NIVERSITE KASDI MERBAH OUARGLA

aculté des Sciences et de la Technologie et Sciences de la Matière

Département de mathématiques & d'informatique

N° d'ordre :
N° de série :

Mémoire

Présenté pour L'obtention du diplôme de

MAGISTER

Spécialité : Informatique
Option : Informatique & Communication Electronique (ICE)

Par : Chafik Berdjouh

Thème

Une approche basée agent
pour la découverte de services Web

Soutenu publiquement le : 25/05/2009

Devant le jury composé de :

• **M. BENMOHAMED Mohamed**	**Professeur à l'université de Constantine**	**Président**
• **M. BELLATAR Brahim**	**Maître de conférences à l'université de Batna**	**Examinateur**
• **M. ZIDANI A/Madjid**	**Maître de conférences à l'université de Batna**	**Examinateur**
• **M. KAZAR Okba**	**Maître de conférences à l'université de Biskra**	**Rapporteur**

ABSTRACT

Web services are emerging and promising technologies for the development, deployment and integration of Internet applications. They are based on three main bricks that are SOAP (Simple Object Access Protocol), WSDL (Web Service Description Language) and UDDI (Universal Description, Discovery and Integration). The language used behind these protocols is XML (eXtensible Markup Language), which makes Web services independent of platforms and programming languages. They have become very effective in the interoperability of systems. The need to introduce semantics in Web services is felt to automate the different phases of their life cycle, namely the discovery phase.

The concept of semantic web services, is the result of convergence in the field of Web services with the Semantic Web, in fact its ultimate goal is to make web services more accessible to the machine by automating tasks that facilitate their use. In this work, we study the problem of semantic discovery of services by providing a method that is based on agents.

Keywords : Web service, multi-agents system, semantic Web.

RÉSUMÉ

Les services Web sont des technologies émergentes et prometteuses pour le développement, le déploiement et l'intégration d'applications Internet. Ils sont basés sur trois briques principales que sont SOAP (Simple Object Access Protocol), WSDL (Web Service Description Langage) et UDDI (Universal Description, Discovery and Integration). Le langage utilisé qui sous-tend ces protocoles est XML (eXtensible Markup Language), ce qui rend les Web Services indépendants des plates-formes et des langages de programmation. Ils sont devenus un moyen très efficace dans l'interopérabilité des systèmes. Le besoin d'introduire la sémantique dans les services Web se fait sentir, afin d'automatiser les différentes phases de leur cycle de vie, en l'occurrence la phase de découverte.

Le concept des services Web sémantiques, est le fruit de la convergence du domaine des services web avec le Web sémantique, en effet son ultime objectif est de rendre les services web plus accessibles à la machine en automatisant les différentes tâches qui facilitent leur utilisation. Dans ce travail, on étudie la problématique de découverte sémantique des services en exposant un état de l'art sur les approches réalisées, et en proposant une méthode qu'est basée sur les agents.

Mots-Clés : service Web, systèmes multi-agents, Web sémantique.

ملخـــص

تعتبر خدمات الواب من التكنولوجيات الناشئة و الواعدة من أجل تطوير ونشر ودمج تطبيقات الانترنات، فهي ترتكز على ثلاث بنيات أساسية : SOAP ، WSDL و UDDI، و لغة XML تعتبر سند هاته البروتوكولات، مما يجعل خدمات الواب مستقلة عن كل البيئات و لغات البرمجة، فأصبحت بذلك وسيلة أكثر فعالية في تبادل الأنظمة. و الحاجة إلى إدخال المفهوم الدلالي في مختلف أطوار دورة حياتها – خاصة في عملية الإكتشاف– قد إزداد.

فمفهوم خدمات الواب الدلالية هو ثمرة التقارب بين مجالي خدمات الواب و الواب الدلالي، الهدف من ذلك جعل خدمات الواب أكثر بلوغا من طرف الآلة بتآلية مختلف أنشطتها تسهيلا لإستعمالها.

في هذا العمل، نحاول دراسة إشكالية الإكتشاف الدلالي للخدمات و ذلك بإستعراض مختلف المناهج المنجزة مع إقتراح طريقة تعتمد على مفهوم الوكلاء.

كلمات مفاتيح : خدمة واب – نظام متعدد الوكلاء – واب دلالي.

Remerciements

Je souhaite, avant toute chose, remercier Dieu pour m'avoir soutenu et permis la réalisation de ce mémoire.

Je tiens à exprimer ma plus grande gratitude au Dr. Kazar Okba, rapporteur, pour leur précieux conseils et recommandations.

Je remercie les membres du jury qui m'on fait l'honneur de juger ce travail :

-Monsieur BENMOHAMED Mohamed, Professeur à l'université de Constantine, Président de mon jury,

-Messieurs BELLATAR Brahim et ZIDANI A/Madjid, Maîtres de conférences à l'université de Batna, en qualité d'examinateurs.

ﷺ

Je remercie mon épouse, pour son soutien dans les moments difficiles.

Je remercie mes parents, pour leur soutien quant à la réalisation de ce mémoire.

Merci à Zakaria Maamar, Professeur à l'université de cheikh Zayed Dubai Emirates, pour son aide si précieuse.

Je remercie aussi tous les enseignants qui ont attribué à ma formation durant l'année théorique et pendant la réalisation de ce travail.

Je remercie mes amis pour leur soutien et leurs encouragements.

Table des matières

➢ Chapitre IV : L'Architecture à base d'agents de découverte de services Web

➢ Chapitre V : Implémentation

Liste des figures

Abréviations

Abbreviation	Nom complet
AASDU	Agent Approach for Service Discovery and Utilization
ACL	Agent Communication Language
AI	Artificial Intelligence
API	Application Program Interface
Axis	Apache eXtensible Interaction System
B2B	business to business
BPEL4WS	Business Process Execution Language for Web Services
DAML	DARPA Agent Markup Language
DARPA	Defense Advanced Research Projects Agency
DCOM	Distributed Component Object Model
DHT	Distributed Hashtables
ebXML	Electronic Business using eXtensible Markup Language
FIPA	Foundation for Intelligent Physical Agents
FTP	File Transfer Protocol
GUI	Graphical User Interface
HTML	Hyper Text Markup Language
IA	Intelligence Artificielle
IAD	Intelligence Artificielle Distribuée
IDL	Interface Definition Language
IR	Information Retrieval
JADE	Java Agent DEvelopement framework
JAXR	Java API for XML registries
JESS	Java Expert System Shell
JSP	JavaServer Pages
jUDDI	Java implementation of the UDDI
KQML	Knowledge Query and Manipulation Language
LSI	Latent Semantic Indexing
NASSL	Network Accessible Service Specification Language
OASIS	Organization for the Advancement of Structured Information Standards
OWL	Web Ontology Language
OWL-S	OWL-based Web service ontology

P2P	Peer to Peer
PSWSD	P2P-based Semantic Web Service Discovery
QoS	Quality of Service
RACER	Renamed A-box and Concept Expression Raisoner
RDF	Resource Description Framework
RDF M&S	RDF Model and Syntax
RDFS	Resource Description Framework Schema
RMI	Remote Method Invocation
SBC	Systèmes à Bases de Connaissances
SDL	Service Definition Language
SMA	Système multi-agents
SOA	Service Oriented Architecture
SOAP	Simple Object Access Protocol
SW	Service Web
TFIDF	Term Frequency Inverse Document Frequency
UDDI	Universal Description, Discovery and Integration
URI	Uniform Resource Identifiers
URL	Uniform Resource Locator
UUID	Universal Unique Identifier
W3C	World Wide Web Consortium
WSDL	Web Service Description Language
WSDL-S	WSDL Semantics
WSDP	Web Services Developer Pack
WSFL	Web Services Flow Language
WSMO	Web Service Modeling Ontology
XML	eXtensible Markup Language

INTRODUCTION

De nos jours, le Web n'est plus simplement un énorme entrepôt de texte et d'images, son évolution a fait qu'il est aussi un fournisseur de services. La notion de "service Web" désigne essentiellement une application mise à disposition sur Internet par un fournisseur de services, et accessible par les clients à travers des protocoles Internet standard. Des exemples de services actuellement disponibles concernent les prévisions météorologiques, la réservation de voyages en ligne, les services bancaires ou des fonctions entières d'une entreprise comme la gestion de la chaîne logistique. Par essence, les services Web sont des composants logiciels autonomes et auto-descriptifs et constituent par ce fait un nouveau paradigme pour l'intégration d'applications.

Actuellement, les services Web sont mis en oeuvre au travers de trois technologies standards : WSDL, UDDI et SOAP. Ces technologies facilitent la description, la découverte et la communication entre services. Cependant, cette infrastructure de base ne permet pas encore aux services Web de tenir leur promesse d'une gestion largement automatisée. Cette automatisation est pourtant essentielle pour faire face aux exigences de passage à l'échelle (il faut être capable de traiter un nombre important de services Web) et de la volonté de réduire les coûts de développement et de maintenance des services. Fondamentalement, elle doit s'accommoder d'un moyen pour décrire les services Web d'une manière compréhensible par une machine.

Le Web sémantique [Ber, 2001] est une vision du Web dans laquelle toute information possède une sémantique compréhensible par une machine. Appliqués aux services Web, les principes du Web sémantique doivent permettre de décrire la sémantique de leurs fonctionnalités, et les raisonnements induits constituent par conséquent une proposition d'automatisation des différentes tâches de leur cycle de vie. La combinaison des technologies des services Web et du Web sémantique a mené au concept des services Web sémantiques.

La découverte des services Web représente un axe de recherche émergent. Au début, la découverte est faite au niveau du registre UDDI, elle est basée essentiellement sur la recherche syntaxique des descriptions WSDL des services Web. Mais avec le développement des technologies du Web sémantique, les techniques de découverte sont devenus essentiellement sémantiques. Cette sémantique est apportée grâce aux ontologies une des technologies

importantes du Web sémantique. Ainsi, des agents logiciels peuvent êtres développés afin de raisonner sur ces ontologies rendant la découverte des services Web dynamique et automatique.

Contribution

Dans ce mémoire, nous proposons une approche de découverte des services Web sémantiques en utilisant la technologie agent et les ontologies.

L'utilisation d'un système multi-agents permet d'assurer une grande flexibilité à l'approche.

Plan du mémoire

Le mémoire est composé de cinq chapitres organisés comme suit :

Chapitre 1 :

Il présente la technologie des services Web et les principaux standards qu'elle supporte, parmi lesquels nous citons les protocoles **SOAP**, **WSDL**, **UDDI.**

Chapitre 2 :

Ce chapitre met le point sur la notion de services web sémantiques, en donnant un aperçu sur les principaux problèmes du domaine, il expose ensuite un état de l'art sur la découverte sémantique et montre un ensemble d'approches de découvertes des services Web.

Chapitre 3 :

Ce chapitre présente un état de l'art sur les systèmes multi-agents et les principaux concepts associés.

Chapitre 4 :

Ce chapitre présente l'approche proposée pour découvrir les services Web sémantiques.

Chapitre 5 :

Ce chapitre présente l'implémentation en décrivant les outils utilisés.

Et enfin nous terminons par une conclusion et des perspectives de notre travail.

Chapitre I

Les Services Web

I.1) Introduction

Le Web a remporté un succès phénoménal en permettant des interactions simples entre les êtres humains et les ordinateurs à l'échelle d'Internet. La pile de protocoles HTTP et HTML, utilisée par tous les explorateurs Web d'aujourd'hui, a permis de projeter des interfaces utilisateurs sur une large gamme d'équipements pour un coût modique. Le principal facteur du succès de HTTP et de HTML réside dans leur relative simplicité : les deux protocoles sont orientés texte et peuvent être mis en œuvre sur de nombreux systèmes d'exploitation, via une multitude d'environnement de développement.

Les services Web reprennent la plupart des idées et des principes du Web, et les appliquent à des interactions entre ordinateurs. Comme pour le World Wide Web, les services Web communiquent via un ensemble de protocoles fondamentaux qui partagent une architecture commune. Ils ont été conçus pour être réalisés sur de nombreux systèmes développés et déployés de façon indépendante. Comme pour le World Wide Web, les protocoles des services Web doivent beaucoup à l'héritage d'Internet, fondé sur le texte. Ils sont conçus pour respecter une structure en couches sans être dépendants de façon excessive de la pile des protocoles.

Mais les services Web diffèrent du World Wide Web dans leur portée. HTTP et HTML ont été définis pour permettre une exploration, essentiellement en mode lecture, d'un contenu souvent statique ou pouvant être mis en cache. À l'opposé, les services Web permettent des interactions fortement dynamiques entre programmes. De nombreux types de systèmes répartis peuvent être implémentés dans l'architecture de services Web : par exemple, des systèmes de messages synchrones et asynchrones, des fermes de serveurs de calculs, des systèmes mobiles connectés en réseau. Les très nombreux impératifs existant dans les interactions de programme à programme imposent à la pile des protocoles des services Web d'être conçue pour un usage plus général que les premiers protocoles Web. Cependant, comme pour le World Wide Web, les services Web reposent sur un petit nombre de protocoles spécifiques. Nous aborderons ce sujet plus en détail par la suite.

I.2) Qu'est ce qu'un service web ?

I.2.1) Une définition

Un service Web est un composant logiciel représentant une fonction applicative (ou un service applicatif). Il est apparu vers 2000. Il peut être accessible depuis une autre application (un client, un serveur ou un autre service Web) à travers le réseau Internet en utilisant les protocoles de transports disponibles (i.e. SOAP sur HTTP). Ce service applicatif peut être implémenté comme une application autonome ou comme un ensemble d'applications (liées ensemble par une infrastructure d'intégration).

Le consortium W3C[1] définit un service Web comme étant une application ou un composant logiciel vérifiant les propriétés suivantes [Boo, 2004] :

– il est identifié par une URI (Uniform Ressource Identifier) ;
– ses interfaces et ses liens peuvent être décrits en XML ;
– sa définition peut être découverte par d'autres services Web ;
– il peut interagir directement avec d'autres services Web à travers le langage XML et en utilisant des protocoles Internet.

I.2.2) Discussion

En réalité, il y a plusieurs manières de définir les services Web :

- si l'on s'en remet à une définition non informatique, le terme «services Web» décrit une fonctionnalité commerciale exposée par une entreprise sur Internet afin de fournir un moyen d'utiliser ce service à distance.
- si l'on s'en tient aux aspects opérationnels, les services Web sont des applications modulaires qui peuvent être décrites, publiées, localisées et invoquées dans un réseau. Ainsi, cela permet à des applications de faire appel à des fonctionnalités situées sur d'autres machines.

Si l'on détaille ce qui précède, on peut dire que l'objectif initial d'un service Web est de permettre l'utilisation d'un composant de manière distribuée. Plus clairement, cela consiste à permettre l'utilisation d'une application à distance. En fait, les services Web facilitent l'invocation de certains traitements depuis Internet. Le modèle logiciel proposé contient à la fois un nouveau mode de développement et une nouvelle méthode de fourniture de services.

I.2.3) Une nouvelle définition

Ainsi, après le client/serveur, les services Web sont présentés comme le nouveau modèle des architectures informatiques, une forme d'aboutissement de l'informatique distribuée.

En effet, les services Web constituent une technologie permettant à des **applications distantes de dialoguer entre elles** via Internet, et ceci**, indépendamment des plates-formes et des langages** sur lesquelles elles reposent. Ces derniers s'apparentent donc à des applications capables de collaborer entre elles, de manière transparente pour l'utilisateur.

Pour ce faire, ces **composants logiciels** s'appuient sur un ensemble de **protocoles standardisant** les modes d'invocation mutuels des composants.

Cependant, cette définition ne serait pas complète si l'on n'évoquait pas ses principaux standards **: SOAP, WSDL et UDDI**, des protocoles mis en place par Microsoft, IBM, Sun et bien d'autres…

[1] World Wide Web Consortium. Organisme de normalisation des standards du Web.

Introduisons-les brièvement avant de leur consacrer une véritable section.

- **SOAP** (**S**imple **O**bject **A**ccess **P**rotocol) est un protocole permettant l'invocation de méthodes à distance par l'échange de messages XML.
- **WSDL (W**eb Services Description Language) est une norme dérivée d'XML qui permet la description de l'interface d'utilisation d'un service Web. Ce standard est donc équivalent à la partie publique d'une classe ou au header d'un programme C.
- **UDDI** (**U**niversal **D**escription, **D**iscovery, and **I**ntegration) est un protocole d'annuaire permettant de trouver le service web que l'on recherche mais aussi d'en annoncer la disponibilité (la publication).

I.2.4) Pourquoi utiliser les services Web ?

Les **services Web** ont été conçus pour faciliter les échanges de données, mais également l'accès aux applications au sein des entreprises et entre les entreprises elles-mêmes.

Les services Web vont ainsi permettre de :

- faciliter les échanges de données entre les applications au sein des entreprises (intra-entreprise).
- et entre les entreprises.
- via n'importe quels types de clients et n'importe quels types de plate-formes.

I.2.5) Les caractéristiques des Web services (WS)

Les caractéristiques des services Web sont : [Fig, 2007]

- *Web based* : les services Web sont basés sur les protocoles et les langages du Web, en particulier HTTP et XML (tout comme le Web lui-même s'appuie sur les protocoles d'Internet en particulier TCP/IP : c'est une « couche » supplémentaire).
- *Self-described, self-contained* : le cadre des services Web contient en lui-même toutes les informations nécessaires à l'utilisation des applications, sous la forme de trois fonctions : trouver, décrire et exécuter. Il est donc nécessaire pour faire fonctionner un cadre de services Web de disposer d'un annuaire des applications disponibles, d'une description du fonctionnement de l'application, et d'avoir accès à l'application elle-même.
- *Modular* : les services Web fonctionnent de manière modulaire et non pas intégrée. Cela signifie qu'au lieu d'intégrer dans une seule application globale toutes les fonctionnalités, on crée (ou on récupère) plusieurs applications spécifiques qu'on fait intéropérer entre elles, et qui remplissent chacune une de ces fonctionnalités.

I.2.6) Domaines d'utilisation

- ***L'intégration d'applications***

L'intégration d'applications n'est pas un concept né avec les services Web. CORBA, DCOM et RMI avaient déjà cet objectif. L'intégration d'applications vise à faire collaborer différentes applications (par exemple : facturation, gestion clients, gestion fournisseurs) dans le but d'atteindre les objectifs métiers de l'entreprise. Le but recherché est la réduction des coûts de transactions en automatisant la majorité des tâches. On est dans le domaine interne de l'entreprise.

L'intégration d'applications peut aussi s'appliquer dans les échanges externes de l'entreprise. On entre alors dans le domaine des échanges inter-entreprises ou B2B (*Business to Business*). Il a pour objectif de favoriser les échanges d'informations entre entreprises partenaires. Le B2B regroupe les mécanismes de communication et la sémantique des informations échangées (comment modéliser un bon de commande, une facture, etc.)

XML et les services Web trouvent leur place dans le cadre de projets d'intégration d'applications et de B2B, comme brique de base pour la communication. De plus, Microsoft, IBM, BEA Systems Inc, Sun Microsystems et d'autres éditeurs poussent cette technologie en prenant part aux actions de normalisation, et en intégrant les services Web à leurs produits [Cru, 2002].

- ***Les portails***

De part leur caractère modulaire, les services Web peuvent s'intégrer dans un portail Web. En effet, un portail est composé de petites applications indépendantes offrant des fonctionnalités à l'utilisateur : une fenêtre donnant la météo, une autre permettant de traduire du texte, etc.

Nous entrevoyons que le marché des services Web s'annonce très important. Les sommes en jeu se chiffrent en milliards de dollars. C'est pour cette raison que les acteurs ne manquent pas [Dan, 2003].

La seule question qui reste ouverte est la viabilité du concept de service Web. Mais, la réponse est cependant toute fournie, car même si les entreprises ne sont pas convaincues, elles n'auront pas le choix car toutes les offres du marché prennent en compte cette technologie.

I.3) Fonctionnement

I.3.1) Principe de base

Jusqu'ici, l'accès via Internet à une ressource applicative ou à une base de données s'effectuait par l'envoi d'une requête s'appuyant sur des langages de script (PHP, JSP, …).

Il s'agissait donc d'un dialogue entre une couche de présentation reposant sur HTML (**protocole HTTP**) et des applications installées sur un serveur distant.

Avec les services Web, un **dialogue** est désormais **instauré entre applications** qui peuvent être installées sur des machines distantes (appareils divers, périphériques, …), et ceci grâce à des **standards XML**. En effet, afin de dialoguer **via Internet**, ces applications doivent « parler » le même langage, langage basé sur le **XML**.

L'architecture de référence des services Web (figure 1) s'articule autour des trois rôles suivants :

- *Le fournisseur de service* : correspond au propriétaire du service.
- *Le client* : correspond au demandeur de service. D'un point de vue technique, il est constitué par l'application qui va rechercher et invoquer un service.
- *L'annuaire des services* : correspond à un registre de descriptions de services offrant des facilités de publication de services à l'intention des fournisseurs ainsi que des facilités de recherche de services à l'intention des clients.

Les interactions de base entre ces trois éléments incluent les opérations de *publication*, de *recherche* et de *lien* (bind). Nous décrivons ci-dessous un scénario type d'utilisation de cette architecture.

1-Le fournisseur de services définit la description de son service et la publie dans un annuaire de service.

2-Le client utilise les facilités de recherche disponibles au niveau de l'annuaire pour retrouver et sélectionner un service donné.

3-Il examine ensuite la description du service sélectionné pour récupérer les informations nécessaires lui permettant de se connecter au fournisseur du service et d'interagir avec l'implémentation du service considéré.

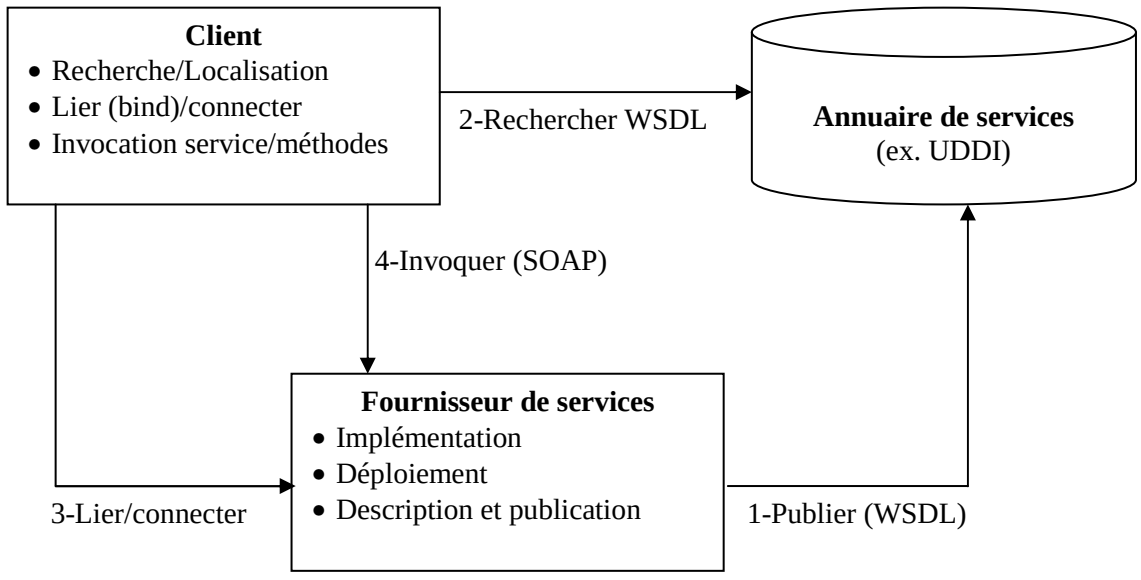

Figure 1 - Architecture des services Web

Les paragraphes qui suivent font le point sur les technologies établies ou en cours de définition dans la sphère des services Web.

I.3.2) Couches de base

Les couches de bases sont au nombre de trois et répondent chacune à un besoin particulier :

•**Découverte** : elle permet l'identification et la localisation les services Web.

•**Description** : elle permet la représentation (modélisation) des fonctions associées aux services Web.

•**Echange** : elle réalise l'échange des messages entre les services Web.

Les **trois principaux protocoles**, liés à chacune de ces couches sont :

• **UDDI :** « **U**niversal **D**escription **D**iscovery and **I**ntegration ».

• **SOAP :** « **S**imple **O**bject **A**ccess **P**rotocol »,

• **WSDL** : « **W**eb **S**ervice **D**escription **L**anguage »

La figure 2 montre la pile des services Web :

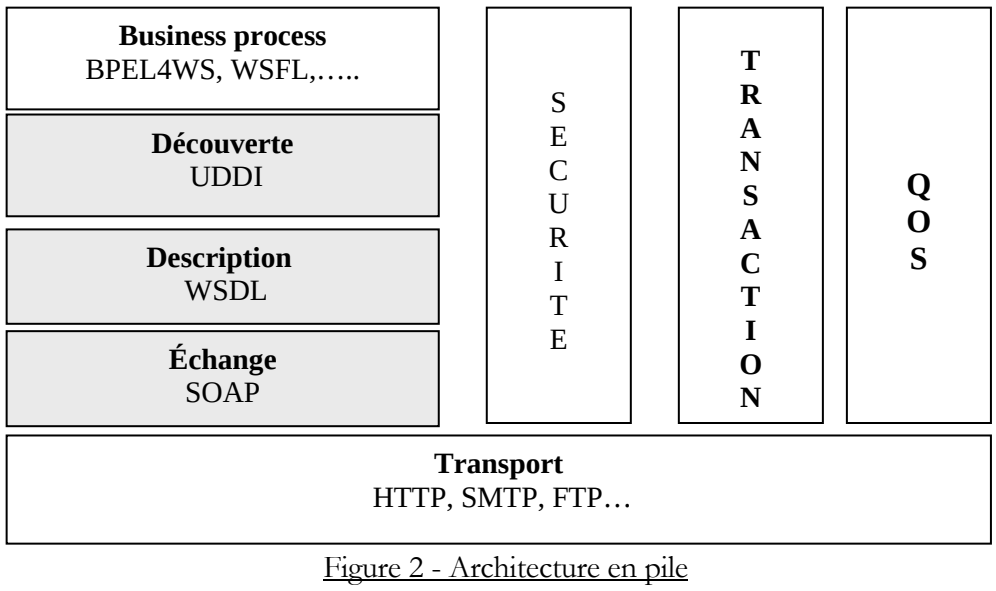

Figure 2 - Architecture en pile

Cette infrastructure définit deux types de couches :

(i) les couches transversales permettant l'échange de messages, la description, la découverte et la composition des services.

(ii) les couches verticales qui peuvent être ajoutées dans toutes les couches horizontales, ils permettent par exemple d'assurer la sécurité d'échange, la gestion des transactions, le maintien d'un niveau de qualité de service…

I.4) Technologies standards

I.4.1) SOAP – Simple Object Access Protocol

SOAP est un protocole de communication entre applications fondé sur XML, visant à satisfaire un double objectif : servir de protocole de communication sur les intranets, dans une optique d'intégration d'applications d'entreprise, et permettre la communication entre applications et services Web, en particulier dans un contexte d'échanges interentreprises [Cha, 2002]. C'est le protocole qui permet d'accès aux web services et d'échanger des messages avec lui.

SOAP est développé conjointement par Microsoft, IBM, Lotus Development (une division d'IBM), DevelopMentor et UserLand Software sous les auspices du W3C. SOAP 1.1 fit l'objet d'une note soumise au W3C en mai 2000, et SOAP 1.2 d'un document de travail (working draft) en juillet 2001.

I.4.1.1) Structure de message SOAP

La structure des messages SOAP se divise en quatre parties [W3C, 2003] (voir figure 3) :

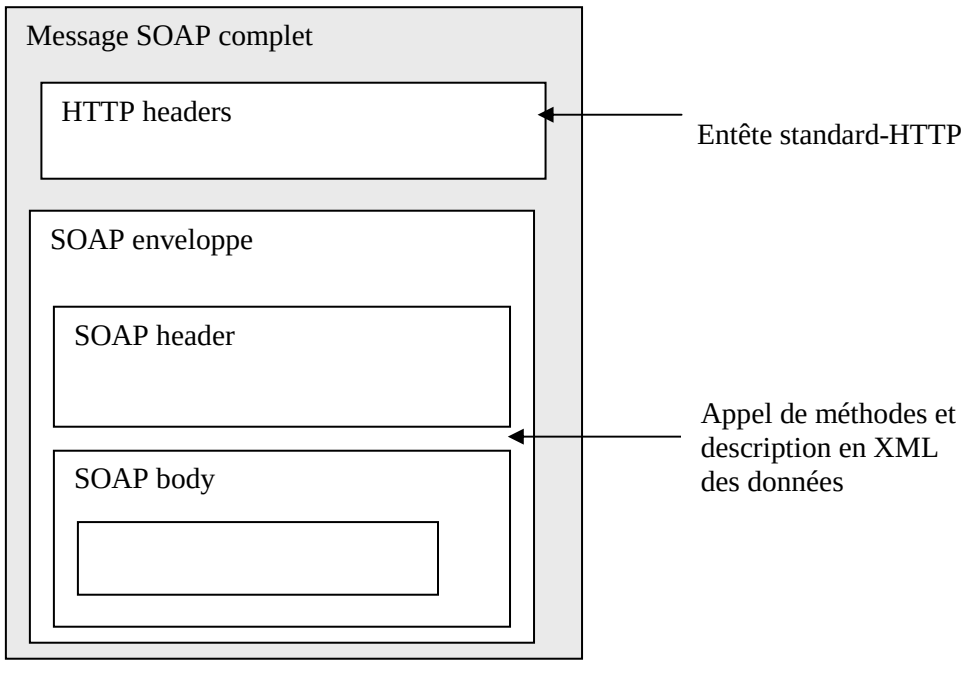

Figure 3 - Structure de l'enveloppe SOAP

1. Le HTTP Header

Le protocole HTTP envoie une requête POST. L'entête HTTP se trouve juste avant le message SOAP, et définit le destinataire du message, les règles d'encodage HTTP, etc.

Le champ SOAPAction peut être utilisé pour indiquer l'intention de la requête SOAP. Cette information peut être utilisée par un firewall pour filtrer les messages. Ce champ est obligatoire mais peut être vide si on n'indique pas l'intention de la requête.

2. L'enveloppe SOAP

L'enveloppe contient l'espace de nommage définissant la version de SOAP utilisée, et les règles de sérialisation, et d'encodage.

3. Le header SOAP

Cette partie du message est optionnelle. Elle sert à transmettre des informations nécessaires pour l'exécution de la requête SOAP aux intermédiaires qui recevront le message. On y précise généralement des informations liées aux transactions, à l'authentification, etc.

Le header est composé d'un ou plusieurs champs : l'attribut actor, désignant le destinataire du header, l'attribut mustUnderstand, qui indique si le processus est optionnel.

Dans l'exemple suivant, on précise des informations sur l'identification de l'utilisateur et la transaction à laquelle appartient le message :

```
<SOAP-ENV :Header
   SOAP-ENV : actor="http://schemas.xmlsoap.org/ soap/actor/next"
   SOAP-ENV:mustUnderstand="1">
   <identifiant numero="124527"/>
   <transaction type="compensees" numero="YU75X"/>
</SOAP-ENV:Header>
```

Figure 4 - Extrait de code Header SOAP

4. Le Body SOAP

Le body SOAP contient toutes les informations que l'on veut transmettre à l'application distante. Le contenu du Body est normalisé dans SOAP RPC, pour modéliser une requête et sa réponse. Le body de la requête contient l'identifiant de l'objet distant, le nom de la méthode à exécuter et les éventuels paramètres. Le body de la réponse contient le résultat de l'exécution de la requête.

I.4.2) WSDL – Web Service Description Language

WSDL (Web Services Description Language) est un standard du W3C qui permet de définir une syntaxe XML pour décrire les méthodes et paramètres des services Web invocables par le biais de messages au format SOAP. Il permet de définir qu'est-ce qu'un service Web est capable de faire, où est-ce qu'il réside et comment l'invoquer.

Le WSDL est aussi l'équivalent de IDL (Interface Definition Language) pour la programmation distribuée (CORBA). Ce langage permet de décrire de façon précise les services Web, en incluant des détails tels que les protocoles, les serveurs, les ports utilisés, les opérations pouvant être effectuées, et les formats des messages d'entrée et de sortie.

Il y a eu d'autres tentatives de langages pour résoudre le problème de la définition des services Web. Microsoft a d'abord proposé SDL (Service Definition Language) avec une implémentation fournie dans leur Toolkit SOAP, puis IBM a proposé NASSL (Network Accessible Service Specification Language), dont une implémentation est fournie dans SOAP4J. Microsoft modifia sa première spécification et proposa SCL (SOAP Contract Language), puis les différents acteurs s'accordèrent sur WSDL [Did-Tan ,2001].

I.4.2.1) Structure d'un document WSDL

La structure du document WSDL est représentée sur la figure 5 :

```
  <definitions>
<message>
            …
</message>
<portType>
            <operation> …</operation>
            <operation> … </operation>
            …
</portType>
<binding>
…
</binding>
<service>
            <port> … </port>
            <port> … </port>
</service>
  </definitions>
```

Figure 5 - Structure d'un document WSDL

□□**<definitions>** : Cet élément contient la définition du service. C'est la racine de tout document WSDL. Cette balise peut contenir les attributs précisant le nom du service, et les espaces de nommage. <definitions> contient trois types d'éléments :

- □□**<message>** et **<portType>** : Ces éléments définissent les opérations offertes par le service, leurs paramètres d'entrée et de sortie, etc. En particulier, un <message> correspond à un paramètre d'entrée ou de sortie d'une <operation>. Un <portType> définit un ensemble d'opérations. Une **<operation>** définit un couple message -entrée / message -sortie. Par exemple, dans le monde Java, une opération est une méthode et un portType une interface.

- **<binding>** : Cet élément associe les <portType> à un protocole particulier. Les bindings possibles sont SOAP, CORBA ou DCOM. Actuellement seul SOAP est utilisé. Il est possible de définir un binding pour chaque protocole supporté.
- **<service>** : Cet élément précise les informations complémentaires nécessaires pour invoquer le service, et en particulier l'URI du destinataire. Un <service> est modélisé comme une collection de ports, un <**port**> étant l'association d'un <binding> à un URI.

I.4.3) Recherche des services Web : UDDI

Après la publication et la mise en place d'un protocole de communication du service Web, la dernière étape est le référencement. Un service Web doit être référencé afin de pouvoir être retrouvé et utilisé par une autre organisation (ou un autre service). Pour cela, il existe des annuaires pouvant être soit internes à l'organisation, soit universels.

Il existe de nombreux annuaires tels que : ebXML, registre se consacrant au e-commerce et UDDI issus des travaux du consortium OASIS. Nous choisissons de décrire le registre UDDI.

I.4.3.1) Principe de UDDI

UDDI (Universal Description, Discovery and Integration) a été créé par IBM, Microsoft et Ariba. C'est une architecture répartie qui permet aux fournisseurs de services Web (Business providers), d'enregistrer leurs services, et aux applications de rechercher les services correspondant à leurs besoins, de façon normalisée. UDDI est donc un annuaire distribué de services Web et d'entreprises (Business/Service Registry). UDDI se comporte lui-même comme un service Web dont les méthodes sont appelées via le protocole SOAP [UDDI-org, 2007]. UDDI est composée de deux parties :

- L'UDDI Business Registry : annuaire d'entreprises et de services Web ;
- Les interfaces d'accès à ces annuaires, et les modèles de données.

Le noyau du projet UDDI est l'UDDI Business Registry, annuaire contenant des informations de trois types, décrites au format XML :

- *Pages blanches (White paper)* : Ce composant permet de décrire l'organisation proposant le service. Cette description est constituée du nom de l'organisation, de son adresse, etc. En résumé, elle contient toutes les informations jugées pertinentes pour identifier l'organisation;
- *Pages jaunes (Yellow paper)* : les pages jaunes de UDDI détaillent la description de l'organisation faite dans les pages blanches en répertoriant les services proposés. Dans cette section sont évoqués : la catégorie de l'entreprise, le secteur d'activité dans lequel exerce l'entreprise, les services offerts par cette organisation.

- *Pages vertes (Green paper)* : informations techniques sur les services proposés. Les pages vertes incluent des références vers les spécifications des services Web, et les détails nécessaires à l'utilisation de ces services : interfaces implémentées, information sur les contacts pour un processus particulier, description du processus en plusieurs langages, catégorisation des processus, pointeurs vers les spécifications décrivant chaque API.

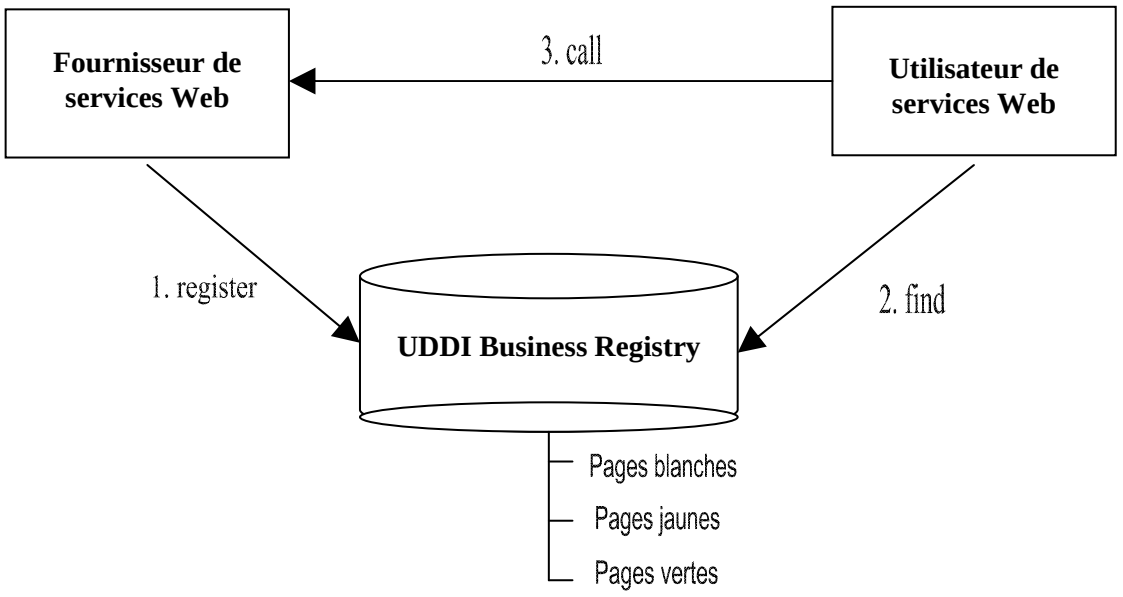

Figure 6 - Scénario classique d'utilisation d'UDDI

La figure 6 représente un scénario classique d'utilisation de UDDI :

- Les fournisseurs de services Web s'enregistrent auprès de l'UDDI Business Registry (sur les pages blanches et jaunes) et ajoutent leurs services (en complétant les pages jaunes et en renseignant les détails techniques sur ces services dans les pages vertes). En principe, un seul enregistrement d'un service sur un référentiel est nécessaire. UDDI étant un service distribué sur Internet, toutes les actions sont automatiquement répercutées sur les autres référentiels (figure 7).
- Un utilisateur recherche une entreprise fournissant un service donné, et obtient une description de l'entreprise qui le fournit par le biais des pages blanches et jaunes, et des détails de l'invocation du service offert par le biais des pages vertes ;
- L'utilisateur peut alors invoquer le service Web distant en utilisant SOAP.

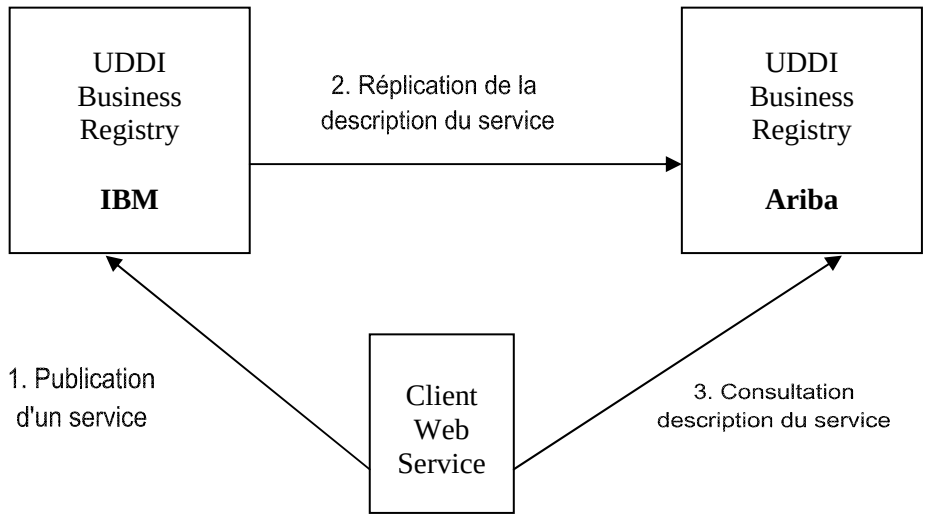

Figure 7 - UDDI- réplication automatique des descriptions des services

I.4.3.2) Organisation structurelle de UDDI

La structure de l'annuaire s'articule autour de 4 éléments :

- L'entité commerciale (Business entity), élément composant les pages blanches
- Les offres de services (Business service) dont la collection constitue les pages jaunes
- Les liaisons UDDI (Binding template)
- Les types de services (tModel ou service type)

La figure 8 montre une simplification du modèle UDDI :

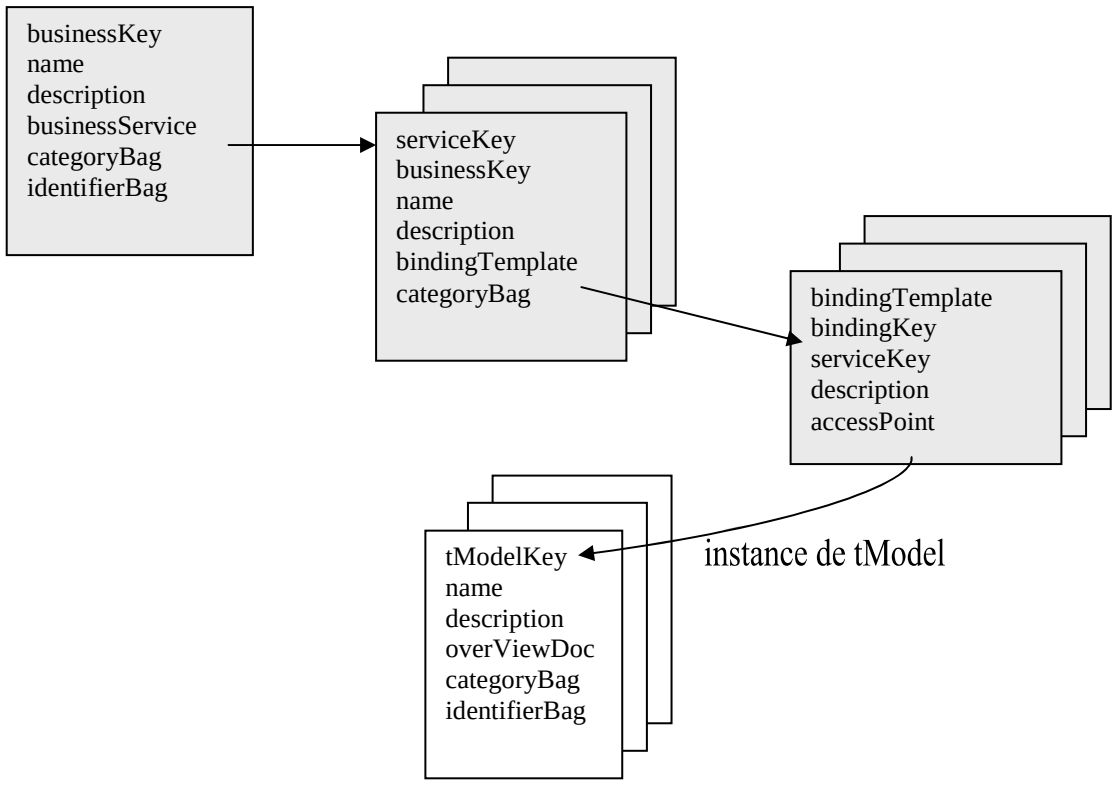

Figure 8 - Structure simplifiée de l'annuaire UDDI

a- L'entité commerciale (business Entity)

Cette structure décrit une entreprise et les divers services qu'elle offre. Elle renseigne aussi sur les catégories auxquelles appartient l'entreprise. Pour ce faire, UDDI dispose d'un référentiel de taxonomies.

Cette entité regroupe plusieurs informations telles que le nom de l'entreprise, son adresse ou encore l'URL de son site Web. Chaque entité commerciale est associée à un identifiant unique UUID (*Universal Unique Identifier*) businessKey. A ces informations s'ajoutent des liens vers les autres entités de l'annuaire UDDI. La figure 9 montre un résumé de la classe Entité commerciale.

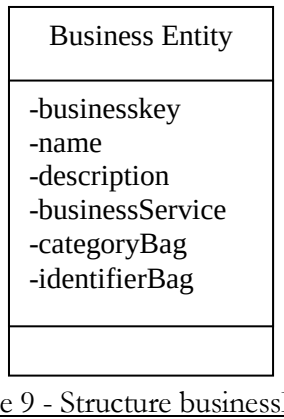

Figure 9 - Structure businessEntity

b- L'entité Offre de service (business Service)

Cette entité distingue les services proposés par l'organisation. Cette offre de service est identifiée par un nom et un UUID (*Universal Unique Identifier*) décrit par l'élément : serviceKey. Sa structure est décrite dans la figure ci-dessous :

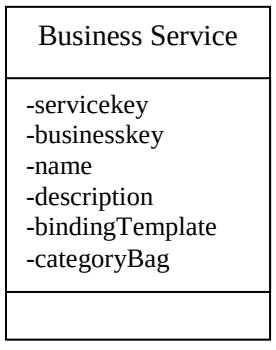

Figure 10 - Structure businessService

c-Les liaisons UDDI (Binding Template)

Ce module décrit les points d'accès aux services Web (URL) et le moyen d'y accéder (les différents protocoles à utiliser). Chaque liaison est identifiée par un UUID déterminé par l'élément bindingKey. Sa structure est décrite dans la figure 11.

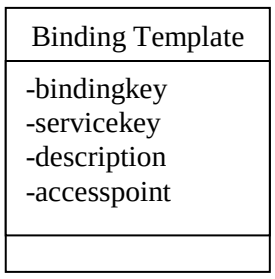

Figure 11 - la structure de l'entité Liaison

d-Les types de services (tModel)

Le tModel permet d'associer un service à sa description en WSDL. L'utilisateur potentiel peut ainsi avoir connaissance des conventions d'utilisation du service. Chaque type de service possède un identifiant UUID décrit dans l'élément tModelKey, comme le montre la structure en figure 12.

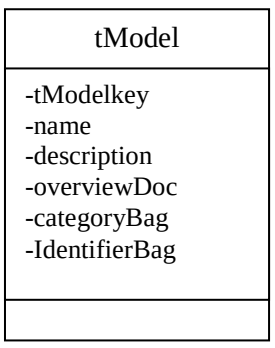

Figure 12 - Structure de l'entité Type de Service

Mais, il ne faut pas perdre de vue que la principale finalité de l'annuaire UDDI est la recherche de services Web. Pour ce faire, on dispose de 2 méthodes :

- Utiliser les interface Web offertes pas les opérateurs, notamment Microsoft UDDI à l'adresse **Http://uddi.microsoft.com et IBM à l'adresse http://www.3.ibm.com/-services/uddi/.**
- Automatiser la recherche en incluant aux applications clientes les API (interfaces de programmation pour les applications) d'interrogations UDDI fournies par les éditeurs.

En règle générale, les API fournies par Microsoft et IBM offrent deux types de méthodes : find_xxx et get_xxx.

Les méthodes find_xxx permettent d'effectuer une recherche parmi les entités de l'annuaire UDDI, xxx représentant le type d'entité recherché.

Les méthodes get_xxx permettent de lire les informations d'un enregistrement d'entité xxx.

Les mêmes API offrent les méthodes de publication d'un service Web. Notamment les méthodes d'enregistrement et de modification d'un service.

I.5) Outils disponibles pour la mise en œuvre de S.W

La base des services Web étant étudiée, cette section est consacrée aux différents outils disponibles afin de mettre en œuvre ce concept.

I.5.1) Plates-formes commerciales

Les logiciels de déploiement de services Web présentés ci-dessous sont des produits d'éditeurs de logiciels reconnus : MicroSoft, IBM et Sun. Nous avons volontairement restreint les éditeurs aux plus importants car le concept de service Web atteint aujourd'hui de nombreux acteurs du monde informatique.

I.5.1.1) Visual Studio .NET[2] de MicroSoft

Le logiciel Visual Studio .NET permet la gestion de toutes les technologies de base du développement de services Web : SOAP, XML, WSDL et UDDI.

Plus particulièrement, cet outil facilite le déploiement de fichiers WSDL. Ces documents de description de services sont générés automatiquement et sont accessibles à partir d'un URL désigné spécialement par l'outil. Les applications clients déployées ont seulement besoin de la référence du document WSDL afin de localiser les services Web. L'accès aux services devient alors transparent.

I.5.1.2) WebSphere Studio Application Developper[3] de IBM

L'intégralité des technologies de base est disponible dans ce logiciel de déploiement de services Web. De plus, ce logiciel intègre un outil de génération automatique des documents WSDL. Le registre UDDI est supporté par WebSphere Studio Application Developper depuis sa version 5.

L'avantage de cet outil est le fait qu'il soit supporté par un serveur d'application permettant de générer automatiquement le code client accédant aux services Web.

I.5.1.3) Sun ONE developper Studio[4] de Sun

Les quatre technologies de base du développement de services Web sont supportées par ce logiciel.

Les protocoles de communication sont assurés par SOAP 1.1 et JAX RPC. JAX RPC est une version java de SOAP. Elle peut produire des fichiers attachés aux services Web par l'intermédiaire du mécanisme WS-Attachements.

Sun One propose le mécanisme de génération croisée de documents WSDL et Java. Ce mécanisme facilite la tâche de l'utilisation en déployant le fichier WSDL à partir d'une classe Java et inversement.

---

[2] http://msdn.microsoft.com/vstudio/productinfo/trial/default.aspx
[3] http://www-3.ibm.com/software/awdtools/studioappdev/
[4] http://www.sun.com/software/sundev/jde/index.html

I.5.2) Plates-formes publiques ou libres

L'utilisation des services Web étant en pleine expansion, l'éditeur Sun met à la disposition d'utilisateurs potentiels de ce concept un logiciel public.

La communauté de logiciels libres Apache participe aussi à la mise en œuvre des services Web en proposant plusieurs outils. Nous en présentons deux : Apache-Soap et Axis.

I.5.2.1) Java WSDP[5] (Web Services Developer Pack) de Sun

Ce logiciel inclut les standards de base des services Web : WSDL, SOAP, et UDDI. A l'identique de Sun One, Java WSDP possède de nombreuses API. Le registre universel UDDI est accessible grâce à l'API JAXR (Java API for XML registries). De plus, ce logiciel possède un serveur permettant l'accès à UDDI, appelé Java WSDP Registry Server.

Le protocole de communication est mise en œuvre grâce à deux API différents : JAX-RPC (Java API for XML-based RPC), et SAAJ (SOAP with Attachements API for Java).

I.5.2.2) Apache-SOAP[6]

Apache-SOAP est le premier projet de la fondation Apache concernant les services Web. Cet outil fonctionne sur toutes plates-fomes supportant le langage Java.

A l'origine, ce projet représentait uniquement une implémentation de la recommandation du W3C, SOAP.

Ce logiciel ne supporte pas WSDL : la génération automatique des fichiers WSDL n'est pas possible. Le développement de ces fichiers doit être manuel malgré son écriture fastidieuse.

I.5.2.3) Axis – Apache eXtensible Interaction System

Axis[7] succède à Apache-SOAP dans le cadre des projets de services Web, et plus particulièrement dans l'implémentation de SOAP.

De même que l'outil précédent, Axis est utilisable sur toute plate-forme supportant le langage Java. Ce logiciel possède une nouvelle architecture afin d'être plus flexible et plus performant que Apache-SOAP. L'amélioration la plus conséquente apportée à Axis est la génération automatique de fichiers WSDL

I.6) Conclusion

Dans ce chapitre nous avons présenté le concept des services Web comme étant la dernière technologie pour l'intégration et l'interopérabilité des systèmes réparties. Basés sur le standard XML, ils sont caractérisés par leurs indépendances aux plates formes et aux systèmes

[5] http://java.sun.com/webservices/downloads/webservicespack.html
[6] http://ws.apache.org/soap/
[7] http://ws.apache.org/axis/

d'exploitation, ce qui a impliqué leur adoption par les différentes organisations commerciales et industrielles offrant leurs services à travers le Web, et par conséquent l'augmentation du nombre de services offerts. La découverte de services devient de ce fait un des aspects les plus importants relatifs aux services Web.

La technologie fondamentale pour la découverte de services Web est le registre UDDI. Destiné à être utilisé par les utilisateurs humains, UDDI permet la recherche et la sélection manuelle de descriptions des services Web. UDDI fournit une API de recherche basée mots clés. De plus, il permet aussi de faire la recherche des services en se basant sur leurs descriptions WSDL. Cependant la recherche dans UDDI (comparaison de la requête avec les descriptions) n'est faite qu'au niveau syntaxique. Cette méthode présente des limitations ; elle ne permet pas de trouver le service demandé à chaque fois ; de plus, un agent logiciel ne peut pas examiner la description textuelle destinée pour des humains, il ne peut pas distinguer entre deux services Web différents ayant la même description syntaxique, ce qui présente un handicap pour l'automatisation de la découverte, composition et collaboration des services Web.

Pour permettre l'automatisation des diverses tâches liées aux services Web, l'idée consiste à enrichir les descriptions des services Web par d'autres informations supplémentaires compréhensibles par les machines. C'est la description sémantique des services Web, puisque deux services Web peuvent avoir la même description syntaxique, mais avoir deux sens différents ou inversement. Des services Web dotés de description sémantique sont dits services Web sémantiques. C'est ce que nous allons voir dans le chapitre suivant.

Chapitre II

Les Services Web Semantiques et Decouverte de services

II.1) Introduction

Depuis le début des années 2000, de nombreux axes de recherches concernant Internet se sont tournés vers le Web sémantique. Internet est un immense réservoir de documents de tout genre (articles, sites personnels, audios, vidéos) mais plus ces ressources s'accumulent, plus la recherche d'une information précise s'avère difficile. Tim Berners-Lee [Ber, 2001], pionnier dans le domaine d'Internet, propose alors d'ajouter à toutes ces ressources une sémantique qui permettrait aux systèmes informatiques d'en « comprendre » le sens en accédant à des collections structurées d'informations et à des règles d'inférence qui peuvent être utilisé pour conduire des raisonnements automatisés : c'est la naissance du Web sémantique.

Suite à l'explosion du commerce électronique, il a été nécessaire d'offrir toutes sortes d'applications et de services via Internet. La communauté des « commerces et organisations accessibles par le Web » a fait le plus gros du travail : ils ont connecté une énorme quantité de produits et services de tout genre à Internet, en les rendant accessibles à des ordinateurs au travers de protocoles de communication simples. Mais parallèlement, le domaine des agents intelligents a également envahit Internet [Bry, 2002].

Maintenant, le but recherché par ces agents est de pouvoir se servir des services Web afin de mieux satisfaire leurs utilisateurs (par exemple, pouvoir proposer l'organisation d'un voyage en faisant appel à différents services Web). Pour cela, il est nécessaire que ces agents puissent comprendre les fonctionnalités proposés par les services : c'est la naissance des services Web sémantiques.

Dans la section suivante, nous présentons le Web sémantique et les ontologies qui représentent la technologie clé pour sa réalisation.

II.2) Le Web sémantique

II.2.1) Architecture du Web sémantique

La vision courante du Web sémantique proposée par Berers-Lee [Ber, 2001] peut être représentée dans une architecture en plusieurs couches différentes (voir figure 13).

Les couches les plus basses assurent l'interopérabilité syntaxique : la notion d'URI (Uniform Resource Identifier) fournit un adressage standard universel permettant d'identifier les ressources tandis que Unicode est un encodage textuel universel pour échanger des symboles. Rappelons que l'URL (Uniform Resource Locator), comme l'URI, est une chaîne courte de caractères qui est aussi utilisée pour identifier des ressources (physiques) par leur localisation.

XML (eXtensible Markup Language) fournit une syntaxe pour décrire la structure du document, créer et manipuler des instances des documents. Il utilise l'espace de nommage

(namespace) afin d'identifier les noms des balises (tags) utilisées dans les documents XML. Le schéma XML permet de définir les vocabulaires pour des documents XML valides. Cependant, XML n'impose aucune contrainte sémantique à la signification de ces documents, l'interopérabilité syntaxique n'est pas suffisante pour qu'un logiciel puisse "comprendre" le contenu des données et les manipuler d'une manière significative.

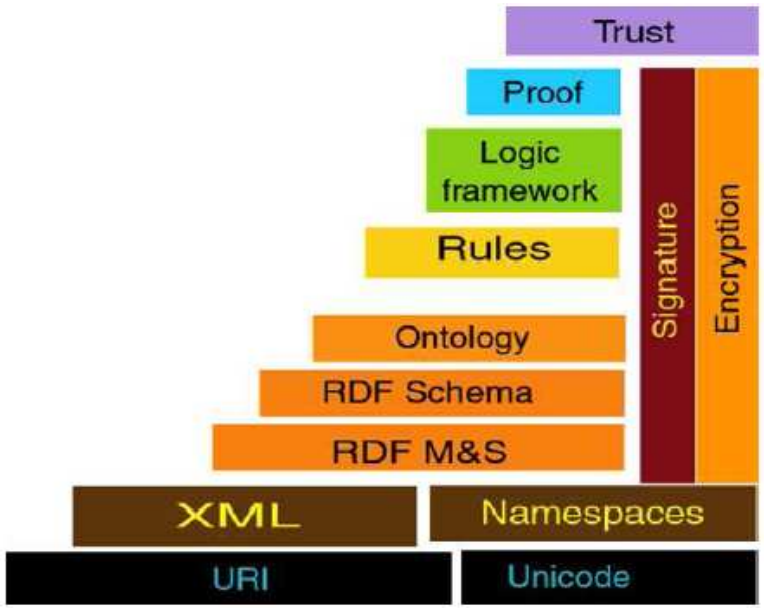

Figure 13 - Architecture du Web sémantique [Ber, 2001]

Les couches RDF M&S (RDF Model and Syntax) et RDF Schéma sont considérées comme les premières fondations de l'interopérabilité sémantique.

Elles permettent de décrire les taxonomies des concepts et des propriétés (avec leurs signatures). RDF fournit un moyen d'insérer de la sémantique dans un document, l'information est conservée principalement sous forme de déclarations RDF. Le schéma RDFS décrit les hiérarchies des concepts et des relations entre les concepts, les propriétés et les restrictions domaine/co-domaine pour les propriétés.

La couche suivante Ontologie décrit des sources d'information hétérogènes, distribuées et semi-structurées en définissant le consensus du domaine commun et partagé par plusieurs personnes et communautés. Les ontologies aident la machine et l'humain à communiquer avec concision en utilisant l'échange de sémantique plutôt que de syntaxe seulement.

Les règles (Rules) sont aussi un élément clé de la vision du Web sémantique, la couche Règles offre la possibilité et les moyens de l'intégration, de la dérivation, et de la transformation de données provenant de sources multiples, etc.

La couche Logique se trouve au-dessus de la couche Ontologie. Certains considèrent ces deux couches comme étant au même niveau, comme des ontologies basées sur la logique et

permettant des axiomes logiques. En appliquant la déduction logique, on peut inférer de nouvelles connaissances à partir d'une information explicitement représentée.

Les couches Preuve (Proof) et Confiance (Trust) sont les couches restantes qui fournissent la capacité de vérification des déclarations effectuées dans le Web Sémantique. On s'oriente vers un environnement du Web sémantique fiable et sécurisé dans lequel nous pouvons effectuer des tâches complexes en sûreté.

D'autre part, la provenance des connaissances, des données, des ontologies ou des déductions est authentifiée et assurée par des signatures numériques, dans le cas où la sécurité est importante ou le secret nécessaire, le chiffrement est utilisé.

II.2.2) Composants principaux du Web sémantique

Nous examinons dans cette section le composant essentiel et important du Web sémantique qu'est les ontologies.

II.2.2.1) Les Ontologies

Le terme ontologie est initialement emprunté de la philosophie signifiant "explication systématique de l'existence". Une ontologie est similaire à un dictionnaire ou un glossaire mais avec une structure détaillée et grande qui permet aux machines de traiter son contenu.

Définition :

"Il s'agit de représentations formelles d'un domaine de connaissance sous la forme de terminologies dotées de relations sémantiques." [Bert, 2002].

On utilise l'ontologie dans différents domaines : la représentation d'informations et de connaissances, l'intégration des systèmes d'informations, la spécification des systèmes, etc. Mais aussi dans :

- **La communication :** L'ontologie ne permet jamais que deux mots différents possèdent la même sémantique.
- **L'interopérabilité :** L'ontologie peut être considérée comme un pont ou une passerelle entre les différents systèmes. "Elle sert à définir le format d'échange entre les systèmes."[Tuan, 2001].

Dans le Web sémantique, l'ontologie permet à l'utilisateur lors d'une recherche sur le Web d'accéder non seulement aux documents liés aux mots clés de la requête, mais aussi à ceux qui sont liés ontologiquement (sémantiquement) à ces derniers, ce qui rend la recherche encore plus pertinente. Elle a pour but de décrire des concepts et les relations qui les lient entre eux, et avec des règles de déduction les rendre plus compréhensibles et utilisables par les différents agents (humains ou logiciels). En dernier lieu, elle est interopérable.

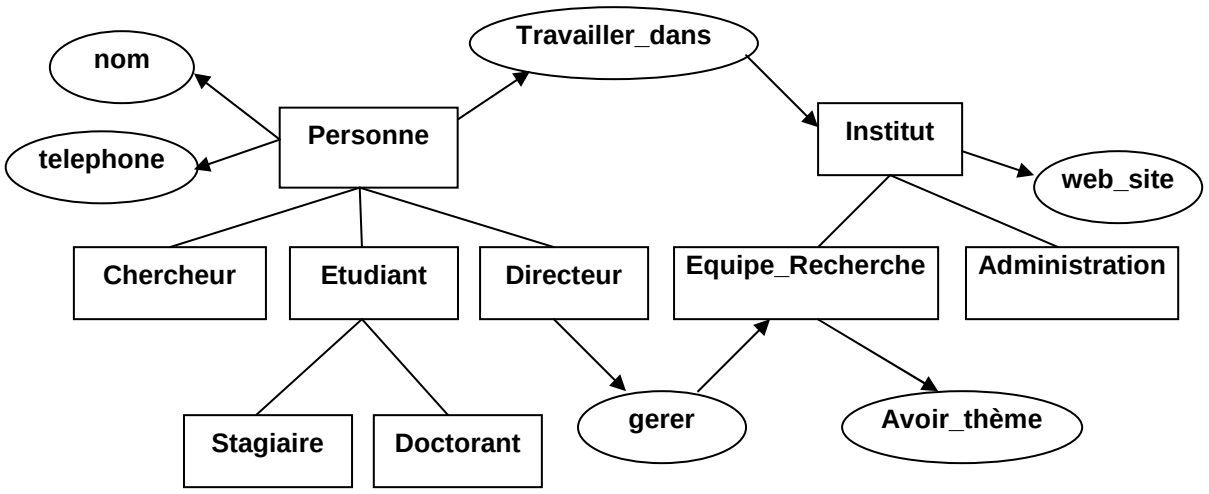

Figure 14 - Exemple d'un extrait de l'ontologie

La figure 14 présente un extrait d'une ontologie décrivant les connaissances du domaine de la recherche scientifique, par exemple des activités d'un institut de recherche. Dans cet extrait, nous avons des concepts tels que Personne, Doctorant, Chercheur qui sont classés en ordre hiérarchique… pour représenter des employés qui travaillent dans (la propriété) les équipes de recherche (le concept Equipe_Recherche) ou dans le département d'administration (le concept Administration), etc.

II.3) Les services Web sémantiques

II.3.1) Exemple de motivation

L'idée des services Web sémantiques est présentée graduellement en regardant d'abord la méthode courante de description des services Web.

Les capacités des services Web sont décrites avec le langage de définition de service (WSDL) qui associe aux opérations des noms et aux paramètres leurs types de données. Considérons un service Web de traduction entre deux langues, Appelé *TranslateArabicToEnglish*, accepte deux paramètres, un paramètre en entrée (Input parameter) nommé *WordToTranslate* de type String, et un paramètre en sortie (Output parameter) appelé *TranslatedWord* de type String aussi (voir figure 15).

TranslateArabicToEnglish		
	Type	Nom de paramètre
Entrée	String	WordToTranslate
Sortie	String	TranslatedWord

Figure 15 – Description syntaxique du service Web TranslateArabicToEnglish

Cependant, le problème avec ce type de description est que lorsqu'on veut automatiser les divers aspects liés aux services Web (découverte, composition,…), l'agent manipulant les paramètres du service Web ne peut pas avoir la signification de ces données. Pour l'agent logiciel, c'est juste des variables dénotées contenant de l'information. La seule chose que l'agent logiciel peut inférer de la description précédente est que les deux paramètres *WordToTranslate* et *TranslatedWord* sont respectivement un paramètre d'entrée et un paramètre de sortie de type String. A ce stade, les services Web ne sont décrits qu'au niveau syntaxique. Un développeur voulant programmer une application client interagissant avec un service Web doit tout d'abord avoir connaissance de la syntaxe de sa description, l'interpréter, puis écrire le code client conforme aux paramètres de la description du service Web. Cependant, un agent logiciel ne peut lire la description d'un service Web comme un humain, il peut avoir connaissance de la structure syntaxique de la description mais pas sa sémantique.

Les services Web sémantiques sont des services Web décrits de telle sorte qu'un agent logiciel puisse interpréter les fonctionnalités offertes par le service Web. Un agent logiciel doit être capable de lire la description d'un service Web pour déterminer si le service Web fournit les fonctionnalités désirées, et s'il est lui-même capable d'utiliser ce service. Pour permettre cela, la description du service Web doit être supplémentée en information sémantique interprétable par machine. Les paramètres du service Web doivent être décrits de façon qu'un agent logiciel puisse avoir connaissance de leur signification. Cela est fait par la définition de vocabulaires organisés en ontologies.

Dans le domaine des services Web, les ontologies sont utilisées pour annoter les descriptions des fonctionnalités des services Web de sémantique. Ainsi pour qu'un agent logiciel puisse avoir connaissance de la sémantique d'une description d'un service Web, il lui suffit juste d'accéder à une ontologie du domaine.

TranslationService			
	Type	Nom	Annotation sémantique
Entrée	String	WordToTranslate	ArabicWordToTranslate
Sortie	String	TranslatedWord	EnglishTranslatedWord

Figure 16 - un service de traduction annoté syntaxiquement.

La figure 16 représente la même description du service précédent, sauf que cette description est augmentée d'une sémantique. Une colonne supplémentaire contenant l'annotation sémantique des paramètres est introduite. Les noms sémantiques utilisés dans la description sont plus

descriptifs que les noms des paramètres. Ces noms font référence à des concepts définis dans des ontologies.

Un agent essayant d'interpréter la description du service Web aura juste à utiliser l'ontologie où les annotations sémantiques sont définies. En utilisant un moteur d'inférence, l'agent peut inférer les similitudes entre la sémantique employée pour décrire le service et la sémantique connue par l'agent.

La signification des descriptions des services est indirectement dérivée par un moteur d'inférence. Pour raffiner ces descriptions les chercheurs ont défini plusieurs paramètres tels que les types d'opérations, les préconditions, et les effets. Les entrées et les sorties représentent une *transformation de l'information* alors que les préconditions et les effets représentent l'état de transformation.

Intuitivement les préconditions (conditions préalables) représenteraient ce qui est nécessaire pour l'usage du service, et les effets représentent les conséquences d'emploi du service [Kva, 2004].

Un service Web sémantique est un service Web décrit en utilisant des annotations sémantiques dans un langage bien défini (par exemple ontologies) qui permettent au service d'avoir une interface interprétable sans ambiguïté, facilitant l'automatisation de certaines tâches telles que la découverte, la sélection, l'invocation et la composition. Ces services s'appuient sur des langages du Web sémantique pour décrire leurs fonctionnalités et les données qu'ils échangent. Les avantages de l'utilisation du Web sémantique pour la description des services Web sont nombreuses. En plus de rendre l'interface du service accessible automatiquement par des machines, ils permettent également, la description de propriétés non-fonctionnelles telles que la qualité de services et les contraintes de sécurité, d'une manière uniforme compréhensible par tous.

Le langage de description de services Web sémantiques de référence est le langage OWL-S, une extension du langage OWL, proposé par le groupe de recherche sur le Web sémantique de la DARPA[8] à l'origine du langage DAML[9].

II.3.2) Le langage OWL-S

Appelé DAML-S dans ses premières versions [Ank, 2003], le langage OWL-S (Ontology Web Language for Service) [Mar, 2004] basé sur DAML+OIL a pour objectif d'ajouter des descriptions sémantiques aux services Web (en plus de leur description syntaxique WSDL). Alors que les efforts industriels sont actuellement focalisés sur la standardisation des mécanismes

[8] Defense Advanced Research Projects Agency
[9] DARPA Agent Markup Language. HTTP ://www.daml.org/

d'enregistrement et de découverte de services, sur l'interopérabilité des services indépendamment de la plateforme, ainsi que l'échange de types de documents syntaxiquement bien formés entre services Web. OWL-S a pour objectif de fournir une plus grande expressivité en permettant la description des caractéristiques des services afin de pouvoir raisonner dessus dans le but de découvrir, invoquer, composer et gérer les services Web de façon la plus automatisée possible.

- **Découverte automatique de services Web**. Actuellement cette tâche doit être réalisée par un humain qui doit utiliser un moteur de recherche ou un annuaire pour trouver le service, lire la page Web qui décrit l'utilité et l'utilisation du service, puis l'exécuter manuellement pour vérifier que celui-ci correspond bien aux attentes de l'utilisateur. OWL-S doit donc fournir une description déclarative des propriétés et des capacités du service Web.
- **Invocation automatique de services Web**. L'invocation automatique d'un service signifie l'exécution du service par un programme informatique ou un agent logiciel. Cet agent doit être capable d'interpréter les descriptions OWL-S afin de délivrer les données nécessaires à l'exécution du service Web.
- **Composition automatique de services Web**. L'objectif qu'un utilisateur veut atteindre nécessite souvent l'utilisation de plusieurs services Web. L'agent logiciel chargé d'atteindre cet objectif doit disposer suffisamment de données afin de pouvoir sélectionner, composer et interopérer automatiquement ces services Web. Les descriptions OWL-S doivent donc pouvoir fournir toutes ces informations.
- **Surveillance automatique de l'exécution de services Web**. Un des problèmes majeurs avec l'exécution des services, du fait de leur nature distribuée, est que l'utilisateur ne connaît pas le temps d'exécution de ces services. Lors de la composition de plusieurs services, l'utilisateur doit pouvoir connaître l'état d'avancement de ses différentes requêtes.

Le langage OWL-S organise la description d'un service en trois zones conceptuelles : le profil (Profile), le modèle de processus (ProcessModel) et les liaisons avec le service (Grounding). La figure 17 représente ces trois concepts qui permettent de répondre aux questions suivantes :

- *Quelles sont les informations concernant l'agent utilisateur (humain ou logiciel)* nécessaires *à l'exécution du service ? Quelles sont les informations renvoyées par le service ?* La réponse à cette question est donnée par le « profil » du service : la classe ServiceProfile.

- *Comment fonctionne le service ?* La réponse est donnée par le « modèle de processus » du service : la classe ServiceModel.
- *De quelle façon le service doit-il être utilisé* ? La réponse à cette dernière question est donnée par la classe ServiceGrounding.

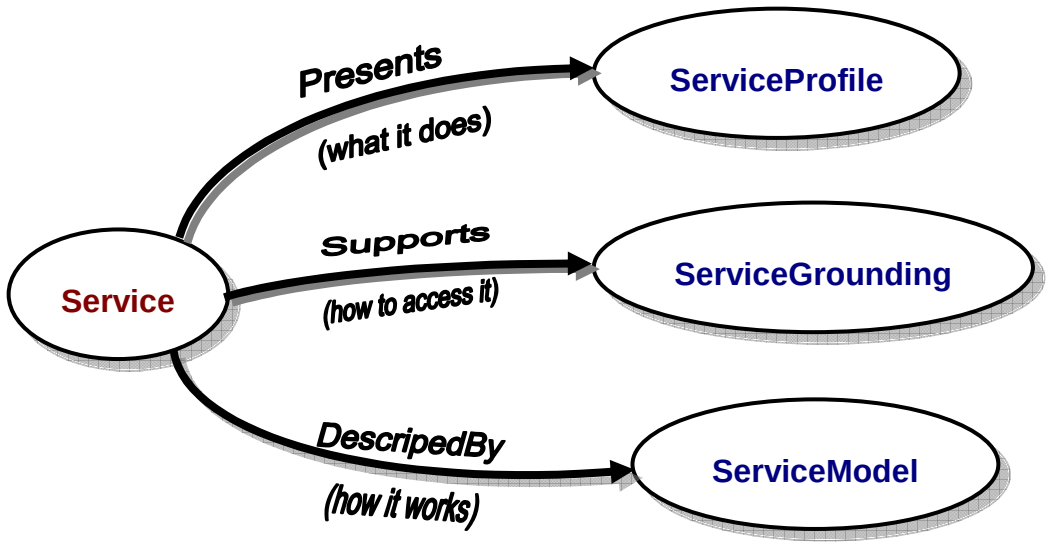

Figure 17 – Ontologie de services

D'une manière générale, la classe ServiceProfile donne les informations nécessaires à un agent pour publier ou découvrir un service. Les profils des services sont généralement organisés sous la forme de taxonomies, qui constituent le premier niveau de discrimination lors de la recherche d'un service Web spécifique [Bry, 2002]. La classe ServiceModel donne les conditions nécessaires à l'exécution d'un service ainsi que ses conséquences. Elle inclut des informations concernant les entrées, sorties, préconditions et effets d'un service. Dans le cas d'un service complexe (composé de plusieurs étapes dans le temps), le modèle de processus détaille la décomposition du service en composants plus simples et en explicitant les liaisons et structures de contrôle permettant l'enchaînement de ces composants. La classe ServiceGrounding spécifie la manière, pour un agent, d'accéder au service concerné. De plus, la classe ServiceGrounding doit définir, pour chaque type abstrait spécifié dans le ServiceModel, une façon non ambiguë d'échanger des données de ce type avec le service.

II.4) Les mécanismes de découverte de services web

La découverte de services Web présente un axe de recherche important. Divers mécanismes de découverte ont été proposée dans la littérature. Dans [Mc, 2004], les auteurs ont défini le mécanisme de découverte comme étant : " l'acte de localisation d'une description, traitable par machine, d'un service Web non connu auparavant décrivant certains critères fonctionnels ".

Actuellement, les descriptions des services Web sont publiées dans des registres spécialement conçus à cet effet (exemple : UDDI). Le but de ces registres est de faciliter la recherche des services publiés par les différents organismes commerciaux. Cependant, vu le nombre important et la diversité des services Web, leur découverte reste une tâche ardue.

Initialement la découverte de services Web était principalement syntaxique (correspondance syntaxique des mots clés de la requête avec la description des services Web). Mais avec le développement des technologies du Web sémantique, les techniques proposées pour la découverte de services Web sont devenus essentiellement sémantiques (degré de similitude entre les termes de la requête et les description sémantiques des services Web). En général, les approches de découverte peuvent être classées en deux catégories :

1- approches basées sur les descriptions syntaxiques des S.W

2- approches Web sémantique.

II.4.1) Approches basées sur la description syntaxique

Le principe général des approches basées sur la syntaxe des descriptions des services est la comparaison syntaxique entre la requête, basée mots clés, de l'utilisateur et les descriptions syntaxiques (WSDL) des services Web.

II.4.1.1) Architecture centralisée

Dans une architecture centralisée les descriptions des services Web sont sauvegardées dans un même registre.

- **Approche UDDI**

UDDI est un registre de descriptions des services Web. Dans le cas d'une architecture centralisée, UDDI est utilisé comme registre central pour la publication et la découverte, basée mots clés, des services Web.

A l'étape de recherche, l'utilisateur ou le programme de recherche envoie une requête constituée de mots clés, cette requête est ensuite comparée avec les mots clés du registre UDDI. Un ensemble de descriptions des services Web est ensuite donné comme résultat de recherche, l'utilisateur sélectionne le service Web qui répond au mieux à ses exigences.

L'origine de cette approche est issue du domaine de recherche d'information (Information Retrieval IR). Malgré sa simplicité et sa facilité d'implémentation, elle présente quand même quelques inconvénients, la méthode renvoie un nombre important de résultats ou au contraire peu de résultats. Pour rendre la découverte de services Web basée mots clés plus efficace, une technique issue du domaine de IR a été adoptée [Atul, 2004]. Elle consiste à représenter les descriptions des services Web sous forme de vecteurs, tel que chaque vecteur contient un ensemble de mots issus des termes utilisés dans toutes les descriptions des services Web. Les

vecteurs de description sont ensuite organisés sous forme de matrice (terme×description). La deuxième étape consiste à appliquer, sur cette matrice, la technique LSI (Latent Semantic Indexing) (indexation sémantique latente) Cette méthode permet de renvoyer toute description de services Web qui a une relation sémantique avec la requête de recherche.

II.4.1.2) Architecture distribuée

a) UDDI distribué

Dans le cas d'une architecture distribuée où les descriptions des services ne sont pas centralisées dans un même registre UDDI, une approche est proposée dans [Porn, 2003]. Cette approche consiste à connecter un nombre arbitraire de nœuds du réseau pour former de façon virtuelle le registre UDDI, tel que chaque nœud contient une partie des descriptions des services Web du réseau. Cet ensemble de nœuds est appelé nuage ou fédération UDDI. Lorsqu'un utilisateur accède à l'un des nœuds pour la recherche d'un service Web, le nœud en question transmet la requête de recherche aux autres nœuds avec lesquels il est connecté, et ainsi de suite pour les nœuds recevant cette requête. Pour éviter qu'un nœud renvoie la même requête, un identificateur unique (ID) est associé à la requête. De plus, le nœud source de propagation de la requête associe à cette requête un nombre de sauts, tel que chaque fois que cette requête est propagée par un nœud, le nombre de sauts est décrémenté de un. Quand le nombre de sauts atteint zéro, la requête n'est plus propagée à travers les nœuds du nuage UDDI. Les résultats de chaque nœud ayant reçu la requête sont ensuite expédiés au nœud source. Dans cette approche la découverte est faite au niveau syntaxique.

b) Approche AASDU

Une approche multi-agents pour la découverte de services Web a été proposée dans [Paul, 2004]. Le système appelé AASDU (Agent Approach for Service Discovery and Utilization) (voir figure 18) contient quatre composants :

-Une interface utilisateur graphique (Graphical User Interface GUI)

-Un agent analyseur de requête (Query Analyser Agent QAA)

-Un système référentiel des domaines d'expertises des agents de service. Ce système permet de référencier les agents selon leur expertise, ainsi il n'est pas nécessaire que chaque agent ait connaissance de tous les services publiés dans les registres distribués. Chaque agent a juste connaissance des services relatifs à son domaine d'expertise. Dans ce système, chaque agent a un profil déterminant ses intérêts et son expertise. L'expertise de l'agent est représentée par un vecteur de mots clés tel que chaque mot clé représente un domaine donné. Pour chaque mot clé un score est assigné, indiquant le degré d'expertise de l'agent dans ce domaine. De plus, chaque agent a une liste d'agents voisins (Neighbor list). Cette

liste indique les voisins de l'agent ainsi que leurs domaines d'expertise. Lorsqu'un agent vient joindre le système, un ensemble de voisins lui est assigné de façon aléatoire.

-Le module de services : ce composant offre trois sous-services. Le premier service permet aux fournisseurs de services de publier les descriptions de leurs services Web. Pour chaque fournisseur de service Web un agent lui est assigné. Le deuxième service consiste en un agent de négociation permettant la sélection de service. Le troisième service est offert par l'agent composition, le rôle de cet agent est d'invoquer un des services issus de l'étape de sélection ou d'invoquer un service similaire lors de la défaillance du service sélectionné en premier.

Dans ce système l'utilisateur entre sa requête de recherche sous forme de chaîne de caractères via l'interface GUI. La requête est ensuite envoyée à l'agent QAA, le rôle de cet agent est de faire ressortir de cette requête les mots clés pertinents qui seront utilisés pour sélectionner des agents du système référentiel des domaines d'expertises des agents service. Pour se faire, l'agent QAA utilise une simple variante de la technique TFIDF (Term Frequency Inverse Document Frequency) pour ressortir les mots clés pertinents, sur la base de ces mots clés, l'agent QAA sélectionne un ensemble d'agents experts [Todd, 1997]. Les agents sélectionnés transmettent par la suite les paramètres des services avec lesquels ils sont liés à l'agent composition. Ce dernier invoque un des services selon le choix de l'utilisateur.

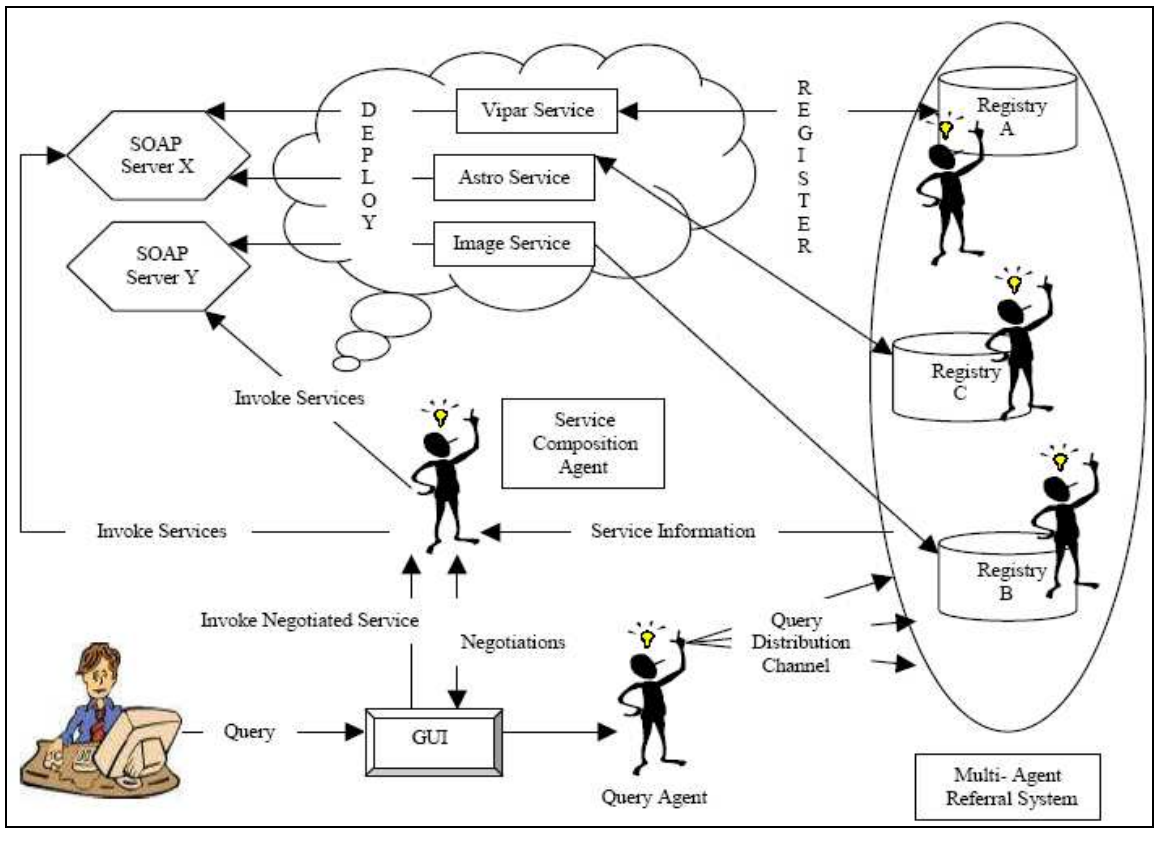

Figure 18 – Architecture AASDU

II.4.2) Approches Web sémantique

De récents travaux se sont focalisés sur la description sémantique des services web. Ce développement est de plus en plus significatif puisqu'il semble pouvoir aborder certaines insuffisances des approches basées sur les mots clés. Les ontologies sont le modèle utilisé pour la représentation sémantique des services Web, elle permet d'établir des relations sémantiques entre les différents concepts d'un domaine.

II.4.2.1) Architecture centralisée

Dans les architectures centralisées, les descriptions sémantiques des services Web sont sauvegardées dans un registre central.

a) Approche OWL-S

Parmi les ontologies proposées pour la description des services Web, nous avons l'ontologie DAML-S. Cette ontologie est basée sur le langage d'ontologie DAML. Elle est remplacée par une nouvelle ontologie basée sur le langage OWL, cette nouvelle ontologie est OWL-S [Dav, 2004]. OWL-S permet d'étendre UDDI avec la description sémantique des services Web [Pao, 2002].

OWL-S décrit un service Web à l'aide de trois classes. Parmi ces classes, ServiceProfile est la classe qui fournit les paramètres fonctionnels nécessaires pour la découverte de services Web tels que les entrées attendues, résultats produits en sortie, Préconditions et Effets. La découverte de services Web sémantique définis par l'ontologie OWL-S est basée sur l'algorithme de Matchmaking (littéralement, « trouver une correspondance »). Cet algorithme permet de rechercher les descriptions des services Web qui ont une correspondance sémantique entre les paramètres fonctionnels définis dans les descriptions des services et ceux introduits dans la requête de recherche [Nav, 2004].

Il y a correspondance sémantique entre les paramètres de sortie mentionnés dans la requête et ceux d'un service Web si seulement si pour chaque paramètre de sortie de la requête il existe un paramètre qui peut lui correspondre dans la description du service Web. Si un des paramètres de Sortie de la requête n'a pas de correspondance sémantique avec un des paramètres de sortie du service alors le service n'est pas sélectionné.

La correspondance sémantique entre les paramètres d'entrée est aussi faite de la même façon que pour les paramètres de sortie, sauf que l'ordre est inversé, c'est-à-dire la recherche d'une correspondance sémantique entre les paramètres d'entrée d'un service Web contre les paramètres d'entrée cités dans la requête.

```
outputMatch (outputsRequest, outputsAdvertisement) {
globalDegreeMatch = Exact
forall outR in outputsRequest do {
find outA in outputsAdvertisement such that
degreeMatch=maxdegreeMatch(outR, outA)
if (degreeMatch=fail) then return fail
if (degreeMatch<globalDegreeMatch) then
  globalDegreeMatch = degreeMatch
return sort (recordMatch); }
```

Figure 19 – Extrait de l'algorithme de matchmaking

Le degré de correspondance sémantique entre deux paramètres de sortie ou deux paramètres d'entrée dépend de la correspondance sémantique entre les concepts associés à ces paramètres d'entrée et de sortie. La correspondance sémantique entre deux concepts est basée sur la relation entre ces deux concepts dans leurs ontologies OWL. Par exemple, considérons un service Web de vente de véhicules dont le paramètre de sortie est spécifié comme « véhicule » et une requête de recherche dont le paramètre de sortie est spécifié comme « car ». Bien qu'il n'y ait pas de correspondance sémantique exacte entre les deux paramètres, étant donné le fragment de l'ontologie de véhicule (voir la figure 20), l'algorithme a pu déterminer un autre niveau de correspondance puisque le concept « véhicule » englobe (subsume) le concept « car ».

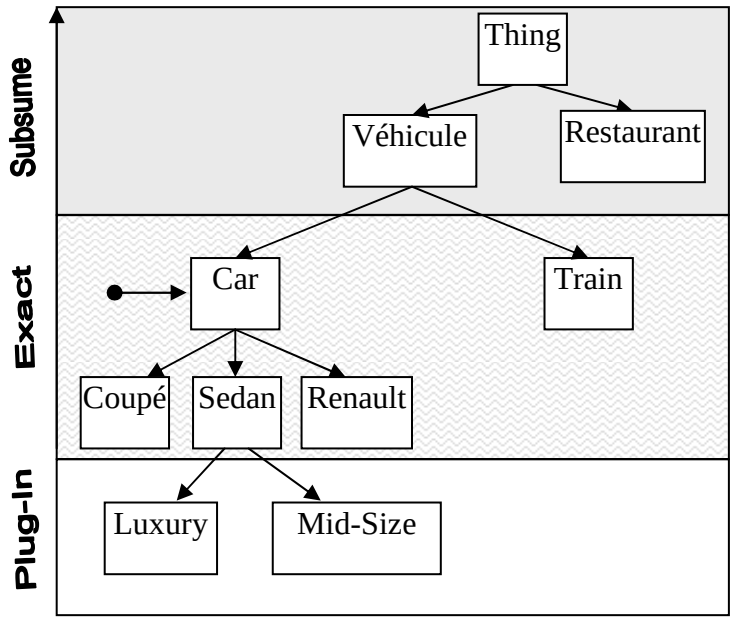

Figure 20 – Niveaux de correspondance

L'algorithme permet de déterminer quatre niveaux de correspondance sémantique entre deux concepts. Soit OutR représente le paramètre de sortie cité dans la requête de recherche, et

OutA le paramètre de sortie d'un service Web. Le degré de correspondance entre OutR et OutA peut être :

- **Exact** : Si OutR est OutA sont les mêmes ou si OutR est une sous classe immédiate de OutA dans l'ontologie du domaine de recherche.
- **Plug in** : Si OutA englobe (subsume) OutR
- **Subsume** : Si OutR englobe (subsume) OutA, dans ce cas le service Web peut ne peut satisfaire la requête complètement.
- **Disjoint** : S'il n'y a aucune relation entre OutA et OutR.

Le niveau de correspondance sémantique entre les paramètres d'entrée est assigné de la même manière que pour les paramètres de sortie, sauf que les arguments sont inversés, c'est-à-dire : degreeOfMatch(inA, inR) dans l'algorithme (figure 21).

```
degreeOfMatch(outR, outA)  {
if outA = outR then return Exact
if outR subclassOf outA then return Exact
if outA subsumes outR then return PlugIn
if outR subsume outA then return subsumes
otherwise fail
```

Figure 21 - Règles d'assignation du niveau de correspondance

La dernière partie de l'algorithme est la partie consacrée à la classification des descriptions de services sélectionnés. Les services sont classés selon le niveau de correspondance sémantique entre leurs paramètres de sortie et ceux cités dans la requête. Si deux services sont du même niveau de correspondance sémantique avec la requête de recherche par rapport à leurs paramètres de sortie, une comparaison sur le niveau de correspondance sémantique par rapport aux paramètres d'entrée est alors effectuée.

Cet algorithme permet de rechercher et de sélectionner un ensemble de services Web sémantiques candidats satisfaisant la requête de recherche en termes de paramètres d'entrée et de sortie.

```
SortRule(match1, match2) {
if match1.output > match2.output then match1>match2
if match1.output = match2.output and match1.input>match2.input
  then match1>match2
if match1.output = match2.output and match1.input=match2.input
  then match1=match2
}
```

Figure 22 - Procédure de classification du résultat de recherche

II.4.2.2) Architecture distribuée

a) Architecture PSWSD

L'architecture PSWSD (P2P-based Semantic Web Service Discovery) [Le-H, 2005] est une architecture pour la découverte de services dans un réseau P2P. Les services Web sont décrits sémantiquement par les concepts de l'ontologie WSMO (Web Service Modeling Ontology) en utilisant les techniques proposées dans [Kaa, 2003]. Chaque description de service contient :

- Une description WSDL du service
- La sémantique des paramètres fonctionnels du service (Entrées, Sorties, Préconditions et Effets)
- des informations de qualité de service.

Dans cette architecture, les fournisseurs de services publient les descriptions de leurs services Web dans divers registres distribués dans le réseau P2P (étape1). Un utilisateur cherchant un service Web, avec certaines qualités de service, peut interroger n'importe quel registre du réseau comme point d'accès (étape2- voir figure 23). Le registre ayant reçu la requête de l'utilisateur va la diriger vers le(s) registre(s) pouvant satisfaire cette requête. Chaque registre du réseau contient un ensemble de modules. Lorsqu'un registre reçoit une requête qui peut la traiter, cette dernière est transmise au module de traitement de requêtes afin de faire ressortir les fonctionnalités du service demandé par l'utilisateur, ces informations sont ensuite envoyées au module matchmaker. Le matchmaker sélectionne les descriptions des services ayant une correspondance sémantique avec la requête de l'utilisateur en terme de fonctionnalités demandées par l'utilisateur et offertes par le service (étape 3). Les résultats sont ensuite transmis à l'utilisateur (étape 4), ce dernier choisit un service pour l'invoquer.

Dans ce système, les utilisateurs peuvent faire des évaluations sur les qualités de service des services Web et les envoyer au registre contenant les descriptions de ces services. Pour vérifier l'authenticité des évaluations des utilisateurs sur les qualités de service des services Web invoqués, les auteurs ont utilisés des agents de surveillance de qualités de service, ces agents ont pour rôle d'évaluer les qualités de service de certains services Web. Leurs évaluations sont ensuite utilisées pour déterminer les utilisateurs malveillants dans leurs rapports d'évaluation de QoS.

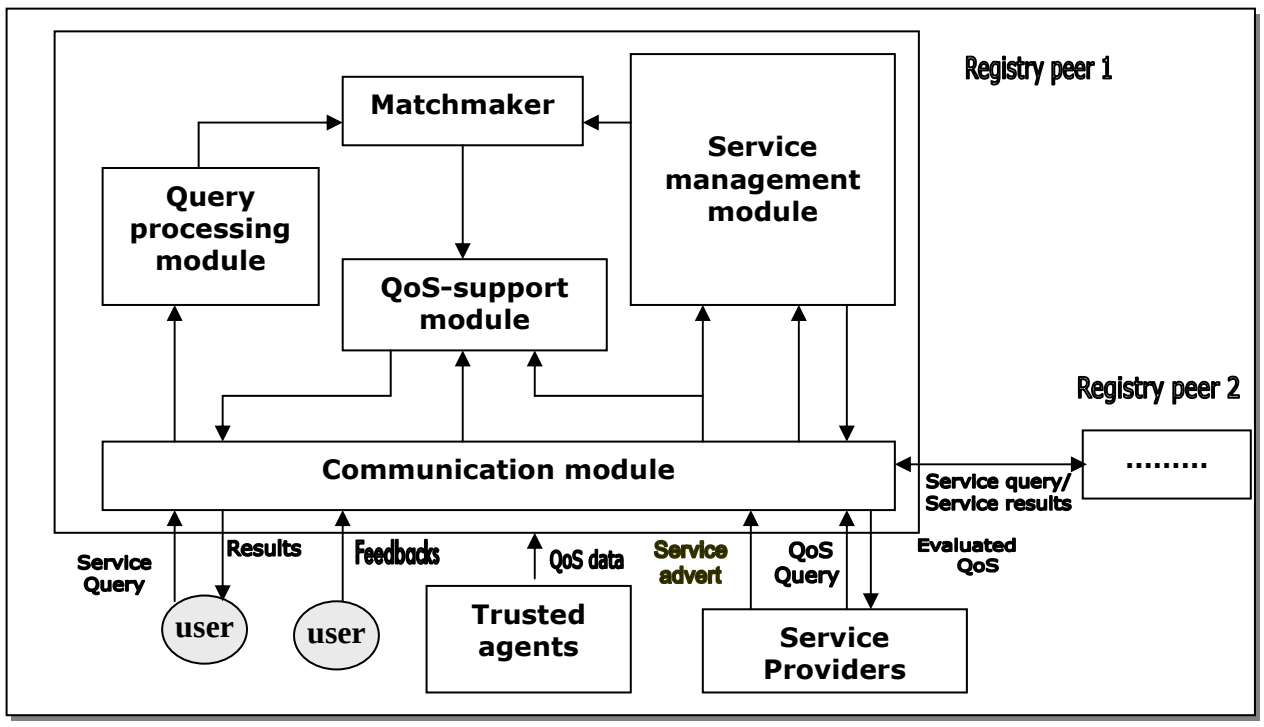

Figure 23 – Structure du régistre [Le-H, 2005]

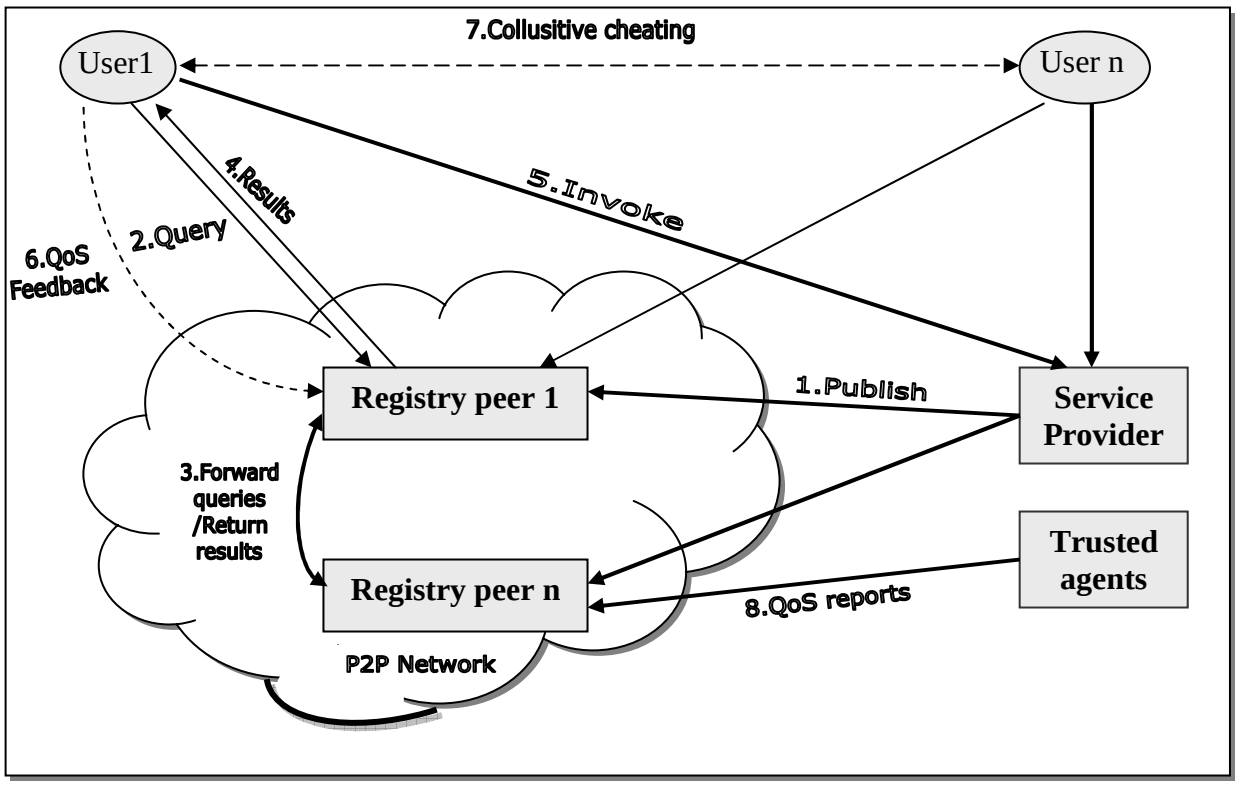

Figure 24 – Découverte sémantique basée P2P [Le-H, 2005]

b) Système Speed-R

Développé à l'université de Géorgie, le système Speed-R [Siva, 2004] vise à connecter l'ensemble des registres UDDI privés (chaque fournisseur de services a son propre registre UDDI) via le réseau P2P.Cependant, au lieu de décrire les services Web avec UDDI tModels, ces

derniers sont décrits en utilisant WSDL. Afin d'apporter une sémantique aux descriptions des services, pour chaque registre du réseau P2P des ontologies spécifiques au domaine du registre sont implémentées à son niveau. La sémantique est apportée aux descriptions des services en faisant un mapping (correspondance) entre les spécifications du service et les concepts des ontologies. Quand un utilisateur veut chercher un service Web, il doit choisir un registre du domaine de recherche, et ce choix est fait en se basant sur l'ontologie de domaine des registres. L'utilisateur peut envoyer une requête simple ou peut utiliser l'agent de recherche sémantique pour exécuter la recherche sémantique de services Web dans le registre. La recherche sémantique des services est basée sur les paramètres fonctionnels du service. De plus, l'interaction entre les utilisateurs et les registres du réseau est réalisée par l'intermédiaire d'agents logiciels. L'implémentation du système est basée JXTA (pour juxtapose) (JXTA est une plate-forme de développement d'applications P2P (poste-à-poste) initié à l'origine par Sun Microsystems) [Ouz, 2004].

II.4.3) Synthèse des approches de découverte

Les services Web constituent un moyen pour les entreprises d'offrir leurs services à travers le Web.

Avec le nombre grandissant de fournisseurs de services, il est nécessaire de disposer de mécanismes pour la découverte de services Web de manière efficace et flexible. La découverte de services Web consiste à localiser les descriptions des services Web décrivant certains critères fonctionnels.

Initialement, les descriptions des services web n'étaient faites qu'au niveau syntaxique, la découverte de services Web était basée sur des techniques issues du domaine de la recherche d'information. Cependant, avec le développement des technologies du web sémantiques et afin d'automatiser la découverte de services Web, de nouvelles approches basées sur la sémantique des descriptions des services ont été proposées.

En général, les approches de découverte dépendent du niveau de représentation, sémantique ou syntaxique, des descriptions des services Web. Cependant, les approches varient selon l'architecture centralisée ou distribuée adoptée. La figure 25 représente une synthèse des principales approches (syntaxiques et sémantiques) étudiées.

	AASDU	OWL-S	SPEED-R	PSWSD
Description des services	WSDL	Ontologie OWL pour service	Mapping entre WSDL et ontologie de domaine	Ontologie WSMO + WSDL
Architecture adoptée	Distribuée	Centralisée	Distribuée	Distribuée
Technique de découverte	TFIDF	Algorithme Matchmaking	Mots clés/ sémantique	Matchmaker
Technologie adoptée	Multi-Agents	SOAP	Multi-Agents sous réseau P2P	Multi-Agents sous réseau P2P

Figure 25 – Synthèse des approches de découverte

Dans la figure 25, on peut remarquer que lorsque l'architecture est distribuée, les approches proposées sont implémentées en utilisant la technologie multi-agents. Cela est bien justifié puisque cette technologie est bien adaptée à la nature distribué du problème. On remarque aussi que les approches sémantiques proposées, tel que OWL-S [Dav, 2004], Speed-R [Siva, 2004] et PSWSD [Le-H, 2005], sont toutes basées sur une même technique qui consiste à calculer le niveau de correspondance sémantique entre les paramètres fonctionnels des services et ceux mentionnés dans la requête de recherche. La différence entre ces approches est le langage d'ontologie choisi, par exemple dans PSWSD une ontologie WSMO est adoptée, tandis que dans une ontologie OWL-S pour service basée OWL est utilisée. De ces deux ontologies, l'ontologie OWL-S est celle qui est sur le point de devenir un standard.

II.5) Conclusion

La découverte de services Web présente un axe de recherche émergent. Diverses approches ont été proposées dans la littérature. L'objectif commun de ces approches était de découvrir des descriptions de services Web décrivant certains critères fonctionnels.

Au début, les descriptions des services Web étaient essentiellement syntaxiques, cependant, avec la fusion des deux technologies services Web et Web sémantique une nouvelle génération de services Web est née connue sous le nom de services Web sémantiques. Avec le développement des services Web sémantiques, de nouvelles approches de découverte ont été proposées.

Les approches de découverte sémantique donnent de bons résultats comparées aux approches syntaxiques. La plupart des approches sémantiques proposées étaient basées sur le calcul du niveau de correspondance sémantique entre les critères fonctionnels définis dans la description sémantique des services et ceux définis dans la requête de recherche telles que (AASDU, OWL-S, SPEED-R et PSWSD).

Dans le chapitre suivant, nous allons approfondir les principaux concepts de la technologie des systèmes multi-agents (SMA).

Chapitre III

Les Systemes multi-agents

III.1) Introduction

L'explosion du réseau Internet et la montée en puissance des capacités des calculateurs offrent aujourd'hui des possibilités énormes quant au passage de relais des humains aux ordinateurs d'une partie de leur effort cognitif. Ainsi, l'arrivée de l'approche orientée objet avait suscité beaucoup d'espérance et d'intérêt puisque plus intuitive : grâce à l'encapsulation et l'héritage, l'on y voyait une représentation qui reflète le mieux la réalité. Seulement, l'approche orientée objet s'est avérée insuffisante : malgré les opportunités qu'ils offrent, les objets restent des structures de données passives et ils restent dépendants en grande partie du système dans lequel ils sont plongés c'est à dire ils n'ont pas une marge significative de contrôle sur leur propre comportement. En effet, le besoin qui se faisait le plus ressentir était d'avoir des entités qui possèdent un grand degré d'autonomie et de flexibilité, qui ont des buts et qui essaient de leur propre arbitre de trouver la manière avec laquelle ils doivent se comporter afin de les atteindre. Cette entité s'appelle un agent.

D'un autre côté, les avancées qu'a connu l'intelligence artificielle avec les systèmes à bases de connaissances (SBC) traditionnels avaient entretenu l'espoir quant à la capacité qu'à un ordinateur pour substituer l'élément humain. Seulement, les SBC qui, certes, se sont avérés très intéressants et ont montré leurs preuves notamment dans le domaine de la médecine et de la reconnaissance du langage naturel, ont déçu notamment dans les applications distribuées et/ou temps réel. En effet, le principal handicap des SBC traditionnel est que le contrôle y est centralisé ce qui limite les capacités et l'efficacité du système. Le besoin d'avoir des systèmes où le contrôle est distribué sont relatifs à un besoin de flexibilité, de facilité de maintenance et d'aptitude à l'émergence d'organisations et capacités non préétablies. Ces systèmes sont appelés Systèmes multi-agents (SMA), ce sont des systèmes dont les composantes principales sont des agents.

Dans ce chapitre, nous présenterons un état de l'art sur les systèmes multi-agents (SMA) et les modes de communication utilisés, en commençant par introduire l'intelligence artificielle distribué (I.A.D) ensuite nous mettons l'accent sur l'évolution de cette discipline vers les SMA.

III.2) l'intelligence artificielle distribuée (I.A.D)

L'évolution des domaines d'application de l'I.A pour recouvrir des domaines complexes et hétérogènes tels que l'aide à la décision, la reconnaissance et la compréhension des formes, la conduite des processus industriels, etc., a montré les limites de l'approche classique de l'I.A qui s'appuie sur une centralisation de l'expertise au sein d'un système unique.

Les travaux menés au début des années 70 sur la concurrence et la distribution ont contribué à la naissance d'une nouvelle discipline : l'Intelligence Artificielle Distribuée (I.A.D) [Fer, 1988].

L'I.A.D a pour but de remédier aux insuffisances de l'approche classique de l'I.A en proposant la distribution de l'expertise sur un groupe d'agents devant être capables de travailler et d'agir dans un environnement commun et résoudre les conflits éventuels. D'ou la naissance de notions nouvelles en I.A, telles que la coopération, la coordination d'actions, la négociation et l'émergence.

L'I.A.D peut alors être définie comme étant la branche de l'I.A qui s'intéresse à la modélisation de comportement "intelligent" (jusque là c'est la définition de l'I.A classique) par la coopération entre un ensemble d'agents [Hun, 1987].

III.3) Thèmes de recherche de L'I.A.D

Après avoir défini l'I.A.D et la différence entre celle-ci et l'I.A classique, nous présentons trois axes fondamentaux dans la recherche en I.A.D :

- *Les systèmes multi-agents (S.M.A)* [Dur, 1990] : il s'agit de faire coopérer un ensemble d'agents dotés d'un comportement intelligent et de coordonner leurs buts et leurs plans d'actions pour la résolution d'un problème. C'est le thème auquel nous nous intéressons le plus.
- *La résolution distribuée des problèmes (R.D.P)* [Les, 1987] : elle s'intéresse à la manière de diviser un problème particulier sur un ensembles d'entités distribuées et coopérantes. Elle s'intéresse aussi à la manière de partager la connaissance du problème et d'en obtenir la solution.
- *L'intelligence artificielle parallèle (I.A.P)* [Da, 1980] [Fehling, 1983]: elle concerne le développement de langages et d'algorithmes parallèles pour l'I.A.D. L'I.A.P vise l'amélioration des performances des systèmes d'intelligence artificielle sans, toutefois, s'intéresser à la nature du raisonnement ou au comportement intelligent d'un groupe d'agents. Cependant, il est vrai que le développement de langages concurrents et d'architectures parallèles peut avoir un impact important sur les systèmes d'I.A.D.

III.4) Problématique de L'I.A.D

Les problèmes que l'I.A.D s'attache à résoudre peuvent être divisés en deux classes : la première concerne les problèmes classiques de l'I.A qui ont pris une nouvelle dimension dans le contexte multi-agents. La seconde regroupe les nouveaux problèmes proprement liés au thème de l'I.A.D. Dans la première classe on peut citer :

- La modélisation de la connaissance et le problème de sa répartition sur les différents agents: comment formuler, décomposer, allouer des problèmes et synthétiser les résultats d'un groupe d'agents?

- Les problèmes de génération de plans d'actions où il faut prendre en considération la présence d'autres agents;
- La gestion des conflits entre les agents (points de vue différents des agents) et le maintien de la cohérence des décisions et des plans d'actions;
- Le problème de la communication : comment permettre la communication et l'interaction entre les agents ? Quel langage et quel protocole faut-il employer ? Une communication dans les univers multi-agents n'est plus une simple tâche d'entrée-sortie, mais doit être modélisée comme un acte pouvant influer sur l'état des autres agents.

Les problèmes de la seconde classe peuvent être divisés en deux sous classes :

- Les problèmes spécifiques au groupe d'agents, qui portent sur l'organisation, l'architecture de l'ensemble des agents et les paradigmes de coopération et d'action;
- Les problèmes liés au comportement d'un agent au sein d'un groupe. On s'intéresse aux capacités sociales d'un agent : le partage des tâches, le partage des ressources, le raisonnement sur les autres agents (pouvoir modéliser leurs connaissances et être en mesure de connaître leurs plans d'actions et de raisonner en fonction de ces plans).

D'autres thèmes de recherche sont présents dans le contexte des systèmes multi-agents, à savoir, le raisonnement temporel, le raisonnement hypothétique, la représentation de la connaissance imprécise, etc.

III.5) Evolution vers les systèmes multi-agents

Contrairement à l'approche centralisée de l'I.A, l'intelligence artificielle distribuée vise la distribution de l'expertise sur un ensemble de composants qui communiquent pour atteindre un objectif global (élaboration d'un diagnostic, résolution d'un problème, etc.). Cette approche nécessite la division du problème en sous-problèmes. Mais cette hypothèse n'est pas toujours réaliste car beaucoup de problèmes ne peuvent être partitionnés de cette manière [Gin, 1987].

Une extension des systèmes d'I.A.D est proposée : les composants doivent être capables de raisonner sur les connaissances et les capacités des autres dans le but d'une coopération effective [Kon, 1991]. Pour ce faire, ils doivent être dotés de capacités de perception et d'action sur leur environnement et doivent posséder une certaine autonomie de comportement, on parle alors d'agents et par conséquent de systèmes multi-agents [Dur, 1987].

Les systèmes multi-agents se caractérisent alors par l'autonomie et l'intelligence des composants impliqués. Toutefois, un agent ne dispose pas d'une vision globale de son environnement.

III.6) Pourquoi distribuer l'I.A ?

Les raisons de la distribution de l'intelligence artificielle peuvent répondre aux investigations suivantes :

III.6.1) Simplifier les applications informatiques distribuées

- Résolution distribuée de problèmes

Toutes les applications relevant de la résolution distribuée de problèmes supposent qu'il est possible d'effectuer une tâche complexe en faisant appel à un ensemble de spécialistes disposant de compétences complémentaires et donc, lorsque c'est l'expertise ou le mode de résolution qui sont distribués, sans que le domaine le soit.

Lorsque l'expertise est en effet si vaste et complexe qu'elle ne peut appartenir qu'à une seule personne, il faut faire appel à plusieurs spécialistes qui doivent travailler ensemble à la poursuite d'un objectif commun. Ces spécialistes coopèrent entre eux pour résoudre un problème général, tel qu'un diagnostic médical, la conception d'un produit industriel, l'acquisition de connaissances, etc.

- Résolution de problèmes distribués

Si le domaine est lui aussi distribué, on parle alors de résolution (distribuée) de problèmes distribués. Il s'agit essentiellement d'applications telles que l'analyse, l'identification, le diagnostic et la commande de systèmes physiquement répartis, pour lesquelles il est difficile d'avoir une vision totalement centralisée. Par exemple, s'il s'agit de contrôler un réseau de communication ou d'énergie, le domaine, qui est représenté par le réseau lui-même, constitue un système réparti qu'il s'agit de surveiller, voire de superviser, en décentralisant au maximum les tâches de surveillance au sein des nœuds du réseau.

- Résolution par coordination

Les agents peuvent aussi servir d'une manière beaucoup plus élémentaire à résoudre des problèmes au sens classique du terme, c'est-à-dire à tenter de trouver une solution à quelque chose dont l'énoncé est bien posé et dont l'ensemble des informations est entièrement disponible, comme, par exemple, trouver une affectation de tâches pour une machine-outil ou définir un emploi du temps pour un collège.

Dans ce cas, le domaine n'est pas distribué et l'expertise ne l'est pas non plus. Et pourtant l'approche multi-agents peut apporter un mode de raisonnement nouveau en décomposant le problème de manière totalement différente. Par exemple, s'il s'agit d'assembler des pièces mécaniques, on peut considérer que les pièces sont des agents qui

doivent satisfaire des buts précis donnés par le plan imposé par le concepteur et que les liaisons sont des contraintes que les agents doivent respecter.

III.6.2) Un paradigme de recherche en intelligence artificielle

C'est une importante motivation portée à la recherche sur les agents : convergence en I.A qui bénéficie de l'orientation vers l'I.A.D suite à l'échec de l'approche centralisée de l'I.A classique**.**

III.7) Les systèmes multi-agents (S.M.A)

III.7.1) Qu'est-ce qu'un agent ?

Le concept d'agent a été l'objet d'études pour plusieurs décennies dans différentes disciplines. Il a été non seulement utilisé dans les systèmes à base de connaissances, la robotique, le langage naturel et d'autres domaines de l'intelligence artificielle, mais aussi dans des disciplines comme la philosophie et la psychologie. Aujourd'hui, avec l'avènement de nouvelles technologies et l'expansion de l'Internet, ce concept est encore associé à plusieurs nouvelles applications comme agent ressource, agent courtier, assistant personnel, agent interface, agent ontologique, etc. Dans la littérature, on trouve une multitude de définitions d'agents. Elles se ressemblent toutes, mais diffèrentes selon le type d'application pour laquelle est conçu l'agent. Nous trouvons dans [Fik, 1971] une discussion sur les différentes définitions attribués aux agents ainsi que la différence entre un agent et un programme classique. À titre d'exemple, voici l'une des premières définitions de l'agent dûe à Ferber :

Définition 1 [Ferber, 1995] ***:***

Un agent est une entité autonome, réelle ou abstraite, qui est capable d'agir sur elle-même et sur son environnement, qui, dans un univers multi-agent, peut communiquer avec d'autres agents, et dont le comportement est la conséquence de ses observations, de ses connaissances et des interactions avec les autres agents.

Il ressort de cette définition des propriétés clés comme l'autonomie, l'action, la perception et la communication. D'autres propriétés peuvent être attribuées aux agents. Nous citons en particulier la réactivité, la rationalité, l'engagement et l'intention.

Récemment, Jennings, Sycara et Wooldridge ont proposé la définition suivante pour un agent :

Définition 2 [Wool, 1995] *:*

Un agent est un système informatique qui est situé dans un environnement et qui est capable d'appliquer des actions autonomes dans le but de satisfaire ses buts.

situé : l'agent est capable d'agir sur son environnement à partir des entrées sensorielles qu'il reçoit de ce même environnement. Exemples : systèmes de contrôle de processus, systèmes embarqués, etc.

autonome : l'agent est capable d'agir sans l'intervention d'un tiers (humain ou agent) et contrôle ses propres actions ainsi que son état interne ;

flexible : l'agent dans ce cas est :

-capable de répondre à temps : l'agent doit être capable de percevoir son environnement et d'élaborer une réponse dans les temps requis,

-proactif : l'agent doit exhiber un comportement proactif et opportuniste, tout en étant capable de prendre l'initiative au bon moment,

-social : l'agent doit être capable d'interagir avec les autres agents (logiciels et humains) quand la situation l'exige afin d'accomplir ses tâches ou aider ces agents à accomplir les leurs.

Toutes ces définitions se basent sur des notions semblables que sont : un agent est autonome, plongé dans un environnement et qui a des objectifs à atteindre.

III.7.2) Les caractéristiques multidimensionnelles d'un agent

Plusieurs propriétés sont reliées aux agents mais un agent ne possède pas forcément toutes ces propriétés. Celles qui sont le plus énoncées dans la littérature sont :

. *La nature :* agents physiques ou virtuels.

. *L'autonomie :* l'agent est plus ou moins indépendant de l'utilisateur, des autres agents, et des ressources (U.C, mémoire, etc.…)

. *L'environnement :* c'est l'espace dans lequel l'agent va agir ; celui-ci peut se réduire au réseau constitué par l'ensemble des agents.

. *La capacité représentationnelle :* l'agent peut avoir une vision très locale de son environnement, mais il peut aussi avoir une représentation plus large de cet environnement et notamment des agents qui l'entourent (accointances).

. *L'objectif (dimension téléonomique) :* l'agent peut poursuivre le but global de système, peut satisfaire des objectifs propres ou même se comporter dans la perspective de s'absoudre une fonction de survie.

. *La perception :* de l'environnement par l'agent.

. *La communication :* l'agent aura plus ou moins de capacités à communiquer avec les autres agents.

. *Le raisonnement :* l'agent peut être lié à un système expert ou à d'autres mécanismes de raisonnements plus ou moins complexes.

. *La quantité de ses congénères :* le système peut contenir de quelques-uns un à plusieurs milliers d'agents.

. *Le contrôle :* il peut être totalement distribué entre les agents mais peut être voué à une certaine classe d'agents comme les agents « facilitateurs ».

. *L'anticipation :* l'agent peut plus ou moins avoir les capacités d'anticiper les événements futurs.

. *La granularité ou complexité :* l'agent peut être très simple comme un neurone mais aussi plus complexe.

. *La contribution :* l'agent participe plus ou moins à la résolution du problème ou à l'activité globale du système.

. *L'efficacité :* l'agent et sa rapidité d'exécution, d'intervention.

. *La bienveillance :* l'agent a plus ou moins le devoir d'aider ses congénères plutôt que de s'opposer à eux.

. *Intentionnalité* **:** Un agent intentionnel est un agent guidé par ses buts. Une intention [Sear, 1990] est la déclaration explicite des buts et des moyens d'y parvenir. Elle exprime donc la volonté d'un agent d'atteindre un but ou d'effectuer une action.

. *Rationalité :* Un agent rationnel est un agent qui suit le principe suivant (dit principe de rationalité) [New, 1982] :

« Si un agent sait qu'une de ses actions lui permet d'atteindre un de ses buts, il la sélectionne ».

Les agents rationnels disposent de critères d'évaluation de leurs actions, et sélectionnent selon ces critères les meilleures actions qui leur permettent d'atteindre le but. De tels agents sont capables de justifier leurs décisions.

. *Adaptabilité :* un agent adaptatif est un agent capable de contrôler ses aptitudes (communicationnelles, comportementales, etc.) selon l'agent avec qui il interagit. Un agent adaptatif est un agent d'un haut niveau de flexibilité.

. *Engagement :* La notion d'engagement [Bond, 1990] est l'une des qualités essentielles des agents coopératifs. Un agent coopératif planifie ses actions par coordination et négociation avec les autres agents. En construisant un plan pour atteindre un but, l'agent se donne les moyens d'y parvenir et donc s'engage à accomplir les actions qui satisfont ce but ; l'agent croit qu'il a élaboré, ce qui le conduit à agir en conséquence.

. *Intelligence :* On appelle agent intelligent un agent cognitif, rationnel, intentionnel et adaptatif.

III.7.3) Différentes catégories d'agents

On peut classer les agents selon plusieurs critères mais par la suite on tiendra compte d'un seul mode de classification qui est la granularité et qui détermine le processus de décision de l'agent. La taxonomie des agents se fait selon :

- *La granularité : agent réactif ou agent cognitif.*
- *La mobilité : agent mobile (ou nomade) et agent stationnaire.*
- *La fonctionnalité : agent logiciel, agent d'information, agent de commerce, agent assistant (secrétaire virtuelle).*

Dans la littérature, on distingue deux grandes catégories d'agents et ceci en se basant sur leur modèle de décision : les agents réactifs et les agents cognitifs. On peut aussi parler d'agent hybride qui se situe entre ces deux extrêmes et qui regroupe les propriétés des agents réactifs et des agents cognitifs. Chaque type d'agent est différent des autres types sur le plan comportemental, structural et interactif. Ceci va être plus amplement détaillé dans ce qui suit.

a. Les Agents réactifs

Les agents réactifs n'ont pas une représentation explicite de leur environnement et ne tiennent pas compte de leurs actions passées. Ils ont un comportement de type stimulus / réponse (perception/action). Les agents réagissent ainsi à des changements survenant dans leur environnement à travers des actions prédéfinies auparavant. La figure 26 montre bien le mécanisme évoqué :

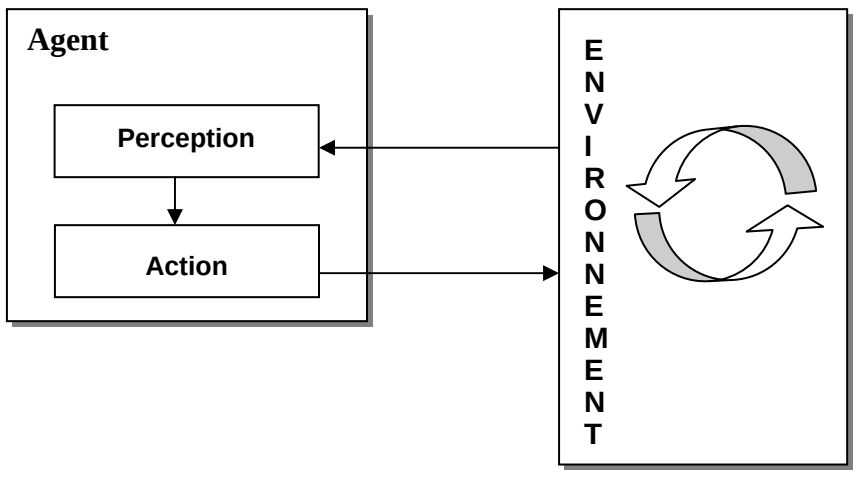

Figure 26 - Modèle d'un agent réactif

Un système multi-agents constitué uniquement d'agents réactifs requiert un nombre élevé d'agents pour pouvoir parler d'un système intelligent. En effet, l'interaction entre ces différents

agents permet l'émergence d'une intelligence bien que le comportement d'un seul agent tende vers la simplicité. [Dro, 1993] présente un exemple d'agents réactifs qui simulent le comportement d'une société de fourmis au sein du projet MANTA (Modeling an ANThil Activity). Il montre que bien qu'une fourmi ne peut effectuer qu'une action simple, prédéfinie et sans aucune intelligence, l'ensemble des fourmis peut effectuer des actions complexes et évoluées, et ainsi faire émerger un comportement intelligent.

b. Les Agents cognitifs

Contrairement aux agents réactifs, les agents cognitifs (ou délibératifs ou intentionnels) ont une représentation plus globale de leur environnement (de soi, du groupe, des autres, de la tâche), ils possèdent une faculté de raisonnement et de choix de l'action adéquate selon les changements perçus afin d'atteindre un but précis et d'une manière optimale (voir figure 27). Ils ont aussi une faculté d'apprentissage qui permet de les qualifier d'agent intelligent. Ainsi un agent cognitif possède des buts, des connaissances et des désirs et peut s'adapter et réagir à des situations nouvelles sans une intervention humaine ou extérieure.

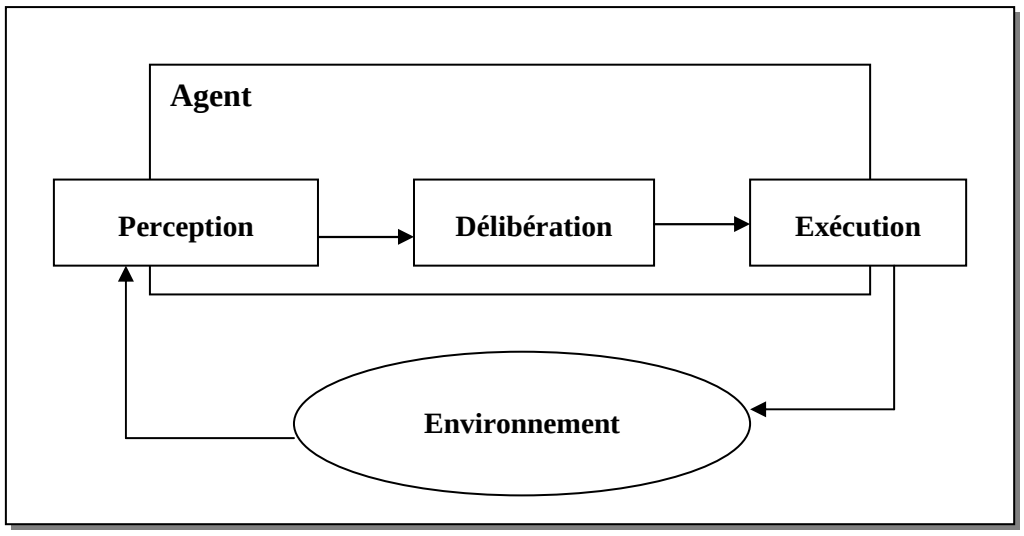

Figure 27 - Cycle Perception / Délibération/ Action d'un agent cognitif

De ce fait, on peut dire qu'un agent cognitif possède un état mental qui se base sur les notions de connaissances, de la croyance, de l'intentionnalité et de l'engagement. Tous ces concepts vont être réunis à l'intérieur de l'agent selon l'architecture qui va être adoptée afin de définir une structure agençant les composants de ce dernier.

La figure 28 résume les différences entre les deux approches : cognitive et réactive.

Système d'agents cognitifs	Système d'agents réactifs
Représentation explicite de l'environnement	Pas de représentation explicite
Peut tenir compte de son passé	Pas de mémoire de son historique
Agent complexe	Fonctionnement stimulus/action
Petit nombre d'agents	Grand nombre d'agents

Figure 28 - Différences entre les deux approches

III.7.4) Architecture d'agent

1) Structure interne

La figure 29 décrit l'architecture globale d'un agent cognitif. C'est une synthèse des architectures d'agents décrites dans la littérature. On distingue essentiellement : le savoir-faire, les croyances, la connaissance de contrôle, l'expertise de l'agent et la connaissance de communication.

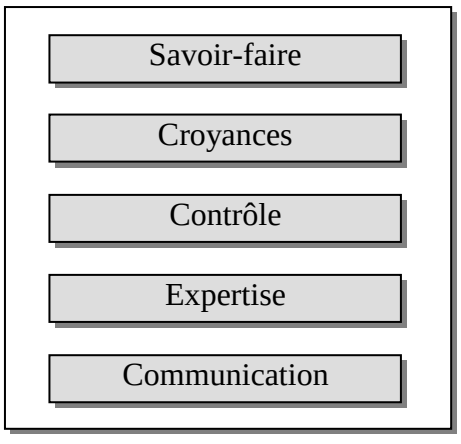

Figure 29 - structure d'un agent

- *Savoir-faire*: le savoir-faire est une interface permettant la déclaration des connaissances et des compétences de l'agent. Il permet la sélection des agents à solliciter pour une tâche donnée. Il n'est pas nécessaire mais il est très utile pour améliorer les performances du système, quelque soit le mode de coopération utilisé.
- *Croyances* : dans un univers multi-agents, chaque agent possède des connaissances sur lui-même et sur les autres. Ces connaissances ne sont pas nécessairement objectives, on parle alors de croyances d'un agent. Les logiques des connaissances et des croyances [Hal, 1984] s'intéressent à la formalisation de telles connaissances considérées comme

incertaines. Cette formalisation est à la base de la conception de tout système multi-agents puisqu'elle détermine en grande partie le comportement "intelligent" des agents.

- *Contrôle* : la connaissance de contrôle dans un agent est représentée par les buts, les intentions, les plans et les tâches qu'il possède.
- *Expertise* : c'est la connaissance sur la résolution de problème. Pour un système expert utilisant le formalisme de règle, par exemple, cette connaissance correspond à sa base de règles.
- *Communication* : l'agent doit posséder un protocole de communication lui permettant d'interagir avec les autres agents pour une bonne coopération et une bonne coordination d'actions. D'autres connaissances de communication peuvent être disponibles, par exemple les connaissances sur les réseaux de communication (tous les agents ne sont pas forcément en liaison directe).

2) Fonctionnement

Dans ce paragraphe, nous présentons l'architecture fonctionnelle d'un agent cognitif et nous détaillons son fonctionnement. Cette architecture est illustrée par la figure 30. Dans cette figure, les ellipses représentent les processus mis en oeuvre lors du fonctionnement de l'agent.

Les agents sont immergés dans un environnement dans lequel et avec lequel ils interagissent. D'où leur structure autour de trois fonctions principales : *percevoir*, *décider* et *agir*. Parmi les sous-fonctions importantes d'un agent on peut citer : la détection de conflits, la révision des croyances, la coopération (négociation, coordination), l'apprentissage, etc. Toutes les fonctionnalités ne sont pas représentées dans la figure 30.

Un agent a la possibilité d'acquérir des connaissances sur l'environnement externe (perception). Il a aussi des capacités d'interaction avec les autres agents (communication, négociation). En fonction des connaissances et croyances dont il dispose et des buts qu'il se fixe suite à une perception ou à une interaction avec le monde extérieur, l'agent doit élaborer un plan d'action. Pour cela, il doit décider quel serait le but à retenir et à satisfaire en premier, ensuite planifier en fonction de ce but et passer à l'exécution. Ces deux derniers processus doivent être alternés du fait du caractère dynamique des environnements multi-agents.

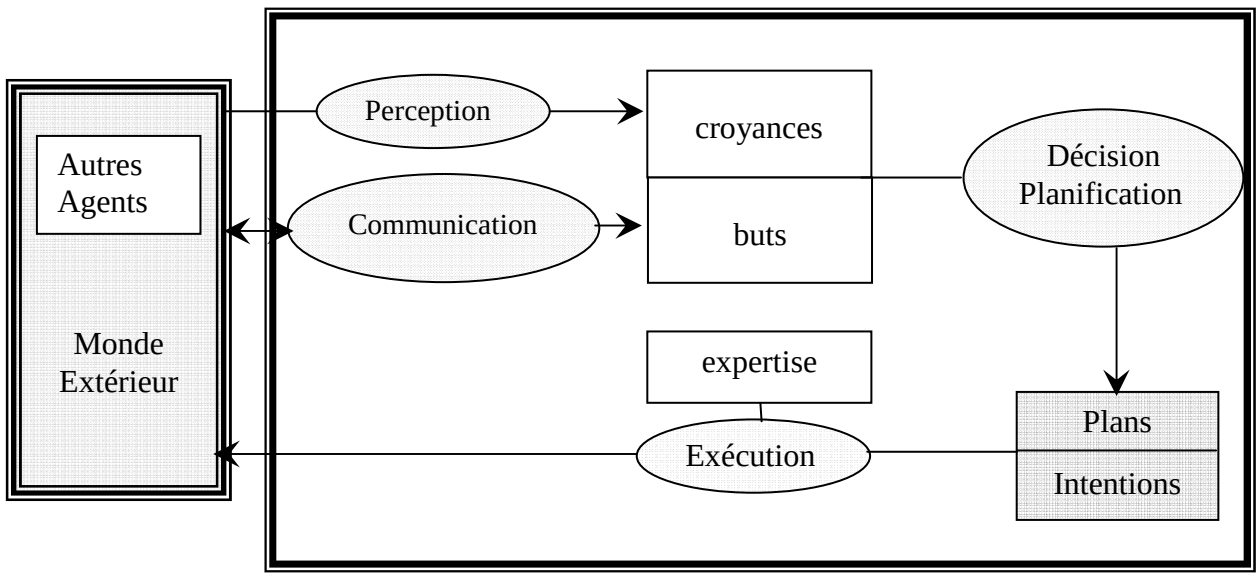

Figure 30 - Fonctionnement d'un agent

- *Perception*

  Les connaissances d'un agent ont plusieurs origines :

  - le savoir initial de l'agent ;
  - la perception de soi (perception proprioceptive) et du monde (perception extéroceptive);
  - la communication avec les autres agents.

- *Prise de décision*

  Durant son exécution, un agent se fixe un certain nombre de buts, suite à ses observations et ses interactions avec le monde (perception, communication négociation).

  Il se trouve donc confronté aux problèmes de la sélection du but à satisfaire en premier et pour chaque but, de l'action qui permet de l'atteindre. Face à de telles situations, l'agent analyse les différentes alternatives en terme d'utilité (quel avantage l'agent pourrait-il en retirer ?) et d'incertitude (Quelle chance a l'action d'être effectuée en fournissant le résultat attendu ?).

  La prise de décision est une des caractéristiques des agents rationnels, l'agent tiendra compte de ses croyances pour faire son choix.

- *Planification*

  La planification dans les systèmes de l'I.A classique repose sur l'hypothèse d'un univers statique (seules les actions des planificateurs sont prises en compte), cet aspect est incompatible avec l'approche multi-agents [Fer, 1988].

  L'I.A.D, en revanche, offre des possibilités de négociations autorisant une gestion locale des conflits et une planification dynamique. En effet, du fait de l'intervention

d'autres agents, un plan peut être remis en cause. L'agent doit pour cela, alterner planification et exécution et réviser des parties de son plan.

La planification dans les SMA est une planification distribuée : il n'existe pas de plan global et chaque agent construit son propre plan en coordonnant avec les autres (cas d'agents coopératifs). Certains systèmes de l'I.A.D présentent une planification centralisée, un organe central se chargera de la gestion des conflits et de l'élaboration d'un plan global**.**

III.7.5) Définition d'un S.M.A

Un système multi-agents est un système distribué composé d'un ensemble d'agents. Contrairement aux systèmes d'IA, qui simulent dans une certaine mesure les capacités du raisonnement humain, les SMA sont conçus et implantés idéalement comme un ensemble d'agents interagissants, le plus souvent, selon des modes de coopération, de concurrence ou de coexistence [Cha, 1994].

Définition [Fer, 1995] *On appelle système multi-agents (ou SMA), un système composé des éléments suivants:*

1. *Un environnement E, c'est-à-dire un espace disposant généralement d'une métrique.*
2. *Un ensemble d'objets O. Ces objets sont situés, c'est-à-dire que, pour tout objet, il est possible, à un moment donné, d'associer une position dans E. Ces objets sont passifs, c'est-à-dire qu'ils peuvent être perçus, créés, détruits et modifiés par les agents.*
3. *Un ensemble A d'agents, qui sont des objets particuliers ($A \subseteq O$), lesquels représentent les entités actives du système.*
4. *Un ensemble de relations R qui unissent des objets (et donc des agents) entre eux.*
5. *Un ensemble d'opérations Op permettant aux agents de A de percevoir, produire, consommer, transformer et manipuler des objets de O.*
6. *Des opérateurs chargés de représenter l'application de ces opérations et la réaction du monde à cette tentative de modification, que l'on appellera les lois de l'univers.*

La figure 31 donne une illustration de la notion de système multi-agents.

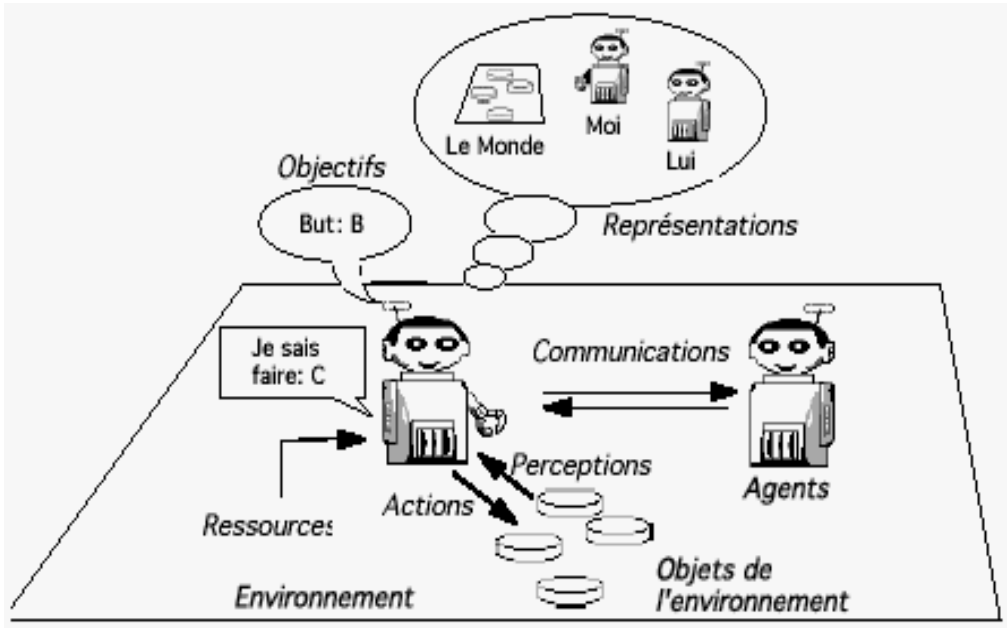

Figure 31 - Représentation d'un agent en interaction avec son environnement et les autres agents [Fer, 1995]

Un SMA est généralement caractérisé par :

– chaque agent a des informations ou des capacités de résolution de problèmes limitées, ainsi chaque agent a un point de vue partiel,

– il n'y a aucun contrôle global du système multi-agents,

– les donnés sont décentralisées,

– le calcul est asynchrone.

III.7.6) Les cinq problématiques des S.M.A

On peut relever cinq problématiques principales lors de la création de systèmes multi-agents (SMA) [WIK, 2007] :

- D'abord, la problématique de l'action : comment un ensemble d'agents peut agir de manière simultanée dans un environnement partagé, et comment cet environnement interagit en retour avec les agents ? Les questions sous-jacentes sont entre-autres celles de la représentation de l'environnement par les agents, de la collaboration entre agents, de la planification multi-agents.
- Ensuite la problématique de l'agent et de sa relation au monde, qui est représenté par le modèle cognitif dont dispose l'agent. L'individu d'une société multi-agent doit être capable de mettre en œuvre les actions qui répondent au mieux à ses objectifs. Cette capacité à la décision est liée à un "état mental" qui reflète les perceptions, les représentations, les croyances et un certain nombre de paramètres "psychiques" (désirs, tendances...) de l'agent. La problématique de l'individu et de sa relation au monde couvre aussi la notion d'engagement de l'agent vis-à-vis d'un agent tiers.

- Les systèmes multi-agents passent aussi par l'étude de la nature des interactions, comme source de possibilités d'une part et de contraintes d'autre part. La problématique de l'interaction s'intéresse aux moyens de l'interaction (quel langage ? quel support ?), et à l'analyse et la conception des formes d'interactions entre agents. Les notions de collaboration et coopération (en prenant coopération comme collaboration + coordination d'actions + résolution de conflits) sont ici centrales.
- On peut évoquer ensuite la problématique de l'adaptation en terme d'adaptation individuelle ou apprentissage d'une part et d'adaptation collective ou évolution d'autre part.
- Enfin, il reste la question de la réalisation effective et de l'implémentation des SMA, en structurant notamment les langages de programmation en plusieurs types allant du langage de type L5, ou langage de formalisation et de spécification, au langage de type L1 qui est le langage d'implémentation effective. Entre les deux, on retrouve le langage de communication entre agents, de description des lois de l'environnement et de représentation des connaissances.

III.7.7) Avantages et objectifs des S.M.A

Les S.M.A sont des systèmes idéaux pour représenter des problèmes qui possèdent de multiples méthodes de résolution, de multiples perspectives. L'approche S.M.A est justifiée par les propriétés :

- La modularité.
- La vitesse, avec le parallélisme.
- Fiabilité, due à la redondance.
- Traitement symbolique au niveau de connaissances.
- La réutilisation et la portabilité.
- L'intervention des schémas d'interaction sophistiqués (coopération, coordination, négociation).

Les deux objectifs majeurs de recherche dans le domaine des S.M.A sont :

- l'analyse théorique et expérimentale des mécanismes.
- La résolution de programmes distribués.
- Pour résoudre deux genres de problèmes :
  - ♦ Modéliser, expliquer et simuler des phénomènes naturels.
  - ♦ La réalisation de systèmes informatiques complexes.

III.7.8) Différences entre S.M.A et Système orienté objets

Un S.M.A offre un puissant répertoire d'outils, de techniques, et de métaphores qui y ont le potentiel d'améliorer considérablement les systèmes logiciels. La technologie agent représente un nouveau paradigme de programmation similaire à la programmation orientée objet et qui d'ailleurs pourrait s'intituler programmation orientée agent.

Il convient de ne pas confondre "agent" et "objet". Tout comme les agents, les objets encapsulent leur état interne (leurs données). Ils peuvent également poser des actions sur cet état par le biais de leurs méthodes et ils communiquent en s'envoyant des messages. À ce niveau, ils différent des agents par leur degré d'autonomie. En effet, une méthode doit être invoquée par un autre objet pour pouvoir accomplir ses effets. Un agent, quant à lui, recevra une requête et décidera de son propre gré s'il doit poser ou non une action.

Une seconde différence provient du caractère flexible du comportement d'un agent. Bien que certains diront qu'il est possible de bâtir un programme orienté-objet qui intègre ces caractéristiques, on doit également voir que le modèle standard d'un objet ne dit rien à propos de ces types de comportements.

La troisième et dernière différence provient du fait qu'on considère un agent comme étant lui-même une source de contrôle au sein du système tandis que dans un système orienté objet, on n'a qu'une seule source de contrôle.

III.8) Organisation multi-agents

III.8.1) Qu'est ce qu'une organisation

Nous reprenons la définition proposée par E.Morin d'une organisation :

> **Définition** [Mor, 1977] *:*
>
> *Une organisation peut être définie comme un agencement de relations entre composants ou individus qui produit une unité, ou système, dotée de qualités inconnues au niveau des composants ou individus. L'organisation lie de façon relationnelle des éléments ou événements ou individus divers qui dès lors deviennent les composants d'un tout. Elle assure solidarité et solidité relative, donc assure au système une certaine possibilité de durée en dépit de perturbations aléatoires.*

Cependant, spécifiquement au domaine des SMA, J.Ferber [Fer, 1995] différencie entre les termes organisation et structure organisationnelle : la structure organisationnelle est ce qui caractérise, sur un plan abstrait, une classe d'organisation. A l'inverse, l'organisation "concrète" est une instanciation (au sens des langages objets) possible d'une structure organisationnelle, une

réalisation comprenant l'ensemble des entités qui participent effectivement à l'organisation ainsi que l'ensemble des liens qui associent ces gens à un moment donné. Alors, une structure organisationnelle est définie par la double donnée :

1. d'un ensemble de classes d'agents caractérisées par les rôles affectés aux agents.
2. de l'ensemble de relations abstraites existent entre ces rôles.

III.8.2) Niveaux d'organisation

En reprenant la classification proposée par G. Gurvitch [Gur, 1963], on peut distinguer trois niveaux d'organisation dans les systèmes multi-agents :

1) *Le niveau micro-social*, où l'on s'intéresse essentiellement aux interactions entre agents et aux différentes formes de liaison qui existent entre deux ou un petit nombre d'agents. C'est à ce niveau que la plupart des études ont été généralement entreprises en intelligence artificielle distribuée.

2) *Le niveau des groupes*, où l'on s'intéresse aux structures intermédiaires qui interviennent dans la composition d'une organisation plus complète. A ce niveau, on étudie les différenciations des rôles et des activités des agents, l'émergence des structures organisatrices entre agents et constitution d'organisations.

3) *Le niveau des sociétés globales* (ou populations) où l'intérêt se porte surtout sur la dynamique d'un grand nombre d'agents, ainsi que la structure générale du système et son évolution. Les recherches se situent alors dans le cadre de la Vie Artificielle.

III.8.3) Comment étudier une organisation ?

Devant l'ensemble des organisations possibles, la demande d'analyse doit être structurée. C'est pourquoi nous distinguons trois volets [Fer, 1995] :

1- *L'analyse fonctionnelle* décrit les fonctions d'une organisation multi-agents dans ses différentes dimensions.

2- *L'analyse structurelle* distingue les différentes formes d'organisation possibles et identifie quelques paramètres structuraux essentiels.

3- *Les paramètres de concrétisation* traitent du passage d'une structure organisationnelle à une organisation concrète et posent la difficile question de la réalisation effective d'une organisation multi-agents.

III.9) Coopération

III.9.1) Définition

La coopération est nécessaire quand un agent ne peut pas atteindre ses buts sans l'aide des autres agents. Cette situation est fréquente même chez des espèces primitives, édification d'une fourmilière par exemple. Souvent les buts nécessitant la coopération sont des buts sociaux, ils assurent la survie du groupe ou de l'espèce. Quelquefois, ce sont des buts individuels, un agent qui en aide un autre peut attendre une aide en retour ou se faire payer son travail. Un agent peut avoir besoin d'un autre agent parce que cet agent a des compétences qu'il n'a pas, ou parce qu'il faut être plusieurs pour réaliser la tâche (pousser un objet trop lourd). La coopération n'est pas forcément consciente, elle peut résulter d'un comportement automatique, comme la construction des ruches ou des termitières.

J.Ferber propose la définition suivante :

Définition [Fer, 1995] :

On dira que plusieurs agents coopèrent, ou encore qu'ils sont dans une situation de coopération, si l'une des deux conditions est vérifiée :

1. L'ajout d'un nouvel agent permet d'accroître différentiellement les performances du groupe.
2. L'action des agents sert à éviter ou à résoudre des conflits potentiels ou actuels.

Il suffit alors que l'une de ces deux conditions soit vérifiée pour que l'on puisse juger la situation étudiée comme coopérative.

III.9.2) Modèles de Coopération

Quelque soit l'organisation d'une société d'agents, un agent peut coopérer suivant les modèles suivants [Vail, 1987] :

- *Coopération par partage de tâches et de résultats*, avec la possibilité de prendre en compte localement les plans des autres.
- *Commande* : un agent supérieur A décompose le problème en sous problèmes qu'il répartit entre les autres agents X_i. Ceux-ci le résolvent et renvoient les solutions partielles à A.
- *Appel d'offre* : A décompose le problème en des sous-problèmes dont il diffuse la liste. Chaque agent X_i qui le souhaite envoie une offre : A choisit parmi celles-ci et distribue les sous-problèmes. Le système travaille ensuite en mode commande.

- *Compétition* : dans le mode compétition, A décompose et diffuse la liste des sous-problèmes comme dans le mode appel d'offre, chaque agent X_i résout un ou plusieurs sous-problèmes et envoie les résultats correspondants à A qui à son tour fait le tri.

III.10) Résolution de conflits

Les agents coopératifs ont besoin d'éviter autant que possible les situations conflictuelles pour résoudre un problème. Pour ce faire, ils peuvent être amenés à coordonner leurs activités et négocier leurs actions pour arriver à une solution.

III.10.1) Coordination

Les agents travaillent sur des problèmes dont les solutions sont utiles pour les autres agents. Leur travail doit donc être coordonné dans le temps. La coordination [Dur, 1987] permet aux agents de considérer toutes les tâches et de ne pas dupliquer le travail. La coordination des actions est liée à la planification et à la résolution des conflits, car c'est à ce niveau qu'on tient compte des actions (plans) des autres agents. Dans les systèmes d'IAD, la coordination des actions des agents peut s'organiser suivant deux schémas principaux [Ferber, 1990] : une coordination au moyen d'un système capable de déterminer et de planifier (globalement) les actions des différents agents ou à l'inverse, on décide de donner une totale autonomie aux agents qui, à leur tour, identifient les conflits pour les résoudre localement.

On peut distinguer deux types de coordination :

- la coordination due à la gêne (problème de navigation : les agents doivent coordonner leurs plans de navigation pour s'éviter mutuellement),
- la coordination due à l'aide (manutention : dans un environnement multi-robots, les agents doivent synchroniser leurs actions pour pouvoir agir efficacement et transporter un objet [Peg, 1988]).

III.10.2) Négociation

Les activités des agents dans un système distribué sont souvent interdépendantes et entraînent des conflits. Pour les résoudre, il faut considérer les points de vue des agents, les négocier, et utiliser des mécanismes de décision concernant les buts sur lesquels le système doit se focaliser [Che, 1992]. La négociation est caractérisée par :

- Un nombre faible d'agents impliqués dans le processus.
- Un protocole minimal d'actions : proposer, évaluer, modifier et accepter ou refuser une solution.

Le problème de la négociation ne consiste pas forcément à trouver un compromis mais peut s'étendre à la modification des croyances d'autres agents pour faire prévaloir un point de vue.

III.11) La communication dans les S.M.A

III.11.1) Définition

La communication est l'ensemble des processus physiques et psychologiques par lesquels s'effectue l'échange d'information entre un ou plusieurs agents (émetteurs) avec un ou plusieurs agents (récepteurs), en vue d'atteindre certains objectifs.

III.11.2) Les modes de communication

Il existe deux principaux modes de communication : la communication par envoi de messages et la communication par partage d'informations.

1) Communication par envoi de messages

Les agents sont en liaison directe et envoient leurs messages directement et explicitement au destinataire. La seule contrainte est la connaissance de l'agent destinataire : " Si un agent A connaît l'agent B alors il peut entrer en communication avec lui".

Les systèmes fondés sur la communication par envoi de messages relèvent d'une distribution totale à la fois de la connaissance, des résultats et des méthodes utilisées pour la résolution du problème (voir figure 32)

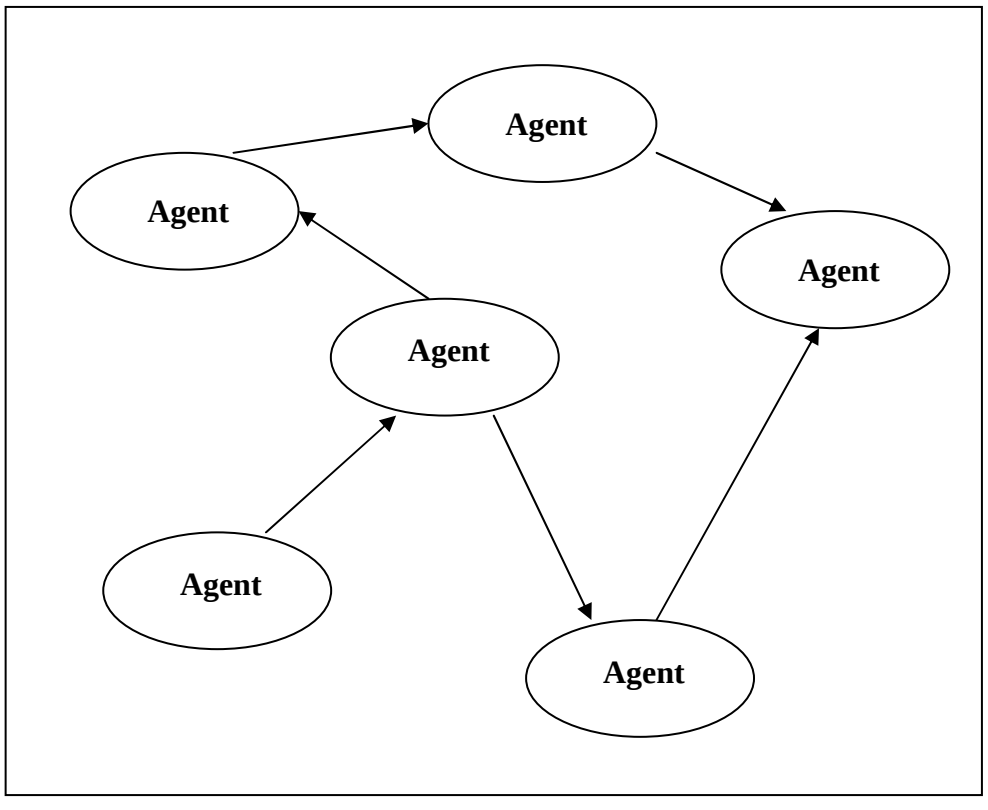

Figure 32 - Communication par envoi de message

2) Communication par partage d'informations

Les composants ne sont pas en liaison directe mais communiquent via une structure de données partagée, où on trouve les connaissances relatives à la résolution (état courant du problème) qui évolue durant le processus d'exécution. Cette manière de communiquer est l'une des plus utilisées dans la conception des systèmes multi-experts. L'exemple parfait d'utilisation de ce mode de communication est l'architecture de blackboard, on parle plutôt de sources de connaissances que d'agents. Ce mode de communication n'existe pas dans les systèmes multi-agents où l'on dispose que d'une vision partielle du système alors que la communication par partage d'informations suppose l'existence d'une base partagée sur laquelle les composants viennent lire et écrire (voir figure 33).

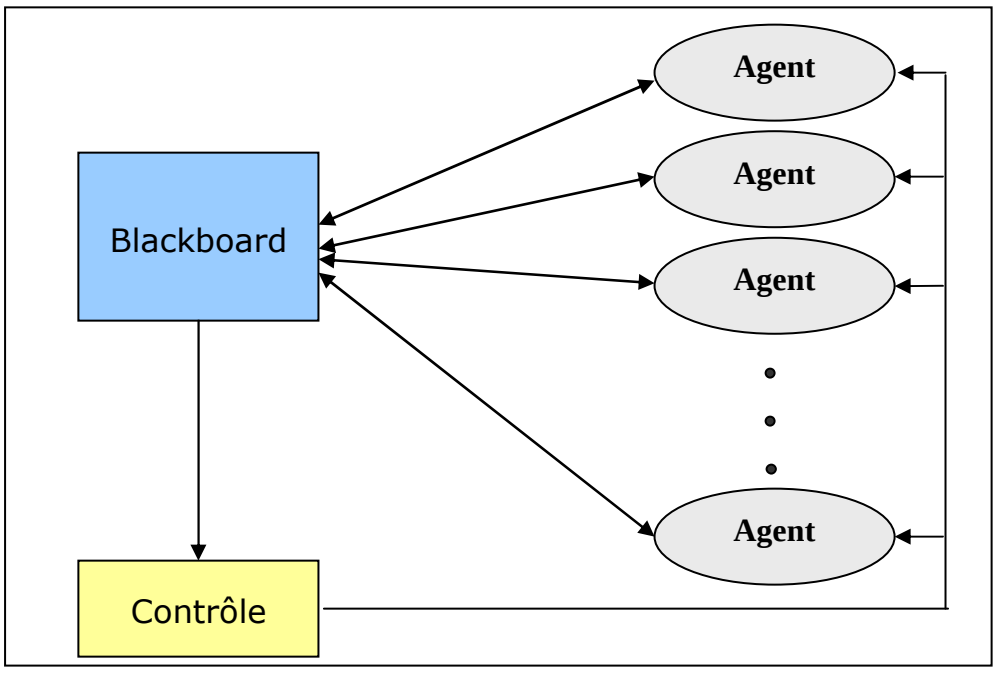

Figure 33 - Communication par partage d'informations

III.12) Méthodes de conception des S.M.A

Les S.M.A sont souvent des systèmes complexes qui demandent de longs efforts de développement, il devient nécessaire de développer de façon systématique des systèmes opérationnels.

Plusieurs méthodes d'analyse et de conception des S.M.A ont été proposées récemment. La plupart d'entre elles, s'appuient sur des techniques de modélisation empruntées à des méthodes connues en développement orienté objets ou en ingénierie des connaissances pour aider à la construction des modèles d'agents, de l'architecture du S.M.A, pour la spécification des modèles d'organisation, d'interaction, etc.

Pour mettre en œuvre un S.M.A, il faut gérer globalement le cycle de vie des agents, la communication entre les agents et le contrôle des agents du système. Le cycle de vie d'un S.M.A est géré par un processus de management qui peut être centralisé même si le processus propre au S.M.A est distribué. Ce processus de management comporte :

- Une phase d'initialisation où les agents sont créés et initialisés. D'autres agents peuvent être crées ensuite si certains agents ont la possibilité d'en créer.
- Une phase de comportement qui consiste à transmettre les messages et à donner la possibilité aux agents du système d'exécuter leur comportement répétitivement jusqu'à ce qu'une condition d'arrêt soit rencontrée.
- Une phase de terminaison où tous les agents du S.M.A doivent être supprimés pour redonner un environnement propre. Tous les agents doivent donc être référencés, soit par le système, soit par un autre agent.

III.13) Conclusion

Dans ce chapitre, nous avons présenté une étude de l'univers multi-agents, tout en essayant de clarifier la terminologie du domaine : I.A.D, SMA.

Les SMA sont largement répandus et développés autour du monde, ils sont parmi les domaines les plus porteurs en Intelligence Artificielle. J. Ferber, précise que les SMA touchent à des domaines différents, variés et pas évidemment reliés directement aux SMA comme les sciences cognitives, la philosophie, la biologie, les théories d'organisation, la gestion des ressources humaines etc.

Dans le chapitre suivant, nous exploitons les avantages de SMA à travers la proposition d'une architecture à base d'agents pour la découverte de services Web.

Chapitre IV

L'Architecture a base d'agents de decouverte des services Web

IV.1) Introduction

Nous avons vu dans les chapitres précédents les aspects les plus importants sur les services Web et les technologies nécessaires pour la conception de systèmes.

Dans ce chapitre nous présenterons une architecture pour la découverte des services Web sémantiques utilisant les ontologies et les agents.

L'approche proposée repose sur la coopération d'un ensemble d'agents qui permettrons d'utiliser les capacités sémantiques des services Web notés IO (inputs, outputs) en vue de les mettre en correspondance avec les concepts de la requête de l'utilisateur.

IV.2) Architecture de découverte de services proposée

L'architecture proposée est une extension de l'architecture orientée services SOA (Service Oriented Architecture). Cette architecture est basée sur des agents pour la découverte de services Web.

Les différents acteurs de l'architecture SOA (fournisseur, demandeur et registre) communiquent par l'intermédiaire de langages standardisés. Le fournisseur de services décrit le service Web en WSDL (Web Service Description Language proposé par le W3C) puis l'enregistre dans un registre universel (UDDI – Universal Description, Discovery and Integration proposé par OASIS). Via ce registre, un demandeur de S.W choisit un service approprié à ses besoins. Enfin, la communication entre demandeur et le fournisseur se fait par l'intermédiaire du protocole SOAP (Simple Object Access Protocol proposé par le W3C).

La description de services Web dans SOA n'est faite qu'au niveau syntaxique. Dans l'architecture que nous avons proposée, les services Web sont décrits sémantiquement via la classe ServiceProfile de l'ontologie OWL-S.

Cette architecture intègre des composants logiciels et exploitant une ontologie de domaine qui est utilisée lors de la phase de découverte des services Web, elle facilite la découverte automatique de services puisqu'elle permet d'affiner le processus de recherche qui met en correspondance une demande et des offres de services. Le recours à cette ontologie permet l'implémentation de mécanismes de filtrage (comparaison) entre une demande et des offres qui mettent en oeuvre autre chose qu'une simple égalité.

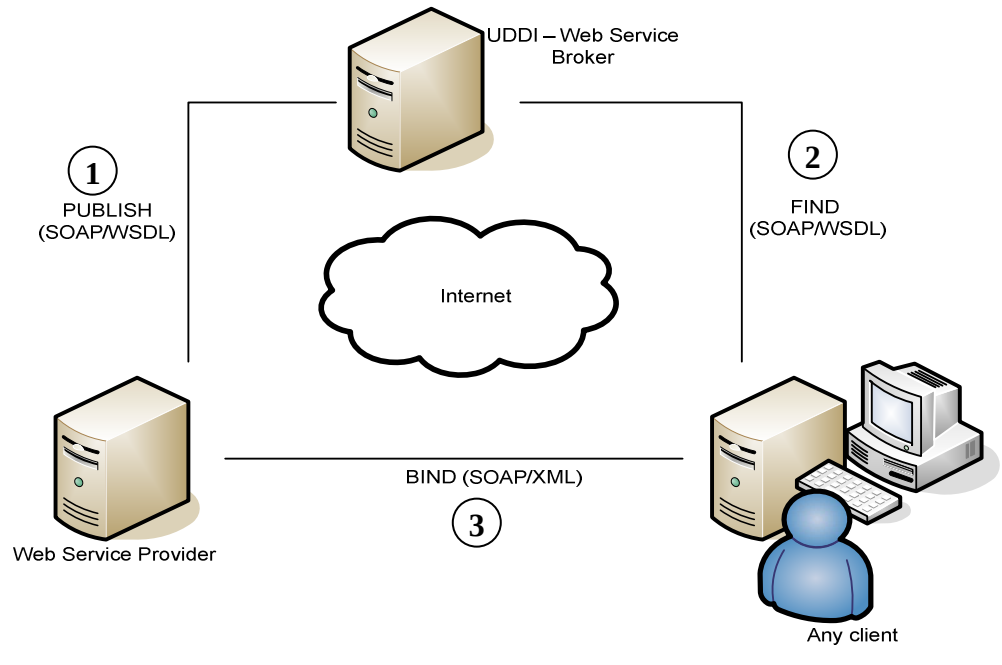

Figure 34 –Architecture Orientée services (SOA – Paradigme "find-bind-publish")

Le système proposé est un système multi-agents. L'intérêt pour les agents est dû à plusieurs raisons :

- Ils s'adaptent bien à la réalité dans la mesure où de nombreux problèmes sont de nature distribuée.
- L'utilisation de plusieurs agents résolvant le problème de façon différente produit généralement des solutions de meilleure qualité en terme de complétude et de précision.
- Ils permettent d'intégrer des connaissances diverses et complexes adoptant des formalismes hétérogènes.
- Ils permettent de résoudre des problèmes de taille et de complexité telle qu'il n'est pas réaliste d'essayer de les résoudre avec un seul agent.
- *Efficacité* : les agents peuvent fonctionner en parallèle lorsque l'environnement matériel et logiciel le permet. Ceci augmente par conséquence la rapidité de la résolution.
- *Réutilisabilité* : un agent peut être réutilisé pour implanter une partie d'un système.
- *Fiabilité* : les SMA sont capables de mieux fonctionner dans un environnement dégradé qu'un système mono agent. L'arrêt de fonctionnement d'un agent n'influe pas sur l'arrêt de la résolution.
- *Modularité* : partitionner un système en n agents (sous-systèmes) réduit sa complexité par un facteur parfois plus grand que n, et la configuration résultante se trouve plus facile à développer et à tester.

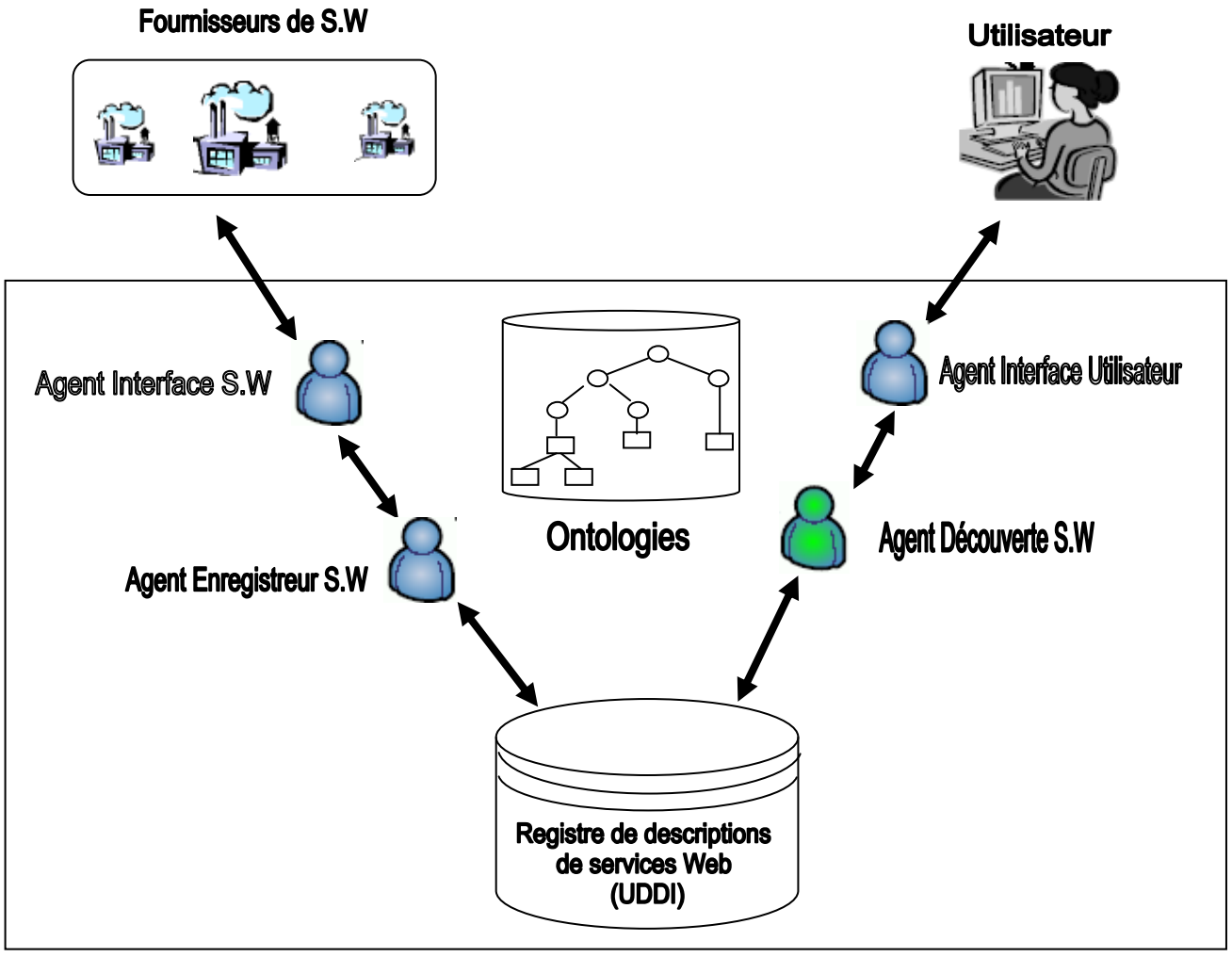

Figure 35 - Architecture multi-agents proposée

IV.3) Descriptions des agents composant l'architecture

IV.3.1) Agent interface service Web

Cet agent sert d'interface entre le système et le fournisseur du service Web, tel que pour chaque service Web un agent lui est associé. L'agent interface service Web permet l'enregistrement de la description sémantique relative au service Web. De plus, il permet des mises à jour des informations relatives au service Web.

L'architecture interne de l'agent interface service Web est composée de trois modules et un registre de sauvegarde, comme indiquée en figure 36. Les trois modules sont comme suit :

- **Le module de communication Fournisseur** : il reçoit la description sémantique de service Web et la transfère au module de traitement. L'opération inverse est également disponible, c'est-à-dire que le module peut recevoir des résultats du module de traitement pour les montrer au fournisseur.

- **Le module de communication inter-agents :** il reçoit du module de traitement, des demandes de transmissions de messages vers les autres agents. Il transfère également les informations reçues du système multi-agents au module de traitement.
- **Le module de traitement :** Il reçoit des informations de description de service Web du module de communication fournisseur et les sauvegarde dans le registre de l'agent. Ce registre contient toutes les informations recueillies par l'agent sur le fournisseur. Il détermine ensuite si toutes les informations nécessaires à la formation de l'offre sont disponibles. Dans l'affirmative, il construit un message et demande au module de communication inter-agents de le transmettre. Dans la négative, il demande les informations complémentaires au fournisseur qui sont transmises au module de communication fournisseur ; ce dernier se chargera ensuite de la faire parvenir au fournisseur.
- **Le registre de sauvegarde** : Il a pour rôle de sauvegarder les données de l'offre que le fournisseur a fournies.

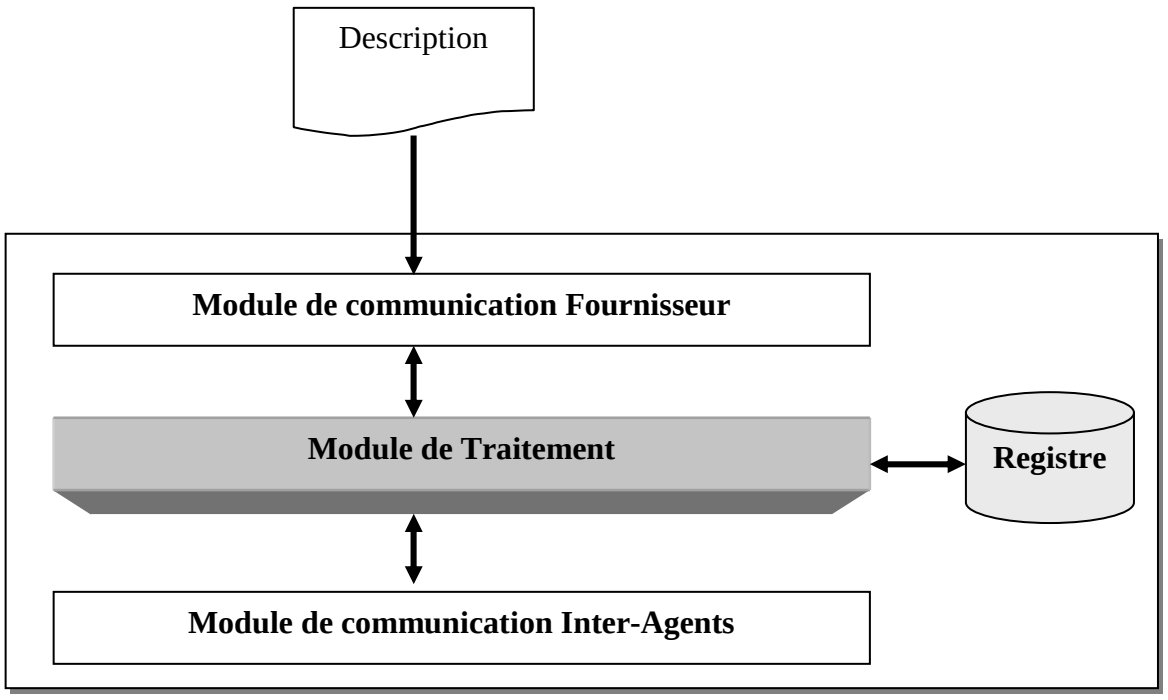

Figure 36 - Architecture de l'agent interface service Web

IV.3.2) Agent enregistreur de services Web

Le rôle de cet agent est la sauvegarde des descriptions sémantiques des services Web au niveau du registre UDDI, il contient deux modules et une interface comme indiquée en figure 37. Ils sont comme suit :

- **Le module de communication inter-agents :** c'est une interface entre l'agent et son environnement. Il est utilisé pour transmettre et recevoir des messages.

- **Le module de traitement :** Il reçoit des données du module de communication inter-agents et les achemine à l'interface de l'UDDI. Il envoie ensuite la notification d'enregistrement reçue par l'interface avec l'UDDI au module de communication inter-agents.
- **L'interface avec l'UDDI :** elle reçoit du module de traitement une demande d'enregistrement des descriptions des services web sémantiques dans UDDI. L'interface exécute l'opération demandée par le module de traitement et retourne à ce module une notification d'enregistrement (une confirmation de publication).

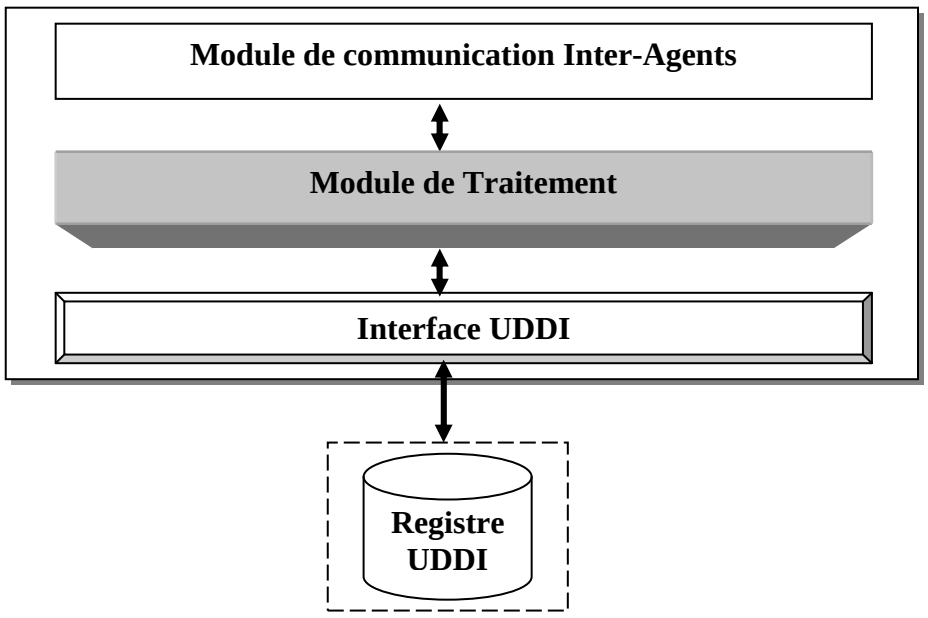

Figure 37 - Architecture de l'agent Enregistreur

IV.3.3) Agent interface utilisateur

L'agent interface utilisateur est la porte d'entrée des requêtes externes au système. Il fournit à l'utilisateur le bon formulaire lui permettant de faire une requête.

C'est l'agent qui va initier la découverte, en émettant à l'agent découverte, une requête constituée d'entrées, de sorties, une référence sur l'ontologie de domaine à utiliser (par exemple l'ontologie des voyages touristiques) et présente les résultats adaptés aux préférences des utilisateurs après le traitement.

L'architecture interne de l'agent utilisateur est composée de trois modules principaux et d'un registre de sauvegarde comme l'indique la figure 38. Les trois modules sont comme suit :

- **Le module de communication utilisateur** : il reçoit les requêtes et les transfère au module de traitement. L'opération inverse est également disponible, c'est-à-dire que le module peut recevoir des résultats du module de traitement pour les présenter à l'utilisateur.

- **Le module de communication inter-agents :** il reçoit du module de traitement, des demandes de transmissions de messages vers les autres agents. Il transfère également les informations reçues du système multi-agents au module de traitement.
- **Le module de traitement :** Il reçoit des données du module de communication utilisateur et les sauvegarde dans le registre de l'agent. Ce registre contient toutes les informations recueillies par l'agent sur l'utilisateur du système multi-agents. Il détermine ensuite si toutes les informations nécessaires à la formation de la requête sont disponibles. Dans l'affirmative, il construit un message et demande au module de communication inter-agents de le transmettre. Dans la négative, il demande les informations complémentaires à l'utilisateur qui sont transmises au module de communication utilisateur; ce dernier se chargera ensuite de la faire parvenir à l'utilisateur.
- **Le registre de sauvegarde** : Il a pour rôle de sauvegarder les données de requête que l'utilisateur a fournies.

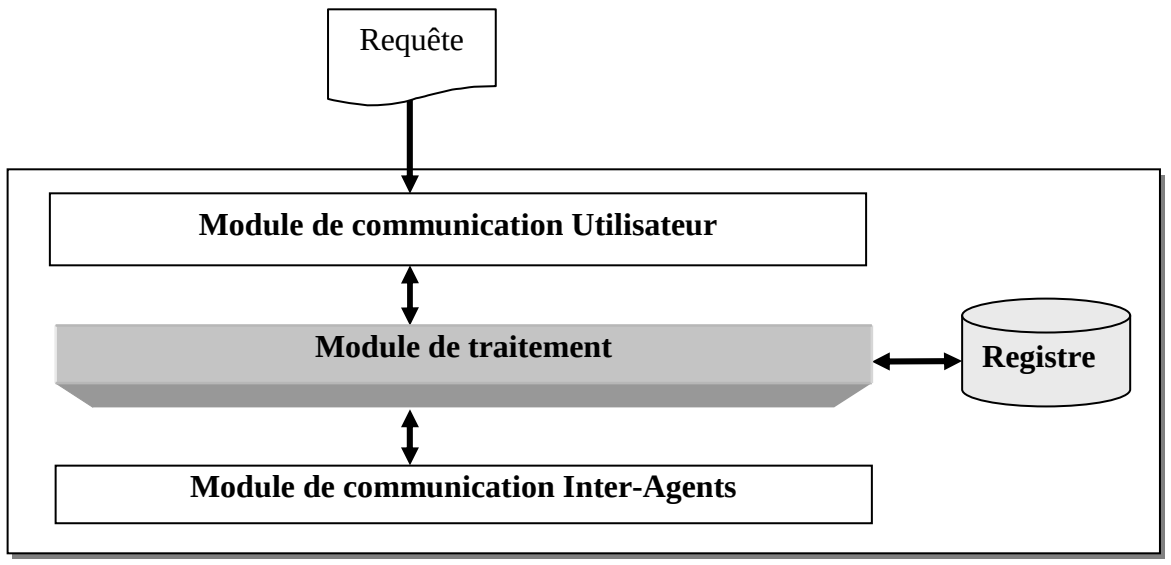

Figure 38 - Architecture de l'agent interface utilisateur

IV.3.4) Agent de découverte de services Web

C'est un agent qui permet la découverte des descriptions des services Web satisfaisant la requête envoyée par l'agent interface utilisateur sur le plan sémantique.

L'architecture interne de l'agent découverte est composée de deux modules et une base de services pour stockage les descriptions sémantiques des services rendus par UDDI comme l'indique la figure 39. Ils sont comme suit :

- **Module de communication inter-Agents :** Il reçoit de l'agent interface utilisateur la requête sous forme d'un message et suite à cela, il appelle le module de traitement. Il reçoit également des demandes de transmission de messages de module de traitement. Ces demandes de transmissions constituent des réponses des requêtes reçues.

- **Base des services** : est utilisée pour stocker les descriptions sémantiques des services Web satisfaisant la requête de l'utilisateur.

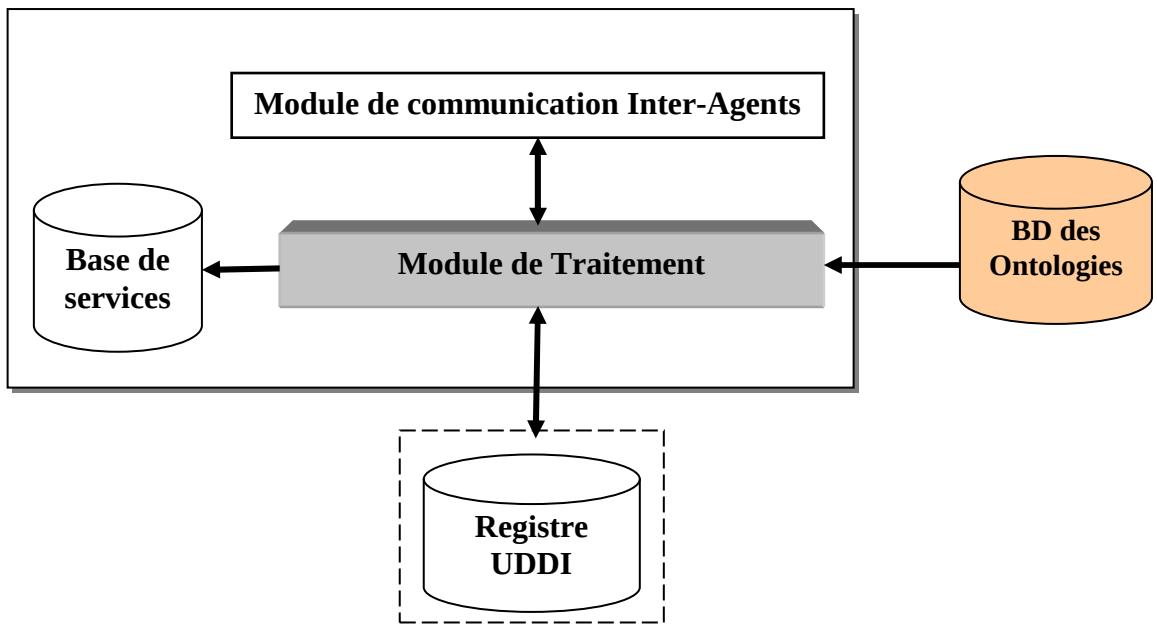

Figure 39 - Architecture de l'agent Découverte

- **Module de traitement** : il a deux tâches :

1) *la tâche d'analyse* : sélectionne l'ontologie de domaine correspondante à la demande (à partir de base d'ontologies qui stocke des ontologies de divers domaines), en extrait les classes et leurs liens et construit l'arborescence correspondante. Dans notre contexte, cette action est possible puisque le vocabulaire défini dans l'ontologie de domaine est décrit sous forme hiérarchique. Chaque sommet de cette arborescence correspond à une classe de l'ontologie et chaque arc correspond à une relation de sous-classe. Cette arborescence permet de déduire des relations de généralisation (subsumption) entre les concepts, c'est-à-dire le fait qu'un concept soit plus général qu'un autre. Un concept C englobe (subsume) un concept C' si l'extension de C' est incluse dans celle de C. On dira alors que C est plus général que (ou englobe) C'. Ce principe nous permet de réaliser des comparaisons flexibles entre les offres et les demandes, c'est-à-dire d'associer à une demande des offres qui ne correspondent pas exactement aux besoins exprimés mais qui s'en rapprochent.

Pour illustrer ce propos, prenons un exemple issu de l'ontologie «Conference» [Bouz, 2007]. Dans cette ontologie (voir figure 40), le concept Domaine englobe le concept Groupware qui lui-même englobe les concepts Workflow et EDI. Ainsi, si l'on dispose d'une part d'un lecteur capable de lire un article dans le domaine de Groupware, et d'autre part, d'un demandeur à la recherche d'un lecteur d'article dans le domaine de Workflow, il est possible de les mettre en contact même si leurs déclaration (de capacités) ne sont pas exactement identiques, dans la mesure où l'on considère que le domaine du Groupware englobe celui du Workflow.

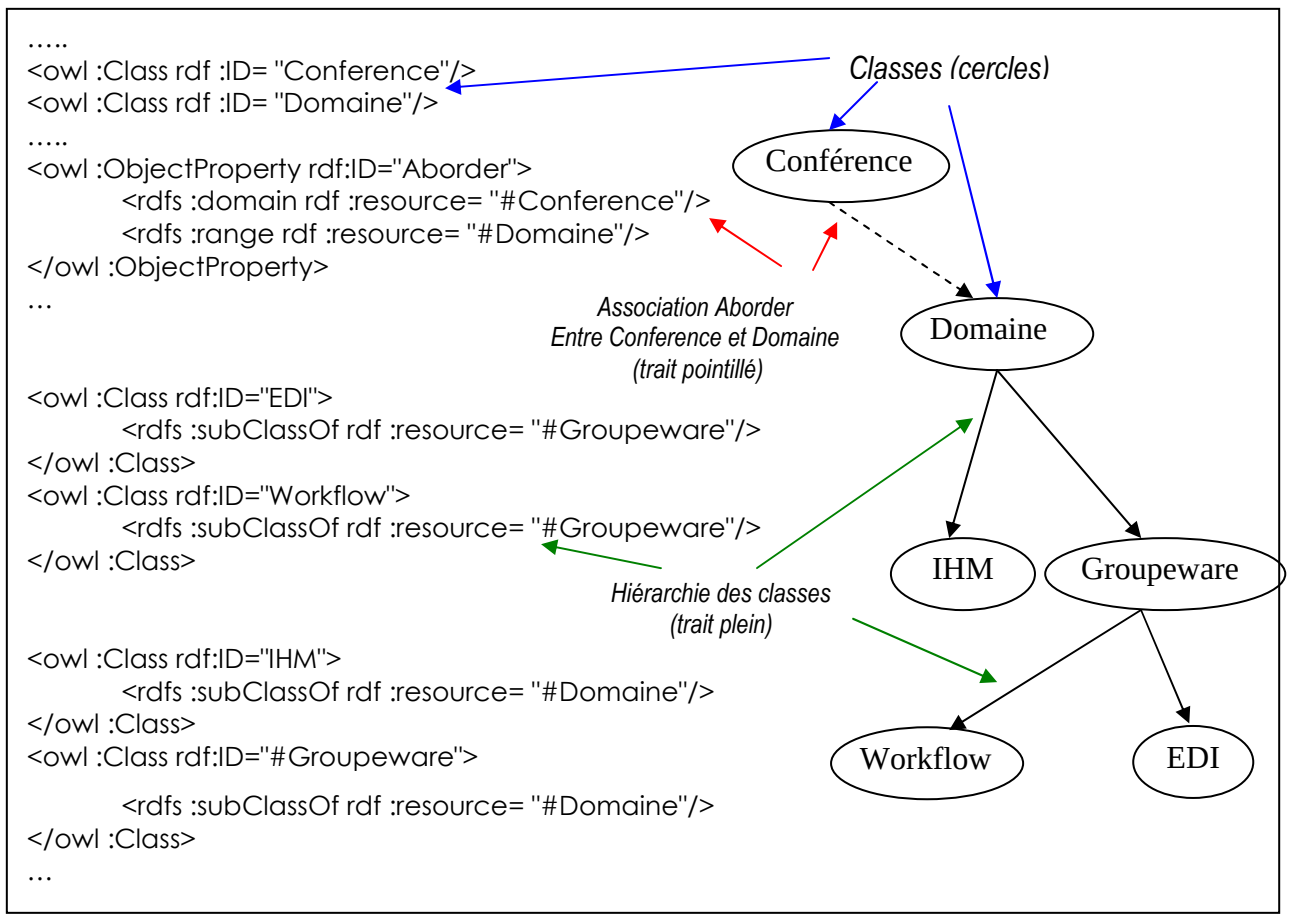

Figure 40 - Extrait de l'Ontologie

2) *La tâche de comparaison :* permet de comparer une demande et des offres de services en considérant l'ontologie (voir figure 41) et ce conformément aux quatre principaux modes de comparaison définis dans [Pao, 2002] en utilisant un algorithme de matchmaking : le mode Exact, le mode PlugIn, le mode Subsume et le mode Fail.

1. *Le mode Exact* sélectionne une offre si elle correspond exactement à une demande (demande = offre) c'est-à-dire les entrées et les sorties de la demande sont équivalents aux entrées et sorties de l'offre (matching exact).
2. *le mode Plug-In* retourne une offre si elle englobe une demande (demande<offre) c'est-à-dire les entrées de la demande englobe les entrées de l'offre et les sorties de la demande sont englobées par les sorties de l'offre dans l'ontologie de domaine (matching inclusif).
3. *le mode Subsume* retourne une offre si elle est incluse dans une demande (demande>offre) (l'Inverse de mode Plug-In) (matching partiel)
4. *le mode Fail* retourne faux, si aucune correspondance entre l'offre et la demande (demande # offre) (echec de matching).

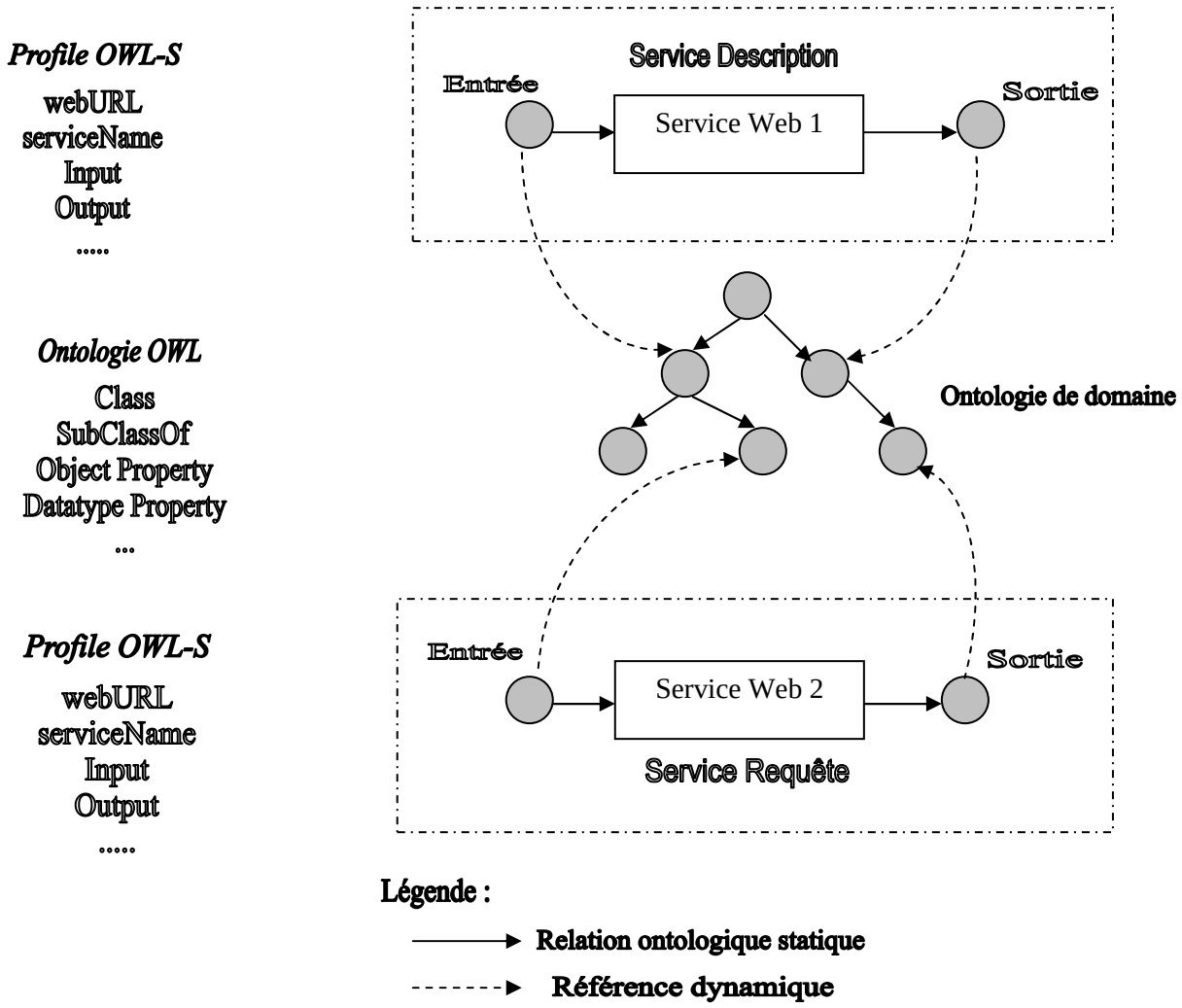

Figure 41 - méthodologie de comparaison

Les modes 2 et 3 de comparaison utilisent l'ontologie de domaine. Plus précisément, les offres et demandes de services étant exprimées en OWL-S, nous comparons, selon les quatre modes précédents, tous les éléments définis dans les clauses «input» et «output» (entrées et sorties) dans la classe ServiceProfile des offres et des demandes (voir Figure 42)

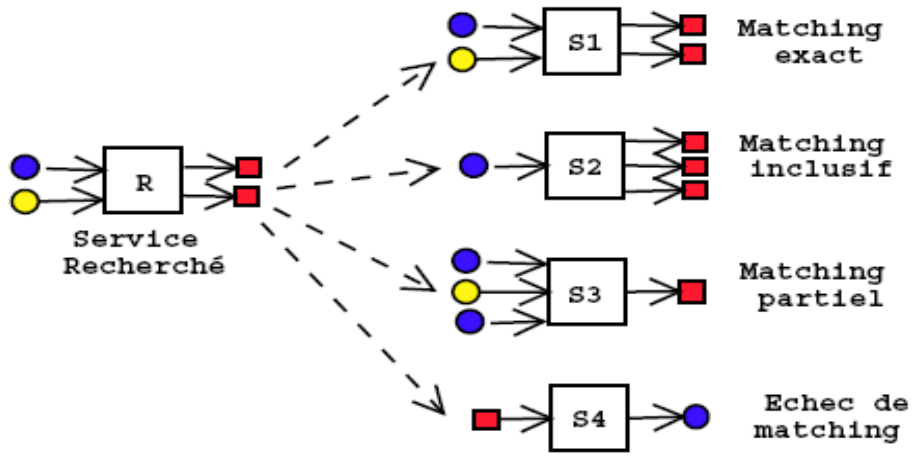

Figure 42 - Matching basé sur les entrées/sorties

L'algorithme de comparaison utilisé à la fois en mode Plug-In et en mode Subsume utilise la fonction Englobe [Lot, 2006] donnée ci-dessous (voir figure 43).

L'agent applique un test de subsomption sur les sorties (outputs) (voir figure 44) ensuite, on attribue un score pour chaque mode de matching : Exact (score=3), PlugIn (score=2), Subsume (score=1), Fail (score=0) (voir figure 45). Par exemple, dans la figure 46, si la requête ayant une sortie "IHM", le matching entre la requête et l'offre ayant comme sortie "IHM" est exact et le score égal 3. Si la sortie de l'offre est "Workflow", le matching est Plug-in et le score égal 2. Si la sortie de l'offre est "conférence", le matching est subsume et le score égal 1.

```
Fonction Englobe (E1 : chaîne, E2 : chaîne) : booléen
  % Cette fonction retourne vrai si E1 englobe E2 faux sinon
  % E1 est un élément de la clause Input ou Output de l'Offre
  % E2 est un élément de la clause Input ou Output de Demande
  % A représente l'ontologie (sous forme arborescente)
  % On utilise les fonctions de haut niveau suivantes :
  % Père(E) : retourne le père de E dans A
  % Racine(A) : retourne la racine de A

  Variables
SommetCourant : UnSommet              % Sommet de A en cours d'examen
LesAncêtres : EnsembledeSommets       % Les ancêtres de E2
Début
      LesAncêtres ← ∅
      Si E2 = racine(A) Alors
            % E2 n'a pas d'ancêtre et ne peut pas être englobé
            LesAncêtres ← ∅
      Sinon
            SommetCourant ← Père(E2)
            LesAncêtres ← Père(E2)
            Tant Que (SommetCourant <>Racine(A)) Faire
                  SommetCourant ← Père(SommetCourant)
                  % «+» désigne l'ajout d'un nouvel élément
                  % dans l'ensemble LesAncêtres
                  LesAncêtres ← LesAncêtres + SommetCourant
            Fin Tant Que
      Fin Si
      Englobe ← (E1 ∈ LesAncêtres)
Fin
```

Figure 43 - Fonction Englobe [Lot, 2006]

```
Procedure degreeOfMatch(OutD,OutO : chaîne )
% Cette Procédure retourne résultat de comparaison
% OutD, OutO sont la sortie de la demande et de l'offre respectivement
Variables Ch : chaîne
Début
    Si   OutO = OutD Alors  ch ← "Exact"
    Si   Englobe(OutO, OutD) Alors   ch ←  "PlugIn"
    Si   Englobe(OutD, OutO) Alors   ch ←  "Subsume"
    Autrement ch ←  "Fail"
    Fin Si
    Return ch
Fin
```

Figure 44 – Procédure de matching des sorties (outputs)

```
Fonction GetScore(rel : chaîne) : Entier
% Cette Fonction retourne le score de matching
Val =0
Début
    Si  rel = "Exact" Alors val = 3
    Si  rel = "PlugIn" Alors   val=2
    Si  rel = "Subsume" Alors   val=1
    Si  rel = "Fail" Alors   val=0
     Fin Si
    GetScore ← val
Fin
```

Figure 45 – Fonction retourne le score de matching

Le niveau de correspondance sémantique entre les paramètres d'entrée (inputs) est assigné de la même manière que pour les paramètres de sortie (outputs).

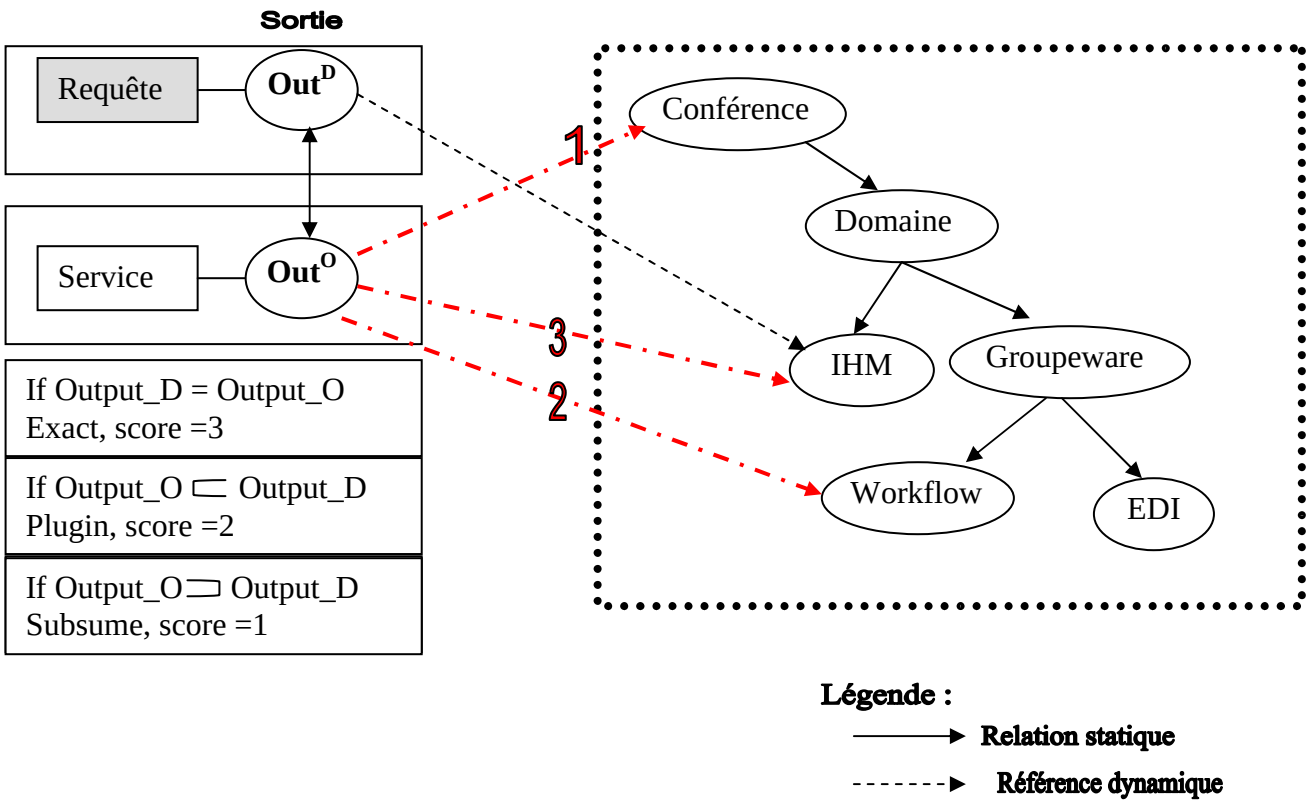

Figure 46 – matching sémantique entre le service et la requête dont sa sortie Out^O est "IHM". Chaque ligne pointilléé (tiret-point) représente un type spécifique de matching

Le matching entre les entrées est calculé avec la même méthode.

L'équation ① généralise la comparaison entre le concept de l'offre de service C_i^O et le concept correspondant de la requête C_i^D :

$$Match(C_i^D, C_i^O) = \begin{cases} 3 & \text{si } C_i^D = C_i^O \\ 2 & \text{si } C_i^D \sqsubset C_i^O \\ 1 & \text{si } C_i^D \sqsupset C_i^O \\ 0 & \text{sinon} \end{cases} \qquad \text{…….. ①}$$

Supposons que l'on a **m** concepts dans la description de l'offre de service et **m** concepts correspondants dans la description de la requête, la similarité ou le match global entre la demande (requête) D et l'offre O peut dériver par prendre la somme de score de la paire de concepts (équation ②) :

$$Similarité(D, O) = \sum_{i=1}^{m} Match(C_i^D, C_i^O) \qquad \text{……..... ②}$$

Par conséquent, le matching entre la requête et un ensemble d'offres de services Web peut être mesuré de façon quantitative. Le service qui a un haut score de similarité représente le plus précis service pour la requête. Et on peut trouver plus d'un service.

Exemple: supposons qu'il existe trois services Web de vente S1, S2 et S3 publiés sur le Web (dans un annuaire UDDI). Ses paramètres fonctionnels (entrées, sorties) sont :

- S1 ayant deux entrées "vehicle" et "parts" et une seule sortie "price".
- S2 ayant deux entrées "parts" et "car" et une seule sortie "price".
- S3 ayant deux entrées "unit" et "material" et une seule sortie "price".

Et supposons qu'un client lance une requête de recherche constitue de deux entrées "Car" et "Parts" et une sortie "Price":

Et nous avons le fragment de l'ontologie de "vehicle" suivant :

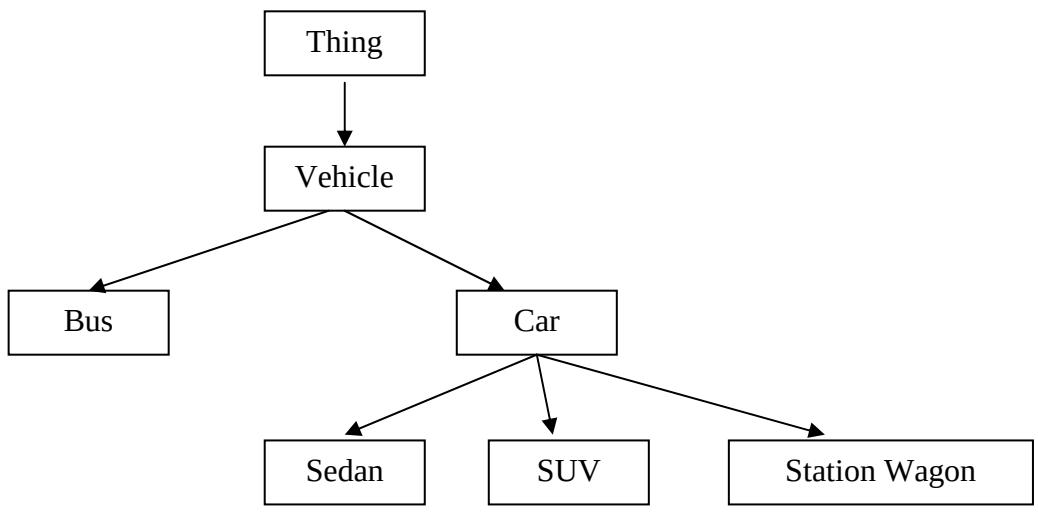

Figure 47 – Un fragment d'ontologie de vehicle

Si nous appliquons l'algorithme de matching, nous allons obtenir les résultats suivants :

- Comparaison des entrées (inputs) :

S1 :

Car → vehicle, mode = Plug-in, score = 2, Total =2

Car → parts, mode= Fail, score = 0, Total = 2

Parts → vehicle, mode= Fail, score = 0, Total = 2

Parts → parts, mode = Exact, score = 3, Total =5

Total du score des entrées = 5

S2 :

Car → parts, mode = Fail, score = 0, Total =0

Car → car, mode= Exact, score = 3, Total = 3

Parts → parts, mode = Exact, score = 3, Total =6

Parts → car, mode = Fail, score = 0, Total =6

Total du score des entrées = 6

S3 :

Car → unit, mode = Fail, score = 0, Total =0

Car → material, mode = Fail, score = 0, Total =0

Parts → unit, mode = Fail, score = 0, Total =0

Parts → material, mode = Fail, score = 0, Total =0

Total du score des entrées = 0

- Comparaison des sorties (outputs) :

S1 :

price → price, mode = Exact, score = 3, Total =3

Total du score des sorties = 3

S2 :

price → price, mode = Exact, score = 3, Total =3

Total du score des sorties = 3

S3 :

price → price, mode = Exact, score = 3, Total =3

Total du score des sorties = 3

- matching global :

S1 :

Total du score (total du score des entrées + total du score des sorties) = 5 + 3 = 8, Bien

S2 :

Total du score = 6 + 3 = 9, Meilleur

S3 :

Total du score = 0 + 3 = 3, Pas bon

►Donc, le service Web S2 est considéré comme le meilleur qui correspond à la requête.

IV.4) Communications des Agents

La communication accroît les perspectives des agents en leur agréant le bénéfice des informations et du savoir-faire des autres agents. Subséquemment, la communication des agents constitue l'un des moyens fondamentaux pour assurer la répartition des tâches et la coordination des actions entre agents. FIPA [Fipa, 1999] acteur dans le domaine des systèmes multi-agents a pour principale mission de mettre au point un standard pour la communication entre agents. Un de ses aboutissements est la norme FIPA-ACL [Cha-Di, 2002]. Il en existe une autre celle de KQML [Fin, 1994].

Un Langage de Communication Agent (ACL) doit être conçu pour l'échange entre agents d'informations, de connaissances ou de services. Le format utilisé pour l'échange des connaissances est fourni par un langage de contenu, indépendant du langage ACL (par exemple : KIF, FIPA-SL, Prolog, Clips).

FIPA-ACL propose un système standard d'échange de messages entre agents, la sémantique de ces messages est similaire à celle du langage KQML.

Le message minimum type du FIPA ACL contient tout d'abord :

Le type du message envoyé = syntaxe de ce message.

- l'expéditeur du message (Sender)
- le destinataire du message (Receiver)
- le contenu du message
- Actes de communication **:** types de communications effectuées (request, inform)

Cependant, ces messages minimums ne suffisent pas toujours pour communiquer : on peut avoir besoin, pour la compréhension du message et pour la rapidité de celle-ci ainsi que la rapidité de traitement du message, d'indiquer d'autres informations telles que :

- le langage utilisé dans le contenu du message ("language ..."),
- le protocole utilisé,
- l'ontologie auquel le message se rattache ("ontology ..."),
- la référence d'un message antérieur auquel le message actuel se rattache ("in-reply-to ..."), ou la référence d'un message ultérieur attendu en retour ("reply-with ...").
- la référence de la conversation.

La figure 48 illustre comme exemple le message **:** l'agent i informe l'agent j d'un rendez-vous. Le contenu, exprimé en Prolog, ce réfère à l'ontologie Scheduling.

```
(INFORM
: SENDER (agent-identifier :name i)    //* l'adresse de l'émetteur du message.
: RECEIVER (agent-identifier :name j) //* la ou les adresses des récepteurs.
: CONTENT "appointment (today, INRIA_RA)"   //* l'information communiquée.
: ONTOLOGY Scheduling       //* l'ontologie du domaine
: LANGUAGE Prolog)          //* le langage de représentation du contenu.
```

Figure 48 - Exemple d'un message FIPA-ACL

Nous avons utilisé comme mode de communication entre les agents composant notre système l'envoi de messages (voir Figure 49).

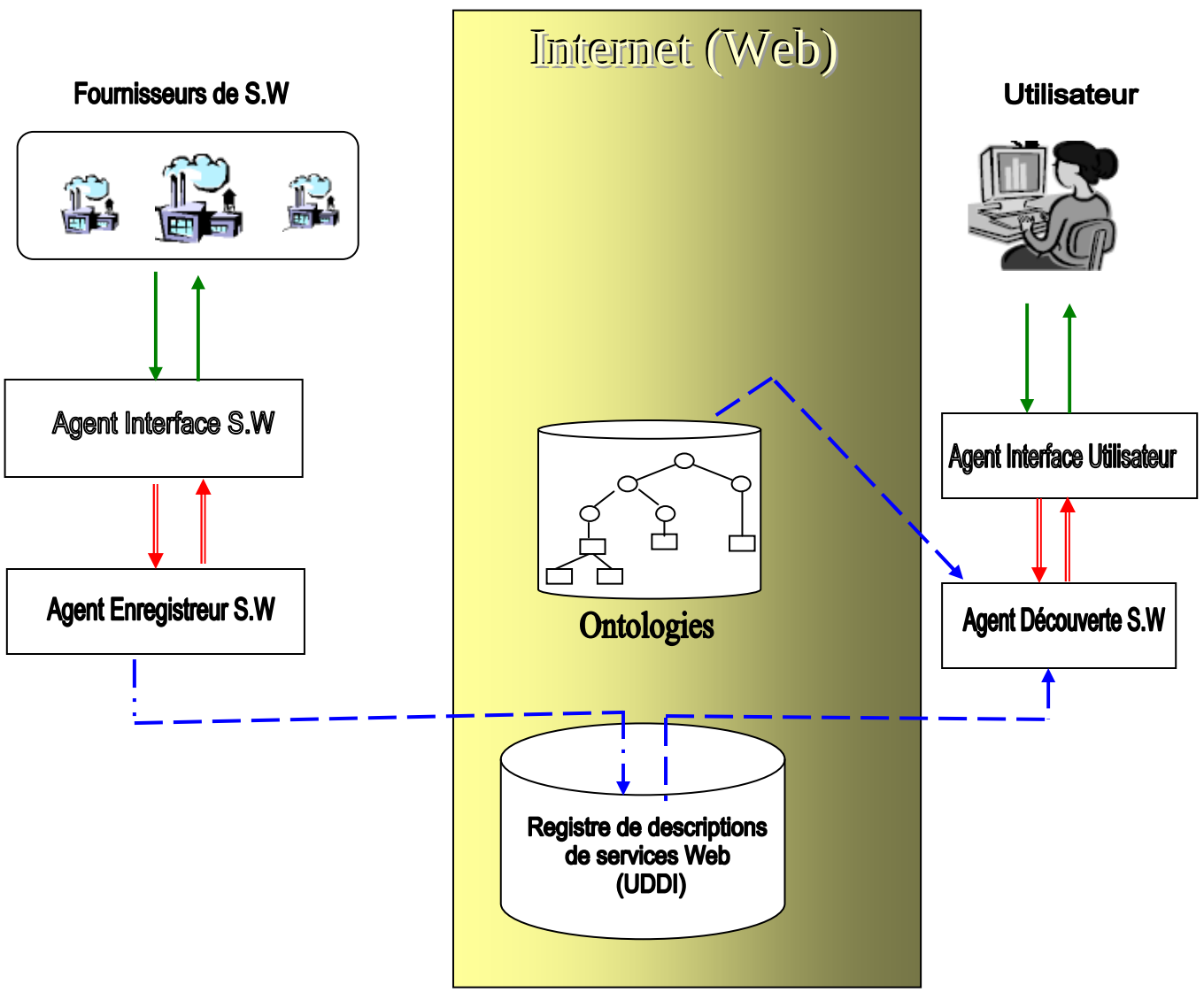

Figure 49 - Modes de communication

IV.5) Protocole d'enregistrement de S.W

Dans l'architecture proposée, un fournisseur de service Web annonce son service (offre) en publiant le service Profil d'une description OWL-S via un formulaire présenté par l'agent interface service Web.

Si toutes les informations nécessaires à la formation de l'offre sont disponibles **Alors** l'agent interface service Web transmet à l'agent enregistreur service Web la description sémantique de service Web **Sinon** informer le fournisseur.

Une fois les paramètres de l'offre sont reçus, l'agent enregistreur enregistre la description OWL-S dans UDDI, ensuite il envoie une notification d'enregistrement (confirmation) à l'agent interface service Web qui à son rôle, il la présente au fournisseur.

La figure 50 représente le diagramme de séquences du protocole de publication des descriptions des services Web dans le système que nous avons proposé.

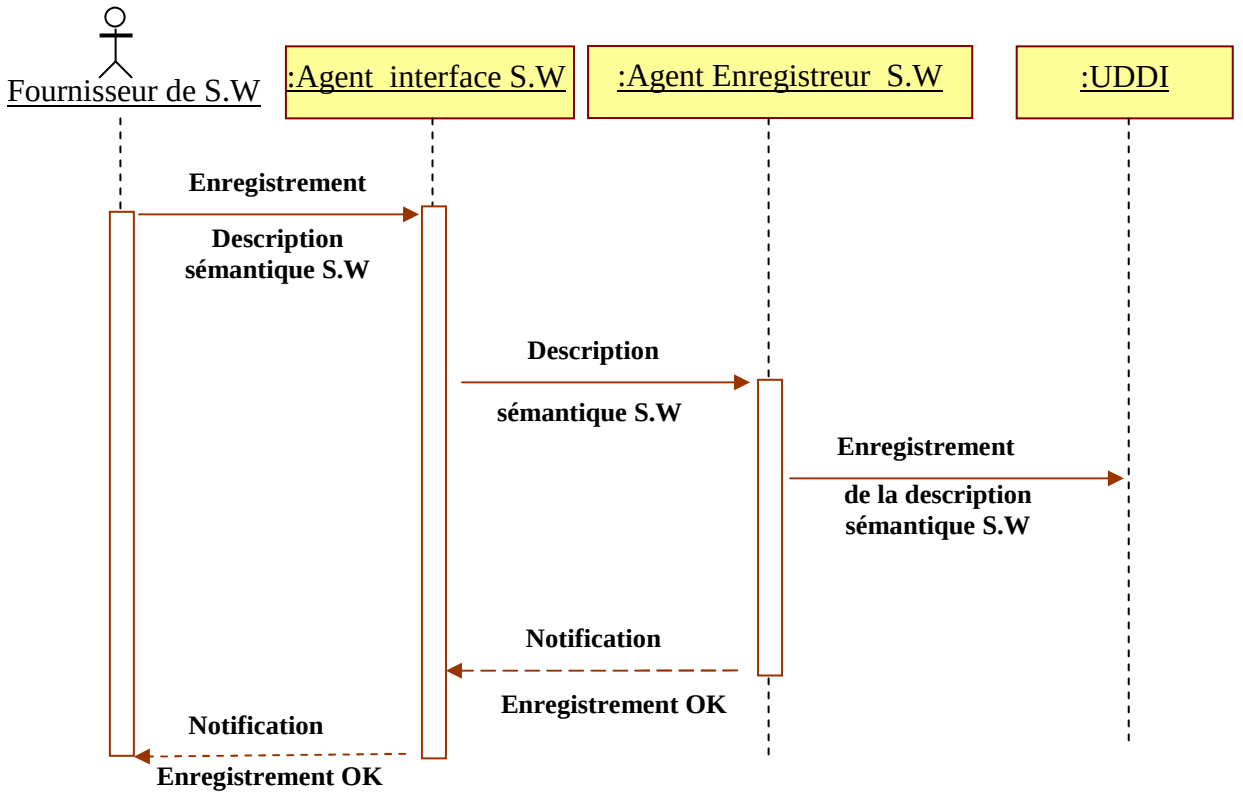

Figure 50 - Diagramme de séquences de publication de services Web sémantiques

IV.6) Protocole de découverte de S.W

Dans notre système, les agents communiquent en échangeant des messages basés sur le langage de communication FIPA ACL. La figure 51 montre le diagramme d'interaction entre agents pour ce scénario qui comprend plusieurs étapes :

- L'agent interface utilisateur accueille l'utilisateur du système, puis lui présente un formulaire à remplir (entrées, sorties …).
- **Si** toutes les informations nécessaires à la formation de la requête sont disponibles **Alors** l'agent interface utilisateur transmet à l'agent découverte les données **Sinon** informer l'utilisateur.
- Une fois la requête est reçue, les informations sémantiques de la requête et de chaque Web service stockée dans UDDI sont traitées par application de l'algorithme de matchmaking. Cet algorithme permet de calculer le degré de correspondance en utilisant l'ontologie de domaine. Le résultat est un ensemble de services Web pertinents selon le degré de correspondance sémantique satisfaisant la requête de l'utilisateur.
- Le résultat est envoyé à l'agent interface utilisateur.
- L'agent interface utilisateur présente le résultat à l'utilisateur.
- Et enfin l'utilisateur peut invoquer le(s) Web service(s) intéressant(s).

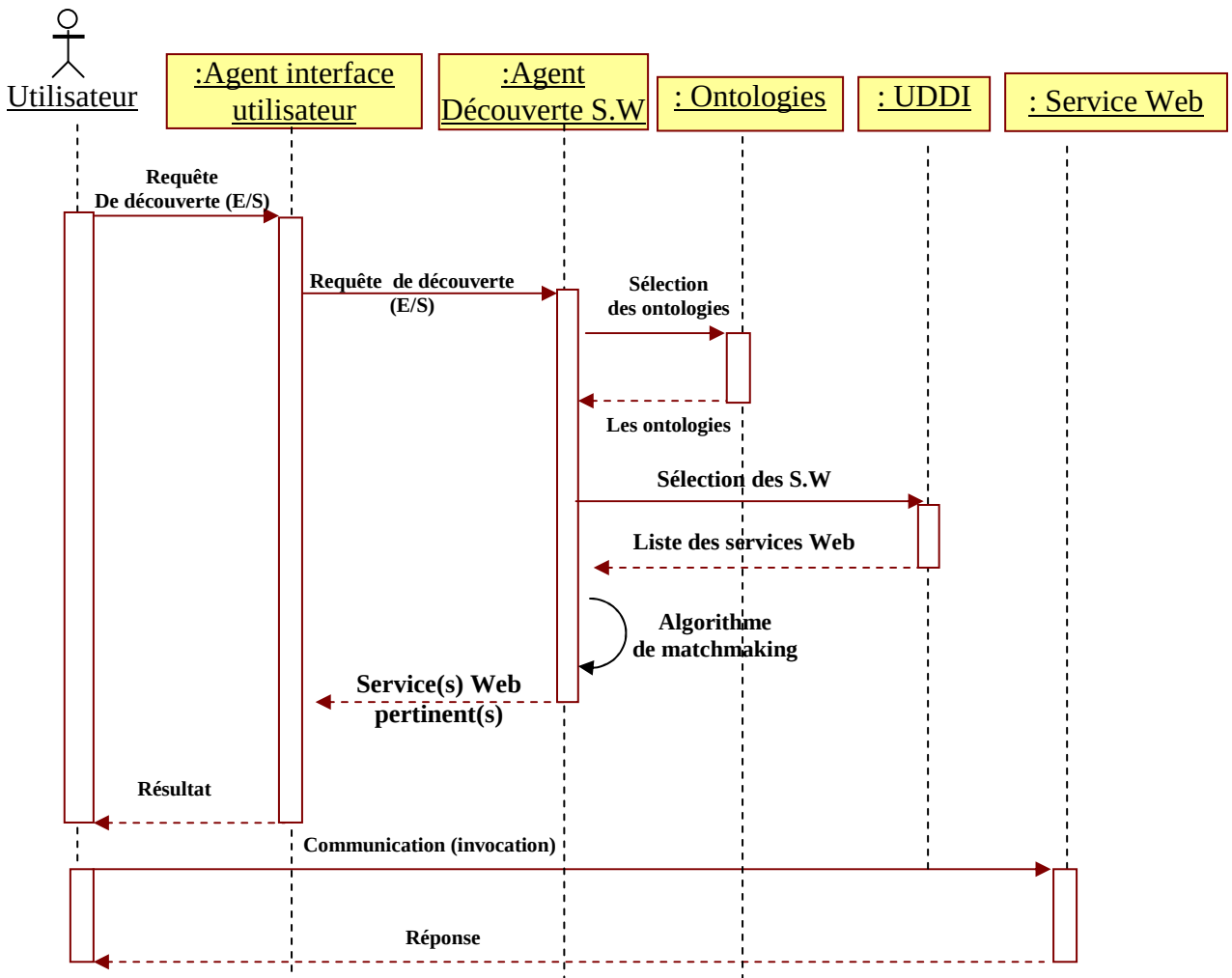

Figure 51 - Diagramme de séquences de découverte de services web sémantiques

IV.7) Conclusion

Dans ce chapitre, nous avons présenté notre architecture de découverte de services Web sémantiques basée agents. La recherche sémantique des services est faite par application de l'algorithme de matchmaking. Nous avons utilisé les agents coopérants pour bénéficier de l'autonomie, la modularité, la distribution et l'intelligence des agents.

Nous avons montré un protocole d'interaction qui regroupe quatre types d'agents: l'agent interface client, l'agent interface fournisseur, l'agent enregistreur et l'agent de découverte de services Web. Ces derniers ont la grande responsabilité dans l'approche proposée, en effet, ils permettent de réaliser une publication des services Web et une correspondance basée sur le test de subsomption entre la requête du client et les capacités sémantiques des services notées Inputs-Outputs.

Dans le chapitre suivant, nous aborderons la phase de l'implémentation en décrivant les outils utilisés.

CHAPITRE V

IMPLEMENTATION

V.1) Introduction

Dans ce chapitre nous allons décrire les grands axes de la réalisation de notre système et les outils utilisés.

Le meilleur moyen pour construire un système multi-agents (SMA) est d'utiliser une plate-forme multi-agents. Une plate-forme multi-agents est un ensemble d'outils nécessaire à la construction et à la mise en service d'agents au sein d'un environnement spécifique. Ces outils peuvent servir également à l'analyse et au test du SMA ainsi créé. Ces outils peuvent être sous la forme d'interface de programmation (API) et d'applications permettant d'aider le développeur. Nous allons utiliser dans notre implémentation la plate-forme JADE (Java Agent DEvelopment framework).

Le développement multi-agents respecte les recommandations de la FIPA, notamment le langage de communication retenu est ACL-FIPA. Pour la saisie et la manipulation d'ontologies, l'environnement Protégé a été retenu.

V.2) Choix techniques

V.2.1) Java comme langage de programmation

Java [Hor, 1999] est un langage orienté objet, multi-plate-forme, conçu pour être facilement utilisable et maniable par une large majorité de développeurs. Développé par Sun microsystems. Java rassemble aujourd'hui derrière lui une large communauté d'acteurs informatiques majeurs tels que HP, IBM, Oracle, Borland.

Quand on aborde ce langage, on met souvent en exergue : « Java est un langage simple, orienté objet, réparti, interprété, robuste, sécurisé, indépendant de l'architecture, portable, efficace, multithreads et dynamique ».

Java a été choisi pour notre travail pour les raisons suivantes :

- Java est absolument basé sur la technologie Objet et emprunte de nombreux éléments au C++. En revanche, les concepteurs de Java ont supprimé les concepts plus difficiles.
- Java s'affranchit des plates-formes : il fonctionne en mode interprété. Nous pouvons développer le programme sous n'importe quel système d'exploitation : Unix, Macintosh, Dos, Windows, DOS, etc. Cet avantage nous permet de réutiliser au maximum toutes les fonctions ou classes en diminuant la durée de développement.

Le fonctionnement de Java est assuré par JVM (la machine virtuelle Java) et JDK (le Java Developement Kit) qui peuvent être installé dans les différents systèmes d'exploitation.

JDK regroupe l'ensemble des éléments permettant le développement, la mise au point et l'exécution des programmes Java. Il inclut de nombreux outils de développement, un jeu de

classes et de services et un ensemble de spécification. Le JDK est en évolution, et la version actuelle est 1.6.

V.2.2) Technologie de développement des ontologies

Le développement des ontologies sera développé en utilisant l'éditeur Protégé 2000 qui permet aussi bien de créer et de visualiser des classes et des propriétés dans l'ontologie et que de créer et visualiser des instances de ces classes

Protégé est le plus connu et le plus utilisé des éditeurs d'ontologie. Open-source, développé par l'Université de Stanford, il a évolué depuis ses premières versions (Protégé-2000) pour intégrer à partir de 2003 les standards du Web sémantique et notamment OWL. Il offre de nombreux composants optionnels : raisonneurs, interfaces graphiques.

Protégé est basé sur la technologie Java, il est donc extensible. Il est alors possible de réaliser des modules additionnels pour modifier ou compléter ce logiciel en créant ainsi de véritables applications.

Enfin, Protégé permet d'extraire une ontologie créée sous un format OWL (Ontology Web Language). Cette ontologie pourra donc être consultée avec un autre éditeur ou être affichée sur une page internet.

V.2.3) Choix d'une plate-forme SMA : JADE

L'outil choisi pour l'implémentation du système multi-agents est JADE.

- *Description de JADE*

JADE (Java Agent DEvelopement framework) est une plate-forme multi-agents créé par le laboratoire TILAB et décrite par Bellifemine et al. dans [BEL 99]. JADE permet le développement de systèmes multi-agents et d'applications conformes aux normes FIPA. Elle est implémentée en JAVA et fourni des classes qui implémentent «JESS» (Java Expert System Shell) pour la définition du comportement des agents. JADE possède trois modules principaux (nécessaire aux normes FIPA).

- DF « Director Facilitor » fournit un service de « pages jaunes » à la plate-forme ;
- ACC « Agent Communication Channel » gère la communication entre les agents ;
- AMS « Agent Management System » supervise l'enregistrement des agents, leur authentification, leur accès et l'utilisation du système.

Ces trois modules sont activés à chaque démarrage de la plate-forme.

La plate-forme d'agent peut être répartie sur plusieurs serveurs. Une seule application Java, et donc une seule machine virtuelle de Java (JVM), est exécutée sur chaque serveur. Chaque JVM est un conteneur d'agents qui fournit un environnement complet pour l'exécution d'agent et permet à plusieurs agents de s'exécuter en parallèle sur le même serveur.

L'architecture de communication offre la transmission de messages flexibles et efficaces. JADE crée et contrôle une file d'attente des messages entrants pour chaque agent. Le modèle global de communication FIPA a été mis en application. Ses composants ont été distingués clairement et ont été entièrement intégrés : protocoles d'interaction, ACL, langues, schémas de codage, protocoles de transport.

Concrètement, un thread est lancé pour chaque agent, mais ces derniers doivent souvent exécuter des tâches parallèles. Avec la solution du multithreading offerte directement par Java, Jade supporte également la gestion des comportements coopératifs. Le run-time inclut également quelques fonctions complexes prêtes l'emploi pour les taches les plus communes dans la programmation agent, comme des protocoles d'interaction de FIPA.

- *La norme FIPA*

La FIPA (Foundation for Intelligent Physical Agents) est une organisation à but non lucratif fondée en 1996 dont l'objectif est de produire des standards pour l'interopération d'agents logiciels hétérogènes. Par la combinaison d'actes de langages, de logique des prédicats et d'ontologies publiques, la FIPA cherche à offrir des moyens standardisés permettant d'interpréter les communications entre agents de manière à respecter leur sens initial, ce qui est bien plus ambitieux que XML, qui ne standardise que la structure syntaxique des documents. Afin d'atteindre ce but, le FIPA émet des standards couvrant :

- Les applications (applications nomades, agent de voyage personnel, applications de diffusion audiovisuelles, gestion de réseaux, assistant personnel…) ;
- Les architectures abstraites, définissant d'une manière générale les architectures d'agents ;
- Les langages d'interaction (ACL), les langages de contenu (comme SL, CCL, KIF ou RDF) et les protocoles d'interaction ;
- La gestion des agents (nommage, cycle de vie, description, mobilité, configuration);
- Le transport des messages : représentation (textuelle, binaire ou XML) des messages ACL, transport (par IIOP, WAP ou HTTP) de ces messages.

Ces standards évoluent, et sont régulièrement mis à jour, ainsi que de nouveaux standards qui sont nouvellement proposés. Les standards qu'édicte la FIPA ne constituent pas vraiment une plate-forme de construction multi-agents. Ce n'est pas non plus l'objectif que s'est fixé la FIPA. Tout au plus, la FIPA normalise une plate-forme d'exécution standardisée dans un but d'interopérabilité. Ces normes s'appliquent donc pour la plupart en phase de déploiement. Elles n'abordent pas les phases d'analyse ni de conception. Elles peuvent cependant guider certains choix d'implémentation.

V.2.4) Technologie de raisonnement sémantique

La découverte de Web services se base très précisément sur la capacité à pouvoir effectuer des inférences sur des concepts liés au domaine ou catégorie associé à un Web service ainsi qu'à ses paramètres. Pour cela, il fallait pouvoir disposer d'un moteur d'inférences pouvant déterminer et mesurer la distance entre les différents concepts.

La principale caractéristique de ce moteur devait être qu'il puisse travailler nativement avec des ontologies de type OWL DL.

Racer[10] (Renamed A-box and Concept Expression Raisoner) développé par Volker Haarslev, professeur associé à l'université de Concordia (Montreal Canada) et Ralf Möeller, professeur à l'université de science et technologie (Hambourg Allemagne) s'est vite imposé comme la solution la plus intéressante. Racer est le seul moteur d'inférence à disposer nativement de la possibilité de raisonner sur des ontologies au format OWL DL et d'être accessible de manière externe. Cela permet de limiter la complexité du client et grâce à cette approche modulaire de pouvoir facilement passer à une architecture distribuée.

Le principal inconvénient de Racer est qu'il n'est pas open source mais est gratuit dans le cadre d'une utilisation non commerciale. La version de Racer utilisée a évolué de la 1.7.15 à la version 1.7.21.

V.2.5) Plateforme d'hébergement des Services Web et des fichiers OWL

En matière de plateforme Web Services, nous choisissons Axis (Apache eXtensible Interaction System) proposé par la fondation Apache qui est une implémentation du protocole SOAP pour Java. Axis permet d'implémenter de manière transparente des Web services écris en Java au sein du serveur d'application J2EE Apache Tomcat et de les déployer. Il en résulte un plateforme Web hybride supportant à la fois des requêtes HTTP et des requêtes SOAP.

La plateforme se veut donc à la fois un serveur Web délivrant des pages HTML dynamiquement au moyen de servlets et de JSP (JavaServer Pages) et une plateforme d'hébergement de Web services.

La version de Tomcat utilisée est la 5.0 et la version d'Axis est la 1.1 compilé avec la version de ANT 1.6.1 (ANT est principalement utilisé pour automatiser la construction de projets en langage Java).

[10] http://www.racer-systems.com

V.2.6) Technologie de l'implémentation du registre des descriptions

Dans notre architecture, les descriptions sémantiques des services Web sont sauvegardées dans le registre jUDDI[11] (open source Java implementation of the Universal Description, Discovery, and Integration).

Nous avons retenu l'implémentation jUDDI de la fondation Apache pour sa faible charge réseau, sa facilité de déploiement et son adoption dans le milieu industriel et peut être utilisé avec n'importe quelle base de données relationnelles supportant le standard ANSI SQL comme par exemple : MySQL, JDataStore,…etc. Dans notre cas jUDDI est utilisé avec MySQL.

Nous utilisons l'annuaire jUDDI pour stocker les informations non-seulement syntaxiques, mais aussi sémantiques, liées aux services disponibles sur le réseau (d'une entreprise, d'un champ de bataille, …). Ces informations sémantiques étant alors exprimées par le biais de déclarations de services au format OWL-S et d'ontologies de référence liées en partie au domaine d'application du service.

Il s'avère malheureusement que la spécification UDDI ne prévoit aucune facilité pour le stockage d'informations sémantiques dans l'annuaire. Pour pallier cette déficience, nous avons utilisé les capacités d'extensions du modèle de données UDDI pour mettre au point une correspondance (ou « mapping ») entre les déclarations au format OWL-S et les structures de données de l'annuaire.

Écrire une correspondance entre un domaine A et un domaine B signifie alors que l'on définit un procédé pour représenter toutes les structures de donnée du domaine A dans les structures de donnée du domaine B. Il doit être aussi possible de reconstituer tout ou partie des données de type A à partir des données mappées dans le type B.

Cette correspondance va permettre d'effectuer la transition d'une gestion des services basée uniquement sur la syntaxe et mise en oeuvre avec UDDI à une gestion basée principalement sur la sémantique et mise en oeuvre grâce à OWL-S et une couche de compatibilité sémantique pour UDDI (voir figure 52)

[11] http : //WS.apache.org/JUDDI

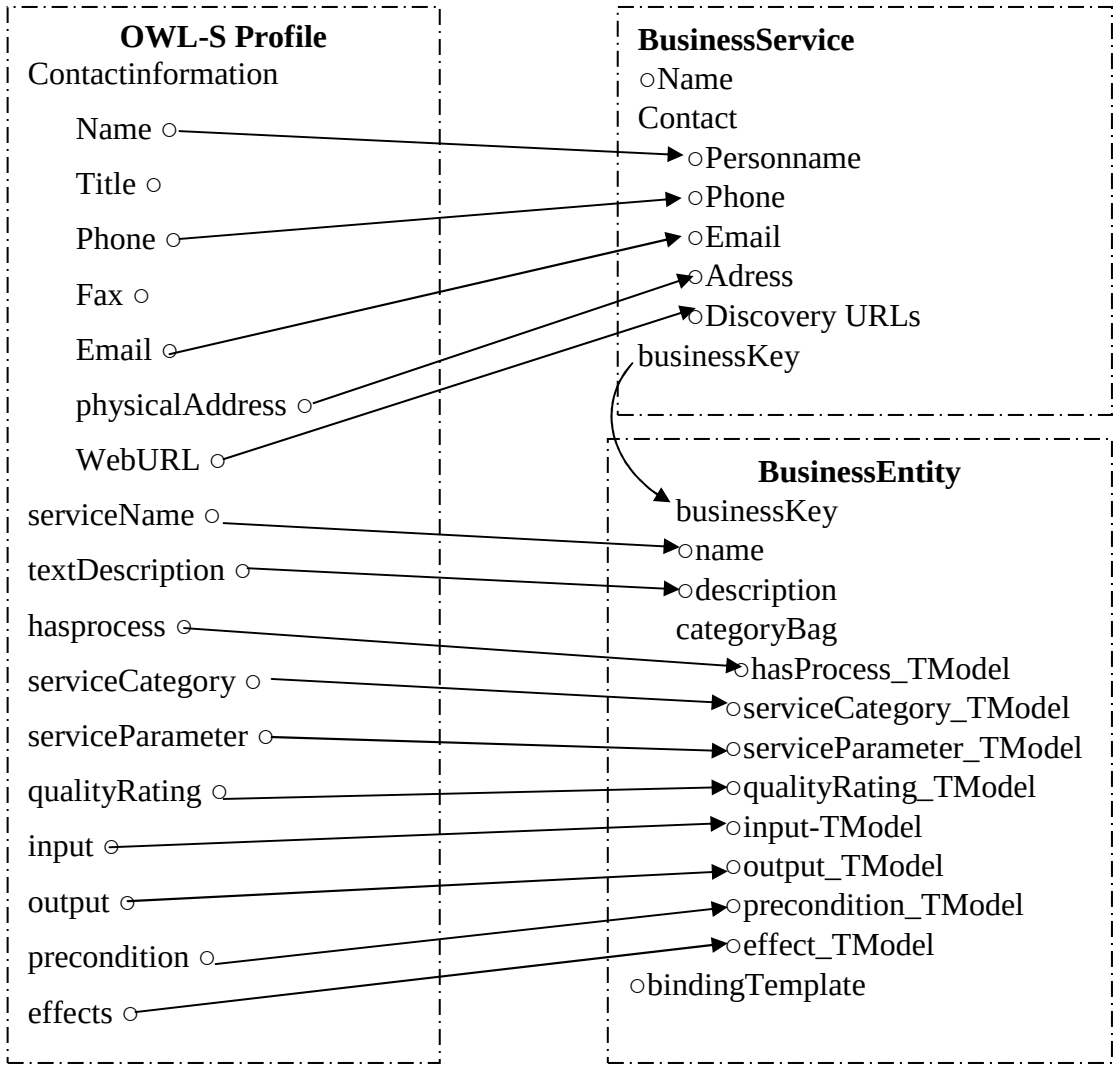

Figure 52- Mapping entre OWL-S et UDDI

V.3) Développement de la partie publication des services Web

Cette partie de l'architecture consiste à intégrer un système développé à l'institut de robotique de l'université de Carnegie Mellon[12].

Appelé OWL-S/UDDI Matchmaker, ce système est composé d'une interface pour la publication des profils OWL-S des services Web, d'une implémentation du registre UDDI pouvant sauvegarder les descriptions sémantiques des services. Pour permettre une recherche sémantique au niveau du registre, le serveur UDDI intègre le moteur de raisonnement RACER.

RACER permet de calculer le niveau de correspondance sémantique entre la requête d'utilisateur et les descriptions des services en terme d'entrées attendus et sorties produits en se basant sur l'algorithme de matching.

[12] http://www.daml.ri.cmu.edu

V.3.1) Interface de publication

L'interface de publication permet aux fournisseurs de services de construire des descriptions OWL-S de leurs services. Ces descriptions sont ensuite soumises au registre UDDI. Pour pouvoir enregistrer une description OWL-S dans UDDI une mise en correspondance doit être faite entre les champs de la description OWL-S et ceux du registre UDDI. Cette mise en correspondance est réalisée en utilisant un script JavaScript au niveau du navigateur Client.

L'interface permet de publier le profil OWL-S d'un service Web via plusieurs champs. Ces champs sont organisés selon le type d'informations qu'ils fournissent (voir figure 53).

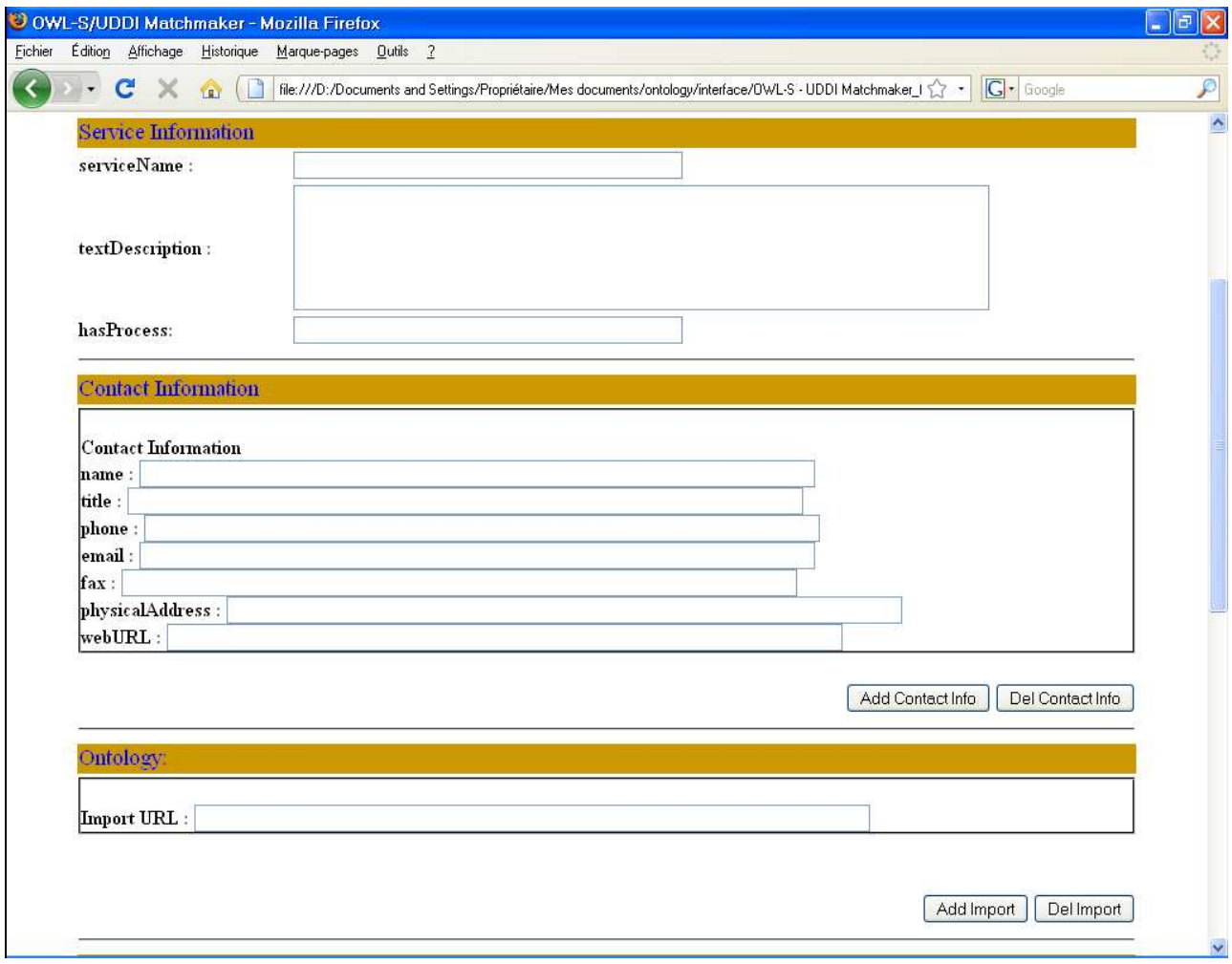

Figure 53 – Capture d'écran de l'Interface de publication des descriptions des services Web

Champs de l'interface

- *Informations sur le service*

  Cette partie de la description permet d'introduire des informations sur le service via trois champs :

  -Nom de service : permet d'introduire le nom de service Web.

  -Description textuelle : ce champ permet d'introduire une brève description du service.

-Processus du service : ce champ est réservé à l'URL du processus qui représente le service.

- *Informations de contact*

  Cette section de la description permet d'introduire des informations de contact relatives au fournisseur de services tel que le nom du fournisseur, fax, adresse mail, URL site Web du fournisseur, adresse….

- *Informations sur les ontologies importées*

  En définissant les fonctionnalités du service, il est nécessaire de définir les classes d'ontologies utilisées. Pour ce fait, un champ permet d'indiquer les URLs des ontologies des classes utilisées afin de les importer.

- *Les entrées attendus du service*

  Cette partie permet de spécifier les entrées du service Web. Elle est constituée de deux champs :

  -Nom du paramètre : qui indique le nom du paramètre d'entrée,

  -Type du paramètre : permet d'indiquer le type (classe OWL) du paramètre d'entrée.

- *Les sorties produits du service*

  Cette partie permet de spécifier les résultats produits en sortie par le service Web. Elle est constituée de deux champs :

  -Nom du paramètre : qui indique le nom du paramètre de sortie,

  -Type du paramètre : permet d'indiquer le type (classe OWL) du paramètre de sortie.

- *Catégorie du service*

  Cette partie permet au fournisseur de spécifier la catégorie de son service sur la base de taxonomie.

V.4) Conclusion

La meilleure manière pour valider une conception d'un système est de l'implémenter en utilisant les outils disponibles et adéquats au problème posé.

Dans ce chapitre, nous avons présenté les principaux outils de l'implémentation. Nous avons commencé par l'argumentation de l'utilisation de JAVA comme un langage de programmation et JADE comme une plateforme muti-agents.

CONCLUSION ET PERSPECTIVES

Les services Web sont la dernière technologie adoptée pour l'intégration et l'interopérabilité des systèmes répartis. Ils sont caractérisés par leurs indépendances aux plateformes et aux systèmes d'exploitation, ce qui a impliqué leur adoption par les différentes organisations commerciales et industrielles offrant leurs services à travers le Web, et par conséquence l'augmentation du nombre de services offerts.

La découverte de services Web constitue un axe de recherche émergent. Diverses approches ont été proposées. Ces approches sont passées d'une recherche basée mots-clés (découverte syntaxique) aux méthodes basées sémantiques. Nous avons proposé aussi une approche à base d'agents qui modélise la découverte des services Web sémantiques.
L'implémentation du test de subsomption réalise la correspondance entre la requête et les services publiés, la liste des résultats est classée par niveaux de correspondance : on trouve les services qui présentent un matching exact et d'autres qui présentent une faible correspondance.

Notre architecture à base agents est composée :

- d'un agent interface fournisseurs services Web.
- d'un agent interface utilisateurs.
- d'un agent enregistreur de service Web au sein de registre UDDI.
- d'un agent de découverte de(s) service(s) Web.

L'agent de découverte de services Web applique des inférences pour apparier la requête de l'utilisateur avec les services offerts. L'appariement (matching) repose sur la comparaison des sorties et des entrées de la requête avec les sorties et les entrées du service.
L'appariement présente différents niveaux de matching : exact, plugin, subsume et fail.

- Le matching exact signifie que les concepts de la requête et des services sont équivalents.
- Le matching plugIn signifie que le service est plus général que la requête.
- Le matching subsume signifie que le service est plus spécifique que la requête.
- Et enfin, le résultat « fail » indique une disjonction entre la requête et le service.

A court terme, nous allons valider de l'implémentation proposée.

En ce qui concerne les perspectives de notre travail, nous prévoyons les points suivants :

- En ce qui concerne l'algorithme de matching on pourra prévoir d'autres paramètres de recherche tels que les préconditions et les effets, ces derniers augmentent les taux de précision, mais ils ont pour prix l'augmentation de la complexité de l'algorithme de matching, et donc le temps de réponse.
- Proposer un matching indirect en cas d'absence de matching direct c'est-à-dire passer à l'étape de composition de services.
- Nous pourrons essayer d'utiliser d'autres types d'agents comme les agents mobiles et évaluer leur effets sur les performances (temps de réponse, espace mémoire et autres) et surtout si elles offrent des possibilités flexibles en matière de programmation.
- Proposer les algorithmes génétiques (algorithmes d'optimisation) comme moyen de correspondance en codant les capacités des services (Inputs, outputs) en terme de chromosomes et en sélectionnant les plus proches à la requête.

BIBLIOGRAPHIE

[Ank, 2003] Ankolekar, A., Burstein, M., Hobbs, J., Lassila, O., Martin, D., McIlraith, S., Narayanan, S., Paolucci, M., Payne, T., Sycara, K., Zheng, H., Daml-s semantic markup for web services», In Proceedings of International Semantic Web Conference (ISWC), (Sardinia, Italy, 2003), pages 348–364, 2003.

[Atul, 2004] Atul Sajjanhar, Jingyu Hou, and Yanchun Zhang, «Algorithm for Web services matching», In APWEB, pages 665-670, 2004.

[BEL, 1999] F. Bellifemine, A. Poggi, and G. Rimassa, «JADE - A FIPA-compliant agent framework», CSELT internal technical report. Part of this report has been also published in Proceedings of PAAM'99, London, pages 97-108, April 1999.

[Ber, 2001] Berners-Lee, T., Hendler, J. et Lassila, O. (2001), «The Semantic Web», In Scientific American, pages 35-43, May 2001.

[Bert, 2002] Bertrand Sajus, «La fonction Thésaurale au coeur des syst`emes d'information», ADBS, www.adbs.fr/adbs/prodserv/jetude/html/prog-110402a.html, avril 2002.

[Bond, 1990] A. H. Bond, «Commitment: A Computational Model for Organizations of Cooperating Intelligent Agents», In Proceedings of the 1990 Conference on Office Information Systems, Cambridge, pages 21-30, 1990.

[Boo, 2004] Booth, D., Haas, H., McCabe, F., NewCorner, E., Champion, M., Ferris, C., and Orchard, D. W3c working group note - «web services architecture», February 2004.

[Bouz, 2007] Bouzguenda L., «Coordination Multi-Agents pour le Workflow Inter Organisationnel Lâche»,Thèse de Doctorat,Université de Toulouse1, mai 2006.

[Bry, 2002] Bryson, J., Martin, D., McIlraith, S., and Stein, L., «Agent-based composite services in daml-s : The behavior-oriented design of an intelligent semantic web », Springer Verlag, 2002.

[Cha-Di, 2002] Chaib-Draa B. and F.Dignum, « Trends in Agent Communication Language», Journal:Computational Intelligence, 2002.

[Cha, 1994] B. Chaib-draa, «Distributed Artificial Intelligence », An overview. In A. Ken, J. G. Williams, C. M. Hall, and R.Kent, editors, Encyclopedia Of Computer Science And Technology, volume 31, pages 215-243. Marcel Dekker, Inc, 1994.

[Cha, 2002] Jean-Marie Chauvet , «Services Web avec SOAP, WSDL, UDDI, ebXML.. », Eyrolles, 2002.

[Che, 1992] V. Chevrier, «Coordination et structuration des échanges par négociation dans les systèmes multi-agents », In Journée Systèmes Multi-Agents PRC-GDR Intelligence Artificielle, Nancy, décembre 1992.

[Cru, 2002] Crusson Tanguy, «Les Services Web » ,DEVOTEAM, 2002.

[Dan, 2003] Daniel Jérome, «Service Web. Concepts,techniques et outils », Vuibert, Paris, 2003.

[Dav, 2004] David Martin, Al., «OWL-S : Semantic Markup for Web services », Technical report, W3C, 2004.

[Da, 1980] R. Davis. Report on the Workshop on Distributed AI. SIGART Newsletter 73, pages 42-52, 1980.

[Did-Tan, 2001] Didier Girard ,Tanguy Crusson , « XML pour l'entreprise », http://www.application-servers.com/livresblancs/xml/, 2001.

[Dro, 1993] A.Drogoul, « De la simulation multi-agent à la résolution collective de problèmes. Une étude de l'émergence de structures d'organisation dans les systèmes multiagents». Thèse de doctorat, Université Paris 6, 1993.

[Dur, 1987] E. H. Durfee, V. R. Gasser, and D. D. Korkill., «Coherent Cooperation Among Communicating Problem Solvers », IEEE Transactions on Computers C-36, pages 1275-1291, 1987.

[Dur, 1990] E. H. Durfee and T. A. Montgomery, «A Hierarchical Protocol for Coordinating Multiagent Behaviors », In Proceedings of the 8th National Conference on Artificial Intelligence, Cambridge, Volume One, pages 86-93, July-August 1990.

[Fer, 1988] J. Ferber and M. Ghallab, «Problématiques des univers multi-agents intelligents », In Actes des journées nationales PRC-GRECO Intelligence Artificielle, Toulouse, pages 295-320, mars 1988.

[Fer, 1995] Ferber J., «Les systèmes multi-agents: Vers une intelligence collective », InterEditions, 1995.

[Fer,1990] J. Ferber, «Conception et Programmation par Objets », Technologies de Pointe. Informatique, Hermes, 1990.

[Fig, 2007] http://www.figoblog.org/document1057.php, 2007.

[Fik, 1971] Fikes, R., and Nilsson, N., «STRIPS: A New Approach to the Application of Theorem Proving to Problem Solving», Artificial Intelligence, 2(3/4), pages 189-208, 1971.

[Fin, 1994] Finin, T., Fritzson, R., McKay, D., McEntire, R. « KQML as an Agent Communication Language », Proceedings of the 3rd International CIKM, USA. ACM Press, pages 456–463, 1994.

[Fipa, 1999] FIPA: Foundation for Intelligent Physical Agents, « Agent Communication Language », FIPA 99 Specification Draft, 1999.

[Gin, 1987] M. L. Ginsberg. Decision Procedures, Morgan Kaufmann, pages 3-28, 1987.

[Gur, 1963] Gurvitch G. , «La vocation actuelle de la sociologie », P.U.F, Paris, pages 401-402, 1963.

[Hal, 1984] J. Y. Halpern and Y. Moses., «Knowledge and Common Knowledge in a Distributed Environment », In 3rd ACM Conference Principles of Distributed Computing, pages 50-61, 1984

[Hor, 1999] Horstmann C.S., Cornell G., « Au cœur de Java 2, volume 1 : Notions fondamentales » 2edition, Paris : campus press, page 826, 2001.

[Hun, 1987] M. N. Huhns, «Distributed Artificial Intelligence », Pitman Publishing-Morgan Kaufman, 1987.

[Kaa, 2003] Kaarthik Sivashanmugam,Kunal Verm,Amit P.Sheth,John A.Miller, «Adding semantics to Web services standards» , In ICWS, pages 395-401, 2003

[Kon, 1991] K. Konolige., «Agents with Attitudes», Invited talk/panel at the AAAI'91, SRI International, 1991.

[Kva, 2004] T.A.Kvaløy, «Semantic Web Services automated Matching, Composition and Selection», Master of Science Thesis Computational Science University of Amsterdam. ,September 2004.

[Le-H, 2005] Le-Hung Vu, Manfred Hauswirth, Karl Aberer, «Towards P2P-Based Semantic Web services Discovery with QoS support », In Business Process Management Workshops, voir lsirpeople.epfl.ch/ hauswirth/papers/-BPS2005.pdf, pages 18-31, 2005.

[Les, 1987] V. R. Lesser and D. D. Corkill., «Distributed Problem Solving Networks», John Wiley and Sons, New York, pages 245-251, 1987.

[Lot, 2008] Lotfi Bouzguenda, Rafik Bouaziz, Eric Andonoff, «Using Ontologies for Coordination in Loose Inter-Organizational Workflow», Dans : International Conference on Research Challenges in Information Science (RCIS 2008), Marrakech, 03/06/2008-06/06/2008, Université Hassan II , pages 123-132, 2008.

[Mar, 2004] Martin, D., Burstein, M., Hobbs, J., Lassila, O., McDermott, D., McIlraith, S., Narayanan, S., Paolucci, M., Parsia, B., Payne, T., Sirin, E., Srinivasan, N., and Sicara, K., «OWL-S : Semantic markup for web services», Tech. rep., France Telecom, MINDL Maryland, NIST, Nokia, 2004.

[Mc, 2004] McCabe, E. Newcomer, M. Champion, C. Ferris, D. Orchard, «Web Services Architecture», W3C Working Group Note , 11 February 2004, (See http://www.w3.org/TR/2004/NOTE-ws-arch-20040211/.)

[Mor, 1977] Morin E., «La Méthode (1): la Nature de la Nature », Le Seuil, Paris, 1977.

[Nav, 2004] Naveen Srinivasan, Massimo Paoulucci, Katia P. Sycara, «An efficient algorithm for OWL-s based semantic search in UDDI», In SWSWPC, pages 96-110, 2004.

[New, 1982] A. Newell, «The Knowledge Level. Artificial Intelligence» ,18(1), pages 87-127, 1982.

[Ouz, 2004] Ouzzani M., «Efficient Delivery of Web services», PhD Thesis, Verginia Polytechnic, USA, 2004.

[Pao, 2002] Paolucci, M., Kawamura, T., Payne, T., Sycara, K., «Semantic matching of web services capabilities», In: Proceedings of the First International Semantic Web Conference, LNCS 2342, Springer-Verlag, pages 333–347, 2002.

[Pao, 2002] Paolucci M., Takahiro K., Terry R. Payne ,Katia P. Sycara, «Semantic matching of Web services capabilities», In ISWC'02 : Proceedings of the First International Semantic Web Conference on the Semantic Web, London, UK, Springer-Verlag, pages 333-347, 2002.

[Paul, 2004] Paul Palathingal and Sandeep Chandra, «Agent approach for service discovery and utilization», IN HICSS, 2004.

[Peg, 1988] C. Pegard, «Coordination de robots mobiles Autonomes, application aux chantiers automatisés», PhD thesis, Université de Picardie, 1988.

[Porn, 2003] Pornpong Rompothong and Twittie Senivongse, «A query federation of UDDI registries», IN ISICT'03 : Proceedings of the 1st international symposium on Information and communication technologies, Trinity college Dublin, pages 561-566, 2003.

[Sear, 1990] J. R. Searle, «Intentions in Communication, chapter 19: Collective Intentions and Actions», MIT PRESS, London, pages 401-415, 1990.

[Siva, 2004] Sivashanmugan K., Verma K., Mulye R.,Zhong Z., Sheth A., «SpeedR: Semantic Peer-to-Peer Environment for Diverse Web Service Registries», http://webster.cs.uga.edu/~mulye/SemEnt/Speed-R.html, 2004.

[Todd, 1997] Todd A. Letsche and Michael W. Berry, «Large-scale information retrieval with latent semantic indexing», Information sciences,100(1-4), pages 105-137, 1997

[Tuan, 2001] TuanAnh Ta., «Web Sémantique et portails —un état de l'art», www.infres.enst.fr/people/saglio/etudes/e-parcours/portails/Sem-Webintro.pdf, 2001.

[Vail, 1987] A. Vailly and M-A. Simon, «Des systèmes experts coopérants, pourquoi, comment ? », In Cognitiva 87 AFCET , Paris, pages 183-188, 1987.

[W3C, 2003] W3C Recommendation SOAP Version 1.2 Part 0, Primer, 24 June 2003.

[Wik, 2007] «Système multi-agents» , Wikipédiafr.wikipedia.org/wiki/Système_multi-agents, 2007.

[Wool, 1995] M. Wooldridge and N. Jennings, «Intelligent agents: Theory and Practice», The knowledge Engineering Review, pages 26-37, 1995.