MINISTRE DE L'ENSEIGNEMENT SUPERIEUR ET DE LA RECHERCHE SCIENTIFIQUE.
UNIVERSITE MENTOURI DE CONSTANTINE.
FACULTE DES SCIENCES DE L'INGENIEUR.
DEPARTEMENT D'INFORMATIQUE.

N° d'ordre:
Série:

THESE

Présentée pour obtenir le diplôme de Doctorat
En Science.

Proposition d'une architecture d'intégration des applications d'entreprise basée sur l'interopérabilité sémantique de l'EbXML et la mobilité des agents

---

Présentée par : ***Dirigée par :***

Razika DRIOUCHE Pr. Mme Zizette BOUFAIDA

SOUTENUE LE / /2007, à h.

Devant le jury

Président :	Pr. Zaidi SAHNOUN	Université de Constantine.
Rapporteur :	Pr. Zizette BOUFAIDA	Université de Constantine.
Examinateurs :	Pr. Mahmoud BOUFAIDA	Université de Constantine.
	Pr. Mohamed BENMOHAMED	Université de Constantine.
	Pr. Aicha MOKHTARI	Université de USTHB, Alger.
	Pr. M. Tayeb LASKRI	Université Badji Mokhtar de Annaba.
Invité	Pr. Fabrice KORDON	Université Marie & Pierre Curie Paris 6.

Table de matière

CHAPITRE 2 : Ontologies : état de l'art et intégration

CHAPITRE 3 : Web services : Définition et Composition

CHAPITRE 4 : Les Standards d'échange et la Collaboration des Partenaires

CHAPITRE 5 : Intégration A2A

CHAPITRE 6 : Collaboration B2B

CHAPITRE 7 : Etude de cas : Illustration et réalisation

Conclusion générale

Liste des figures

Liste des tableaux

La conscience d'en savoir plus suscite en moi la conscience d'en savoir peu ...
—José SARAMAGO, Histoire du siège de Lisbonne.

Introduction générale

1. Contexte du travail

Lors de ces dernières années, l'informatique de l'entreprise est devenue un domaine très vaste qui inclut plusieurs thèmes et axes de recheches. Aujourd'hui, le SI (Système d'Information) est consideré comme le noyau central de l'entreprise. Cette dernière essaie d'exploiter au maximun les nouvelles technologies de l'information pour faciliter la réalisation de ses tâches quotidiènnes. Cependant, l'évolution rapide de ces technologies d'une part, et la multifonctionnalité de l'entreprise, d'autre part, ont crée un grand problème pour l'homogénéité du SI. Ce dernier est constitué de nombreuses applications hétérogènes baties sur des technologies differentes. L'hétérogénéité peut apparaître dans l'utilisation de différents langages de programmation, le fonctionnement sur des plateformes et des systèmes d'exploitation variés. En outre, ces applications ont été développées indépendamment. Il devient alors un système incohérent, mal adapté et incompatible.

Le développement des applications a eu tendance à se faire en fonction des besoins des entreprises. Une nouvelle application était créée à chaque fois qu'un nouveau besoin émergeait. Il est souvent nécessaire de développer des applications permettant de résoudre les problèmes à court terme afin de s'adapter à la vitesse de croissance des entreprises. La plupart des entreprises utilisent diverses applications fonctionnant sur des postes de travail ou des serveurs. Les données des entreprises peuvent prendre différentes formes : des tableurs aux bases de données en passant par les logiciels commerciaux spécifiques et de comptabilités.

2. Problématique

Le SI de l'entreprise est l'un des moyens dont elle dispose pour améliorer ses performances économiques : disposer d'une information plus complète, plus analytique, plus fiable, plus rapide constitue un enjeu stratégique.

Les SI et leurs architectures évoluent bien sûr en fonction de l'apparition de nouvelles technologies mais surtout en fonction d'impératifs économiques (pression concurrentielle accrue, déréglementation du marché, contrôles de régulation sur les prix, contingences économiques courantes …) qui encouragent des initiatives qui ajoutent à la complexité de l'environnement technologique de l'entreprise en forcant les directions informatiques à un raccourcissement des délais de prise en compte des nouveaux besoins fonctionnels avec des coûts toujours réduits. Le SI doit être assez souple pour absorber rapidement ces nouveaux besoins tout en accompagnant les évolutions technologiques.

Pour garder la maîtrise de leur organisation et respecter leurs objectifs économiques, le entreprises font souvent évoluer leur système d'information en évitant de faire table rase du patrimoine applicatif, et ainsi sacrifient la cohérence de leur système informatique en liant les applications entre elles, en ajoutant progressivement un développement spécifique sur une nouvelle technologie ou un progiciel dont les fonctionnalités correspondent mieux aux besoins du domaine (approche dite du « best of breed »).

Les SI complexes sont donc forcément hétérogènes en termes d'architectures (centralisée, client-serveur, Internet, etc.) de plates-formes (Mainframe, Windows, UNIX, etc.), de bases de données (Oracle, DB2, Sybase, SQL Server…), d'environnements de développement (C/C++, Java, ASP, Java), et ils utilisent des applications répondant à des besoins fonctionnels précis, certaines ayant fait l'objet d'un développement spécifique, et d'autres étant des solutions packagées qui ont pu nécessiter des investissements importants (ERP, SCM ou CRM). De plus, les rachats, fusions ou réorganisations d'entreprises sont des facteurs qui vont accentuer cette tendance …

Avec l'évolution d'Internet, les entreprises se regroupent afin de constituer des holdings sans qu'elles changent leur situation géographique, ni leur infrastructure et ni leur SI. Elles font collaborer les services concernés et surtout donnent accès à leurs informations. Donc, la mondialisation n'a fait qu'accentuer les échanges interentreprises par le biais d'internet. Depuis quelques années, des méthodes et des outils d'interfaçage ont été créés afin d'échanger, de transformer et de rediriger les données. A présent, l'EAI (Enterprise Application Integration) va plus loin, en proposant une architecture qui constitue un véritable cadre d'intégration des applications existantes et celles à venir. L'intégration pose certains problèmes, comment traiter l'hétérogénéité des applications et fournir une vue consolidée des données tout en offrant l'accès aux logiciels de l'intérieur ou de l'extérieur de l'entreprise. De plus, comment peut-on enchaîner automatiquement les tâches liées à une fonction business.

Aujourd'hui, plus de 40 % des budgets de développement en informatique sont liés à l'intégration de données dans les SI. Il s'agit donc d'une problématique stratégique pour les entreprises L'intégration entre des bases de données et/ou des applications hétérogènes, avec un traitement batch (en différé) ou en temps réel, implique souvent la mise en oeuvre de projets coûteux et complexes.

L'intégration est devenue pour beaucoup de nos entreprises, notamment pour les grands comptes, une véritable préoccupation et une nécessité incontournable pour faire face aux exigences sans cesse évolutives du marché. Les fortes 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, plus flexibles et plus réactifs, afin de pouvoir mieux soutenir la stratégie de l'entreprise. Pour faire face à ces multiples contraintes et à ces nouvelles exigences, il devient alors nécessaire défaire appel à la notion d'intégration d'applications, qui devient l'un des enjeux majeurs des SI actuels.

3. Contribution

Les SI ayant atteint un certain stade de complexité sont confrontés à un problème classique : comment intégrer les applications entre elles ? Par exemple : l'application les ventes a besoin de données présentes dans l'ERP, et la gestion des commandes a besoin de données présentes dans les SGBDR, et le CRM.

Les solutions traditionnelles n'abordent le problème de l'intégration entre applications que par les données : transferts périodiques de fichiers, partage de base de données, réplication et transformation des données utilisées par les applications …

La thèse que nous défendons s'articule autour de deux (2) contributions associées aux deux dimensions principales de l'EAI à savoir : l'intégration intra-entreprise (A2A) et l'intégration inter-entreprises (B2B).

La première contribution de cette thèse concerne l'integration des applications d'entreprise (A2A). A ce niveau l'hétérogénéité est gérée par le développement des ontologies d'applications.

- Construction d'ontologie d'application ;
- Proposition d'un processus d'intégration d'ontologies basées sur le frameworMAFRA (MApping FRAmework) et le concept de semantic bridge ;
- Proposition d'un scénario de communication pour étabir l'échange entre les applications hétérogènes et satisfaire le client.

La deuxième contribution concerne la collaboration B2B par la proposition d'une ontologie de processus métiers sert à supporter la collaboration entre les partenaires.

- Construction d'ontologie des processus par l'intégration des ontologies de services basé sur un scénario d'intégration et les ontologies de service ;
- Utilisation du langage BPEL4WS pour spécifier l'enchaînement des services modélisant les processus métiers ;
- Étendre le scénario de collaboration EbXML par des agents de recherches et des agents mobiles de négociation des transactions commerciales.

4. Plan du document

Ainsi notre thèse vise à proposer une architecture pour l'intégration d'applications d'entreprise. En parcourant cette thèse, le lecteur pourra découvrir les différents aspects de notre proposition.

La thèse est structurée en sept (7) chapitres principaux, plus cette introduction et une conclusion générale.

Le premier chapitre, intitulé *Intégration des Applications d'Entreprise*, présente un état de l'art du domaine d'integration en général et l'EAI en particulier. Nous donnons un aperçu des types, approches, architectures et modèles d'intégration. De plus, un panorama des technologies (middlewares et progiciels) et standards developpés dans ce contexte. Nous achevons ce chapitre en présentant des axes de recherches pour l'intégration d'applications, tout en positionnant les idées de notre travail à proposer.

Le deuxième chapitre, intitulé *Les ontologies : Etat de l'art et intégration*, décrit le point de départ qui a servi de base à régler le problème d'hétérogeneité. Nous avons présenté la notion d'ontologie, les principaux formalismes de représentation de connaissances, les méthodologies de developpement et les langages d'implémentation (les éditeurs). Enfin, nous focalisons sur le problème d'hétérogénéité et les approches d'intégration.

Le troisième chapitre, intitulé *Les Web services : Définition et Composition*, c'est l'élément de base utilisé pour décrire l'enchainement des services composant le processus métier. Nous présentons les élements de définition des Web services et les technologies les plus importantes qui les constituent, en l'occurrence : XML, SOAP, WSDL, et UDDI. Nous introduisons dans ce chapitre quelques approches de composition des Web services et les langages de spécification tels que : WSFL, XLANG et BPEL4WS.

Le quatrième chapitre, intitulé *Les standards d'échange et la collaboration des partenaires*, présente une description détaillée, des standards d'échange à savoir l'EDI, BizTalk, RosettaNet et EbXML permettant de mettre effectivement en oeuvre la collaboration entre partenaires pour assurer l'intégration B2B. Partant de cette description, nous avons adopté le scénario de collaboration ebXM. Une éxtension de ce dernier par les agents mobile afin d'assurer la recherche des meilleurs partenaires et la négociation des paramètres commerciales sera détaillé en chapitre 6.

Le cinquième chapitre, intitulé *Intégration A2A*, présente une architecture d'intégration des applications de l'entreprise à deux niveaux, applicatif et collaboratif. Nous avons montré l'apport des ontologies pour gérer l'hétérogeneité et assurer un fonctionnement global cohérent du système d'intégration à travers le scénario de communication introduit. De plus, nous avons proposé un processus d'intégration des applications incluant deux (2) sous-processus. Le premier sert au developpement des ontologies d'application. Le second porte sur l'intégration des ontologies developpées. Ensuite, un scénario de communication est proposé afin de montrer l'acheminement des requêtes clients vers les applications de l'entreprise

Le sixième chapitre, intitulé *Collaboration B2B,* décrit le niveau d'intégration inter-entreprise par la construction d'ontologie des processus obtenue par le developpement des ontologies de services et l'utilisation d'une strategie de composition. En plus, le langage BPEL4WS est adopté pour spécifier l'enchainement des services sélectionnés. Enfin, nous proposons une extension du scénario de collaboration EbXML par l'introduction des agents mobile pour assurer d'interopérabilité entre les entreprises partenaires utilisant des modèles d'échanges hétérogènes. Enfin, nous réalisons ce scénario de collaboration avec Jbuilder (version X) et la plateforme Jade.

Le septième chapitre, intitulé *Etude de cas : Illustration et réalisation,* décrit l'exemple des agences de voyage pour illustrer notre propositionnous avons choisi ce cas parce qu'il fait intervenir plusieurs partenaires pour assurer les reservations des vols, de chambre d'hôtels et la location de voiture. Nous présentons le processus de construction de l'ontologie d'application avec toutes ses étapes, partant de la méta-modélisation à l'implémentation. Au niveau collaboratif, nous détaillons le scénario de collaboration EbXML étendu par les agents de recherche et ceux de négociation, ainsi l'apport du scénario d'intégration des processus métier pour bien illustrer la collaboration entre les partenaires.

Finalement, un dernier chapitre du manuscrit présente une conclusion générale de ce travail et définit quelques perspectives de continuation.

CHAPITRE 1

Intégration des Application d'Entreprise

1. Introduction

Nous assistons, ces dernières années, à l'avènement du concept d'intégration d'applications qui a révolutionné fortement la manière avec laquelle les applications de l'entreprise doivent et peuvent communiquer. Ce concept qui est à la fois récent et très ancien constitue un enjeu majeur pour les systèmes d'information. L'intégration d'applications constitue sans doute un domaine prometteur en constante évolution. Ce qui montre son importance, notamment pour les grandes entreprises dont - les enjeux se déclinent souvent en termes de souplesse et de réactivité de leurs systèmes d'information.

L'intégration d'applications est donc un concept clé pour toute entreprise qui veut rester compétitive ou bénéficier d'un avantage concurrentiel. Mais devant la prolifération de solutions, de technologies et d'approches d'intégration, il devient de plus en plus difficile de maîtriser l'éventail des connaissances liées à ce domaine. Partant de ce constat, nous avons cherché à présenter dans ce chapitre un panorama des approches et des technologies d'intégration d'applications, ainsi que quelques orientations futures qui nous semblent prometteuses.

L'intégration est souvent considérée comme l'un des derniers challenges de l'informatique. Il désigne l'action d'intégrer. De son étymologie latine "integrare", intégrer signifie "rendre entier". 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 l'interopérabilité et à la cohérence des systèmes d'information.

2. Aperçu sur l'intégration

Que recouvre exactement l'intégration des applications? Quels en sont les bénéfices? Comment les entreprises peuvent-elles réagir aux changements rapides des informations, des technologies et des systèmes?

Au sein de l'entreprise, il peut exister plusieurs approches permettant d'appréhender le problème d'intégration. Principalement, on peut distinguer quatre types fondamentaux. Il s'agit respectivement en fonction de leur degré de complexité, de l'intégration de données, de processus, des interfaces et des applications [17].

2.1. Intégration de données

C'est la forme la plus simple de l'intégration. Elle apparaît au niveau des bases de données. D'une part, elle est assurée par duplication des copies d'une partie ou de toute la base de données dans une ou plusieurs applications. D'autre part, l'intégration s'effectue par

le transfert des données, en utilisant des outils pour permettre aux données d'émigrer d'une application à une autre. Ce transfert de données est généralement réalisé par ETL (Extract, Transform and Load) [6]. ETL est un moteur qui extrait, transforme, épure puis charge les données à partir de différentes applications vers des entrepôts de données. Il est aujourd'hui la solution la plus préconisée dans l'intégration des données.

2.2. Intégration des applications

L'intégration d'applications porte sur l'interconnexion d'applications hétérogènes, le plus souvent développées de façon indépendante et voire de façon incompatible [39]. L'AI permet principalement de faire communiquer tout type d'applications (CRM - Customer Relationship Management, ERP -Entreprise Ressource Planning, SCM - Supply Chain Management, etc.), ce qui peut constituer des enjeux énormes notamment pour les grosses entreprises qui disposent d'une masse importante d'applicatifs. Sur le terrain, l'Ai s'affiche par une multitude de produits commerciaux portant des logos assez variés tels que EAI ou ESB (Business Work de Tibco, Integrator de Mercator, e*Gate Integrator de SeeBeyond, Websphere d'IBM, Biztalk de Microsoft, Businessware de Vitria, Intégration Server de WebMethods, EntireX de SoftwareAG, XMLBus d'Iona, Sonic ESB de Sonic Software, etc.) [20] [28], et dont l'objectif est de permettre de rationaliser et fluidifier le système d'information afin de le rendre plus flexible et plus réactif.

2.3. Intégration des processus

C'est la forme la plus complexe de l'intégration. Elle sert à rendre valable une application dans le contexte d'une autre sans la dupliquer. Elle permet aussi de construire de nouveaux processus métier à base des applications et progiciels existants. Ceci crée de nouvelles opportunités pour l'entreprise à moindre coût. Les données circulant dans la nouvelle organisation sont accédées et maintenues selon une logique de métier (business logic) qui a des règles et une sécurité de données. Ces données ne sont plus simples mais des objets métier (BOD : Business Object Document, ex bon de commande) qui portent déjà un sens [5]. Grâce à cette forme d'intégration, les nouveaux processus métier qui les manipulent sont crées.

L'intégration du SI passe par l'intégration des briques le composant étant présent à un moment donné. Aujourd'hui, la brique élémentaire du SI est l'application. C'est donc tout logiquement que l'intégration du SI est considérée comme l'intégration des applications qui le composent. Néanmoins, il ne suffit pas de connecter les applications entre elles selon les besoins en information de telle ou telle application pour dire que l'on fait de l'intégration de SI. Il ne faut pas prendre l'application comme un élément stable et autonome qu'il faudrait connecter à d'autres éléments stables et autonomes. Il faut plutôt intégrer ce pourquoi les applications ont été conçues. Il faut donc revenir à leur processus de conception afin de définir l'objet de l'intégration de SI.

3. Intégration d'applications d'entreprise

L'EAI est un terme qui regroupe les méthodes et les outils visant à moderniser, consolider et coordiner les différentes solutions logicielles d'une entreprise Typiquement, une entreprise dispose d'applications et de bases de données qu'elle désire continuer à utiliser tout en développant ou en migrant de nouvelles applications dans le but d'exploiter un site Web (e-commerce) ou construire un extranet avec ses partenaires [1].

Elle peut également vouloir ajouter de nouvelles fonctionnalités à ses applicatifs, afin de les pérenniser, mais ne pas vouloir/pouvoir modifier ses systèmes d'information. Dans ce cas, elle peut développer une solution en intranet et, grâce à l'EAI, la relier aux autres systèmes existants. De plus, il est ainsi possible d'effectuer un reporting centralisé (consolidé) des données en provenance de sources très diverses. Ainsi, l'EAI peut être vu comme une meta-structure coordonnant des applicatifs intranet, Internet.

L'objectif majeur des technologies d'intégration est de rendre l'entreprise plus efficace, plus rentable et plus accessible aux tiers ; en d'autre termes, d'en faire un partenaire commercial plus efficace [3] (Gartner Group).

Avec l'utilisation des techniques d'intégration, les bénéfices pour les systèmes d'informations sont [5] :

- **Flexibilité** : une modification dans une application n'a d'impact que sur le serveur d'applications, et non sur les X destinataires qui l'utilisent,
- **Robustesse** : la centralisation des flux permet un réel suivi, des sauvegardes, des reprises, etc.,
- **Sécurité** : les technologies d'intégration se fondent sur des mécanismes asynchrones et peuvent offrir une gestion poussée de la sécurité. L'EAI offre une infrastructure qui permet d'assurer la sécurité des flux de données véhiculés et les fonctions d'administration et d'exploitation.
- **Modularité** : l'EAI donne la possibilité de modéliser les processus et de rationaliser les échanges inter-applicatifs au sein du SI.
- **Gestion de l'hétérogeneité** : l'EAI permet d'encapsuler l'hétérogeneité par unification des interfaces et alignement des processus métiers.

3.1. Définitions et challenges

L'EAI : c'est un concept qui regroupe un ensemble de méthodes, de technologies et d'outils pour consolider et coordonnée différentes applications d'une entreprise afin d'unifier son SI [5].

De point de vue architecture, l'intégration d'applications d'entreprise est une plate-forme reliant les applications hétérogènes du système des systèmes d'information autour d'un bus logiciel commun, chargé du transport des données [19].

Plusieurs définitions ont été proposées pour l'EAI :

Bordage définit l'EAI comme une plate-forme reliant les applications hétérogènes du SI autour d'un bus logiciel commun, charger du transport des données [87].

Pour Linthicum, EAI est une approche stratégique permettant de relier plusieurs SI les uns aux autres, aussi bien au niveau service qu'au niveau informationnel, en permettant ainsi le partage à la fois des informations et des processus [31].

En ce qui nous concerne, nous considérons l'EAI comme étant une approche assurant l'interopérabilité sémantique et le partage des données, des traitements et des processus entre les applications hétérogènes, dans le but d'accroître la flexibilité et la réactivité du SI. [74].

L'EAI sert à réaliser les objectifs suivants :

- Créer une infrastructure intégrée pour relier des systèmes (applications et sources de données) dispersés au sein de l'entreprise incorporée.
- Fournir une solution full duplex et bidirectionnelle pour partager des informations de manière similaire entre les diverses applications de l'entreprise et les nouveaux progiciels commerciaux
- Permettre l'échange de données et de processus entre les différentes applications d'une entreprise de manière rapide, efficace et transparente.

Bien que l'intégration des applications soit pertinente dans les approches et les technologies qu'elle met en œuvre, il n'en demeure pas moins qu'elle présente certaines limitations. Et il est possible d'en résumer les plus importantes, comme suit [5] [9] [20] [21] :

- *BPM*: Les solutions actuelles d'intégration des applications souffrent du manque de services pour la modélisation et l'intégration des processus métiers. Ceci engendre un fossé en rendant délicat l'alignement métier technologie.

- *Sémantique*: Les solutions actuelles d'intégration des applications ne résolvent pas le problème lié à la sémantique, dans la mesure où les mécanismes proposés sont basés au mieux sur des mappings ou des transformations syntaxiques.

- *Ouverture*: Les solutions d'Aï sont généralement propriétaires, ce qui pose le problème de standardisation et d'ouverture et ce malgré la percée des e-Services qui n'ont toujours pas atteint le degré de maturité souhaité en demeurant toujours limités notamment au niveau sécurité, transactions, BPM, et qualité de services .

- *Adaptation*: Les solutions actuelles souffrent du manque d'adaptation et d'apprentissage qui peuvent s'avérer nécessaires dans certaines situations, notamment celles liées aux changements métiers, technologiques ou stratégiques.

Ces quelques problèmes, que nous avons cités, et qui s'avèrent de véritables challenges, peuvent constituer à notre connaissance des perspectives assez prometteuses.

En ce qui nous concerne, nos travaux s'inscrivent principalement dans le developpement d'une architecture d'intégration des applications hétérogènes. Notre contribution concerne les trois (3) premiers aspects. Notre travail s'inscrit dans deux volets complémentaires qui portent respectivement sur l'utilisation des ontologies et des web services [74] [75] [76].

4. 2 Principe de fonctionnement

Une plate-forme EAI fonctionne sur le modèle d'une multiprise. Chaque application possède un connecteur standard (la prise) qui est relié au « bus EAI » (la multiprise). Le connecteur est un exécutable ou une classe Java, installé sur la machine qui héberge l'application. Il traduit les données provenant de l'application dans un format lisible par un courtier de message, et vice versa. Il existe deux types de connecteurs : techniques et applicatifs [87].

Les connecteurs techniques sont reliés aux applications depuis leur base de données, des fichiers plats, etc. Les connecteurs applicatifs interfacent directement leurs API (Application

Programing Interface). Encore propriétaire au début des années deux mille, les connecteurs se standardisent peu à peu autour de technologies telles que les web services (WSDL, Soap, HTTP) ou JCA (J2EE Connector Architecture). Au coeur de la plate-forme, le « bus EAI » traditionnel est constitué d'un courtier de messages (message broker) et d'un MOM (middleware orienté messages). Le courtier de messages applique des transformations sur les messages entrants avant de les renvoyer vers les applications. Il est également capable de router une information sur une file d'attente particulière du MOM. Ainsi, si une application destinataire n'est pas accessible, le MOM stocke les messages entrants et sortants jusqu'à ce qu'ils soient récupérés par leurs destinataires. C'est ce mécanisme de communication asynchrone qui permet de découpler les applications les unes des autres. Dans cette architecture de type « publication et abonnement », chaque application s'abonne à des files de messages sur lesquelles elle peut publier et recevoir des messages, le plus souvent aujourd'hui au format XML. Il existe un autre modèle qui consiste à relier directement deux applications entre elles via un mécanisme de type RPC (Remote Procedure Call ou appel de procédure distante) [12].

4. Services de l'EAI

Les fonctionnalités essentielles d'une solution EAI sont regroupées en services:

- **Le service d'interfaçage :** est assuré par les connecteurs, qui se chargent d'intégrer les API vers un transport standard (fichier, émail,…) vers un progiciel (ERP, CRM,…), et de réaliser la correspondance entre la structure des informations en provenance ou à destination de l'extérieure

- **Le service de queuing :** permet de garantir l'intégrité de l'information tout au long du traitement du flux dans la plate-forme d'intégration. Il remplit le rôle de gestionnaire de messages, réceptionne les évènements produits par les connecteurs et les propage aux composant d'intégrations internes ou aux connecteurs abonnés publier et s'inscrire.

- **Le service de pilotage :** envoyer des commandes de pilotage (arrêt, démarrage,…) et de réceptionner les alertes émises par les composants d'intégrations [93].

5. Types de projets d'intégration

L'EAI intéresse toutes les entreprises qui ont une implantation multi-sites, des matériels différents (du PDA au mainframe), des bases de données multiples, des systèmes multiapplicatifs ou des problématiques B2B. Toute entreprise a besoin d'un EAI à des degrés divers en fonction de sa configuration et de ses systèmes, comme en fonction de ses moyens.

Un projet d'intégration est généralement un projet complexe et d'envergure dans la mesure où il est rarement ponctuel, et de plus il concerne et impacte l'ensemble du SI. Pour cette raison, il doit s'inscrire dans une optique de projet global d'infrastructure du SI. Ce projet peut être réaliser selon une démarche 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.

5. 1 Projets EAI stratégiques

L'aspect stratégique concerne l'intégration des SI de l'entreprise ainsi que ses partenaires B2B. L'intégration stratégique demande beaucoup plus de moyens et de temps, qui n'est pas forcément couronné du succès escompté [92]. L'EAI stratégique implique une prise en compte et une évolution éventuelle du SI global de l'entreprise et de ses process. Ce sont des solutions qui se justifient dans des environnements applicatifs complexes ou des environnements critiques, de grandes entreprises. L'EAI devient alors l'un des piliers de l'architecture du SI.

5. 2 Projets EAI tactiques

Le terme tactique signifie un périmètre limité et parfaitement ciblé. L'intégration tactique nécessite souvent des moyens faibles en ressources, temps et budget, et dispose un retour sur investissement rapide, cependant elle doit être menée comme un morceau du puzzle d'intégration stratégique, puzzle que l'entreprise sera peut être amenée à faire dans l'avenir.

6. Approches d'intégration d'applications

L'analyse de la littérature sur l'EAI permet de mettre en évidence principalement quatre (4) approches :

- L'intégration d'applications par les données,
- L'intégration d'applications par les fonctions (traitements),
- L'intégration d'applications par les interfaces (présentations),
- L'intégration d'applications par les processus.

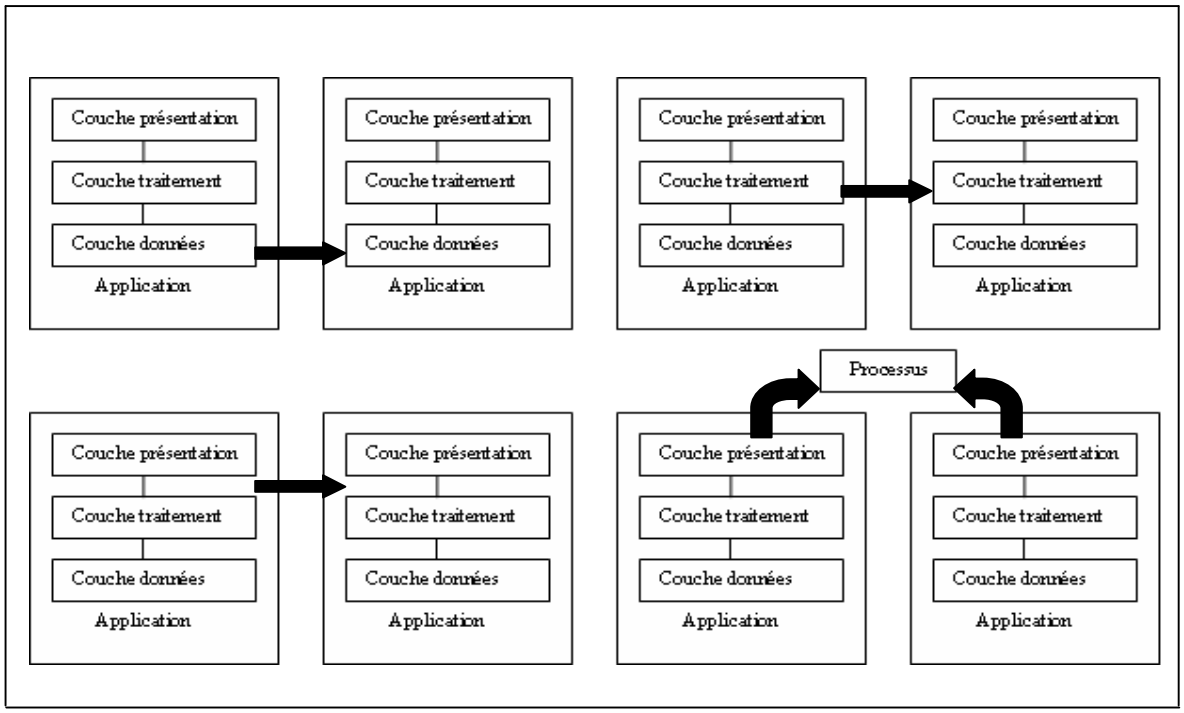


Figure 1.1 : Les différents types d'EAI.

Les trois premières approches approches de base, peuvent être considérées comme le résultat logique de la structuration des applications en couches, tandis que la quatrième approche, qui se décline par la suite en une combinaison des trois approches de base, résulte de la mise en œuvre d'un médiateur (moteur de workflow ou de BPI (Business Process Integration)), et qui permet ainsi d'interconnecter les applications via l'orchestration des processus.

6 1 Intégration d'applications par les données

L'intégration d'applications par les données constitue la manière la plus répandue pour intégrer des applications. Elle se réalise en faisant appel à l'intégration de données. Elle consiste généralement à déplacer les données entre les applications en utilisant très souvent des outils de migration, 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 dans le fait que l'on peut facilement introduire des incohérences au niveau des données en outrepassant les contraintes d'intégrité [23].

Les objectifs sont alors de :

- Fluidifier les échanges entre applications,
- Diminuer les interdépendances entre applications,
- Standardiser les échanges,
- S'affranchir de l'hétérogénéité des plates-formes et architectures.

6.2 Intégration d'applications par les fonctions (traitements)

L'intégration d'applications 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) [24], d'application composites [25], ou encore les récents e-Services [24]. 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 et qui peut être destinée à la fois au B2B et au B2C. 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 [22].

6.3 Intégration d'applications par les interfaces (présentations)

L'intégration d'applications 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, 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 [23].

6.4 Intégration d'applications par les processus

L'intégration d'applications par les processus constitue sans doute la forme la plus évoluée et la plus flexible afin d'interconnecter des applications. Cette approche se base très souvent sur l'utilisation de l'approche BPI qui met en œuvre un moteur d'orchestration (BPI engine) ou éventuellement un moteur de workflow (WFMS - Workflow Management System). Ces moteurs qui constituent des middlewares perfectionnés permettant de relier les applications aux processus [34], permettent alors d'interconnecter dynamiquement les multiples applications en fonction des événements métiers. Ceci permet généralement d'obtenir une plus forte valeur ajoutée [24]. 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.

Notons le fait que ces différentes approches d'EAI ne sont pas exclusives, si bien qu'elles peuvent être mise en œuvre simultanément. C'est à dire que l'on peut intégrer certaines applications par les donnés, d'autres par les traitements, etc. [31].

7. Principaux champs d'intégration d'applications

Tout d'abord, précisons que l'intégration des applications ne traite pas de la communication à l'intérieur d'une application, qui est du ressort du génie logiciel, mais elle traite plutôt des échanges entre les applications. Ainsi, le champ d'intervention de l'intégration des applications est le domaine d'applications. Et en fonction du périmètre couvert par les échanges interapplications, il est possible de distinguer essentiellement deux champs d'intégration des applications: l'intégration intra-entreprise et l'intégration extra-entreprise (figure 1. 2) [31] [94].

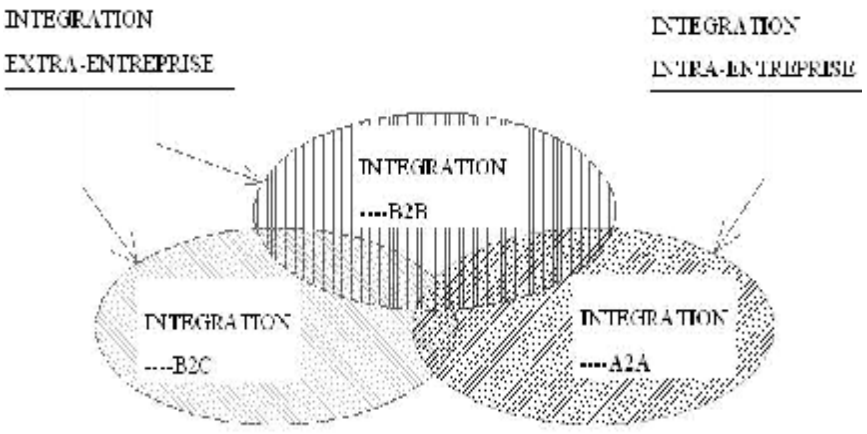


Figure 1. 2 : Champs d'intégration d'applications.

L'intégration intra-entreprise, qui est le plus souvent désignée par le sigle A2A (Application to Application), ou le sigle EAI, est une intégration qui se limite aux applications internes à l'entreprise. Tandis que l'intégration extra-entreprise étend la précédente pour inclure l'intégration des applications des partenaires et des fournisseurs (B2B - Business to Business intégration. Pour simplifier, on suppose que le B2B inclut le B2A - Business to Administration), ainsi que l'intégration des clients (B2C - Business to Customer intégration), ce qui permet de s'insérer dans le cadre du commerce électronique.

8. Typologies des applications intégrées

L'intégration des applications ne peut se faire qu'en examinant les différents types d'applications à intégrer (batch, transactionnelles, client-serveur, Web, etc.) et leurs caractéristiques spécifiques

L'examen de chaque application à intégrer permettra de déterminer les contraintes découlant des caractéristiques propres aux différents types d'applications avec lesquelles on va lui demander de communiquer. Par exemple, une application Web mise en relation avec une application "batch", dont le traitement s'effectue de manière différée, ne pourra se faire que sous certaines conditions compte tenu des différences de conception de ces applications spécifiques.

Dans l'analyse des applications à intégrer, il faut tenir compte:

- des formats des données ou des événements,
- du volume de ces données et événements,
- de leur capacité d'échange avec d'autres applications,
- du rythme de fonctionnement de ces applications.

8. 1 Les applications batch

Ce sont les plus anciennes applications conçues avec une philosophie d'application monolithe lourde. Elles traitent en différé un ensemble d'événements groupés dans des fichiers ou dans des bases de données. La périodicité est particulière à chaque application.

Les volumes supportés par de telles applications peuvent être importants: un lot peut contenir des millions d'événements. Par ailleurs, les formats des données et des événements sont généralement assez simples (par exemple des formats "plats" développés en langage Cobol). Le calcul des salaires dans une entreprise est un bon exemple d'application batch.

L'intégration de ce type d'applications doit obligatoirement se faire via un fichier (par exemple au format XML) ou via la base des données de cette application.

6. 2 Les applications transactionnelles

Ces applications traitent les événements les uns après les autres. La plupart de ces applications assurent des interfaces avec les utilisateurs.

Le format des données et des événements dépend des technologies utilisées. En général, ce type d'applications traite des formats un peu plus complexes que les applications batch. En ce qui concerne les volumes traités, ces applications peuvent absorber un grand nombre d'événements. Cependant, par événement, le volume des données est le plus souvent petit ou moyen. Les seules solutions de communication avec ce type d'applications sont la base de données et l'interface utilisateur.

7. 3 Les applications client-serveur

Ces applications sont considérées comme une forme plus évoluée des applications transactionnelles**.** Tout en gardant les caractéristiques des applications transactionnelles, on y trouve en plus:

- La capacité à traiter des structures de données complexes (ex : bases de données, XML),
- Les caractéristiques du modèle client-serveur :
  - Le client contacte le serveur par envoie de demandes,
  - Le serveur traite les demandes et répond au client,
  - Plusieurs clients accèdent en même temps au serveur,
- La communication s'effectue via :
  - Des bases de données ou fichiers,
  - Des technologies Internet : HTTP, SMTP, SOAP.

Il existe d'autres types d'applications récentes plus complexes et ouvertes, les applications Web et les progiciels [9].

7. 4 Les applications fil de l'eau

Ces applications sont apparues assez recemment. Elles se basent sur les échanges de messages inter-applications : MOM (Message Oriented Middleware) et fonctionnent d'une façon hybride ; en batch ou en transactionnel par traitement des évènements selon le rythme de leur apparition. Les stuctures de données traitées sont complexes et les volumes traités peuvent être très importants. Le dialogue entre ces applications est assuré via les files d'attente du MOM.

8. 5 Les progiciels

Logiciels spécialisés métier, couvrent une large palette de fonctionnalités nécessaires dans un SI. Parmi les progiciels les plus répandus, nous citons les CRM (Customer Relationship Management), les SCM (Supply Chain Management) et les ERP (Enterprise Resource Planning). Ces progiciels offrent des fonctions de plus en plus complexes (gestion des stocks, comptabilité analytiques, etc).

La connectivité est leur point faible majeur, donnant parfois à l'entreprise la sensation d'être prisonnière du progiciel. Elle est désormais un facteur de différenciation entre les constructeurs. L'échange des informations avec ces progiciels est assuré à travers de fichiers, files d'attente, bases de données, API, protocole http, etc. Leur integration aux systèmes de l'entreprise est largement favorisée. La table 1 ci-dessous, nous avons récapitulé l'essentiel des caractéristiques de chaque type d'application ainsi que les approches qui peuvent leur être appliquées.

Remarquons que la dernière ligne de la table est assez floue et ambiguë, ceci peut être expliqué par le fait que les progiciels sont tellement diversifiés.

Type d'application	Caractéristiques	Approches d'intégration
Application batch	Rythme de fonctionnement : par lot. Format des évènements et des données traitées : simple (fichier). Volume des données : très grand. Capacité d'échange avec d'autres applications : faible. Type d'architecture de l'application : 1-tiers.	Donnée
Application transactionnelle	Rythme de fonctionnement : par unité / temps réel. Format des évènements et des données traitées : complexe (base de données, XML). Volume des données : grand. Capacité d'échange avec d'autres applications : faible. Type d'architecture de l'application : 2-tiers.	Donnée Interface
Application client/serveur	Rythme de fonctionnement : par unité / temps réel. Format des évènements et des données traitées : complexe (base de données, XML). Volume des données : grand. Capacité d'échange avec d'autres applications : faible. Type d'architecture de l'application : 2-tiers.	Donnée Interface
Application Web	Rythme de fonctionnement : par unité / temps réel. Format des évènements et des données traitées : complexe (base de données, XML). Volume des données : très grand. Capacité d'échange avec d'autres applications : grande. Type d'architecture de l'application : 3-tiers.	Donnée Interface Traitement
Application basée MOM	Rythme de fonctionnement : par unité / temps réel. Format des évènements et des données traitées : complexe (base de données, XML). Volume des données : très grand. Capacité d'échange avec d'autres applications : grande. Type d'architecture de l'application : 3-tiers.	Donnée Interface Traitement
Application progiciel	?	?

Table 1. 1 : Types des applications à intégrer.

9. Topologies d'architectures d'intégrations

En générale une architecture décrit la structure du système et ses composants ainsi leurs liens. Nous distinguons les types d'architectures d'intégration suivantes :

9. 1 Architecture point à point

L'intégration point à point est généralement la plus utilisée, peu coûteuse et rapide à mettre en place [14]. Elle consiste à développer des solutions d'intégrations spécifiques capables de répondre rapidement au besoin d'intégration, ces solutions sont généralement basées sur des outils client serveur et sur divers middlewares de communication [15].

Les inconvénients de ce type d'architecture :

- Le couplage fort entre les applications.
- Le changement dans une application peut provoquer des disfonctionnement avec les autres applications qui l'intègre [14].
- Le nombre des liens point à point augmente de manière exponentielle lorsque une nouvelle application être intégrée. L'administration et la maintenance deviennent difficiles. En plus, les risques d'erreurs se multiplient [15].

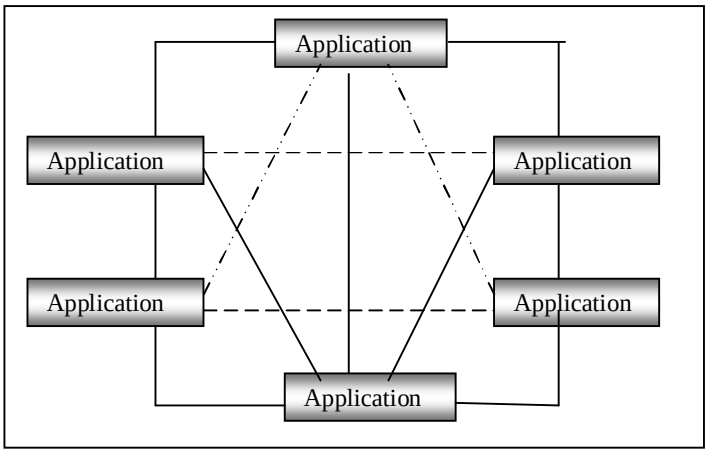


Figure 1.3 : Intégration point à point

9. 2 Architecture d'intégration par broker

Cette architecture consiste à utiliser un négociateur central qui gère les interactions entre les applications. Le nœud central apporte une notion de découplage très intéressante.

L'utilisation d'un format intermédiaire de communication sert à éviter de faire intégrer **n** fois une application avec **n** autres applications, mais simplement une fois sur le nœud central qui assure ensuite la communication avec les autres application [15].

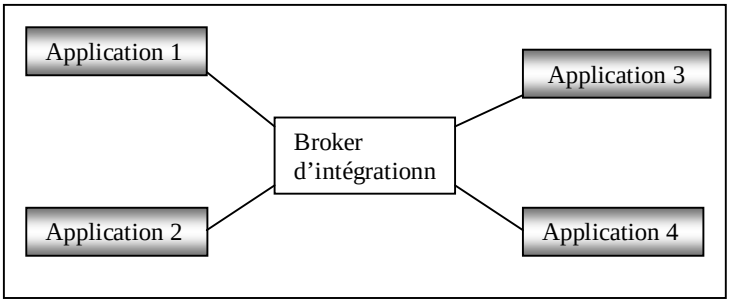


Figure 1. 4: Intégration broker.

9. 3 Architecture d'intégration par bus

Cette architecture repose généralement sur l'utilisation d'un bus de communication. La plus part des applications peuvent assez aisément se brancher sur ce bus de communication purement logiciels par l'intermédiaire d'interfaces génériques [4]. Elle suit le principe de

publier et s'inscrire décentralisé, chaque application s'abonne à une diffusion broadcast ou multicast. Les applications déposent les messages sur les bus et les applications abonnées les récupèrent.

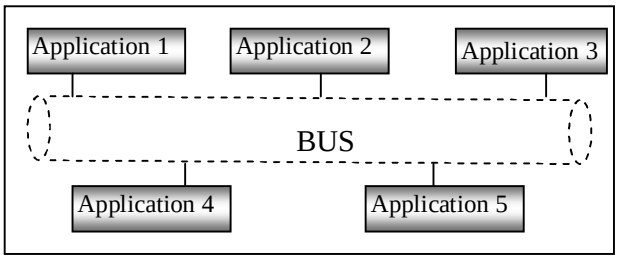


Figure 1. 5 : Intégration par bus.

9. 4 Architecture d'intégration Hub and Spoke

L'architecture Hub and Spoke (littéralement moyeu et rayon) propose une architecture EAI centralisée dans laquelle des branches sont reliées via des connecteurs à un bus unique dont le rôle est d'aiguiller les communications. On peut comparer le Hub and Spoke à une architecture en étoile dans le domaine des LAN: le hub (concentrateur) représente le nœud central et les spoke (rayons) permettent de relier les applications au nœud central.

Ce modèle architectural a pour avantages :

- Une intrusivité minimale dans les applications,
- Une centralisation des règles de routage et de transformations (au sein du hub),
- Une administration simplifiée de l'architecture.

Les inconvénients majeurs de cette architecture sont :

- Le point de faiblesse qu'introduit le hub (s'il n'est plus disponible, l'architecture d'intégration ne l'est plus non plus),
- La gestion de la montée en charge qui ne peut se faire que par ajout de hub supplémentaires [4].

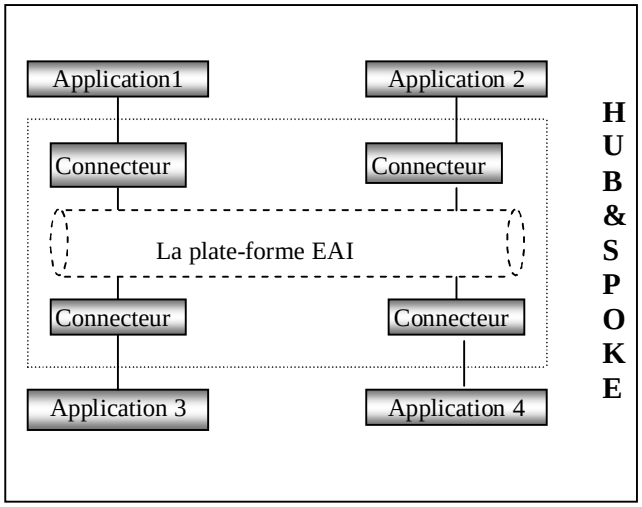


Figure 1 .6 : Intégration hub & spoke.

9. 5 Architecture Network Centric

L'architecture Network Centric propose une architecture EAI décentralisée. L'approche Network Centric reprend le paradigme « The network is the computer » et est axée sur les réseaux et la connectivité. Un tel système centré réseau est constitué de plusieurs machines, connectées entre elles. La charge du système se répartit ainsi sur l'ensemble des processeurs du réseau.

Ce modèle architectural a pour avantages :

- Une distribution de la charge sur les différents nœuds,
- Une distribution du référentiel sur les différents nœuds,
- Une capacité de support de l'augmentation de la charge grâce à l'absence de goulot d'étrangement central.

Les inconvénients de cette solution sont :

- Une administration difficile du fait de la décentralisation des composants,
- Une intrusivité dans les applications plus importante que pour le mode Hub and Spoke car l'intelligence de l'EAI se déporte sur chaque nœud [4].

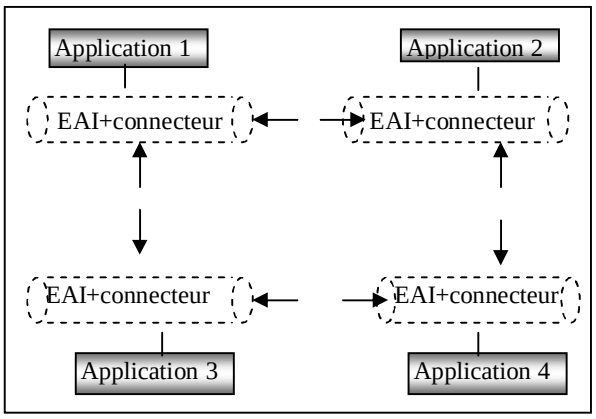


Figure 1. 7 : Intégration network centric.

10. Modèles de l'EAI

Les modèles de l'architecture sont un allié important de la technologie d'information professionnel pour éviter la complexité de l'intégration et la confusion.

Les modèles de l'architecture de l'EAI fournissent une vue architecturale d'EAI et constituent de solides fondations aux problèmes de l'intégration, ils permettent d'aider les techniciens et les experts à déterminer les meilleures approches et outils permettant de trouver une solution au problème d'intégration, les modèles consistent en général, d'avoir une certaine flexibilité et surtout une compréhension globale du système.

Un modèle d'architecture exprime un schéma pour l'organisation et la structuration de systèmes logiciels. Il fournit un ensemble de sous-systèmes prédéfinis en spécifiant leurs

relations et incluant les règles et les principes permettant d'organiser les relations entre eux. Dans le domaine d'EAI, cinq modèles d'architectures ont été recenses [16].

10. 1 Modèle d'adaptateur de l'intégration

Ce modèle permet, comme son nom l'indique, d'adapter l'interface d'une application dans un format attendu de l'application cliente (celle qui va invoquer les services de l'interface). Ce modèle permet de façon sous-jacente, de fournir une description d'interface pouvant être réutilisée par plusieurs applications clientes.

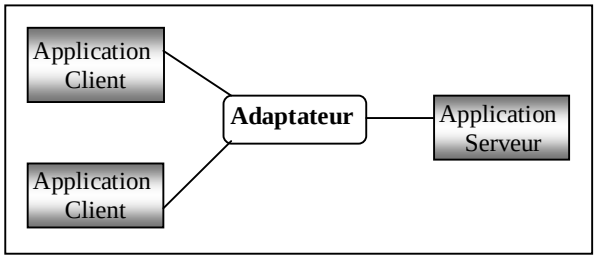


Figure 1. 8 : Principe de l'adaptateur.

10. 2 Modèle du messager de l'intégration

Ce modèle décrit une approche architecturale à intégrer des applications où c'est inutile ou irréaliste à découplé la logique de l'interaction de l'application des applications elles mêmes. L'avantage clé est que les dépendances en terme communication entre applications sont minimes. Cela offre une approche d'intégration plus flexible.

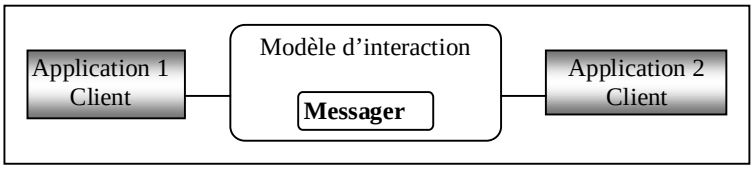


Figure 1. 9 : Principe du messager.

11. 3 Modèle de la façade de l'intégration

Ce modèle décrit une approche architecturale à intégrer des applications clients avec les applications serveurs, il permet la réutilisation et la flexibilité des applications et des services permettant l'intégration.

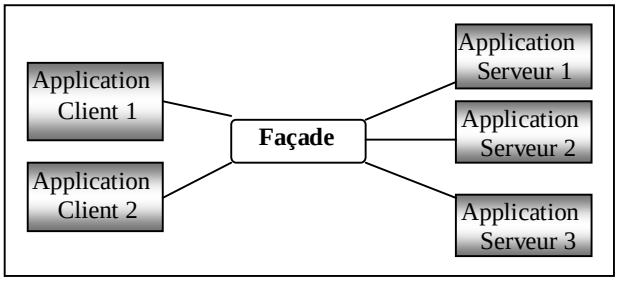


Figure 1. 10 : Principe de la façade.

10. 4 Modèle du médiateur de l'intégration

Ce modèle décrit une approche architecturale à intégrer des applications où la logique de l'interaction de l'application est encapsulée et découplée des applications participantes, les bénéfices de cette approche sont :

- La maintenance est simplifiée du fait que la logique d'interaction est centralisée au niveau du composant médiateur.
- La réutilisation peut être correctement construit autour de la logique d'intégration.
- Les dépendances et les différences avec d'autre application sont minimisées.

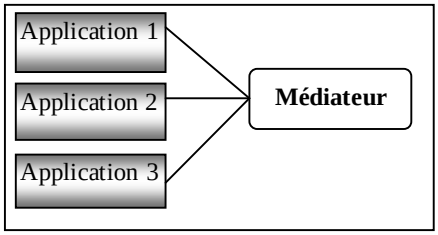


Figure 1. 11: Principe du médiateur.

10. 5 Modèle de l'automate du processus de l'intégration

Ce modèle décrit une approche architecturale pour minimiser les dépendances entre la logique du contrôle de l'automatisation du processus et les systèmes des technologies de l'information. En effet, ce concept d'activité fait que l'ensemble des interactions entre systèmes est caché du contrôleur de processus.

Les avantages d'une telle approche sont :

- Des moteurs de processus différents peuvent être implémentés de façon relativement économique.
- L'architecture fournit une analyse des processus qui n'a pas son équivalent (« goulots d'étranglements », statistiques, utilisation des ressources, etc.).
- L'architecture offre une flexibilité dans la redéfinition et le déploiement d'application centrée processus.
- L'implémentation des applications centrées processus est alignée sur la vision du gestionnaire de projet (manager), ainsi, l'écart entre la vision du manager et l'implémentation informatique de cette vision considérablement entre les besoins et les solutions informatiques censées satisfaire ces besoins.
- Comme les modèles de façade et du médiateur, la logique d'intégration est encapsulée et peut être partagée.

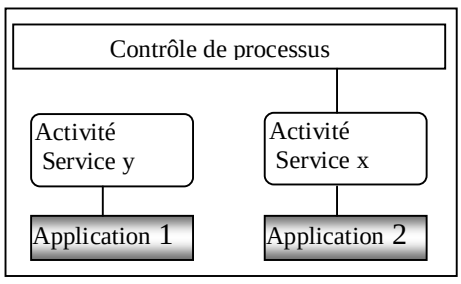


Figure 1. 12. Principe du moteur de processus.

11. Technologies d'intégration d'applications

Différentes technologies ont tenté d'apporter des réponses aux besoins d'EAI et ce en cherchant à proposer des modèles plus ou moins pertinents. Tout d'abord les progiciels intégrés ERP (Enterprise Resource Planning), CRM (Customer Relationship Management) et SCM (Supply Chain Management) offrent des solutions centralisées, souvent partielles et limitées à leur périmètre de déploiement.

Ainsi, d'autres technologies peuvent être mise en œuvre dans le cadre de l'EAI. Elles reposent toutes sur un concept clé de « middleware ». Un middleware est un ensemble des technologies permettant de faire communiquer des applications en fournissant des services de transport et de routage [5] [9] [27]. Signalons que, nous classons les middleware en trois catégories: middlewares orientés serveurs d'intégration, middlewares orientés portails et middlewares orientés services.

Type de middleware	Description	Type d'intégration	Exemples
DOM (Database-oriented middleware)	Intégration en utilisant les bases de données	A2A	ODBC, JDBC, EDI, …
WOM (Warehouse-oriented middleware)	Intégration en utilisant des entrepôts de données	A2A	ETL,…
ROM (Remote procedure call- oriented middleware)	Intégration par appels à distance	A2A	RPC, …
MOM (Message oriented middleware)	Intégration par échanges de messages	A2A	MSMQ, MQSeries, …
OOM (Object-oriented middleware)	Intégration par distribution d'objets	A2A	CORBA, DCOM, OLE, …
COM (Component-oriented middleware)	Intégration par utilisation de composants	A2A	Java RMI, …
POM (Portal-oriented middleware)	Intégration par utilisation de portails	B2B	Portail B2C, B2B, …
TOM (Transaction-oriented middleware)	Intégration via la gestion de transactions par des serveurs d'applications	A2A	Serveurs d'applications
SOM (Service-oriented middleware)	Intégration en utilisant la notion de service	B2B	ESB, .NET, Websphere, …
IOM (Integration-server-oriented middleware)	Intégration en utilisant un serveur d'intégration	A2A	Hub-and-spoke (Vitria BW, …)

Table 1. 2 : Principales technologies d'intégration.

L'analyse du tableau 1.2 permet de dégager trois solutions émergentes, et qui sont complémentaires quant à leur couverture du périmètre d'intégration des applications. Il s'agit des middlewares orientés serveurs d'intégration, portails et enfin services.

Signalons, tout de même, que très souvent l'intégration des applications se présente sous forme d'un problème fort complexe qui peut nécessiter la mise en œuvre simultanée de plusieurs technologies. A titre d'exemple, on peut mettre en œuvre à la fois une technologie IOM pour l'intégration intra entreprise, POM pour le B2C et SOM pour le B2B.

11. 1 Middleware orientes serveurs d'intégration

Ces middlewares constituent à l'heure actuelle la forme la plus aboutie des technologies d'intégration intra entreprise. Bien qu'ils reposent très souvent sur des produits propriétaires, on constate une certaine convergence en matière de fonctionnalités proposées. Aussi, les serveurs dits de première génération offrent des fonctionnalités de base: connectivité (connecteurs), routage, transformation, ainsi qu'un certain nombre de services annexes mais fondamentaux tels que la sécurité, le management et la qualité de services. Tandis que les outils dits de deuxième génération tentent d'inclure, en sus, des fonctionnalités avancées, liées notamment à la gestion des processus et parfois même à l'intégration des partenaires externes. Certains IOM tentent même d'inclure des moteurs de workflow, permettant ainsi d'intégrer aussi bien l'aspect informationnel qu'organisationnel [169].

11. 2. Middleware orientes portails

Ces middlewares permettent une intégration légère d'applications, en offrant aux utilisateurs un seul point d'entrée leur permettant d'accéder, à l'aide d'une interface unique, à un ensemble d'applications et/ou de sources de données [31] [168]. Ce type de middleware a une finalité plus orientée front-end, alors que l'IOM est orientée beaucoup plus vers le back-end. Mais en règle générale, ces deux solutions sont complémentaires, et on utilise généralement les portails pour l'intégration B2C.

11. 3. Middleware orientes services

Ce sont principalement des middlewares qui se basent sur le concept d'e-Service et qui référencent généralement le standard XML. Les systèmes construits autour de ce type de middleware reposent généralement sur une panoplie de plus en plus enrichie de technologies permettant principalement de développer (WSDL -Web Services Description Language, etc.), publier (UDDI - Universal Discovery, Description and Invento-ry), de communiquer (XML Schéma, SOAP - Simple Object Access Protocol, etc.) et d'orchestrer des services métiers (WSFL - Web Services Flow Language, WS-Security, BPEL4WS - Business Process Execution Language for Web Service, etc.) [9] [31] [167].

12. Domaines de recherches

On peut distinguer quatre (4) axes de recherche qui déterminent les travaux de recherche proposés dans ce cadre :

12. 1 EAI et les ontologies

Dans le domaine de médiation des sources de données, l'intégration des ontologies joue un rôle majeur. Les connexions entre les systèmes de médiation autorisent la recherche des données dans des sources non directement connectées aux sources du serveur interrogé. Chaque système de médiation possède sa propre ontologie. Pour connecter ces différents systèmes, l'intégration des ontologies permet de construire une ontologie globale et facilite l'accès à toutes les sources de données hétérogènes.

De plus, l'utilisation des ontologies peut apporter aux solutions actuelles de l'intégration des applications, plus d'efficacité, de flexibilité et d'adaptabilité. En effet, le recour aux ontologies peut rendre les échanges inter-applications plus intelligibles et donc plus

performants. Ce qui pourrait éventuellement constutuer un pas considérable dans la course vers l'ultime objectif qui est l'intégration plug-and-play.

12. 2 EAI et les Web Services

Les Web Services peuvent-ils remplacer les solutions d'EAI ?

Il est indéniable que les services web vont jouer un rôle majeur dans l'intégration orientée méthode/fonction, celle-ci représentant l'une des formes les plus courantes A2A et B2B. Les web services peuvent être utilisés pour l'intégration d'applications orientée méthode/fonction.

L'intégration orientée méthode ou fonction est intrinsèquement synchrone: elle repose sur des interactions requête/réponse entre le client (programme demandeur) et le serveur (programme répondant).

La technologie des services web fournit un mécanisme standard permettant aux applications de publier et souscrire à des services logiciels sur un intranet ou sur internet. Les applications clientes (utilisateurs de services web) peuvent localiser ces services publiés par des applications serveur (prestataires de services web) à l'aide du standard UDDI (Universal Discovery, Description, and Integration) ; elles peuvent déterminer la définition de l'interface du service à l'aide du langage WSDL (Web Services Description Language) et échanger des données par des documents XML (Extensible Markup Language) reposant sur le protocole SOAP sur des protocoles universels tels que HTTP, FTP et SMTP.
L'application cliente de services web a l'impression d'appeler une méthode localement, alors qu'en réalité, l'appel de méthode est converti en XML (basé sur SOAP) et transmis via le réseau à l'application prestataire de services web. Le prestataire renvoie la réponse à l'appel de méthode sous forme d'un document XML (basé sur SOAP).
Dans la mesure où les services web sont accessibles par le biais d'adresses URL, du HTTP et du XML, les applications s'exécutant sur n'importe quelle plate-forme et dans n'importe quel langage peuvent accéder aux services web XML.

Avec Microsoft .NET, l'intégration orientée méthode/fonction peut être réalisée comme suit:

- L'application serveur (prestataire de services web) peut implémenter le service web dans n'importe lequel des langages .NET, tels que C#, VB.NET ou Managed C++ (extensions managées pour C++), qui peut être compilé dans le langage MSIL (Microsoft Intermediate Language) puis exécuté sur une machine virtuelle appelée Common Language Runtime (CLR).
- Le service web peut être déployé sur une plate-forme .NET.
- L'application cliente (utilisateur de services web) peut mettre en oeuvre le mécanisme d'écoute de services web en utilisant MSXML ou ASP.NET et appeler le service web à l'aide d'un appel de méthode.

Avec JAX-RPC, l'intégration orientée méthode/fonction peut être réalisée comme suit:

- Définir et implémenter un service web basé sur JAX-RPC. L'implémentation du service web peut se faire dans une application Java autonome ou un composant EJB (Enterprise

Java Bean). L'API JAX-RPC peut être utilisée pour créer des encapsuleurs (wrappers) fondés sur SOAP qui confirment à l'interface WSDL l'existence des classes Java ou des EJB.

- Déployer le service web sur un système d'exécution JAX-RPC serveur. Le déploiement est régi par l'implémentation du service web. Par exemple, si l'implémentation est assurée par EJB, le déploiement se fera dans un conteneur EJB.

- Appeler le service web à partir de l'application cliente sur le port décrit dans le document WSDL. Pour l'application cliente, l'appel du service web semble être un appel de méthode local.

12. 3 EAI et Urbanisation

L'urbanisation des systèmes, couplée à l'esprit très pragmatique d'un projet d'EAI, devient un moyen très pratique et presque simple de découvrir et rationalisés les processus métier. La démarche d'urbanisation des SI prend naissance dans ces constat : « des systèmes informatiques des entreprises ont été construits par morceaux, en réponse à des besoins précis, mais sans plan de l'ensemble. Ils arrivent à un stade où tous les composants sont tellement imbriqués que, à moins d'y consacrer de gros budgets, ils ne peut plus évoluer ». Il faut distinguer les données des traitements et dissocier le SI en fonctions indépendamment les une des autres. Il s'agit d'une démarche objet appliquée à l'ensemble du SI.

Les plates-formes EAI combinent en effet plusieurs atouts :

- La capacité à se connecter aux sources de données de l'entreprise et à transformer les flux de données pour permettre à plusieurs applications hétérogènes de communiquer ensemble.
- La capacité à invoquer des services métier réutilisables développés sous forme de composants.
- La capacité à détecter les événements métier survenus dans les applications, et les utiliser pour déclencher les processus qui orchestreront les échanges inter-applicatifs.
- La capacité à inclure des outils BPM pour prendre en charge des processus métier complexes.
- Les points communs entre l'urbanisation et l'EAI sont nombreux :
- Les deux besoins sont nés du contact que les SI dans leur globalité sont extrêmement complexes.
- Les deux démarches sont des démarches transversales de réflexion entre les activités et entité de l'entreprise. Elles sont flexibles dans la mesure où elles peuvent être initiées depuis une problématique précise pour s'étendre ensuite à un ensemble de services ou à tout l'entreprise [3].
- Les projets d'EAI et d'urbanisation sont souvent considérés comme des préalables indispensables pour ouvrir un chantier e-buisiness.
- Les projets d'EAI et d'urbanisation sont avant tout une affaire de grands comptes aux prises avec un existant informatiques complexe [10].

Les intérêts de l'EAI par rapport à l'urbanisation sont aussi nombreux :

- L'EAI facilite l'urbanisation du SI [18].
- L'EAI offre une solution en l'état de l'art et d'implémenter le middleware de communication dont a besoin l'urbanisation.
- Pour prendre terminologie de l'urbanisation du SI qui signifie le découpage du SI en zones, quartiers puis en blocs, les applications contenues dans un bloc fonctionnel

communiquent avec les applications du autre bloc en passant par des voies de communication fournies par l'infrastructure EAI.

- EAI est donc un moyen permettant de passer d'un SI non urbanisé basé sur un modèle point à point, à un SI urbanisé basé sur un modèle en étoile [4].

12. 4 EAI et MDA

Le Model Driven Architecture (MDA) [161] est une démarche de développement proposée par l'OMG [162]. Elle permet de séparer les spécifications fonctionnelles d'un système des spécifications de son implémentation sur une plate-forme donnée. A cette fin, le MDA définit une architecture de spécifications structurée en modèles indépendants des plates-formes (PIM) et en modèles spécifiques (PSM) [164].

L'approche MDA permet de réaliser le même modèle sur plusieurs plates-formes grâce à des projections standardisées. Elle permet aux applications d'interopérer en reliant leurs modèles et supporte l'évolution des plates-formes et des techniques. La mise en oeuvre du MDA est entièrement basée sur les modèles et leurs transformations.

Grâce à son architecture de méta-modèlisation, le MDA devrait apporter un important gain de productivité en permettant de capitaliser les efforts à tous les niveaux. Il n'existe pas de plate-forme universelle. Celle-ci peut dépendre du matériel, du réseau, des équipes de développement ou de ce qui est déjà en place au niveau de l'entreprise. De ce fait, le meilleur environnement de développement doit permettre de choisir la plate-forme cible après avoir modélisé l'application, et passer de l'une à l'autre relativement facilement. De plus, il doit être possible d'interopérer avec toutes les plates-formes quelle que soit celle qui a été choisie, afin de pouvoir tirer parti des avantages qu'elles offrent. L'MDA traite le problème d'hétérogeneité au niveau des plate-formes, des langages de programmation, des systèmes d'exploitation mais aussi des middlewares. De même, il est possible de faire interopérer différentes applications en exprimant leurs relations au niveau PIM. Aumoment 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. C'est l'ambition du MDA.

Plusieurs profils ont été developpé par l'OMG :

- Le profil EDOC (Enterprise Distributed Object Computing) [160] qui doit permettre de définir un système d'information d'entreprise à base de composants. C'est grâce à ce profil que les PIM composants d'entreprise doivent être définis.
- Le profil EAI (Enterprise Application Integration) [159] pour les systèmes faiblement couplés, il permet de représenter les communications asynchrones et les envois de messages, quelque soit la plate-forme (PIM).
- Le profil Scheduling [163] pour l'ordonnancement, typiquement utilisé pour la modélisation des applications temps réels, quelque soit la plate-forme (PIM).

En ce qui nous concerne, nos travaux s'inscrivent principalement dans les deux (2) premiers axes de recherches qui portent respectivement sur l'utilisation des ontologies et des web services dans les approches d'intégration.

Les solutions actuelles d'intégration des applications ne résolvent pas le problème lié à la sémantique, dans la mesure où les mécanismes proposés sont basés au mieux sur des mappings ou des transformations syntaxiques. L'aspect sémantique est motivé par le fait que

l'ajout d'une couche intelligente aux solutions actuelles d' d'intégration des applications devrait apporter plus d'efficacité, de flexibilité, et d'adaptabilité. En effet, le recours à la sémantique, en introduisant une couche basée sur des connaissances, peut rendre les échanges interapplications plus performants, ce qui pourrait éventuellement constituer un pas considérable vers l'ultime objectif qui est l'intégration plug-and-play.

L'utilisation des web services peut être motivée par les caractères d'universalité et d'ubiquité qu'ils fournissent et qui confèrent ainsi plus d'ouverture et plus de distribution aux architectures d'intégration. Comparée aux autres technologies traditionnelles d'intégration des applications, les web services devraient apporter plus de fléxibilité, de simplicité, de neutralité, d'efficacité et d'économie.

13. Conclusion

Ce chapitre a présenté un état de l'art sur l'EAI. Il a permis d'exposer les approches ainsi que les principales technologies. Nous souhaitons, à travers cette brève description, avoir montré à quel point l'EAI peut constituer une pièce maîtresse de l'intégration et de l'évolution des SI de l'entreprise, dans la mesure où elle peut constituer un moyen efficace pour rendre ces derniers plus interoperables et plus agiles.

Devant les limitations que présente l'EAI actuelle, nous avons évoqué quelques orientations qui nous semblent pertinentes. Et nous avons privilégié, dans le cadre de nos travaux, l'orientation basée sur la sémantique et les web services.

L e prochain chapitre donne un aperçu sur le domaine des ontologies et les approches d'integration.

CHAPITRE 2

Les ontologies : état de l'art et intégration

« The Semantic Web is an extension of the current web in which information is given Well-defined meaning, better enabling computers and people to work in cooperation. »
Tim Berners-Lee

1. Introduction

Nées des besoins de représentation des connaissances, les ontologies sont à l'heure actuelle au cœur des travaux menés en Ingénierie des Connaissances (IC). Visant à établir des représentations à travers lesquelles les machines puissent manipuler la sémantique des informations, la construction des ontologies demande à la fois une étude des connaissances humaines et la définition de langages de représentation, ainsi que la réalisation de systèmes pour les manipuler. Les ontologies participent donc pleinement aux dimensions scientifique et technique de l'Intelligence Artificielle (IA) : scientifique comme étude des connaissances humaines et plus largement de l'esprit humain, ce qui rattache l'IA aux sciences humaines, et technique comme création d'artefacts possédant certaines propriétés et capacités en vue d'un certain usage.

Au fur et à mesure des expérimentations, des méthodologies de construction d'ontologies et des outils de développement adéquats sont apparues. Émergeant des pratiques artisanales initiales, une véritable ingénierie s'est constituée autour des ontologies, ayant pour but leur construction mais plus largement leur gestion tout au long d'un cycle de vie. Les ontologies apparaissent ainsi comme des composants logiciels s'insérant dans les systèmes d'information en leur apportant une dimension sémantique qui leur faisait défaut jusqu'ici.

Dans ce chapitre, nous décrivons la notion d'ontologie, les méthodologies de developpement (Tove, Enterprise et Methontology), les formalismes de représentation (les frames, les graphes conceptuels et les logiques de description) et les langages de spécification (KIF, KL-ONE, RDF, RRD(S), OIL, DAML-OIL et OWL). Ensuite, nous mentionnons les approches d'intégration des ontologies tout en citant le problème d'hétérogénèité.

2. Définitions d'une ontologie

Le concept d'ontologie existe depuis très longtemps, notamment en philosophie. L'ontologie étymologiquement « ontos » (être) et « logos » (science, langue), s'interesse à la science de l'être en tant qu'être, indépendamment de ses manifestations particulières. Elle relève donc de la métaphysique (le terme lui-même apparaît tardivement en 1692, emprunté au latin scientifique ontologia 1646).

" Since the science which is about God calls itself Theosophy or Theology, it would seem fitting to call Ontosophy or Ontology that science which does not deal with this and that being, as distinct from the others owing to its special name or properties, but with being in general" [Clauberg 1647]

En informatique et d'après l'encyclopédie Wikipédia, « une ontologie est un ensemble structuré de concepts. Les concepts sont organisés dans un graphe dont les relations peuvent être :

- des relations sémantiques ;
- des relations de composition et d'héritage (au sens objet). »

L'objectif premier d'une ontologie est de modéliser un ensemble de connaissances dans un domaine donné.

« Les ontologies informatiques sont des outils qui permettent précisément de représenter un corpus de connaissances sous une forme utilisable par une machine. » [165].

De nombreuses définitions ont été proposées pour donner le sens exact au terme ontologie. Ce mot est utilisé dans différents contextes avec des définitions variées, mais aucune de ces définitions ne s'est explicitement imposée. Nous avons recensé les définitions suivantes :

Neches [172], a définit une ontologie comme suit :

« Une ontologie définit les termes et les relations de base du vocabulaire d'un domaine ainsi que les règles qui indiquent comment combiner les termes et les relations de façon à pouvoir étendre le vocabulaire » (1991).

Dans le cadre d'IA, neches montre que l'élaboration d'une ontologie s'effectue par des directives relativement floues. Ce qui signifie qu'on peut utiliser des règles pour combiner les termes et les relations entre eux pour déduire d'autres termes et relations.

Selon Gruber, « Une ontologie est une spécification explicite et formelle d'une conceptualisation partagée» (1993).

« Ontologie » est un terme emprunté à la philosophie qui implique une branche de la philosophie qui traite la nature et l'organisation de la réalité. Plusieurs définitions d'ontologies sont données dans la dernière décade, mais celle qui caractérise l'essentiel d'une ontologie est fondée sur la définition reliée dans [48] : « Une ontologie est une **spécification formelle**, et **explicite** d'une c**onceptualisation partagée** ». Conceptualisation réfère à un modèle abstrait de certains phénomènes dans le monde qui identifie les concepts appropriés de ce phénomène. Explicite signifie que le type de concepts utilisés et les contraintes sur leur utilisation doivent être explicitement définis. Formelle réfère au fait qu'une ontologie doit être compréhensible par la machine, c'est-à-dire cette dernière soit capable d'interpréter la sémantique de l'information fournie. Partagée indique que l'ontologie supporte la connaissance consensuelle, et elle n'est pas restreinte à certains individus mais accepté par un groupe [97].

Grüninger [46], défini une ontologie selon les composants de base, les entités et les propriété. Cette permet de distinguer une ontologie qui est un ensemble définie de concepts et leurs relations, d'un modèle qui est un ensemble d'instance de ces concepts.

« Une ontologie est une description formelle d'entités et leurs propriètés, relations, contraintes et comportement» (1995).

En 1997, Guarino [173] accentue l'ambiguité du rme conceptualisation. La spécification des ontologies est partielle car une conceptualisation ne peut toujours être entièrement formalisée dans un cadre logique, du fait d'ambiguités ou qu'aucune représentation de leur sémantique n'existe dans le langage de représentation choisi.

« Les ontologies sont des spécifications partielles et formelles d'une conceptualisation commune » (1997).

D'autres définition se basent sur la notion e partage des connaisances liés à un domaine :

- " Une ontologie est une description concise et non ambiguë de principales entités appropriées d'un domaine d'application et de leurs relations potentielles entre eux." (S. Schulze-Kremer).

- " Une ontologie est un vocabulaire commun pour les chercheurs qui doivent partager les informations dans un domaine (Noy et al.)

- "Une ontologie est une spécification d'un vocabulaire des symboles non-logiques et inclut : le type des entités, les attributs et les propriétés, les relations et les fonctions, et les contraintes." (R. Fikes).

Nous concluons que les définitions du terme ontologie adondent dans la littérature scientifique. Les définitions dans leur diversité offrent des point de vue à la fois différents et complémentaires.

Les taxonomies ou les thésaurus sont aussi des ontologies car elles se cantonnent à décrire des liens sémantiques du type "est une sorte de" et son inverse "est représenté par" ou,
plus spécifiquement, "est une sous-classe de." Des ontologies plus complexes permettent la représentation de liens sémantiques plus spécifiques, par exemple, "est localisé dans,". Mais surtout les ontologies les plus abouties permettent également l'intégration de propriétés particulières, de règles d'utilisation et de contraintes.

3. pourquoi les ontologie ?

Les ontologies ont pour objectif de représenter de façon générique et réutilisable la sémantique d'un domaine [48]. A l'origine, elles fournissent pour un domaine particulier, une compréhension commune utilisable pour partager de la connaissance ou des données. Actuellement,

Une ontologie définit un vocabulaire commun pour les chercheurs qui ont besoin de partager l'information dans un domaine. Elle inclut des définitions lisibles en machine des concepts de base de ce domaine et de leurs relations.
Pour quelles raisons développer une ontologie ? En voici quelques-unes [166] :

- Partager la compréhension commune de la structure de l'information entre les personnes ou les fabricants de logiciels.
- Permettre la réutilisation du savoir sur un domaine.
- Expliciter ce qui est considéré comme implicite sur un domaine.
- Distinguer le savoir sur un domaine du savoir opérationnel.
- Analyser le savoir sur un domaine.

4. Notion de base

Une ontologie contient les primitives terminologiques du domaine (le vocabulaire conceptuel) ainsi que des axiomes qui restreignent l'interprétation des primitives. Le vocabulaire conceptuel est structuré en un ensemble de concepts et un ensemble de relations existantes entre ces concepts.

4.1 Les concepts

Les concepts sont appelés aussi termes ou classes de l'ontologie, correspondent aux abstractions pertinentes d'un segment de la réalité (le domaine du problème), retenues en fonction des objectifs qu'on se donne et de l'application envisagée pour l'ontologie. Selon [136] ces concepts peuvent être classifiés selon plusieurs dimensions : 1) niveau d'abstraction (concret ou abstrait) ; 2) atomicité (élémentaire ou composée) ; 3) niveau de réalité (réel ou fictif).

Un concept est caractérisé par une extension qui est l'ensemble des objets (appelés instances du concept) manipulés à travers ce concept, et une intension qui est l'ensemble des propriétés spécifiant la sémantique du concept.

4.2 Les rôles

Les rôles traduisent les associations (pertinentes) existant entre les concepts présents dans le segment analysé de la réalité. Ces relations incluent les associations suivantes: 1) Sous-classe-de (généralisation – spécialisation) ; 2) Partie-de (agrégation ou composition) ; 3) Associée-à ; 4) Instance-de, etc. Ces relations nous permettent d'apercevoir la structuration et l'interrelation des concepts, les uns par rapport aux autres.

Un role est caractérisé par un terme, une extension qui est l'ensemble des tuples d'instances liés par cette relation, et une intension qui spécifie d'une part les concepts pouvant être liés par cette relationet d'autre part la sémantique d'usage de la relation.

4. 3 Les axiomes

Les axiomes constituent des assertions, acceptées comme vraies, à propos des abstractions du domaine traduites par l'ontologie.

Les axiomes sont employés pour modéliser les phrases qui sont toujours vraies. Ils peuvent être inclus dans une ontologie pour plusieurs buts, tels que définir la signification de composants d'ontologie, définir les contraints complexes sur les valeurs des attributs, les arguments de relations, etc., vérifier l'exactitude d'informations indiquées dans l'ontologie ou déduire une nouvelle information.

5. Développement des ontologies

La construction d'une ontologie va donc consister d'une part, à identifier les concepts d'un domaine de connaissance et la sémantique qui leur est associée et, d'autre part, à représenter cette sémantique de façon indépendante des diverses applications dans lesquelles pourra être utilisée l'ontologie. En effet, une ontologie n'est pas opérationnelle, au sens où elle n'inclue

pas de mécanismes de raisonnement, puisqu'elle doit justement être indépendante de tout objectif opérationnel. Le langage cible doit donc permettre de représenter les différents types de connaissances (connaissances terminologiques, faits, règles et contraintes) et de manipuler ces connaissances à travers des mécanismes adaptés à l'objectif opérationnel du système conçu.

5. 1.Principes de développement

Il existe un ensemble de critères et de principes qui ont fait leurs preuves dans le développement des ontologies et qui peuvent être résumés comme suit [48] :

- **Clarté et Objectivité** : L'ontologie doit fournir la signification des termes définis en fournissant des définitions objectives ainsi qu'une documentation en langage naturel.

- **Complétude** : Une définition exprimée par des conditions nécessaires et suffisantes est préférée à une définition partielle (définie seulement par une condition nécessaire et suffisante).

- **Cohérence** : Une ontologie cohérente doit permettre des inférences conformes à ces définitions.

- **Extensibilité monotonique maximale** : De nouveaux termes généraux et spécialisés devraient être inclus dans l'ontologie d'une façon qui n'exige pas la révision des définitions existantes.

- **Engagements ontologiques minimaux** : Ce principe invite à faire aussi peu de réclamations que possible au sujet du monde représenté. L'ontologie devrait spécifier le moins possible la signification de ses termes, donnant aux parties qui s'engagent dans cette ontologie la liberté de spécialiser et d'instancier l'ontologie comme elles le désirent.

- **Principe de distinction ontologique** [126] : les classes dans une ontologie devraient être disjointes. Le critère utilisé pour isoler le noyau de propriétés considérées comme invariables pour une instance d'une classe est appelé le critère d'*Identité*.

- **Modularité** [135] : Ce principe vise à minimiser les couplages entre les modules.

- **Diversification des hiérarchies** [125] : Ce principe est adopté pour augmenter la puissance fournie par les mécanismes d'héritage multiple. Si suffisamment de connaissances sont représentées dans l'ontologie et que suffisamment de différentes classifications de critères sont utilisées, il est plus facile d'ajouter de nouveaux concepts (puisqu'ils peuvent être facilement spécifiés à partir des concepts et des classifications de critères pré-existants) et de les faire hériter de propriétés de différents points de vue.

- **Distance sémantique minimale** [125]. Il s'agit de la distance minimale entre les concepts enfants de mêmes parents. Les concepts similaires sont groupés et représentés comme des sous-classes d'une classe, et devraient être définis en utilisant les mêmes primitives, considérant que les concepts qui sont moins similaires sont représentés plus loin dans la hiérarchie.

- Normaliser les noms [125]. Ce principe indique qu'il est préférable de normaliser les noms aussi autant que possible.

Cet ensemble de critères est généralement accepté pour guider le processus d'ingénierie ontologique.

5. 2. Méthodologies de developpement

Le développement des ontologies est une activité de modélisation qui est habituellement effectuée par les ingénieurs ontologique (également appelés les ontologists) qui ont la compréhension suffisante du domaine. Il est facile avec des langues de représentation de la connaissance. Quant à la conception des systèmes basés sur la connaissance, l'aide des experts du domaine est exigée pour construire et valider des ontologies.

La construction d'une ontologie est une activité complexe et longue. Par conséquent, les méthodologies ont émergé basé sur les expériences acquises dans la construction de grandes ontologies. Visent à faire au développement des ontologies plus un processus de technologie plutôt qu'à un art. Un aperçu récent des méthodologies de développement d'ontologies est présenté dans [111]. Les méthodologies plus complètes pour le développement d'ontology sont récapitulées dans la Table 2. 1. à l'exeption de la méthode de capture d'ontologie IDEF5 [95] qui est une partie de la méthodologie de modélisation d'entreprise complète IDEF (Integrated Computer Aided Manufacturing DEFinition), les autres méthodologies résultent des projets importants visant à construire de grandes ontologies :

TOVE	Enterprise	Methontology	IDEF5
Scenarios de motivation	objectif: niveau de formalité	Objectif et portée	Objectif et besoins
Besoins (questions informelles de compétence)	objectif: masse d'information i.e. liste des concepts appropriés	acquisition de connaissance par l'analyse des textes et des interviews d'expert	objectif: limites de l'ontologie collection des données basée sur les techniques d'acquisition de connaissances
specification formelle des objects, attributs and relations	definition formelle des termes	Conceptualisation: spécification des concepts, instances, relations basée sur une représentation informelle	Analyse de données : produire une liste des objets.
Besoins formels (questions formelles de compétence)		Intégration à partir des autres ontologies implémentation formelle dans un langage de représentation	Ontologie initiale Développement: définition des "proto-concepts" dans un langage schématique
specification des axiomes	Formal specification of axioms		raffinement de l'ontologie dans un langage plus structuré basé sur KIF.
Evaluation	Evaluation	Evaluation + documentation	Teste and validation utilisant les données actuelles.

Table 2. 1: Comparaison des méthodologies de construction d'ontologies.

- Projet TOVE (Toronto Virtual Enterprise) project [45],
- L'ontologie Enterprise [Uschold and Gruninger, 45],
- La methode Methotonlogy [47].

Les étapes principales qui peuvent être dérivées de ces méthodologies comprennent ce qui suit :

- Identification de la tâche pour laquelle l'ontologie est développée ;
- Définition des besoins de l'ontologie : objectif et portée ;
- Spécifications informelle : construire une spécifications informelle des concepts ;
- Codage : Représenter formellement les concepts et les axiomes dans un langage ;
- Évaluation de l'ontologie.

L'extensibilité et la maintenance des ontologies sont considérés par la plupart des auteurs mais ne sont pas une partie des méthodologies.

5.2. 1 Tove

TOVE (Toronto Virtual Enterprise) developpé par l'université de Toronto. Cette méthodologie repose sur les expériences de developpement d'une entreprise [45][51]. Elle s'appuie également sur les principales étapes suivantes pour le developpement d'une ontologie :

- Capturer les scénarios de motivations : Cette étape consiste à identifier les scénarios qui clarifient le domaine que l'on investit et les différentes applications dans lesquelles l'ontologie sera appliquée.
- Formuler les questions de compétences informelles : Cette étape consiste à formuler un ensemble de questions basées sur les scénarios, exprimées en langage naturel, afin de déterminer la portée de l'ontologie. Ces questions et leurs réponses sont utilisées pour extraire les concepts principaux, leurs propriètés et les relations qui existent entre ces concepts.
- Spécifier la terminologie de l'ontologie : Cette étape consiste à représenter les termes (concepts, propriètés et relations), identifier dans l'étape précédente, en utilisant le formalisme de la logique du premier ordre. Les concepts seront représentés sous forme e constantes ou bien des variables. Par ailleurs, les propriètés et les relations seront représentées par des prédicats.
- Evaluer la complétude de l'ontologie.

5. 2. 2 Enterprise

Uschold et Grüninger proposent le squelette d'une méthode basé sur l'expéience de construction d'ontologies dans le domaine de la gestion des entreprises [45]. La méthode Enterprise repose sur les quatre (4) étapes suivantes :

- Identifier le rôle et la porté de l'ontologie.
- Réaliser l'ontologie où les activitées suivantes sont distinguées : identifier les concepts et les relations fondamentaux, définir provisoirement ces élements, coder l'ontologie dans un langage adapté, intégrer les ontologies existantes.

- Evaluer l'ontologie.
- Rédiger une documentation et une trace des actions réalisées lors des différenes étapes.

Les étapes de la méthode Enterprise sont décrites de façon abstraite. Les techniques utilisées pour les activitées ne sont pas précises (ex : comment identifier les concepts fondamentaux ? quel langage utilisé pour représenter l'ontologie ?).

5.2. 3 Methontology

Methontology se situe entre le GL (Génie Logiciel) et l'IC (Ingénierie des Connaissances) [154] [155]. Elle identifie une sequence d'activitées de gestion, de developpement et de support pour le developpement de l'ontologie. Dans Methontology, nous distinguons les étapes (cf. figure 2. 1) :

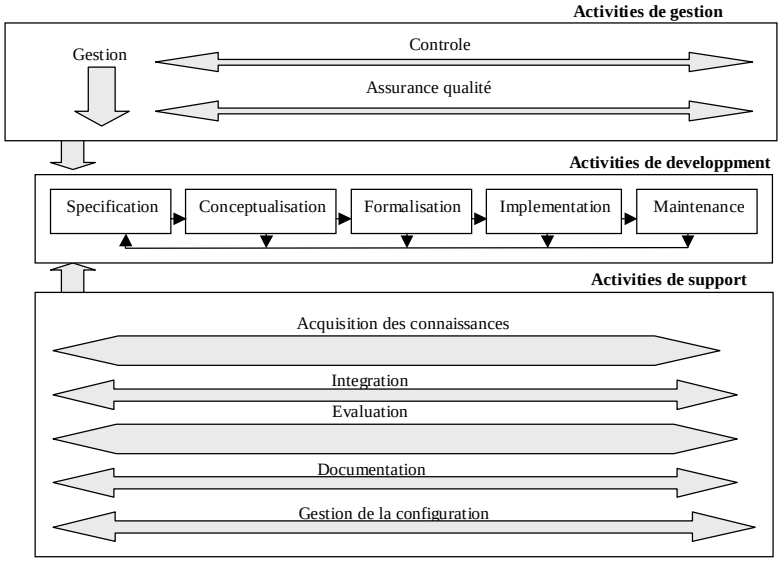


Figure 2. 1 : Cycle de vie d'une ontology dans Methontology.

- Spécification : Le developpement d'une ontology commence par la définition de sa domaine et de sa porté. Cela est basé sur la réponse à certaines questions :
  - Quel est le domaine que couvert l'ontologie ?
  - A quoi servir cette ontologie ?
  - Quels types de questions l'ontologie doit fournir des réponses ?
  - Qui va utiliser l'ontologie ?
  - etc.

  Les reponses à ces questions peuvent changer durant le processus de developpement de l'ontologie. Elles permetent de limiter la portée de l'ontologie.

- Conceptualisation : Elle consiste à identifier et à structurer les connaissances du domaine à partir des sources d'informations. L'acquisition de ces connaissances peut s'appuyer à la fois sur l'analyse de documents et l'interview des expets du domaine. Une fois que les concepts sont identifiés par leurs termes. La semantique de ces concepts est décrite par un langage semi-formel à travers leurs propriètés et leurs instances.
- Formalisation : Cette étape consiste à formaliser le modèle conceptuel obtenu dans l'étap précédentepar un formalisme de représentation d'ontoogie telle que DL.

- Implementation : Cette étape requiert le choix d'un langage permettant de représenter les connaissances terminologiques et les axiomes puis de manipuler ces représentations pour raisonner.
- Maintenance : il s'agit d'une maintenance corrective ou évolutive (nouveaux besoins de l'utilisateur), ce qui permet l'évolution et la validation de l'ontologie. Cette activitée est réalisée par le constructeur et des experts du domaine. La vaidation se base sur les services d'inférences associés au DL, offerts pa des raisonneurs.

6. Formalismes de représentation

Les modèles de représentation de connaissance utilisés en ingénierie ontologique peuvent être regroupés selon les paradigmes conceptuels qu'ils réifient. Sont ainsi distingués :

a) Les graphes

i) Les frames

ii) Les graphes conceptuels.

iii) Orienté-Web : RDF et RDF Schéma

b) Logique

i) Du premier ordre: KIF

ii) Descriptive: KL-One, OIL, DAML+OIL, OWL

c) Orienté objet : UML + OCL

Chacun de ces modèles de représentation est implémenté dans un ou plusieurs langages implémentant une partie ou la totalité du modèle, en particulier des langages adaptés au sémantique. Les langages implémentant ces modèles sont souvent opérationnels, c'est-à-dire qu'ils offrent des mécanismes de raisonnement, et servent alors à représenter des ontologies déjà opérationnelles. Le modèle de la logique de description étant celui que nous utilisons dans nos travaux, nous en détaillerons la présentation.

6.1 Frames

Le modèle des Frames est un classique de l'Intelligence Artificielle et a été initialement proposé comme langage de représentation d'ontologies par T. GRUBER [48]. Le principe de ce modèle est de décomposer les connaissances en ***classes*** (ou frames) qui représentent les concepts du domaine. À un frame est rattaché un certain nombre d'***attributs*** (slots), chaque attribut pouvant prendre ses valeurs parmi un ensemble de ***facettes*** (facets) [148]. Une autre façon de présenter ces attributs est de les considérer comme des ***relations*** binaires entre classes dont le premier argument est appelé ***domaine*** (domain) et le deuxième ***portée*** (range) [150]. Des ***instances*** des classes, correspondant à l'extension de chaque concept, peuvent être ajoutées, ainsi que des ***fonctions*** qui sont des types particuliers de relations liant un ensemble de classes à une valeur calculée à partir des valeurs des attributs des classes. La spécification de propriétés conceptuelles des attributs (ou relations) recourt à des formules de la logique du premier ordre. La sémantique de la subsomption est purement extensionnelle : une frame F1 est plus spécifique qu'une frame F2 si toute instance de F1 est instance de F2. F-Logic est l'exemple le plus connu de langage opérationnel à base de frames [148]. L'Open Knowledge Base Connectivity (OKBC [151]), protocole et API de requête et d'interfaçage entre bases de connaissances, utilise également le modèle de frame. Le Knowledge Interchange Format (KIF) est un langage, non opérationnel, implémentant le modèle des frames en logique du premier ordre pour la représentation de connaissance [149].

6. 2 Graphes conceptuels

Introduit par J. SOWA au début des années 80 [147], le modèle des Graphes Conceptuels (GC) appartient à la famille des réseaux sémantiques [146]. Les réseaux sémantiques modélisent les connaissances sous forme de graphes, les noeuds étant associés à des concepts et les arêtes à des relations. Ce modèle se prête bien à des présentations graphiques des connaissances mais présente, comme nous allons le voir, certains problèmes particuliers de représentation.

Le modèle des GCs se décompose en deux (2)parties :

- Une partie terminologique dédiée au vocabulaire conceptuel des connaissances à représenter, c'est-à-dire les types de concepts3, les types de relations et les instances des types de concepts4. Cette partie correspond à la représentation du modèle conceptuel mais intègre également des connaissances sur la hiérarchisation des types de concepts et de relations ;
- Une partie assertionnelle dédiée à la représentation des assertions du domaine de connaissance étudié.

6. 3 Logiques de Description

Les Logiques de Description (DL) combinent des représentations intentionnelles et extensionnelles des connaissances [40] :
_ une TBox (T pour terminologique) contient les déclarations des primitives conceptuelles organisées
en *concepts* et *rôles* (relations binaires entre concepts dotées d'un domaine et d'une portée similaires aux attributs des frames). Ces déclarations décrivent les propriétés des concepts et des rôles et constituent donc une définition intentionnelle des connaissances ;

_ une ABox (A pour assertionnel) contient les déclarations d'*individus*, instances des concepts définisdans la TBox.

La spécification de propriétés des concepts et rôles utilise des constructeurs ensemblistes et logiques. Plusieurs modèles de DL peuvent être utilisés en fonction des constructeurs nécessaires à la modélisation. Le fait d'utiliser tous les constructeurs ou seulement une partie d'entre eux joue sur la décidabilité et la complexité des calculs de raisonnement. Loom [144] et CLASSIC [145] sont des exemples de modèles opérationnels implémentant ce modèle. Il est de plus utilisé dans les langages de représentation de connaissance OIL et OWL développés pour le Web.

7. Langages de spécification d'ontologies

Plusieurs langages de spécification d'ontologies ont été developpés pendant ces dernières années. Parmi ces langages, nous citons :

7. 1 KIF

KIF (Knowledge Interchange Format) [14] est un langage basé sur les prédicats du premier ordre avec des extensions pour représenter des définitions et des méta-connaissances. Tant que la logique du premier ordre est un langage de bas niveau pour l'expression d'ontologies,

l'outil *Ontolingua* [140] permet aux utilisateurs de construire des ontologies KIF à un niveau plus élevé de la description par l'importation des définitions des ontologies prédéfinies.

7. 2 KL-ONE

KL-ONE [100] est un langage basé sur la logique de description [99]. Il est une formalisation de représentation de la connaissance à base de cadres[5] (« frame-based »). Ce système maintient la définition des concepts par un simple nommage, et l'indication de la correspondance des concepts dans une hiérarchie de généralisation/spécialisation. De nouveaux termes peuvent être définis par des opérations de conjonction des concepts. Par exemple l'opérateur « *and* » peut être utilisé pour préciser qu'un nouveau concept est une pécialisation commune de plusieurs autres concepts. De nouveaux rôles peuvent être introduits pour représenter les relations qui peuvent exister entre des individus dans le domaine modélisé. Les définitions des concepts peuvent inclure des restrictions sur les valeurs possibles, sur les nombres de valeurs, ou sur le type de valeurs qu'un rôle peut avoir pour un concept.

7. 3 RDF(S) Resource Description Framework and RDF schema

RDF [102] « *Resource Description Framework* » est un formalisme graphique pour représenter des méta-données. Il est basé sur la notion de triplet (sujet, prédicat, objet). Le sujet et l'objet sont des ressources liées par le prédicat. RDF utilise la syntaxe XML, mais il ne donne aucune signification spécifique pour le vocabulaire comme sous classe de, ou le type. Les primitives de modélisation offertes par RDF sont très basiques. RDF Schéma [102] est un langage qui étend RDF avec un vocabulaire de termes et les relations entre ces termes, par exemple *Class*, *Property*, *type*, s*ubClassOf*, *subPropertyOf*, *range* et *domain*. RDFS est reconnu comme un langage d'ontologie qui définit :

- Des classes et des propriétés.
- Les sous-classes, les super-classes, les sous-propriétés, et les super-propriétés.
- Le domaine de définition et le domaine image des propriétés.

7. 4 OIL (Ontology Interchange Language)

Dans l'optique d'une utilisation d'ontologies sur leWeb, le langage RDF(S) a été enrichi par l'apportdu langage OIL (Ontology Interchange Language) qui permet d'exprimer une sémantique à travers le modèle des frames tout en utilisant la syntaxe de RDF(S). OIL offre de nouvelles primitives permettant de définir des classes à l'aide de mécanismes ensemblistes issus des logiques de description (intersection de classes, union de classes, complémentaire d'une classe). Il permet également d'affiner les propriétés de RDF(S) en en contraignant la cardinalité ou en en restreignant la portée [152].

Le langage OIL a été fusionné avec le langage DAML pour former le DAML+OIL. DAML (Darpa Agent Markup Language5) est conçu pour permettre l'expression d'ontologies dans une extension du langage RDF. Il offre les primitives usuelles d'une représentaton à base de frames et utilise la syntaxe RDF [153]. L'intégration de OIL rend possible les inférences compatibles avec les logiques de description, essentiellement les calculs des liens de subsomption.

7. 5 DAML+OIL (DARPA Agent Markup Language +OIL)

Beaucoup de travaux ont été faits dans le domaine de la représentation des connaissances parmi lesquels on peut citer les plus importants : SHOE, OntoBroker [104], OIL [105], et encore DAML + OIL [103] qui a remplacé DAML – ONT6. DAML + OIL est un langage construit sur des normes précédentes du W3C telles que RDF et RDF Schéma, et étend ces langages avec des primitives de modélisation plus riches. DAML+OIL a été conçu à partir du langage d'ontologie DAML-ONT (DARPA Agent Modelling Language-Ontology, Octobre 2000) en vue de combiner plusieurs composants du langage OIL [105]. OIL7 « *Ontology Inference Language* » est une représentation basée sur le Web, et une couche d'inférence pour des ontologies. Il combine les primitives de modélisation des langages à base de cadres (frames) avec la sémantique formelle et le raisonnement fournis par la logique de description.

7. 6 OWL (Web Ontology Language)

Développé par le groupe de travail sur le Web Sémantique du W3C, OWL peut être utilisé pour représenter explicitement les sens des termes des vocabulaires et les relations entre ces termes. OWL vise également à rendre les ressources sur le Web aisément accessibles aux processus automatisés [101], d'une part en les structurant d'une façon compréhensible et standardisée, et d'autre part en leur ajoutant des méta-informations. Pour cela, OWL a des moyens plus puissants pour exprimer la signification et la sémantique que XML, RDF, et RDF-S [102]. De plus, OWL tient compte de l'aspect diffus des sources de connaissances et permet à l'information d'être recueillie à partir de sources distribuées, notamment en permettant la mise en relation des ontologies et l'importation des informations provenant explicitement d'autres ontologies [96].

(i) Pourquoi OWL ?

XML [11 abdul ghafour] fournit une syntaxe pour des documents structurés, mais n'impose aucune contrainte sémantique à la signification des documents. RDF est un modèle de données pour représenter les objets (Ressources) et les relations entre eux, fournissant une sémantique simple pour ce modèle qui peut être représenté dans une syntaxe XML. RDF Schema [102] est un langage de définition de vocabulaires pour la description de propriétés et de classes représentées par des ressources RDF. RDFS permet de définir des graphes de triplets RDF, avec une sémantique de généralisation / hiérarchisation de ces propriétés et de ces classes. OWL ajoute des vocabulaires pour la description des propriétés et des classes, des relations entre classes (exemple disjointness), des cardinalités, des caractéristiques de propriétés (symmetry), et des classes énumérées. OWL est développé comme une extension du vocabulaire de RDF et il est dérivé du langage d'ontologies DAML + OIL [103].

(ii) Sous langages de OWL

OWL a trois sous langages de plus en plus expressifs: OWL Lite, OWL DL, et OWL Full :

- **OWL Lite** : supporte les utilisateurs ayant besoin principalement d'une hiérarchie de classification et des contraintes simples (un ensemble est limité à 0 ou 1 élément, par exemple). Il a une complexité formelle inférieure à celle de OWL DL. OWL Lite supporte seulement un sous-ensemble de constructions du langage OWL.

- OWL DL : D'après son nom OWL DL utilise la logique de description DL [16 abdul ghafour]. Il supporte les utilisateurs qui réclament l'expressivité maximale tout en retenant la complétude informatique (toutes les conclusions sont garanties d'être calculables), et la possibilité de décision (les calculs finiront en un temps fini). Il inclut toutes les constructions du langage OWL, qui ne peuvent être utilisées que sous certaines restrictions.

- **OWL Full** : a été défini pour les utilisateurs qui veulent une expressivité maximale et une liberté syntaxique de RDF sans des garanties informatiques. OWL Full permet à une ontologie d'augmenter la signification du vocabulaire prédéfini (RDF ou OWL). Il est peu probable que n'importe quel logiciel de raisonnement soit capable de supporter le raisonnement complet de chaque caractéristique de OWL Full. Autrement dit, en utilisant OWL Full en comparaison avec OWL DL, le support de raisonnement est moins prévisible puisque l'implémentation complète de OWL Full n'existe pas actuellement.

OWL Full et OWL DL maintiennent le même ensemble de constructions de OWL. La différence se situe dans les restrictions sur l'utilisation de certaines de ses caractéristiques et sur l'utilisation des caractéristiques de RDF. OWL permet le mélange libre de OWL avec RDF Schéma et, comme RDFS, n'impose pas une séparation stricte des classes, des propriétés, des individus, et des valeurs de données. OWL Full peut être vu comme étant une extension de RDF, tandis que OWL Lite et OWL DL peuvent être vus comme des extensions d'une vue restreinte de RDF. Alors les utilisateurs de RDF devraient se rendre compte que OWL Lite n'est pas simplement une extension de RDFS. OWL Lite met des contraintes sur l'utilisation du vocabulaire de RDF (par exemple, *disjointness* des classes, des propriétés, etc.). OWL Full est conçu pour la compatibilité maximale de RDF. Quant à opter OWL DL et OWL Lite, il faut considérer si les avantages du OWL DL/Lite (par exemple, support de raisonnement) l'emportent aux restrictions de DL/Lite à l'utilisation des constructions de OWL et de RDF.

7. 7 Modèle orienté objet

la vaste communauté d'utilisateurs et le support commercial pour les normes orientées objet justifient l'investigation des techniques de modélisation objet pour le développement des ontologies [106]. Par conséquent, on peut exploiter la grande croissance des technologies orientées objet en adoptant UML + OCL comme un formalisme alternatif pour la représentation des ontologies.

a) UML

UML est un langage et une notation graphique associée pour l'analyse et la modélisation orientée objet. Le paradigme de modélisation orientée objet est devenu la technique principale dans le développement de logiciel de l'industrie basé sur la vue largement acceptée que la modélisation orientée objet s'adapte bien aux modèles intuitifs du monde [107].

b) OCL

OCL (Object Constraint Language) est un langage qui permet d'écrire des expressions et des contraintes sur des modèles orientés objet. Une expression est une indication ou une spécification d'une valeur. Une contrainte est une restriction sur une ou plusieurs valeurs sur un modèle orienté – objet ou une partie de ce modèle.

Divers langages de contraintes ont été utilisés dans des modèles orientés objets, par exemple Syntropy [109], Catalysis [108] et BON [110] et des langages de programmation Eiffel [143]. OCL est le langage standard qui fait partie de UML mis par l'OMG comme un standard pour l'analyse et le modèle orienté – objet [111]. Dans les diagrammes des classes, OCL peut être utilisé pour contraindre des valeurs d'attributs et des instances possibles de relations. Les expressions peuvent être utilisées dans un modèle UML afin de :

- spécifier la valeur initiale d'un attribut ou association « end ».
- spécifier la règle de dérivation pour un attribut ou une association « end ».
- spécifier le corps d'une opération.

Il existe 4 types de contraintes :
- Invariant : est une contrainte qui affirme qu'une condition est satisfaite par toutes les instances de la classe, type, ou interface.
- Précondition : une restriction qui doit être vraie au moment où l'opération va être exécutée.
- Postcondition : une restriction qui doit être vraie au moment où l'opération a fini son exécution.
- Guard : une contrainte qui doit être vraie avant qu'un état de transition soit déclenché.

8. Environnements et outils de modélisation des ontologies

Un ensemble d'environnements d'ingénierie ontologique ont été développés afin de systématiser l'ingénierie des ontologies. Selon [128], les plus connus sont : ONTOLINGUA [140], ONTOSAURUS [141], ODE (Ontology Design Environment)[127], PROTéGé WIN (maintenant connu sous le nom de PROTéGé 2000) [134] [137] et enfin, TADZEBAO et WEBONTO [131]. Ces outils font l'objet d'une étude comparative des outils d'ingénierie ontologique proposée par l'équipe WonderTools de l'Université d'Amsterdam [132]. D'autres outils recensés dans [138] méritent d'être cités : HOZO [133], KAON (anciennement connu sous le nom de OntoEdit) [129], [130] et OILED [139] :

- **KAON** (Karlsuhe Ontology and Semantic Web) est un environnement open source modulaire, basé dans Java, destiné à la conception, au développement et à la gestion d'ontologies. L'environnement intègre les modules suivants : API, Query, Serveurs (d'ontologie et d'application), Générateur de portails web (basés sur les ontologies), Éditeur d'ontologie (construction et maintenance).

- **HOZO** du MizLab de l'Université d'Osaka; Hozo est un environnement composé d'un éditeur et d'un serveur d'ontologies. L'éditeur est développé en applets Java afin de pouvoir fonctionner comme un client via l'Internet. Hozo gère les ontologies et ses instances pour chaque programmeur. Chacun peut lire et copier toutes les ontologies et les instances présentes dans Hozo, mais ne peut pas modifier celles développées par d'autres. La vérification de la consistance d'une instance se fait en utilisant les axiomes définis dans l'ontologie. Hozo gère l'exportation des ontologies et modèles en format XML, RDF, DAML+OIL.

- **ODE** du laboratoire d'Intelligence Artificielle de l'Université de Madrid. Les principaux avantages de ODE sont le module de conceptualisation pour construire des ontologies et le module pour construire des modèles conceptuels ad hoc.

- **ONTOLINGUA** de l'Université Stanford; Le serveur Ontolingua est le plus connu des environnements de construction d'ontologies en langage Ontolingua. Il consiste en un ensemble d'outils et de services qui supportent la construction en coopération d'ontologies, entre des groupes séparés géographiquement.

- **ONTOSAURUS** de l'Information Science Institute de l'Université de Southern California. Ontosaurus consiste en un serveur utilisant LOOM comme langage de représentation des connaissances, et en un serveur de navigation créant dynamiquement des pages HTML qui affichent la hiérarchie de l'ontologie; le serveur utilise des formulaires HTML pour permettre à l'usager d'éditer l'ontologie. Des traducteurs du LOOM en Ontolingua, KIF, KRSS et C^{++}, ont été développés.

- **OILED** de l'Information Management Group de l'Université de Manchester développé par Sean Bechhofer et Gary Ng se veut un éditeur freeware d'ontologies, destiné à supporter le développement d'ontologies de petites et moyennes tailles, basées sur le standard DAML+OIL. OILEd n'est pas un environnement de développement d'ontologies offrant des fonctionnalités supportant le cycle complet de conceptualisation et d' opérationnalisation.

- **PROTéGé-2000** du département d'Informatique Médicale de l'Université Stanford; Protégé-2000 successeur de PROTéGéWIN, est un outil, une plateforme et une librairie d'ontologies, qui permettent : 1) de construire une ontologie du domaine, 2) de personnaliser des formulaires d'acquisition de connaissances et 3) de transférer la connaissance du domaine.

- **WEBONTO** du Knowledge Media Institute de l'Open University; WebOnto et TADZEBAO sont des outils complémentaires. Tadzebao permet aux ingénieurs des connaissances de tenir des discussions sur les ontologies, en mode synchrone et asynchrone. WebOnto supporte la navigation collaborative, la création et l'édition d'ontologies sur le Web.

9. Systèmes de raisonnement

Il est également possible de raisonner sur les ontologies en utilisant un moteur d'inférence général tel que FACT [38], ou des outils d'inférence spécifiques au Web sémantique basés sur des DL tels que RACER [39]. Ces deux outils peuvent être facilement intégrés à Protégé. Les logiques de description permettent de définir les bases logiques des différents formalismes de représentation de la connaissance tant sur le plan de la représentation que sur le raisonnement.

Dans notre travail, notre choix est porté sur le système RACER (Renamed Abox and Concept Expression Reasoner). Ce dernier se présente sous la forme d'un serveur qui peut être accédé par le protocole TCP ou http. C'est un système de représentation de connaissances. Il implémente des tableaux de calcul hautement optimisé pour und DL très expressive. Il interprète les documents OWL et offre des services de raisonnement aussi bien pour le niveau terminologique que pour le niveau assertionnel de l'ontologie. Il permet aussi de vérifier la consistence et la relation de subsomption entre les concepts et d'autres tests plus élaborés sur les instances et les rôles.

10. Intégration des ontologies

Le mot intégration est utilisé avec plusieurs significations dans le domaine d'ontologie. [142] définie trois différentes significations :

a) **Intégration** : Construction d'une nouvelle ontologie réutilisant (en assemblant, étendant, spécialisant, ou adaptant) d'autres ontologies disponibles. Ces différentes ontologies font partie de la nouvelle ontologie.

b) **Fusion** : Construction d'une ontologie par la fusion de différentes ontologies dans une seule ontologie qui les unifie.

c) **Utilisation** : Construction d'une application utilisant une ou plusieurs ontologies pour spécifier ou implémenter un système à base de connaissance.

La distribution et l'hétérogénéité de l'information ont créé le besoin de l'intégration pour améliorer l'accès aux données et assurer une recherche pertinente. Différentes approches existent et sont évaluées dans [112] vis-à-vis de quatre caractéristiques principales:

Le rôle et l'architecture des ontologies affectent profondément le formalisme de représentation d'une ontologie. Les ontologies sont utilisées pour la description explicite de la sémantique de l'information source. Ces ontologies sont employées selon différentes méthodes :

a) ***Approche d'une ontologie simple*** : Cette approche utilise une ontologie globale fournissant un vocabulaire partagé pour la spécification de la sémantique. Toutes les sources d'informations sont reliées à une ontologie globale. Une approche importante pour ce type d'intégration des ontologies est SIMS [113]. Un modèle indépendant de chaque source d'information doit être décrit pour ce système en reliant les objets de chaque source au modèle du domaine global.

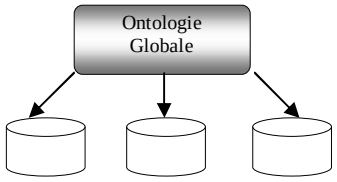


Figure 2. 2 : Approche d'une ontologie simple.

b) ***Ontologies multiples*** : Chaque source d'information est décrite par sa propre ontologie. L'avantage est que chaque ontologie source peut être définie sans prendre en considération les autres sources ou les autres ontologies. Mais le manque d'un vocabulaire commun conduit à une difficulté extrême pour comparer différentes ontologies sources. Pour surmonter ce problème, un formalisme de représentation additionnelle définissant le mapping inter-ontologie est fourni. Ce dernier identifie sémantiquement les termes correspondants des différents ontologies sources. Ce mapping est difficile à définir à cause de divers problèmes

d'hétérogénéité sémantique qui peuvent se produire. L'approche OBSERVER [114] est utilisée pour ce type.

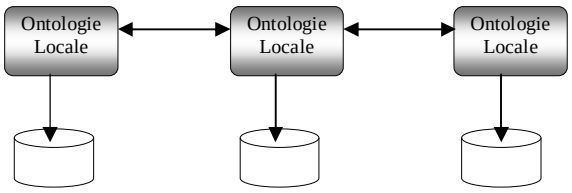


Figure 2. 3 : Approche d'ontologie multiple

c) ***Approche hybride*** : Comme l'approche multiple, la sémantique de chaque source est décrite par sa propre ontologie. Mais pour rendre les sources d'ontologies comparables, elles sont construites sur un vocabulaire partagé global [115] [116]. Ce vocabulaire partagé comprend les termes de bases ou primitives d'un domaine. Afin de construire des termes complexes, ces primitives sont combinées par des opérateurs, et les termes seront comparés plus facilement qu'une approche multiple. Parfois, ce vocabulaire partagé est représenté sous forme d'ontologie [117]. Dans COIN [115], la description locale d'une information, qu'on appelle contexte, est simplement un vecteur de valeurs d'attributs. Dans METACOTA [116], chaque source d'information est annotée par un label qui indique la sémantique de l'information. Ce label combine les termes primitifs du vocabulaire partagé. Dans BUSTER [117], le vocabulaire partagé est une ontologie générale. Une source d'ontologie est un raffinement de l'ontologie générale. L'avantage de cette approcheest que des nouvelles sources peuvent être ajoutées sans le besoin de la modification du vocabulaire partagé. Elle maintient encore l'acquisition et l'évolution des ontologies.

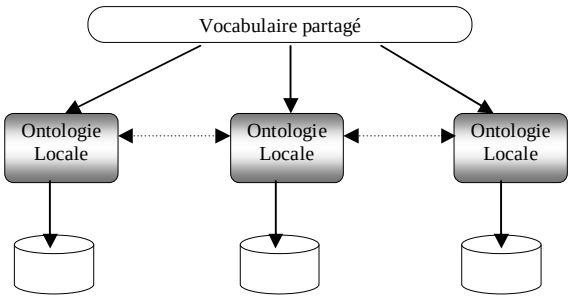


Figure 2. 4 : Approche d'ontologie hybride.

10. 1 Alignement des ontologies

L'alignement d'ontologies consiste à trouver des correspondances entre les connaissances spécifiées dans les deux ontologies, de manière à pouvoir les exploiter conjointement dans le même système. En pratique, il s'agit d'identifier des concepts (ou des relations) de la première ontologie avec

des concepts (ou des relations) de la seconde, ou de trouver des liens conceptuels (subsomption,...) entre eux.

10. 2 Fusion des ontologies

Contrairement à l'alignement où les deux ontologies de départ reste intactes, la fusion d'ontologies consiste, à partir de deux ontologies, à en créer une troisième qui intègre les connaissances spécifiées dans les deux premières.

10. 3 Mapping des ontologies

La relation d'une ontologie avec son environnement joue un rôle essentiel dans l'intégration de l'information. Mappings réfère à la mise en correspondance d'éléments d'ontologies entre eux ou avec des éléments d'autres types de ressources sémantiques.

Les deux plus importantes sortes de mapping utilisées pour l'intégration de l'information sont le mapping entre les ontologies et l'information qu'elles décrivent, et le « mapping » entre les différentes ontologies utilisées dans un système.

a) **Connexion aux sources d'informations**: différentes approches générales sont utilisées pour établir une connexion entre ontologies et sources d'informations:

- *Ressemblance de la structure* : une approche directe est de produire une simple copie de structure de la source d'information et l'encoder dans un langage qui rend possible le raisonnement automatique. Cette approche est implémentée par le médiateur SIMS [113] et le système TSIMMIS [121].

- *Définition des termes* : afin de clarifier la sémantique des termes dans un schéma de base de données, des approches comme BUSTER [122] utilisent les ontologies pour définir des termes supplémentaires de la base de données ou de son schéma.

- *Enrichissement de la structure* : combine les deux approches précédentes. Un modèle logique est construit qui ressemble à la structure de l'information source et contient des définitions additionnelles des concepts. Les systèmes qui utilisent l'enrichissement de la structure pour l'intégration de l'information sont OBSERVER [114], Kraft [120], PICSEL [118] et DWQ [119].

- *Méta-annotation* : une nouvelle approche est l'utilisation des méta-annotations pour ajouter la sémantique sur la source d'information. Cette approche est devenue proéminente dans le besoin d'intégrer des informations présentes sur le WORLD WIDE WEB où l'annotation est un moyen naturel pour l'ajout de la sémantique. Des approches développées pour être utilisées sur le WWW sont Ontobroker [123], et SHOE [124].

b) **Mapping inter-ontologies**: le problème de mapping entre différentes ontologies est bien reconnu dans l'ingénierie de la connaissance. Des approches générales sont utilisées dans les systèmes d'intégration de l'information :

- *Les mappings définis* : une approche commune est de fournir la fournir la possibilité de définir des mappings. Elle est utilisée dans Kraft [120] où les translations entre différentes ontologies sont réalisées par des agents médiateurs spéciaux qui peuvent être adaptés pour

établir la traduction entre différentes ontologies et même entre différents langages. Cette approche permet une grande flexibilité, mais elle ne peut pas assurer la préservation de la sémantique : l'utilisateur est libre de définir des mappings arbitraires même s'ils n'ont pas de significations ou s'ils produisent des conflits.

- *Relations lexiques* : Ces approches étendent le modèle de la logique de description par des relations inter-ontologies empruntées à la linguistique, par exemple « synonyme », « disjoint », « hyponyme », et « hypernyme » qui sont utilisés dans le système OBSERVER [114].

- *Connaissance de base haut niveau* : afin d'éviter la perte de la sémantique, on doit rester dans le langage de la représentation formelle en définissant des « mappings » entre différentes ontologies. Une méthode directe pour rester dans le formalisme est de relier toutes les ontologies utilisées à une ontologie simple du haut – niveau. Tant que cette approche permet d'établir des connections entre des concepts de différentes ontologies en termes des superclasses communes, elle ne permet pas la correspondance directe.

- *Correspondance sémantique*: pour surmonter l'ambiguïté causée par le « mapping » indirect (comme l'approche précédente), une solution est d'identifier des correspondances sémantiques entre les concepts des différentes ontologies. Ces approches comptent sur un vocabulaire commun pour la définition des concepts des différentes ontologies.

En ce qui concerne notre approche de mapping d'ontologie, nous avons adopté le concept de sémantique bridge. Ce dernier est approprié pour représenter les mapping entre les ontologies d'application developpées. Sémantique bridge considère différentes dimensions. Chacune d'elles décrit un aspect particulier [36] :

- **Entité** : cette dimension reflète le type d'entités d'ontologie à lier (par exemple les classes, les propriétés et les relations).

- **Cardinalité** : cette dimension reflète le nombre d'entités d'ontologie à lier (1:1, 1:n, n:m).

- **Structurel** : cette dimension reflète la manière dont les liens élémentaires peuvent être combinés dans des mapping plus complexes (spécialisée, abstrait, composé ou alternative).

- **Contrainte** : cette dimension reflète les contraintes appliquées pendant la phase d'exécution aux instances de l'ontologie source.

- **Transformation** : cette dimension reflète comment les instances de l'ontologie source sont transformées pendant le processus de mapping.

Ces différentes approches traitant de l'intégration des ontologies pour un domaine ont permis la distinction des problèmes d'hétérogénéité entre les différentes ontologies.

11. Hétérogénéité

Cependant, beaucoup de projets opérationnels et de recherche ont abouti à la création des ontologies, parfois pour un même domaine. Ces ontologies ne sont homogènes ni dans leur structure, ni dans leur sémantique, et l'hétérogénéité est aperçue lors de la mise en commun

de la connaissance représentée avec ces ontologies. Un certain nombre de recherches tente à étudier les causes du problèm d'inconsistance ou de discordance entre différentes ontologies. Ceci a amené à la mise en oeuvre de multiples approches qui visent à traiter ce type de problème. Ces approches mettent en considération les différents cas qui peuvent y exister. Wiederhold [156] a distingué dans son approche les cas où la consistance, pour un domaine, n'est pas maintenue. Il a référé les différences entre des ontologies aux différences de la nomination et de la portée en terme des noms et de la sémantique des méta-informations sur les attributs qui apparaissent dans le schéma, et des noms et la sémantique qui apparaissent comme des valeurs dans le contenu de la source d'information. Quatre différences y sont déterminées :

1- Deux items d'attributs dans deux ontologies ont deux noms différents est le cas le plus commun et le plus facile à résoudre. Une simple table peut être utilisée pour maintenir la correspondance désirée, et rassembler l'information.

2- La différence de portée des concepts manipulés est plus insidieuse, et doit être déterminée par une analyse du contenu. La résolution exige l'établissement, la validation, et le traitement des règles.

3- Des différences relatives aux unités de mesure prises par des nombres sont communes. Une conversion peut être établie par une formule (un mètre = pieds/0.305). Dans ce cas des règles ou des tables doivent être introduites.

4- Des portées sur des attributs sont subjectives. Le terme «*table*» a une signification différente dans le domaine des meubles dans le domaine des bases de données. Les différences dans la portée mènent aux différences dans la mise en référence qui fait leur résolution pourtant plus critique.

L'approche Mitra [157] traite l'hétérogénéité sémantique entre multiples sources. Afin d'établir une correspondance entre les termes des ontologies, cette approche explore les discordances et leurs solutions en utilisant des graphes objets. Quatre types de discordance sont déterminés :

1- Discordance de la sémantique des termes : si un terme de deux ontologies différentes réfère à des concepts différents.

2- Discordance de structure : un même terme dans une source correspond à plusieurs termes dans une autre source, et cause qu'un noeud dans un graphe s'approprie à plusieurs noeuds dans un autre. Cette approche permet ce type de correspondance.

3- Discordance des instances : une instance d'une classe dans une source n'est pas une instance de la même classe d'une autre source. Les règles d'articulation doivent spécifier explicitement quelles correspondances on veut générer dans ces cas.

4- Discordance de granularité : si on a deux noeuds correspondants, et le grand-père d'un noeud correspond avec le père de l'autre. Autrement dit, dans le premier graphe un concept est organisé dans une hiérarchie plus élaborée que dans le deuxième graphe.

Le besoin de traduire une représentation symbolique d'une connaissance en différents formats était le motif de la création du système « *Ontomorph* » par Chalupsky [158]. Ce système

permet de traiter les problèmes qui peuvent se produire lors d'une telle traduction. Le scénario est de traduire les différents modèles du même domaine. Ces modèles se diffèrent à travers de multiples dimensions. Selon Chalupsky, six types de discordances communes qui peuvent être trouvées sont identifiés :

1- Syntaxe du langage de représentation de la connaissance (KR) : deux ontologies peuvent avoir des syntaxes et des conventions syntaxiques différentes.

2- Expressivité du langage de KR : chaque langage supporte l'expressivité d'une manière différente des autres langages. Par exemple, la négation qui peut être représentée par un langage mais pas dans un autre. Dans ce cas, des choix difficiles doivent être pris sur la manière de traduire les idiomes représentatifs.

3- Conventions de modélisation : même si le langage de représentation de la connaissance pour une base de connaissance source et, une autre cible sont les mêmes, ça n'empêche pas d'avoir des différences à cause de la manière de modélisation d'un domaine particulier. Par exemple, la classe personne peut avoir deux sous-classes « male » et « femelle », ou on peut ajouter une relation « sexe » pour déterminer le sexe de la personne.

4- Modèle de recouvrement et granularité : Des modèles diffèrent par leur recouvrement d'un domaine particulier, et la granularité avec laquelle des distinctions sont faites. C'est la raison la plus forte pour la fusion des ontologies. Par exemple, une ontologie modélise les voitures mais pas les camions, une autre représente les camions sous des catégories.

5- Paradigmes de représentation : différents paradigmes sont utilisés pour représenter des concepts comme le temps, l'action, le plan, la causalité.
6- Systèmes d'inférence : une raison pour que des modèles semblent différents est due à leur construction pour produire des inférences désirées avec un moteur d'inférence particulier.

Enfin, nous constatons la classification des types d'hétérogénéité entre les ontologies en deux classes : les discordances liées à la syntaxe, et celles liées à la sémantique. Ces dernières sont beaucoup plus complexes, et leur résolution nécessite de grandes recherches. Cependant, cette approche ne prend pas en considération l'étude du langage OWL.

12. Conclusion

Dans ce chapitre, l'ontologie est présentée comme un outil essentiel pour représenter le monde réel par la conception des concepts de ce monde dans un modèle représentatif, décrit par les classes y existantes, et les propriétés qui définissent les relations établies entre ces concepts. Le standard OWL s'est montré capable de jouer un rôle du pivot, grâce à sa compatibilité avec d'autres langages DAML+OIL, RDF, et RDFS.

Les approches d'intégration des ontologies nous ont apporté des points forts sur l'étude structurelle et les mécanismes de « mapping ». Cette intégration cause l'apparition des problèmes d'hétérogénéité entre différentes ontologies dans un même domaine. L'étude de ces types de problèmes nous a rendu conscient des difficultés que nous pouvons rencontrer durant notre travail.

Dans lechapitre suivant, nous allons présenter les standards d'échange tel que : EDI, BizTalk, RosettaNet, et ebXML.

CHAPITRE 3

Les Web services : Définition et Composition

1. Introduction

Ces dernières années, qui ont vu la standardisation d'XML par W3C, son adoption rapide par une très grande partie des acteurs de l'industrie informatique et sa déclinaison non moins rapide à travers une variété considérable d'application et de scénarios d'usage, marqueront un tournant dans l'industrie des logiciels. C'est peut être, avec le développement du langage de programmation Java et la généralisation d'UML, l'un des exemples les plus frappants de l'accélération changement de nature de la diffusion des innovations technologiques dans le secteur du logiciel. Il est encore difficile de distinguer les facteurs techniques de ceux à teneur économique et marketing sous-tendant cette récupération rapide des standards des protocoles du Web par l'informatique d'entreprise.

Les méthodes dites Web services proposent une méthodologie et un éventail de protocoles standardisés et ouverts permettant à des composants logiciels distribués d'interopérer via Internet. Les Web services, apparus avec la mouvance Internet et le développement rapide des applications client/serveur accessibles via un navigateur Web, se veulent un aboutissement de la logique des applications distribuées.

Ce concept peut être analysé comme une réponse au désir des entreprises et des administrations de relier différents composants logiciels internes et externes afin, notamment, d'offrir de nouveaux services à leurs clients ou usagers. Ils ont été crées à l'origine pour faciliter les relations B to B entre les entreprises et doivent donc se fonder sur des protocoles ouverts et interopérables.

Tandis que l'EAI se fonde sur une centralisation des flux et tente de gommer le point à point entre les applications, les Web services se fondent sur une communication point à point avec un langage standardisé et ouvert [1].

Les Web services sont des applications qui relient des programmes, des objets, des bases de données ou des processus d'affaires à l'aide de XML et de protocoles Internet standard. Les Web services sont des compléments aux programmes et applications existantes , développées aux langages tel que Visuel basic , C ,C++, C #(C Sharp), Java ou autre, et servent de pont que ces programmes communiquent entre eux .

Les Web services définissent non seulement les données transmises entre deux applications, mais aussi comment traiter ces données et le relier à l'intérieur et à l'extérieur d'une application logicielle sous-jacent [2]. De ce fait, les Web services permettent aux entreprises et individus de publier des liens vers leurs applications de la même manière qu'ils publient des liens vers leurs pages Web. Conséquemment, les Web services peuvent faire en sorte que toutes les ressources informatiques dont une entreprise à besoin soient des ressources distribuées à la grandeur de l'Internet [3]. Contrairement à la plupart des applications de type client/serveur, les Web services ne fournissent pas d'interface usager [4]. Ils sont utilisés afin d'envoyer des données destinées à être lues par des machines [5].

Cependant, les programmeurs peuvent tout de même développer une interface graphique pour l'utilisateur, auxquels ils pourront ajouter une panoplie de Web services afin de personnaliser une page Web ou pour offrir une fonctionnalité spécifique à des utilisateurs.

2. Eléments de définition

Plusieurs acteurs définissent les Web services par des caractéristiques technologiques distinctives. D'un point de vue technologique par :
« Une application logicielle, légèrement couplée, à interaction dynamique, identifiée par un URI, pouvant interagir avec d' autres composantes logicielles et dont les interfaces et liaisons ont la capacité d'être publiée, localisée et invoquée via XML et l'utilisation des protocoles Internet communs. Ils sont les bases permettant de construire des systèmes distribués et ouverts sur Internet, utilisant des technologies indépendantes des plates-formes » [4].

D'autres éléments principaux apparaissent importants :

- Une application logicielle identifiée par un URI (Uniforme Resource Identifier) : Un Web service est une application autosuffisante en ce sens qu'elle effectue une seule tache et que ses composantes décrivent ses propres entrées et sorties de telle sorte que d'autres logiciels qui invoquent le Web service puissent interpréter ce qu'il fait, comment invoquer sa fonctionnalité et à quel résultat cet autre service peut s'attendre. Les Web services sont, en quelque sorte, le prolongement de la programmation objet. Ainsi, un Web service est donc un objet avec une seule fonctionnalité permettant, avec d'autres Web services la composition d'une application plus large pouvant avoir plusieurs fonctionnalités. En fait, c'est l'une des briques d'un mur qui en comporte plusieurs. Un URI est la façon d'identifier un point de contenu sur le Web, que ce soit un fichier texte, audio ou vidéo. L'URL le plus connu est l'adresse d'une page Web, par exemple : http://www.univ-constantine.dz. Le Web service est donc accessible en spécifiant son URI.

- Capacités des interfaces et liaisons (bindings) d'être publiées, localisées et invoquées via XML. Un Web service peut être publie dans un registre situé à l'intérieur ou à l'extérieur d'un SI. Un Web service peut être localisé en interrogeant le registre qui l'héberge. Une fois localisé, un Web service peut être invoqué (même par un autre Web service) en envoyant une requête appropriée.

- Capacité d'interagir avec d'autres composantes logicielles via des éléments XML et utilisant des protocoles de l'internet. L'une des bases des Web services est l'utilisation de protocoles standard de l'Internet tels que HTTP(Hypertext Transfer Protocol, le protocole de Web), SMTP(Simple Mail Transfer Protocol, le protocole du courrier électronique) et XML. Contrairement à une page Web ou à une application de bureautique, les Web services ne sont pas destinés à une interaction humaine directe. Ils sont plutôt conçus pour être utilisés par d'autres logiciels.

- Composante logicielle légèrement couplée à interaction dynamique. Me couplage léger signifie que le Web service et le programme qui l'invoque peuvent être modifiés indépendamment l'un de l'autre. Cela veut aussi dire que, contrairement à une composante logicielle qui serait fortement couplée, il n'est pas nécessaire de connaître la machine, le langage, le système d'exploitation ou tout autre détail, pour qu'une

communication puisse avoir lieu. Cela offre une flexibilité qui permet aux entreprises d'éviter les coûts engendrés par l'EDI, par exemple.

3. Fonctionnement

Pour que deux applications distantes qui ne se connaissent pas au préalable puissent se communiquer, il faut réunir plusieurs conditions :

- Elles doivent être reliées par un réseau de communication standardisé.
- Elles doivent disposer d'un langage commun d'interrogation et de dialogue.
- Elles doivent être à même de se localiser les unes les autres.

Le réseau de communication basé sur l'Internet Protocol avec tous les protocoles associés tels que TCP et HTTP.

Le langage commun utilisé pour que les applications échangent des données est XML ; Cependant, tout cela ne suffit pas car pour que deux applications ne se connaissant puissent s'invoquer et échanger des données, il est nécessaire qu'elles puissent établir les termes et les paramètres de leurs dialogues. Pour cela, des technologies fondamentaux sont utilisés : SOAP, WSDL et UDDI qui seront détaillés dans la section suivantes [1].

4. Technologies des Web services

XML, SOAP (Simple Object Language) et UDDI (Universal Description Discovery and Integration) sont les technologies dominantes des Web services. Une pléthore d'autres technologies viendront, aux fil du temps, garnir l'architecture des Web services. Sans entrer dans tous les détails techniques, nous vous exposons ici les grandes lignes de chacune de ces technologies et de celles, qui à court et à moyen terme, risquent de se positionner à la suite [4].

Fondamentalement, SOAP, WSDL et UDDI sont des technologies issues de l'intérêt parmi des membres de la communauté Internet à développer un mécanisme pour échanger des documents WML sur le Web entre SI.

4.1. Langage XML

XML (eXtensible Markup Language) est une famille de technologies développées au sein du W3C (Worle Wide Web Consortium). La première spécification de XML est apparue en Février 1998 et se concentre sur les données, contrairement à HTML qui focalise sur la présentation. XML permet donc de transformer Internet d'un univers d'information et de présentation de sites Web statiques à un univers WEB programmable et dynamique, centré sur les données.

XML est largement utilisé par les entreprises et supporté par les manufacturiers informatiques. Il est indépendant des plates-formes informatiques. Il est lisible par l'humain mais est destiné à être lu par la machine.

XML se rapporte aux spécifications qui environnent les types de documents de la clientèle en utilisant un format lisible. Les spécifications XML renforcent les règles des documents,

comme n'importe quel langage humain par la manipulation d'un alphabet, des mots et d'une ponctuation.

Les Web services communiquent via XML dans le but de décrire leurs interfaces et encoder leurs messages.

Contrairement aux formats de fichiers dits binaires, des documents sortant bruts de fonderie des traitements de texte ou des tableurs, le format XML a été conçu pour permettre le déchiffrement direct par l'utilisateur comme par les programmes. XML permet de définir des balises et de leur associer une interprétation. Elles permettent d'associer toute sorte d'informations au fil du texte. Le compromis trouvé dans XML est celui de texte contenant des balises ouvrantes et fermantes qui décrivent la nature des données qu'elles encadrent.

En plus, les protocoles de communication dans les Web services utilisent les interfaces et les messages d'XML, que n'importe quelle application peut interpréter, pour définir les interactions.

4.2. Le protocole SOAP (Simple Object Access Protocol)

Le SOAP est un protocole de communication entre applications par les échanges de messages en XML à travers le Web, il est indépendant des langages de programmation ou des systèmes d'exploitations employés pour l'implémentation des Web services. Il est considéré comme la technologie la plus importante des Web services [2].

Les concepteurs ont préservé la plus grande généralité dans la représentation en XML des principes des protocoles de communication. SOAP vise à servir des protocoles de communication sur les Intranets, dans un optique d'intégration d'applications d'entreprise, et à permettre la communication entre applications et Web services dans un contexte d'échanges interentreprises. Ce protocole ne demande pas de modifications majeures aux serveurs Web déjà installés. Il permet de s'interfacer avec des applications préexistantes grâce au support d'XML schéma, en capturant leurs structures de données quelle que soit la complexité. En plus, SOAP propose un mécanisme simple de représentation des différents aspects d'un message entre applications, donc il peut être utilisé dans tous les styles de communication, Intranet ou Internet.

La spécification SOAP est divisée en quatre parties. L'enveloppe SOAP qui est définit le contexte d'un message, sa destination, son contenu et différentes options. Les règles de codage SOAP, définissent la représentation des données d'une application dans le corps d'un message SOAP. Le protocole RPC (Remote Procedure Call) qui définit la succession des requêtes et des réponses et l'utilisation de http comme couche de transport des messages SOAP.

Un message SOAP est un document XML constitué d'une enveloppe contenant un en-tête (header) facultatif et le corps (body) du message [8] comme apparu dans le schéma suivant :

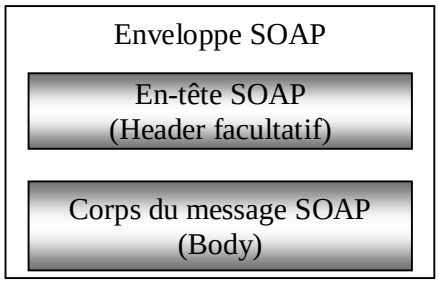


Figure 3. 1 : Structure du message SOAP [6].

- L'enveloppe SOAP : représente la racine du document contenant le message SOAP, marquée par la balise (Enveloppe). La spécification impose que tous les attributs de cette balise et de celles imbriquées soient explicitement associées à un en suppriment toute ambiguïté.

  Par convention, la spécification SOAP à deux namespaces fréquemment utilisés :

– SOAP-ENV associé à l'URI pour définir le message de l'enveloppe.

– SOA¨-ENC associé à l'URI pour définir les formats de types de données.

- L'en-tête SOAP (SOAP-ENV : Header) : la balise (Header) permet de passer dans le message des informations complémentaires sur ce même message. Cet élément est facultatif, mais s'il est présent, doit être le premier dans l'enveloppe SOAP du message. L'en-tête peut contenir par exemple des informations authentifiant l'émetteur ou bien encore le contexte d'une transaction dont le message n'est qu'une des étapes. Pour les couches de transport qui ne fournissent pas, à l'émission, d'adresse de retour, on peut utiliser l'en-tête pour identifier l'émetteur du message.

- Le corps du message SOAP (SOAP-ENV : Body) : il contient un ou plusieurs sous éléments qui sont :

– FAULT : indique une erreur ou une défaillance en réponse à une requête.

– Des données pour le destinataire du message, à un format défini par les règles de codage SOAP.

4.3. Langage WSDL

WSDL (Web Services Description Langage) est un langage basé sur XML. Il permet de décrire les composants des Web services nécessaires au protocole SOAP et à l'interaction avec un autre service.

Un document WSDL décrit quelles sont les fonctionnalités qu'offre un Web service, comment communique-t-il et par où est-il accessible. WSDL pourvoi un mécanisme structuré pour décrire les opérations qu'un Web service peut remplir, les formats des messages qu'il peut traiter, les protocoles qu'il supporte et les points d'accès à une instance d'un Web service.

On peut comparer WSDL aux langages de description requis par CORBA et DCOM [9]. Cependant, WSDL est différent de ceux-ci en ce sens qu'il est tout aussi neutre du point de vue du protocole que du point de vue de l'implantation.

Un document WSDL possède sept sous éléments distincts, divisés en deux parties. Une première partie réutilisable, appelée abstraite, détaille le service. Une seconde partie non réutilisable, appelée concrète, indique la localisation du service. La partie abstraite est composée de cinq éléments : document, import, types, messages et port type. La partie concrète comporte deux éléments: binding et service [7].

- Document : cet élément permet simplement d'écrire des commentaires.
- Import : cet élément rend possible, si, nécessaire, l'import de documents XML. Deux attributs doivent être connus. Le premier est l'espace de noms du document à importer qui permet de faire le lien entre ce dernier et le document WSDL. Le second est l'URI de localisation du document à importer.
- Types : permet de définir les structures de données contenues dans les messages.
- Message : les éléments message définissent l'ensemble des messages qui seront échangé.
- Port Type : concerne la mise en place d'opérations, dont chacune est définie par une opération et les messages d'entrée et de sorties associées.
- Binding : définit les protocoles de communication utilisés lors des appels du Web service.
- Service : permet de décrire les points d'accès du Web services.

4.4. Annuaire UDDI

Un UDDI (Universal Description, Discovery and Integration) définit les mécanismes permettant de répertorier des Web services. Ce standard régit donc l'information relative à la publication, la découverte et l'utilisation d'un Web service. En d'autres mots, UDDI détermine comment nous devons organiser l'information concernant une entreprise et les Web services qu'elle offre à la communauté afin de permettre à cette communauté d'y avoir accès. En fait, UDDI définit un registre des Web services sous un format XML. Ce registre peut être public, privé ou partagé.

Tout comme WSDL, UDDI est une création du trio IBM, Microsoft et Ariba. Au départ ils ont développé la spécification puis ont rassemblé quelques 300 entreprises sous le chapeau de l'organisation UDDI.org afin de continuer le développement et de légitimer leurs efforts. UDDI a été déposé à OASIS [11] (Organisation for the Advancement of Structured Information Standards) en juillet 2002 [12] afin de permettre à cet organisme de standardisation de parrainer la spécification et d'en assurer son développement technique de façon indépendante. OASIS officialisé son implication en créant le OASIS UDDI Specification Technical Commitee en août 2002. En septembre 2002, il existait trois registres publics (aussi appelés URB ou Universal Business Registry), avec chacun son nœud de registre d'affaires et son nœud de tests. Ils sont entretenus et offerts par IBM, Microsoft et SAP.

Une entreprise choisit le répertoire dans lequel elle désire déposer les informations pertinentes à ses Web services. Pour pouvoir publier ou rechercher un Web service, des messages SOAP ont été définis. Ainsi, SOAP peut être utilisé pour publier (Publisher API) et localiser (Consumer AIP) d'autres services SOAP. Les entreprise et individus utilisent ces interfaces définies avec SOAP, ou encore l'interface utilisateur fournie par le propriétaire du registre UDDI en question, afin de s'enregistrer ou d'accéder aux informations du registre.

L'architecture de UDDI est représentée par la figure suivante dont les différents composants sont des documents XML.

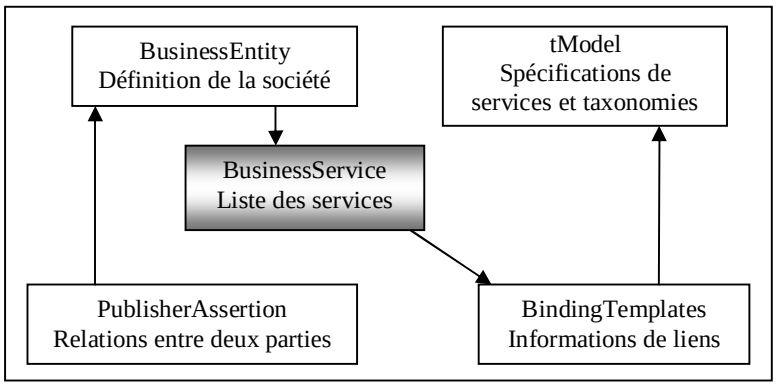


Figure 3. 2: Entités composant un annuaire.

- Affirmation d'éditeur (Publisher assertion): cette partie est facultative, elle permet de décrire l'organisation dans son intégrité si elle est composé de plusieurs divisions. toutes les parties de l'organisation et leurs liens sont décrites dans ce module.

- Entité commerciale (business entity) : les informations concernant l'organisation sont contenues dans cette entité. Toutes les informations supplémentaires nécessaires à l'identification de l'entreprise sont répertoriées dans ce document. Afin de symboliser cette organisation, l'entité commerciale lui attribue un identifiant unique UDDI décrit par l'élément Business Key.

- Offres de service (Business service): ce composant distingue les services proposés par l'organisation. Cette offre de service est identifiée par un nom et un UDDI décrit par l'élément Service Key. Les points d'accès (Binding Template) à ce service sont énumérés dans le Business Service par des liaisons UDDI.

- Liaisons UDDI (Binding Template): ce module décrit les points d'accès au Web services (URL) et le moyen d'y accéder.

- Types de services (tModel): permet d'associer un service à sa description en WSDL.

5. Avantages des Web services

L'idée fondamentale derrière les Web service est de morceler les applications et les processus d'affaires en morceaux réutilisables appelés service de telle sorte que chacun de ces segments effectuent une tache distincte. Ces services peuvent alors servir à l'intérieur et à

l'extérieur de l'entreprise, facilitant l'interopérabilité entre tous ces services. Par leur nature, les Web services [13]:

- Permettent à des portions de logiciels écrits dans différents langages, ou évoluant sur différents systèmes d'exploitation, de communiquer entre elles facilement et à peu de frais;

- Permettent à des applications supportant différents processus d'une organisation ou de différentes organisations, de communiquer entre elles et /ou d'échanger des données facilement et à peu de frais.

Plus spécifiquement, les Web services devraient permettre aux entreprises [14] de:

- Donner aux clients un accès direct à l'information, aux données et aux fonctionnalités dont ils ont besoin pour interagir avec une entreprise;

- Donner aux partenaire d'une entreprise un accès direct à la fonctionnalité dont ils besoin pour mieux servir les clients qu'ils ont en commun avec cette entreprise;

- Donner aux fournisseurs d'une entreprise un accès direct à l'information et à la fonctionnalité dont ils ont besoin pour leur permettre d'ajuster les inventaires ;

- Intégrer des applications de bout en bout, de manière abordable, facile à implanter autant hors des frontières de l'entreprise qu'à l'intérieur de celle-ci;

- Avoir des équipes de développement travaillant indépendamment et efficacement sur des systèmes qui interagiront, parce que ces équipes cherchent à développer des interfaces communes plutôt que d'avoir à synchroniser les processus;

Les Web services offrent donc aux entreprises la flexibilité de réponse et d'anticipation de besoin changeant des clients, la rationalisation des infrastructures logiciels et la flexibilité d'interaction et de configuration des alliances externes avec les partenaires et fournisseurs.

Dans ce chapitre nous avons vu des éléments de définition des Web services et les technologies les plus importantes qui les constituent, en l'occurrence : XML, SOAP, WSDL et UDDI, en plus au protocole de communication universels tel que HTTP. Nous avons vu aussi les avantages des Web services par rapport aux entreprises.

6. Stratégies de composition

La composition de services permet de combiner des Web services élémentaires afin d'obtenir des services plus élaborés.

La composition décrit un ensemble d'interactions ou *processus métier*, faisant intervenir différents Web services. Elle est décrite indépendamment de son implémentation future i.e. elle indique uniquement les types de Web services nécessaires (Web service bancaire par exemple) mais ne précise pas nominativement les Web services qui seront utilisés (Web service de la banque nationale Algérienne). Par ailleurs, la composition peut comporter

plusieurs niveaux en permettant à des Web services élaborés d'être à leur tour combinés pour construire de nouveaux services.

6. 1 Concepts de base

La définition d'une application Web repose sur la description du flot de données et du flot de contrôle unissant les Web services auxquels elle fait appel. Comme dans le modèle des objets ou des composants logiciels, une application Web est « assemblée » à partir de Web services techniques et métier. Dans le cas du Web, on parle de « composition », d'« orchestration » ou même de « chorégraphie » de services.

6. 1. 1 Composition

C'est une relation ou un accord avec l'implémentation d'une application (offerte comme un service) qui, sa logique nécessite l'invocation d'une opération offerte par d'autres services.

- Le nouveau service est appelé le service composite : *Composition :* construction de spécification du service composite (i.e., schéma de composition)
- les services invoqués sont appelés les services composants

La composition peut être soit manuelle ou automatique ;

- le schéma de composition spécifie le processus du composite service, c'est la notion du Workflow du service
- des clients différents, par l'interaction avec le service composite, aboutissent à leurs besoins spécifiques et enrichissent leurs buts ;
  L'instance d'orchestration c'est l'exécution spécifique du schéma de composition pour une donnée client.

6. 1. 2 Chorégraphie

C'est la coordination d'une conversation de N services, c'est la spécification globale de la conversation de N paire de services, la chorégraphie concerne l'exécution et la description d'un model global (N paires), voici des exemples qui illustrent la chorégraphie :

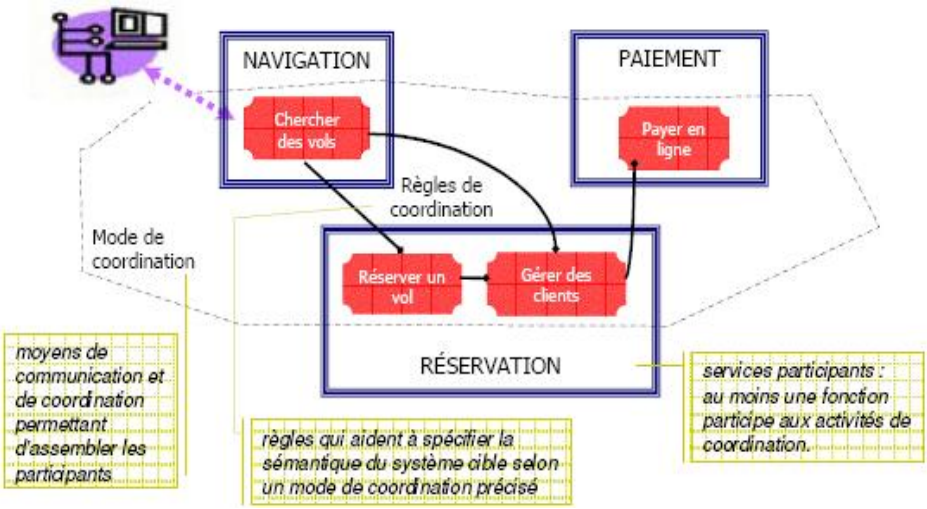


Figure 3.3 : chorégraphie des services.

6. 1. 3 Orchestration

Le déploiement d'un grand nombre de Web services posera rapidement le problème de la gestion de la cinématique d'appel des différents Web services et du rapprochement des définitions sémantiques des objets métiers manipulés pour supporter des processus métier complexes de bout en bout. Orchestration est utilisée pour palier à ce problème.

Orchestration : la prise en charge du temps d'exécution du service composite (invocation d'autres services, prévoir les différentes étapes etc…)

- composition schéma c'est le programme conçu pour être exécuté
- ressemblance avec le WFMS (Workflow Management Systemes)

L'orchestration concerne l'exécution et la description d'une seule paire.

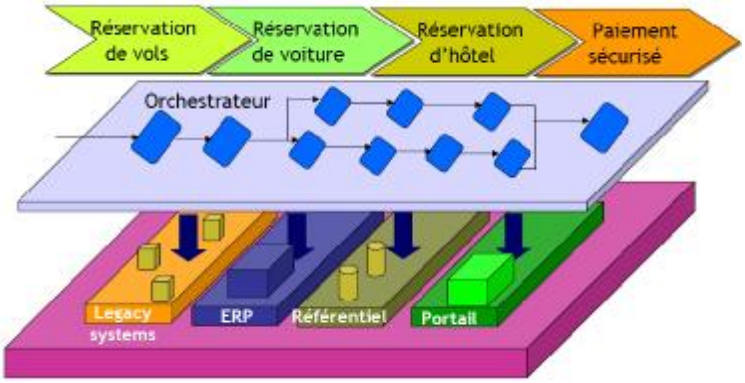


Figure 3. 4 : Orchestration des services.

L'Orchestration a des problèmes de sécurité, nous énumérons quelques uns :

1. **Confidentialité** *:* les informations sensibles ne sont pas révélées aux usagers non autorisés ;
2. **Intégrité** *:* les données et les activités de coordination sont modifiées ou détruites seulement d'une manière contrôlée et sécurisée ;
3. **Inobservabilité** *:* l'usage d'une ressource ou d'une fonction n'est observé que par celui qui l'utilise.

En plus de cela l'orchestration est vulnérable, car on peut :

- modifier des messages entre des participants ;
- voler des informations dans les messages entre des participants ;
- réutiliser des messages pour accéder au système.

La sécurisation d'orchestration porte sur les ressources et les transactions :

a) sécurisation des ressources :

- Sécurisation de l'espace partagé : authentification des services pertinents
- Sécurisation de l'objet partagé : autorisation de l'accès aux objets

b) sécurisation des transactions :

- Protocoles sécurisés dans les environnements non sécurisés : vérification / chiffrement des informations échangées
- Environnements sécurisés : authentification des services émis /reçus, vérification / chiffrement des informations échangées

6. 2 Dimensions clés de la composition

La composition de Web services est fondée sur deux activités différentes : l'orchestration et la chorégraphie. L'orchestration de services fait référence à l'activité de création de processus, exécutables ou non, qui utilisent des Web services. La chorégraphie se concentre sur les séquences de messages échangés entre différents acteurs (typiquement l'échange public de messages entre des Web services) plutôt qu'un processus spécifique exécuté par un acteur en particulier. Actuellement il existe différents standards permettant de réaliser l'orchestration et la chorégraphie, par exemple BPEL4WS, WSCI et BPML [9].

- ü Statique de la composition du système : ontologies des services (pour le partage sémantique de donnée/ information), entrées et sorties, etc…
- ü Dynamique des services composants : formes complexes des flux de données, attitudes transactionnelles, adaptabilité pour la variation des circonstances.
- ü Dynamique du service cible :

Actions atomiques — Le service cible exposé comme :

- une seule étape
- (série de) étapes séquentielles
- (série de) étapes conditionnelles
- pendant/ boucles ; exécution en batch
- pendant / boucles ; mise en marche sous un control externe

processus

- ü Degré de complétude dans la spécification de :
  - aspects statiques de la composition du système
  - aspects dynamiques des services composants
  - spécification tragique du service

6. 2. 1 Travaux de Papazoglou [41]

- *Services disponible : I/ O interfaces*
  - service composant *:* service simple, ou complexe pré-existant est enveloppé dans un Web composant
  - ils sont sauvegardés dans un service composant de classe bibliothèque
  - Les opérations sont offertes à travers une interface
- *Service composite :* description de comportement complexe
  - une série de services composants associés ensembles par des logiques composition (définit l'ordre d'exécution séquentiel

ou concurrent, des services composants …

- support pour une composition manuelle : le concepteur spécifie le service composite en utilisant le SSL (Service Scheduling Language) et le SCEL (Service Composition Execution Language)

6. 2. 2 Travaux de Bouguettaya [30]

- Services disponibles : actions atomique
  - sémantiquement décrit en termes de leur interfaces d'E/ S et les propriétés non fonctionnelles comme leurs but, leurs catégorie, et leur qualité.
  - Les services disponibles sont stockés dans une ontologie sue la base de leurs propriétés non fonctioneelles
- Requête client :
  - définit dans le CSSL (Composite Service Specification Language) : elle spécifie la séquence des opérations désirées que le composite service doit performé et contrôlé le flux entre opérations.

- *Le problème de la composition d'un service :*
  - Entrés :
    i. Description d'E/ S des services disponibles
    ii. Expression de la requête client avec CSSL
  - Sorties :

    le service composite est comme une séquence d'opération (Semi-automatiquement) obtenues à partir de la spécification du client en identifiant, pour chaque opération, les opérations du service disponible qui l'égale, sur la base de l'interface E/ S et les caractéristiques non fonctionnelles.

6. 2. 3 Travaux du groupe Romain [174]

- Service disponible : description comportementale
  - Service est comme un programme inter-acitf : à chaque étape, il présente le client avec une série d'actions parmi lesquelles on choisi le prochaine à exécuter.
  - Le choix du client dépend sur le résultat des actions exécutées précédemment, mais le rationnel derrière ce choix dépend entièrement du client
  - Le comportement est modélisé par un système de transition à état fini, chaque transition est traitée par une action (atomique) déterministe, vue comme l'abstraction des messages E/ S efficaces et les opérations offertes par le service.

- La requête client :
  - Une série d'exécutions organisées comme un système de transition pour les activités qu'il s'intéresse à faire.

- *le problème de la composition de service :*
    - Entrés : système de transition d'état fini d'un service disponible et système de transition d'état fini d'un traget service

    - Sorties : (obtenus automatiquement) le composite service qui réalise la requête client, tel que chaque action du traget service est déléguée jusqu'à le dernier service disponible, en accord avec le comportement de chaque service.

7. Langages de spécification des processus métiers

La définition d'une application Web repose sur la description du flot de données et du flot de contrôle unissant les web services auxquels elle fait appel. On parle donc de composition, d'orchestration ou même de chorégraphie de services.

Plusieurs initiatives ont vu le jour qui visent à spécifier la composition des web services sous forme de documents XML dont l'interprétation pilote alors directement l'exécution de l'application web elle-même. Ces documents XML s'appuient sur WSDL pour la description des services mis en jeu dans l'application et détaillent la succession d'appels aux opérations qui y figurent en foction des besoins de l'application.

7. 1 WSFL

WSFL (Web Services Flow Language) est une proposition de standard offrant deux tyles de composition de web services :

- La description explicite de la succession des étapes et de l'enchainement des appels aux opérations des web services, appelée processus métier ;
- Un modèle dintéractions de web services pris deux à deux, le contrat.

C'est le premier cas qui se rapproche le plus de la procedure au sens des langages de programmation. WSDL utilise l'expression modèle de flux pour désigner ce style de spécification qui s'apparente à la programmation dans un langage de script. Le second cas, appelé modèle globale dans WSFL, décrit une collection de liens entre les opérations de web services, prises deux à deux sans indiquer de structure de contrôle explicite. La métaphore employée ici est celle du contrat liant liant deux parties dans l'univers commercial.

WSFL permet de créer des modèles recursifs : une composition de web services est elle-même considéré comme un web service, utilisable à son tour dans d'autres compositions.

Dans WSFL, les étapes du processus sont appelées les activitées et sont reliées par les liens de contrôle (passage explicite du contrôle d'un processus élementaire à un autre) et des liens d'information (passage de données avec transformation eventuelle, d'un processus élémentaire à un autre). Pour chaque activité, on indique l'identité du web service responsable de son execution et les opérations à appeler grace aux élements <Export> et <PlugLink>. Les activités sont sous la responsabilité d'un rôle qui se déroule dans un espace de travail spécifique, appelé couloir [9].

7. 2 XLANG

XLANG (Web Services for Business Process Design) est le premier langage implementé dans le produit BizTalk de Microsoft. En tant que langage propriètaire à l'usage des utilisateurs de BizTalk, XLANG n'a pas vocation de standard. Toutefois, Microsoft a publié une spécification de XLANG, qui a fortement influencé la spécification de BPEL.

XLANG permet de représenter en XML l'orchestration d'activités qui constituent un procesus métier. Motivé par l'emergence des standards de facto comme SOAP et WSDL, XLANG est un format intermédiaire de stockage de l'environnement de developpement BizTalk Server ; ces fichiers XLANG, encore appelés schedules, sont exécutés par le serveur BizTalk en phase de production.

Plus précisément, XLANG comme WSFL d'IBM mais avec une division différente des tâches, couvre les points suivants :

- Flot de contrôle séquentiel et parallèle ;
- Transactions longues ;
- Corrélations des messages entre eux ;
- Gestion des défaillances et des erreurs ;
- Découverte dynamique des services ;
- Contrats multipartites.

XLANG s'appuie sur WSDL en réutilisant un certain nombre des concepts définis dans cette dernière norme. XLANG reprend, en particulier, totalement la description WSDL d'un service en termes de groupe de ports et de liaisons à des protocoles de transport, chaque port étant constitué à son tour d'opérations caractérisées par un échange de message [9].

7. 3 BPEL4WS

BPEL4WS est un langage de composition proposé par IBM, Microsoft et BEA. Il remplace les précédents langages de workflow WSFL et XLANG. BPEL4WS correspond à une grammaire XML qui décrit des processus métiers. Ceux-ci peuvent être interprétés et exécutés par un moteur d'orchestration (BPWS4J par exemple.).
BPEL4WS est utilisé pour décrire les processus métier exécutables « besiness process »ainsi que les processus abstraits

Les processus abstraits sont utilisés pour créer les spécifications comportementales des messages mutuellement échangés entre les différentes parties transactionnelles exécutant le processus métier.

7. 3. 1 Fichier BPEL4WS

Structurellement, le fichier BPEL4WS décrit un workflow en déterminant quels sont les participants, que les services doivent implémenter pour être dans le bon endroit, et le flux de control du processus.
Le BPEL4WS process model est construit à base de WSDL 1.1 service model et assume toutes les primitives des actions décrites en WSDL portTypes. La BPEL4WS description décrit la chorégraphie d'un ensemble de messages qui sont tous décrits par leur WSDL définitions. C'est important de faire savoir que WSDL est aussi utilisé pour décrire l'interface externe au workflow. Cela permet au BPEL4WS d'être en matière de composition complet,

cela veut dire que la composition des Web services est exposée comme un seul Web service éligible de participer dans d'autres compositions.

Exemple

La figure suivante présente le contenu d'un fichier WSDL pour une publication disponible Web service d'échange de devises.

- **Definitions** : Spécifie plus d'un service
- **Types** : La section qui donne des informations sur toute donnée de type complexe utilisée dans le document.
- **Message** : C'est une définition abstraite de l'information qui va être échangée. Ce Web service définie deux messages: "getRateRequest" et "getRateResponse".
- **PortType** : Donne une abstraction d'un ensemble d'opérations supportées par « endpoints ».
- **Operation** : Décrit l'action venant d'un service. Ce service a une opération nommée « GetQuote" qui prend le "getRateRequest" message et retourne le « GetRateResponse » message.
- **Binding** : Décrit comment l'opération est invoquée.

7. 3. 2 Structure générale d'un fichier BPEL4WS

```
<process>
    <partners>
        <partner/>
    </partners>
    <variables>
        <variable/>
    </variables>
    <faultHandlers>
      <catch/>

      <catchAll/>
    </faultHandlers>
  < !-- Workflow Definition   -->
</process>
```

Figure 3. 5 : Les sections primaires d'un fichier BPEL4WS.

La figure présente une squelette des sections primaires du fichier BPEL4WS.

- La section **process** spécifie un processus
- La section **partner** déclare les différentes parties qui participent dans le processus.
- La section **variable** définit les variables utilisées par le processus.
- La section **faultHandlers** pour capturer les erreurs. Toute erreur générée doit avoir un nom. Le fault handler définit les activités qui vont être exécutées quand une erreur est produite.
- **catch** capture une erreur spécifique
- **catchAll** //default handler for faults that are not specifically caught.

7. 3. 3 Types des activités BPEL4WS

Comme nous l'avons indiqué précédemment, un processus métier correspond à une séquence d'opérations ou plus exactement à un flux d'activités. Ces activités peuvent faire intervenir de un à plusieurs Web services. On peut distinguer les activités de base des activités structurées.

Les ***activités de base*** de ce langage permettent :

• D'invoquer une opération d'un Web service , activité ***invoke***,
• De présenter la composition comme un nouveau Web service avec l'activité ***receive*** pour décrire la réception d'une requête et l'activité ***reply*** pour générer une réponse.

Les ***activités structurées*** utilisent les activités de base pour décrire :

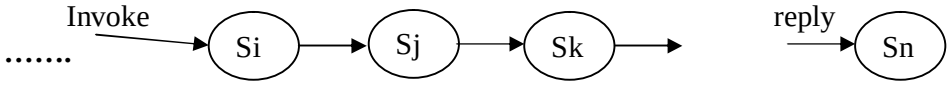
• Des séquences ordonnées ***sequence*** et des exécutions en parallèle ***flow***,
• Des branchements ***switch***, ***if*** et des boucles ***while***,
• Des chemins alternatifs ***pick***.

Les activités structurées peuvent à leur tour être combinées pour former une nouvelle activité structurée, celle de plus haut niveau correspondant au processus métier lui-meme.

7. 3. 4 Syntaxe et la sémantique des activités de BPEL4WS

Un processus Workflow est définit en BPEL4WS utilisant des :

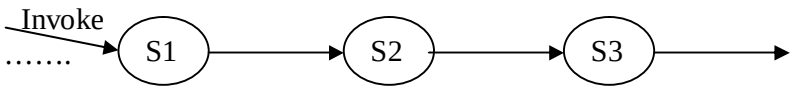
Primitives des activités
<invoke> invocation d'un WS
<receive> réception de l'invocation ; attente du message du service interface
<reply> génération de la réponse de l'opération d'in/output



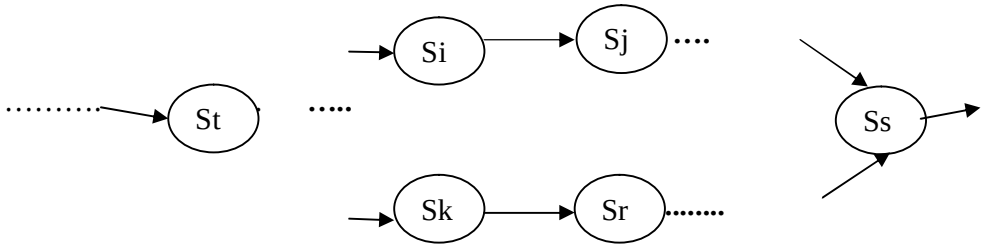
<wait> arrête l'exécution pour une période bien déterminée
<throw> utilisé pour détecter les erreurs
<catch> exception de capturation BPEL4WS définie le mécanisme pour "**catching"** et "**handling"** des fautes (comme Java)
<terminate> la fin de l'exécution du WS
<empty> pour ne rien faire.

Structures des activités
Peuvent être récursivement combinées

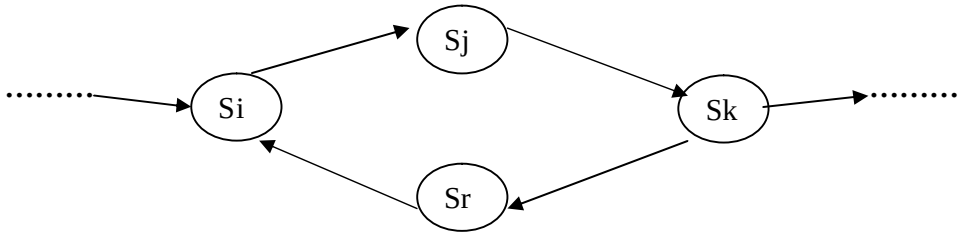
<sequence> spécifie que le contenu doit être exécuté dans l'ordre présenté;



<switch> permet l'exécution conditionnelle des activités (branchement comme case) ;

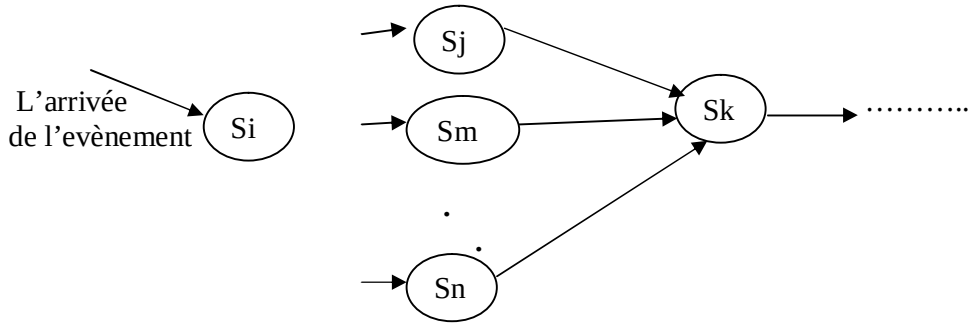


<while> indique que une activité doit être bouclée jusqu'à ce que le critère de condition est atteint (loop);

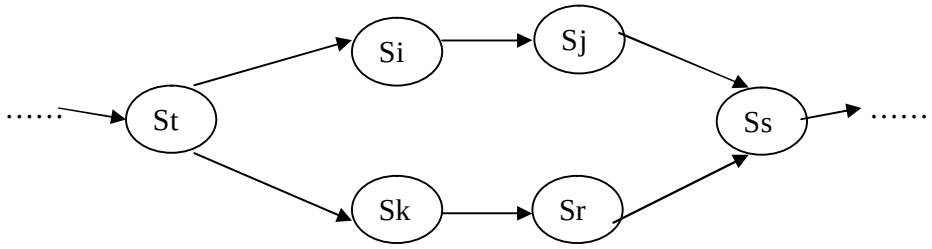


Le contrôle séquentiel entre les activités se fait par sequence, switch et while.

<pick> exécute une alternative parmi plusieurs. Le choix est non déterministe se base sur des événements externes (arrivé d'un message spécifié ou l'achèvement d'une durée de temps) quand l'un d'eux arrivent, les activités sont exécutées ;



<flow> spécifie que la collection d'étapes est exécutée en parallèle; (ordre d'exécution est définie par links). Concurrency and synchronization between activities is provided by flow;



<***links***> sont utilisés pour spécifier un ordre complexe des contraintes. Chaque lien a un *Source_ and _target_* constructeur. Le *target* est seulement exécuté après que la source ait fini.

7. 3. 5 Mécanismes du langage BPEL4WS

Le langage intègre également des mécanismes supplémentaires
Le mécanisme de gestion des exceptions (***throw***, ***catch***)
Le mécanisme de compensation (***scope***) qui permet d'annuler une transaction dans son intégralité lorsque celle-ci échoue

Il constitue une couche supérieure au langage de description WSDL. Il utilise, en effet, WSDL pour définir les opérations de Web services élémentaires à appeler et pour présenter le processus métier comme un nouveau Web service.
Exemple : Web service de réservation d'un voyage

Le client invoque le service de réservation en fournissant un message de demande qui contient ses coordonnées personnelles et bancaires et le séjour choisi. En retour, il obtient un numéro de réservation et recevra sa facture par courrier dans les jours suivants.

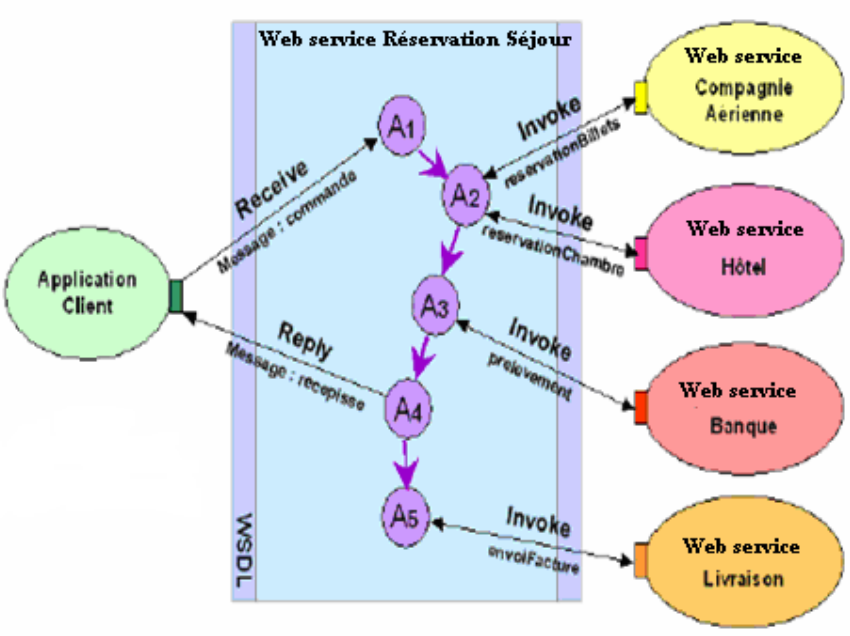


Figure 3. 6 : Modélisation du Web service de réservation d'un sejour avec BPEL4WS

Ce Web service de réservation peut être modélisé sous la forme d'un processus métier (figure) en combinant différents Web services élémentaires. Ainsi, à la réception de la demande de réservation, le processus métier invoque en parallèle le Web service d'une compagnie aérienne et celui d'une chaîne d'hôtels pour effectuer les réservations. Ensuite, il effectue le prélèvement bancaire auprès du Web service d'une banque et donne au client son numéro de réservation. Enfin, il invoque un Web service de livraison pour l'envoi de la facture papier au domicile du client.

La description BPEL4WS correspondante se décompose en trois parties

- Definition des partenaires :

<partners>
< partner name=”client” partnerRole=”ClientAgenceDeVoyage”>
< partner name=”WebserviceCompanieAerienne” partnerRole
=”FournisseurReservationBillets”>
< partner name=”Webservicehotel” partnerRole =”FournisseurReservationChambre”>
< partner name=”WebserviceLivraison” partnerRole =”FournisseurEnvoiFacture”>
</partners>

- Definition des types de données et des messages échangés:
<variables>
<variables name= “commande” messagetype =”typeCommande”>
< variables name= “recepisse” messagetype =”typeRecepisse”>
< variables name= “reservationBillets” messagetype =”typeReservationBillets”>
< variables name= “reservationChambre” messagetype =”typeReservationChambre”>
< variables name= “prelevement” messagetype =”typeprelevement”>
< variables name= “envoiFacture” messagetype =”typeEnvoiFacture”>

<variables/>

- Definition du processus métier
<sequence name=”seqReservationSejour” >
<receive partner=”client” operation “reservationSejour” variable= “commande”>
<flow>
<invoke partner=”webserviceCompaniesAerienne” operation =”reservationBillets”>
<invoke partner=”webservicehotel” operation =”reservationChambre” >
</flow>
<invoke partner =”webserviceBanque” operation = “prelevement”>
<reply partner =”client” operation=”reservationSejour” variable=”recepisse”>
<invoke partner =”webserviceLivraison” operation = “envoiFacture”>
</sequence>

8. Conclusion

Dans ce chapitre nous avons vu des éléments de définition des Web services et les technologies les plus importantes qui les constituent, en l’occurrence : XML, SOAP, WSDL et UDDI, en plus au protocole de communication universels tel que HTTP. Nous avons vu aussi les avantages des Web services par rapport aux entreprises.

Dans le prochain chapitre nous allons présenter les standards d’echange tel que EDI, BizTalk, RosettaNet et EbXML, leurs principe de collaboration et leurs objectifs.

Chapitre 5

Les standards d'échange et la Collaboration des Partenaires

1. Introduction

La croissance importante de l'Internet pendant les dix dernières années s'est traduire par l'émergence d'un nouveau marché, ce qu'on appelle le commerce électronique, qui constitue le premier domaine d'application des Web Services, porté par le web et les nouvelles catégories d'applications comme les places de marché, les portails, les échanger entre entreprises, etc. Plus rapide, moins cher, court-circuitant de nombreux intermédiaires traditionnels et plus proche du client, le commerce électronique a eu un effet considérable dans presque tous les secteurs industriels en redéfinissant les modèles d'activités, et en agissant comme facilitateur permettant un accès incomparable au marché libre.

2. Définition

En se référant à la traduction proposée par le site dictionary.com du mot commerce : « The buying and selling of goods, especially on a large scale, as between cities or nations » traduit par : « Les actes d'achat et de vente de marchandises sur une grande échelle comme entre les villes ou les nations » [79].

Le commerce électronique ou "e-commerce" désigne l'échange de biens et de services entre deux entités sur les réseaux, notamment Internet. Il peut être défini comme étant la rencontre sur un réseau international de télécommunications, d'une offre et d'une acception constituant entre l'émetteur et la récepteur un contrat de vente, de prestation de service, ou pour toute opération commerciale.
C'est un nouveau canal de distribution dans lequel l'achat et la vente se déroulent entièrement par voie électronique. Le grand avantage est que ce canal est disponible 24 heures sur 24 en continu, ce qui est rarement le cas des autres canaux de distribution.

L'e-commerce n'implique pas nécessairement le paiement et/ou la livraison en ligne [80]. Il concerne l'utilisation de supports électroniques pour des relations commerciales entre une entreprise et les particuliers:

- Publicité ;
- Présentation de catalogue ;
- Commande en ligne ;
- Paiement électronique ;
- Distribution.

Le client effectuant des achats sur Internet est appelé "Cyberconsommateur".

Nous confondons souvent les notions d'e-business et d'e-commerce. La notion d'e-business est plus vaste, elle couvre un plus large champ où la préoccupation principale est l´utilisation de la technologie Internet pour optimiser ou rationaliser l´échange d´information. [81]. Elle concerne la gestion électronique des processus internes, souvent invisibles pour le client. Le e-commerce est une sous activité du e-business (utilisation de la technologie Internet à des fins économiques).

3. Types de marchés

Selon les partenaires concernés par la transaction informatisée de marchandises ou de services, On peut distinguer :

- L'échange électronique entre entreprises (professionnels), souvent appelé B2B (se prononce *bi-tou-bi*), acronyme anglais de Business to Business : c'est une forme d'e-business qui concerne les relations inter-entreprises et qui se rapporte à la technologie de chaîne d'approvisionnement. C'est la forme de technologie e- business la plus répondue et la plus réussie de nos jours. Un exemple concret serait de connecter une entreprise par exemple une banque à ses partenaires afin d'avoir une visibilité sur les offres et les paramètres des services de ces derniers [82].

- Le commerce électronique à destination des particuliers, ou B2C (se prononce *bi-tou-ci*), acronyme anglais de Business to Consumer : est le nom donné à l'ensemble d'architectures techniques et logicielles informatiques permettant de mettre en relation des entreprises avec leurs clients (consommateurs finals) : en français, « des entreprises aux particuliers ». Il fait référence en général aux affaires commerciales dirigées d'une entreprise vers une cible de particuliers via un site marchand, type télé-achat [83]. Les services B2C sont rendus le plus souvent par des intermédiaires commerciaux, qu'ils soient de nouveau entrants. La différence principale entre cette forme et le B2B consiste dans le fait que le B2B les parties sont des associés connus et de confiance quant au B2C les parties sont généralement inconnus.

- Le commerce électronique entre particuliers, ou C2C (se prononce *ci-tou-ci*), acronyme anglais de Consumer to Consumer : dans cette forme d'e-business l'achat ou la vente des services se fait via un intermédiaire entre le consommateur vendeur et le consommateur acheteur. Il s'agit de sites web permettant la vente entre particuliers [83].

- Le C2B (se prononce *ci-tou-bi*), acronyme anglais de Consumeras to Business : représente une tentative de renversement de la logique des rapports entre demande et offre. Son principe de base est de s'appuyer sur les réseaux électronique pour consolider la demande des particuliers et mettre en concurrence les offreurs [84].

Dans la letterature, nous distinguons d'autres type du marché :

- L'échange électronique entre une entreprise et ses employés, souvent appelé Intranet ou B2E (se prononce *bi-tou-i*), acronyme anglais de Business to Employee : c'est une autre forme d'e- business qui se rapporte généralement à la demande des approvisionnements par des employés pour un usage relatif aux taches de travail [83].

- Le B2M (Business to Machines) : c'est une forme d'e-business qui commence à prendre une grande ampleur. L'idée de ce concept est de pouvoir connecter des entreprises et des machines via Internet. Un exemple serait une entreprise de gestion des machine ou

distributeurs de boissons. Avant d'effectuer l'alimentation des machines en boissons, l'entreprise demande aux machines d'envoyer les informations nécessaires pour une distribution plus efficace et plus rapide.

- Le B2A (Business to Administration) concerne l'utilisation de supports électroniques pour tout ou partie des échanges d'information entre des entreprises et des administrations publiques en vue de l'établissement et de l'exécution de marchés publics.

- L'échange électronique entre les entreprises privées et le gouvernement, souvent appelé B2G (se prononce ***bi-tou-dgi***), acronyme anglais de Business to government [83].

4. Standards d'échange

Depuis plus de 25 ans, l'Echange de Données Informatisé a présenté aux entreprises la perspective d'éliminer les documents papier, de réduire les coûts et d'améliorer leur productivité en échangeant de l'information sous forme électronique. L'échange de ses données informatisé n'est pas lié à un outil particulier, qu'il s'agisse d'un réseau et de protocoles de transport, ou qu'il s'agisse de syntaxes. IL est d'abord un effort d'organisation et l'important est la manière dont les relations peuvent s'établir et automatiser entre des acteurs qui communiquent. Les standards (EDI, Biz-talk, RosettaNet, et ebXML) facilitent les échanges dans le domaine d'e-commerce et plus précisément pour le B2B.

4.1 EDI

Depuis de nombreuses années, l'EDI (Electronic Data Intrechange) et la norme EDIFACT ont été utilisés pour standardiser les échanges de données dans les relations transactionnelles entre des grandes entreprises et avec certains de leurs sous-traitants. EDI est un outil au service de l'échange électronique consistant à transporter automatiquement de l'application informatique d'une entreprise à l'application informatique d'une autre entreprise, par des moyen de télécommunication, des données structurées selon des messages types convenus à l'avance.

Dans le monde EDI, il existe deux normes : ANSI X12 pour l'Amérique du Nord et EDIFACT (Centre for Trade Facilitation and Electronic Business) pour le reste. L'organisation OASIS et ONU ont réuni tous les groupes professionnels qui avaient développé des solutions entièrement propriétaires sur des réseaux privés, tous les acteurs d'Internet, pour concevoir un standard qui comprend à la fois une syntaxe, des dictionnaires de données et es regroupements de ces données en segments et messages. Les mêmes groupes se sont occupés des méthodes, des accords d'échange, des listes de codes des modèles d'affaires. C'est tout un mode EDIFACT qui s'est construit [85].

L'EDI vise à créer les conditions d'une interopérabilité entre des systèmes hétérogènes, afin que, dans la plupart des cas, ils puissent échanger sans intervention humaine. Cette interopérabilité n'est pas assurée par la conformité à une même syntaxe, ni aux mêmes protocoles de communication. L'environnement des échanges est tout aussi voire plus important. En fait, les éléments centraux de l'interopérahilité sont d'abord la communauté ou l'interopérabilité des annuaires, des référentiels et des outils de sécurisation [90].

Les messages EDIFACT sont des unités finales avec une signification entière de la communication. Ils peuvent remplacer les documents en papier dans la méthode

traditionnelle. Par contre, la structuration des messages est très variée en fonction des domaines d'activité.

L'accord d'échange fournit à chaque message un diagramme de structure, une table de segments comportant le statut, la répétition de chaque segment, une spécification des segments (le statut, le longeur, la forme de représentation). De plus, l'accord d'échange précise les méthodes à utiliser pour la transmission physique de données, les méthodes de codage et les problèmes juridiques liés au transfert de l'information.

4.2 BizTalk

BizTalk est une initiative de Microsoft, il est divisé en trois grands chapitres : un annuaire public d'objets et de processus métiers (à l'URL www.biztalk.org), une surcouche de SOAP pour l'échanges entre processus métier et une implémentation d'un serveur BizTalk et d'une suite d'outils graphiques de développement et de déploiement dans BizTalk server.

Les objectifs de BizTalk visent à faciliter les échanges électroniques entre systèmes d'information des entreprises via internet. Originellement, la vocation de BizTalk est à un emploi dans l'intégration d'applications d'entreprise. Néanmoins, la montée en puissance de

Windows sur marché des serveurs et de l'architecture. NET dans son ensemble laisse également envisager des usages extranets et Internet de BizTalk. Le scénario d'usage de BizTalk est classique : une application fait appel à un web service fourni par un serveur BizTalk, dit BFC (BizTalk Framework Compilant Server), par échange de documents formalisés en XML via le web.

Les éléments suivants sont ainsi définis dans BizTalk :

- **Le document BizTalk** : il s'agit d'un message dont le corps contient un document ayant trait à une transaction commerciale et l'en-tête des informations spécifiques à la nature de cette transaction.
- **Les BizTag** : ce sont des balises spécifiques à BizTalk, employées pour délimiter le contenu purement BizTalk dans les en-têtes des messages. Les BizTag sont reconnues par les serveurs BFC.
- **Les messages BizTalk** : ils permettent d'échanger des documents BizTalk entre serveurs BFC (avec éventuellement des pièces jointes).
- **La couche de transport** : Les messages BizTalk sont véhiculés par des messages SOAP.

L'architecture générale empile trois couches logiques : l'application, le serveur BFC et la couche de transport (cf. figure 4.1). Chacune des interfaces (application-BFC et BFC-transport) est spécifiée précisément, mais les implémentations sont laissées libres. L'application soumet ses documents à échanger avec d'autres applications au serveur BFC qui les « enrobe » de BizTag, les transformant en documents BizTalk. Ils sont ensuite routés via la couche de transport. Les opérations se déroulent, bien entendu, dans l'ordre inverse du côté de destinataire.

Les messages BizTalk sont échangés entre serveurs BizTalk selon une « orchestration » définie dans les outils de développement et elle-même sauvegardée dans un document XML

au format XLANG (format proposé par Microsoft pour représenter en XML l'orchestration d'activités qui est utilisé pour automatiser les processus métiers.

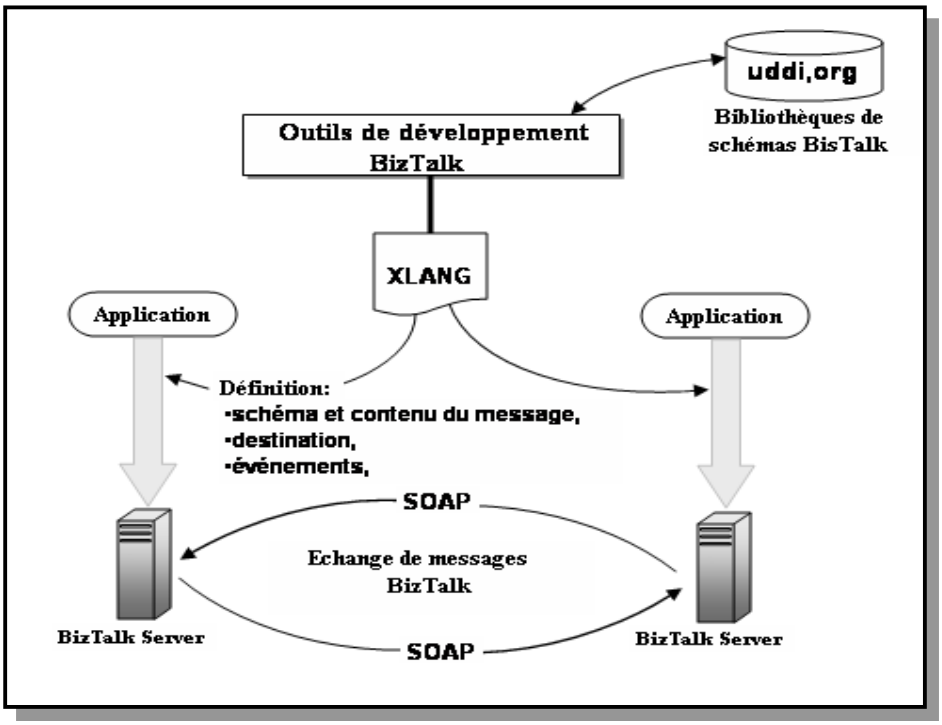


Figure 4. 1: Architecture technique BizTalk [86].

Un document BizTalk peut contenir plus d'un document applicatif : ceux-ci apparaissent alors à la suite les uns des autres. Entre les balises *Body* du message SOAP. Ces documents d'application peuvent également utiliser le mécanisme de références multiples de SOAP pour partager des éléments, le cas échéant.

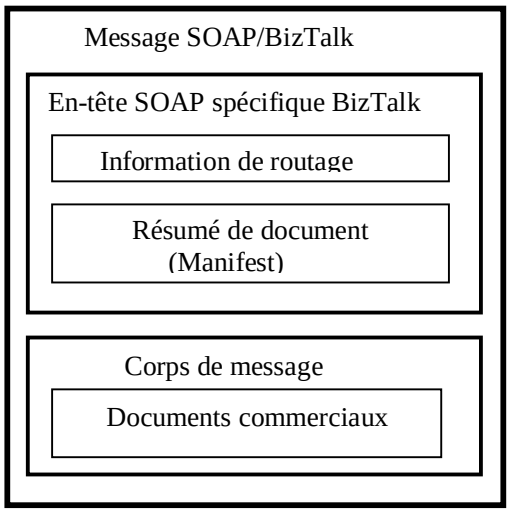


Figure 4. 2: Structure d'un document BizTalk [86]

4.3 RosettaNet

RosettaNet visent à simplifier la mise en place et la gestion des processus d'affaires pour les constructeurs, les éditeurs de logiciel, les grossistes et les revendeurs des canaux de distribution, les services d'achat des grandes entreprises, les organismes de financement, ainsi que pour les distributeurs [9].

Les domaines d'application de RosettaNet sont caractérisés par une utilisation intensive des standards EDI. L'enjeu de consortium, c'est de proposer une solution de substitution pragmatique à ces standards en s'appuyant sur l'infrastructure généraliste du web et sur XML. La première réussite de RosettaNet est d'avoir su réunir les vendeurs, les distributeurs, les constructeurs, les grossistes et les grands utilisateurs autour de cet objectif.

4.3.1 RosettaNet et le dialogue entre partenaires commerciaux :

D'un point de vue conceptuel, RosettaNet a pour objet de formaliser le dialogue entre partenaires lors de transaction commerciales, comme le montre la figure 4.3. Ce formalisme vise donc à simplifier et automatiser, du côté de l'émetteur, la création d'information, son incorporation dans des messages, puis l'émission de ses messages, du côté du destinataire.La réception de ses messages et l'extraction, puis le traitements de l'information commerciale ainsi véhiculée.

Pour que ce dialogue a lieu, l'architecture technique de RosettaNet définit trois notions :

• Des dictionnaires et des codes définissent le vocabulaire commercial employé dans les documents RosettaNet transitant sur le Web.

• Un modèle de processus métier appelé *Partner Interface Process* (PIP), et un catalogue de processus préfabriqués sont mis à la disposition des utilisateurs de RosettaNet.

• Un cadre d'implémentation, le *RosettaNet Implementation Framework* (RNIF), définissent les choix techniques du consortium pour le format et l'échange des messages RosettaNet.

La création et le traitement de l'information commerciale mettent en jeu les dictionnaires et les codes RosettaNet qui assurent une compréhension partagée des documents échangés. Tandis que les échanges de messages RosettaNet sont pilotés par le PIP régissant la transaction entre partenaires RosettaNet.

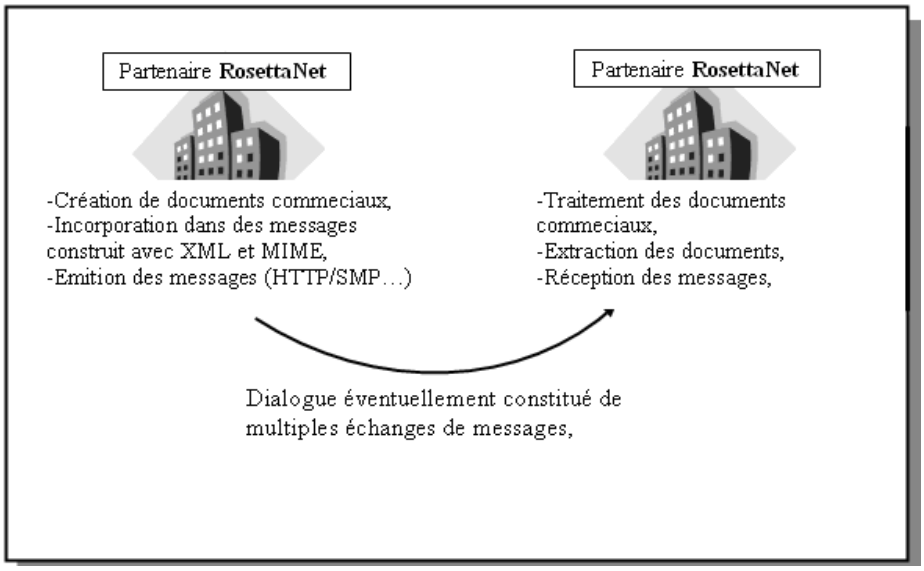


Figure 4. 3: Le dialogue RosettaNet [85].

4.3.2 Dictionnaires RosettaNet :

RosettaNet a recours, pour la codification des sociétés et des produits, à des codes existants, également employés dans l'univers EDI. Les codes RosettaNet : RosettaNet met en jeu trois codes industriels ou sectoriels dans son architecture technique :

- L'identification des partenaires d'une transaction RosettaNet emploi le code DUNS, pour identifier et localiser les acteurs des processus PIP.
- Le code GTIN (Global Trade Item Number) est utilisé dans le processus PIP pour identifier les produits.
- Le code UN/SPSC est utilisé pour construire une classification hiérarchique des produits utilisée par certain processus.

Sur la base des codes précédents, RosettaNet offre deux dictionnaires, le RNTD (RosettaNet Technical Dictionary) technique, et le RNBD (RosettaNet Business Dictionary) métier, regroupant les termes et les informations fondamentales manipulés par les processus PIP.

Ainsi, le RNTD articule les descriptions de produits autour de propriétés techniques et de relations en vue de faciliter la création de catalogues, de documents de spécifications, et l'alimentation de bases d données descriptives de produits. Doté d'une API, on peut y accéder directement par programme.

Le RNBD, quand à lui, contient des descriptions des objets métier communs aux différents processus PIP (bon de commande, facture, etc…). Le RNDB contient encore des instances particulières, utiles à certains processus PIP ou standards dans un secteur donné.

L'usage combiné des eux dictionnaires RosettaNet vise à éliminer toute ambiguïté sur les données transmise dans les documents échangés par messages RosettaNet entre deux partenaires ayant adopté se standard.

4.4 EbXML

EbXML (Electronic Business eXtensible Markup Language), qui est une initiative profitant des capacités du XML pour développer une plate-forme e-commerce, a été lancé dans le but de définir un cadre de travail global pour le commerce électronique. Ses promoteurs en sont OASIS (Organization for the Advancement of structured Information Standards) et le CEFACT (Centre for Trade Facilitation and Electronic Business), sous l'égide de l'ONU [85].

L'idée générale d' ebXML est l'utilisation des possibilités de XML et des outils techniques d'Internet pour proposer une solution accessible aussi bien aux PME (Petites et Moyennes Entreprises) qu'aux grandes entreprises [88]

L'ambition très vaste du projet aurait pu faire craindre qu'il en soit d'ebXML comme il en fut des nombreux autres consortiums (d'intéressants efforts théoriques sans suites pratiques). Cependant, les participants aux groupes de travail, provenant du monde de l'EDI, amènent plus de vingt années d'expérience industrielle et garantissent le sérieux d'ebXML. Il convient également de noter que ebXML ne s'occupe pas de documents échangés dans les transactions commerciales mais bien de la formalisation du processus lui-même de transaction, le processus métier. L'architecture ebXML décrit les formats de messages et les composants génériques utilisés dans ces processus.

EbXML propose également une structure d'annuaire répartis. Chacune de ces normes fut confiée à des équipes de rédaction différentes, réunies une première fois en Août 2000, à mi-parcours dans le calendrier de 18 mois initialement fixé pour la livraison de la spécification ebXML. Les promoteurs d'ebXML, OASIS, d'une part, apportant son savoir-faire et son expérience sur XML, et l'organisation SEFACT qui dépend des Nations Unies, d'autre part, amenant toute l'expertise EDIFACT, on pu constater la réussite de leur pari de fin 1999, visant à réunir un consensus autour des composants essentiels au commerce électronique :

- Des services d'annuaires (Registry Services) et un modèle d'information associé (Registry Information Model), similaire à l'UDDI.
- Des services de messageries (Messages Services).
- Des protocoles et des contrats de collaboration (Collaboration protocol Profil, ou CCP, et Collaboration Protocol Agreement, ou CPA), définissent respectivement les services et les accords entre Web services sur la base de leurs capacités.
- Un schéma de spécification de processus métier, permettant de convertir en document XML (modèles de processus construits) en UML selon une méthodologique proposé par UN/CEFACT, UMM (UN/CEFACT Modeling Methodology)[8].

4.4.1 Messagerie

Le Service de Messagerie ebXML fournit un mécanisme sécurisé, cohérent et fiable pour l'échange de messages ebXML entre des utilisateurs de l'infrastructure ebXML et un ensemble de protocoles de transport incluant par exemple SMTP, HTTP/S, FTP, ….

Le service de messagerie ebXML peut être décomposé en trois parties logiques :

- Une interface de Service.
- Des fonctions fournies par le gestionnaire de service de messagerie.

- Des correspondances avec le système de transport sous-jacent.

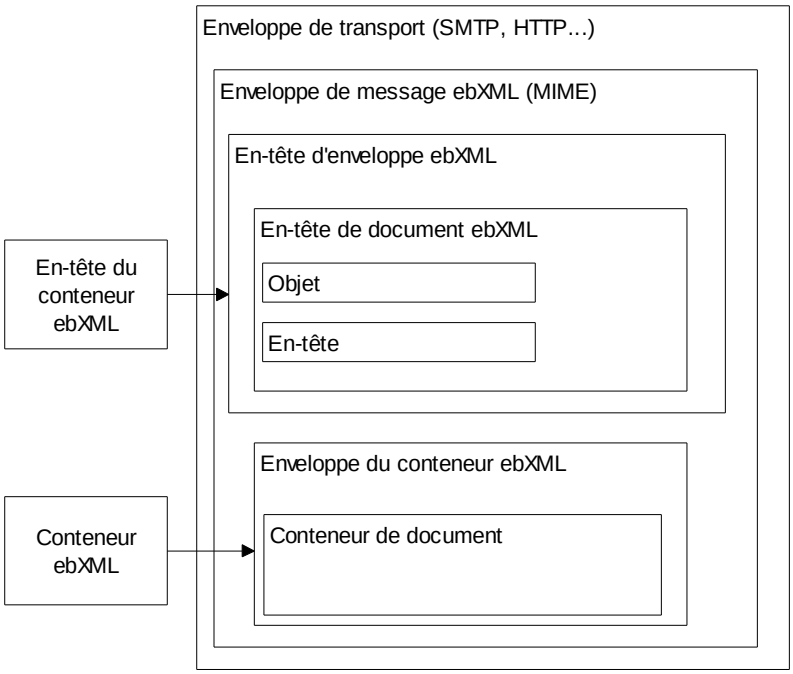


Figure 4. 4 : Architecture du service de messagerie [90].

Un message ebXML est constitué d'un protocole de transport spécifique et optionnel indépendant de l'Enveloppe du Protocole de Communication et d'un Protocole ebXML indépendant d'Enveloppe de Message [90].

4.4.2 Registres ebXML

Un Registre ebXML fournit un ensemble de services destinés à partager l'information entre des partenaires commerciaux. Le registre fournit un archivage stable où l'information soumise par une organisation soumissionnaire est conservée de manière persistante. Cette information est destinée à être utilisée pour faciliter les transactions et formations de partenariat de type "B2B" basées sur ebXML. Les contenus soumis peuvent être des documents et schémas XML, des descriptions de processus, des composants de communs ebXML, des descriptions de contexte, des modèles UML, des informations sur les partenaires et même des composants logiciels.

4.4. 3 Protocoles de collaboration

Pour faciliter l'échange d'informations, ebXML recommande l'utilisation d'un profil de protocole de collaboration CPP (Collaboration Protocol Profile), et d'un accord de protocole de Collaboration CPA (Collaboration Protocol Agreement). Le CPP décrit les capacités qui sont supportées par un Partenaire Commercial et les spécifications des Interfaces de Service qui ont besoin d'être « accordées »

Avant de pouvoir procéder à l'échange de documents d'affaires avec ce Partenaire Commercial. Un CPA décrit : (1) le Service de Messagerie, (2) les spécifications des Processus d'Affaires qui sont agréés par deux ou plusieurs Partenaires Commerciaux Conceptuellement. [90] Ce document XML définit un accord entre deux partenaires.

EbXML offre un scénario de collaboration, qui aboutit à l'exécution d'une transaction d'affaires dans un interchange, en suivant ces étapes :

1. Une Compagnie A est informée qu'un registre ebXML est accessible sur Internet.
2. Après consultation du registre ebXML, la compagnie A construit son système ebXML.
3. La compagnie A soumet les informations concernant son profil d'affaires (CPP) au registre ebXML.
4. La compagnie B découvre les scénarios d'affaires supportés par compagnie A en consultant le Registre ebXML, et procède au téléchargement des profils (de la compagnie A).
5. Avant de s'engager dans l'exécution du scénario, la compagnie B soumet directement à l'Interface ebXML de la compagnie A une proposition d'accord d'interchange. La compagnie A accepte l'accord d'interchange (CPA). Les deux compagnies sont prêtes à engager des échanges électroniques.
6. Engagement des transactions commerciales.

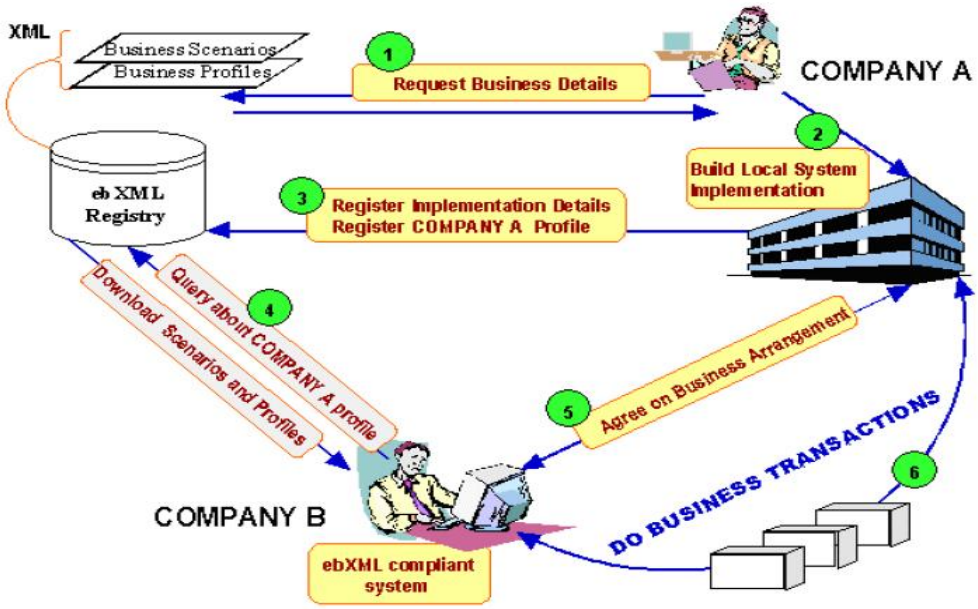


Figure 4.5 : Scénario de collaboration EbXML.

EbXML se veut un cadre de référence pour le B2B. Cependant, et compte tenu de l'explosion du nombre de partenaires sur Internet, de nouveaux besoins, tels que l'accélération de la recherche des partenaires potentiels et l'optimisation de la négociation des offres, sont exigés. Nous allons augmenter les performances du scénario de collaboration d'ebXML, par l'introduction des agents mobile.

Les agents sont introduits dans ce scénario pour deux raisons :

- L'optimiation du scénario en terme de temps : aussi bien en phase de recherche qu'en phase de négociation, l'utilisation d'un agent s'avère très performant :

    - Par rapport à un être humain : vu que la tâche est réalisée automatiquement, c'est l'avantage des outils informatiques sur l'homme. Nous citons cet avantage, car les tâches dédiées actuellement aux agents, sont généralement celles réalisées par des êtres humains.
    - Par rapport à des programmes classiques : ce sont des programmes passifs par opposition aux agents qui prennent de l'avance via leurs caractéristiques distinctives, comme l'autonomie par laquelle un agent peut s'adapter à des situations différentes. De plus, la mobilité permet à un agent de traiter l'information sur le serveur et d'envoyer uniquement les informations jugées utiles. Ceci permet de minimiser les communications et par conséquent diminuer considérablement le temps d'éxécution de la tâche.

- L'optimisation en terme d'éffecacité : surtout en phase de négociation. Les mécanismes de collaboration et de communication entre les agents augmentent l'effecacité. Par exemple, le cas de négociation des offres des partenaires potentiels. Genéralement, ce sont des représentants humains qui s'occupent de cette tâche. Même en écartant le problème de subjectivité, la communication peut engendrer des surcoûts en terme de temps et de frais. Le manque d'une vision globale et d'une stratégie de négociation peut engendrer des pertes qu'une organisation peut éviter.

5. Conclusion

Dans ce chapitre, nous avons présente le domaine du commerce électronique et les types des marchés les plus importants qui sont : le B2B, le B2C, le C2B, etc. Nous avons vu aussi que le commerce electronique c'est avant tout du commerce, c-à-d il s'appuie sur l'échange de documents d'affaires : factures, états de compte, bons de commande, accusés de réception, bons de livraison, d'expédition, etc. c'est pour cette raison qu'il faut mettre en œuvre des standards d'échanges : EDI, BizTalk, RosettaNet, et ebXML. De ces standards, nous nous intéressons particulièrement è l'ebXML, car c'est le plus approprié pour le commerce electronique et le B2B en particulier.

Le prochain chapitre sera l'objet de notre architecture d'intégration proposée, à deux niveaux : applicatif et collaboratif. Nous étudions comment les ontologies peuvent être utilisés pour assurer l'intégration des applications hétérogènes.

CHAPITRE 5

Intégration A2A

1. Introduction

La croissance du e-commerce, en particulier le B2B (Business to Business) soulève beaucoup de questions de la manière que les entreprises fonctionnent avec leurs fournisseurs, leurs clients et les autres partenaires commerciaux. Ceci nécessite fréquemment le developpement de nouvelles applications en couplant celles existantes. Ceci implique l'apparition d'un nouveau paradigme, appelé l'EAI (Enterprise Application Integration). L'EAI fournit des outils d'interconnexion des systèmes hétérogènes de l'entreprise tels que les CRM (Customer Relationship Management), les SCM (Supply Chain Management), les ERP (Enterprise Resource Planning) et les systèmes classiques ou legataires. Il est difficile de réaliser cette interconnexion parce que les systèmes intégrés sont développés indépendamment [57].

L'EAI est difficile parce qu'il doit considérer l'hétérogèneité des données, des processus et des services. L'hétérogénéité peut apparaître dans l'utilisation de différents langages de programmation, le fonctionnement sur des plateformes et des systèmes d'exploitation variés. En outre, ces applications ont été développées indépendamment et ne partagent pas ainsi la même sémantique pour la terminologie de leurs modèles d'applications. De cela, l'hétérogénéité sémantique se produit parce qu'il y a un désaccord du sens.

La collaboration des partenaires hétérogènes mène à l'issue d'interopérabilité [62], qui représente un obstacle majeur dans le secteur du business. L'hétérogeneité résultent du fait que les partenaires ne partagent pas le même modèle du processus métier. Ceci signifie qu'une compagnie qui modèle un ordre prédéfini d'un groupe d'activités ne peut pas s'intégrer directement avec des compagnies ayant différents ordres, Même, si cette compagnie est disposée à se conformer à ces sélections. Il signifie également qu'une entreprise echange des informations, disposées dans un ensemble de messages et d'activités, ne peut pas s'aligner avec une autre entreprise qui échange la même information dans différents messages et/ou activités [58]. Une solution au problème est de modeliser les processus de l'entreprise d'une façon flexible pour assurer une plus grande portée des collaborations B2B aux lesquels les compagnies se conforment, sans la nécessité d'une nouvelle conception du modèle [54].

Les approches proposées pour explorer le domaine de l'EAI se sont concentrées sur une intégration technique et syntactique. Néanmoins, ils doivent adresser le niveau sémantique qui est le plus difficile intra (A2A) et interentreprise (B2B) [31]. D'une part, l'intégration offre un cadre pour échanger les données et unifier les composants logiciels. À cette fin, les ontologies sont une solution candidate facilitant l'échange d'information sémantiquement enrichie. Elles encapsulent l'hétérogénéité des applications intégrées et offrent une terminologie commune. D'autre part, l'intégration permet d'aligner les processus métier. La solution proposée dans notre cas est de modeliser les processus métier d'une façon flexible pour augmenter les capacités des processus de s'assortir sans exiger des changements de leur conception.

Dans ce chapitre, nous étudions comment les ontologies peuvent être employées pour assurer l'intégration des applications. En particulier, nous proposons une architecture d'intégration à deux niveaux : applicatif et collaboratif.

- Au niveau applicatif, nous nous concentrons sur une approche de mapping pour intégrer les ontologies d'applications. L'architecture supporte plusieurs applications hétérogènes accédant aux ontologies locales d'application. Un composant appelé le mapper permet de générer l'ontologie globale pour gérer l'hétérogénéité des applications basée sur le concept de sémantique bridge [33]. Ce dernier encapsule les informations exigées pour translater les instances de l'ontologie source aux instances de l'ontologie cible.

- Au niveau collaboratif, nous fournissons un support de collaboration entre les partenaires basés sur le scénario ebXML étendu [64]. Par la suite dans le prochain chapitre, nous détaillons notre approache de developpement de l'ontologie de processus métier basée sur une stratégie de composition des web services. Nous utilisons des mécanismes pour contrôler les compositions potentielles et superviser l'exécution du processus. Enfin, le langage BPEL4WS est utilisé pour exprimer la spécification des flux de contrôle des processus métier et les contraintes.

2. Architecture d'intégration des applications

Dans le système, nous identifions plusieurs types d'applications légataires, client/server et web, développées en utilisant des langages de programmation différents. Ils fonctionnent sur de plateformes et de systèmes d'expoitation variés et utilisent divers format pour l'échange des données. Les ontologies d'application permettent d'assurer la communication entre les applications, au profit de l'intégration. Par conséquent, les ontologies servent de base pour interopérer les applications hétérogènes.

Le système d'intégration proposé vise à offrir un support d'intégration des applications hétérogènes, autonomes et réparties accédant à des ontologies multiples (figure 5. 1). Il permet un échange d'information approprié entre les ontologies d'applications et genère l'ontologie globale basé sur un processus de mapping pour traiter l'hétérogénéité. Le processus d'intégration est basé sur le concept de sémantique bridge pour indiquer l'équivalence sémantique des entités d'ontologies afin de les intégrer.

En outre, nous donnons un aperçu de deux niveaux d'intégration d'applications, l'applicatif et le collaboratif.

2. 1 Niveau applicatif

Le niveau applicatif se compose des applications hétérogènes et réparties. Chaque application a sa propre ontologie. Dans notre architecture, nous visons à surmonter le problème d'hétérogeneité entre les ontologies d'application, selon les relations sémantiques. Un composant spécial, appelé mapper, est invoqué pour accomplir ses tâches afin de construire l'ontologie globale. Le dernier peut être vu comme une ontologie d'entreprise et permet la résolution des conflits sémantiques au niveau des concepts et des propriétés.

2. 2 Niveau collaboratif

Le niveau collaboratif prend place dans le cadre des processus métier et la collaboration des partenaires. Chaque Companie est doté d'un agent mobile. C'est le responsable de demander et fournir les services et la négociation pour selectionner le meilleur partenaire basé sur des critères (par exemple, limites des prix, configurations de produit ou dates-limites de la livraison). Cet agent utilise le scénario de collaboration ebXML étendu pour accomplir les de recherche et de négociation. Le scénario bXML étendu est basé sur l'intégration du paradigme d'agent pour garantir une économie de temps de recherche, de négocier les paramètres commercials et d'offrir une grande performance particulièrement en présence de la caractéristique de mobilité, ce qui résout les problèmes liés aux réseaux tout en diminuant la consommation des ressources [76].

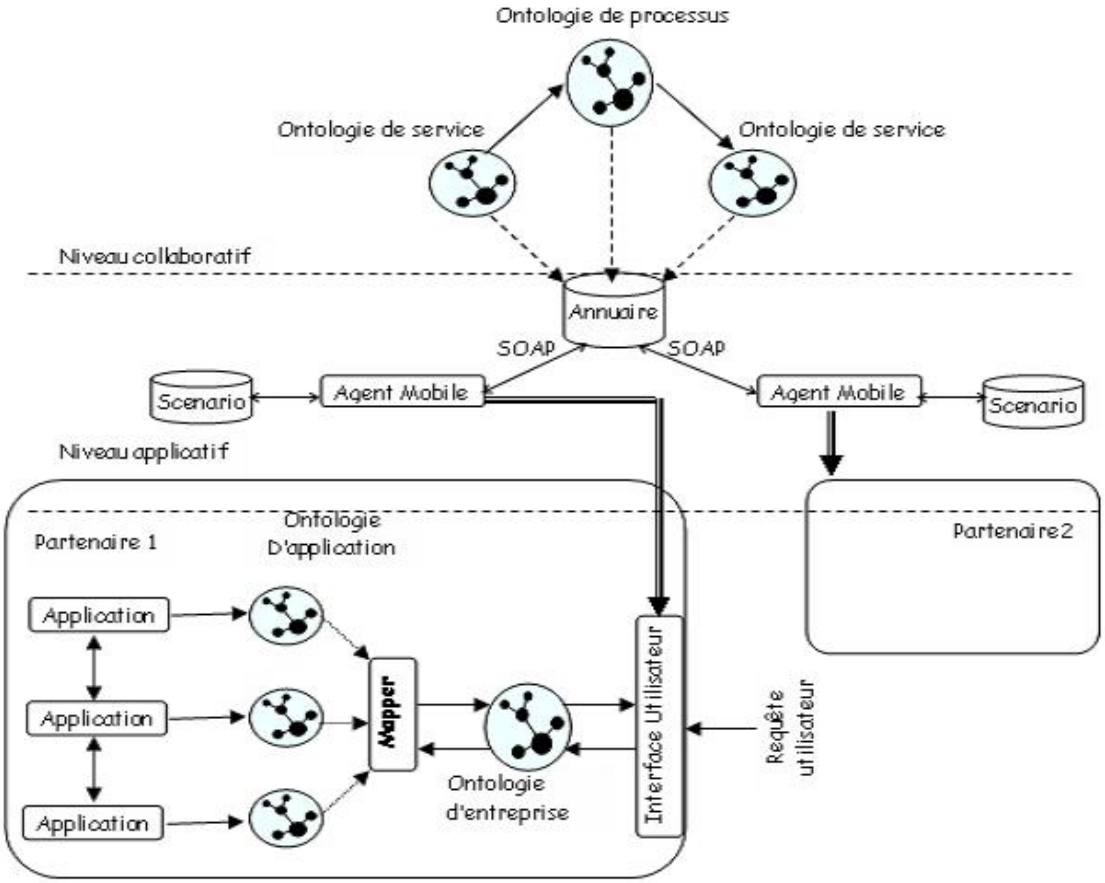


Figure 5.1 : Architecture du système d'intégration.

En plus, un aperçu du composant mapper est donnée par la suite. La tâche principale du mapper est de découvrir les relations sémantiques entre les concepts des ontologies d'application et de réutiliser ces information de mapping dans le scenario de communication. Ce composant contient quatre (4) parties :

- L’identifiant est utilisé pour découvrir les concepts ou les attributs concernés des ontologies et les relations entre eux. Ceci peut être fait automatiquement ou manuellement avec l'aide des experts du domaine. Par exemple, il peut introduire des ontologies lexiques tels que WordNet ou les thesaurus spécifique du domaine pour définir les relations de synonymie et d'antonymie [33].

- L'adaptateur est utilisé pour représenter les relations d'équivalence identifiées entre les ontologies basées sur le concept de sémantique bridge. Il combine plusieurs d'algorithmes pour mesurer la similarité. Il adopte une approche multi-stratégique pour calculer la similarité de concepts à des niveaux différents, tels que les similarités lexique et syntaxique des propriétés (des rôles et des attributs), des super/sous concepts dans la hiérarchie et des instances. Puis, l'adaptateur vérifie l'inclusion du domaine et du cardinalité.

- L'éditeur de liens transforme les instances de l'ontologie d'application source en instances de l'ontologie d'application cible en évaluant les relations d'équivalence définies précedamment par l'adaptateur. Les mapping absents peuvent être traités par le mécanisme d'inférence.

Le communicateur présenté dans la figure 5.2 est employé pour assurer la communication entre l’ontologie globale (ontologie d'entreprise) et les ontologies locales (les ontologies d'application). C'est le responsible d'échange de l'information pour réponde au requête utilisateur basé sur les informations de mapping appropriées. Il traduit la requête envoyée par l'utilisateur à une autre équivalente. Ceci est fait en appliquant les informations de mapping, selon les concepts de l'ontologie d'application concerné. Le communicateur comporte des sous-composants : le wrapper, le traitement, le demandeur et le récepteur.

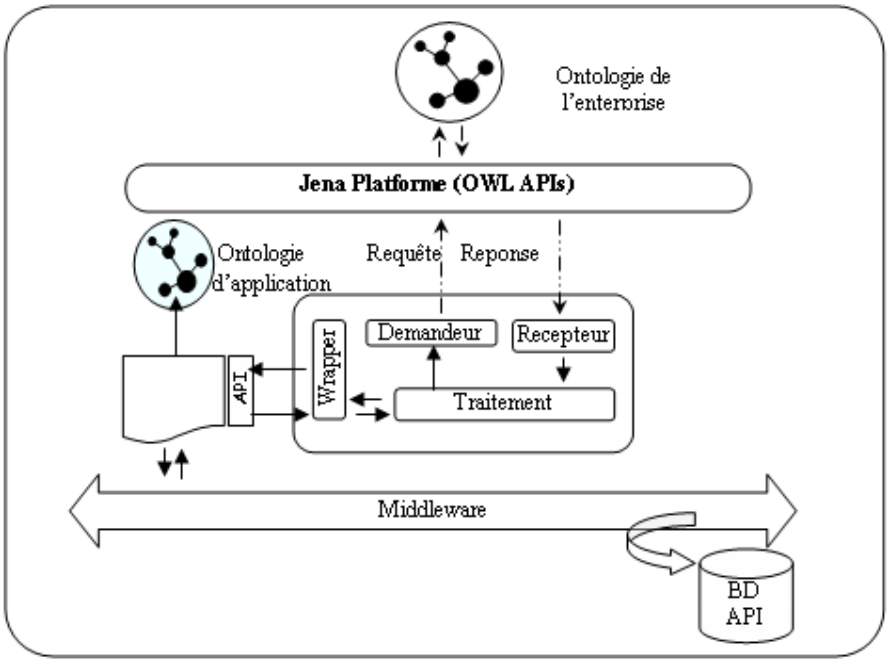


Figure 5.2 : Architecture du communicateur.

- Le wrapper : C'est une interface application/communicateur. Il est utilisé pour gérer l'hétérogénéité et invoquer l'application.
- Le traitement : Il analyse la requête, c.-à-d. l'initialiser puis envoyer le résultat au demandeur.
- Le demandeur : le rôle de ce composant est d'envoyer la requête pour invoquer l'ontologie d'entreprise.
- Le récepteur : Il obtient les résultats ou la réponse venant de l'ontologie d'entreprise.

- La base de données api (application programming interface) contient les interfaces des applications disponibles sur le middleware, qui est mis à jour quand nous ajoutons ou nous enlèvent une application. Cependant, chaque application a l'aptitude pour savoir toutes les applications reliées au middleware par son base de données api.

En outre, les applications connectées au middleware doivent communiquer via les ontologies. À cette fin, le communicateur récupère l'api de l'application intéressée. Puis, le composant traitement utilise l'api pour invoquer l'ontologie. Par exemple, quand une application source reçoit un message venant d'une application cible, le composant de traitement doit l'analyser basé sur les informations de mapping fournie par le mapper (identifiant-adaptateur-éditeur de liens) pour découvrir les activités nécessaires et répondre à l'application cible. Les sous-modules du communicateur ont été implémenté par Java en utilisant les APIs Jena [170]. De la même manière, les API Protégé 2000 sont employés pour invoquer l'ontologie. Ceci est appellé l'interface de requête : edu.stanford.smi.protege.model.query, et d'autre APIs fourni par l'éditeur Protégé2000.

3. Processus d'intégration des ontologies d'application

Un processus d'intégration des applications basé sur les ontologies est une nouvelle approche qui a été rapidement développée avec l'émergence du Web sémantique [65]. C'est principalement dû au fait que actuellement, les ontologies fournissent une base pour assurer l'interopérabilité et l'intégration des applications hétérogènes. Le processus d'intégration des applications proposée est caractérisé par l'itération. Il considère l'ajout d'une nouvelle application, l'évolution des besoins de l'entreprise et le changement des critères du marché. Le processus est divisé en quatre étapes de base : capture, developpement, intégration et exploitation.

Le processus suggéré est doté de deux sous-processus, essentiels pour le developpement et l'intégration des ontologies, représenté sur la figure 5. 3 par des cadres pointillés. Le premier montre comment construire les ontologies d'application. Il inclut plusieurs d'étapes du méta modélisation à l'implémentation. Le second montre la manière d'intégration des ontologies d'application. Ce sous processus commence à partir de la capture des besoins d'applications et aboutit à une ontologie globale.

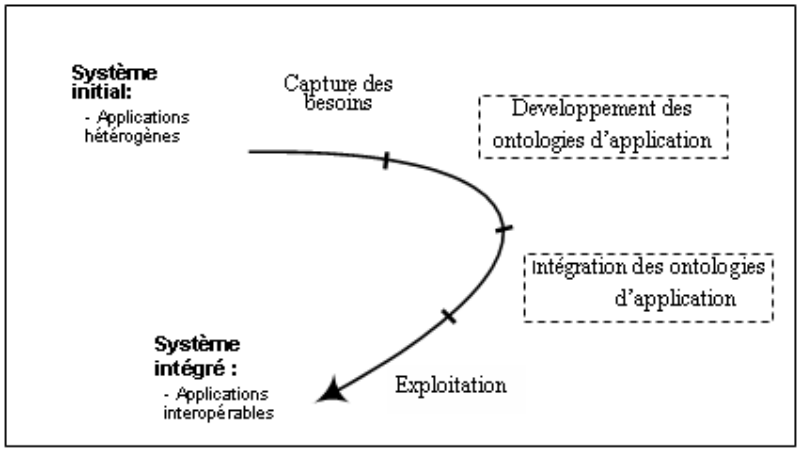


Figure 5. 3 : Processus d'integration des applications.

3.1 Capture des besoins

Un modèle de l'entreprise est une représentation informatique de la structure comprenant, les activités, les processus, les informations, les ressources, les acteurs, le comportement, les objectifs et les contraintes commerciales. Cette étape prend en compte les informations liées au champ de l'intégration tout en identifiant le comportement des applications aussi bien que les relations qu'ils maintiennent entre elles. Le but est de capturer l'ensemble des applications de l'entreprise, les activités qu'elles exécutent, les ressources exigées par ces activités, les données manipulées et les messages émis/reçus. Puis, nous identifions les flux d'information, de leur structure et l'infrastructure technique pour les supporter. Cette étape est effectuée par des spécialistes et des experts du domaine.

3. 2 Développement de l'ontologie d'application

Dans le cadre de notre travail, nous appliquons le diagramme de classe d'UML [70] pour représenter le méta-modèle d'ontologie d'application. Dans ce but, le processus de developpement d'ontologie contient toutes les tâches nécessaires pour construire l'ontologie d'application en tenant compte des informations identifiée dans l'étape précédente. Il sert à classifier les applications selon des caractéristiques appropriées. Puis, la formalisation implique la représentation de l'ontologie dans un langage formel. L'implémentation permet d'établir le modèle opérationnel de l'ontologie et verifie sa consistence. Notre processus de developpement contient quatre étapes : méta-modelisation de l'application, la formalisation, l'implémentation et l'adaptation.

3. 2. 1 Méta-Modélisation des applications

L'un des concepts de base qui nous permet d'appréhender correctement la notion d'EAI est celui d'architecture d'une application. On peut recenser, dans la littérature, plusieurs modèles qui se différencient par le degré d'abstraction (logique-physique) et le degré de raffinement. Dans le cadre de nos travaux, nous avons retenu la vue en couches (logique) qui nous paraît la plus pertinente. Ce type d'architecture permet de structurer toute application selon trois couches qui sont respectivement la couche donnée (qui traite l'accès aux données manipulées), la couche traitement (qui correspond aux règles métier implémentées), et la

couche présentation (qui gère l'interaction homme-machine). Ces différentes couches sont, bien entendu, interdépendantes dans la mesure où elles entretiennent des échanges entre-elles via des middlewares intra-applications.

Dans notre travail, nous maintenons la structure logique en couches. Les concepts de l'ontologie d'application sont représentés par un diagramme de classe UML (cf. figure 5. 4). Les ontologies incluent des hiérarchies de classe/sous classe, des relations entre les classes, les attributs de classe et les axiomes qui indiquent les contraintes. Ces informations sont habituellement modelées par un diagramme de classe (la raison pour laquelle UML est une notation prometteuse pour des ontologies). Notre ontologie permet d'offrir une couche intelligente pour gérer l'hétérogénéité sémantique au niveau des applications intégrées.

Le méta-modèle illustre le principe de l'ontologie déployé. Il indique les activités invoquées par les messages (reçu/emis) et les ressources nécessaires. Les données manipulées sont représentées par des schémas. Chaque application a un comportement représenté par des liens. Les propriétés d'application (niveau, comportement, données et activités) se composent d'un certain nombre de concepts. Chaque concept est décrit par des propriétés, des axiomes, des types et des relations reliant les concepts entre eux. Ces relations peuvent être des relations d'aggregation, de composition, de synonymie, etc. Les axiomes permettent de capturer des règles plus complexes sur les concepts. Ces relations et ces axiomes nous permettront de résoudre les conflits liés à l'hétérogénéité sémantique concernant les données et les activités des applications.

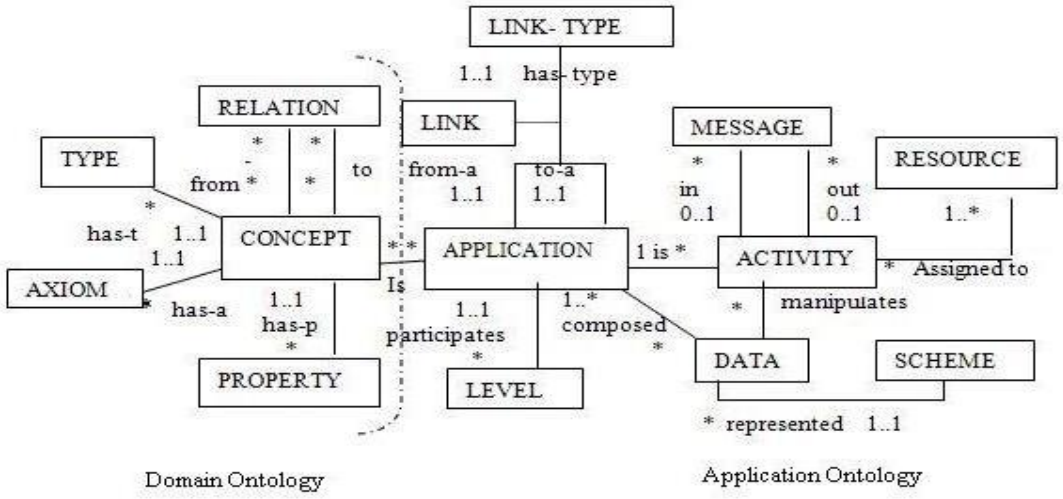


Figure 5.4 : Meta-modèle de l'ontologie d'application.

L'ontologie de domaine donne une représentation formelle des concepts du domaine étudiée, aussi bien que leurs relations. Cependant, l'ontologie de domaine ne peut pas décrire les fonctionnalités et les différences entre les applications dans un domaine donné. Il essaye de trouver l'ensemble de concepts couvrant le domaine. Dans notre travail, nous considérons les entreprises qui fournissent des produits et des services pour ses partenaires et ses clients potentiel. Nous identifions deux domaines, le ventes et l'achat.

Nous donnons une description formelle des concepts de l'ontologie d'application (Ont) :

Ont Application = {Application, Activité, Donnée, Schema, Ressource, Message, Niveau, Lien-Type, Concept, Relation, Type, Axiome, Propriété}

Nous créeons le dictionnaire de concepts selon Methontology [47]. Dans ce dictionnaire, nous définissons pour chaque concept : synonymes, instances, attributs, son poids dans le système et les rôles. La table 5. 1 représente un dictionnaire de concepts pour l'ontologie d'application.

Les relations binaires sont représentées sous forme de propriétés qui attachent un concept à l'autre. Pour chaque relation dont la source est dans la hierarchie de concepts. Nous définissons son nom, le nom du concept source, le nom du concept cible, la cardinalité source et cible et le rôle inverse (voir la table 5. 2).

Nom de Concept	Synonyme	Instances	Attributs	Comment Approprié?	Roles
APPLICATION	Programme Software	Achat-produit Vente-produit	Nom-Application Profill-Application Domain-Application comportement-Application	Obligatoire Important Important Important	De-application A-application a-type est compose participe ..
ACTIVITE	Fonction Service	Demande client Verifier-stock Invoquer-achat Solder-facture Livrer-article	Nom-Activite Type-Activite Precondition parametre Sous-activities Temps-début Temps-fin Ressource-assignée postcondition	obligatoire Important Important Important Important Bien-Avoir Bien-Avoir Important	entrant sortant est manipule assigne-a
DONNEE	Parametre Input/output	Client Facture produit	Data-name Data-type Access-right	obligatoire Important obligatoire	compose represente manipule
SCHEMA	Description	XML-shema	Nom-Schema	obligatoire	represente
RESSOURCE	Besoin	Memoire CPU	Nom-Ressource Type-Ressource Etat-Ressource	obligatoire Important Important	Assigne-a
MESSAGE	Requête	MySQL Requête OWL API	Id-Message Contenu-Message mode-communication	obligatoire Important Important	entrant sortant
NIVEAU	-	Applicatif Collaboratif	Nom-Niveau	obligatoire	participe
LIEN-TYPE	Connection	Exchange-donnée Partage-Service collaboration	Nom-lien	obligatoire	a-type
..	..	..	..	..	..

Table5. 1: Dictionaire de concepts de l'ontologie d'application.

Par example, le rôle manipule lie le concept source activité et le concept cible donnée est spécifié par: manipule: (Activité, Donnée).

Nom de rôle	Concept Source	cardinalité Source	Concept Cible	Cardinalité Cible	Rôle inverse
compose	APPLICATION	1..N	DONNEE	0..N	Appartient-a
represente	DONNEE	0..N	SCHEMA	1..1	decrire
est	APPLICATION	1..N	ACTIVITE	0..N	referer-a
manipule	ACTIVITE	0..N	DONNEE	0..N	manipule-par
assigne-a	ACTIVITE	0..N	RESSOURCE	1..N	utilise-par
de-application	APPLICATION	1..1	APPLICATION	1..1	a-application
a-application	CONCEPT	0..N	RELATION	0..N	a
a-type	APPLICATION	0..N	LIEN-TYPE	1..1	a-comportement
participe	APPLICATION	1..1	NIVEAU	0..N	indique
entrant	ACTIVITE	0..1	MESSAGE	0..N	sortant
a-p	CONCEPT	1..1	PROPRIETE	0..N	concerne
..	..	..	..	..	..

Table5. 2: Description de Rôles de l'ontologie d'application.

Remarquons qu'à partir du diagramme de classe, l'instantiation varie selon les applications. En particulier, chaque application peut avoir ses propres activities et ses données. Par exemple, une application de vente contient des activités : recevoir les demandes client, vérifier la disponibilité, invoquer l'achat, solder la facture, livraison des produit et manipule des données : client, produit, fournisseur, facture.

3. 2. 2 Formalisation

Un cadre standard pour représenter les connaissances d'une ontologie est la théorie de la DL (Logique de Description). La DL forme une famille de languges de représentation des connaissances. Elle permet de représenter la connaissance concernant un domaine spécifique en utilisant les "descriptions" qui peuvent être des concepts, des relations et des instances. Nous considérons la famille SH [38] [42], qui inclut SHOIN (D) DL, une extention du SHOQ (D) DL. OWL (Ontology Web Language) [67] est équivalent à SHOIN (D), c.-à-d. SHOIN (D) est une variante syntaxique du OWL DL.

La formalisation se compose de deux parties : langage terminologique TBOX dans lequel les concepts et les relations sont définis, et un langage assertionnel ABOX dans lequel nous présentons les instances.

- *Construction de TBOX :* Nous définissons les concepts de domaine et les rôles au moyen de constructeurs fournis par DL. Par exemple, une activité doit avoir un et seulement un nom. il appartient à une application. Elle manipule les données auxquelles une ressource est assignée. Elle réagit aux messages entrants et produit un message sortant pour envoyer la réponse.

  Activite := (∃ Nom-Activite.String) ∩ (= 1 Nom-Activite.String) ∩ (= 1 Manipule.Donnee) ∩ (= 1 Assigne-a. Ressource) ∩ (= 1 Referer-a.Application) ∩ (= 1 Entrant.Message) ∩ (= 1 Sortant.Message).

De plus, nous indiquons les relations de subsumption existant entre les divers concepts. Par exemple pour indiquer que la classe activité est subsumée par la classe application, nous écrivons : Activité $\subseteq$ Application (cf. tableau 5. 3).

Concept	Definition	Subsomption relation
APPLICATION	(∃ Nom-Application.String) ∩ (=1 Nom-Application-.String) ∩ (≥1 est. Activite) ∩ (≥1 Compose.Donnee) ∩ (=1 Participe.Niveau) ∩ (≥1 a-type.Lien-type)∩ (=1 De-application.Application)∩(=1a-application .Application)∩ (≥1 est. Concept) .	APPLICATION ⊆ THING
ACTIVITE	(∃Nom-Activite.String) ∩ (∃Type-Activite.Thing) ∩ (=1 Nom-Activite-.String) ∩ (=1 Type-Activite.Thing) ∩ (≥1 Manipule.Donnee) ∩ (≥1 Assigne-a. Ressource) ∩ (≥1 Referer-a.Application) ∩ (=1Entrant.Message)∩ (=1 Sortant.Message).	ACTIVITE ⊆ APPLICATION
DONNEE	(∃ Nom-Data-.String) ∩ (∃Type-Donnee-.Thing) ∩ (=1 Nom-Data-.String) ∩ (=1 Type-Donnee.Thring) ∩ (≥1 Appartient-a.Application) ∩ (≥1 Manipule-par.Activite) ∩ (=1 Represente.Schema)	DONNEE ⊆ APPLICATION
SCHEMA	(∃ Nom-Schema.String) ∩ (=1 Nom-Schema-.String) ∩ (=1 Decrire.Donnee)	SCHEMA ⊆ DONNEE
RESSOURCE	(∃ Nom-Ressource.String) ∩ (∃Type-Ressource.Thing) ∩ (=1 Nom-Ressource.String) ∩ (=1 Type-Ressource.Thing) ∩ (≥1 Utilise-par.Activite)	RESSOURCE⊆ ACTIVITE
MESSAGE	(∃ Id-Message.String) ∩ (=1 Id-Message.String) ∩ (=1 Entrant. Activite) ∩ (=1 Sortant.Activite)	MESSAGE ⊆ ACTIVITE
NIVEAU	(∀ Nom-Niveau.{ Niveau-Collaboratif, NiveauApplicatif })	NIVEAU ⊆ THING
LIEN-TYPE	(∃ Nom-Lien.String) ∩ (=1 Nom-Lien.String) ∩ (=1 a-comportement. Application)	LIEN-TYPE ⊆ THING
..	..	..

Table 5. 3: Definition de concepts d'ontologie d'application dans TBOX.

- *Construction ABOX:* Un language d'assertion permet de décrire des facts, par la spécification des instances (avec leurs classes) et les relations entre elles de la manière suivante:
  a: C indique que a est une instance de la classe C.
  Example: achat-produit: APPLICATION.
  (a1, a2): r indique que les deux instances a1 and a2 sont connectées par le rôle r.
  Example: (solder-facture, facture): manipule.

3. 2. 3 Implémentation et test

L'implémentation permet d'établir un modèle opérational en utilisant le languge appropriée. L'effort est concentré sur OWL DL [38], qui est équivalent à SHOIN(D) [71]. Pour vérifier certaines propriétés, nous utilisons les services d'inférence fournis par plusieurs systèmes tels que FACT [38], RACER [39] et le DLP [38]. Nous employons le système

RACER pour lire les fichiers OWL et les convertir sous forme de bases de connaissance DL afin de fournir des services d'inférence.

Nous choisissons Protégé-OWL [50] pour implémenter les ontologies d'application. Protégé-OWL offre des facilités pour imposer des contraintes sur les concepts et les relations. Nous basons également sur Protégé-visualisation pour passer en revue l'ontologie et assurer sa consistence.

3. 2. 4 Adaptation et évolution

Les ontologies changent à travers le temps et doivent être adapté pour les besoins spécifiques de l'application. Par conséquent, des mécanismes sont demandés pour découvrir les changements (par exemple OnToKnowledge [36] ou Prompt [35]). Les versions d'ontologie peuvent ne pas être compatibles l'une avec l'autre. Ainsi, les informations doivent être disponible pour indiquer la compatibilité (backward, upward, fullcompatible, incompatible). OWL fournit quelques constructions intégrées aux compatibilités pour marquer les versions.

Pour contrôler l'évolution de l'ontologie, une classification a lieu une fois qu'un nouveau concept est introduit. À cette fin, nous proposons l'utilisation de l'algorithme de classification de DL [40].

3.3 Intégration des ontologies d'application

Afin de réaliser une intégration sémantique basée-ontologie, nous proposons un sous-processus qui inclut les étapes principales suivantes: la comparison, le bridging et l'inference.

- La comparison sert à comparer chaque concept A de l'ontologie source à chaque concept B de l'ontologie cible et de déterminer les métriques de similarité. Nous adoptons un processus multi-stratégique [33] qui calculent la similarité entre les entités des ontologies combinant différents algorithmes. Le premier se concentre sur acquésition d'une similarité lexique. Le deuxième calcule la similarité des propriétés qui est responsable d'acquérir la similarité entre les concepts basés sur leurs propriétés (les attributs et les rôles). Puis, nous propageons la similaité à d'autres parties de la taxonomie, les super et les sous concepts. Enfin, nous vérifions l'inclusion de domaine et de cardinalité.

- Le bridging est responsable d'établir la correspondance entre les entités basées sur les similarités calculées précédemment. Chaque instance représentée selon l'ontologie source est translaté en instance la plus semblable décrit selon l'ontologie cible. Il associe une procedure de transformation pour translater les entités similaires.

- Le mécanisme d'inference est utilisé quand le concept source n'a pas son concept cible. Il signifie que le concept source n'a pas son équivalent direct dans l'ontologie cible. Par exemple, le concept chambre 4-étoile peut différer entre plusieurs systèmes d'information de tourisme. Selon les équipements donnés aux chambres, un système d'information de tourisme peut classer une chambre 4-étoiles d'autres seulement en chambre 3-étoiles en raison du manque de certains services [36].

3.4 Exploitation

Cette étape traite la réalisation du modèle informatique d''intégration, qui permet de relier les applications d'entreprise. Une fois que le système est déployé, ce dernier doit maintenu l'infrastructure technique de l'intégration. En outre, cette étape couvre l'exploitation du système d'intégration.. Il est nécessaire d'adresser les évolutions de l'entreprise (par exemple intégration d'une nouvelle application) et de contrôler les versions d'ontologie.

4. Scénario de communication

Dans un système d'intégration, la modification ou l'addition d'une application a parfois un impact négatif sur son fonctionnement. D'abord, nous devons remodeler complètement le système ou développer d'autres modules afin de garantir sa cohérence et sa flexibilité. Notre système doit répondre à quelques questions :

- Quels sont les composants qui ont un effet sur le système d'intégration ?
- Quelles sont les entités concernées par le changement ?
- Comment procéde-t-on pour rétablir la communication dans le système d'intégration ?

Cependant, les ontologies fournissent une base pour faciliter l'interopérabilité entre les applications hétérogènes et cela par le developpement d'une ontologie d'application pour chaque application. Le problème est comment l'application se comporte-elle pour qu'elle soit en mesure d'utiliser l'ontologie ?. Cette difficulté apparaît parce que l'application n'a pas été précédemment conçue pour utiliser l'ontologie. Pour cela, nous devons rétablir la communication entre les applications à intégrer afin d'en fournir une sémantique interopérable entre elles.

Nous montrons dans cette section comment les applications d'entreprise fonctionnent à la reception d'une requête client. Pour cela, nous donnons une vue des entités nécessaires à introduire pour que le changement apporté par le déploiement des ontologies au sein d'un système d'intégration soit robuste et flexible. Pour cela, nous présentons le rôle de chaque entité ajoutée au système et le trajet que va prendre la requête du client depuis sa déposition jusqu'à ce que ce dernièr reçoive la réponse.

Comme la montre la figure 5. 5, nous donnons une description du rôle de chaque entité introduite dans le système d'intégration. Ces entités doivent apporter une solution adéquate de telle sorte que les applications du système peuvent aisement manipuler les ontologies. Ces composants sont : module médiateur, communicateur et la plateforme Jena.

- **Module médiateur** : Le but de ce module est également de recevoir les requêtes provenant des clients, puis les décomposer et les distribuer vers les applications destinatrices appropriées y compris la plateforme Jena. Une fois que les réponses sont renvoyées, ce module doit les composer afin de satisfaire la requête client.

- **Communicateur** : Comme son nom l'indique, ce composant a pour but d'interpréter et analyser les messages venant du module de médiateur vers les applications destinatrices du message ou bien du côté application vers le module médiateur (voir la section 3). Ces deux derniers ne sont pas forcément sur le même middleware.

- **Plateforme Jena** : Ce module offre le un certain nombre d'APIs OWL servant à exploiter l'ontologie d'entreprise (version 2.0 Jena).

Maintenant nous décrivons les différentes étapes du scénario de communication présenté par la figure 5.5. Nous distinguont neuf (9) étapes essentielles :

Le scénario commence une fois que le module de médiateur reçoit la requête client et la décompose en sous-requêtes (0). Puis, la plateforme Jena fournit l'APIs pour découvrir les applications appropriées de l'entreprise. La recherche peut également contenir des restrictions sur le domaine auquel le client est intéressé (1). Après, le résultat contient l'ensemble d'applications l'entreprise choisie par le module médiateur selon les paramètres de la requête client (2). Ce choix doit être accompagné de la liste d'interfaces des applications concernées reliées au middleware (3). En conséquence, le module médiateur reçoit toutes les applications et activités selon les besoins du client (4). Une fois l'ensemble d'applications désirées ont été découverts, nous incluent également toute information concernant la structure d'applications. Par la suite, les applications choisies sont invoquées selon leur structure qui les relie au middleware (5). Ainsi, tous les résultats seront retournés au module de médiateur pour les reconstituer et composer avant de les passer au client (6). La réponse globale est retournée au client (7). En conclusion, nous adressons les mises à jour nécessaires dans l'ontologie s'il exige un changement (8).

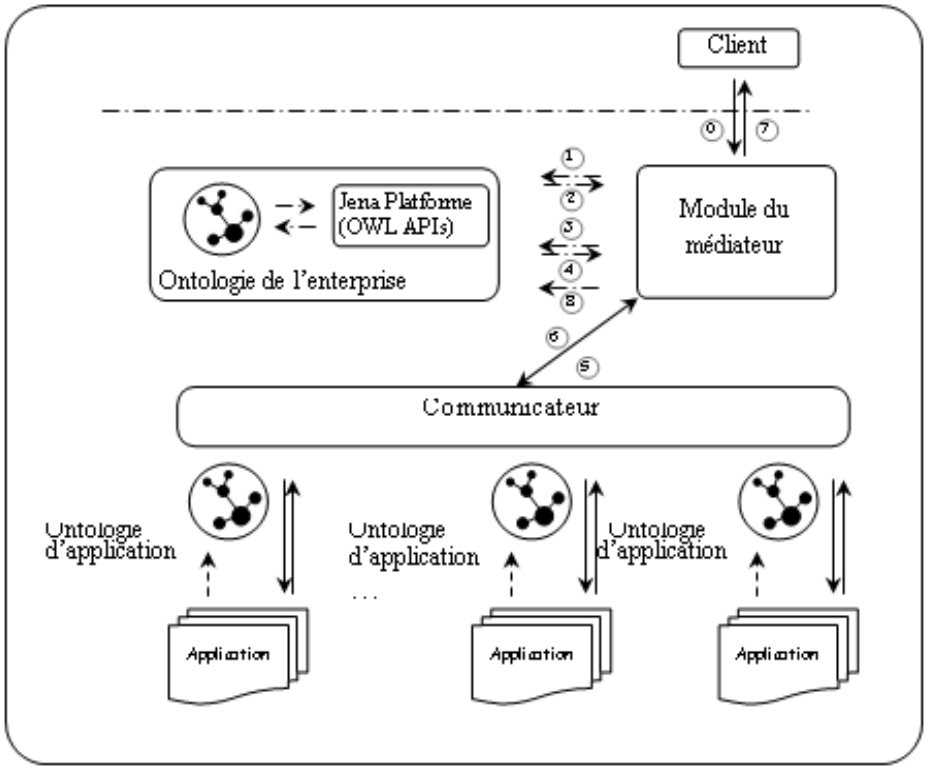


Figure 5.5 : Scénario de communication entre le client et les applications.

5. Conclusion

Dans ce chapitre, nous avons proposé une architecture d'intégration des applications de l'entreprise à deux niveaux, applicatif et collaboratif. Nous avons montré l'apport des ontologies pour gérer l'hétérogeneité et assurer un fonctionnement global cohérent du système d'intégration à travers le scénario de communication introduit. De plus, nous avons proposé un processus d'intégration des applications incluant deux (2) sous-processus. Le premier sert au developpement des ontologies d'application. Le second porte sur l'intégration des ontologies developpées.

Le chapitre suivant concernera la collaboration B2B par la proposition d'une ontologie de processus métiers afin de supporter la collaboration entre les partenaires. Nous allons présenter les points suivants :

- Construction d'ontologie des processus par l'intégration des ontologies de services basé sur un scénario d'intégration
- Utilisation du langage BPEL4WS pour spécifier l'enchaînement des services modélisant les processus métiers.
- Étendre le scénario de collaboration EbXML par des agents de recherches et des agents mobiles de négociation des transactions commerciales.

CHAPITRE 6

Collaboration B2B

1. Introduction

Dans ce chapitre, nous décrivons en détail le niveau de collaboration entre partenaires proposé dans le chapitre précédent. L'objectif de notre modèle de représentation de processus métier est d'étudier et évaluer les activitées constituant le processus métier selon une strategie de composition. Si l'entreprise se limite uniquement à l'intégration A2A, cela peut réduire sa flexibilité et son agilité. À cette fin, les entreprises doivent réagir pour faire face aux exigences sans cesse évolutives du marché. Les fortes 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, plus flexibles et plus réactifs, afin de pouvoir mieux soutenir la stratégie de l'entreprise. Pour faciliter la collaboration sur une échelle globale, les entreprises doivent aligner leurs processus métier. Ainsi, la collaboration B2B est étroitement liée à l'intégration A2A. Plus de détails au sujet de processus métier est donnés dans les sections suivantes.

2. Collaboration des partenaires

Les entreprises sont des organismes qui effectuent le travail collectif. Afin de réaliser ses activités, le travail a besoin d'être régi par les processus qui définissent le flux du travail dans toute l'organisation. Selon Coase [72], l'exitence de l'entreprise lui-même dépend de sa capacité de réaliser les transactions internes. Malheureusement, les entreprises ne peuvent pas entretenir chaque besoin intérieurement, parce qu'à un certain point ces opérations excèdent leurs capacités. Quand ceci se produit, les entreprises doivent établir des relations commerciale et des accords de collaboration entre elles.

2.1 Modèle flexible des processus métiers

Comparé aux processus traditionnels, la complexité des processus métier est due considérablement aux différences stratégiques, structurales et culturelles entre les partenaires. L'attribution des performances et des ressources des partenaires, la détermination des responsabilités et des liens d'échange financiers, l'échange des informations à travers les interfaces doivent être plannifiés et coordonnées. Ainsi, les exigences vis-à-vis de BPM (Business Process Management) augmentent [62]. Avoir un consensus entre les différents partenaires est considerablement assuré par une approche flexible. Elle encapsule le modèle commun des processus métier collaboratif [34].

Pour spécifier les processus, notre modèle prend en compte la description du profil du processus. Il indique ce que le processus fait et comment il invoque ses fonctionnalités dans la perspective de demandeur de processus, liée à un partenaire particulier. Son fonctionnement décrit le contexte commercial et les contraintes organisationnelles exigées. De plus, ce modèle indique quel scénario de collaboration exigé. Le modèle considère quatres (4) vues liées aux aspects collaboratif, fonctionnel, organisationnal et technique.

- La vue collaborative définit la choreographie du processus collaboratif et la structure des informations échangées. Une collaboration commerciales est généralement performée par deux partenaires (collaboration binaire) ou plus (collaboration multi-partie). Cela peut être complexe parce qu'il implique plusieurs services des différents partenaires. Cependant, la collaboration la plus fondamentale est binaire réalisée par d'un côté par un demandeur et d'autre côté par un fournisseur.

- La vue fonctionnelle décrit le profil de processus métier. Elle montre la structure du processus : l'ensemble de services, les paramètres d'E/S, l'ordre d'exécution et les partenaires impliqués.

- La vue organisationnelle sert à rassembler les informations sur les partenaires et les contraintes organisationnelles afin d'indiquer le contexte commercial. À cette fin, le processus métier doit être définie dans son contexte exact, c-à-d le même scénario d'éxecution du processus exactement dans son organisation.

- La vue technique supervise l'exécution du processus métier basée sur les objectifs des partenaires. Le flux de contrôle est important pour décrire l'enchaînement des activitées d'un processus métier. Les partenaires collaboratifs doivent comparer les résultats de l'exécution avec leurs objectifs afin d'ajuster les deviations en traitant les exceptions possibles.

Dans notre approche, les aspects fonctionnels sont utilisés pour construire l'ontologie de processus. Les aspects organisationnels sont exploités pour présenter le contexte commercial approprié. La vue technique permet d'améliorer l'exécution de processus selon le scénario de collaboration.

2.2 Développement de l'ontologie de service

Un service est principalement décrit par un ensemble d'opérations et des paramètres d'E/S fournit la description de l'interface de service. Pré et post-conditions sont énoncés sur chaque opération et sur le service en entier. Ce sont des expressions logiques sur les entités d'E/S. Elles doivent être vérifiées avant l'exécution d'une opération ou d'un service et doivent être satisfaites après l'exécution.

Par exemple, un service de vente peut exiger comme précondition une carte de crédit valide et comme entrée le numero de la carte crédit et la date l'expiration. Comme sortie, il produit un reçu et comme post-condition, la carte est chargé. D'autres caractéristiques de service considèrent l'ordre dans lequel les opérations doivent être performé. Puis, quels messages le service réagit et quels messages le service produit. De plus, les services maintiennent entre eux des relations sémantiques (synonymie, spécialisation, agrégation…) et conjoncturelles (localisation, composition, échange …).

Une ontologie de service est un 9-tuple définit comme suit:

Ont Service = (Service, ServiceCluster, ServiceProfil, ServiceRessource, ServiceMessage, ServicePartenaire, ServiceLocation, ServiceQualite, ServiceException)

Dans notre approche, l'ontologie de service vise à définir la sémantique des services composant le processus métier. Trois (3) types de service sont intégrés :

- Service atomique : C'est un service élémentaire directement invocable. C'est la description du service proposé par le fournisseur. IL est décrit en terme de propriétés d'interface (entrée, sortie, opérations, paramètre d'erreur et types de message), de comportement, de qualité et d'implémentation. Dans notre ontologie, nous étudions la description des services atomiques pour établir les relations sémantiques entre eux pour les grouper en des ensembles (clusters) de services similaires.

- Service composé : Il représente un cluster des services atomiques similaires. C'est généralement approprié aux grandes et complexes entreprises avec différents domaines commercials (production, vente…). Trois types de relation entre les services composés sont considérés. La relation ***est-similaire*** tient compte quand un service composé a les mêmes opérations qu'un autre service. La relation ***est-équivalente*** est maintenue quand les deux services considérés ont les mêmes pré-conditions, entrées, opérations, sorties et post-conditions. La relation ***est-partie*** apparut quand un service composé inclut l'autre.

- Service de domaine : C'est un service composé enrichi avec les propriétés organisationnelles. Ils sont augmentées avec des informations sur les partenaires demandeur/fournisseur, le temps de disponibilité, les contraintes de causalité, la synchronisation et les objets qu'ils manipulent : ressource, bon de commande, facture et produit. Trois types de relation entre les services de domaine sont considérés, ***avoir-même*** c'est une relation qui tient compte quand un service de domaine a le même domaine commercial qu'un autre. La relation ***est-complément*** est maintenue quand l'exécution d'un service de domaine implique l'exécution de l'autre. La relation ***est-indépendant*** montre que les deux services sont disjoignent.

Selon le modèle présenté de service, nous proposons un processus pour developper une ontologie de service, articulé en quatre étapes : identification des concepts et des rôles, description de service, mapping et clustering.

2.2.1 Identification des concepts et des roles

Dans cette étape, les caractéristiques de service sont identifiées comme concepts avec ses propriétés (les rôles et les attributs), qui sont représentées dans le tableau 6.1. Les concepts spécifiques des services tels que le profil, le cluster, les ressources, les messages, le partenaire, la location, la qualté et les exceptions sont intégrés dans l'ontologie de service. Comment moyen appropriés, les caractéristiques identifiée doivent être bien définie leurs poids dans le système (important, obligatoire, Bien-Avoir). Puis, le tableau 6. 2 montre le concept source et cible, cardinalité source et cible et le rôle inverse de chaque rôle.

Nom Concept	Synonymes	Instances	Attributs	Comment approprié ?	Roles
Service	Fonction Methode	test-disponibilite Invoque-achat faire-payment	ServiceID ServiceEtat ServiceType EstSimilaireA	Obligatoire Important Important Important	EstSimilaire AppartAuCluster ContientService Decrire AServiceProfil UtiliseParService ReagirAuMessage
ServiceCluster	ServiceGroupe	-	ClusterID NombreDeService	Obligatoire Bien-Avoir	AppartAuCluster ContientService ..
ServiceProfil	ServiceDescription	-	ServiceOperation ServOrdreExe ServInput/Output ServicePre/Post	Important Important Important Important	AServiceProfil Decrire ..
ServiceRessource	Ser-requirement Service-besoin	-	ServRessourceID ServRessourceType ServRessourceURL	Obligatoire Bien-Avoir Important	UtiliseRessource UtiliseParService ..
ServiceMessage	ServiceInvocation	-	MessageID MessageContenu	Obligatoire Important	ProduireMessage ReagirAuMessage
ServicePartenaire	Compagnie Fournisseur	-	FourniPartenaire DemanPartenaire	Bien-Avoir Bien-Avoir	Fournir Demander
..	..	..	..	..	..

Table 6.1: Dictionaire de concepts de l'ontologie de service.

Nom Role	Concept Source	Cardinalite Source	Concept Cible	Cardinalite Cible	Role Inverse
AppartAuCluster	Service	0..1	ServiceCluster	0..N	ContientService
EstSimilaire	Service	0..N	Service	0..N	EstSimilaire
Decrire	ServiceProfil	1..1	Service	1..1	AServiceProfil
ProduireMessage	ServiceMessage	1..N	Service	1..N	ReagirAuMessage
UtiliseRessource	Service	1..1	ServiceRessource	1..N	UtiliseParService
..	..	..	..	..	..

Table 6. 2: Description de rôles de l'ontologie de service.

2.2.2 Description et comparaison des services

Dans ce travail, la langage WSDL [9] est utilisé pour représenter la structure des web services en termes de propriétés d'interface, de comportement et de qualité. Dans cette étape, les services atomiques sont comparés sur la base de leurs propriétés pour évaluer leur similarité. Les relations sémantiques sont établis sur la base du similarité calculée. Deux services atomiques sont groupés dans le même cluster quand le poids de la relation sémantique les reliant est plus grand qu'un seuil.

2.2.3 Mapping des services

Afin de performer une analyse sémantique des services sur la base de leurs propres propriétés, nous adoptons un processus multi-stratégique pour calculer la similarité entre les entités en utilisant différents algorithmes. Nous utilisons le même processus de mapping inspiré du concept de sémantique bridge appliqué dans les ontologies d'application (voir chapitre 5). Nous calculons la similarité basée sur des propriétés d'interface, de comportement et de qualité pour chaque service source et service cible :

- **Similarité d'interface :** L'interface de service inclut les entités d'E/S, les opérations et les types de message. Pour performer une analyse sémantique d'interface de service, trois (3) types de similarité d'entités sont distingués. Pour chaque type, les experts du domaine affectent une valeur dans l'intervalle [0..1].

  - La similarité d' E/S se rapporte à l'évaluation des noms d'entités de l'information d'entrée-sortie des deux services Si et Sj, dénoté par E/SSim (E_i, E_j). Des ontologies lexiques tels que WordNet ou des thesaurus spécifique du domaine sont utilisés pour définir des relations terminologiques (par exemple synonyme et antonyme) pour l'acquisition de la similarité [63]. Les noms n_i et n_j des entités d' E/S sont comparés pour évaluer leur degré d'affinité A ().

$$A(n_i, n_j) = \begin{cases} 1 : \exists \text{ au moins un chemin des relations terminologiques dans le thesaurus entre } n_i \text{ et } n_j. \\ 0 : \text{Autrement.} \end{cases}$$

  Nous définissons E_i comme vecteur des entités d'E/S du service S_i et E_j comme vecteur des entités d'E/S du service S_j.

$$E_i = (e_{i1}, e_{i2}\ldots, e_{ik}) \Rightarrow |E_i| = k$$
$$E_j = (e_{j1}, e_{j2}\ldots, e_{jl}) \Rightarrow |E_j| = l$$

$$E/SSim(E_i, E_j) = \sum_{i,j} (A(n_i, n_j))/|E_i| + |E_j|$$

  - la similarité des opérations se rapporte à l'évaluation des opérations des deux sevices S_i et S_j dénoté OSim (O_i, O_j) est calculé en comparant leurs noms, leurs entités de l'information d'entrée-sortie basées sur WordNet ou des thesaurus spécifique du domaine. Deux opérations sont similaires si leurs noms et leurs entités d'E/S ont une affinité dans le thesaurus. La valeur de similarité de deux opérations est obtenue en sommant les valeurs d'affinité de leurs éléments correspondants. La valeur OSim (O_i, O_j) est la moyenne de paires d'opérations similaires, l'une du premier service et l'une du second.

  Nous définissons O_i comme vecteur des opérations du service S_i et O_j comme vecteur des opérations du service S_j.

$$O_i = (o_{i1}, o_{i2}\ldots, o_{im}) \Rightarrow |O_i| = m$$
$$O_j = (o_{j1}, o_{j2}\ldots, o_{j}n) \Rightarrow |O_j| = n$$

$$OSim(O_i, O_j) = \sum_{i,j} (A(n_i, n_j))/|O_i| + |O_j|$$

- Similarité de message porte sur le nom et le type de message pour mesurer la similarité MSim (S_i, S_j). Deux services ayant le même type des messages (reçu resp. produit) et leurs noms ayant une affinité dans le thesaurus, sont similaires.

La similitude globale d'interface ISim(S_i, S_j) pour chaque paire de services est évaluée en prenant une somme pondérée de similarités d'E/S, d'opération et de message. Les services où la similarité de l'interface est plus grande qu'un seuil sont les seuls pris en considération.

$$ISim(S_i, S_j) = (E/SSim(E_i, E_j) + OSim(O_i, O_j) + MSim(S_i, S_j))/3$$

- **Similarité de comportement :** Elle concerne l'évaluation des ensembles d'opérations des services. Un service est défini comme un vecteur S à N dimension :

$$S = (O_1, O_2 \ldots, O_N)$$

$\forall i \in \{1, 2\ldots, N\} / O_i$ est une opération du service S.

La similarité de comportement se fonde sur l'idée du concept du sémantique bridge [33], qui signifie que pour chaque opération source dans le premier vecteur il y a une opération cible dans le second, donnant deux services S_i et S_j où leurs vecteurs ont la même dimension, l'opération o_{ik} du service S_i est unifié avec l'opération o_{jh} du service S_j si :

1- o_{ik} et o_{jh} ont le même nom ;
2- o_{ik} et o_{jh} ont les mêmes paramètres d'E/S ;
3- o_{ik} et o_{jh} ont des préconditions (resp. postconditions) logiquement équivalentes.

Ceci signifie que les deux opérations sont équivalentes.

Nous pouvons maintenant étendre la définition du sémantique bridge à deux services. Deux services sont bridgés s'il y a une relation bijective entre leurs vecteurs associés. Chaque opération dans le premier service doit être compatible avec l'opération correspondante dans le second service.

Quand l'opération source n'a pas une opération cible qui signifie que l'opération source n'a pas une contre-partie directe dans le vecteur cible. Par exemple, si une entreprise envoie un ordre d'achat et reçoit la confirmation en tant qu'une opération simple dans le service, elle ne peut pas s'aligner avec une autre entreprise qui envoie l'ordre et reçoit la confirmation en tant que deux opérations séparées dans un autre service. À cette fin, l'opération source peut être bridgé avec la composition des deux opérations cible.

- **Similitude de qualité :** Chaque service est caractérisé par un ensemble de paramètres de qualité fournis par des standards de classification (par exemple, le standard ISO 9000). Quelques services peuvent être fiables et rapides en réponse ; d'autres peuvent être incertains, lents, ou même malveillants. Des paramètres de qualité spécifique application sont considérés (par exemple, le nombre de cartes crédit accepté par un service en ligne de réservation de billet). Chaque paramètre de qualité est décrit au moyen d'un type, d'une métrique et d'un prédicat. Le type de qualité signifie le caractère quantitatif et qualitatif du paramètre. La métrique indique l'unité de mesure (e.g: second …). Le prédicat est une expression logique (e.g: temps de réponse < 1s). La similarité de qualité des deux

services S_i et S_j, dénotée QSim (S_i, S_j) est évaluée en comparant leurs propriétés. S_i et S_j sont similaires ssi :

1- S_i et S_j ont le même type ;
2- S_i et S_j ont la même métrique ;
3- S_i et S_j ont des prédicats logiquement équivalents.

En conclusion, la similarité globale GSim de chaque paire de services S_i et S_j est évaluée en prenant la somme pondérée d'ISim (S_i, S_j), BSim (S_i, S_j) et QSim (S_i, S_j). Seulement les services où la similarité est plus grande qu'un seuil seront pris en compte

Le mécanisme d'inference est utilisé quand le service source n'a pas un service similaire cible. Ce qui signifie que le service n'appartient à aucun cluster.

2.2.4 Clustering et classification des services

Le résultat de la description basée-similarité est un ensemble de cluster de services similaires avec des relations de similarité définies. Un algorithme de clustering hierarchique [69] est employé pour déterminer les cluster basés sur la similarité globale établie entre les services. Quelques problèmes apparaissent, quand le service appartient à plusiers clusters. Si l'intersection des clusters n'est pas l'ensemble vide ainsi, ils doivent être fusionnés.

Après l'application des quatre (4) étapes, l'effort est concentré sur OWL DL, qui est équivalent au SHOIN(D) pour décrire formellement l'ontologie de service obtenue [38]. Une définition de concept est une relation de la forme A:=D, où A est un nom de concept et D est une description de concept. La formalisation des concepts : Service, ServiceProfil, ServiceCluster sont présentés dans le tableau 6.3.

Concept	Definition	Relation de subsomption
Service	(∃ ServiceID.String) ∩ (∃ AServiceProfil. ServiceProfil) ∩ (≥1 AServiceCluster. ServiceCluster) ∩ (∃ ReagirAuMessage. ServiceMessage) ∩ (≥0 EstSimilarA. Service) ∩ (≥1UtiliseRessource. ServiceRessource).	Service ⊆ Thing.
ServiceProfil	ServiceProfil := (∃ ServiceOperation.String)∩ (∃ ServiceOrdreExe.Integer)∩ (∃ Decrire.Service)∩(∃ ServiceInput/Output.String) ∩ (∃ ServicePre/Post.Boolean).	ServiceProfil ⊆ Service.
ServiceCluster	ServiceCluster: = (∃ ClusterID.String) ∩ (∃ NombreDeService.Integer)∩(≥1 ContientService.Service).	ServiceCluster ⊆ Service.
..	..	..

Table 6. 3: Definition de concepts de l'ontologie de service dans TBOX.

Pour vérifier, nous avons utilisé les services d'inférence fournis par le formalisme de DL pour améliorer la qualité de l'ontologie de service developpée. L'utilisation du system RACER [39] peut rendre possible la lecture des fichiers OWL pour les convertir sous forme de bases de connaissance de DL. Ce système permet également de faire des services d'inférence. Pour manipuler l'ontologie de sevice, Protégé-OWL [50], version 3.1.1 offre une interface utilisateur graphique convivial. Il permet de visualiser et éditer les relations hiérarchiques d'ontologie. Par exemple, la propriété de transitivité de rôle EstSimilaire est décrite par OWL DL comme suit.

<owl: ObjectProperty rdf: ID="EstSimilaire">
<rdf:type rdf:resource ="http://www.w3.org/2002/07/owl#TransitiveProperty"/>
<rdfs:range rdf:resource ="#Service"/>
<rdfs:domain rdf:resource ="#Service"/>
<owl:inverseOf rdf:resource ="#EstSimilaire"/></owl:ObjectProperty>

3 Stratégie de composition

Les processus collaboratifs sont définis comme une abstraction de processus métier complexes impliquant différents partenaires. Ils indiquent l'interaction entre les web services et leur coordination. Dans ce travail, les processus métier sont définis comme une composition des services de domaine. Seulement au moment de l'exécution, pour un service de domaine donné, un services composé est invoqué. Si le service demandé n'est pas disponible, un autre service similaire dans le même cluster est appelé. À cette fin, une stratégie de composition est définie comme suit :

Une SC (Stratégie de Composition) des services web est formellement définie comme un ensemble de triplets :

SC = (S, R, C)
S : un ensemble (S_1…, Sn) des services de domaine.
R : une requête client.
C : un schéma de composition qui accomplit la requête R en orchestrant convenablement les services (S_1…, Sn).

Le schéma de composition C est défini comme un ensemble de triplets :

C = (Pc, Pr, Co)
Pc : composition potentielle des services de domaine.
Pr : Partenaires impliqués dans la composition.
Co : Contraintes utilisées pour le choix de composition courante (disponibilité de service, qualité de service, temps de réponse, coût…).

Le schéma de composition est supervisé par deux mécanismes :

3. 1 Manager de composition

Ce mécanisme utilise l'information fournie par l'annuaire d'UDDI pour établir une classification de compositions potentielles selon les contraintes organisationnelle et la requête client. Il essaye de trouver une conformité entre les contraintes organisationnelle et celles du client. Ce dernier analyse les contraintes exigées par le client et vérifie la disponibilité des services pour déduire la meilleure combinaison de l'ensemble de composition. Après avoir pris une décision, il envoie la composition choisie au superviseur. Le comportement de manager de composition est décrit dans la figure 6.1.

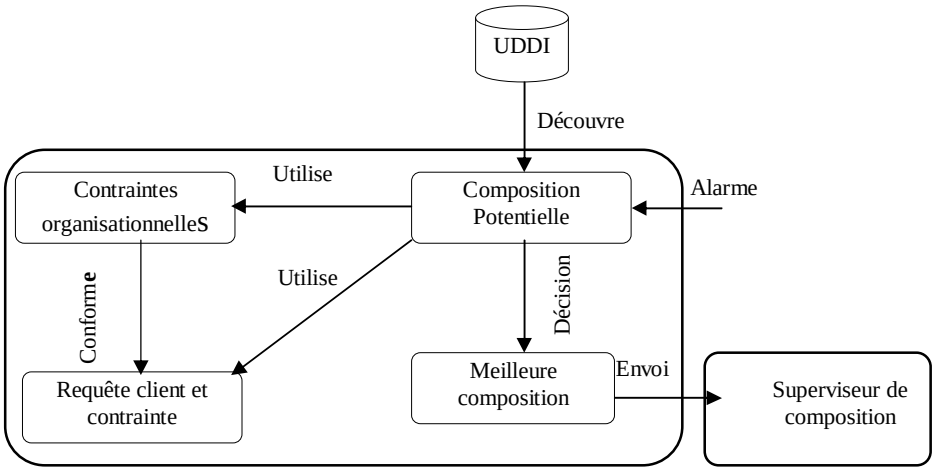


Figure 6.1 : Manager de composition.

3. 2 Superviseur de composition

Le mécanisme de supervision est le noyau du schéma de composition. Il surveille le moteur d'exécution de processus afin de commander les contraintes et manipuler les exceptions possibles. Il appelle les services appropriés après réception de la meilleure composition. Ce mécanisme emploie les contraintes du choix pour contrôler l'exécution des services. Il est basé sur des mécanismes fournis par le langage BPEL4WS. Le mécanisme de compensation (portée) est employé pour arrêter l'exécution de la composition courante et envoyer un signal d'alarme au manager pour choisir une nouvelle combinaison. D'ailleurs, le mécanisme d'exceptions (throw, catch) se rapporte à superviser les problèmes au moment de l'exécution (cf. figure 6.2).

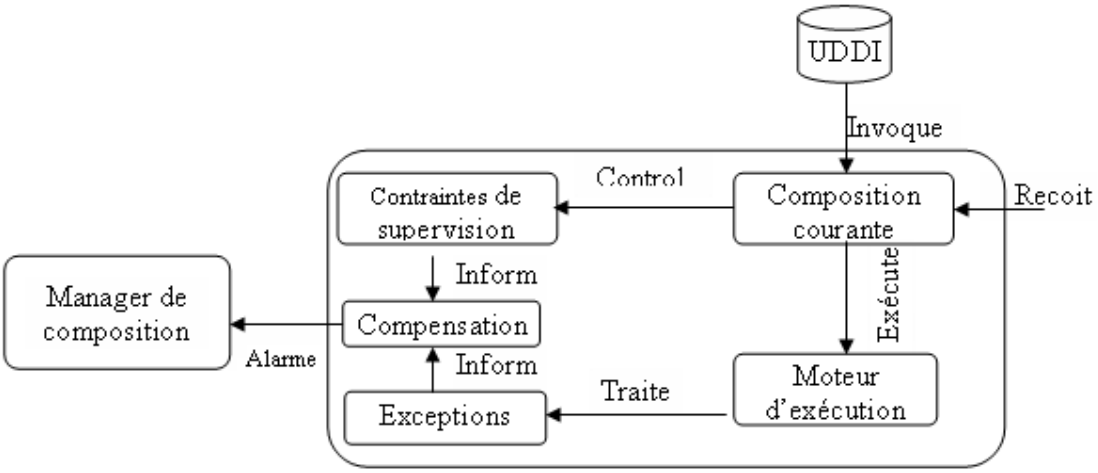


Figure 6.2 : Superviseur de composition.

4 Ontology des processus métiers

Dans notre travail, le scénario d'intégration inclut les quatre (4) vues de processus métiers : collaboratif, fonctionnel, organisationnel et technique. Le schéma de composition montre la gestion du processus métier (invocation et ordonnancement de services au moment de l'exécution). Chaque service est enrichi par des contraintes sur le traitement et les données basés sur les information du fournisseur. Puis, nous définissons les contraintes de causalité pour donner l'ordre d'exécution supervisé par les objectifs des partenaires. Le schéma dynamique résultant est conformé au besoins du client. À cette fin, le schéma de composition est l'ensemble des étapes à exécuter pour le management de processus métier.

Dans cette section, nous présentons le scénario d'intégration des processus métier.

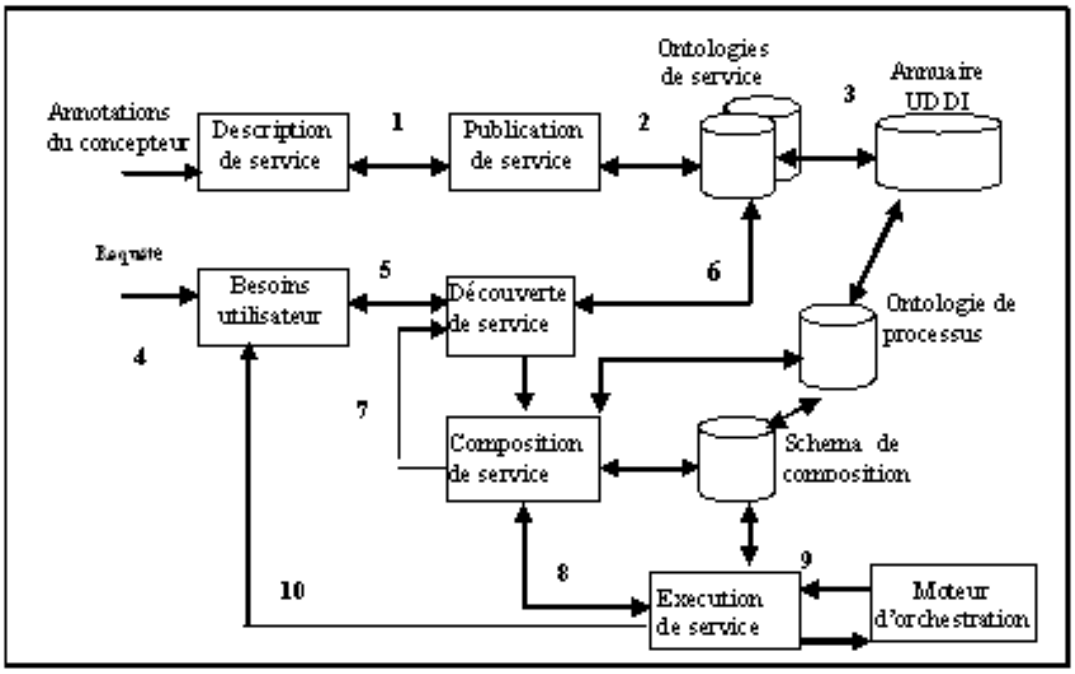


Figure 6.3 : Scénario d'intégration des processus métier.

Comme représenté sur la figure 6. 3, le scénario d'intégration commence une fois que les services web ont été décrits en utilisant le langage WSDL (1). Puis, ces services sont publiés dans les d'ontologies de service et dans l'annuaire UDDI (2, 3). Après, les besoins utilisateur sont utilisés pour détecter les services demandés (4). Quand les services sont identifiés, ils peuvent être invoqués. Si un service donné n'est pas disponible au moment de l'exécution, un autre service similaire dans le même cluster est sélectionné et invoqué dans l'ontologie de service approprié (5, 6). Une fois que les services nécessaires sont découverts, une stratégie de composition est utilisée pour former le processus métier selon le schéma de composition. Ces services sont ordonnés selon les contraintes des partenaires impliqués et celles du client pour choisir la meilleure composition (7). Dans cette approche, une trace de chaque schéma de composition est maintenue dans le dépôt pour réduire au minimum le temps de réponse.

Les composition courantes sont exécutées sous le contrôle du superviseur par un moteur d'orchestration (8). Ce dernier surveille l'exécution de service et manipule les exceptions possibles (9). En conclusion, le résultat est retourné à l'utilisateur (10).

5 BPEL4WS et les processus métiers

Les languges basées processus courant telles que BPEL4WS [43] supportent la flexibilité, avec le contrôle de flux commun qui construit le séquence, l'itération, et/ou le choix. BPEL4WS représente l'union des langages de spécification WSFL [53] et de XLANG [9]. Cette fusion a créé la consolidation du marché pour faire de BPEL4WS un standard pour spécifier les web services composants le processus métier. Structurellement, un fichier BPEL4WS décrit l'ordre par la définition des participants, les services qu'ils doivent mettre en application afin d'appartenir à la composition, et le contrôle de flux de processus.

Par exemple, dans une agence de tourisme, le processus de réservation a besoin de plusieurs services orchestrés dans cet ordre : un service pour la réservation de vol, un service pour la réservation d'hôtel et un service pour le paiement. L'agence de tourisme collabore avec des companies des lignes aériennes, des hôtels et des banques pour satisfaire la requête client. Pour chaque service, le partenaire fournisseur donne ses contraintes organisationnelles pour l'exécuter. Chaque contrainte est décrite par un nom (nom de la contrainte), type (contrainte-opération, contrainte-donnée), un partenaire (nom partenaire), un prédicat (expression logique), un ordre (service-avant, service-après) et une fréquence (le nombre de fois de vérifier la contrainte). Nous nous servons du langage de BPEL4WS pour décrire le processus métier :

- Definition des partenaires

```
<partners>
< partner name=”custmor” partnerRole= ”clienttourismagency”>
< partner name=”webserviceairlinecompany” partnerRole =”reservationticket”>
< partner name=”webservicehotel” partnerRole =”reservationroom”>
< partner name=”webservicebank” partnerRole =”confirmationpayment”>
</partners>
```

- Definition des constraintes organisationnelle

```
< name= “c1- card-credit” constraint type=”data-constraint” partner=” webserviceairline ” predicate=”card credit = valid” order =”before : reservationticket” frequency=”1” constraint/>
< name=”c2-card-credit” constraint type=”operation-constraint” partner=”webservicehotel” predicate=” card credit = valid” order=”after: reservationticket ” frequency=”1” constraint/>
< name=”c3-confirmationpayment” constraint type=”message-constraint” partner=”customer” predicate=”card credit = valid” order =” after: reservationroom” frequency=”3” constraint/>
```

- Definition des types de données et des messages

```
<variables>
<variables name= “request” messagetype =”typerequest”>
< variables name= “recepisse” messagetype =”typerecepisse”>
< variables name= “reservationticket” messagetype =”typereservationticket”>
< variables name= “reservationroom” messagetype =”typereservationroom”>
< variables name= “confirmationpayment” messagetype =”typeconfirmationpayment”>
<variables/>
```

- Definition de processus métier

```
<sequence name=”seqsejourreservation” name= “c1- card-credit” >
```

<receive partner=”customer” operation “sejourreservation” variable= “request”>
<flow>
<invoke partner=”webserviceairline” operation =”reservationticket”>
<invoke partner=”webservicehotel” name=”c2-card-credit” operation =”reservationroom” >
</flow>
<invoke partner =”webservicebank” operation = “confirmationpayment”>
<reply partner =”customer” name=”c3-confirmationpayment”operation=”sejourreservation”
variable=”recepisse”>
</sequence>

6 EbXML et la collaboration B2B

L’architecture ebXML définit les processus métier entre les partenaires par la description des compétences d’une entreprise et l’établissement d’ententes de collaboration. Les technologies des Web services fournissent donc l’infrastructure logicielle pour les interactions interentreprises et l’agent sert à un représentant de l’entreprise pour la recherche et la négociation.. Notre système se compose ainsi : d’un ensemble de partenaires disposant, chacun, de son propre agent, d’un ensemble de spécification des profils en langage WSDL, d’un registre UDDI pour le référencement et la recherche des profils et d’un protocole de communication basé sur SOAP. Nous détaillons par la suite la structure et le role de chaque agent.

6. 1 Rôle et structure des agents

Les agents sont introduits afin d’optimiser le scénario en terme de temps et d’efficacité.

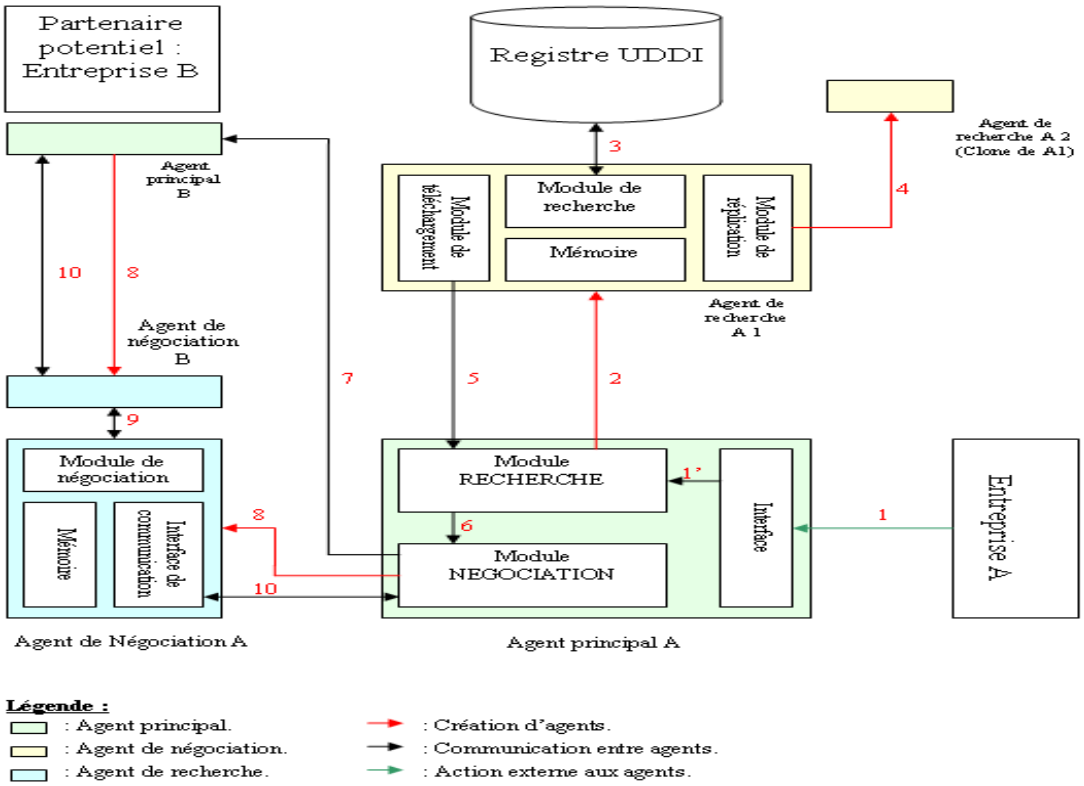


Figure 6. 4 : Types des agents et leurs rôles.

Trois types d'agents sont proposés : un agent principal, un agent de recherche et un agent de négociation. L'agent principal est responsable de la création et la coordination avec les deux autres agents. Seuls l'agent de recherche et l'agent négociateur sont mobiles. Les différents types d'agents ainsi que leurs structures sont représentés dans la figure Figure 6.4.

- L'entreprise A, déjà informée de l'existence des registres ebXML, invoque l'agent par une requête qui contient les informations sur l'affaire commerciale que veut mener l'entreprise (action 1). L'action 1' présente l'extraction des différents paramètres du message qui vient d'être reçu au niveau de l'interface.

- Le module recherche, après avoir reçu les paramètres requis, crée un agent de recherche qui migre vers une adresse connu à l'avance (action 2), celle d'un registre ebXML. Dans l'absence d'autres registres, un seul agent suffit pour accomplir la tâche de la recherche (action 3) mais on peut envisager le clonage d'un agent de recherche sur le même site pour améliorer les performances dans le cas où le registre est trop volumineux. L'agent de recherche peut se cloner et envoyer ses répliques vers d'autres sites si d'autres registres ebXML existent (action 4). Ces répliques peuvent eux aussi se cloner migrer leurs répliques sur d'autres sites. Pour qu'un agent ne visite pas un site qu'un autre a déjà visité, chaque agent qui crée un autre lui donne la liste des adresses de registres qu'il a hérité, excepté celle du parent, celle du fils et toutes celles de ses frères déjà créés. Cette méthode n'élimine pas toutes les redondances de recherches mais elle permet d'établir un compromis entre la perte du temps et des ressources dans des recherches déjà faites, et l'excès de messages qui peut parfois être pénalisant sans résultat satisfaisant.

- Tout agent qui ne retourne pas de résultats dans un délai prédéfinit selon les besoins de l'entreprise, est ignoré. L'introduction d'un délai de réponse justifie l'utilisation du clonage local pour partager la tâche de recherche entre plusieurs agents quand le registre est volumineux.

- Tout agent qui n'aboutit pas à un résultat termine, il n'a pas besoin d'informer les autres agents car il sera ignoré dans un bout de temps défini.

- Tout agent qui aboutit à un résultat, envoie un message direct à l'agent principal (action 5), dont l'adresse est connue par tous les agents recherche. Ce message contient un document WSDL contenant les informations nécessaires sur une certaine entreprise (ou une branche d'entreprise).

- Le module recherche filtre les résultats reçus et les envoie au module negociation (action 5). Le filtrage peut être fait au fur et à mesure de la réception des données pour améliorer les performances en terme de temps. Il peut être effectué par le module Recherche ou par un module séparé.

- Le module negociation informe l'agent principal d'une entreprise B (action 7), par exemple, qu'ont révélé les recherches comme partenaire potentiel. L'adresse de l'entreprise ainsi que celle de son agent principal se trouvent dans le document téléchargé lors de la phase de recherche. Les deux agents principaux créent alors un agent négociation pour chacun (action

8), il entrent en négociation (action 9) et restent en communication avec l'agent principal (action 10).

6.2 Scénario de collaboration étendu

Le schéma de la figure 6. 5 présente l'extension du scénario de collaboration d'ebXML par un ensemble d'agents et les Web services. L'utilisation des Web services peut apporter plus de flexibilité, de simplicité, de neutralité et d'efficacité dans ce scenario. En plus, les Web services possèdent des caractères d'universalité et d'ubiquité ce qui offre un framework interoperable entre les partenaires. Bien qu'à l'heure actuelle, la technologie ne soit pas encore viable, dans la mesure où elle manqué de maturité, notamment au niveau de la securité et le support des transactions :

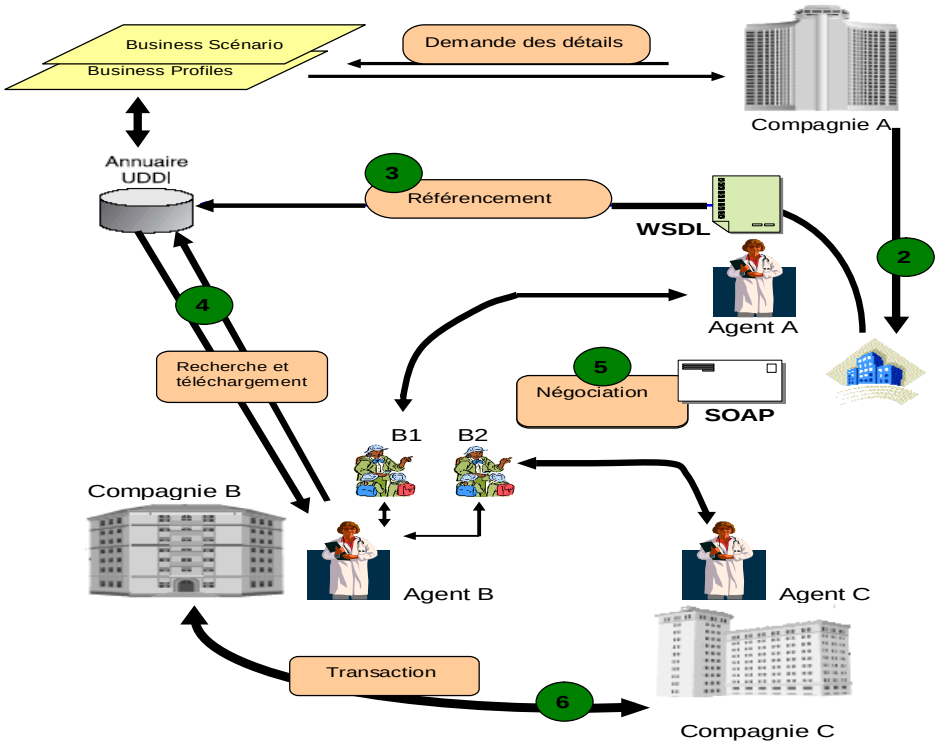


Figure 6. 5 : Scénario de collaboration étendu.

1. La compagnie A est informée qu'un registre ebXML (UDDI) est accessible sur Internet.
2. La compagnie A, après avoir consulté le contenu du registre UDDI, décide de construire et déployer son propre système ebXML.
3. La compagnie A soumet alors l'information concernant le profil d'affaire de A (WSDL) au registre UDDI.
4. L'agent principal d'une compagnie B découvre les scénarios d'affaires supportés par une compagnie A ou plusieurs, et lance ses agents de recherche qui seront présents sur les différents sites opérateurs du registre UDDI pour effectuer la recherche et le téléchargement des documents WSDL (décrivant les formats des messages attendus par A).

5. Les agents de négociation de B peuvent collaborer simultanément avec les autres agents (par envoi des requêtes et des réponses SOAP). Chaque agent négociateur soumet à l'agent de négociation d'une compagnie une notification de demande d'établissement d'une relation d'échange économique. Après collaboration, l'agent principal B, suivant les résultat obtenu (par ses agents négociateurs), décide avec quelle compagnie établir une transaction commerciale.
6. les deux compagnies peuvent dès lors engager des transactions commerciales.

7 Conclusion

L'intégration sémantique est un problème majeur qui peut affecter les données, les applications et les processes. Une solution candidate est l'utiliation d'une approche combinant les ontologie et les web services. Les ontologies fournissent la base sémantique pour unifier la terminologie dans les modèles d'applications. Les web services encapsulent l'hétérogénéité des processus métiers et leur composition fournit un support pour orchestrer l'éxecution.

Nous avons décrit une architecture d''intégration des applications avec deux niveaux : applicatif et collaboratif. Elle est complète dans le sens qu'elle (a) comporte différents niveaux d'intégration et (b) inclut des ontologies pour la gestion de l'hétérogénéité et assurer la flexibilité dans les processus métiers. Elle est générique parce qu'elle n'est pas conçue en fonction des technologies ou des applications spécifiques.

Nous jugeons que les résultats présentés ici donnent beaucoup de perspectives intéressantes pour les chercheurs qui souhaitent explorer l'intégration B2B plus complètement. Notre architecture sera également utile pour les praticiens qui souhaitent évaluer la flexibilité de l'intégration B2B.

CHAPITRE 7

Etude de cas : illustration et réalisation

1. Introduction

Dans ce chapitre, nous allons présenter une réalisation de notre proposition à l'aide de l'exemple des agences de voyage qui permettent d'offrir des reservations des billets, de chambre d'hôtels et de location de voiture. Nous présentons le processus de construction de l'ontologie d'application avec toutes ses étapes, partant de la méta-modélisation à l'implémentation. Au niveau collaboratif, nous détaillons le scénario de collaboration EbXML étendu par les agents de recherche et ceux de négociation, ainsi l'apport du scénario d'intégration des processus métier pour bien illustrer la collaboration entre les partenaires.

Les applications sont souvent hétérogènes, et plus l'intégration sera fait à deux niveaux A2A et B2B des agences de voyage. L'intégration s'effectue en deux (2) étapes :

- L'intégration des applications de l'agence de voyage entre elles ;
- L'intégration avec les partenaires : les hôtels, les compagnies aérienne, les agences de location des voitures et les clients.

2. Agences de voyage

Une agence de voyage nécessite le developpement d'une ontologie de voyage pour représenter explicitement la connaissance qui sera employée par ses partenaires. Nous nous focaliserons au reservations des billets, de chambre d'hôtels et de location de voiture. Nous savons que quand un client fait un voyage, il choisit le moyen de transport et la chambre. Par conséquent, nous commençons par déterminer le moyen de transport qui sont actuellement disponibles pour l'agence de voyage. Nous aurons dans notre ontology les suivants : avions, trains, voitures, et bateaux. De tous, l'agence de voyage est particulièrement intéressée par les vols, car c'est le moyen de transport la plupart du temps utilisé par ses clients. Nous savons que chaque modèle de transport appartient seulement à un genre de transport (par exemple, c'est un avion, ou un autobus, ou une voiture, etc…

Pour chaque vol, l'agence sait : la date de départ, la date d'arrivée, la ville de départ, la ville d'arrivée, l'aéroport de départ, l'aéroport d'arrivée, les prix sur la première classe, classe d'affaires et classe d'économie, le temps de départ et le temps d'arrivée. L'heure et la date seront considérées en tant que date absolue. Quant aux destinations des voyages des clients, elles sont diverses. Les destinations les plus communes sont les grandes villes, Londres, Paris Madrid, Cairo, Tunis, etc. Nous savons que le client peut employer le transport suivant pour se déplacer à l'intérieur de la ville : autobus, taxis, et voitures de location.

Pour ce qui concerne les hôtels, l'agence recommande dans toutes les villes. Les catégories d'hôtels sont variés d'hôtels de 1 étoile à 5 d'étoile et à chaque hôtel appartient à une de ces cinq catégories. Pour tous, l'agence connaît leurs informations : l'adresse, le numéro de téléphone, URL, la capacité, nombre de salles, chambres disponibles, descriptions, distance à la plage, distance au ski, distance au centre ville, etc… L'agence connaît également les équipements des salles : nombre de lits, TV disponible, connexion d'Internet, etc… Une fois que nous avons défini ce qui sont les éléments principaux dans notre domaine, nous pouvons

aller plus loin et essayer de représenter quelques contraintes et déductions qui peuvent être exécutées. Par exemple, nous savons qu'il n'est pas possible d'aller de l'Amérique en Europe en train, voiture, vélo. Avoir cette information dans notre système l'évitera pour rechercher employer possible d'itinéraires ces le moyen de transport quand un client veut voyager à l'Europe.

3. Construction des ontologies d'application

4. Description des Web services liés au processus de réservation

Nous présentons les documents WSDL des services suivants : service réservation d'avion, service reservation d'hôtel et service réservation de voiture.

a- Service reservation d'avion :

< ? xml version= "1.0" encoding = "ISO-8859-1" ?>

<definitions

Name= "ReservationService-interface"

targetNamespace= "http://www.ws-aviation-xml.org:8080/reservation_sw-aviation-xml/services-type/ReservationService-interface"

xmlns:tns="http://www.ws-aviation-xml.org:8080/reservation_sw-aviation-xml/services-type/ReservationService-interface"

xmlns: soap= "http:// schemas.xmlsoap.org/wsdl/soap/"

xmlns: xsd= "http://www"

xmlns: xsd1= "http://www"

5. Scénario de collaboration

Les données nécessaires à l'élaboration d'un voyage personnalisé proviennent de fournisseurs différents (compagnies aériennes, chaînes hôtelières et compagnies de location de voitures).

Le système multi-agents de l'agence de voyage est constitué d'un ensemble d'agents reliés par l'intermédiaire de l'Internet et qui fournissent divers types de services :

- des agents représentant des compagnies aériennes, Air Algérie et Air France ;
- des agents représentant des compagnies de location de voitures ;
- des agents représentant des hôtels, Sofitel et Hilton ;
- des agents représentant des agences de voyage.

Nous voulons représenter la connaissance au sujet d'un voyage concret. Un client demande à son agent assistant d'organiser un voyage de Constantine à Paris pour passer un stage de 10 jours. Cet agent qui sélectionne des agents serveurs de voyage et organise l'itinéraire. Il effectue les réservations des billets d'avion (de Constantine à Paris), d'une voiture (de l'aéroport de Paris à l'hôtel) et la réservation de l'hôtel pour 10 nuits. Chaque agent représentant une compagnie (agence de voyage, compagnie aérienne, chaîne hôtelière ou compagnie de location de voiture) est un agent principal responsable de la création et la coordination avec les agents de recherche et de négociation.

Les agents de recherche de l'agence de voyage effectuent leurs recherches, sur l'annuaire UDDI, pour des compagnies aériennes (avec l'itinéraire Constantine Paris), des Hôtels à Paris et des compagnies de voitures de location. Nous supposons que l'agent principal de l'agence a eu comme résultats, de ses agents de recherche, Air Algérie et Air France comme compagnies aériennes faisant l'itinéraire Constantine à Paris, et Sofitel et Hilton comme Hôtels à Paris.

Ces différentes compagnies doivent avoir publier, au préalable, leurs Web servies (documents WSDL) dans l'annuaire UDDI (figure 7. 1).

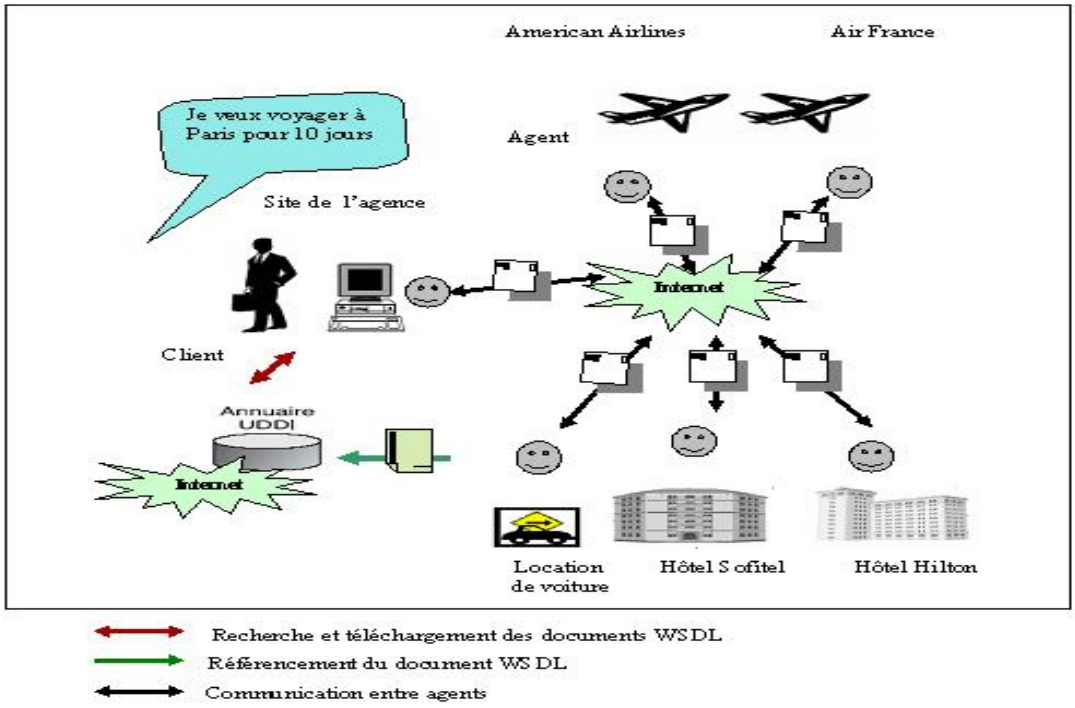


Figure 7. 1 : Présentation du scénario de l'agence de voyage.

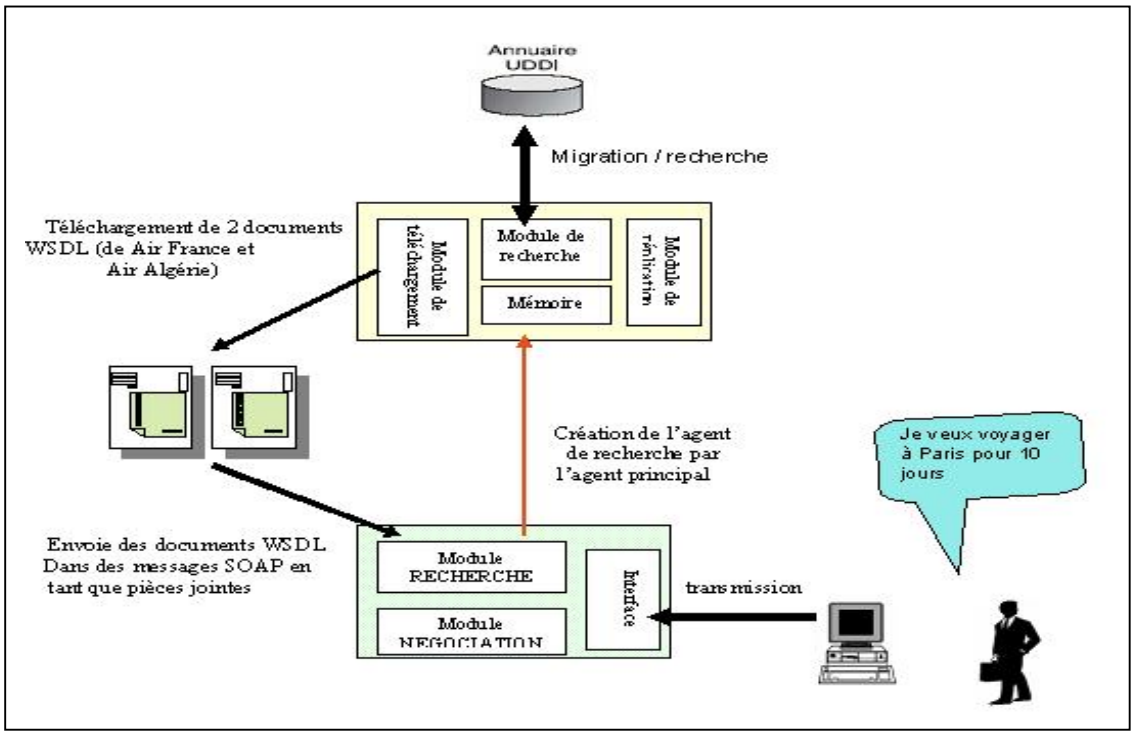


Figure 7. 2 : Comportement de l'agent de recherche.

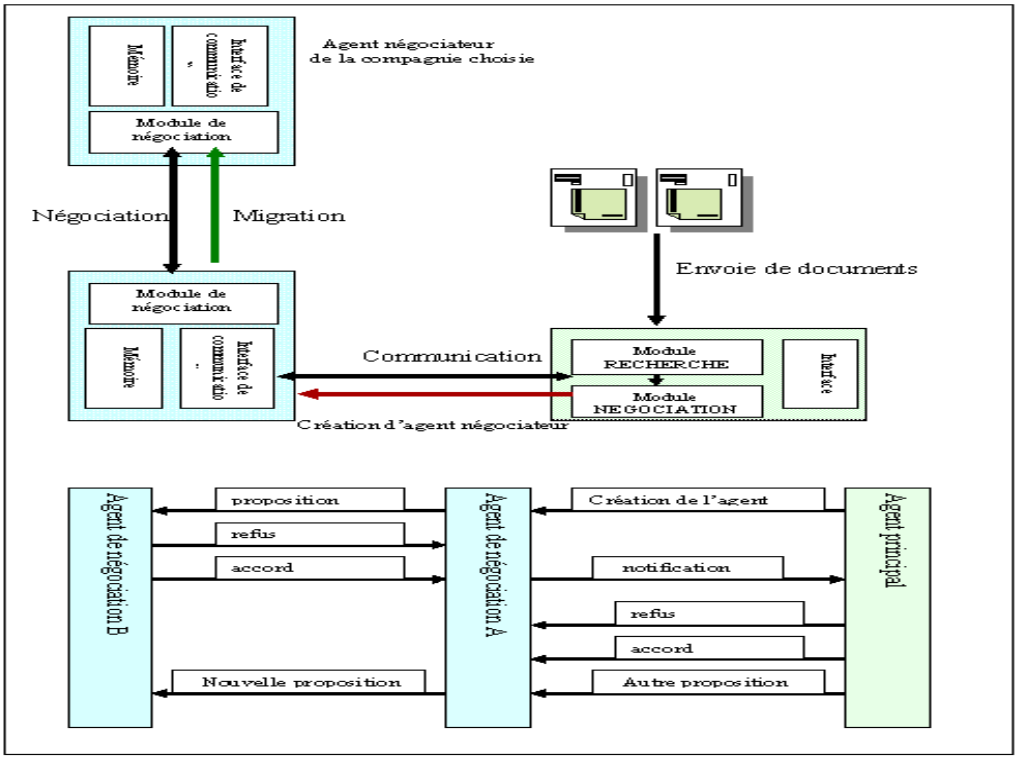


Figure 7. 3 : Migration et négociation de l'agent négociateur.

6. Stratégie de composition du processus de réservation

Après avoir choisi une agence de voyages parmi la liste disponible dans l'UDDI, le client lui envoie sa requête R ainsi que ses contraintes (prix fixé, délai, choix.…).

A la réception de R par l'agence touristique choisie, elle effectue les opérations suivantes :

2.1 Analyse de la requête R (déterminer les services qu'il faut)

2.2 Etude de la requête R (déterminer l'enchaînement des services)

2.3 Construction d'un SC (schéma de composition) après une chorégraphie et des services métiers pour l'exécution de la requête R

2.4 Invocation des services (pour retirer les différents partenaires répondant aux contraintes client et construire une SC final qui contient les services, contraintes client, contraintes partenaire et l'enchaînement avec tout les activités structurées .

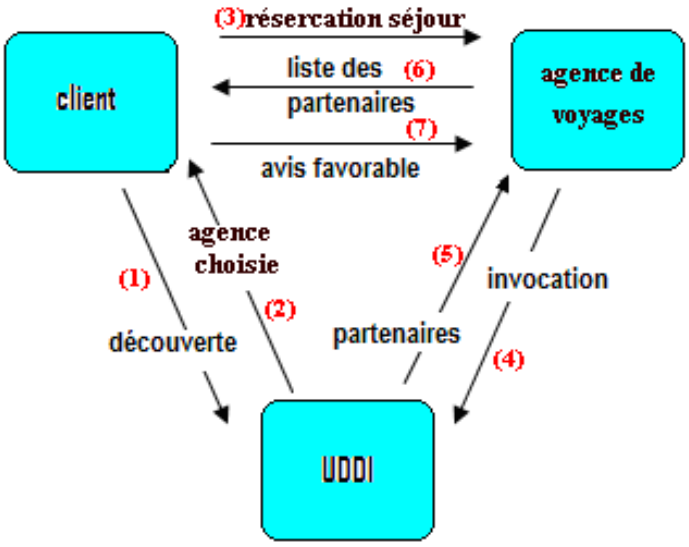


Figure 7.4 : Réservation de séjour.

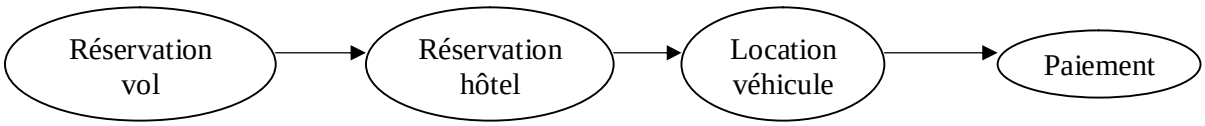
1. *opération de découverte :* le client lance une recherche au prés de l'UDDI, pour connaître les différents agences touristiques,
2. *l'UDDI :* lui fourni la liste des agences disponibles sur le Web,
3. *envoi de R :* le client envoi sa requête 'R' de réservation de séjour à l'agence qui a fait son choix

   - *À la réception de 'R' :* l'agence effectue les opérations suivantes :

Analyse de la requête : R est destinée pour une réservation de voyage
déterminer les services à invoquer :

S1 : location voiture
S2 : paiement
S3 : réservation vol
S4 : t réservation hôtel

Ici On fait le SC initial et globale : ici on marque que les services nécessaires pour le requête client, sans prendre en considération la listes des différents partenaires



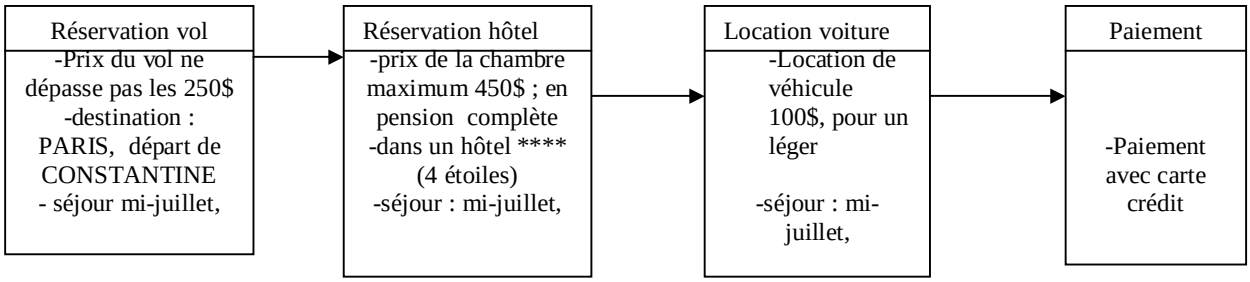
4. l'agence invoque l'UDDI, pour faire la chorégraphie des services nécessaires, pour la résolution de R, ces services doivent respecter les contraintes du client, sachant que pour chaque service invoqué, l'existence de plusieurs partenaires est possible.

Les contraintes client sont comme exemple :
Contraintes du client:

- destination : PARIS, départ de CONSTANTINE
- le voyage ne doit pas dépasser 800$
- prix du vol ne dépasse pas les 250$

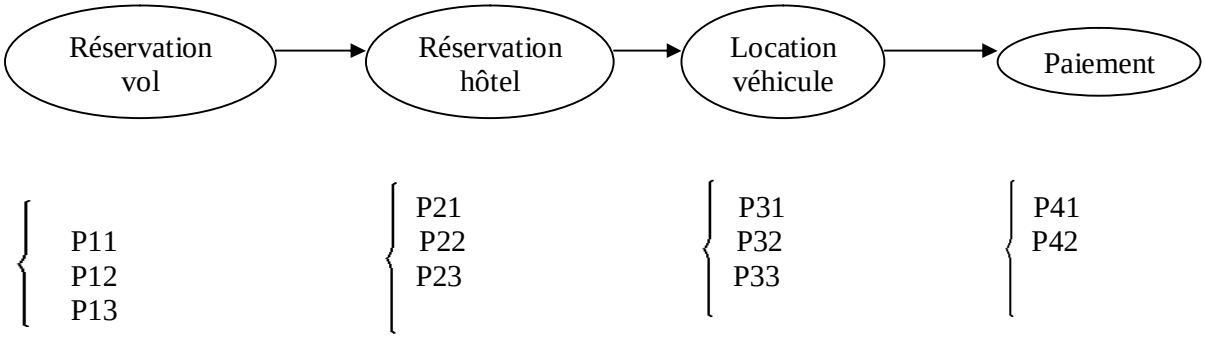
- prix de la chambre maximum 450$; en pension complète
- dans un hôtel **** (4 étoiles)
- séjour : mi-juillet (10 jours)
- Location de véhicule 100$, pour un léger
- Paiement avec carte crédit

Ici, l'agence lance la requête comme un diagramme d'état, on met pour chaque service ses contraintes imposées par le client comme suit :



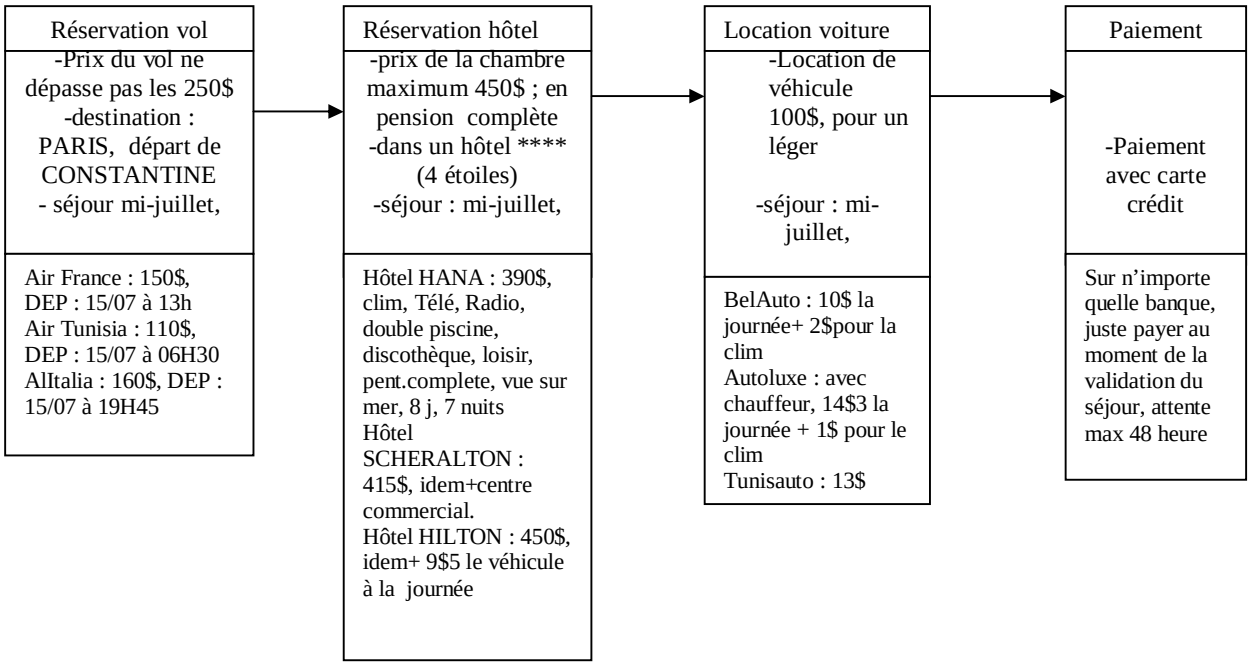
5. l'UDDI fournit à l'agence la liste des services avec pour chaque service les différents partenaires qu'il l'offre, en utilisant un SC bien détaillé, Regrouper les partenaires similaires de même service dans l'ordre (le premier et plus performant que le deuxième et le deuxième est plus performé que le troisième…etc, jusqu'au dernier).

Tracer le schéma de composition final qui contient les services et leurs partenaires comme suit :



Sachant que dans ce diagramme seuls services don les partenaires sont disponibles vont figurer.

6. à la réception de SC l'agence connaît les codes de chaque partenaire pour chaque service, l'instanciation de ces codes est montrés qu'au client sur le diagramme d'état final, en mentionnant aussi, les contraintes pour chaque partenaire offrant un service :



Ce diagramme est envoyé au client, avec les contraintes de l'agence, sur le schéma de composition les services qui appartiennent aux différents partenaires sont occupés, ils attendent la validation du client.

Contraintes de l'agence :

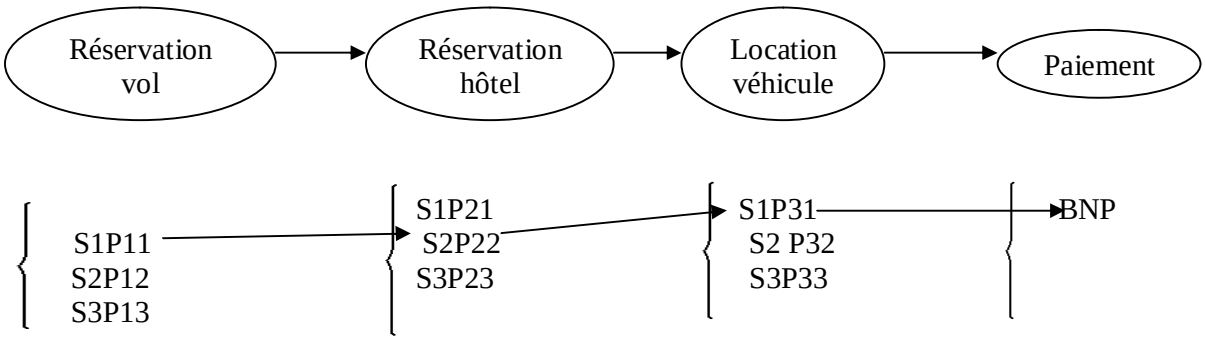
- Proposer les coûts les moins chers en premier
- Favoriser les partenaires avec qui l'agence a l'habitude de travailler le plus souvent
- Mettre les services qui ont un temps de réponse rapide en premier
- Attendre 72 heures, si le client ne s'est pas manifesté, alors annulation de toutes réservations
- En cas de réservation, valider le numéro de carte de crédit

7. une fois avoir fait son choix, parmi les propositions qui lui ont été offerts, le client le renvoi à l'agence pour établir le schéma de service, sur le SC, en délibérant les services qui n'ont pas fait le choix du client pour une éventuelle requête.

Le choix du client peut être par exemple comme suit :

- réservation de vol : Air France
- réservation hôtel : hôtel Shératon
- location voiture : BelAuto
- paiement : sur BNP Parisbas

Donc le SC devient comme suit :



: Le schéma de service : le chemin que suit la requête R réservation séjour

7. Modélisation du processus avec BPEL4WS

8. Conclusion

Dans ce chapitre, nous avons illustré notre proposition par une étude de cas sur l'agence de voyage. Nous avons developpé une ontologie d'application de reservation basé sur le processus proposé en chapitre 5. Après instantiation du méta-modèle, nous décrivons les concepts et rôles de l'ontologie. Pour les formaliser, nous avons adopté le foralisme de la logique de description. L'intérêt de ce dernier est qu'il est d'une part simple pour que les non-spécialistes puissent l'utiliser, et d'autre part, qu'il est plus expressif. Basé sur cette formalisation, nous avons choisi le langage OWL, pour codifier l'ontologie formelle, en utilisant l'éditeur graphique PROTEGE-OWL, afin de guider l'implémentation et générer le document OWL. Ensuite, le système RACER sert à raffiner et vérifier l'ontologie.

Au niveau collaboratif, nous avons developpé le scénario de collaboration EbXML étendu. D'abord, nous avons implémenté notre registre UDDI ainsi que les intéractions avec ce registre en java sur la plate-forme J2EE. Ensuite, nous avons conçu les documents WSDL pour le service voyage et les services qui le composent (les services de reservation de vols, de chambre d'hotels et de voiture de location). Nous avons généré leurs squelettes en java sur la plate-forme J2EE. Enfin, nous avons généré les composants des différents agents en code executable Java. Ces composants sont executés sur la plate-forme d'agents mobiles JADE (les agents de recherche et les agents de négociation).

Conclusion générale

1. Bilan

Ces dernières années, à l'avènement du concept d'intégration d'applications qui a révolutionné fortement la manière avec laquelle les applications de l'entreprise doivent et peuvent communiquer. Ce concept qui est à la fois récent et très ancien constitue un enjeu majeur pour les SI. L'intégration d'applications constitue sans doute un domaine prometteur en constante évolution. Ce qui montre son importance, notamment pour les grandes entreprises dont les enjeux se déclinent souvent en termes de souplesse et de réactivité de leurs SI.

Devant les limitations que présente l'EAI actuelle, nous avons évoqué quelques orientations qui nous semblent pertinentes. Et nous avons privilégié, dans le cadre de nos travaux, l'orientation basée sur la sémantique. Pour cela, nous avons proposé une architecture d'intégration basée ontologie qui est née de la volonté de pérenniser, rationaliser et flexibiliser davantage les SI d'entreprise. Notre proposition s'oriente dans le sens du renforcement de l'aspect connaissance dans les structures d'intégration, en suggérant de greffer une couche intelligente par le developpement des ontologies d'application. Nous avons retenu en plus la notion de la composition des Web service pour décrire l'enchainement des processus métiers, cette direction nous semble plus prometteuses.

L'intégration sémantique concerne les données, les applications et les processus. Ceci nécessite une approche basée ontologie pour le résoudre.

L'approche proposée assure l'intégration à deux niveaux applicatif (A2A) et collaboratif (B2B):

1. Niveau applicatif (Intégration A2A)

- Développement de l'ontologie d'application (Protégé OWL).
- Raffinement du processus d'intégration d'ontologies basées sur le framework MAFRA et le concept de sémantic bridge.
- Proposition d'un scénario de communication pour établir l'échange entre les applications hétérogènes et satisfaire le client.

2. Niveau collaboratif (Collaboration B2B)

- Construction d'ontologie des processus par l'intégration des ontologies de services.
- Étude du langage BPEL4WS pour développer notre stratégie de composition des Web services.
- Extension du scénario de collaboration EbXML par l'introduction des agents mobile pour assurer d'interopérabilité entre les entreprises partenaires utilisant des modèles d'échanges hétérogènes.
- Implémentation du scénario de collaboration avec Jbuilder (version X) et la plateforme Jade.

2. Perspectives

Nous avons décrit une architecture d'intégration des applications avec deux niveaux : applicatif et collaboratif. Elle est complète dans le sens qu'elle (a) comporte différents niveaux d'intégration et (b) inclut des ontologies pour la gestion de l'hétérogénéité et assurer la flexibilité dans des processus métiers. Elle est générique parce qu'elle n'est pas conçue en fonction des technologies ou des applications spécifiques.

Nous jugeons que les résultats présentés ici donnent beaucoup de perspectives intéressantes pour les chercheurs qui souhaitent explorer l'intégration B2B plus complètement. Notre architecture sera également utile pour les praticiens qui souhaitent évaluer la flexibilité de l'intégration B2B.

Il y a un certain nombre de directions intéressantes pour des travaux de recherche future. La première direction doit se concentrer sur la reconfiguration dynamique de la stratégie de composition des web services pour améliorer la qualité de service. Une deuxième voie doit détailler et tailorer l'architecture pour des technologies spécifiques. À cette fin, nous tiendrons compte de l'hétérogénéité dans les normes de l'échange comme RosettaNet et BizTalk afin de proposer un portail pour supporter la collaboration des partenaires [53]. Un troisième axe doit détailler et examiner empiriquement la similarité dans le processus de mapping en utilisant des propriétés de contexte. Une quatrième orientation pourrait adresser quelques perspectives des ontologies de service : (i) expérimentation pour valider l'ontologie de service ; (ii) utilisation du framework MAFRA [36] pour supporter l'intégration des ontologies de service et (iii) l'amélioration des similarités de services basé sur les aspects d'OWL.

3. ESB : prochaine génération de l'EAI

Les ESB (Enterprise Service Bus) ne sont autres que des EAI standardisés. Pour ce faire, il fait appel à la technologie XML à toutes les étapes du processus d'intégration. Ils exploitent ce langage de structuration de données pour décrire à la fois le vocabulaire d'appel aux applications mais également le format des messages en lui même (ou connexion WSDL/SOAP). Là encore, les Web Services entrent en action... Avec l'évolution des travaux de standardisation, il est fort à parier que cette tendance à la normalisation gagnera à terme les niveaux les plus élevés d'un EAI, jusqu'au BPM et au BAM (Business Activity Management).

Glossaire

- **A2A** : Application to Application, mot tendance mode, style B to B et B to C, désignant l'intégration des applications entre elles.

- **T*axonomie* :** ou *taxinomie* (du grec taxis : rangement et nomos : loi) est l'étude théorique de la classification, de ses bases, de ses principes, des méthodes et des règles. La taxonomie permet d'organiser un vocabulaire sous une forme hiérarchique simple. Cette hiérarchisation correspond souvent à une spécialisation, autrement dit, c'est la classification des termes d'un domaine dans une hiérarchie.

- **Jena** : Framework pour le web sémantique en Java, Jena fournit un environnement permettant de travailler avec RDF, RDFS et OWL.

- **Métadonnée** : Une métadonnée est une information permettant de décrire une autre information, quels qu'en soient la nature et le support.

- **Ontologie :** A l'origine domaine philosophique de la « science de l'être en tant qu'être », une ontologie est, selon l'entendement du Web sémantique, un ensemble structuré de savoirs. Une ontologie définit les termes employés pour décrire et représenter un domaine de connaissance.

- **OWL** : Web Ontology Langage

- **Protégé** : éditeur d'ontologies et framework de gestion des connaissances, développé en open-source au sein de l'université de médecine de Stanford.

- **RDF** : Ressource Description Framework, permet de présenter des données et des métadonnées.

- **RDFS** : RDF Schema, permet de définir des vocabulaires RDF.

- **CRM** : Customer Relationship Management, ou en français GRC (Gestion de la Relation Client) vise à proposer des solutions technologiques permettant de renforcer la communication entre l'entreprise et ses clients afin d'améliorer la relation avec la clientèle en automatisant les différentes composantes de la relation client.

- **ERP** : (Enterprise Resource Planning) sont des applications dont le but est de coordonner l'ensemble des activités d'une entreprise (activités dites verticales telles que la production, l'approvisionnement ou bien horizontales telles que le marketing, les forces de vente, la gestion des ressources humains, etc.) autour d'un même SI. Le terme ERP provient du nom de la méthode MRP (Manufacturing Resource Planning) utilisée depuis les années 70 pour la gestion et la planification de la production industrielle.

- **SCM** : (Supply Chain Management) ou en français GCL (Gestion de la Chaîne Logistique) désigne un ensemble d'échange entre les partenaires, et leur coordination ; il s'agit d'une approche globale qui couvre tous les aspects logistiques de l'entreprise, depuis la planification des ressources jusqu'à la livraison des produits, en passant par les prévisions, la conception et la fabrication.

Bibliographie

[1] « L'accès transparent aux applications : l'EAI propose une véritable architecture d'intégration », http:// www.grd-publications.com/art/ls043/ls043110.htm.

[2] C. Le duc, N. Le thanh, « Sémantique des modèles d'échange de données », Projet Mecosi, Rapport de recherche, laboratoire Informatique Signaux et Systèmes de Sophia Antipolis, septembre 2002.

[3] R. Altman, « Intégration globale », Gartner Group, février 2001.

[4] «l'intégration des applications par l'intégration des données » REPintegr_onepage_application-data_f.doc, data intégration, 29septembre2001 2002 consyst SQL, INC.

[5] D. Serain, « Enterprise application intégration », 3ème édition, Dunod, Paris 2001.

[6] X. Yang, G. Yu & G. Wang, « Efficient mapping integrity constraints from relational database to XML document », A. Caplinskas and J. Eder édition, ADBIS, LNCS2151, pp 338-351, Springer Verlag Berlin Heidelberg, 2001.

[7] W. H. Inmon, Richard D. Hackathorn, « Building the Data Warehouse », Second Edition, John Wiley & Sons, ISBN N°0471-14161-5, 1996.

[8] K. Alva-jorgensen, « Our enterprise intégration approach », 2002. http://www.accenture .com /xd/xd.asp ?it = enWeb&xd=Services\se\ebs_capa_approach.xml.

[9] J. Chauvet, « Services Web avec SOAP, WSDL, UDDI, ebXML », editions Eyrolles 2002.

[10] S. Van den enden, E. Hoeymissen, G. Neven & P. Verbaeten « A case study in application intégration », Business Object Component Workshop VI : Enterprise Application Intégration, 2000.

[11] M. Adkisson, « A component approach to EAI for a heterogeneous world », Unisys World, 2000. http://www.unisysworld.com/monthly/2000/10/eaifront.shtml.

[12] I. Gorton & al « Workshop on Architecture for Complex Application Intégration », IEEE-CS, 2003.

[13] H. VERJUS, « Conception et construction de fédérations de progiciels » Thèse de Doctorat, ESIA, 2001.

[14] M. Uschold, M. King, S. Moralee & Y. Zorgios « The enterprise ontology », University of Edinburgh,1997.

[15] M. Stonebraker, Integrating islands of information », eAI Journal, Septembre/Octobre, 1999.

[16] J. T. Pollok, « The big issue : interoperability vs intégration », eAI Journal, 2001.

[17] R. Hailstone, « Intégrations strategies : the start of convergence », IDC, 2003.

[18] S. Wong, « Web services : the next evolution of application intégration », Web services architecture, 2002.

[19] « EAI au cœur de l'e-business», Octo technology, Edition Eyrolle, Novembre 2000.

[20] A. Gruden, P. Strannergrad, « BPM : the next wave », eAI Journal, Janvier, 2003.

[21] Dante Consulting, « Web service overview », Dante Consulting, 2003.

[22] D. S. Linthicum, « Enterprise application intégration », Mercator, 2001.

[23] J. Schmidt, « Enabling next generation enterprise », eAI Journal,2000.

[24] E. Stohr, J. V. Nickerson, « Intra enterprise intégration : methods and directions », Addision Wesley, 2001.

[25] Panorama EAI : les architectures en concurrence », Dreamsoft, http://solutions. Journaldunet.com/0202/020211_comparo_eai_archi.shtml.

[26] J. C. Lutz, « EAI : architecture patterns », eAI Journal, Mars 2000.

[27] N. Erasala, D. C. Yen, T. M. Rajkumer, « Enterprise application integration in the electronic commerce world », Elsevier, CSI, 25 (69-82), 2003.

[28] « Spécification du profil UML d'assemblage cible CCM », Projet Rntl accord, Mai 2002.

[29] A. Napoli, « Une introduction aux logiques de descriptions », Rapport de recherche N°3314, INRIA, 1997.

[30] Medjahed, B.; Bouguettaya, A.; Elmagarmid, A. K.: Composing Web Services on the Semantic Web. Very Large Data Base Journal, 12 (4), 2003

[31] Linthicum,D. S.: Next Generation Application Integration, Addison-Wesley edition, 2004.

[32] Jones D., Bench-Capon T., Visser P., Methodologies for Ontology Development, in Proceedings of IT&KNOWS (Information Technology and Knowledge Systems) of the 15th IFIP World Computer Congress, Budapest, Hungary, 1998.

[33] Maedche, A.; Motik, B.; Silva, N.; Volz, R.: MAFRA: A MApping FRAmework for Distributed Ontologies in the Semantic Web. In ECAI'02 Workshop Knowledge Transformation, Lyon, France, 2002.

[34] Gronau, N.; Uslar, M.: Skill Management Catalogues Built via KMDL- Integrating knowledge and Business Process Modelling. ECAI'04 Workshop Knowledge Management and Organizational memory, Valencia, Spain, 2004.

[35] Noy, N. F.; Musen, M.A.: PROMPT-Algorithm and Tool for Automated Ontology Merging and Alignment. Seventeenth National Conference on Artificial Intelligence, AAAI, Austin, TX, 2000.

[36] Gahleitner, E.; Wob, W.: Enabling Distribution and Reuse of Ontology Mapping Information for Semantically Enriched Communication Services. In IEEE Computer Society, of the 15th International Workshop on Database and Expert Systems Applications (DEXA), Zaragoza, Spain, 2004.

[37] Michalowski, M.; Ambite, J.L.; Thakkar, S.; Tuchinda, R.; Knoblock, C.A.; Minton S.: Retrieving and Semantically Integrating Heterogeneous Data from the Web. IEEE Intelligent Systems, 19(3), 2004.

[38] Horrocks, I.; Patel-Schneider, P. F.; Harmelen, F. V.: From SHIQ and RDF to OWL: The making of a web ontology language. In Web Semantics Journal, 1 (1), 2003.

[39] Haarslev, V.; Möller, R.: RACER System Description. In Proc. of the Int. Joint Conf. on Automated Reasoning (IJCAR 2001), volume 2083 of Lecture Notes in Artificial Intelligence, 2001, pp 701–705.

[40] Baader, F., Calvanese, D., McGuinness, D., Nardi, D. and Patel-Schneider, P.F. eds. The Description Logic Handbook: Theory, Implementation and Applications. Cambridge University Press, 2003.

[41] Yang, J.; Papazoglou, M.P.: Service Components for Managing the Life-cycle of Service Compositions. In Information Systems, 29 (2), (2004)

[42] Knublauch, H.; Mugen, M.; Rector, A.: Editing Description Logic Ontologies with Protégé OWL Plugin, In DL'04, Workshop on Description Logic, Canada. 2004.

[43] BPEL4WS: http://xml.coverpages.org/bpel4wx.html.

[44] Lopez, M. F.; Perez, A. G.: Overview and Analysis of Methodologies for Building Ontologies, In Knowledge Engineering Review. 17(2), 2002.

[45] Uschold, M.; Grüninger, M.: Ontologies Principles Methods and Applications, In Knowledge Engineering Review. 11(2), 1996.

[46] Grüninger, M.; Fox, M. S.: Methodology for the Design and Evaluation of Ontologies, In IJCAI'95 Workshop on Basic Ontological Issues in Knowledge Sharing, Montreal, 1995.

[47] Fernandez, M.; Gomez-Perez, A.; Juristo, N.: Methontology from Ontological Art Toward Ontological Engineering, In AAAI'97 Spring Symposium Series on Ontological Engineering, USA, 1997.

[48] Gruber, R. T.: A Translation Approach to Portable Ontology Specification. In Knowledge Acquisition, Vol 5, pp199-220, 1993.

[49] Borgida, A.; Serafini, L.: Distributed Description Logics Assimilating Information from Peer Sources, In Journal of Data Semantics, 2003.

[50] Protégé OWL, version 3.1.1, 2005, http://protege.stanford.edu.

[51] Gruber, R. T.: Towards Principles for the Design of Ontologies Used for Knowledge Sharing. Journal of Human Computer Studies, 1995.

[52] Abels, S.; Haak, L, Hahn, A.: Identification of Common Methods Used for Ontology integration tasks. IHIS'05, November 4, Bremen, Germany, 2005.

[53] Holfreiter, B.; Huemer, C.; Modelling Business Collaborations in Context, In the Proceedings of On the Move to Meaningful Internet Systems OTM Workshops, Springer-Verlag, 2003.

[54] Bauer, B.; Müller, J. P.; Roser S.: A Decentralized Broker Architecture for Collaborative Business Process Modelling and Enactment. 2nd International Conference I-ESA'06, Interoperability for Enterprise Software and Application, Bordeaux, 2006.

[55] Kalinichenko, L.; Missikoff, M.; Schiappelli, F.; Skvortsov, N.: Ontological Modelling. 5th Russian Conference on Digital Libraries, RCD'03, St-Petersburg, Russia, 2003.

[56] Stelzer, D.; Fischer, D.; Nirsberger, I.: A Famework for Assessing Inter-organizational Integration of Business Information Systems. International Journal of Interoperability in Business Information Systems, IBIS, 2(2), 2006, pp. 9-20.

[57] Izza, S.; Vincent, L.; Burlat, P.: Unified Framework for Application Integration, In 7th International Conference on Enterprise Information Systems, USA, 2005.

[58] Martin, H.; Zdravkovic, J.; Johannesson, P.: Service-Based Processes Design for Business and Technology. ICSOC'04, November 15-19, New York, USA, 2004.

[59] Alonso, G.; Casati, F.; Harumi, K.; Machiraju, V.: Web Services-Concepts, Architectures and Applications, Berlin, Springer-Verlag edition, 2004.

[60] Acharya, R.: EAI- A Business Perspective, eAI Journal, www.bijonline.com/, 2003.

[61] Zang, S.; Hofer, A.; Adam, O.: Cross Enterprise Business Process Management Architecture–Methods and Tools for Flexible Collaboration, In OTM Workshop, LNCS, vol. 3292, Springer-Verlag, Heidelberg R. Meersman et al. edition, Berlin, 2004.

[62] Panetto, H.; Scannapieco, M.; Zelm, M.: INTEROP NoE- Interoperability Research for Networked Enterprise Application and Software. OTM Workshop, LNCS, Vol 3292, Heidelberg, R.; Meersman et al edition, Berlin, 2004, pp. 866-882.

[63] Bianchini, D.; De Antonellis, V.; Melchiori, M.: Domain Ontologies for knowledge Sharing and Service Composition in Virtual Districts, In the Proceedings of the 14th International Workshop on Database and Expert Systems Applications (DEXA), 2003.

[64] Mertz, D.: Understanding ebXML, Understanding the Business Web of the Future Phenomenological Unifier, Gnosis Software, Inc, 2001.

[65] Dogac, A.: Semantic Web Services, In the Proceedings of International Conference on Web Engineering, Munich, 2004.

[66] Dogac, A.; Laleci, G.; Kirbas, S.; Kabak, Y.; Sinir, S.; Yildiz, A.; Gurcan, Y.: Artemis-Deploying Semantically Enriched Web Services in the Healthcare Domain , http://www.ing.uinbs.it/~deantone/interdata_tema3/Artemis/artemis.html , 2004.

[67] Martin, D.; Burstein, M.; Lassila, O.; Paolucci, M.; Payne, T.; Mcllraith, S.: Describing Web Services Using OWL-S and WSDL, http:// www.daml.org/services/owl-s/1.0/, 2003.

[68] Roman, D.; Lausen, H.; Keller, U.; Oren, E.; Bussler, C.; Kifer, M.; Fensel, D.: Web Service Modeling Ontology, WSMO working draft, http://www.wsmo.org/2004/d2/v1.0/20040920/, 2004.

[69] Jain, A. K.; Dubes, R. C.: Algorithms for Clustering Data, Prentice-Halle, 1988.

[70] OMG: OMG Unified Modeling Language, Version 2.0. 2003. http://www.omg.org/technology/documents/modeling spec catalog.htm#UML.

[71] Horrocks, I. and Sattler, U. Ontology Reasoning in the SHOQ (D) Description Logic. In Nebel, B. ed. Proc. of the 17th Int. Joint Conf. on Artificial Intelligence (IJCAI 2001), Morgan Kaufmann, 2001, pp 199–204.

[72] Williamson, O. E.; Winter, S.G.; Coase, R. H.: The Nature of the Firm: Origins, Evolution and Development. Oxford University Press, New York, 1991.

[73] Driouche R, Touati K, Saada-Khelkhal F, Abdemouche H., 2005 " Vers une Extension du Scénario de Collaboration EbXML par les Agents Mobiles et les Web Services" Conférence Internationale sur la Productique, IEEE. Tlemcen.

[74] Driouche, R., Boufaida, Z., Kordon, F., 2006 "An Ontology Based Architecture for Integrating Enterprise Applications", MSVVEIS'06 Workshop, INSTICC Press, Cyprus, Paphos.

[75] Driouche, R., Boufaida, Z., Kordon, F., 2006 "A Multi-Views Business Process Ontology for Flexible Collaboration ", EI2N'06 Workshop, Hermes Edition, Bordeaux, pp 51-62.

[76] Driouche, R Boufaida, Z. Kordon, F. 2006 "Towards Integrating Collaborative Business Process based on Process Ontology and EbXML Collaboration Scenario", In: 6th International Workshop on Web Based Collaboration'06, IEEE Computer Society, Krakow, Poland. 4 - 6 September. Pp 299-303.

[77] Driouche, R Boufaida, Z. Kordon, F. 2006 " An Enterprise application integration architecture supporting ontology based approach for B2B collaboratio" International Journal of Interoperability in Business Information Systems, IBIS, 2007.

[78] Guergour, H, Driouche, R Boufaida,. 2006 "An Approach for Application Ontology Building and Integration Enactment" , Workshop SWAP, Italy.

[79] Http://www. dictionary.com.

[80] www.feb.be.

[81] http:// www.Dicodunet.com.

[82] http://www.wikipedia.org/wiki/Commerce_electronique.

[83] http://www.wikipedia.org/wiki/B2C.

[84] E. Brousseau The governance of transaction by commercial intermediaries.

[85] Chan LE DUC, Nhan LE THANH ; projet MECOSI Sémantique des modèles d'échange de données Rapport de recherche I3S/RR-20026336fr ; Septembre 2002 http://www.is3.unice.fr/I3S/FR.

[86] Luc Rochat HEGNE, EbXML Présentation d l'initiative Décembre 2003.

[87] Bordage, F. " Relier les Applications Hétérogènes", 2003.

[88] Marc LANGLOIS Les apports d'ebXML, avantages et perspectives.

[89] L'EDI : langages et mises en œuvre CANOPE – tenor.

[90] EbXML Technical Architecture Specification v 1.04. Edition : 16 février 2001.

[91] M. Minsky, "A framework for Representing Knowledge", in WINSTON, P.H., The Psychology of Computer Vision, New York, McGraw-Hill, pp 211-277, 1975.

[92] S. Lauras, " Urbanisation et cartographie du système d'information", http://www.mega.con/fr/consulting/Enterprise-architecture/it-planning/index.asp.

[93] The SMB Internet Scenario, Gartner Research 2001.

[94] F. Rivard, "EAI: de l'intégration à l'e-business", Cosmosbay, 2000.

[95] IDEF5 Ontology Description Capture Method Overview, KBSI Report, 1994, http://www.idef.com/idef5.html.

[96] Michael K. Smith, Chris Welty, and Deborah L. McGuinness, Editors, OWL Web Ontology Language Guide,W3C Recommendation, 10 February 2004. http://www.w3.org/TR/owl-guide/

[97] J. Broekstra, M. Klein, S. Decker, D. Fensel, F. van Harmelen, and I. Horrocks. Enabling Knowledge Representation on the Web by extending RDF Schema. In Proceedings of the tenth World Wide Web conference WWWW'10, Hong Kong, May 2001.

[98] T. Bray, J. Paoli, C.M. Sperberg-McQueen and E. Maler, Editors. Extensible Markup Language (XML) 1.0, Second Edition, World Wide Web Consortium. 6 October

[99] Franz Baader, Diego Calvanese, Deborah McGuinness, Daniele Nardi, Peter Patel-Schneider, editors. The Description Logic Handbook. Cambridge University Press, 2003;

[100] R. J. Brachman and J. G. Schmolze. An overview of the KL-ONE knowledge Representation System. Cognitive Science, 9(2):171–216, April 1985.

[101] Deborah L. McGuinness and Frank van Harmelen, Editors. OWL Web Ontology Language Overview, W3C Recommendation, 10 February 2004. http://www.w3.org/TR/owl-features/.

[102] Dan Brickley and R. V. Guha, Editors. RDF Vocabulary Description Language 1.0: RDF Schema, W3C Recommendation, 10 February 2004.http://www.w3.org/TR/rdf-schema/

[103] Dan Connolly, Frank van Harmelen, Ian Horrocks, Deborah L. McGuinness, Peter F. Patel-Schneider, and Lynn Andrea Stein. DAML+OIL Reference Description. W3C Note 18 December 2001. http://www.w3.org/TR/daml+oil-reference.

[104] D. Fensel, S. Decker, M. Erdmann, and R. Studer. Ontobroker: The very high idea. In Proceedings of the 11th International Flairs Conference (FLAIRS-98), Sanibal Island, Florida, May 1998.

[105] I. Horrocks, D. Fensel, J. Broekstra, S. Decker, M. Erdmann, Goble, F. van Harmelen, M. Klein, S. Staab, R. Studer, and E. Motta. OIL: The Ontology Inference Layer. Technical Report IR-479, Vrije Universiteit Amsterdam, Faculty of Sciences, Sept. 2000. http://www.ontoknowledge.org/oil/.

[106] S. Cranefield and M. Purvis. UML as an ontology modelling language. In Proceedings of the Workshop on Intelligent Information Integration, 16th International Joint Conference on Artificial Intelligence (IJCAI -99), 1999. http://sunsite.informatik.rwth-aachen.de/Publications/CEUR-WS/Vol-23/cranefield-ijcai99- iii.pdf.

[107] G. Booch. Object-Oriented Analysis and Design with Applications. Addison-Wesley, 2nd edition, 1994.

[108] D'Souza, D., Wills, A.: Objects, Components and Frameworks with UML: The Catalysis Approach. Addison-Wesley, 1998. http://www.trireme.com/catalysis

[109] Cook, S., Daniels, J.: Designing Object Systems: Object-Oriented Modelling with Syntropy. Prentice Hall, UK (1994).

[110] Richard F. Paige and Jonathan S. Ostroff : A Comparison of the Business Object Notation and the Unified Modeling Language. Beyond the Standard. Second International Conference, Fort Collins, CO, USA, October 28-30. 1999.

[111] Klass Objecten for quality in Object and Component Technology http://www.klasse.nl/ocl/ocl-introduction.html

[112] H. Wache, T. Vgele, U. Visser, H. Stuckenschmidt, G. Schuster, H. Neumann, S. Hbner: “Ontology-based integration of information A survey of existing approaches”. In Proceedings of IJCAI-01 Workshop: Ontologies and Information Sharing. 2001.

[113] Yigal Arens, Chun-Nan Hsu, and Craig A. Knoblock. Query processing in the SIMS information mediator. In Advanced Planning Technology. AAAI Press, California, USA, 1996.

[114] E. Mena, V. Kashyap, A. Sheth, and A. Illarramendi. Observer: An approach for query processing in global information systems based on interoperability between preexisting ontologies. In Proceedings 1st IFCIS International Conference on Cooperative Information Systems (CoopIS ’96). Brussels, 1996.

[115] Cheng Hian Goh. Representing and Reasoning about Semantic Conflicts in Heterogeneous Information Sources. Phd, MIT, 1997.

[116] H. Wache, Th. Scholz, H. Stieghahn, and B. K¨onig-Ries. An integration method for the specification of rule–oriented mediators. In Yahiko Kambayashi and Hiroki Takakura, editors, Proceedings of the International Symposium on Database Applications in Non- Traditional Environments (DANTE’99), pages 109–112, Kyoto, Japan, November, 28-30 1999.

[117] Heiner Stuckenschmidt, Holger Wache, Thomas V¨ogele, and Ubbo Visser. Enabling technologies for interoperability. In Ubbo Visser and Hardy Pundt, editors, Workshop on the 14th International Symposium of Computer Science for Environmental Protection, pages 35–46, Bonn, Germany, 2000. TZI, University of Bremen.

[118] François Goasdoué, Véronique Lattes, and Marie-Christine Rousset. The use of Carin language and algorithms for information integration: The PICSEL project. International Journal of Cooperative Information Systems (IJCIS), 9(4):383 – 401, 1999.

[119] Diego Calvanese, Giuseppe De Giacomo, and Maurizio Lenzerini. Description logics for information integration. In Computational Logic: From Logic Programming into the Future (In honour of Bob Kowalski), Lecture Notes in Computer Science. Springer- Verlag, 2001.

[120] A.D. Preece, K.-J. Hui, W.A. Gray, P. Marti, T.J.M. Bench-Capon, D.M. Jones, and Z. Cui. The kraft architecture for knowledge fusion and transformation. In Proceedings of the 19th SGES International Conference on Knowledge-Based Systems and Applied Artificial Intelligence (ES’99). Springer, 1999.

[121] S. Chawathe, H. Garcia-Molina, J. Hammer, K. Ireland, Y. Papakonstantinou, J. Ullman, and J. Widom. The TSIMMIS project: Integration of heterogeneous information sources. In Conference of the Information Processing Society Japan, pages 7–18, 1994.

[122] Heiner Stuckenschmidt and Holger Wache. Context modelling and transformation for semantic interoperability. In Knowledge Representation Meets Data Bases, 2000.

[123] Dieter Fensel, Stefan Decker, M. Erdmann, and Rudi Studer. Ontobroker: The very high idea. In 11. International Flairs Conference (FLAIRS-98), Sanibal Island, USA, 1998.

[124] Jeff Heflin and James Hendler. Semantic interoperability on the web. In Extreme Markup Languages 2000, 2000.

[125] Arpirez J., Gómez-Perez A., Lozano A. et Pinto S. (1998). (ONTO)2Agent: An ontology-based WWW broker to select ontologies. Paper presented at the Workshop on Applications of Ontologies and PSMs, Brighton, England.

[126] Borgo S., Guarino N. et Masolo C. (1996). Stratified Ontologies: the case of physical objects. Paper presented at the ECAI96. Workshop on Ontological Engineering, Budapest.

[127] Blazquez M., Fernandez M., Garcia-Pinar J. M. et Gomez-Perez A. (1998). Building Ontologies at the Knowledge Level using the Ontology Design Environment. Paper presented at the Proc. of the 11th KAW, Banff, Canada.

[128] Gómez-Pérez A. (2000). Ontological Ingineering: A state of the art. Madrid: Facultad de Informatica, Universidad Politecnica de Madrid.

[129] Maedche A. (2002). Ontology Learning for the Semantic Web. Boston: Kluwer Academic Publishers.

[130] Maedche A., Motik B., Stojanovic L., Studer R. et Volz R. (2003). Ontologies for Enterprise Knowledge Management. IEEE Intelligent Systems, Volume 18(2), 26-33.

[131] Domingue J. (1998). Tadzebao and WebOnto: Discussing, Browsing, and Editing Ontologies on the Web. Paper presented at the 11th Workshop on Knowledge Acquisition, Modeling and Management, KAW'98, Banff, Canada.

[132] Duineveld A. J., Stoter R., Weiden M. R., Kenepa B. et Benjamins V. R. (1999). WonderTools? A comparative study of ontological engineering tools.

[133] Mizoguchi R., Kozaki K., Sano T. et Kitamura Y. (2000). Construction and Deployment of a Plant Ontology. The 12th International Conference, EKAW2000, (Lecture Notes in Artificial Intelligence 1937), 113-128.

[134] Musen M. A., Tu S. W., Eriksson H., Gennari J. H. et Puerta A. R. (1993). PROTEGE-II: An Environment for Reusable Problem-Solving Methods and Domain Ontologies. Paper presented at the International Joint Conference on Artificial Intelligence, Chambery, Savoie, France.

[135] Bernaras A., Laresgoiti I. et Corera J. (1996). Building and Reusing Ontologies for Electrical Network Applications. Paper presented at the Procedings of the 12th ECAI96.

[136] Gómez-Pérez A. (1999). Ontological Engineering: A state of the art. Expert Update, 2(3), 33-43.

[137] Noy N. F., Fergerson R. W. et Musen M. A. (2000). The knowledge model of Protege-2000: Combining interoperability and flexibility. Paper presented at the 2th International Conference on Knowledge Engineering and Knowledge Management (EKAW'2000), Juan-les-Pins, France.

[138] Psyché V. (2003). État de l'art sur l'ontologie - application au téléapprentissage. Rapport technique, Montréal: Télé-université, Centre de recherche LICEF.

[139] Bechhofer, S., Horrocks, I., Goble, C. et Stevens, R. (2001). OilEd: a Reason-able Ontology Editor for the Semantic Web. Paper presented at the Proceedings of KI2001, Joint German/Austrian conference on Artificial Intelligence,, Vienna. Springer-Verlag.

[140] Farqhuar A., Fikes R. et Rice J. (1996). The Ontolingua Server: Tool for Collaborative Ontology Construction. Paper presented at the Proc. of the 10th Knowledge Acquisition for Knowledge-Based Systems Workshop, Alberta, Canada.

[141] Swartout B., Patil R., Knight K. et Russ T. (1997). Towards Distributed Use of Large-Scale Ontologies. Spring Symposium Series on Ontological Engineering, pp.138-148.

[142] Pinto, H. S., Gomez-Perez, A., and Martins, J. P. (1999). Some issues on ontology integration. In Proceedings of the Workshop on Ontologies and Problem Solving Methods during IJCAI-99, Stockholm, Sweden.

[143] B. Meyer. Eiffel: the language. Prentice-Hall, 1992.

[144] Robert M. MacGregor. Using a description classifier to enhance deductive inference. In ProceedingsSeventh IEEE Conference on AI Applications, pages 141–147, 1991.

[145] A. Borgida, Brachman a R. J., D. L. McGuiness, and L. A. Resnick. Classic: A structural data model for objects. In ACM SIGMOID International Conference on Management of Data, Portland, Oregon, USA, 1989.

[146] M.R. Quillian. Semantic Memory. In Semantic Information Processing, pages 227–270. MIT Press, 1968.

[147] J. Sowa. Conceptual Structures: Information Processing in Mind and Machine. Addison-Wesley, 1984.

[148] M. Kifer, G. Lausen Et J. Wu. Logical Foundations of Object-Oriented and Frame-Based Languages. Journal of the ACM, 42(4):741–843, 1995.

[149] M. L. GinsberG. Knowledge Interchange Format: the KIF of death. AI Magazine, 12(3):57–63, 1991.

[150] A. Gomez-Perez, M. Fernandez-Lopez Et O. Corcho. Ontological Engineering. Springer-Verlag, Advanced Information and Knowledge Processing, 2003.

[151] V.K. Chaudhri, A. Farquhar, R. Fikes, P.D. Karp Et J.P. Rice. OKBC: a Programmatic Foundation for Knowledge Base Interoperability. In Proceedings of the 15th National Conference on Artificial Intelligence (AAAI'98), pages 600–607. MIT Press, 1998.

[152] D. Fensel, I. Horrocks, F. Van Harmelen, S Decker, M. Erdmann Et M. Klein. Oil in a nutshell. In Proceedings of European Knowledge Acquisition Workshop (EKAW'2000), volume 1937, pages 1–16. Springer-Verlag LNAI, 2000.

[153] J. Hendler Et D.L. Mcguinness. The Darpa Agent Markup Language. http://www.daml.org, 2001.

[154] Fernàndez M., Gòmez-Perez A. and Juristo N. Methontology: From Ontological Art Toward Ontological Engineering. Paper presented at the Spring Symposium Series on Ontological Engineering. AAAI, Stanford, USA, 1997.

[155] Gòmez-Pérez A., Knowledge Sharing and ReuseThe Handbook of Applied Expert Systems. Edited by J. Liebowitz, CRC Press 1998:

[156] G. Wiederhold, (1994a), "An Algebra for Ontology Composition", in Proceedings of Proceedings of the 1994 Monterey Workshop on Formal Methods, Monterey, CA, USA, September, pp. 56-61.

[157] P. Mitra, G. Wiederhold, and J. Jannink,(1999a),"Semi-automatic integration of Knowledge sources",in Proceedings of Proceedings of the Fusion'99 Conference, Sunnyvale, CA, USA, July.

[158] H. Chalupsky, (2000), "OntoMorph: A Translation System for Symbolic Knowledge", in Proceedings of Proceedings of the 7th International Conference on Principles of Knowledge Representation and Reasoning (KR2000), Breckenridge, Colorado, USA, April, 12-15, pp. 471-482.

[159] UML Profile for Event-based Architecture in Entreprise Application Integration, http ://cgi.omg.org/docs/ad/00-03-07.pdf.

[160] UML Profile for Enterprise Distributed Object Computing, http ://cgi.omg.org/docs/ad/99-03-10.pdf.

[161] OMG Model-Driven Architecture Home Page, http ://www.omg.org/mda.

[162] Object Managment Group Home Page, http ://www.omg.org.

[163] UML Profile for Scheduling, Performance and Time, http ://cgi.omg.org/docs/ad/99-03-13.pdf.

[164] Joaquin Miller and Jishnu Mukerji. Model Driven Architecture (MDA) http ://cgi.omg.org/docs/ormsc/01-07-01.pdf, July 2001.

[165] « Ontologie informatique » de l'encyclopédie Wikipedia http://fr.wikipedia.org/wiki

[166] Natalya F. Noy et Deborah L. McGuinness, « Développement d'une ontologie 101 : Guide pour la création de votre première ontologie », Université de Stanford, Stanford, Traduit de l'anglais par Anila Angjeli. http://www.bnf.fr/pages/infopro/normes/pdf/no-DevOnto.pdf

[167] Programmez (Thévenon D., Gueguen, E., etc.), Réussir votre intégration, Revue Programmez, 2003.

[168] Rivard F., Patoureaux Y., Bernadac J-C., Knab F., BIP - Le projet de portail d'information de l'entreprise, Cosmosbay, 2001.

[169] Stohr E., Nickerson J. V., Intra Enterpise Intégration: Methods and direction, Addison Wesley, 2001.

[170] C. Torniai, "The Jena Ontology API," https://mailman.stanford.edu/pipermail/protege owl/2006-December/000833.html. 2006.

[171] R., Driouche, H., Guergour, Z., Boufaida "An Approach Based Ontology for Semantically Enriched Communication Scenario", The Eighth International Symposium On Programming and Systems, 2007.

[172] What is ontology? http://mged.sourceforge.net/ontologies/index.php.

[173] F. Fust, " L'ingénierie ontologique", Rapport de recherche N°02-07, 2002.

[174] D. Berardi, D. Calvanese, G. De Giacomo, M. Lenzerini, M. Mecella: ESC: A Tool for Automatic Composition of e-Services based on Logics of Programs, VLDB-TES, 2004

Annexe

Les compagnies partenaires (agence de voyage, compagnies aériennes, hôtels, compagnies de location de voiture) disposent chacune de son agent. C'est un agent principal. Ce dernier contient une interface (sert à communiquer les documents WSDL à l'agent), un module de recherche (pour la création et la coordination avec l'agent de recherche) et un module de négociation (pour la création et la coordination avec l'agent de négociation). Nous donnons ici, les squelettes en Java des différents types d'agents.

1- **Squelette de l'agent principal** :

```
package Princproject;
import jade.core.Agent;
import jade.core.behaviours.Behaviour;
import java.io.File;
import jade.wrapper.ContainerController;
public class AgentPrincipal extends Agent {
   String[] addresses = new String[]{};
  public void setup(){
    this.addBehaviour(new Recherche());
    this.addBehaviour(new InitNegociation ());
    this.addBehaviour(new GestNegociation ());
      }
  public void takeDown(){
  }
}
class Recherche extends Behaviour{
  public void action(){
  }
  public boolean done(){
    return true;
  }
}
class InitNegociation extends Behaviour{
  public void action(){
  }
  public boolean done(){
    return true;
  }
}
class GestNegociation extends Behaviour{
  public void action(){
  }
  public boolean done(){
    return true;
  }
}
```

2- **Squelette de l'agent de recherche** :

```
package rechproject;
import jade.core.Agent;
import jade.core.behaviours.Behaviour;
public class AgentRecherche extends Agent {
   String[] addresses = new String[]{};
  public void setup(){
    this.addBehaviour(new Recherche());
    this.addBehaviour(new Reponse());
    this.addBehaviour(new Delai());
      }
  public void takeDown(){
  }
}
class Recherche extends Behaviour{
  public void action(){
  }
  public boolean done(){
    return true;
  }
}
class Reponse extends Behaviour{/*envoie une réponse à l'agent principal*/
  public void action(){
  }
  public boolean done(){
    return true;
  }
}
class Delai extends Behaviour{/*initialise un compteur pour la destruction de l'agent*/
  public void action(){
  }
  public boolean done(){
    return true;
  }
}
```

3- **Squelette de l'agent de négociation** :

```
package negproject;
import jade.core.Agent;
import jade.core.behaviours.Behaviour;
public class AgentNegociation extends Agent {
   String[] addresses = new String[]{};
  public void setup(){
    this.addBehaviour(new Negociation());
```

```
    this.addBehaviour(new Proposition());
    this.addBehaviour(new Refus());
    this.addBehaviour(new Consultation());
  }
  public void takeDown(){
  }
}
class Negociation extends Behaviour{
  public void action(){
  }
  public boolean done(){
    return true;
  }
}
class Proposition extends Behaviour{
  public void action(){
  }
  public boolean done(){
    return true;
  }
}
class Refus extends Behaviour{
  public void action(){
  }
  public boolean done(){
    return true;
  }
}
class Consultation extends Behaviour{ /*consulter l'agent principal s'il ya accord ou non*/
  public void action(){
  }
  public boolean done(){
    return true;
  }
}
```