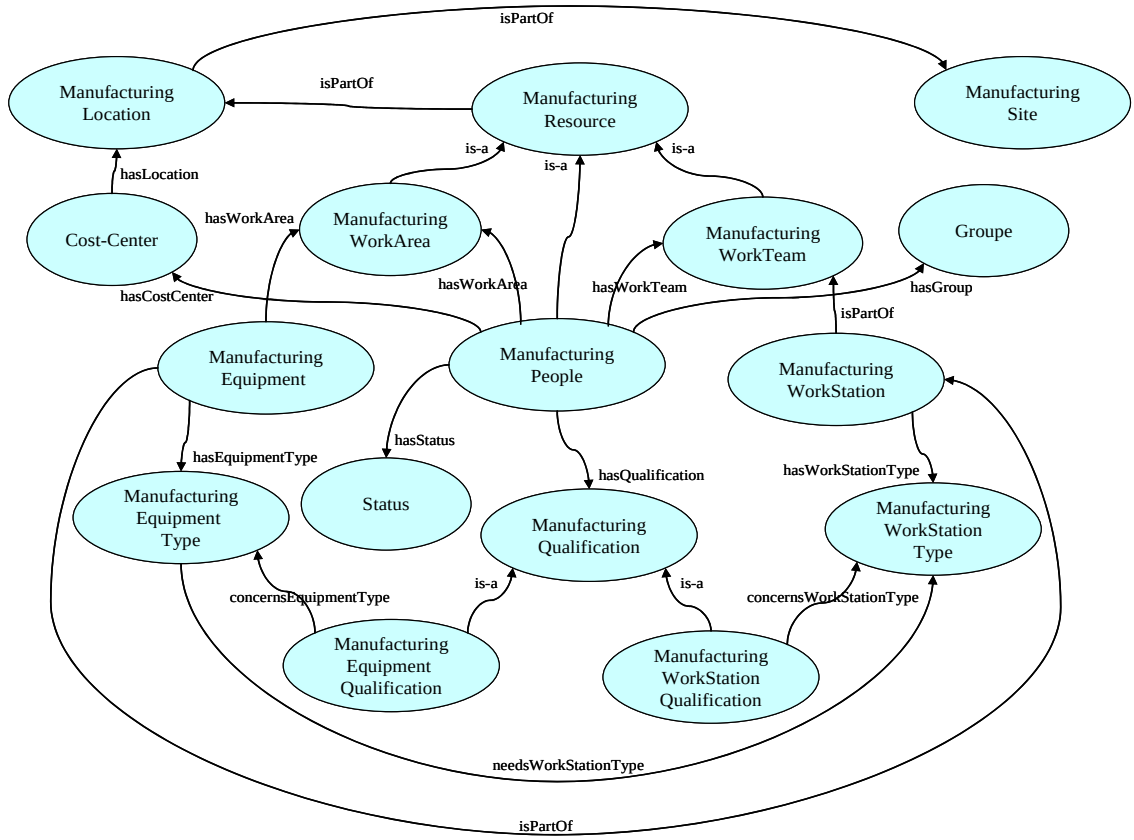

Figure IX.37. Extrait de l'ontologie de données de domaine (MaintenanceArea)

De même que pour les données, nous avons construit les ontologies fonctionnelles de notre cas d'étude. Les figures IX.38, IX.39, IX.40 présentent des extraits des ontologies fonctionnelles. Les deux premières figures concernent les ontologies fonctionnelles locales tandis que la dernière figure concerne l'ontologie fonctionnelle de domaine.

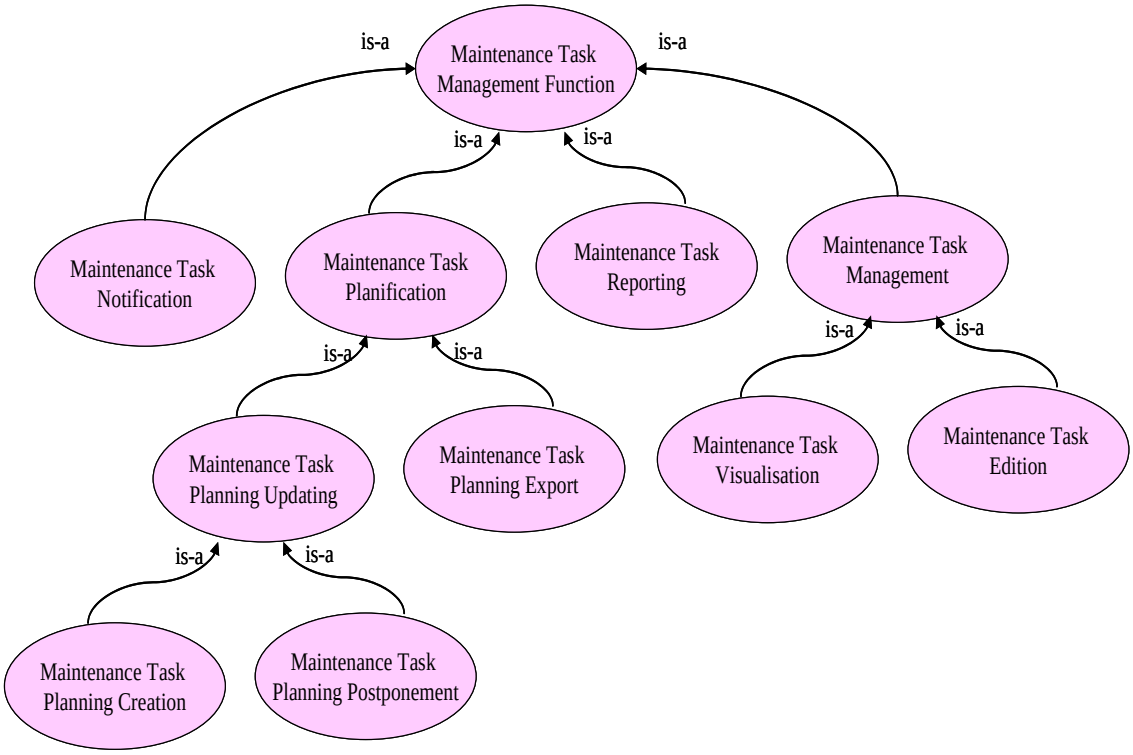

Figure IX.38. Extrait de l'ontologie fonctionnelle locale (MaintenanceTaskDistrict)

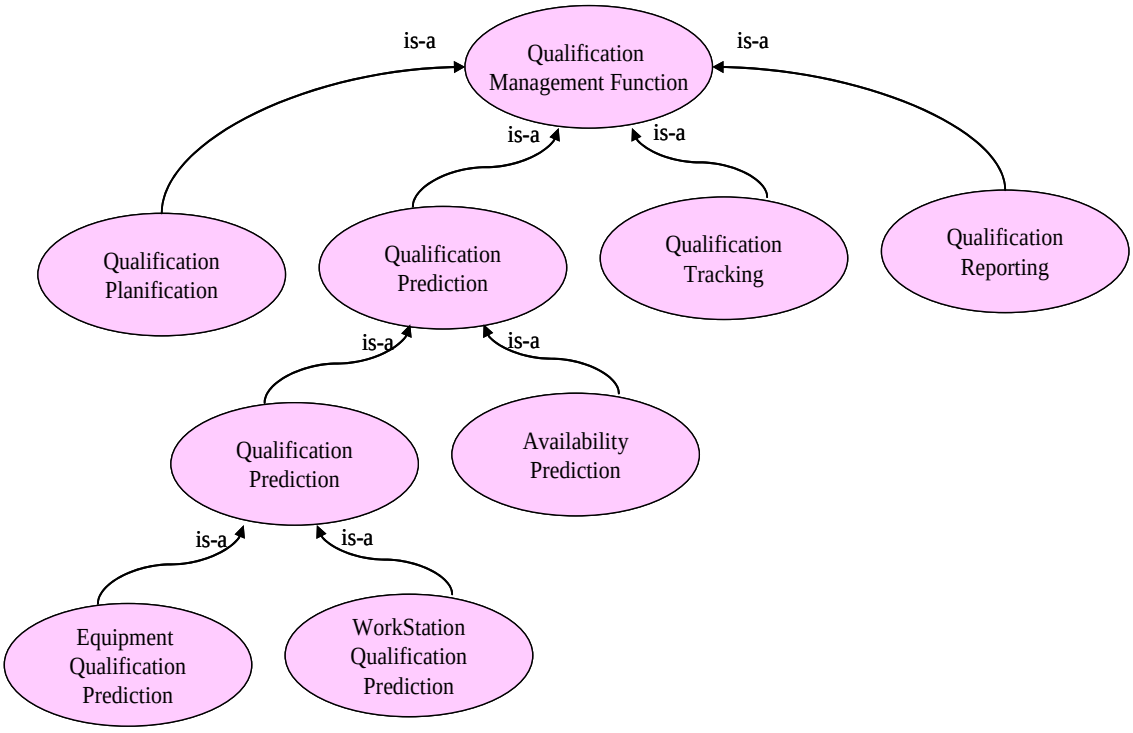

Figure IX.39. Extrait de l'ontologie fonctionnelle locale (MaintenanceOperatorDistrict)

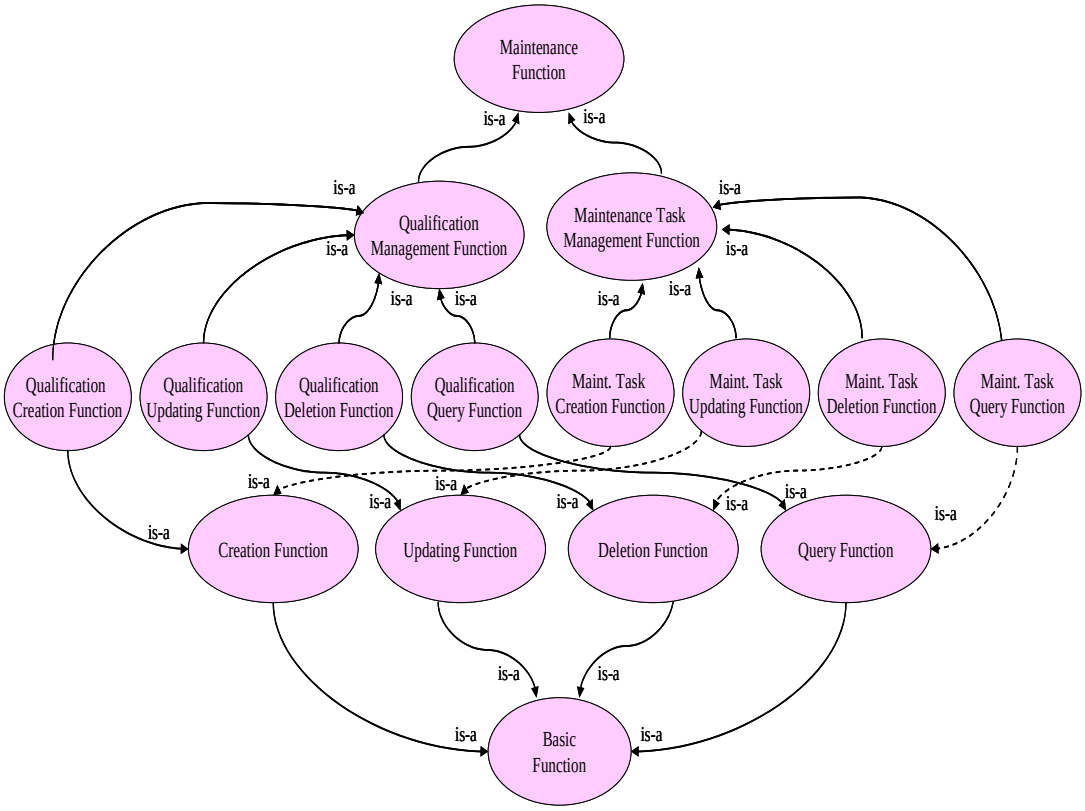

Figure IX.40. Extrait de l'ontologie fonctionnelle de domaine (MaintenanceArea)

IX.4.3.2. Construction des mappings syntaxiques

Une fois que les ontologies locales ont été construites, nous avons procédé à la construction des ontologies de mappings syntaxiques (cf. § VII.6.4). Nous fournissons ci-dessous (figure IX.41) un extrait de cette ontologie qui a été remplie en utilisant le module présenté précédemment et qui mappe dans cet exemple précis la description syntaxique du service SFE_Maintenance_Rapporter avec l'ontologie locale de données correspondante.

```
<SourceOntology rdf:ID ="MaintenanceDataLocalOntology"/>
<TargetSyntaxe rdf:ID ="SFE_Maintenance_Rapporter"/>
 <SyntacticMapping rdf:ID="WorkArea-WSDLWorkArea">
   <targetComponent>
     <SyntacticComponent rdf:ID="WorkAreaType"/>
   </targetComponent>
   <type>
     <TypeMapping rdf:ID="Concept-Type"/>
   </type>
   <sourceComponent>
     <OntologyComponent rdf:ID="WorkArea"/>
   </sourceComponent>
 </SyntacticMapping>
 <SyntacticMapping rdf:ID="MaintenanceUser-WSDLOperator">
   <sourceComponent>
     <OntologyComponent rdf:ID="PopulationType"/>
   </sourceComponent>
   <targetComponent>
     <SyntacticComponent rdf:ID="ManufacuringPeople"/>
   </targetComponent>
   <type rdf:resource="#Concept-Type"/>
 </SyntacticMapping>
 <SyntacticMapping rdf:ID="Equipment-WSDLEquipement">
   <targetComponent>
     <SyntacticComponent rdf:ID="EquipmentType"/>
   </targetComponent>
   <sourceComponent>
     <OntologyComponent rdf:ID="Equipment"/>
   </sourceComponent>
   <type rdf:resource="#Concept-Type"/>
 </SyntacticMapping>
 ...
```

Figure IX.41. Un extrait de l'ontologie de mapping syntaxique

IX.4.3.3. Construction des mappings sémantiques

Nous avons également procédé à la construction des ontologies de mappings sémantiques (cf. § VII.6.5). La figure ci-dessous fournit un extrait de cette ontologie.

```
<SourceOntology rdf:ID ="MaintenanceDataLocalOntology"/>
<TargetOntology rdf:ID ="MaintenanceDataDomainOntology"/>
<LocalSemanticMapping rdf:ID="WorkArea-ManufacturingWorkArea">
   <mappingComponentType rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
   >Concept</mappingComponentType>
   <sourceComponent>
     <SourceComponent rdf:ID="WorkArea"/>
   </sourceComponent>
   <targetComponent>
     <TargetComponent rdf:ID="ManufacturingWorkArea"/>
   </targetComponent>
   <type>
     <TypeMapping rdf:ID="Equivalence"/>
   </type>
 </LocalSemanticMapping>
 <LocalSemanticMapping rdf:ID="Equipment-ManufacturingEquipement">
   <targetComponent rdf:resource="#Equipment"/>
   <mappingComponentType rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
   >Concept</mappingComponentType>
   <type rdf:resource="#Equivalence"/>
   <sourceComponent>
     <SourceComponent rdf:ID="Equipment"/>
   </sourceComponent>
 </LocalSemanticMapping>
 <TargetComponent rdf:ID="ManufacuringPeople"/>
 <LocalSemanticMapping rdf:ID="MaintenanceUser-ManufacturingPeople">
   <type rdf:resource="#Equivalence"/>
   <targetComponent rdf:resource="#ManufacuringPeople"/>
   <mappingComponentType rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
   >Concept</mappingComponentType>
   <sourceComponent>
     <SourceComponent rdf:ID="MaintenanceUser"/>
   </sourceComponent>
 </LocalSemanticMapping>
...
```

Figure IX.42. Extrait de l'ontologie de mapping sémantique

IX.4.3.4. Construction des ontologies de service

Nous avons ensuite construit les squelettes d'ontologies de service (cf. § VII.6.6) grâce au module de conversion (cf. § IX.3.3.2.3) que nous avons ensuite complétés (en utilisant l'outil de construction de l'ontologie de service (cf. § IX.3.3.2.3)) afin d'avoir des ontologies de service exploitables (cf. § VII.6.7). Nous fournissons ci-dessous un extrait de l'ontologie de profil de service fondamental qui décrit le service SFE_Maintenance_Rapporter.

Nous pouvons constater, sur la figure IX.43, que nous avons écrit en caractères gras surlignés certains aspects spécifiques de notre ontologie de service. Ces aspects qui constituent l'ontologie OWL-S+ sont implémentés au sein de l'ontologie importée dont l'URI est *"http://www.owl-ontologies.com/eso.owl"*.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE uridef [
  <!ENTITY rdf "http://www.w3.org/1999/02/22-rdf-syntax-ns">
  <!ENTITY rdfs "http://www.w3.org/2000/01/rdf-schema">
  <!ENTITY owl "http://www.w3.org/2002/07/owl">
  <!ENTITY xsd "http://www.w3.org/2001/XMLSchema">
  <!ENTITY service "http://www.daml.org/services/owl-s/1.1/Service.owl">
  <!ENTITY profile "http://www.daml.org/services/owl-s/1.1/Profile.owl">
  <!ENTITY process "http://www.daml.org/services/owl-s/1.1/Process.owl">
  <!ENTITY grounding "http://www.daml.org/services/owl-s/1.1/Grounding.owl">
  <!ENTITY eso "http://www.owl-ontologies.com/eso.owl">
  <!ENTITY odsof "http://www.Odsof.org/2004/owl-s/1.1/OdsofProfileHierarchy.owl">
  <!ENTITY odsof-concept "http://www.owl-ontologies.com/concept-odsof.owl">
  <!ENTITY odsof-view "http://www.owl-ontologies.com/odsof-view.owl">
  <!ENTITY odsof-cluster "http://www.owl-ontologies.com/odsof-cluster.owl">
  <!ENTITY odsof-visibility "http://www.owl-ontologies.com/odsof-visibility.owl">
  <!ENTITY odsof-qos "http://www.owl-ontologies.com/odsof-qos.owl">
]>

<rdf:RDF
  xmlns:rdf="&rdf;#" xmlns:rdfs="&rdfs;#" xmlns:owl="&owl;#" xmlns:xsd="&xsd;#" xmlns:service="&service;#"
  xmlns:profile="&profile;#" xmlns:process="&process;#" xmlns:grounding="&grounding;#" xmlns:eso="&eso;#"
  xmlns:odsof="&odsof;#" xmlns:odsof-concept="&odsof-concept;#" xmlns:odsof-cluster="&odsof-cluster;#"
  xmlns:odsof-view="&odsof-view;#" xmlns:odsof-visibility="&odsof-visibility;#"xmlns:odsof-qos="&odsof-qos;#"
  xml:base="http://www.Odsof.org/2004/owl-s/1.1/SFE_Maintenance_Rapporter.owl"
>

<owl:Ontology rdf:about="">
  <owl:imports rdf:resource="&eso;"/>
  <owl:imports rdf:resource="&odsof;"/>
  <owl:imports rdf:resource="&odsof-cluster;"/>
  <owl:imports rdf:resource="&odsof-view;"/>
  <owl:imports rdf:resource="&odsof-cluster;"/>
  <owl:imports rdf:resource="&odsof-view;"/>
  <owl:imports rdf:resource="&odsof-concept;"/>
  <owl:imports rdf:resource="&odsof-qos;"/>
</owl:Ontology>

<!-- Enterprise Service description -->
  <service:Service rdf:ID="SFE_Maintenance_RapporterService">
  <service:presents rdf:resource="#SFE_Maintenance_RapporterProfile"/>
  <service:describedBy rdf:resource="#SFE_Maintenance_RapporterProcess"/>
  <service:supports rdf:resource="#SFE_Maintenance_RapporterGrounding"/>
  </service:Service>
<!-- Enterprise Service description -->

<!-- Enterprise Service Profile description -->
<eso:EnterpriseFundamentalServiceProfile rdf:ID="SFE_Maintenance_RapporterFundamentalProfile"/>
  <odsof:FunctionService rdf:ID="SFE_Maintenance_Rapporter">
    <service:isPresentedBy rdf:resource="#SFE_Maintenance_RapporterService"/>
    <profile:serviceName xml:lang="en">SFE_Maintenance_Rapporter</profile:serviceName>
    <profile:textDescription xml:lang="en">Service fonctionnel de gestion de la maintenance
</profile:textDescription>
    <profile:hasInput rdf:resource="&odsof-concept;#Population"/>
    <profile:hasInput rdf:resource="&odsof-concept;#StartDate"/>
    <profile:hasInput rdf:resource="&odsof-concept;#EndDate"/>
    <profile:hasInput rdf:resource="&odsof-concept;#Equipment"/>
    <profile:hasOutput rdf:resource="#QualificationResponse"/>
    <eso:enterpriseView rdf:resource="&odsof-view;#MaintenanceManagementQueryFunction/>
    <eso:serviceCluster rdf:resource="&odsof-cluster;#MaintenanceTaskDistrict"/>
    <eso:qualityOfService rdf:resource="&odsof-qos;#HighQuality"/>
    <eso:serviceVisibility rdf:resource="&odsof-visibility;#AllVisibilities"/>
  </odsof:FunctionService>
<eso:EnterpriseFundamentalServiceProfile>
<!-- Enterprise Service Profile description -->

</rdf:RDF>
```

Figure IX.43. Extrait de l'ontologie de profil de service fondamental complétée

IX.4.3.5. Construction des méta-ontologies

Une fois que les ontologies d'entreprise sont construites, nous leurs avons associés des profils (cf. § VII.6.8) afin de pouvoir les publier par la suite. Nous reproduisons ci-dessous quelques extraits de profils d'ontologies destinés à leur publication.

```
<?xml version="1.0"?>
<rdf:RDF
    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
    xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
    xmlns:owl="http://www.w3.org/2002/07/owl#"
    xmlns:meta-onto="http://www.owl-ontologies.com/meta-onto.owl#"
    xmlns="http://www.owl-ontologies.com/metaOntology.owl#"
  xml:base="http://www.owl-ontologies.com/metaOntology.owl">
  <owl:Ontology rdf:about="">
    <owl:imports rdf:resource="http://www.owl-ontologies.com/meta-onto.owl"/>
  </owl:Ontology>
  <meta-onto:EnterpriseOntologyProfile rdf:ID="MaintenanceTaskDataLocalOntologyProfile">
    <meta-onto:ontology>
      <meta-onto:EnterpriseOntology rdf:ID="MaintenanceTaskDataLocalOntology"/>
    </meta-onto:ontology>
    <meta-onto:ontologyLevel rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >Local</meta-onto:ontologyLevel>
    <meta-onto:version rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >1.0</meta-onto:version>
    <meta-onto:ontologyType rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >Data</meta-onto:ontologyType>
    <meta-onto:date rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >2006-03-13</meta-onto:date>
    <meta-onto:serviceCluster>
      <meta-onto:ServiceCluster rdf:ID="MaintenanceTaskManagementDistrict"/>
    </meta-onto:serviceCluster>
    <meta-onto:name rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >MaintenanceTaskDataLocalOntologyProfile</meta-onto:name>
  </meta-onto:EnterpriseOntologyProfile>
  <meta-onto:EnterpriseOntology rdf:ID="QualificationDataLocalOntology"/>
  <meta-onto:ServiceCluster rdf:ID="QualificationManagementDistrict"/>
  <meta-onto:EnterpriseOntologyProfile rdf:ID="QualificationDataLocalOntologyProfile">
    <meta-onto:serviceCluster rdf:resource="#QualificationManagementDistrict"/>
    <meta-onto:version rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >1.0</meta-onto:version>
    <meta-onto:name rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >QualificationDataLocalOntologyProfile</meta-onto:name>
    <meta-onto:ontologyType rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >Data</meta-onto:ontologyType>
    <meta-onto:ontologyLevel rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >Function</meta-onto:ontologyLevel>
    <meta-onto:ontology rdf:resource="#QualificationDataLocalOntology"/>
    <meta-onto:date rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >2006-03-13</meta-onto:date>
  </meta-onto:EnterpriseOntologyProfile>
  <meta-onto:EnterpriseOntology rdf:ID="MaintenanceDataDomainOntology">
    <meta-onto:EnterpriseOntologyRepository>
      <meta-onto:EnterpriseOntologyProfile rdf:ID="MaintenanceDataDomainOntologyProfile">
        <meta-onto:version rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
        >1.0</meta-onto:version>
        <meta-onto:date rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
        >2006-03-13</meta-onto:date>
        <meta-onto:ontologyLevel rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
        >Domain</meta-onto:ontologyLevel>
        <meta-onto:serviceCluster>
          <meta-onto:ServiceCluster rdf:ID="MaintenanceArea"/>
        </meta-onto:serviceCluster>
        <meta-onto:ontologyType rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
        >Data</meta-onto:ontologyType>
        <meta-onto:name rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
        >MaintenanceDataDomainOntologyProfile</meta-onto:name>
      </meta-onto:EnterpriseOntologyProfile>
    </meta-onto:EnterpriseOntologyRepository>
  </meta-onto:EnterpriseOntology>
…
```

Figure IX.44. Extraits de profils d'ontologies d'entreprise

IX.4.3.6. Publication des ontologies d'entreprise

Une fois que les méta-ontologies sont construites (cf. § VII.6.8), et en particulier la partie portant sur les profils d'ontologies, nous procédons alors à leur publication. Ceci va permettre aux divers modules de run-time de pouvoir localiser dynamiquement les ontologies dont ils ont besoin. Pour cela, nous avons mis en place un référentiel implanté dans sa version 1.0 sous forme d'une base de données relationnelle (mySQL). De même que pour le référentiel de services, le référentiel d'ontologies devra évoluer à terme pour devenir le référentiel sémantique (d'ontologies) du site Rousset de l'entreprise STMicroelectronics. D'autres référentiels sont, bien entendu, à prévoir lors de la généralisation du projet MiMISI (cf. § I.2).

IX.4.4. Utilisation de la couche d'intégration

Une fois que l'architecture de services et l'architecture d'ontologies ont été construites, les utilisateurs pourront alors définir des scénarios d'intégration en utilisant les modules de run-time de notre prototype.

Rappelons que parmi les requêtes auxquelles le module de run-time (dans sa version actuelle) peut répondre nous pouvons citer (cf. § VIII.4.1.2.3) :

- les requêtes de découverte qui comprennent les requêtes de service, les requêtes de données, les requêtes fonctionnelles,
- les requêtes de médiation qui comprennent les requêtes de médiation de données et les requêtes de médiation fonctionnelle,
- et les requêtes d'invocation qui permettent d'exécuter un service particulier.

La figure IX.45 permet de rappeler les utilisations typiques de notre prototype en mode run-time dans le cadre de notre étude de cas.

Comme illustré sur la figure IX.45, le module de run-time exploite les ontologies de service (OSE - Ontologie de Service d'Entreprise) et les ontologies fondamentales[71] pour effectuer certaines tâches d'intégration dont les plus typiques sont celles portant sur la découverte de services et celles portant sur la médiation de services.

Dans le cadre de notre expérimentation, nous avons principalement testé les fonctionnalités du module run-time qui portent sur la découverte et la médiation de services au niveau données.

[71] Signalons aussi que le terme "ontologie fondamentale locale" utilisé dans la figure est quelque peu ambigu du fait qu'il renvoie réellement à deux ontologies fondamentales locales dans notre cas et qui sont : l'ontologie de données locale et l'ontologie fonctionnelle locale.

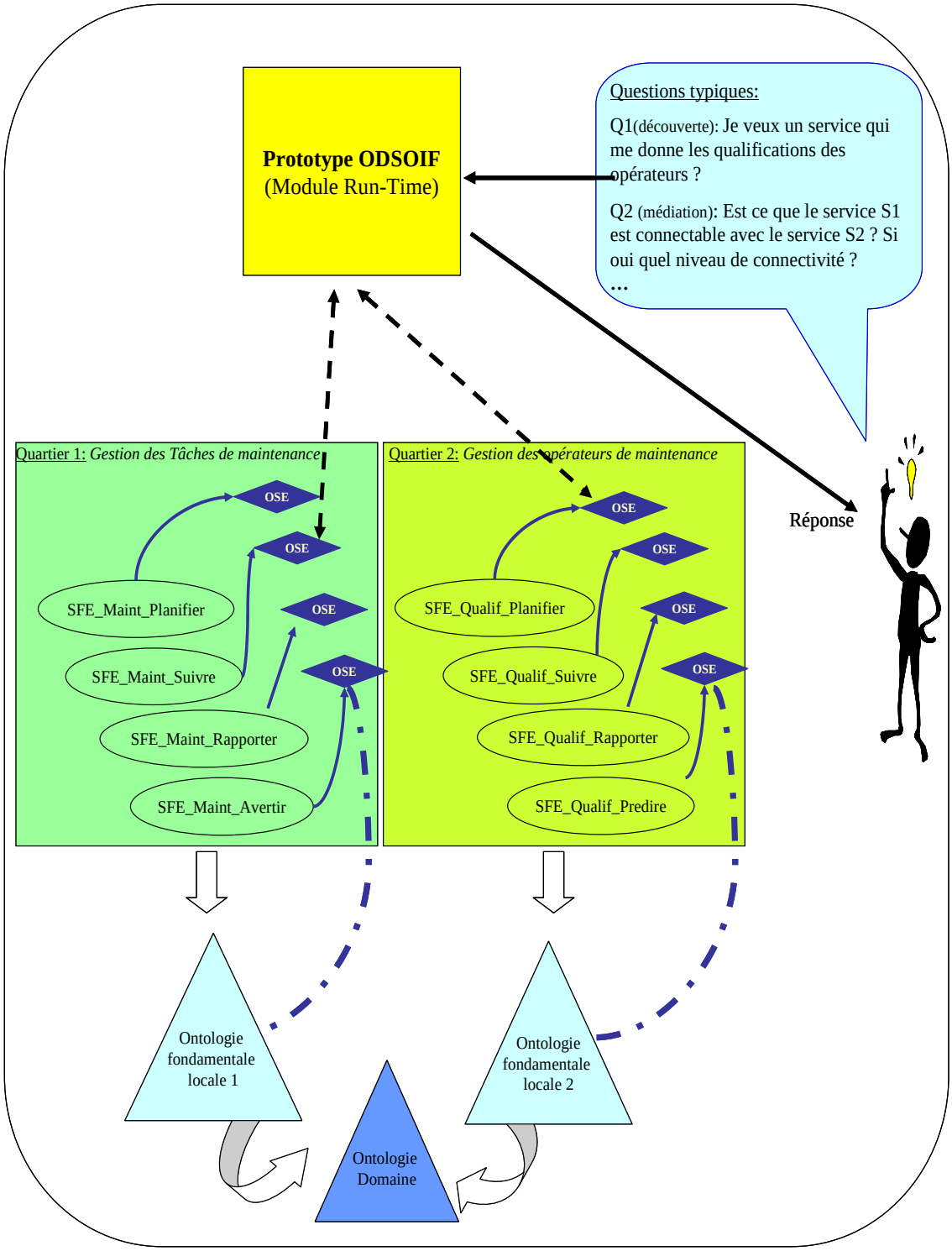

Figure IX.45. Scénario typique d'utilisation du module run-time d'intégration

IX.4.4.1. Découverte de services d'entreprise

L'expérimentation de la découverte de services a permis de tester le module de découverte (cf. § IX.3.4.2) sur un échantillon de services d'entreprise. Pour cela, nous avons formulé une multitude de requêtes portant principalement sur les trois aspects de services : le contexte, la signature, et l'aspect non fonctionnel.

La figure IX.46 illustre un exemple simple de requête ainsi que les réponses restituées par le prototype. La requête porte sur la découverte de services fonctionnels (*FunctionService*) du cluster Maintenance (*MaintenanceArea*) et produisant des qualifications (*Qualification*). Le module de découverte exécute alors l'algorithme de comparaison (cf. § VIII.4.3.2.1) et restitue alors les réponses triées selon l'ordre décroissant du degré de similarité.

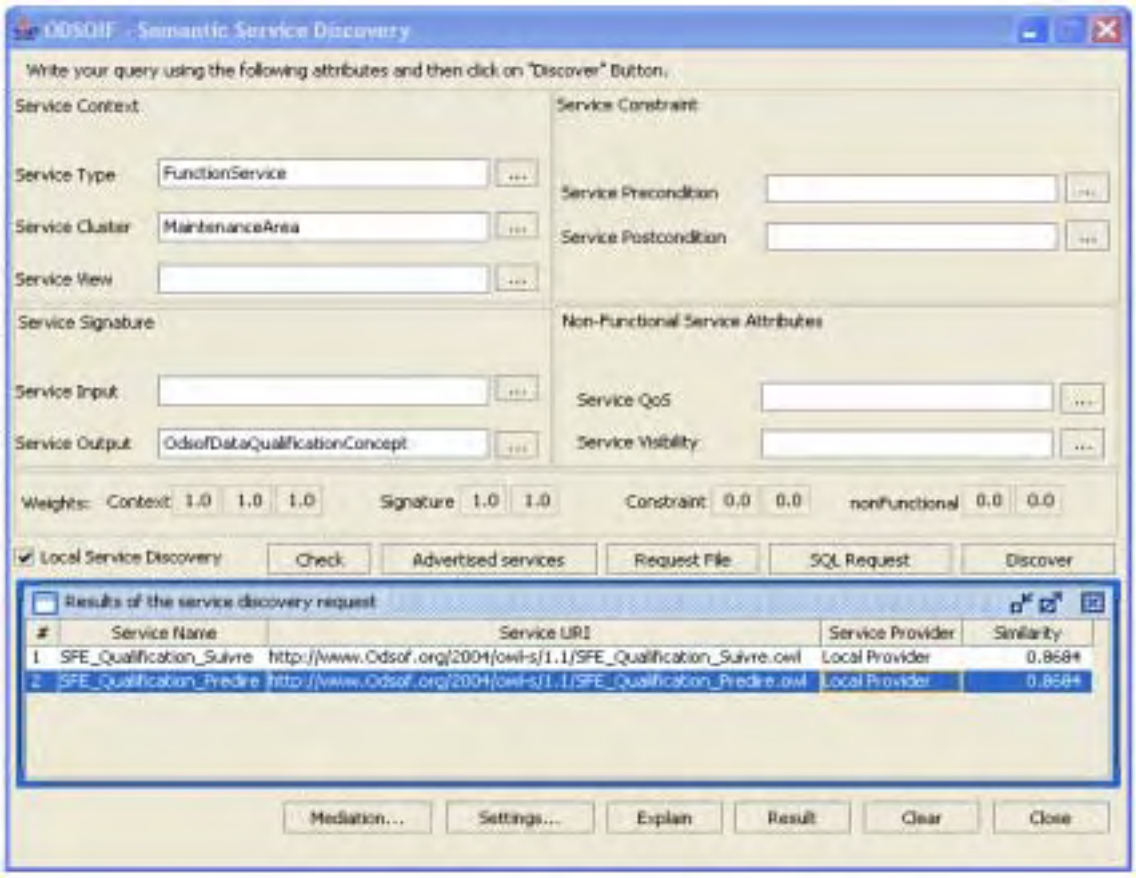

Figure IX.46. Découverte de services d'entreprise

Comme nous pouvons le remarquer sur la figure précédente, le module de découverte restitue deux réponses de même degré de similarité. Ceci s'explique par le fait que la comparaison est principalement effectuée sur la base de l'output (car les autres éléments renseignés sont identiques dans ce cas pour les deux services restitués). Et comme ces deux services produisent tous les deux le même output désiré, c'est la raison pour laquelle les deux services présentent le même degré de pertinence (`0.8684`). Afin de pallier ce genre de problème, nous exploitons alors l'ontologie fonctionnelle qui permet de préciser si le service que l'on veut porte sur les tâches de planification, de prédiction, du reporting, etc. Ceci a pour intérêt d'améliorer les performances de l'outil (réduction du bruit).

En exploitant l'élément *Service View* de l'interface de découverte, nous pouvons saisir le fait que nous nous intéressons non pas à la fonction de suivi des qualifications mais plutôt à la fonction de prédiction. En indiquant dans la zone de texte du champ *Service View* le concept fonctionnel "*QualificationQueryFunction*" qui est l'un des concepts de l'ontologie fonctionnelle de domaine le plus proche de la fonction prédiction, nous arrivons à différencier substantiellement les deux réponses précédentes et nous obtenons respectivement les degrés de similarité suivants : `0,2921` et `0,8684`. Ces degrés peuvent être affinés davantage en faisant des recherches non pas au niveau de la zone mais plutôt au niveau des quartiers. Ce changement de cluster a pour effet de sélectionner les concepts des ontologies fonctionnelles locales qui permettent d'obtenir ainsi des réponses plus précises.

Nous devons noter que l'interface graphique du module de découverte s'est révélée importante pour guider le processus de recherche de services. L'utilisateur peut saisir d'une façon conviviale tous les éléments de sa requête. En cliquant sur les petits boutons de recherche qui accompagnent chaque zone de texte correspondant à chaque élément de la requête, le système affiche l'ontologie appropriée (en fonction du type de l'élément) sous forme d'une arborescence, et l'utilisateur construit alors sa requête en choisissant le concept désiré. Il peut également agir sur les poids des aspects de la requête dont les valeurs par défaut sont prises égales à 1. Deux améliorations notables que nous avons prévues dans les prochaines versions du prototype sont la prise en compte des valeurs numériques dans la formulation des requêtes (prendre en compte une fiabilité de service supérieure à 0.9) et aussi la possibilité de visualiser les réponses sur la cartographie graphique des services d'entreprise (visualisation graphique conviviale des réponses).

IX.4.4.2. Médiation de services

L'expérimentation de la médiation de services dans le cadre de notre étude de cas consiste à tester quelques scénarios d'intégration de services d'entreprise. Le scénario typique sur lequel nous nous sommes basés pour notre expérimentation est celui qui consiste à savoir si deux services sont connectables, à établir ensuite la connexion dans le cas affirmatif et à suggérer des services compatibles dans le cas contraire (cf. § VIII.4.4).

Nous allons dans ce qui suit décrire un cas de médiation que nous avons expérimenté. Il s'agit de la médiation de deux services d'entreprise qu'un utilisateur désire connecter en vue de définir un service processus plus complexe. Il s'agit des deux services fonctionnels *SFE_Qualification_Predire* et *SFE_Maintenance_Planifier*. Ces deux services peuvent éventuellement être le résultat d'une découverte sémantique telle que décrite dans la section précédente.

Comme les deux services considérés (*SFE_Qualification_Predire* et *SFE_Maintenance_Planifier*) sont des services sémantiques, le prototype (cf. § IX.3.4.3) teste alors leur connectivité sémantique. Le module de médiation retourne une réponse positive pour la requête de connectivité sémantique. Cette réponse positive nous permet de continuer le processus de médiation (car si la réponse était négative, le processus s'arrêterait du fait que les deux services ne sont pas sémantiquement connectables). Le prototype vérifie alors si les deux services présentent une connectivité syntaxique, et le prototype répond par la négation. Le prototype effectue alors une transformation dirigée par la sémantique des données en procédant de la même manière que celle exposée dans la section § VIII.4.4.1 (figure IX.47).

Dans notre cas, la transformation dirigée par la sémantique des données exploite les ontologies de données locales (*OWL-D-1* et *OWL-D-2*) et l'ontologie de données de domaine (OWL-DD) ainsi que les mappings sémantiques et les mappings syntaxe-sémantique.

L'ontologie de données locale *OWL-D-1* qui est l'opérationnalisation de l'ontologie de données locale du quartier *MaintenanceOperatorDistrict,* est déjà présentée sous forme conceptuelle en figure IX.36. De même, l'ontologie de données locale *OWL-D-2* est l'opérationnalisation de l'ontologie de données locale du quartier *MaintenanceTaskDistrict*. Elle est présentée sous forme conceptuelle en figure IX.35. Quant à l'ontologie de données de domaine *OWL-DD,* elle est l'opérationnalisation de l'ontologie de données de domaine de la zone *MaintenanceArea*. Elle est déjà présentée sous forme conceptuelle en figure IX.37.

Les mappings sémantiques qui relient les ontologies de données locales à l'ontologie de domaine) sont tels que décrits dans la section IX.4.3.3. Ce sont ces mappings qui permettent ensuite d'établir les liens qui peuvent exister entre les concepts des deux ontologies locales. Nous avons présenté sur la figure quelques exemples de mappings. Il s'agit des mappings qui permettent de lier la propriété *manufacturingPeople* de l'ontologie de domaine aux deux propriétés *maintenaceUser* et *people* des ontologies locales des deux quartiers.

Les mappings syntaxe-sémantique qui relient les ontologies de données locales aux schémas syntaxiques de données sont tels que décrits dans la section IX.4.3.2. Ce sont ces mappings qui permettent d'établir les liens qui peuvent exister entre les concepts des ontologies locales avec les éléments des fichiers WSDL. Plus précisément, les mappings syntaxe-sémantique permettent de relier l'ontologie locale *OWL-D-1* à la description syntaxique *XSD-1* et l'ontologie locale *OWL-D-2* à la description syntaxique *XSD-2*.

En exploitant ces différents éléments, le module de médiation transforme les données de sortie du premier service en données d'entrée du second service. Cette médiation est typiquement effectuée à la volée. Ceci apporte de la flexibilité mais rend le système plus lent du fait que le chemin d'exécution est long. Ceci constitue la principale limitation de notre module de médiation. Afin d'augmenter les performances, des améliorations sont prévues. Il s'agit en particulier de dériver de façon automatique la transformation syntaxique (en utilisant XSLT ou XQuery) et de la sauvegarder pour une utilisation ultérieure, permettant ainsi une intégration statique des systèmes industriels.

Nous devons également noter que les tests effectués sont relativement simples. Nous n'avons pas encore effectué des transformations complexes. Pour prendre en charge les transformations complexes, nous avons déjà envisagé d'utiliser la possibilité de mappings complexes, tels que nous les avons prévus, en utilisant des langages de transformation standardisés tels que XSLT ou XQuery. Ceci permettrait de renforcer le module de médiation en le dotant de routines plus performantes permettant d'effectuer des transformations complexes.

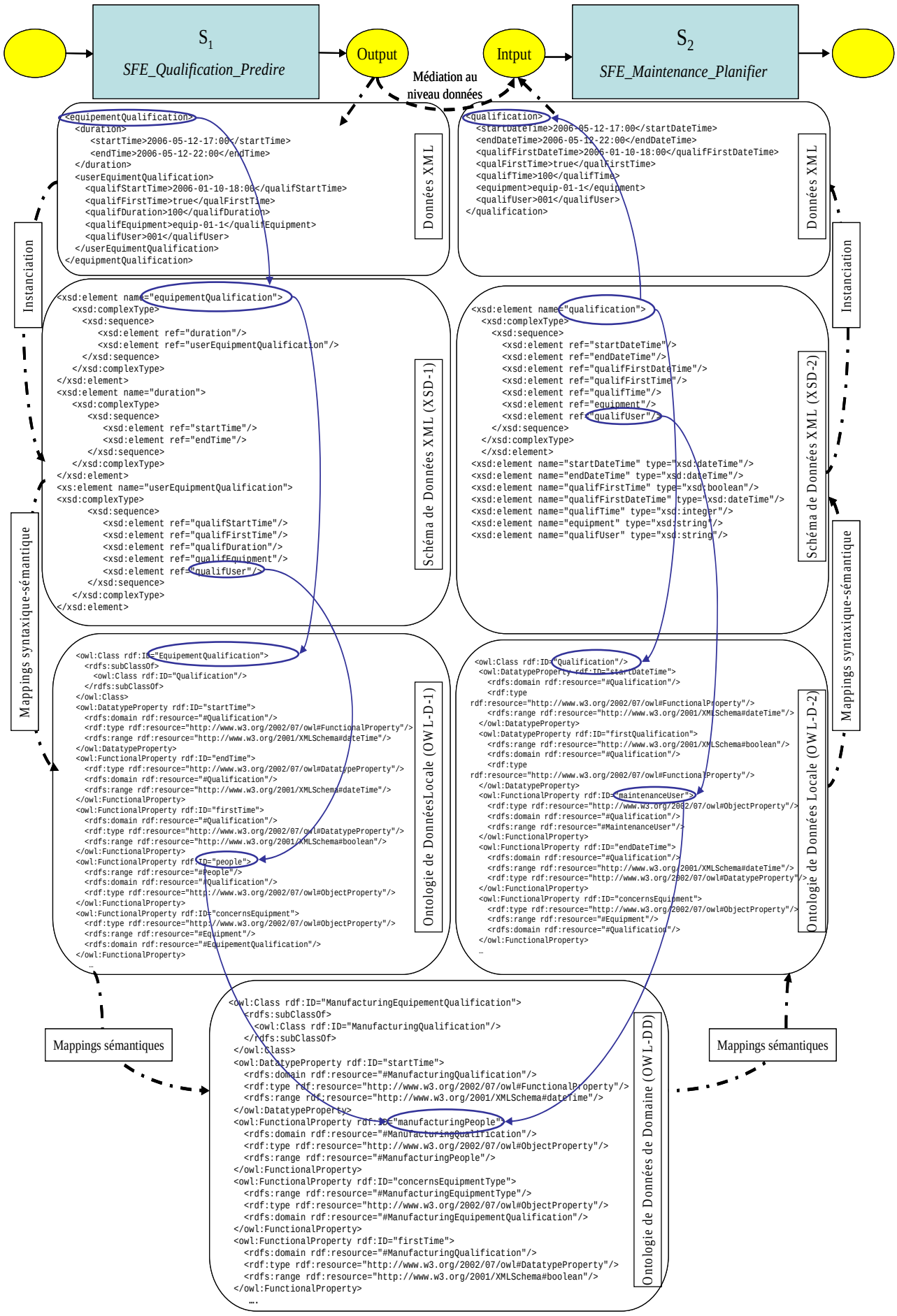

Figure IX.47. Exemple d'expérimentation de la médiation de services

IX.5. Conclusion

Dans ce chapitre, nous avons présenté une implémentation et une expérimentation du prototype ODSOIF (Ontology-Driven Service-Oriented Integration Framework) qui a pour but de valider certains éléments du cadre méthodologique d'intégration que nous avons proposé dans les chapitres précédents (chapitres VI, VII et VIII).

Le prototype est développé sous forme de deux modules complémentaires: le module de design-time et le module de run-time (cf. § IX.3.2). Le premier module (design-time) fournit les fonctionnalités essentielles pour construire un projet d'intégration sémantique de services d'entreprise. Il permet plus précisément de construire l'architecture de services d'entreprise (ASE) (cf. § VI.4.2) et l'architecture d'ontologies d'entreprise (AOE) (cf. § VII.6.2). Le second module (run-time) permet de construire l'architecture d'intégration d'entreprise (AIE) en fournissant les fonctionnalités essentielles pour mettre en œuvre certains mécanismes d'intégration basés sur la sémantique des services d'entreprise (cf. § VIII.4).

Nous avons également expérimenté notre prototype sur un cas réel, celui de la gestion de la maintenance au sein de la société STMicroelectronics. Pour cela, nous avons construit une architecture de services d'entreprise sur laquelle nous avons ensuite greffé une architecture d'ontologies. Nous avons ensuite expérimenté certains mécanismes d'intégration dont la découverte et la médiation de services. L'expérimentation a révélé que l'approche est plus flexible car l'intégration se fait au niveau de la sémantique, mais en revanche elle a révélé aussi le fait qu'il peut y avoir des problèmes de performances dus aux longs chemins d'exécution et aussi au manque de performances de certains outils qui gèrent les ontologies.

Nous devons noter que tous les éléments de notre cadre méthodologique et/ou de notre prototype n'ont pas été testés sur l'étude de cas en raison du nombre encore limité de services disponibles et de la faible prise de conscience des utilisateurs de l'intérêt des approches sémantiques dans l'intégration des applications industrielles. Aussi, il devient intéressant et pertinent de tester et d'expérimenter notre prototype sur un périmètre plus élargi. De plus, nous devons noter que certaines fonctionnalités du prototype sont actuellement implémentées de façon relativement simplifiée et en conséquence nous demeurons convaincus, compte tenu de l'intérêt de cette approche, que ce prototype doit être soutenu et poursuivi afin de lui apporter toutes les améliorations nécessaires.

CONCLUSION

Chapitre X

CONCLUSIONS ET PERSPECTIVES

X.1. Rappel du cadre et des objectifs de la thèse

Les travaux décrits dans cette thèse portent sur la problématique d'intégration sémantique des applications d'entreprise, et en particulier des applications du secteur de la microélectronique. Ces travaux sont réalisés dans le contexte d'une collaboration industrielle entre l'Ecole des Mines de Saint-Étienne et la société STMicroelectronics, dans le cadre du projet MiMISI (**Mi**croelectronics **M**anufacturing **I**nformation **S**ystem **I**ntegration). Ces travaux s'inscrivent dans le cadre de la démarche d'intégration initiée par cette société pour répondre à la problématique d'intégration flexible des applications industrielles. Ils viennent compléter certains travaux de [Chapron 2006] qui viennent de s'achever et qui portent sur la problématique de gestion de l'évolution des systèmes d'information dans un environnement réactif.

Ces travaux ont pour principal objectif de concevoir une architecture d'intégration flexible pouvant permettre de mieux interconnecter les applications industrielles. Ils se sont focalisés sur trois sous-problématiques complémentaires qui sont respectivement la problématique de construction (ou de migration vers) d'une architecture de services (P_{Syn}), la problématique d'enrichissement sémantique des services (P_{Sem}), la problématique de construction de l'architecture d'intégration permettant d'offrir des mécanismes d'intégration basés sur la sémantique (P_{Int}).

Une étude de l'art sur l'intégration des Systèmes d'Information nous a permis de retenir principalement deux types d'approches d'intégration qui peuvent être mises en œuvre dans le domaine industriel, et en particulier dans le domaine de la microélectronique et qui sont : les approches syntaxiques et les approches sémantiques. Nous nous sommes intéressés à ces dernières car elles constituent le moyen le plus flexible intégrer des applications d'entreprise. L'analyse des solutions d'intégration sémantique actuelles, et en particulier des approches basées sur la notion de services sémantiques, a montré un certain nombre de limites. Il s'agit notamment des limites liées aux architectures des ontologies actuelles, au manque de méthodologies efficaces pour construire les ontologies d'entreprise et les services sémantiques, au manque de maturité des mécanismes de description et de découverte de services et enfin au manque de mécanismes de médiation de services. Partant de ce constat, nous nous sommes intéressés à la recherche d'une méthodologie et d'une solution flexible d'intégration pouvant être appliquée dans le cadre de grandes entreprises en générale et dans le cadre du projet MiMISI en particulier. Nous avons alors proposé une approche flexible basée

sur les services sémantiques, que nous appelons approche ODSOI (**O**ntology-**D**riven **S**ervice-**O**riented **I**ntegration).

X.2. Principales contributions

L'approche que nous proposons (ODSOI) se base sur un enrichissement de l'approche OWL-S, qui est à notre sens l'initiative la plus mature en matière de développement et d'intégration de services sémantiques. Cette approche repose sur trois principes majeurs qui sont l'ouverture, l'unification et l'urbanisation. Le principe d'ouverture, qui est plutôt une contrainte industrielle, impose de s'inscrire dans le cadre d'utilisation de standards industriels. Le principe d'unification permet de représenter de façon uniforme les différents composants du système d'information d'entreprise. Le principe d'urbanisation permet de définir des règles de structuration des services d'entreprise, des ontologies d'entreprise et aussi des services d'intégration. Chacun de ces principes est fondamental dans le sens où il intervient dans la réalisation de chacune de nos trois sous-problématiques.

En tenant compte de ces principes, nous avons alors proposé de construire un modèle d'intégration en trois couches (ou architectures) hiérarchisées. La construction de chaque couche constitue une contribution principale de notre travail. Elle répond à une des sous-problématiques précédemment citées (***P*$_{Sem}$**, ***P*$_{Sem}$**, et ***P*$_{Int}$**). Ces trois contributions principales sont :

- La construction de l'*architecture de services d'entreprise* (ASE) qui permet de définir l'ensemble des services d'entreprise. Les principales contributions que nous avons apportées lors de la construction de cette architecture portent sur la dichotomie SOA IT/SOA métier et aussi sur la proposition d'une démarche de construction. La dichotomie SOA IT / SOA métier est motivée par le fait qu'elle permet de distinguer les deux préoccupations complémentaires qui sont celle de la maîtrise d'œuvre et celle de la maîtrise d'ouvrage. La démarche de construction, quant à elle, permet de définir les différentes étapes nécessaires pour la construction de notre architecture de services. Un point pertinent de notre démarche de construction est l'urbanisation des services. Cette dernière permet d'étendre le concept d'urbanisation du système d'information aux architectures orientées services en proposant ainsi une urbanisation orientée services du système d'information d'entreprise. Cette urbanisation est importante car elle permet de définir des regroupements de services qui servent de base pour l'identification des îlots sémantiques.
- La construction de l'*architecture d'ontologies d'entreprise* (AOE) qui permet de définir l'ensemble des ontologies d'entreprise permettant de décrire la sémantique des services d'entreprise. Les principales contributions que nous avons apportées lors de la construction de cette architecture sémantique portent sur la typologie des ontologies d'entreprise et la démarche de construction de ces ontologies. En effet, notre approche se base sur une typologie qui distingue deux types d'ontologies d'entreprise: les ontologies de service et les ontologies fondamentales. Les ontologies de services, qui sont une extension de l'ontologie OWL-S, permettent de décrire la sémantique des services d'entreprise. En revanche les ontologies fondamentales, qui sont référencées au sein des ontologies de services, permettent de décrire la sémantique spécifique des systèmes d'information d'entreprise. La démarche de construction, quant à elle, définit les différentes étapes permettant la

construction des ontologies de services et des ontologies fondamentales. Un aspect majeur de notre démarche de construction d'ontologies d'entreprise est l'urbanisation. En effet, nos ontologies d'entreprise sont urbanisées dans le sens où elles sont regroupées en clusters d'ontologies. Cette urbanisation permet de construire une architecture flexible où les changements dans un cluster impactent moins les autres clusters.

- La construction de l'*architecture d'intégration d'entreprise* (AIE) qui permet de définir les mécanismes permettant d'intégrer les services d'entreprise (définis dans l'ASE) en se basant sur les ontologies d'entreprise (définies dans l'AOE). Les principales contributions que nous avons apportées lors de la construction de cette architecture sont l'amélioration du mécanisme de découverte de services d'entreprise et la proposition d'un mécanisme de médiation sémantique des services d'entreprise. Notre approche de découverte de services se base sur un algorithme de comparaison qui exploite des indices de similarité. Ces indices mesurent le degré de similitude entre une requête et une annonce. Ils sont évalués grâce à la similarité des concepts d'ontologies d'entreprise mesurée à l'aide des mécanismes d'inférence. Notre approche de médiation, quant à elle, permet d'effectuer l'intégration des services au niveau de données et au niveau des fonctions en se basant principalement sur les ontologies de données et les ontologies fonctionnelles. De plus, notre architecture d'intégration repose sur le principe de cohabitation syntaxe-sémantique qui permet de prendre en compte l'intégration des services non encore enrichis sémantiquement, ce qui permet d'apporter de la flexibilité durant le projet d'intégration.

X.3. Principales limites

Le travail présenté dans cette thèse est guidé par les nombreuses questions issues des préoccupations industrielles. Comme la problématique d'intégration est généralement complexe, nous nous sommes efforcés tout au long de ce travail de prendre suffisamment de recul afin de proposer un cadre relativement global et complet afin de mieux guider les utilisateurs dans leur processus d'intégration flexible. Cependant, le travail présenté demeure incomplet et limité du fait que certaines limitations ont été effectuées. Il s'agit en particulier des limites liées à :

- la construction de l'architecture de services,
- la construction de l'architecture d'ontologies,
- la construction de l'architecture d'intégration,
- l'implémentation du prototype,
- et l'expérimentation du prototype.

X.3.1. Limites liées à la construction de l'architecture de services

Les principales limites que nous pouvons mentionner vis à vis de notre architecture de services portent sur la *simplification de notre approche d'urbanisation de services*. En effet, cette dernière est relativement rudimentaire. Elle se base sur la technique TF-IDF pour la détermination des indices de similarité entre les services. Le choix de cette technique est principalement motivé par des raisons de simplicité de mise en œuvre. Il devient alors intéressant d'apporter certaines améliorations afin de rendre plus efficace

le processus d'urbanisation de services. Une autre limite tient au fait que le processus d'urbanisation n'est pas totalement automatique. Ceci veut dire qu'actuellement, le module de calcul des degrés de similarité délivre des résultats qui sont ensuite saisis manuellement dans l'outil externe de clustérisation (dans notre XLSTAT). Des améliorations sont toutefois prévues afin d'intégrer cet outil externe.

X.3.2. Limites liées à la construction de l'architecture d'ontologies

Les principales limites que nous pouvons mentionner en ce qui concerne notre architecture d'ontologies sont :

- *La complexité du modèle sémantique* : Le modèle sémantique que nous avons proposé est relativement complexe du fait qu'il a pour ambition d'apporter une solution relativement complète. Cette complexité doit, à notre sens, être masquée par des outils appropriés qui vont assister les utilisateurs dans leur processus de construction d'ontologies, et c'est dans cet objectif que nous avons proposé de développer un prototype.
- *Le peu d'automatisation du processus de construction d'ontologies fondamentales* : Le processus de construction des ontologies fondamentales tel que nous le préconisons jusqu'à présent est principalement manuel. Toutefois ce processus peut grandement gagner en efficacité s'il est supporté par des outils semi-automatiques d'extraction de la sémantique. Ces outils semi-automatiques devront normalement se baser sur les clusters définis lors de la construction de l'architecture de services et déduire des esquisses des ontologies locales, puis des esquisses des ontologies de domaine et enfin une esquisse de l'ontologie globale, en se basant notamment sur les descriptions des services issus de ces clusters. Ces esquisses seront présentées aux utilisateurs (cogniticiens, intégrateurs, etc.) qui vont alors leur apporteront les ajustements nécessaires.
- *La non prise en compte de la maintenance des ontologies* : Une autre limite, ou plutôt une lacune, non moins importante est l'absence du processus de maintenance des ontologies. En effet, notre approche s'est focalisée sur la construction des ontologies et a totalement négligé l'aspect maintenance.

X.3.3. Limites liées à l'architecture d'intégration

Les principales limites que nous pouvons citer en ce qui concerne notre architecture d'intégration sont :

- *La non exhaustivité de l'étude menée* : Notre étude est relativement incomplète du fait que nous avons négligé certains aspects d'intégration, notamment l'intégration au niveau des données, l'intégration au niveau du processus et la distribution. En effet, l'intégration au niveau des données qui ne fait intervenir que les services de données n'a pas été traitée mais il a été prévu qu'elle soit traitée dans le cadre des travaux de [Ghurbhurn 2006]. Quant à l'intégration au niveau du processus, qui peut s'apparenter à un prolongement de l'intégration au niveau des fonctions, elle n'a pas été totalement traitée dans le cadre de ces travaux. Nous nous sommes contraints, faute de temps, à négliger ces aspects. De plus, l'architecture d'intégration que nous avons proposée est semi centralisée voire centralisée lors de l'implémentation des référentiels logiques, ce qui peut limiter sensiblement la flexibilité de notre approche.

- *Une complexité temporelle de la médiation de services* : Une autre limitation est liée au fait que notre approche d'intégration nécessite un chemin d'exécution plus long que celui qui est nécessaire dans l'intégration traditionnelle. En effet, l'intégration au niveau sémantique implique une transition par l'architecture d'ontologies au lieu d'exploiter directement l'architecture syntaxique des services. Cependant, nous estimons que quand bien même notre approche d'intégration est quelque peu plus lente, il n'en demeure pas moins qu'elle constitue une approche pertinente du fait qu'elle peut apporter plus de flexibilité car l'intégration se fait au niveau conceptuel.

X.3.4. Limites liées à l'implémentation du prototype

Les principales limites que nous pouvons formuler à l'encontre de l'implémentation de notre prototype sont :

- *La simplification de l'implémentation de certaines fonctionnalités du prototype* : Certaines fonctionnalités du prototype sont développées de façon relativement simple, voire parfois de façon non optimisée. Comme le but du prototypage est l'efficacité et non pas l'efficience, nous avons alors relégué en second rang toutes les préoccupations portant sur l'optimisation des performances.
- *La non prise en compte de mappings complexes* : Dans la version actuelle du prototype, seulement des mappings simples sont pris en charge. Par exemple, des mappings complexes nécessitant d'effectuer des conversions d'unité ne sont pas pris en charge. Ce qui limite le module de médiation dans sa version actuelle à prendre en charge uniquement des situations relativement simples.
- *Le développement relativement traditionnel du prototype* : Le prototype est développé selon une approche relativement traditionnelle en se basant notamment sur le langage Java. Nous croyons vu l'intérêt des approches basées sur les services sémantiques, et notamment de l'approche ODSOI, qu'une implémentation basée sur notre approche ou à la limite sur une approche orientée services serait plus pertinente.

X.3.5. Limites liées à l'expérimentation

Les principales limites que nous pouvons formuler concernant notre expérimentation portent principalement sur son périmètre qui est relativement restreint. Ce périmètre porte, rappelons-le, sur la gestion de la maintenance des équipements de fabrications de produits microélectroniques au sein de la société STMicroelectronics. Cette restriction a pour conséquence que les éléments de notre cadre méthodologique et/ou de notre prototype n'ont pas tous été testés sur l'étude de cas en raison du nombre encore limité de services disponibles. Ceci découle en grande partie du fait que le processus de construction de services nécessite d'importants efforts, notamment lors de la ré-ingénierie de certains applicatifs pour les exposer sous forme de services d'entreprise.

X.4. Principales conclusions

A l'issue de ce travail, nous croyons pouvoir affirmer que l'intégration sémantique d'applications d'entreprise constitue un problème dur et complexe qui nécessite à notre sens des services d'intégration basés sur la publication, la découverte et la médiation sémantique. Cette intégration doit être, à notre sens, basée sur des ontologies et elle est mieux résolue dans un contexte d'architecture orientée services. Les travaux réalisés dans le cadre de cette thèse nous ont apporté beaucoup de leçons et d'enseignements qui sont principalement les suivantes :

- Les architectures orientées services constituent un paradigme pertinent pour le développement et l'intégration des applications industrielles. L'existence des standards industriels (SOAP, WSDL, UDDI, etc.) et l'existence de certains outils de gestion de services de plus en plus performants (notamment les ESB) laissent augurer de quoi seront constitués les systèmes d'information industriels à venir. De plus, nous devons noter que le modèle de base des architectures de services doit être adapté au contexte spécifique de chaque entreprise. L'extension de l'architecture de base des services et la démarche de construction telles que nous les avons présentées dans cette thèse peuvent être appliquées à notre sens à toute entreprise et en particulier aux grandes entreprises industrielles.
- Les ontologies constituent encore un domaine relativement complexe qui a du mal à s'insérer ou qui s'insère de façon relativement timide au sein des entreprises industrielles. Ceci peut s'expliquer par le fait que ces notions ne sont pas encore suffisamment vulgarisées et qu'elles sont en cours de maturation. Mais ceci n'enlève rien à leur pertinence et leur nécessité dans le cadre des projets d'intégration. Nous demeurons convaincus que le degré de leur maturation est acceptable pour des architectures raisonnables et que l'optimisation et l'amélioration des performances des outils sémantiques (permettant de gérer les ontologies) ne cessent de s'effectuer. De même que pour l'architecture de services, nous devons noter que l'architecture sémantique (ou d'ontologies) ainsi que la démarche de construction de cette architecture telles que nous les avons présentées dans cette thèse peuvent être appliquées dans le cadre de projet d'intégration de toute entreprise et en particulier des grandes entreprises industrielles.
- L'intégration sémantique des applications industrielles constitue à notre sens l'aboutissement de tout projet d'intégration industriel. Ce dernier doit reposer sur une architecture flexible basée sur une solution semi centralisée qui exploite en particulier des services de publication, de découverte et de médiation. En outre, il est à notre sens important de faire cohabiter à la fois des services syntaxiques et des services sémantiques. Cette cohabitation découle de la nature même des projets d'intégration qui sont par essence des projets qui s'inscrivent dans la durée.
- La dernière leçon que nous avons retenue porte sur la nécessité de guider l'utilisateur tout au long de la démarche d'intégration par des outils appropriés. Ces outils doivent masquer toute la complexité inhérente aux différentes technologies manipulées, que ce soient les technologies liées à l'utilisation et à la gestion des services d'entreprise ou celles liées à l'utilisation et à la gestion des ontologies d'entreprise. Le prototype développé, combien même limité, s'inscrit dans ce sens.

X.5. Perspectives et travail futur

Conscients des limites que présentent nos travaux et soucieux de la continuité des travaux qui peut être mise en œuvre, nous avons envisagé les perspectives principales suivantes :

Amélioration de la technique d'urbanisation des services : Améliorer la technique d'urbanisation des services en expérimentant d'autres techniques d'identification des degrés de similarité, et en se basant en particulier sur des approches plus évoluées comme LSI, les approches floues ou encore les approches orientées ontologies. Une autre perspective importante consiste à outiller le processus d'interprétation du dendrogramme pour la détermination des clusters.

Extraction semi-automatique des ontologies fondamentales : Il serait intéressant et utile de définir des mécanismes d'extraction semi-automatiques de la sémantique à partir des composants du système d'information existant.

Maintenance des services et des ontologies d'entreprise : Ce point est important du fait qu'il permet de mettre en œuvre des mécanismes qui vont gérer la cohérence de l'architecture de services et de l'architecture des ontologies lors du processus de maintenance.

Amélioration du modèle sémantique : Cette amélioration porte sur certains besoins et souhaits que les utilisateurs ont formulé a posteriori, une fois que cette thèse est terminée. Il s'agit notamment de la possibilité de gérer de façon plus efficace la duplication des données au sein de l'entreprise. Par exemple, les utilisateurs désirent savoir quel est le service de référence qui produit une donnée particulière. Ceci permettrait alors de mieux connaître les véritables sources des informations qui transitent au sein de l'entreprise.

Amélioration du service de découverte sémantique : Cette amélioration doit permettre au service de découverte de prendre en charge des requêtes plus complexes incluant des contraintes comportant des valeurs numériques pour mieux cibler la recherche de certains services.

Amélioration du service de découverte sémantique : Cette amélioration doit permettre au service de découverte de prendre en charge des requêtes plus complexes incluant des contraintes comportant des valeurs numériques pour mieux cibler la recherche de certains services. Par exemple, il serait possible de formuler une contrainte de type: fiabilité du service > 0.9.

Optimisation des performances du prototype et passage à l'échelle : Des optimisations peuvent être apportées à notre prototype, notamment en ce qui concerne l'amélioration de certains modules comme le module de gestion des ontologies (afin de mieux assister l'utilisateur dans la création et la gestion de ses ontologies) et le module de médiation sémantique (en ce qui concerne le choix de stockage des règles de mapping et du moteur de raisonnement). Ceci permet d'améliorer l'efficacité de l'outil. Un autre point important est l'intégration de ce prototype au sein d'une infrastructure du marché (idéalement un ESB) pour pouvoir faire le passage à l'échelle permettant ainsi d'effectuer des tests en grandeur réelle.

Hiérarchisation de l'approche en niveaux de complexité : Afin d'apporter plus de flexibilité et d'adaptation lors de la mise en œuvre de l'approche, il devient nécessaire de proposer plusieurs niveaux ou couches de complexité. Plus précisément, nous envisageons de nous inspirer du principe de structuration en couches du langage OWL pour définir les trois niveaux de complexité de notre approche qui sont: ODSOI Lite,

ODSOI Medium, ODSOI Full. Chaque approche de niveau supérieur doit inclure l'approche de niveau inférieur. Ces différentes approches devraient ainsi couvrir la problématique d'intégration d'un large spectre d'entreprises allant des petites et moyennes entreprises (en appliquant l'approche ODSOI Lite) jusqu'aux très grandes entreprises (en appliquant l'approche ODSOI Full).

Extension de l'approche à l'intégration par le processus et à l'intégration extra-entreprise : Un dernier point important porte sur l'extension de ces travaux à l'intégration par le processus et à l'intégration extra-entreprise (B2B - Business to Business). L'intégration par le processus a été déjà entamée dans le cadre de ces travaux à travers notamment la définition des services processus et la construction des ontologies métier. En revanche, l'intégration extra-entreprise qui est aussi importante a été totalement ignorée tout au long de cette thèse.

Il est clair qu'atteindre tous ces objectifs laissés en perspectives nécessitera un travail de longue haleine et dont l'aboutissement prendra sans doute plusieurs années. Cependant, nous demeurons convaincus, compte tenu de son intérêt, que ce travail doit être soutenu et poursuivi.

BIBLIOGRAPHIE

BIBLIOGRAPHIE

[Abiteboul et al. 2000] Abiteboul S., Buneman P., and Suciu D., "Data on the Web - from relations semistructured data and XML". *Morgan Kaufann*, San fransisco, CA, 2000.

[Accary-Barrier et Calabretto 2002] Accary-Barrier T. et Calabretto S., "XML : Syntaxe". *Techniques de l'Ingénieur*, H3500 (1-15), 2002.

[Acharya 2003] Acharya R., "EAI: A business perspective". *eAI Journal*, [On Line] www.bijonline.com/PDF/Apr03Acharya.pdf, 2003.

[Akehurst et Kent 2002] Akehurst D. h., and Kent S., "A relational approach to defining transformations in a meta-model". In Jean-Marc Jézéquel, Heinrich Hussmann, and Stephen Cook, editors, *UML Model Engineering*, 2002.

[Alexaki et al. 2001] Alexaki S., Christophides V., Karvounarakis V., Plexousakis D., and Tolle K., "The ICSFORTH RDFSuite: Managing Voluminous RDF Description Bases". 2nd International Workshop on the Semantic Web (SemWeb'01), in conjunction with *Tenth International World Wide Web Conference* (WWW10), pp. 1-13, Hong Kong, 2001.

[Alonso 2002] Alonso G., "Web services: Concepts, Architectures and Applications". ETH Zürich, 2002.

[AMICE 1993] AMICE, "CIMOSA: Open System Architecture for CIM". Second extended and revised version, Springer-Verlag, Berlin, 1993.

[Amiour 1999] Amiour M., "Vers une fédération de composants interopérables pour les environnements centrés procédés logiiels". Thèse de Doctorat, Université Joseph Fourier, Grenoble I, France, 1999.

[Andrews et al. 2003] Andrews T., Curbera F., Dholakia h., Goland Y. Klein J., Leymann F., Liu K., Roller D., and Smith D., "Specification: Business Process Execution Language for Web Services Version 1.1". [On Line] http://www.ibm.com/developerworks/library/ws-bpel/, 2003.

[Apache 2005] Apache, "Apache Beehive". [On Line] http://beehive.apache.org/, 2005.

[Arabie et al. 1996] Arabie P., Hubert L.J., and De Soete G., "Clustering and Classification". *Wold Scientific*, Singapore, 1996.

[Arens et al. 1993] Arens Y., Chee C. Y., Hsu C. N., and Knoblock C. A., "Retrieving and integrating data from multiple information sources". In the *International Journal on Intelligenet and Cooperative Information Systems*, vol 2(2) pp: 127-158 [On Line] http://citeseer.ist.psu.edu/arens93retrieving.html, 1993.

[Athena 2005] Athena, "Athena Project deliverables". [OnLine] http://www.athena-ip.org, 2005.

[Bachimont 2000] Bachimont B., "Engagement sémantique et engagement ontologique : conception et réalisation d'ontologies en ingénierie de connaissances". In J. Charlet, M. Zachlad, G; kassel and D. Bourigault editors, Ingénierie des connaissances, évolutions récentes et nouveaux défis. *Eyrolles*, Paris, 2000.

[Bachimont 2001] Bachimont B., "Modélisation linguistiqye et modélisation logique des ontologies : l'apport de l'ontologie formelle". In 12th *Journées Francophones d'Ingénierie des Connaissances* (IC'01), pages 349-268, Grenoble, 2001.

[Banavar et al. 1999] Banavar G., Chandra T., Sturman D., Mukherjee B., Nagarajaro J., and Storm J., "An efficient multicast protocol for content-based Publish-Subscribe systems". Proceedings of the 19th *International Conference on Distributed Computing Systems*, Austin, TX, USA IEEE Computer Society: 262-272, [On Line] http://www.research.ibm.com/gryphon/, 1999.

[Baril 2003] Baril X., "Un modèle de vues pour l'intégration de sources de données XML : VIMIX". Thèse de Doctorat, Université des Sciences et Techniques du Languedoc, France, 2003.

[Benslimane et al. 1999] Benslimane D., Juanot F., Laurini R., Yétongnon K., Cullot N., et Savonnet M., "Interopérabilité de SIG: un état de l'art". Revue Internationale de Géomatique, *Hermès Science Publications*, Vol. 9 (No. 3), pp. 279-316, 1999.

[Bentallah et al. 2003] Benatallah B., Hacid M-S., Rey C., Toumani F., "Semantic Reasoning for Web Services Discovery", WWW Workshop on E-Services and the Semantic Web, Budapest, Hungary, 2003.

[Bergamaschi et al. 2001] Bergamaschi, S., Castano, S., Beneventano D. and Vincini M., "Semantic integration of heterogeneous information sources". *Special Issue on Intelligent Information Integration, Data & Knowledge Engineering*, Vol 36 (No. 1) pp. 215-249, 2001.

[Berlin et Motro 2002] Berlin J., and Motro A., "Database schema matching using machine learning with feature selection". In Proc. *Conference on Advanced Information System Engineering* (CAiSE), 2002.

[Berners-Lee et al. 1997] Berners-Lee T., Fielding R. T., Frystyk Nielsen H., Gettys J., and Mogul J., "Hypertext Transfer Protocol HTTP/1.1". RFC 2068, 1997.

[Berners-Lee et al. 2001] Berners-Lee T., James Hendler J., and Lassila O., "The Semantic Web". *Scientific Amercian*, 2001.

[Bernstein et Klein 2002] Bernstein A., Klein M., "Discovering Services: Towards High Precision Service Retrieval". In CaiSE workshop on Web Services, e-Business, and the Semantic Web: Foundations, Models, Architecture, Engineering and Applications. Toronto, Canada, 2002.

[Bézivin et al. 2003] Bézivin J., Farcet N., Jézéquel J-M., Langlois B., and Pollet D., "Reflective model driven engineering". In G. Booch P. Stevens, J. Whittle, editor, *Proceedings of UML 2003*, San Francisco,, 2003.

[Bézivin et Blanc 2002] Bézivin J., et Blanc X., "MDA: Vers un important changement de paradigme en génie logiciel". *Développeur Référence*, [On Line] http://www.devreference.net/, 2002.

[Bialecki 2000] Bialecki, "E-Commerce Integration Meta-Framework – overview". 2001.

[Blanc 2004] Blanc L., "Modélisation des processus métier mis en oeuvre dans une approche EAI en vue de leur pilotage: Le pilotage des applications intégrées". Thèse de Doctorat, Université de Savoie, France, 2004.

[Bobillo et al. 2005] Bobillo F., Gomez-Gomero J., and Perez-Perez R., "Towards Semantic Web Services: A brief Overview". In P. Isaías & M. B. Nunes (Editors.), *Proceedings of the IADIS international conference* (WWW/Internet), 2005.

[Boehm et Abts 1999] Boehm B., and Abts C., "COTS Integration: Plug and Play ?". *IEEE Computer*, Vol. 32 (No. 1), pp. 135-138, 1999.

[Booch 1992] Booch G., "Conception orientée objets et applications". *Editions Addison-Wesley*, 1992.

[Borst 1997] Borst W. N., "Construction of Engineering Ontologies for Knowledge Sharing and Reuse". PhD thesis, *Tweente University*, Enschede, NL- Centre for Telematica and Information Technology, 1997.

[Bowers et Delcambre 2000] Bowers S., Delcambre L., "Representing and Transforming Model-Based Information". In Proceedings of the Workshop on the Semantic Web at ECDL-00; Lisbon, Portugal, 2000.

[Boyer 1994] Boyer F., "Coordination entre outils dans un environnement intégré de développement de logiciels". Thèse de Doctorat, Université Joseph Fourier, Grenoble I, France, 1994.

[BPMI 2003] BPMI, "Business Process Modeling Language (BPML), Proposed Recommendation". *Business process Management Initiative* (BPMI). [On Line] http://www.bpmi.org, 2003.

[Brockschmidt 1995] Brockschmidt A., "Inside OLE". 2nd edition, *Microsoft Press*, 1995.

[Broekstra et al. 2001] Broekstra J., Kampman A., and van Harmelen F., "Sesame: a Generic Architecture for Storing and Querying RDF and RDF Schema". in the 1st *International Semantic Web Conference* (ISWC2002), Sardinia, Italy, 2002.

[Bruijn 2003] De Bruijn J., "Using Ontologies". TR, *Digital Enterprise Research Institute (*DERI), 2003.

[Bruijn 2003-b] De Bruijn J., "Semantic information Integration". Master Thesis, *Digital Enterprise Research Institute* (DERI), 2003.

[Bruijn et al. 2005] Bruijn J., Lausen H., Krummenacher R., Polleres A., Kifer M. and Fensel D., "Deliverable D16.1v0.2, The Web Service Modeling Language WSML". Final Draft, *WSMO project*, [On Line] http://www.wsmo.org/ TR/d16/d16.1/v0.2/20050320/, 2005.

[Bruijn et Polleres 2004] Bruijn J. and Polleres A., "Towards an Ontology Mapping Specification Language for the Semantic Web". DERI Technical report 2004-06-30, *Digital Enterprise Research Institute* (DERI), 2004.

[Bruno et al. 2005] Bruno M., Canfora G., Di Penta M., and Scognamiglio R., "An approach to support web ser-vice classification and annotation". In Proceedings of the IEEE *International Conference on e-Technology, e-Commerce and e-Services* (EEE 2005), Honk-Kong, 2005.

[Bussler 2003] Bussler C., "Semantic Web Services: Reflections on Web Service Mediation and Composition". Proceedings of the Fourth *International Conference on Web Information Systems Engineering* (WISE'03), 2003.

[Bussler 2003-b] Bussler C., "The Role of Semantic Web Technology in Enterprise Application Integration". In the *Bulletin of the IEEE Computer Society Technical Committee on Data Engineering*, Vol 26 (No. 4) pp. 62-68, 2003.

[Bussler 2003-c] Bussler C., "Semantic Web Services: Reflections on Web Service Mediation and Composition". In the Proceedings of the Fourth International Conference on Web Information Systems Engineering (WISE'03), 2003.

[Bussler 2004] Bussler C., "Semantic Web Services". ICWE'04, Munich, 2004.

[Bussler et al. 2002] Bussler C., Fensel D., and Maedche A., "A Conceptual Architecture for Semantic Web Enabled Web Services". *ACM-SIGMOD*, Special Section on Semantic Web and Data Management, [On Line] http://www.sigmod.org/record/ issues/0212/SPECIAL/4.Bussler1.pdf, 2002.

[Bussler et al. 2004] Bussler C. et al., WSMO. The Eleventh International Conference on Artificial Intelligence: Methodology, Systems, 2004.

[C4ISR 1998] C4ISR Interoperability Working Group, "Levels of Information Systems Interoperability (LISI)". Department of Defense, Washington, DC, 1998.

[Cabral et al. 2004] Cabral L., Domingue J., Motta E., Payne T., and Hakimpour F., "Approaches to Semantic Web Services: An Overview and Comparisons". *European Semantic Web Symposium* (ESWS'04), 2004.

[Calvanese et al. 2001] Calvanese D., De Giacomo G., and Lenzerini M., "A framework for ontology integration". Proceedings of the 1st *Internationally Semantic Web Working Symposium* (SWWS) 303-317, 2001.

[Candillier 2004] Candillier L., "La classification non supervisée". [On Line] http://www.grappa.univ-lille3.fr/~candillier/rapports/prstClustering.ps, 2004.

[Cangemi et al. 2003] Cangemi A., Guarino N., Masolo C., and Oltramari A., "Sweetening Wordnet with DOLCE". *AI magazine*, Vol. 24 (No. 3) pp.13-24, 2003.

[Cardoso 2006] Cardoso J., "Approaches to Developing Semantic Web Services". *International Journal of Computer Science*, Vol. 1 (No. 1), ISSN 1306-4428, 2006.

[Cardoso et al. 2002] Cardoso J., Bussler C., Sheth A., and Fensel D., "Semantic Web Services and Processes: semantic Composition and Quality of Service". Tutorial at *International Federated Conferences*: DOA/ODBASE/CooPIS, 2002.

[Cardoso et Sheth 2004] Cardoso J., and Sheth A., "Semantic Web services and web processes composition". *Springer-Verlag*, New York Inc, 2004.

[Cardoso et Sheth 2005] Cardoso J., Sheth A., "Introduction to Semantic Web Services and Web Process Composition". In Semantic Web Process: powering next generation of processes with Semantics and Web Services, Lecture Notes in Computer Science, Springer, 2005.

[Casati et al. 2000] Casati F., Ilnicki S., and Jin L., "Adaptive and dynamic service composition in EFlow". In Proceedings of 12th *International Conference on Advanced Information Systems Engineering* (CAiSE), Stockholm, Sweden, Springer Verlag, 2000.

[Casati et al. 2001] Casati F., Sayal M., and Shan M. C., "Developing e-services for composing e-services". In Proceedings of 13th *International Conference on Advanced Information Systems Engineering* (CAiSE), Interlaken, Switzerland, Springer Verlag, 2001.

[CEN 2006] CEN/ISO DIS 19440, "Constructs for Enterprise Modelling", 2006.

[Chakraborty et al. 2001] Chakraborty D., Perich F., Avancha S., Joshi A., "DReggie: Semantic Service Discovery for M-Commerce Applications". In Workshop on

Reliable and Secure Applications in Mobile Environment, 20th Symposium on Reliable Distributed Systems, pp. 28–31, 2001.

[Chalupsky 2003] Chalupsky H., "OntoMorph: A Translation System for Symbolic Knowledge". In Proceedings of the 9th International Conference on Principles of Knowledge Representation and Reasoning (KR-04), *AAAI Press*, Henrik Nottelmann and Umberto Straccia, 2003.

[Chappell 1996] Chappell D., "Understanding ActiveX and OLE". *Microsoft Press*, Redmond, 1996.

[Chappell 2005] Chappell D., "Enterprise Service Bus". *O'Reilly media* Inc., 2004.

[Chapron 2006] Chapron J., "Aide à la décision pour la gestion des processus informationnel de l'entreprise dans un environnement de réactivité". Thèse de Doctorat, *Ecole des Mines de Saint-Etienne*, Laboratoire G2I, Equipe OMSI, 2006.

[Chauvet 2002] Chauvet J.-M., "De XML aux services Web pour les entreprises". *Techniques de l'ingénieur*, 2002.

[Chauvet 2002-b] Chauvet J.-M., "Services Web avec SOAP, WSDL, UDDI, ebXML". *Eyrolles*, 2002.

[Chawathe et al. 1994] Chawathe S., Garcia-Molina H., Hammer J., Ireland K., Papakonstantinou Y., Ullman J., and Widom J., "The Tsimmis project: Integration of heterogeneous information sources". In *Conference of the Information Processing Society*, Japan, pp. 7-18, 1994.

[Chelli 2003] Chelli H., "Urbaniser l'entreprise et son système d'Information". Guide des entreprises agiles, *Vuibert*, 2003.

[Chen et Doummeingt 2003] Chen D., and Doumeingts G., "European initiatives to develop interoperability of enterprise applications: basic concepts, framework and roadmap". *Annual Reviews in Control*, Vol. 27, pp. 153–162, 2003.

[Chevassus 2005] Chevassus M., "Enterprise Application integration : EAI". *Techniques de l'ingénieur*, H2915, pp.1-20, 2005.

[Chiaramonti 2005] Chiaramonti C., "Echnage de données informatisé (EDI)". Techniques de l'Ingénieur (H7405), 2005.

[Chrisment et al. 2004] Chrisment C., Pujolle G., Ravat F., Teste O., and Zurfluh G., "Entrepôts de données". *Techniques de l'ingénieur*, H3870, 2004.

[Clark et Jones 1998] Clark T., Jones R., "Organisational Interoperability Maturity Model for C2". Defence Science and Technology Organisation C3 Research Centre, Fern Hill Park, 1998.

[Cloux 2003] Cloux P. Y., Doussot D., and Géron A., "Technologies et Architectures Internet: Corba, COM, XML, J2EE, .NET, Web Services". *Dunod*, 2003.

[Club-Urba 2005] Club Urba SI, "Pratiques de l'Urbanisation des Systèmes d'Informatique en entreprises". *Urba SI Club*, 2003.

[Cluet et al. 1998] Cluet S., Delobel C., Siméon J., Smaga K., "Your mediators need data conversion!". In Proceedings of the SIGMOD International Conference on Management of Data, pages 177–188. ACM Press, 1998.

[Corcho 2004] Corcho O., "A declarative approach to ontology translation with knowledge preservation". PhD thesis, *Universidad Politécnica de Madrid*, 2004.

[Corcho et al. 2003] Corcho O., Fernandez-Lopez M., and Gomez-Pérez A., "Methodologies, tools and languages for building ontologies : Where is their

meeting point ?". *Data & Knowledge Engineering* Vol. 46 (No. 1) pp. 41-64, Elsevier, 2003.

[Corcho et al. 2003-b] Corcho O., Gomez-Perez A., Leger A., Rey C., and Toumani F., "An ontology-based mediation architecture for e-commerce applications". In Proceedings of *Intelligent Information Systems* (IIS2003), [On Line] http://citeseer.ist.psu.edu/corcho03ontologybased.html, 2003.

[Corella et Castells 2006] Corella M. A., and Castells P., "A Heuristic Approach to Semantic Web Services Classification". 10th *International Conference on Knowledge-Based & Intelligent Information & Engineering Systems* (KES 2006), Invited Session on Engineered Applications of Semantic Web (SWEA). Bournemouth, UK, Springer Verlag Lecture Notes in Artificial Intelligence, 2006.

[Cousot et Cousot 2002] Cousot P. and Cousot R., "Systematic Design of Program Transformation Frameworks by Abstract Interpretation". POPL '02, Portland, USA, ACM SIGPLAN Notices Vol. 31 (No 1), pp. 178-190 [On Line] http://citeseer.ist.psu.edu/cousot02systematic.html, 2002.

[Crubézy et Musen 2003] Crubézy M., and Musen M. A., "Ontologies in Support of Problem Solving". In Staab S. and Studer R., editors, *Handbook on ontologies*, pp. 321-342 Springer, 2003.

[Dang Ngoc 2003] Dang Ngoc T. T., "Fédération de données semi-structurées avec XML". Université de Versailes, France, 2003.

[Davidson et Kosky 1997] Davidson S. B., Kosky A., "WOL: A language for database transformations and constraints". In Proceedings of the 13th International Conference on Data Engineering (ICDE), pages 55–65. IEEE Computer Society, 1997.

[Dayal et al. 2001] Dayal U., Hsu M., and Ladin R., "Business Process Coordination: State of the Art, Trends, and Open Issues". In Proceedings of the 27th *International Conference on Very Large DataBases*, 2001.

[Denis 2003] Denis A., "Contribution à la conception d'une plate-forme haute performance d'intégration d'exécutifs communicants pour la programmation des grilles de calcul". Thèse de doctorat, Université de Rennes 1, IRISA, Rennes, France, 2003.

[Derwester et al. 1990] Derwester S., Dumais S.T., Furnas G.W., Landauer T.K. and Harshmann R., "Indexing by Latent Semantic Analysis". *Journal of the American Society for Informartion Science*, pp. 391-407, 1990.

[Devogele 1997] Devogele T., "Processus d'intégration et d'appariement de bases de données géographiques. Application à une base de données routières multi-échelles". Thèse de Doctorat de l'Université de Versailles, 1997.

[Dieng et al. 2001] Dieng R., Corby O., Gandon F., Giboin A., Golebiowska J., Matta N., et Ribière M., "Méthodes et outils pour la gestion des connaissances : une approche pluridisciplinaire du knowledge management". *Dunod Edition Informatiques*, Séries Systèmes d'Information, 2001.

[Ding 2002] Ding Y., "Ontology Research and Development - Part 1: A Review of Ontology Generation". *Journal of Information Science*, Vol. 28, (No. 2), pp. 123-136, 2002.

[Ding 2002-b] Ding Y., "Ontology Research and Development - Part 2: a review of ontology mapping and evolving". *Journal of Information Science*, Vol. 28 (No. 5), pp. 383-396, 2002.

[Dip 2005] DIP Project, "DIP Deliverables". [On Line] http://dip.semanticweb.org, 2005.

[Doan et al. 2002] Doan A., Madhavan J., Domingos P., and Halevy A., "Learning to Map Between Ontologies on the Semantic Web". In The Eleventh International WWW Conference (WWW'02), Hawaii, US, [On Line] http://citeseer.ist.psu.edu /doan02learning.html, 2002.

[Dogac 2004] Dogac A., "Semantic Web Services". Conference on Web Engineering (ICWE'04), Munich, 2004.

[Dong et al. 2004] Dong X., Halevy A., Madhavan J., Nemes E., and Zhang J., "Similarity Search for Web Services". Proceedings of the 30th *VLDB Conference*, Toronto, Canada, [On Line] http://www.vldb.org/conf/2004/RS10P1.PDF, 2004.

[Dou et al. 2003] Dou D., McDermott D., and Qi P., "Ontology translation on the semantic web". In *International Conference on Ontologies, Databases and Applications of Semantics* (ODBASE'03), Catania, Italy, 2003.

[Downing 1998] Downing T. B., "RMI : Developing Distributed Java Applications with Remote Method Invocation and Object Serialization". *IDG Books*, San Mateo, CA, USA, 1998.

[Duineveld et al. 1999] Duineveld J. A., Stoter R., Weiden R. M., Kenepa B., and Benjamins R. V., "WonderTools ? A comparative study of ontological engineering tools". *International Journal of HumanComputer Studies*, Vol. 52(No. 6), pp.1111-1133, 1999.

[ebXML 2001] ebXML Requirements Team, "ebXML requirements specification v1.06". Technical report, *ebXML*, 2001.

[ECIMF 2001] ECIMF, "E-Commerce Integration Meta-Framework – General Methodology (ECIMF-GM)", CEN/ISSS/WS-EC/ECIMF Draft, version 0.3, 2001.

[ECIMF 2005] ECIMF, "E-Commerce Integration Meta-Framework". [On Line] http://www.ecimf.org/, 2005.

[EIF 2004] European Commission, "EIF: European Interoperability Framework". White Paper, Brussels, [OnLine] http://www.comptia.org, 2004.

[Erl 2004] Erl T., "Service-Oriented Architecture - A Field Guide to Integrating XML and Web Services". *Printice Hall*, 2004.

[Ermes 1994] ERMES- Groupe ESCP, "Systèmes d'Information: La perspective du management". *Masson*, 1994.

[Espinasse 2002] Espinasse B., "Présentation générale du langage de modélisation objet UML". Cours DEA - *G2I, Université Aix-Marseille 3*, 2002.

[EU-NSF 2002] EU-UNSF, "Semantic Web: Workshop Report and Recommandations". Sophia Antipolis, 2002.

[Euzenat 2001] Euzenat J., "Towards a principled approach to semantic interoperability". Proceedings of the IJCAI-01, *Workshop on Ontologies and Information Sharing*, 2001.

[Euzenat et Valtchev 2004] Euzenat J., and Valtchev P., "Similarity-based ontology alignment in OWL-Lite". In the 16th *European Conference on Artificial Intelligence* (ECAI'04), 2004.

[Evans 2003] Evans C., "Web Services Reliability (WS-Reliability) Ver1.0". 2003.

[Fensel 2001] Fensel D.," Ontologies: A Silver Bullet for Knowledge Management and Electronic Commerce". *Springer*, 2001.

[Fensel et al. 2003] Fensel D., Motta, E. Benjamins V., Decker S., Gaspari M., Groenboom R., Grosso W., and Musen M., "The unified problem-solving method development language UPML". Knowledge and Information Systems (KAIS): An international journal, Vol. 5 (No.1), 2003.

[Fensel et Bussler 2002] Fensel D., and Bussler C., "The Web Service Modeling Framework WSMF". In Electronic Commerce Research and Applications, 1(2), *Elsevier, Scinces B. V.*, 2002.

[Fernandez et al. 1997] Fernandez M., Gomez-Perez A., and Juristo N., "METHONTOLOGY". Proceedings of the *AAAI97*, Spring Symposium Series on Ontological Engineering, Stanford, USA, pp. 33-40, 1997.

[Fürst 2002] F. Fürst, "Ingénierie ontologique". Rapport de Recherche, *Institut de Recherche en Informatique de Nantes*, 2002.

[Gandon 2002] Gandon F., "Distributed Artificial Intelligence and Knowledge Management: ontologies and multi-agent systems for a corporate semantic web". PhD Thesis, *INRIA and University of Nice - Sophia Antipolis*, 2002.

[Gangemi et al. 2003] Cangemi A., Guarino N., Masolo C., and Oltramari A., "Sweetening wordnet with DOLCE". *AI magazine*, Vol. 24 (No. 3), pp.13-24, 2003.

[Geib 2000] Geib J. M., et Merle P., "CORBA : des concepts à la pratique". *Techniques de l'ingénieur*, 2000.

[Ghurbhurn 2006] Ghurbhurn R., Beaune P., et Solignac H., "Accès à des données hétérogènes par des processus métiers intégrés". *Revue des sciences et technologies de l'information*, Thème: Ingénierie des Systèmes d'Information, Vol. 11 (No. 3) pp. 1633-1311, 2006.

[Giunchiglia et Walsh 1992] Giunchiglia F., and Walsh T., "A Theory of Abstraction". Artificial Intelligence, Vol. 57 (No. 2-3) pp. 323–390, 1992.

[Go 2004] Go Albert France, "AMI: Fondations techniques". 2004.

[Goh 1997] Goh C. H., "Representing and Reasoning about Semantic Conflicts in Heterogeneous Information Sources". Phd Thesis, *MIT*, 1997.

[Gomez-Perez 2000] Gomez-Perez A., "Développements récents en matière de conception, de maintenance et d'utilisation d'ontologies". *3èmes rencontres Terminologie et intelligence artificielle TIA*, Vol 19, pp. 9-20, 2000.

[Gomez-Perez 2001] Gomez-Perez, A, "Evaluation of Ontologies". *International Journal of Intelligent Systems*, Vol. 16 (No. 3), 2001.

[Gomez-Perez 2003] Gomez-Perez A., "Ontology Engineering". *Springer Verlag*, 2003.

[Gonzàlez-Castillo et al. 2001] Gonzàlez-Castillo J., Trastour D., Bartolini C., "Description Logics for Matchmaking of Services". In KI-2001 Workshop on Applications of Description Logics Vienna, Austria, [On-Line] http://sunsite.informatik.rwth-aachen.de/Publications/CEUR-WS/Vol-44, 2001.

[Gruber 1993] Gruber T., "A Translation Approach to Portable Ontology Specifications". *Knwledge Acquisition*, Vol. 5 (No. 2) pp. 199-220, 1993.

[Gruber 1995] Gruber R. T., "Towards Principles for the Design of Ontologies Used for Knowledge Sharing". *International Journal of Human Computer Studies*, Vol. 43, (No 5/6), pp. 907-928, 1995.

[Gruber et Olsen 1994] Gruber, T. and Olsen, R., "An Ontology for Engineering Mathematics". In Jon Doyle, Piero Torasso, & Erik Sandewall, Eds., *Fourth International Conference on Principles*, 1994.

[Gruninger 2003] Gruninger M., "A guide to the Ontology of the Process Specification Language". In S. Staab and R. Studer, editors, *Handbook on ontologies*, *Springer*, 2003.

[Guarino 1995] Guarino N., "Formal Ontology, Conceptual Analysis and Knowledge Representation". *International journal of Human and Computer Studies*, Vol. 43 (No. 5/6), pp. 625-640,1995.

[Guarino 1997] Guarino N., "Undestanding, building, and using ontologies". *International journal of Human and Computer Studies*, Vol. 45 (No. 2/3), pp. 293-310, 1997.

[Guarino 1998] Guarino N., "Formal Ontology in Information Systems". Proceedings of *FOIS'98*, Trento, Italy, pp. 3-15, 1998.

[Guarino 1999] Guarino N., "Some Ontological Principles for Designing Upper Level Lexical Resources". Proceedings of *First International Conference on Language Resources and Evaluation*, ELRA - European Language Resources Association, Granada, Spain, pp. 527-534, 1999.

[Haarslev et Moller 2003] Haarslev V., and Moller R., "Racer: A Core Inference Engine for the Semantic Web". [On Line] http://www.cs. concordia.ca/faculty/ haarslev/racer, 2003.

[Haarslev et Moller 2003] Haarslev V., Moller R., RACER User's Guide and Reference Manual Version 1.7.19, 2003.

[Haller et al. 2005] Haller A., Gomez J. M., and Bussler C., "Exposing Semantic Web service principles in SOA to solve EAI scenarios". Web Service Semantics: Towards Dynamic Business Integration Workshop, in conjunction with The 14th *International World Wide Web Conference* (WWW2005), Chiba, Japan, 2005.

[Hameed et al. 2004] Hameed A., Preece A., and Sleeman D., "Ontology reconciliation". In Steffen Staab and Rudi Studer, editors, *Handbook of ontologies*, 2004.

[Hasselbring 2000] Hasselbring W., "Information Systeme Integration". *Communications of the ACM*, Vol. 43 (No. 6), pp. 33-38, 2000.

[Hellerstein et al. 1999] Hellerstein J., Stonebraker M., and Caccia R., "Independent, open enterprise data integration". *IEEE Data Engineering Bulletin*, Vol. 22 (No. 1) pp. 43-49, 1999.

[Herzum 1999] Herzum S., "Business Component Factory: A Comprehensive Overview of Component-Based Development for the Enterprise". Editor: John Wiley & Sons, Published: December 1999.

[Hoffman 2003] Hofmann M. A., "Essential preconditions for coupling model-based information systems". NATO M&S Group (NMSG) Conference on C3I and M&S Interoperability, RTO-MP-123, Antalya, Turkey, 2003.

[Hohpe et Woolf 2003] Hohpe G., and Woolf B., "Enterprise Integration Patterns". *Adison Wesley*, 2003.

[Hohpe et Woolf 2004] Hohpe G., Woolf B., "Enterprise Integration Patterns: Designing, Building and Deploying Messaging Solutions". Peason Education, 2004.

[Huang et al. 2004] Huang K., Zhou Z., Han Y., Li G., and Wang J., "An Algorithm for Calculating Process Similarity to Cluster Open-Source Process Designs". *GCC Workshops*, pp. 107-114, 2004.

[Huet 2002] Huet F., "Objets mobiles : conception d'un middleware et évaluation de la communication". Thèse de doctorat, Université de Nice – Sophia Antipolis, Projet OASIS, 2002.

[IAC 2003] Industry Advisory Council (IAC), "Interoperability Strategy: Concepts, Challenges and Recommendations". Concept Level White Paper, developped for the Federal Enterprise Architecture, 2003.

[IBM 2005] IBM., "Web Services Coordination". [On Line] http://www-128.ibm.com/developerworks/library/specification/ws-tx/, 2005.

[IBM et al. 2005] IBM, BEA Systems, Microsoft, SAP AG, Siebel Systems, "BPEL4WS - Business Process Execution Language for Web Services version 1.1". [On Line] http://www-128.ibm.com/developerworks/library/specification /ws-bpel/, 2005.

[Ideas 2003] IDEAS, "IDEAS Project Deliverables (WP1-WP7)". Public reports, [On-line] www.ideas-roadmap.net, 2003.

[IEC 2002] IEC - International Electrotechnical Commission, "IEC-65-290-DC - TC65: Industrial Process Measurement and Control". *IEC*, 2002.

[Interop 2005] INTEROP, "Deliverables of INTEROP Project". [On Line] www.interop-ne.org, 2005.

[ISO 14258] ISO, "ISO-14258, Concepts and Rules for Enterprise Models". 1998.

[ISO-OSI] ISO/IEC ISP 12061, "Technologies de l'information. Interconnexion de systèmes ouverts (OSI)".

[Izza et al. 2004] Izza S., Vincent L., Burlat P., Solignac H., et Lebrun P., "Intégration d'applications: Etat de l'art et Perspectives". *Journées de l'Informatique de l'Entreprise* (JIE 2' 04), Algérie, pp. 242-253, 2004.

[Izza et al. 2005-a] Izza S., Vincent L., and Burlat P., "A Framework for Semantic Enterprise Integration". In Proceedings of *INTEROP-ESA'05*, Geneva, Switzerland, pp-78-89, 2005.

[Izza et al. 2005-b] Izza S., Vincent L., and Burlat P., "A Unified Framework for Application Integration : an Ontology-driven Service-oriented Approach". Chin-Sheng Chen, Joaquim Filipe, Isabel Seruca, José Cordeiro (Eds.): ICEIS 2005, *Proceedings of the Seventh International Conference on Enterprise Information Systems*, Miami, USA, pp. 165-170, 2005.

[Izza et al. 2005-c] Izza S., Vincent L., and Burlat P., "Exploiting the Combination of Web Services and Semantic Web in Achieving Semantic Application Integration". *6ème Congrès de Génie Industriel* (GI 2005), Besançon, France, pp-175-185, 2005.

[Izza et al. 2005-d] Izza S., Vincent L., and Burlat P., "Ontology Urbanization for Semantic Integration: Dealing with Semantics within Large Enterprises of Heterogeneous and Autonomous Application Systems". The 9th IEEE *International EDOC Conference*, Enschede, The Netherlands, pp. 83-94, 2005.

[Izza et al. 2006-a] Izza S., Vincent L., Burlat P., Lebrun P., and Solignac H., "Extending OWL-S to Solve Enterprise Application Integration Issues". *Interoperability for Enterprise Software and Applications Conference* (I-ESA'06), Bordeaux, France, 2006.

[Izza et al. 2006-b] Izza S., Vincent L., Burlat P., Lebrun P., and Solignac H., "An Ontology-Driven Data Mediation Framework for Enterprise Application integration". 12th *IFAC Symposium on Information Control Problems in Manufacturing* (INCOM'06), Saint-Etienne, France, 2006.

[Izza et al. 2006-c] Izza S., Vincent L., and Burlat P., "Dealing with Semantic Application Integration Withn Large and Dynamic Enterprises". *International Journal of Cooperative Information systems*, Vol. 15, No. 4, pp. 507-533, [OnLine] http://www.worldscinet.com/ijcis/15/1504/S02188430061504.html, 2006.

[Izza et al. 2006-d] Izza S., Vincent L., and Burlat P., "A Framework for Semantic Enterprise Integration". In *Konstantas D., Bourrières J. P., Léonard M., Boudjlida N., Editors, Interoperability of Enterprise Software and Applications.* Springer-Verlag, London, 2006.

[Jacobson 1993] Jacobson I., "Le génie logiciel orienté objet". *Addison Wesley*, 1993.

[jduNet 2006] 01Net, "Intégration, nouvelle génération". *01Net*, 2006.

[Johannesson et Perjons 2001] Johannesson P., and Perjons E., "Design principles for process modelling in enterprise application integration". Information System Vol. 26, pp. 165-184, 2001.

[Kadima et Monfort 2003] Kadima H., et Monfort V., "Les Web Services". *Dunod*, 2003.

[Kalfoglou et Schorlemmer 2003] Kalfoglou Y., and Schorlemmer M., "IF-Map: an ontology mapping method based on information flow". *Journal on Data semantics*, Vol. 1 (No. 1) pp. 98-127, 2003.

[Kalfoglou et Schorlemmer 2003-b] Kalfoglou Y., and Schorlemmer M., "Ontology Mapping: The State of The Art". *The Knowledge Engineering Review*, Vol. 18 (No. 1), pp. 1-31, Cambridge University Press, 2003.

[Kassel 2002] Kassel G., "OntoSpec : une méthode de spécification semi-informelle d'ontologies". in Actes des *journées francophones d'Ingénierie des Connaissances* (IC'2002), pp. 75-87, 2002.

[Kavouras 2003] Kavouras M., "A unified ontological framwork for semantic integration". *International Workshop on Next Generation Geospatial Information*, Cambridge, UK, 2003.

[Kellert et Toumani 2003] Kellert P., and Toumani F., "Les web services sémantiques". Research Report LIMOS/RR 03-1511, 2003.

[Klein 2001] Klein M., "Combining ans relating ontologies: an analysis of problems and solutions". International Joint Conferences on Artificial Intelligence (IJCAI), Workshop on Ontologies and Information Sharing, Seattle, USA, 2001.

[KnowledgeWeb 2004] KnowledgeWeb, "Deliverables of KWEB Project". EU-IST-2004-507482, [On Line] http://knowledgeweb.semanticweb.org/, 2004.

[Kontogiannis et al. 2002] Kontogiannis K., Smith D., and O'Brien L., "On the Role of Services in Enterprise Application Integration". In the Proceedings of the *10th International Workshop on Software Technology and Engineering*, 2002.

[Kosanke 2005] Kosanke K., "ISO Standards for Interoperability: a comparison". In Proceedings of *INTEROP-ESA'05*, Geneva, Switzerland, 2005.

[Kraft 2004] KRAFT, "Knowledge Re-use And Functional Transformation". [On Line] http://www.cs.cf.ac.uk/kraft/, 2004.

[Krishnamurthy et al. 1991] Krishnamurthy R., Litwin W., Kent W., "Language features for interoperability of databases with schematic discrepancies". In Proceedings of the SIGMOD International Conference on Management of Data, pages 40–49. ACM Press, 1991.

[Lakshmanan 2003]Laks F. S., Lakshmanan V.S., "Interoperability on XML data". In Proceedings of the International Semantic Web Conference (ISWC), volume 2870, pages 146–163. Lecture Notes in Computer Science, 2003.

[Lammermann 2003] Lammermann S., "Runtime Service Composition via Logic-Based Program Synthesis". PhD thesis, Department of Microelectronics and Information Technology, *Royal Institute of Technology*, 2002.

[Lapassat 2003] Lapassat G., "Urbanisme Informatique et Architetctures Applicatives". Hermes Sciences, *Lavoisier*, 2003.

[Lee et al. 1994] Lee J., Yost G. and the PIF Working Group, "The PIF Process Interchange Format and Framework". [On Line] http://ccs.mit.edu/pifmain.html, 1994.

[Lemoigne 1984] Lemoigne J. L., "La théorie du système général". *Editions Presse Universitaire Française*, 1984.

[Leroux 2004] Leroux B., "Urbanisation et Modernisation du Système d'Information". *Lavoisier*, 2004.

[Leroux 2005] Leroux B., "Interopérabilité du SDET". [On Line] www.educnet. education.fr/chrgt/interoperabiliteV1.pdf, 2005.

[Li et Clifton 1994] Li W. S., and Clifton C., "Semantic integration in heterogeneous databases using neural networks". Jorge B. Bocca, Matthias Jarke, Carlo Zaniolo (Eds.): In Proceedings of the *20th International Conference on Very Large Data Bases* (VLDB'94), Santiago de Chile, Chile. Morgan Kaufmann, pp. 1-12, 1994.

[Li et Horrocks 2004] Li L., and Horrocks I., "A software framework for matchmaking based on semantic web technology". In the *International Journal of Electronic Commerce*, Vol 8 (No. 4), pp. 39-60, 2004.

[Linthicum 1999] Lithicum D. S., "Enterprise Application Integration". *Addison-Wesley*, 1999.

[Linthicum 2001] Lithicum D. S., "Enterprise Application Integration". *Mercator*, 2001.

[Linthicum 2004] Linthicum D. S., "Next Generation Aplication Integration". *Addison-Wesley*, 2004.

[Lonpégé 2002] Lonpégé C., "Le projet d'urbanisation du système d'information" *Dunod*, 2002.

[Lublinsky 2001] Lublinsky B., Achieving the Ultimate EAI Implementation, *eAI Journal*, 2001.

[Lublinsky 2003] Lublinsky B., and Tyomkin D., "Dissecting Service Oriented Architetcures". *Bunsiness Integration Journal*, 2003.

[Mädche et Zacharias 2002] Mädche A. and Zacharias V., "Clustering ontologybased metadata in the semantic web". In *Proceedings 13th ECML and 6th PKDD, Helsinki (FI)*, 2002.

[Maedche et al. 2002] Maedche A., Motik B., Silva N., and Volz R., "MAFRA - MApping FRAmework for Distributed Ontologies". In Proc. of the *13th European Conf. on Knowledge Engineering and Knowledge Management* EKAW-02, 2002.

[Maedche et Staab 2002] Maedche A., and Staab S., "Measuring Similarity between Ontologies". In Proc. of the *13th European Conf. on Knowledge Engineering and Knowledge Management* EKAW-02, 2002.

[Manouvrier 2001] Manouvrier B., "EAI - Intégration des applications d'entreprise". *Hermes*, 2001.

[Mathieu 2004] Mathieu H., "Modélisation conjointe de l'infrastructure et des processus pour l'administration pro-active de l'entreprise distribuée". Thèse de Doctorat, INSA de Lyon, France, 2004.

[Mayer 1995] R. J. Mayer, "Information Integration for Concurent Engineering (IICE), DEF3 Process Description Capture Method Report". AL-TR-81-4023, Air Force Material Laboratory, Wright Patterson AFB, Ohio 45433, 1995.

[McBride 2001] McBride B., "Jena: Implementing the RDF Model and Syntax Specification". In Steffen Staab et al (eds.): Proceedings of the *Second International Workshop on the Semantic Web* (SemWeb2001), 2001.

[McGuinness et al. 2000] McGuinness D. L., Fikes R., Rice J., and Wilder S., "The Chimaera Ontology Environment". In Proceedings of the *17th National Conference on Artificial Intelligence* (AAAI), 2000.

[McIlraith et al. 2001] McIlraith S., Son T., and Zeng H., "Semantic Web Services". *IEEE Intelligent Systems* Vol. 16 (No. 2), pp. 46-53, 2001.

[Medjahed 2004] Medjahed B. "Semantic Web Enabled Composition of Web Services". PhD Thesis, Virginia Polytechnic Institute and State University, USA, 2004.

[Meinadier 2002] Meinadier J.-P., "Le métier d'intégration des systèmes". *Lavoisier-Hermes*, 2002.

[Melnik et Decker 2000] Melnik S. and Decker S., "A layered approach to information modeling and interoperability on the Web". In Proceedings of the *European Conference on Digital Libraries* (ECDL'00) Workshop on the Semantic Web, Lisbon, Portugal, 2000.

[Mena el al. 1996] Mena E., Kashyap V., Sheth A., and Illarramendi A., "OBSERVER: An approach for query processing in global information systems based on interoperability across pre-existing ontologies". In Proceedings 1st *International Conference on Cooperative Information Systems* (IFCIS), Brussels, 1996.

[Mena et al. 2000] Mena E., Illarramendi A., Kashyap, V., and Sheth, A. P., "Observer: An approach for query processing in global information systems based on interoperation across pre-existing ontologies". *Distributed and Parallel Databases*, Vol. 8 (No. 2) pp. 223-271, 2000.

[Meteo-s 2005] METEOR-S, "METEOR-S project deliverables". [On Line] http://lsdis.cs.uga.edu/Projects/METEOR-S/, 2005.

[Microsoft 1992] Microsoft Corporation, "ODBC specification". ftp://ftp.microsoft.com/developr/ODBC, 1992.

[Microsoft 1996] Microsoft Corporation, "Distributed Component Object Model Protocol DCOM 1.0". *Internet Draft, Network working group*, 1996.

[Microsoft et DEC 1995] Microsoft Corporation and Digital Equipment Corporation, "The Component Object Model Specification, Version 0.9". [On Line] http://groups.csail.mit.edu/medg/ftp/emjordan/COM/THE%20COM.DOC, 1995.

[Miller et al. 2001] Miller R. J., Hernández M., Haas L. M., Yan L-L., Ho T. H., Fagin R., and Popa L., "The Clio Project: Managing Heterogeneity". SIGMOD Record Vol. 30 (No. 1), pp. 78-83, 2001.

[Mizoguchi 1995] Mizoguchi R., "Ikeda: Task Ontologies for reuse of Problem Solving Knowledge". In N. J. I. Mars (editors), Towards Very Large Knowledge Bases, *IOS Press*, 1995.

[Morris et al. 2004] Morris E., Levine L., Meyers C., Place P., and Plakosh D., "System of Systems Interoperability - Final Report". Techincal Report, CMU/SEI-2004-TR-004, ESC-TR-2004-004, 2004.

[Motta el al. 2003] Motta E., Domingue J., Cabral L., and Gaspari M., "IRS-II: A Framework and Infrastructure for Semantic Web Services". [On Line] http:// www.cs.unibo .it/gaspari/www/iswc03.pdf, 2003.

[Napoli 1997] Napoli A., "Une introduction aux logiques de descriptions". Rapport de recherche RR-3314, *INRIA*, 1997.

[Narayanan et McIlraith 2002] Narayanan S., Mcllraith S. A., "Simulation, Verification and Automated Composition of Web Services". 11th Intl. World Wide Web Conference (WWW2002), Honolulu, 2002.

[Niles et Pease 2001] Niles I., and Pease A., Towards a standard upper ontoogy. In *2th International Conference on Formal Ontologiy in Information Systems* (FOIS), Ogunquit, Maine, 2001.

[NIST 2005] National Institute of Standards and Technology, http://www.nist.gov/, 2005.

[Nist 2005] NIST, "PSL - Process Specification Language". [On Line] http://www.mel.nist.gov /psl/ontology.html , 2005.

[Noia et al. 2003] Noia T., Sciascio E. D., Donini F. M., and Mongiello M., "A System for Principled Matchmaking in an Electronic Marketplace". In Proceedings of the *Twelfth International Conference on World Wide Web* (WWW), 2003.

[Noy 2004] Noy N. F., " Semantic integration: a survey of ontology-based approaches". *SIGMOD Record*, Special Issue on Semantic Integration, Vol. 33 (No. 4), pp. 65-70, 2004.

[Noy et Musen 2003] Noy N. F., and Musen M. A., "The Prompt suite: Interactive tools for ontology merging and mapping". *International Journal of Human Computer Studies*, 59(6): 983-1024, 2003.

[NWG 1993] NWG - Network Working Group, "MIME - Multipurpose Internet Mail Extensions : Mechanisms for Specifying and Describing the Format of Internet Message Bodies". [On Line] http://www.nacs.uci.edu/indiv/ehood/MIME/MIME.html, 1993.

[Oasis 2004] OASIS, "Web Services Security: SOAP Message Security 1.0, (WS-Security 2004)". *OASIS Standards*, 2004.

[Octo 1999] Octo Technlogy, "Livre Blanc de l'EAI - Intégration des Applications d'Entreprise". *Octo Technology*, 1999.

[Octo 2005] Octo Technology, "Architecture Orientée Services (SOA), Une politique de l'interopérabilité". Livre Blanc, *Octo Technology*, 2005.

[Odsadzinski 1988] Odsadzinski A., "The Network File System (NFS)". *Computer Standards & Interfaces*, Vol. 8, The Netherlands, 1988.

[Ogbuji 2000] Ogbuji U., "Supercharging WSDL with RDF". [On Line] http://www-128.ibm.com/developerworks/library/ws-rdf/?dwzone=ws, 2000.

[Oki et al. 1993] Oki B., Pflueg M., Siegel D. and Skeen D., "The Information Bus - An architecture for extensible Distributed Systems". *Operating Systems Review*, Vol. 27(No. 5), pp. 58-68 , 1993.

[Oldham et al. 2004] Oldham N., Thomas C., Sheth A., and Verma K., "METEOR-S Web Service Annotation Framework with Machine Learning Classification". In Proceedings of the *1st International Workshop on Semantic Web Services and Web Process Composition* (SWSWPC'04) at the the 2nd International Conference on Web Services (ICWS '04), California, 2004.

[OMG-CORBA 1998] Object Management Group, "The Common Object Request Broker : Architecture and Specification, revision 2.2". OMG, USA, OMG TC Document formal/98-07-01, [On Line] http://www.omg.org/corba/corbiiop.htm, 1998.

[OMG-CWM 2005] Object Management Group, "Data Warehousing". CWM and MOF Resource Page, [On Line] http ://www.omg.org/cwm/, 2005.

[OMG-MDA 2006] Object Management Group, "OMG Model Driven Architecture". [On Line] http://www.omg.com/mda, 2006.

[OMG-MOF 2005] Object Management Group, "Meta-Object Facility (MOF), version 1.4". [On Line] http ://www.omg.org/technology/documents/formal/mof.htm, 2005.

[OntoMat 2002] ONTOMAT SOEP, "An ontology editor that includes features for semantic annotation of Web pages". Un-Supported and The Attick, [On Line] http://kaon.semanticweb.org/Members/rvo/Folder.2002-08-22.4813/ontomat_ soep, 2002.

[OntoWeb 2000] OntoWeb Consortium, "Deliverable 1.3: A survey on ontology tools - OntoWeb Ontology-based information exchange for knowledge management and electronic commerce". *IST Project* IST-2000-29243, 2000.

[OSF DCE 1995] OSF DCE, "Introduction to OSF DCE. Révision 1.1". *Open Software Foundation*, 1995.

[OWL 2004] OWL Coalition, "OWL - Web Ontology Language - (W3C Recommendation 10 February 2004)". *W3C*, [On Line] http://www.w3.org/ TR/2004/REC-owl-guide-20040210/, 2004.

[OWLSC 2004] OWL-S Coalition, "OWL-S: Semantic Markup for Web Services". *OWL-S Coalition*, [On Line] http://www.daml.org/services/, 2004.

[Ozsu et Valduriez 1999] Ozsu M.T., and Valduriez P., "Principles of Distributed database systems". *Prentice Hall*, 1999.

[Paolucci et al. 2002] Paolucci M., Kawamura T., Payne T. R., and Sycara K., "Semantic Matching of Web Services Capabilities". in Proceedings The First International Semantic Web Conference (ISWC), Sardinia, Italy, 2002.

[Paolucci et al. 2003] Paolucci M., Ankolekar A., Srinivasan N., and Sycara, K., "The DAML-S virtual machine". In Proceedings of the Second International Semantic Web Conference, 2003.

[Papazoglou et Heuvel 2006] Papazoglou M. P., and van den Heuvel W. J., "Service Oriented Architectures: Approaches, Technologies, and Research Issues". To appear in *VLDB Journal*, 2006.

[Parent et Spaccapietra 1998] Parent C., Spaccapietra S., "Issues and approaches of database integration". Communications of the ACM, 41(5):166–178,1998.

[Peer 2002] Peer J., "Bringing together Semantic Web and Web services". In Proceedings of the *First International Semantic Web Conference* (ISWC), number 2342 in Lecture Notes in Computer Science, pp. 279-291, *Springer-Verlag*, 2002.

[Peltier et al. 2001] Peltier M., Bézivin J. and Guillaume G., "MTRANS: A general framework, based on XSLT, for model transformations". *Workshop on Transformations in UML* (WTUML), Genova, Italy, 2001.

[Petrie 1992] Petrie C., editor, "Enterprise Integration Modelling". The MIT Press, Cambridge, MA, 1992.

[Pontacq 2002] Pontacq M., "Commerce électronique sur Internet, Architecture des solutions". *Techniques de l'ingénieur*, H5304, pp.1-13, 2002.

[Pottinger et Bernstein 2003] Pottinger R., Bernstein P. A., "Merging models based on given correspondences". In Proceedings of the 29th International Conference on Very Large Data Bases (VLDB), pages 826–837. Morgan Kaufmann, 2003.

[Printz 2000] Printz J., Morganti G., et Wajnflasz J., "Programmation et systèmes transactionnels". *Techniques de l'ingénieur*, H2708, pp. 1-19, 2000.

[Programmez 2003] Thévenon, D., Gueguen, E., Galpultos, M., Kolawa, A., et K'Dual, E., "Réussir votre intégration: XML, EAI, WS". *Revue Programmez*! Hors Série, Go-o2 SARL, 2003.

[Psyché et al. 2003] Psyché V., Mendes O., et Bourdeau J., "Apport de l'ingénierie ontologique aux environnements de formation à distance". *Revue Sciences et Technologies de l´Information et de la Communication pour l´Éducation et la Formation*, Vol. 10, [On Line] http://sticef.univ-lemans.fr/num/vol2003/psyche-06s/sticef_2003_psyche_06s.htm, 2003.

[Quinot 2003] Quinot T., "Conception et réalisation d'un intergiciel schizophrène pour la mise en œuvre de systèmes répartis interopérables". Thèse de doctorat de l'Université Paris VI — Pierre-et-Marie-Curie, France, 2003.

[Rao et Su 2004] Rao J. and Su. X., "A survey of automated Web service composition methods". In Proceedings of the *First International Workshop on Semantic Web Services and Web Process Composition*, SWSWPC'2004, LNCS, San Diego, USA, Springer-Verlag. 2004.

[Reiss 1990] Reiss P., "Connecting Tools Using Message Passing in the Field Environment". *IEEE Software*, pp. 57-66, 1990.

[Reix 1995] Reix R., "Systèmes d'Information de Gestion". *Editions d'Organisation*, 1995.

[Rhizomik 2005] Rhizomik, "XML Schema to OWL". [On Line] http://rhizomik. upf.edu/, 2005.

[Ribière 1998] Ribière G., "Communication et traitement en mode message avec MQSeries". *Techniques de l'ingénieur*, Rapport technique H2-768, 1998.

[Rivard et Plantain 2003] Rivard F., et Plantain T., "L'EAI par la pratique". *Eyrolles*, 2003.

[Riveill et al. 2000] Riveill M., Balter R., et Boyer F.," Communication synchrone entre programmes par RPC et RMI". *Techniques de l'ingénieur*, 2000.

[Robertson et Jones 1976] Robertson S., and Jones S. K., "Relevance weighting of search terms". *Journal of the American Society for Information Science*, No. 27, pp. 129-146, 1976.

[RogueWave 2005] RogueWave, "Lightweight Enterprise Integration Framework". [On Line] http://www.roguewave.com/products/leif/, 2005.

[RosettaNet 2002] RosettaNet, "Rosetta Implementation Framework (RNIF): Core specification". Technical report, *RosettaNet*, 2002.

[Ross 1977] Ross D. T., "SADT, un langage pour communiquer". *Softech Inc*, USA, 1977.

[Rumbaugh et al. 1991] Rumbaugh J., Blaha M., Premerlani W., Eddy S., and Lorensen W., "Object-Oriented Modeling and Design". Englewood Cliffs NJ *Prentice Hall*, 1991.

[Rumbaugh et al. 2005] Rumbaugh J., Jacobson I., and Booch G., "UML 2.0 : Guide de Référence". *Campus Press France*, 2005.

[SAP 2005] SAP France, "SAP Netweaver". [On Line] http://www.sap.com/france/ solutions/netweaver/index.epx, 2005.

[Sassoon 1998] Sassoon J., "Urbanisation des systèmes d'information". *Hermes*, 1998.

[Schmidt 2000] Schmidt J., "Enabling next generation enterprises". *eAI Journal*, 2000.

[Schneider 2002] Schneider M. Interopérabilité et intégration des systèmes d'information, Revue "Ingénierie des systèmes d'information", Volume 6, n°3, Hermes, Paris, 2002.

[Sciore et al. 1994] Sciore E., Siegel M., Rosenthal A., "Using semantic values to falilitate interoperability among heterogeneous information systems". ACM Transactions on Database Systems, 19(2):254–290, 1994.

[Sekt 2004] Sekt, "Deliverables of SEKT Project". [On Line] http://www.sekt-project.com/, 2004.

[Semwebcentral 2005] SemWebCentral, "WSDL2OWL-S". [On Line] http://projects. semwebcentral.org/projects/wsdl 2owl-s/, 2005.

[Serain 2001] Serain D., "Enerprise Application Integration - L'architecture des solutions e-Business". *Dunod*, 2001.

[Serviceoriented 2003] Serviceoriented.org, "WS-Transaction". 2003.

[Sharon et Kemmerer 1999] Sharon and Kemmerer, "STEP, the Grand Experience". NIST special publication 939, *National Institute of Standards and Technology*, Gaithersburg, MD, 1999.

[Sheth 1999] Sheth A., "Changing focus on interoperability in information systems: from system, syntax, structure to semantics". In Norwell, Massachusetts*: Kluwer Academic Publishers*, 1999.

[Sheth et Kashyap 1993] Sheth A., and Kashyap V., "So far schematically yet so near semantically". In Proceeding IFIP TC2/WG2.6 [On Line] http://citeseer.ist. psu.edu/sheth93so.html, 1993.

[Shoe 2001] Shoe, "The SHOE Knowledge Annotator". [On Line] http://www.cs.umd. edu/projects/plus/SHOE/KnowledgeAnnotator.html, 2001.

[Singh et Huhns 2005] Singh M., and Huhns M., "Service-Oriented Computing, Semantics, processes, agents". *Wiley*, 2004.

[Sirin et al. 2002] Sirin E., Hendler J., and Parsia B., "Semi-automatic composition of Web services using semantic descriptions". In Proceedings of *Web Services: Modeling, Architecture and Infrastructure workshop in conjunction with ICEIS2003*, 2002.

[Sivashanmugam et al. 2003] Sivashanmugam K. Sheth A., Miller J., and Verma K., "Metadata and Semantics for Web Services and Processes". Book Chapter, Datenbanken und informationsystem, *Publication Hagen* , 2003.

[Smith et Welty 2001] Smith B., and Welty C., "Ontology: Towards a New Synthesis". Formal Ontologies in Information Systems (FOIS'01), *ACM*, 2001.

[Soley et Stone 1995] Soley R. M., and Stone C. M., "Object Management Architecture Guide, revision 3.0". *Object Management Group, Inc. et John Wiley & Sons*, Inc., USA, 1995.

[Sonic et al. 2005] Sonic Software Corporation, Amberpoint Inc., BearingPoint Inc., and Systinet Corporation, "A new SOA Maturity Model". [On Line] https://bpxstream1.bitpipe.com/xm/FdfAction/post, 2005.

[Spring 1996] Spring M. B., "Reference model for data interchange standards". *IEEE Computer*, Vol. 29, pp. 87-88, 1996.

[Srinivasan et al. 2006] Srinivasan N., Paolucci M., and Sycara K., "Semantic Web service Discovery in the OWL-S IDE". In the Proceedings of the *39th Hawaii International Conference on System Sciences*, 2006.

[Stohr et Nickerson 2001] Stohr E., and Nickerson J. V., "Intra Enterpise Integration: Methods and direction". *Addison Wesley*, 2001.

[Stojanovic et al. 2003] Stojanovic N., Studer R., and Stojanovic A., "An Approach for the Ranking of Query Results in the SemanticWeb". In Proceedings of the *Second International Semantic Web Conference* (ISWC), Vol. 2870 of Lecture Notes in Computer Science, pp. 500-516. *Springer-Verlag*, 2003.

[Stonebraker 1999] Stonebraker M., "Integrating Islands of Information". *eAI Journal*, 1999.

[Stuckenschmidt 2000] Stuckenschmidt H., "Ontologies for Semantic Information Integration: Opportunities and Open Problems". *EKAW-00*, [On Line] http://www.cs.vu.nl/~heiner/public/EKAW-00.pdf, 2000.

[Stumme et Madche 2001] Stumme G., and Madche A., "FCA-Merge: Bottom-up merging of ontologies". In *7th International. Conference on Artificial Intelligence* (IJCAI'01), pp. 225-230, Seattle, WA, 2001.

[Sumo 2005] SUMO home page, [On Line] http://ontology.teknowledge.com, 2005.

[Sun-JDBC 2005] Sun, "Java Database Connectivity (JDBC)". [On Line] http://java.sun.com/products/jdbc/, 2005.

[Sure et al. 2002] Sure Y., Erdmann M., Angele J., Staab S., Studer R. and Wenke D., "OntoEdit: Collaborative Ontology Engineering for the Semantic Web". In Proceedings of the *International Semantic Web Conference* 2002 (ISWC 2002), Sardinia, Italia, 2002.

[Swartout et al. 1997] Swartout B., Patil R., Knight K. and Russ T., "Use of Large-Scale Ontologies". *Spring Symposium Series on Ontological Engineering*. Stanford University, CA, pp. 138-148, 1997.

[Sycara et al. 2002] Sycara K., Widoff S., Klusch M., and Lu J., "Larks : Dynamic matchmaking among heterogeneous software agent in cyberspace". *Autonomous Agents and Multi-Agent Systems*, Vol. 5, pp. 173–203, 2002.

[Tardieu et al. 2002] Tardieu H., Rochfeld A., et Rolland C., "La méthode Merise Principes et Outils". *Editions d'organisation*, 2002.

[Tessier 1995] Tessier C., "La pratique des méthodes en informatique de gestion: typologie, analyse comparative, choix et mise en œuvre". *Editions d'organisation*, 1995.

[Tolk 2003] Tolk, Andreas. "Beyond Technical Interoperability – Introducing a Reference Model for Measures of Merit for Coalition Interoperability". 8th International Command and Control Research and Technology Symposium (ICCRTS), Washington, DC, Washington DC: Command and Control Research Program (CCRP), 2003.

[Tolk et Muguira 2003] Tolk A., Muguira J. A., "The Levels of Conceptual Interoperability Model". Fall Simulation Interoperability Workshop Orlando, Florida, 2003.

[Tosic et al. 2001] Tosic V., Mennie D., and Pagurek B., "On dynamic Service Composition and Its Applicability to E-business Software Systems". WOOBS'01 (Workshop on Object-Oriented Business Solutions) workshop (at ECOOP 2001), Budapest, Hungary, 2001.

[Toublant et al. 2002] Toublant P. J., Crepet S., Rafii N., et Graton Y., "Traitement des données industrielles par micro-informatique". *Techniques de l'ingénieur*, S8190, pp.1-13, 2002.

[Tuyet 2004] Tuyer L. A., "Fédération: une architecture logicielle pour la construction d'applications dirigée par les modèles". Thèse de Doctorat, Université Joseph Fourier de Grenoble, France, 2004.

[Ullman 1997] Ullman J. D., "Information integration using logical views. In Proceedings of the International Conference on Database Theory (ICDT), volume 1186, pages 19–40. Lecture Notes in Computer Science, 1997.

[Uschold et Gruninger 1996] Uschold M., and Gruninger M., "Ontologies: Principes, Methods and Applications". *Knowledge Engineering Review*, Vol. 11(No. 2), 1996.

[Uschold et Gruninger 2002] Uschold M., and Gruninger M., "Creating Semantically Integrated Communities on the World Wide Web". In *Semantic Web Workshop*, Honolulu, Hawaii-Invited Tallk, 2002.

[Valtchev et Euzenat 1997] Valtchev P., and Euzenat J., "Dissimilarity measure for collections of objects and values". In Proceedings Coen X. Liu and M. Berthold, editors, *Proc. 2nd Symposium on Intelligent Data Analysis.*, Vol. 1280, pp. 259–272, 1997.

[Van Heijst et al. 1997] Van Heijst G., Schreiber A. Th., and Wielinga B. J., "Using explicit ontologies in KBS development". *International Journal of Human-Computer Studies*, Vol. 46 (No. 2/3) pp.183-292, 1997.

[Verjus 2001] Verjus H., "Conception et construction de fédérations de progiciels". Thèse de Doctorat, Université de Savoie, France, 2001.

[Vernadat 1996] Vernadat F. B., "Enterprise modelling and integration: Principles and and applications". *Chapman & Hall*, London, 1996.

[Vernadat 1999] F. Vernadat, "Techniques de Modélisation en Entreprise. Applications aux processus opérationnels". Economica, 1999.

[Vernadat 2002] Vernadat F. B., "Enterprise modeling and integration (EMI)- Current status and research perspectives". *Pergamon - Annual Reviews in Control,* Vol. 26, pp. 15-25, 2002.

[Villalobos 2003] Villalobos J., "Fédération de composants: Une architecture logicielle pour la composition par coordination". Thèse de Doctorat, Université Joseph Fourier de Grenoble, France, 2003.

[Visser et Stuckenschmidt 2002] Visser U., and Stuckenschmidt H., "Interoperability in GIS - Enabling Technologies". In Proceeding of *5th AGILE Conference on GIS*, Spain, 2002.

[W3C 2006] W3C, "The World Wide Web Consortium". [On Line] http://www.w3.org/, 2006.

[W3C-RDFS 2002] W3C, "RDFS - Resource Description Framework Schema Specification". [On Line] http ://www.w3.org/TR/rdf-schema/, 2002.

[W3C-WSA 2006] W3C, "Web Services Activity". [On Line] http://www.w3.org/2002/ws/, 2006.

[W3C-XML 2004] World Wide Web Consortium (W3C), "XML". [OnLine] http://www.w3.org/XML, 2004.

[W3C-XSD 2005] W3C, "XML Schema Part 0: Primer Second Edition". *W3C Recommendation* 28 October 2004. [On Line] http://www.w3.org/TR/2004/REC-xmlschema-0-20041028/, 2004.

[W3C-XSLT 1999] W3C, "XSLT". [On Line] http://www.w3.org/TR/xslt, 1999.

[Wache et al. 1999] Wache H., Scholz Th., Stieghahn H., and Konig-Ries B., "An integration method for the specification of rule–oriented mediators". Proceedings of *DANTE'99*, pp. 109-112, Kyoto, Japan, 1999.

[Wache et al. 2001] Wache H., Vögele T., Visser U., Stuckenschmidt H., Schuster G., Neumann H., and Hübner S., "Ontology-Based Integration of Information - A Survey of Existing Approaches". In Stuckenschmidt, H., editor, IJCAI-2001 *Workshop on Ontologies and Information Sharing*, pp., 2001.

[Wangler et Paheerathan 2000] Wangler B., and Paheerathan S., "Horizontal and vertical integration of organizational IT systems". *Information Systems Engineering*, 2000.

[Ward 1963] Ward J. H., "Hierrachical grouping to optimize an objective function". *Journal of the American statistical association*, Vol 58, pp. 238-244, 1963.

[Wasserman 1990] Wasserman A I., "Tool Integration in Software Engineering Environments". In Software Engineering Environments: *Work-shop on Environments*, Berlin, 1990.

[Wegner 1996] Wegner P., "Interoperability". *ACM Computing Survey*, Vol. 28 (No. 1), pp. 285-7, 1996.

[Weston 1993] Weston R. H., "Steps towards enterprise wide integration: a definition of needs and first generation open solutions". International Journal of Production Research, 31(9), 2235-2254, 1993.

[WfMC 2000] Workflow Management Coalition, "The Workflow reference model". Document Number WFMC-TC-1003, *Workflow Management Coalition*, 2000.

[Wiederhold 1992] Wiederhold G., "Mediators in the Architecture of Future Information Systems". *IEEE Computer*, Vol. 25 (No. 3), pp. 38-49, 1992.

[WonderWeb 2005] WonderWeb, Deliverables of WonderWeb project, [On Line] http://wonderweb.semanticweb.org/, 2005.

[Woods 1975] Woods W. A., What's in a link: foundations of semantic networks. In D.G.Bobrow & A. M. Collins (Eds.), 1975.

[Wroe et al. 2003] Wroe C. J., Stevens R., Goble C. A., and Ashburner M., "A methodology to migrate the gene ontology to a description logic environment using DAML+OIL". *Pacific Symposium on Biocomputing* Vol. 8, pp. 624-635, 2003.

[WSDL-S 2005] WSDL-S, "Web Service Semantics - WSDL-S, Version 1.0". W3C Member Submission, [On Line] http://www.w3.org/Submission/WSDL-S/, 2005.

[WSMF 2005] WSMF, "Semantic Web Services Framework (SWSF) Overview". W3C Member Submission, [On Line] http://www.w3.org/Submission/SWSF/, 2005.

[WSMO 2005] WSMO, "Web Service Modeling Ontology Project Deliverables". *WSMO Project*, 2005.

[Xu et al. 2003] Xu Y., Sauquet D., Degoulet P., and Jaulent M. C., "Component-based mediation services for the integration of medical applications". Elsevier, *Artificial Intelligence in Medicine* Vol. 27, pp. 283-304, 2003.

[Yang et Papazoglou 2002] Yang J. and Papazoglou P., "Web component: A substract for Web service reuse and composition". In Proceedings of the *14th International Conference for Advanced Information Systems Engineering* (CAiSE), Vol LNCS 2348, Toronto, Canada, 2002.

[Zweigenbaum 1994] Zweigenbaum P., "MENELAS : An Access System for Medical recods Using Natural Lnguage". *Computer Methods and Programs in Biomedecine*, Vol. 45, pp. 117-120, 1994.

Ecole Nationale Supérieure des Mines
de Saint-Etienne

Order N° : 418 I

First Name, Last Name : Saïd IZZA

Title of the thesis : INTEGRATION OF INDUSTRIAL INFORMATION SYSTEMS
A flexible approach based on semantic services

Speciality : Computer Science

Keywords : Information System, Application, Integration, Interoperability, Ontology, Semantics, Web Service, Discovery, Mediation, Urbanisation, Flexibility.

Abstract :
Over the last decade, the field of industrial information systems was deeply transformed under the influence of the evolution of software technologies (objects, components, web services,…), the evolution of hardware technologies (Moore law), and also the evolution of organisations (fusions, acquisitions, globalisation). Consequently, the information systems became more and more complex and heterogeneous. Those need to be integrated in order to make them communicate and cooperate. It is the problem of information system integration. This work treats this latter problem and precisely the semantic integration one. It proposes a flexible approach that is based on semantic services and that combines both ontologies and web services in order to overcome some issues related to the semantic integration problem.
After having exposed our research problematic, we reviewed the most important related works that concern the integration of industrial information systems. The analysis of the state of the art let us to consider mainly two integration levels: syntactic and semantic integration. This latter constitutes a crucial problem that is not is not correctly addressed by today's integration solutions that focus mainly on the syntactical integration. Addressing the semantic aspect will promote the integration by providing it more consistency and robustness.
Focalising our work on semantic services, that constitute the most efficient and flexible approaches that deal with semantic integration, allowed us to note some important limitations that are mainly: the discrepancy of current ontology architectures to correctly capture the semantics of industrial applications, the lack of methodologies in order to build ontologies and also semantic services, the lack of pertinent discovery and mediation approaches for intra-entreprise integration issues, and the complexity of the technologies related to the exploitation of the enterprise semantics.
Thus, we have proposed a flexible approach for integrating industrial applications that is named *ODSOI* (Ontology-Driven Service-Oriented Integration). This approach focuses on three complementary problematics that are respectively: the building of the architecture of enterprise services ($\boldsymbol{P_{Syn}}$) that defines and structures enterprise services, the building of the semantic architecture ($\boldsymbol{P_{Sem}}$) that semantically describes enterprise services, and the building of the integration architecture ($\boldsymbol{P_{Int}}$) that defines integration mechanism based on enterprise semantics. Our approach is based on three major principles that are openness, unification and urbanisation. First, the openness principle imposes us to use and to conform to industrial standards such as WSDL and OWL. Second, the unification principle allows to make information system components uniform. Third, the urbanisation principle allows to correctly structuring the service architecture, the semantic architecture and also the integration architecture.
Basing on these three complementary architectures, we implement a prototype that creates, manages and exploits integration projects. Finally, we led various experimentations of the prototype that concern the domain of preventive maintenance within an industrial enterprise.

Ecole Nationale Supérieure des Mines
de Saint-Etienne

N° d'ordre : 418 I

Prénom, Nom : **Saïd IZZA**

Titre de la thèse : INTEGRATION DES SYSTEMES D'INFORMATION INDUSTRIELS
Une approche flexible basée sur les services sémantiques

Spécialité : Informatique

Mots clefs : Système d'Information, Application, Intégration, Interopérabilité, Ontologie, Sémantique, Service, Découverte, Médiation, Urbanisation, Flexibilité.

Résumé :
Le domaine des systèmes d'information industriels s'est profondément transformé ces dernières années sous l'influence de l'évolution des technologies logicielles (objets, composants, service web, …), de l'évolution des technologies matérielles (loi de Moore), et aussi de l'évolution des organisations (fusions, acquisitions, mondialisation). Conséquence de tous ces facteurs, les systèmes d'information deviennent de plus en plus complexes et hétérogènes qu'il convient alors d'intégrer afin de les faire communiquer et les faire coopérer. Il s'agit du problème d'intégration des systèmes d'information. Notre travail s'inscrit dans cette problématique, et plus précisément dans le cadre de l'intégration sémantique de systèmes d'information de grandes entreprises industrielles. Il propose une approche flexible basée sur les services sémantiques, en combinant à la fois les ontologies et les Services Web.
Après avoir exposé la problématique, nous avons présenté les différentes techniques d'intégration des systèmes d'information industriels. L'analyse de l'état de l'art nous a permis de retenir deux niveaux d'intégration: l'intégration syntaxique et l'intégration sémantique. Cette dernière constitue un problème crucial de l'intégration des systèmes d'information. Jusqu'à présent, ce problème n'est toujours pas correctement traité. Les solutions actuelles se focalisent plutôt sur les techniques d'intégration syntaxique. La prise en compte de l'aspect sémantique peut promouvoir l'intégration en lui apportant plus de consistance et de flexibilité.
En nous focalisant sur les services sémantiques, nous avons constaté un certain nombre de lacunes dont l'inadéquation des architectures actuelles des ontologies à capturer de façon flexible et efficace la sémantique des applications industrielles, le manque de méthodologie à mettre en œuvre pour définir les ontologies et aussi les services sémantiques, le manque d'approches de découverte et de médiation de services dans le contexte intra-entreprise, et la complexité inhérente à l'utilisation des technologies associées à l'exploitation de la sémantique.
Partant de ce constat, nous avons alors proposé une approche flexible d'intégration des applications industrielles qui s'intitule *ODSOI* (Ontology-Driven Service-Oriented Integration). Cette approche se focalise principalement sur trois sous-problématiques complémentaires qui sont respectivement la construction d'une architecture de services ($\boldsymbol{P_{Syn}}$) permettant de définir et de structurer les services d'entreprise, la construction d'une architecture sémantique ($\boldsymbol{P_{Sem}}$) permettant de définir et de structurer les ontologies d'entreprise servant à enrichir sémantiquement les services d'entreprise, et la construction d'une architecture d'intégration ($\boldsymbol{P_{Int}}$) permettant d'offrir des mécanismes d'intégration basés sur la sémantique. Notre approche repose sur trois principes majeurs qui sont l'ouverture, l'unification et l'urbanisation. Le principe d'ouverture impose de s'inscrire dans le cadre d'utilisation de standards industriels tels que WSDL et OWL. Le principe d'unification permet d'uniformiser les composants du système d'information. Et en dernier lieu, le principe d'urbanisation permet de mieux structurer l'architecture des services, l'architecture sémantique et aussi l'architecture d'intégration.
Nous basant sur ces trois architectures, nous avons implémenté un prototype permettant de créer, de gérer, et de mettre en œuvre des projets d'intégration. Nous avons enfin réalisé diverses expérimentations portant sur le domaine de la maintenance préventive en milieu industriel.