Remerciements

Tout d'abord, je tiens à remercier les êtres qui me sont les plus chères au monde, mes parents, qui ont toujours cru en moi.

Mes remerciements vont également aux membres de ma famille qui m'ont aidé par leurs conseils et encouragement.

J'adresse mes remerciements spécialement à mon encadreur KAZAR OKBA d'avoir accepté de m'encadrer ainsi pour son aide et soutien et aussi pour ses précieux conseils.

Je voudrais également remercier tous les membres de jury

Mr. LASKRI Mohamed Tayeb, professeur à l'université d'Annaba.
Mr. KIMOUR Mohamed Taher, maître de conférences à l'université d'Annaba.
Mr. BOUDOUR Rachid, maître de conférences à l'université d'Annaba.

Pour m'avoir fait l'honneur d'accepter de juger ce travail.

Je tiens aussi à remercier mes amis qui m'ont toujours encouragé durant la réalisation de ce travail.

Enfin je tiens à remercier tous ceux et celles qui m'ont aidée de prés ou de loin à l'accomplissement de cette thèse.

Table des matières

Table des figures

Résumé

Les techniques actuelles de découverte des services web, effectuent seulement une recherche syntaxique basée sur les mots clés avec une intervention humaine dans presque chaque étape de cycle de vie des services web. L'apparition de web sémantique a muni plusieurs chercheurs à l'exploiter pour palier aux limites actuelles de découverte des services web dans l'UDDI affin de rendre possible la recherche automatique (sémantique) des services Web en se basant sur leurs capacités fonctionnelles et sémantiques tels que leurs inputs, outputs, preconditions, effects, . . .etc dans leurs descriptions. Le web sémantique peut être exploité par un ensemble d'outils et d'applications affin d'optimiser au maximum la plateforme web actuelle. Ces outils offre l'exécution d'une interaction plus complexe qui obéi à des protocoles spécifiques en vue de parvenir à la fonctionnalité attendue. La mise en place de ces outils résulte par l'utilisation des systèmes multi-agents.

Dans ce travail nous avons présenté une architecture basée agents pour mettre en œuvre une nouvelle plateforme basée sur le web sémantique pour effectuer une découverte sémantique des capacités fonctionnelles des services web.

Mots Clés

Services web, Système multi-agents, Web sémantique, Ontologie, Description fonctionnelle, Découverte des services web, OWL-S.

Résumé

Les techniques actuelles de découverte des services web, effectuent seulement une recherche syntaxique basée sur les mots clés avec une intervention humaine dans presque chaque étape de cycle de vie des services web. L'apparition de web sémantique a muni plusieurs chercheurs à l'exploiter pour palier aux limites actuelles de découverte des services web dans l'UDDI affin de rendre possible la recherche automatique (sémantique) des services Web en se basant sur leurs capacités fonctionnelles et sémantiques tels que leurs inputs, outputs, preconditions, effects, . . .etc dans leurs descriptions. Le web sémantique peut être exploité par un ensemble d'outils et d'applications affin d'optimiser au maximum la plateforme web actuelle. Ces outils offre l'exécution d'une interaction plus complexe qui obéi à des protocoles spécifiques en vue de parvenir à la fonctionnalité attendue. La mise en place de ces outils résulte par l'utilisation des systèmes multi-agents.

Dans ce travail nous avons présenté une architecture basée agents pour mettre en œuvre une nouvelle plateforme basée sur le web sémantique pour effectuer une découverte sémantique des capacités fonctionnelles des services web.

Mots Clés

Services web, Système multi-agents, Web sémantique, Ontologie, Description fonctionnelle, Découverte des services web, OWL-S.

Introduction générale

Contexte générale

Avec le temps le Web a commencé à soutenir les interactions humaines avec les données textuelles et graphiques. Les gens utilisent Internet chaque jour à, lire les dernières nouvelles, acheter sur le web...etc. Ce niveau d'interaction est suffisant pour la plupart des gens. Mais, malheureusement le web basé sur les textes statique ne supporte pas bien l'interaction entre logiciels, en particulier les transferts de grandes quantités de données. Le Web actuel doit changer d'orientation d'un Web présentable qui se contente d'afficher les informations disponibles, à un Web "intelligent" permettant aux machines de mieux exploiter les informations.

Les Services web sont devenus ces dernières années prédominant dans la conception des architectures modulaires des systèmes d'informations. Cette prépondérance s'est avant tout affirmée grâce à l'utilisation de standards éprouvés tel que HTTP (Hypertext Transfer Protocol), ou bien encore, XML (Extensible Markup Language) qui permettent une interopérabilité accrue entre plateformes hétérogènes. Aujourd'hui, les Services web sont partout présents en nombres. Qu'il s'agisse de services de réservations en ligne ou bien de gestion de comptes bancaires et même d'applications métiers, tous ces services partagent en commun le fait d'être maintenant accessibles sous forme de Services web.

En outre, un service web est un composant logiciel qui offre des services à travers une interface standardisée. La particularité des services Web réside dans le fait qu'elle utilise la technologie Internet comme infrastructure pour la communication entre les composants logiciels et ceci en mettant en place un cadre de travail basé sur un ensemble de standards.

Le processus de standardisation touche actuellement trois couches du modèle de fonctionnement global : un protocole de communication permettant de structurer les messages

échangés entre les composants logiciels (SOAP), une spécification de description des interfaces des services (WSDL) et enfin une spécification de publication et de localisation de services (UDDI). Actuellement ce modèle supporte principalement des composants logiciels présentant des services sous forme d'une collection d'unités de traitement (opérations) dont l'invocation n'excède pas un échange simple de messages (généralement requête-réponse). A ce niveau de complexité des composants logiciels, le paradigme des services Web est suffisant pour mettre en place des composants interopérables et facilement intégrables.

Toutefois, certains composants logiciels nécessitent l'exécution d'une interaction plus complexe qui obéi à un protocole spécifique en vue de parvenir à la fonctionnalité attendue. Ce type de composant résulte d'une nécessité conceptuelle de la mise en place d'un service (composant agent).

Les systèmes multi-agents sont l'une des dernières générations de systèmes informatiques intelligents, émergeant de la recherche en intelligence artificielle distribuée dans les années quatre-vingt, les systèmes multi-agents constituent aujourd'hui une grande part de la recherche et du développement de l'intelligence artificielle et de la programmation distribuée.

Un autre point de vue sur le Web actuel est qu'il est essentiellement syntaxique, dans le sens que la structure des documents (ou ressources au sens large) est bien définie, mais que son contenu reste quasi inaccessible aux traitements machines. Seuls les humains peuvent interpréter leurs contenus. La nouvelle génération de Web " Le Web sémantique " a pour ambition de lever cette difficulté. Les ressources du Web seront plus aisément accessibles aussi bien par l'homme que par la machine, grâce à la représentation sémantique de leurs contenus.

Les services web sémantiques se situent à la convergence de deux domaines de recherche importants qui concernent les technologies de l'Internet : le Web sémantique et les services web. Le Web sémantique s'intéresse principalement aux informations statiques disponibles sur le Web et les moyens de les décrire de manière intelligible pour les machines. Par contre les services web ont pour préoccupation première l'interopérabilité entre applications via le Web en vue de rendre le Web plus dynamique.

Problématique

Le problème réel des services Web actuel est qu'ils ne représentent qu'un mécanisme de transfert de données d'un système à l'autre, ils n'apportent, en aucun cas, plus de valeurs à l'information déjà possédée. Ils permettent juste une meilleure diffusion auprès des clients et des fournisseurs. L'utilisation d'XML dans l'architecture des services web permet de structurer la construction d'un document et non de spécifier le sens à donner au document.

Concernant le mécanisme de découverte des services web actuels dans le registre UDDI, il est purement syntaxique, c'est-à-dire que les services peuvent être recherchés par exemple par nom, par emplacement ou par tModels. Cependant, cette recherche est restreinte à la détection des mots-clés et ne supporte aucune inférence des informations présentes dans l'UDDI (inputs, outputs, precondition et effect). Ce problème est causé par l'absence de la sémantique dans les technologies de base des services web. Pour résoudre ce problème, les recherches faites sur le Web sémantique ont été exploitées dans le domaine des services Web.

Solution proposée

Pour résoudre la problématique posée nous allons proposer un modèle générique, générale et extensible pour la découverte basée sur la sémantique des descriptions des services web. Pour le faire, nous allons exploiter les solutions proposées dans les recherches sur le web sémantique. Nous allons aussi exploiter l'environnement des systèmes multi-agent, et voir comment l'interopérabilité, la communication et la coopération dans un système multi-agents peut être exploitée pour résoudre ce cas complexe qui est la découverte basée sur la description sémantique des services web tout en minimisant les coûts d'intégration dans des systèmes tel que les services web.

Structure de mémoire

Ce travail est constitué de cinq chapitres.

Dans le premier chapitre, nous présentons un aperçu général sur les services en commençant par une définition générale et en terminant avec la présentation des limites actuelles dans les techniques de découverte des services web.

Dans le deuxième chapitre on va parler des ontologies ainsi que le Web sémantique, nous allons étudier aussi quelques techniques et approches existante dans le domaine de découverte des services web sémantique.

Le troisième chapitre s'intéresse aux systèmes multi-agents SMA, en donnant des définitions, et en montrant les avantages et les objectifs de ces systèmes. A la fin de ce chapitre nous présenterons la relation service web et agent.

Le quatrième chapitre illustre notre approche générale pour la découverte sémantique des services web a base d'agents.

Le chapitre cinq présente une étude de cas, où nous avons présenté la validité de notre modèle sur un exemple réel qui est la vente des téléphones mobiles sur le web.

Chapitre 1

Les services Web

1.1 Introduction

Avec le temps le Web a commencé à soutenir les interactions humaines avec les données textuelles et graphiques. Les gens utilisent l'Internet chaque jour à, lire les dernières nouvelles, acheter sur le web..etc. Ce niveau d'interaction est suffisant pour la plupart des gens. Mais, malheureusement le web basé sur les textes statiques ne supporte pas bien l'interaction entre logiciels, en particulier les transferts de grandes quantités de données. Une méthode plus efficace est nécessaire pour permettre aux applications d'interagir directement les unes avec les autres. Les entreprises qui font du commerce sur le Web doivent trouver un moyen de publier des liens vers les applications et les données qu'ils offres, de la même manière qu'ils publient des liens vers leurs pages Web [ERIC 02].

La technologie des services Web offre une architecture qui change fondamentalement les règles du commerce sur le Web. Cette architecture relie des applications les unes aux autres dans des points éloignés dans le monde à travers des protocoles Internet universels, le résultat est plus rapide, meilleur et une communication plus productive pour les entreprises et les consommateurs. Avec les services Web, Internet se transforme en une plate-forme de composants auto-descriptifs facilement intégrables et faiblement couplés.

Dans ce chapitre nous allons présenter la technologie de service Web, en commencent par définir ce paradigme ainsi que sa différence par rapport aux autres technologies comme CORBA, RMI ou DCOM ... etc. Enfin nous présenterons l'architecture des services Web et leurs technologies de base.

1.2 Les services Web

1.2.1 Définition

A l'origine, la technologie des services web a été initiée par IBM et Microsoft, puis en partie normalisée sous le W3C (World Wide Web Consortium), l'organisme chargé de standardiser les évolutions du web. Les services Web sont des composants logiciels encapsulant des fonctionnalités métier de l'entreprise. Ils peuvent être accessibles depuis une autre

application (un client, un serveur ou un autre service Web) à travers le réseau Internet en utilisant les protocoles standard du Web.

Techniquement, 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. En effet, afin de dialoguer via Internet, ces applications doivent parler le même langage, langage basé sur le XML.

Un service Web est décrit dans un document WSDL (Web Services Description Language), précisant les méthodes pouvant être invoquées, leur signature et les points d'accès du service (URL, Port., etc.). Ces méthodes sont accessibles via le protocole SOAP (Simple Object Access Protocol). Un service Web est une technologie permettant à des applications de dialoguer à distance via Internet par le mécanisme requête/réponse, qui sont des messages XML transportés par HTTP (Hypertext Transfer Protocol), FTP (File Transfer Protocol) ou SMTP (Simple Mail Transfer Protocol). Les services Web sont centralisés (leurs publications) dans un référentiel commun UDDI (Universal Description Discovery and Integration) affin de faciliter leurs recherches [ALEX 02].

Comme le montre la *figure 1.1*, les services Web présentent au réseau une façon d'interface avec un arrière plan des logiciels, tels que les systèmes de gestion de bases de données, .NET, J2EE (Java2 Platform, Enterprise Edition), ou CORBA (commune Object Request Broker Architecture), des objets, des adaptateurs d'Enterprise Resource Planning (ERP) et d'autres. L'interface de service Web reçois un message XML de l'environnement de réseau, ensuite elle transforme les données XML en un format compréhensible par les logiciels et éventuellement retourner un message de réponse. Le logiciel sous-jacent un service Web peut être créé en utilisant n'importe quel langage de programmation, système d'exploitation ou système de middleware [ERIC 02].

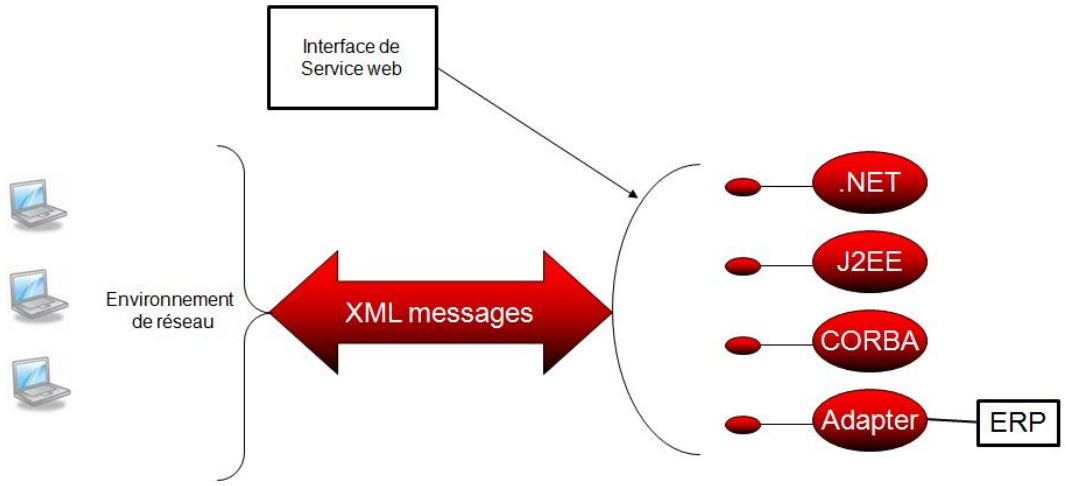

Figure 1.1 – Les protocoles du service Web

1.3 Les services Web et les autres technologies

Les Services Web ont été conçus pour répondre en premier lieu à des problèmes d'interopérabilité sur l'Internet, que les middlewares été incapables de résoudre de façon satisfaisante.

Un middleware *est un logiciel de communication qui permet à plusieurs processus s'exécutant sur une ou plusieurs machines d'interagir à travers un réseau [ALLI 03].*

La figure suivante (Figure 1.2) montre le vecteur d'évolution des middlewares jusqu'à apparition des Web services.

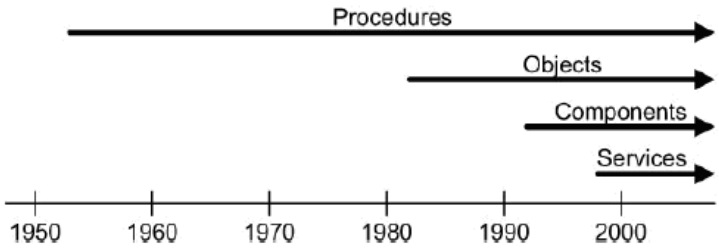

Figure 1.2 – Evolution des middlewares

RPC (Remote Procédure Call), Sun RPC est un protocole d'abord développé par Xerox, puis employé par Sun, qui s'est aujourd'hui généralisé dans le monde UNIX. C'est un des plus vieux middlewares, il utilise le principe d'appel de procédure type client/serveur

s'exécutant sur une machine distante dans un environnement d'applications distribuées et fonctionne de manière synchrone, il est facile à comprendre et à coder. Mais nécessite beaucoup de ressources et complexe à administrer et ne supporte pas la programmation orienté objet [RAHE].

CORBA (Common Object Request Broker Architecture), CORBA est une norme de communication crée par l'OMG (Object Management Group), elle se base totalement sur le paradigme de la programmation objet et utilise le principe d'RPC. CORBA permet l'interopérabilité quelque soit le langage ou le système (C, C++, Java, ADA, etc) [ALLI 03], car elle utilise un langage IDL (Interface Définition Language) qui décrit les traitements effectués et les formats de données en entrée et en sortie, ainsi q'un bus applicatif, ORB (Object Request Broker) qui établit les relations entre les différents objets distribués sur le réseau [RAHE]. La première version de CORBA n'était pas interopérable. La seconde version de CORBA définit un protocole (IIOP Internet to Internet Operating Protocol) qui est censé permettre l'interopérabilité. Cependant, cette dernière reste encore délicate. CORBA reste une solution complexe à appréhender et à mettre en place. Elle ne peut pas gérée le problème de pare-feu.

COM/DCOM (Comportent Object Model), Modèle de Microsoft pour le développement de composants logiciels réutilisables, orientés objet et indépendants du langage de programmation. C'est une solution simple d'utilisation, mais malheureusement elle est spécifique à Microsoft, Aussi, elle présente l'inconvénient qu'elle n'est pas portable au niveau de code-sources [RAHE].

Java RMI (Remote Method Invocation), Solution native de Java. Le but de RMI est de permettre l'appel, l'exécution et le renvoi du résultat d'une méthode exécutée dans une machine virtuelle différente de celle de l'objet l'appelant. Cette solution présente l'avantage qu'elle est plus simple que le développement des sockets JAVA, supporte la POO (Programmation Orienté Objet) avec une transparence dans les communications entre objets. Cependant cette solution présente l'inconvénient qu'elle est limitée à la plate-forme JAVA et

aussi, c'est une architecture fortement couplée [RAHE].

Tous ces middlewares ont été développés à une époque où les systèmes distribués étaient limités à un réseau local ou privé d'une entreprise, d'un gouvernement ou d'une université. Ils ont été conçus pour répondre à des exigences spécifiques et dans un contexte bien limité. A la fin des années 90, les développeurs du monde informatique se sont aperçus que les solutions précédentes ne convenaient pour supporter les systèmes distribués sur l'Internet. Plusieurs initiatives ont donc démarré pour créer une nouvelle solution plus adaptée à l'Internet. En fait les développeurs du monde informatique sont rendus compte qu'un nouveau formalisme pour représenter l'information n'était pas suffisant pour développer les applications dans le contexte de l'Internet, il fallait aussi changer de paradigme. Ainsi le paradigme service a été davantage développé et expérimenté pour prendre en charge les exigences de l'Internet.

1.4 Une architecture orienté service

Les architectures en informatique relèvent plus de la philosophie que de la spécification. Elles proposent une vision sur le fonctionnement et la dynamique des systèmes.

***L'architecture orienté service (SOA)**, est un modèle abstrait qui définit un système par un ensemble de composants logiciels distribués qui fonctionnent affin de réaliser une fonctionnalité globale préalablement établie* [HEAT 01].

Dans une SOA, les services peuvent communiquer entre eux. Cette communication peut soit consister en un simple passage de données ou impliquer la coordination de deux ou plusieurs composants pour l'accomplissement d'une activité. Le choix d'une architecture SOA entre dans la perspective de transformer le Web en une énorme plate-forme a composants (services) faiblement couplés et automatiquement intégrables.

Beaucoup considèrent que la première SOA est apparue avec l'utilisation des DCOM ou ORB basés sur les spécifications de CORBA. Cependant, la principale différence entre SOA et les autres architectures distribuées, telles que CORBA, est le faible couplage des composants par l'expression des "interactions" entre les services. En effet, le point de départ

des spécifications SOA a été le besoin croissant de sélectionner et d'intégrer, au fil de l'eau, des services hétérogènes, aussi bien à travers le web qu'au sein d'environnements intelligents.

1.5 Architecture des services Web

1.5.1 Présentation

L'architecture des services Web (figure 1.3) est une instance d'architecture orientée service SOA. La définition d'architecture pour les services Web (WSA : Web Services Architecture), consiste à mettre en évidence les concepts, les relations entre ces concepts ainsi qu'un ensemble de contraintes qui assurent l'objectif premier des services Web à savoir l'interopérabilité. Ces éléments sont structurés à travers un modèle de déploiement et de fonctionnement des services Web. Les concepts de l'architecture ne sont pas supposés avoir une existence dans la réalisation informatique, ils désignent des personnes, des organisations ou encore une entité logicielle ou matérielle. La définition ou la description des concepts de l'architecture se présente sous forme d'une identification nominale du concept ainsi que les relations qu'il entretient avec les autres concepts. Les principaux concepts intervenant dans l'architecture des services Web sont :

- ***Le fournisseur du service,*** d'un point de vue conceptuel, il désigne la personne ou l'organisation responsable juridiquement de l'agent logiciel (i.e. le service). D'un point de vue opérationnel, les fournisseurs de services peuvent également désigner le serveur qui héberge les services déployés.
- ***Le client du service,*** comme pour le fournisseur, il représente une personne ou une organisation, client potentiel des services. D'un point de vue opérationnel il désigne, l'application cliente qui invoque le service.
- ***L'annuaire des services,*** il représente la personne ou l'organisation responsable de l'hébergement ou la publication des services. D'un point de vue fonctionnel, il représente l'entité logicielle qui joue le rôle de l'intermédiaire entre les clients et les fournisseurs de services. Le concept registre ou annuaire de service est essentiel dans l'architecture services Web.

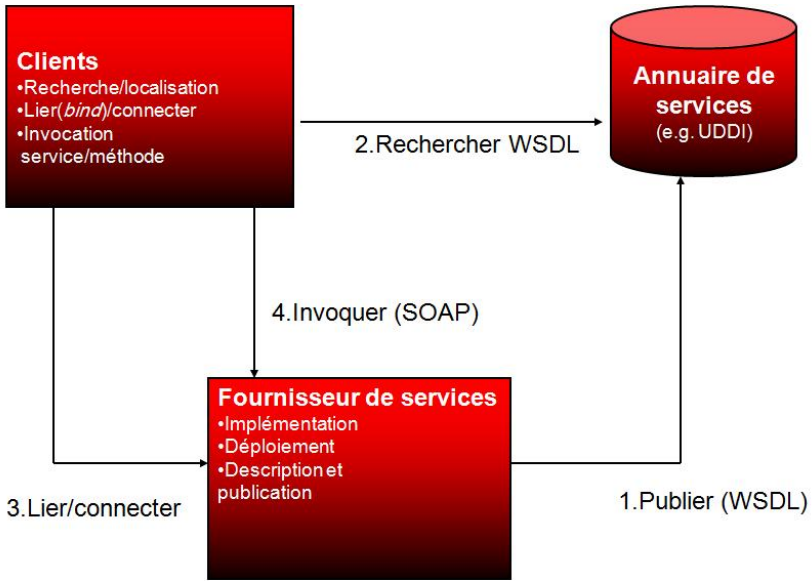

Figure 1.3 – Architecture des services Web

1.5.2 Technologies de base des services Web

Pour garantir l'interopérabilité des trois concepts précédentes (Fournisseurs de services, Client du service, Annuaire de publication), des propositions de standards (technologie pour les services Web) ont été élaborées : XML (Extensible Markup Language), SOAP (Simple Object Access Protocol), WSDL (Web Services Description Language) et UDDI (Universal Description, Discovery and Integration).

- ***XML***, est aujourd'hui un standard qui permet de décrire des documents structurés transportables sur les protocoles communs d'Internet. XML constitue la technologie de base des architectures services web. En effet, il apporte à l'architecture l'extensibilité et la neutralité vis à vis des plates-formes et des langages de développement. De plus, grâce à la structuration, XML permet la distinction entre les données des applications et les données des protocoles permettant ainsi une correspondance facile entre les différents protocoles. L'interopérabilité entre les systèmes hétérogènes demande des mécanismes puissants de correspondance et de gestion des types de données des messages entre les fournisseurs et les clients. C'est une tâche où les schémas de type de données XML s'avèrent bien adaptés (le langage XML sera détaillé dans le chapitre2).

- ***SOAP***, c'est un protocole de transport réseaux, soumis à un format de message XML et permettant a un client d'invoquer un service distant.
- ***WSDL***, c'est un langage de description basé sur XML et utilisé pour décrire l'interface d'utilisation d'un service Web. Le fournisseur de service utilise un document WSDL afin d'indiquer les opérations fournies par un service Web ainsi que les paramètres et les types de données associés à ces opérations.
- ***UDDI***, c'est un protocole d'annuaire basé sur XML qui permet de stocker ou de rechercher des informations sur des services Web. Il permet aussi d'automatiser la découverte des services Web.

1.5.3 Vue globale de la technologie des services web

L'architecture des services Web (WSA), représente une vue globale sur les concepts ainsi que les relations qui les unissent. Chaque concept dans le modèle est défini par l'ensemble des relations qu'il mène avec les autres concepts. Le modèle de l'architecture donne également des indications sur le cycle de vie des services Web. La *figure 1.4* présente une numérotation des relations qui indique les différentes étapes de cycle de vie des services Web : la recherche, découverte et l'invocation d'un service Web.

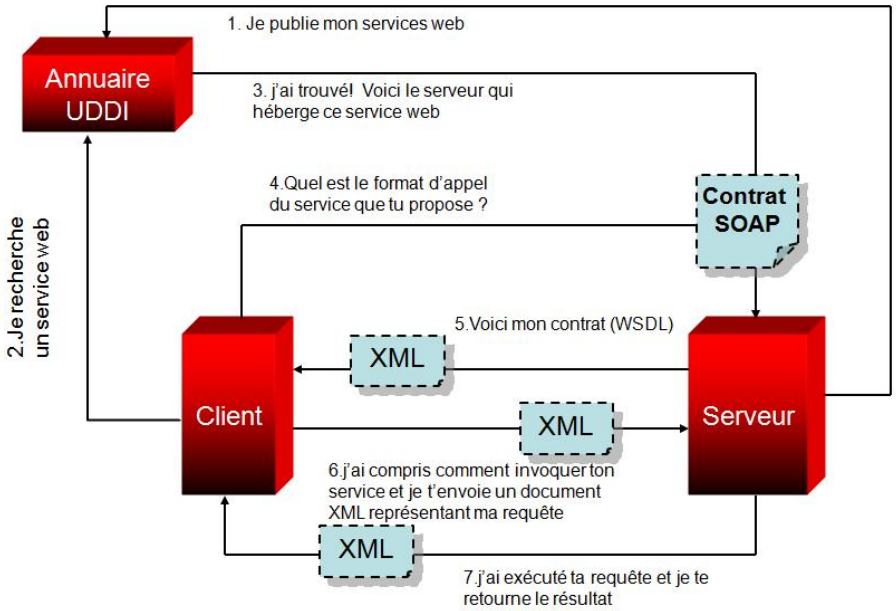

Figure 1.4 – Interopérabilité dans une WSA

Selon la *figure 1.4*, les acteurs demandeur de service, fournisseur de service et annuaire maintiennent une relation par les biais des technologies de service Web, cette relation est comme suit :

1. Le fournisseur de service déploie et publie ses Web Services sur l'annuaire en utilisant UDDI. Cette opération se fait en envoyant directement à l'annuaire un message UDDI (encapsulé dans une enveloppe SOAP) via un protocole de transport. Les informations fournies regroupent la localisation du service, la méthode d'invocation (et les paramètres associés) ainsi que le format de réponse.
2. Un Client recherche un Web service. Cette recherche se fait en envoyant un message UDDI (requête) encapsulé dans une enveloppe SOAP via un protocole de transport.
3. L'annuaire trouve le service désiré, et envoie les informations sur le fournisseur du service et sur le service, le client reçoit (via un protocole de transport) ces informations de l'annuaire. Ces informations sont un message WSDL encapsulé dans une enveloppe SOAP.
4. Le demandeur de service (Client) demande le contrat du Web service au fournisseur ;
5. Le fournisseur envoie le contrat du service (WSDL) ;

6. Le demandeur de service appelle le Web service selon son contrat. La demande de service s'effectue à l'aide d'un message SOAP via un protocole de transport (une requête d'invocation).
7. Après avoir exécuter la requête, le fournisseur envoie au client une réponse du service Web (résultats) via un protocole de transport, sous la forme d'un message SOAP.

1.5.4 Infrastructure des services Web

L'originalité de l'infrastructure des services Web consiste à mettre en place des services en se basant exclusivement sur les protocoles les plus répandus d'Internet. Aujourd'hui, l'infrastructure des services Web s'est concrétisée autour de trois spécifications considérées comme des standards SOAP, WSDL et UDDI.

1.5.4.1 Le transport SOAP

La communication par message constitue un point crucial dans toute architecture SOA. Le protocole SOAP (Simple Object Access Protocol) est un standard qui assure la messagerie (figure 1.5), il est basé sur XML, il permet l'échange de données structurées indépendamment des langages de programmation ou des systèmes d'exploitation. Du fait que le protocole SOAP est une spécification XML, il est indépendant du contenu du message transporté, à proprement parler, il laisse la responsabilité de l'interprétation aux couches de communication supérieures [GUDG OO]. Il se contente d'offrir la possibilité de structurer des messages destinés à des objectifs particuliers allant d'un simple échange de données jusqu'à l'appel de procédures à distance. Il peut être employé dans tous les styles de communication, synchrones ou asynchrones, point à point ou multipoint.

L'atout principal du SOAP c'est qu'il permet le passage des messages à travers les pare-feux sans aucun problème parce qu'il utilise HTTP comme protocole de transport. Cet atout constitue aussi un défaut majeur des services Web qui est l'absence de sécurité.

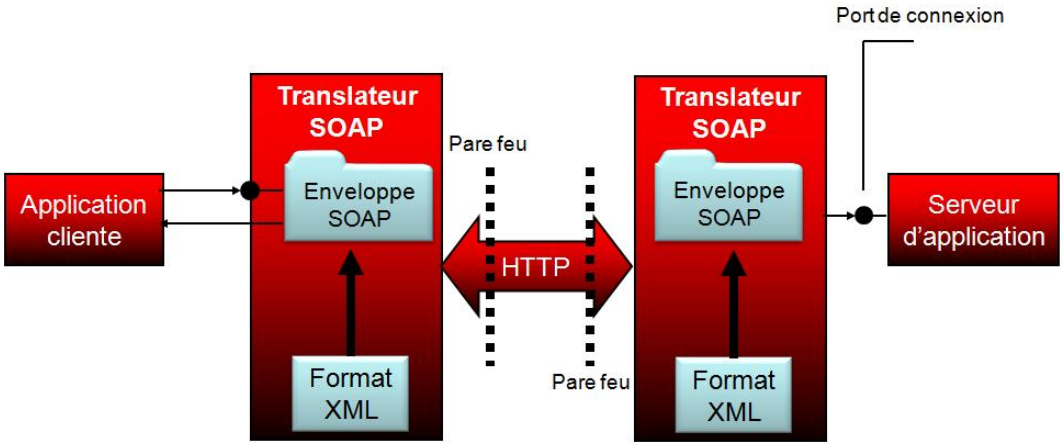

Figure 1.5 – Le transport des messages SOAP

➢ Composants d'un message SOAP

Un message SOAP est constitué de trois parties (figure 1.6) :

- *L'entête du message*, spécifie le protocole support (HTTP, SMTP,.. .etc.).
- *L'enveloppe SOAP*, définit l'espace de nom (namespace : URI permettant de connaître la provenance de chaque balise) précisant la version supportée de SOAP. L'enveloppe SOAP, lui-même est constitué d'un en-tête facultatif (SOAP header) et d'un corps obligatoire (SOAP body).
- *SOAP header*, peut avoir plusieurs fils (SOAP blocks). Ces fils sont utilisés pour ajouter des fonctionnalités au message comme l'authentification et la gestion des transactions. SOAP header peut contenir les attributs suivants : L'attribut Actor, Permet de préciser le destinataire final du message (message path) et l'attribut mustUnderstand qui peut prendre les valeurs "1" ou "0", il Spécifie que le récepteur du message doit obligatoirement comprendre cet élément. Si ce n'est pas le cas, le récepteur arrêté tout traitement.
- *SOAP body*, contient l'information destinée au receveur. Il doit fournir le nom de la méthode a invoquée par une requête, ou le nom de la méthode pour générer la réponse. Il doit aussi, fournir l'espace de nom correspondant au nom du service. Le SOAP body, peut contenir aussi un attribut SOAP fault. Ce bloque est utilisé pour transmettre l'information des erreurs durant le traitement du message.

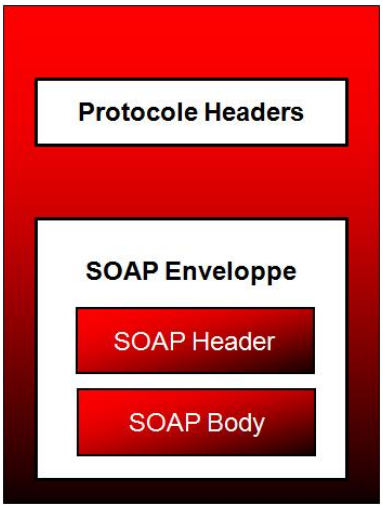

Figure 1.6 – Format d'un message SOAP

➢ Le transport sur HTTP

HTTP (HyperText Transfer Protocol) est le protocole défini pour le Web au dessus de la couche TCP/IP. Il est aussi le protocole le plus utilisé pour le transport des messages SOAP. Il constitue un excellent moyen de transport des données en raison de sa popularité sur le Web ainsi que sa simplicité. Il existe deux règles à respecter pour pouvoir utiliser SOAP via HTTP.

- Le mécanisme d'envoi doit utiliser la méthode HTTP POST standard.
- La destination de la requête HTTP doit être adressée à une URI donnée sur le serveur Web.

Le bloc de données de la requête HTTP est composé du message SOAP lui-même (par conséquent le *Content-Type* doit être *text/xml*) et d'un attribut *SOAPAction* qui est destiné à indiquer au serveur que le message HTTP transporte un message SOAP.

➢ Exemple

la *figure 1.7*, illustre comment la requête SOAP Somme, est envoyée via SOAP à un service Web pour que ce dernier calcule la somme de deux nombres envoyés dans la requête et retourne le résultat comme réponse a cette requête (figure 1.8). Dans la *figure 1.7*, les quatre premières lignes du code représente l'entête http standard. Les lignes suivantes définissent la

méthode à invoquer. L'élément enveloppe SOAP constitue le premier élément de document XML, il sert à distinguer les identifiants SOAP, des identifiants spécifiques à l'application. La méthode à invoquer est spécifiée dans l'élément body du message SOAP.

```
POST /path/foo.pl HTTP/1.1
Content-Type: text/xml
SOAPAction: interfaceURI#Add
Content-Length: nnnn

<soap:Envelope xmlns:soap='uri for soap'>
 <soap:Body>
  <Add xmlns='interfaceURI'>
   <arg1>24</arg1>
   <arg2>53.2</arg2>
  </Add>
 </soap:Body>
</soap:Envelope>
```

Figure 1.7 – Requête SOAP encapsulée dans une requête HTTP

```
200 OK
Content-Type: text/xml
Content-Length: nnnn

<soap:Envelope
   xmlns:soap='uri for soap'>
 <soap:Body>
  <AddResponse xmlns='interfaceURI' >
   <sum>77.2</sum>
  </AddResponse>
 </soap:Body>
</soap:Envelope>
```

Figure 1.8 – Réponse SOAP encapsulée dans une requête HTTP

1.5.4.2 L'interface WSDL

WSDL (Web Services Description Language), est un langage qui permet de décrire les services web, et en particulier, les interfaces des services Web, Il est né de la convergence

de NASSL (Network Accessible Service Spécification Language) d'IBM et de SDL (Service Description Language) de Microsoft. Ses descriptions sont des documents XML, présentant une interface d'utilisation du service Web.

Un document WSDL décrit de manière indépendante du langage de programmation, l'ensemble des fonctionnalités offertes par un service. Il permet de connaître les protocoles, les serveurs, les ports, le format des messages, les entrées, les sorties, les exceptions possibles et les opérations réalisées par un service web. Un document WSDL est accessible depuis une URL. L'interface d'un document WSDL est similaire aux interfaces des technologies CORBA et DCOM.

➢ Composants d'un document WSDL

Un document WSDL décrit un service Web en deux parties : une partie abstraite et une concrète. C'est à dire, La séparation de "quelle" fonctionnalité est fournie de "comment" et "où" celle-ci est offerte. La *figure 1.9*, présente la structure fondamentale d'un document WSDL.

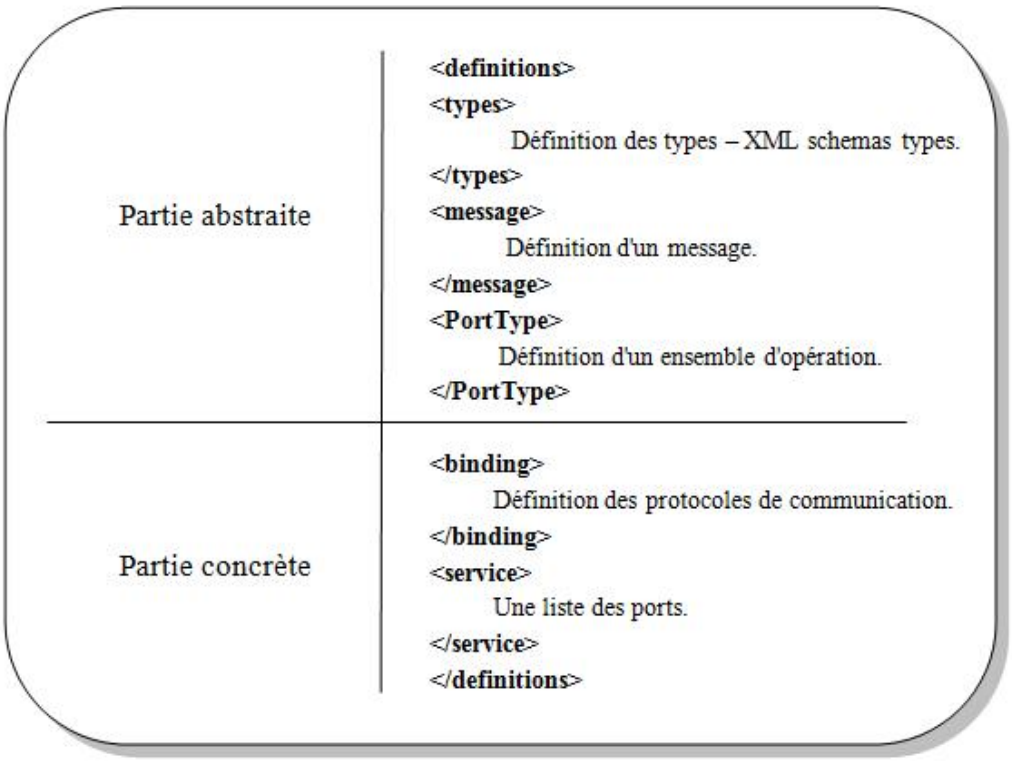

Figure 1.9 – Structure fondamentale d'un document WSDL

La partie abstraite contient :

- **<types>** définit les types de données qui sont échangées par les messages XML. WSDL utilise les schémas XML pour définir les types de données.
- **<message>** cet élément décrit les messages SOAP qui pourrait être des entrées (inputs) ou des sorties (outputs). Un message SOAP est subdivisé en parties représenter par des éléments <part>. Ces dernières peuvent être comparées à des paramètres d'appel de fonctions dans la programmation classique.
- **<operation>** cet élément permet seulement de définir les inputs, outputs et les faux messages associés à l'opération, vous pourrait alors consulter les détails de message pour déterminer quels paramètres d'inputs/outputs sont concernés [JEFF 06].
- **<portType>** : c'est l'élément WSDL le plus important. Il liste toutes les opérations offertes par le service web. L'élément <port> correspond a un seul service web par contre l'élément <portType> décrit les opérations disponibles [JEFF 06].

La partie concrète contient :

- **<binding>** cet élément relais la partie concrète et la partie abstraite de document WSDL. Il est associer a un <portType> spécifique. Il liste l'adresse du service web associé avec l'élément <portType>. Le <binding> liste aussi les protocoles utilisés dans la communication avec le service Web [JEFF 06].
- **<Service>** définit une collection de **<port>**, c'est à dire une collection d'adresses URI permettant d'invoquer un service, il implémente aussi l'élément <binding>.

➢ **Relation entre les composants d'un document WSDL**

Comme illustré dans la *figure 1.10*, chaque document WSDL définit un service comme une collection de points finaux ou ports. Chaque port est associé à une liaison spécifique qui définit la manière avec laquelle les messages seront échangés. Chaque liaison établit une correspondance entre un protocole et un type de port. Ce dernier se compose d'une ou plusieurs opérations qui représentent une définition abstraite des capacités fonctionnelles du service. Chaque opération est définie en fonction des messages échangés au cours de son invocation (inputs/outputs) [MELL 04].

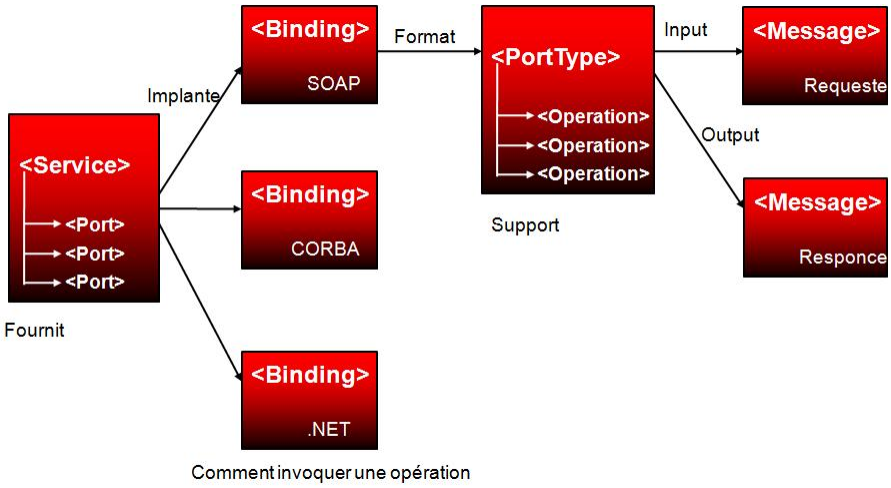

Figure 1.10 – Relation entre les éléments d'un document WSDL

1.5.4.3 Le registre UDDI

Les deux standards exposés précédemment (SOAP, WSDL) définissent ensemble l'aspect le plus basique de développement de l'infrastructure des services Web. Toutefois, dans un environnement ouvert comme Internet, le modèle de description des services Web n'est d'aucune utilité s'il n'existe pas un moyen de localiser aussi bien les services que leurs descriptions WSDL. Un troisième standard a été conçu pour réduire l'écart entre les applications clientes et les services Web, appelé UDDI (Universal Description, Discovery and Intégration).

➢ Présentation

Le projet UDDI a commencé en octobre 2000 par une collaboration entre Microsoft, Ariba, et IBM. D'autres entreprises s'y sont jointes comme Sun Microsystems, Oracle, HP ou encore SAP. UDDI a été conçu pour une utilisation conjointe avec le registre ebXML (Electronic Business eXtensible Markup Language) pour le commerce électronique et qui recouvre l'ensemble des paradigmes proposé par UDDI et WSDL mais avec une méthode appelée UMM (UN/CEFACT's Modeling Methodology) et qui repose sur UML.

UDDI est une technologie qui s'articule autour des protocoles HTTP, SOAP et le langage XML [UDDI 00]. UDDI offre deux fonctionnalités au sein de l'architecture globale qui permet aux fournisseurs de publier leurs services selon un modèle de description et au client la recherche des services publiés. Les services peuvent être organisés dans l'UDDI en

fonction de leur utilisation prévue : ils peuvent être groupés en domaines d'application, par régions géographiques, ou tout autre schéma approprié [CABR 04].

Grâce à UDDI, les entreprises peuvent enregistrer des renseignements concernant les services Web qu'ils proposent et des informations techniques sur le mode d'accès à ces services tel que l'adresse pour accéder au Services Web, mais également des informations beaucoup plus contextuelles, tel le nom de la personne qui s'occupe de leur gestion, la description sommaire de leurs fonctionnalités ou encore le nom et la branche d'activité de l'entreprise dont ils dépendent. L'UDDI offre des APIs aux applications clientes, pour consulter et extraire des données concernant un service et/ou son fournisseur. L'UDDI actuellement est orienté vers la recherche manuelle que automatique, les services dans l'UDDI peuvent etre rechercher par nom, par location, par business par bindings ou par tModels.

La mise en place d'un registre UDDI suit un processus uniforme imposé par la spécification. Chaque organisation qui veut mettre en place un registre UDDI doit suivre ce processus pour devenir un opérateur UDDI. Les registres UDDI créés sont organisés en réseaux. Ils partagent les différentes informations publiées. La publication d'un service chez un opérateur, donne lieu automatiquement à un processus de propagation des informations aux différents registres UDDI. L'accès à l'ensemble d'informations des registres peut se faire par n'importe quel opérateur UDDI.

➢ Composants d'un annuaire UDDI

Les annuaires UDDI contiennent des informations détaillées concernant les services Web et leurs emplacements. Une entrée d'annuaire UDDI se compose de trois parties principales le fournisseur du service, les services Web offerts et les liens vers les implémentations. Chacune de ces parties donne progressivement davantage d'informations sur le service Web.

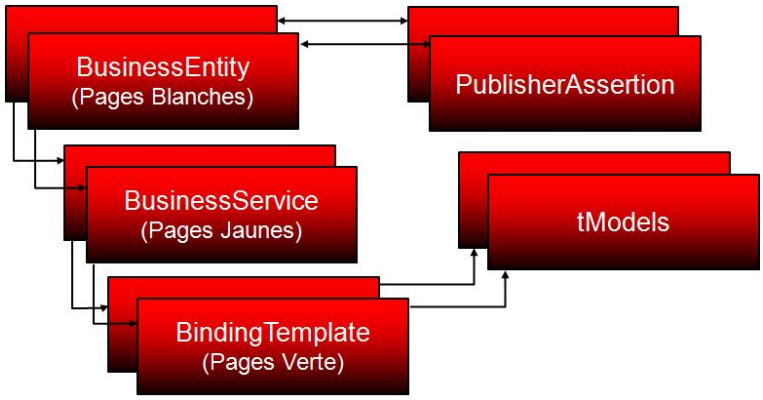

Figure 1.11 – Composants d'un registre UDDI

Business Entity (pages blanches), elles contiennent les éléments relatifs à l'entreprise qui propose le service (nom de l'entreprise, coordonnées, domaine d'activité, la description de l'entreprise mais également l'ensemble de ses identifiants.. .etc.). BusinessEntity est associé à un ou plusieurs BesinessServices qui sont la description des services spécifiques que le fournisseur offre. Il est aussi associé avec un ou plusieurs BindingTemplate qui spécifié les points d'accès finaux.

Business Service (pages jaunes), permet la description groupée d'une série de services Web liées par le domaine d'activité ou le type de services offerts. Il contient des informations relatives au métier de l'entreprise. Il réalise le lien entre une interface de service (tModel) et son implémentation, par l'intermédiaire des sous-éléments bindindTemplates qui se contentent de pointer vers des descriptions externes.

BindingTemplate (pages vertes), i1 contient des références pour tModel, ces références désigner les spécification d'interface pour un service.

tModels, ils contiennent des informations permettant de connaître les normes que respecte le service Web, le format des messages, le protocole de transport et le mode de gestion de la sécurité requise. Il permet aussi d'identifier la catégorie à laquelle appartient le service (la description abstraite des services proposés).

PublisherAssertion, représente un ensemble de règles contractuelles d'invocation de services sous forme de protocoles entre deux partenaires métier. Pour qu'une assertion soit publiée, les deux partenaires doivent publier les mêmes informations.

➢ **Relation entre WSDL et UDDI**

Un document WSDL contient deux parties :

- La description de l'interface d'un service qui correspond aux informations techniques du service, c'est en quelque sorte une classe abstraite qui sera utilisée pour implémenter un ou plusieurs autres services. L'interface du service (WSDL) -> tModel (UDDI).
- La description de l'implémentation du service qui correspond aux informations relatives à l'endroit où est publié le service. L'implémentation du service (WSDL) -> Business Service (UDDI).

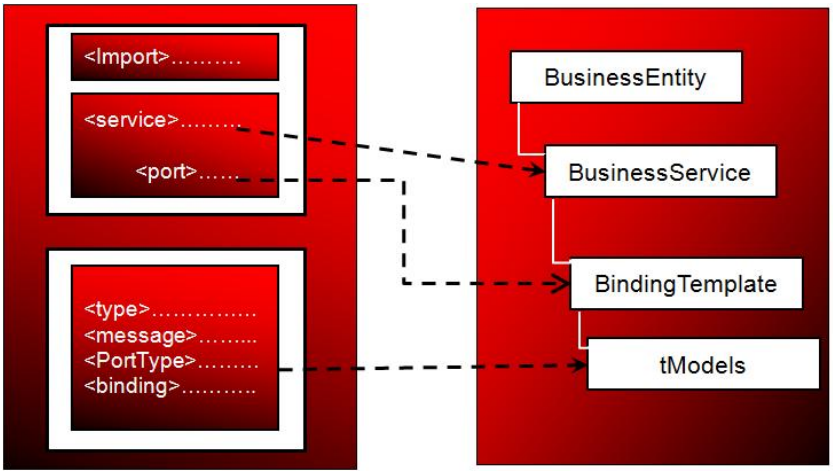

Figure 1.12 – Relation entre WSDL et UDDI

1.5.5 Publication et découverte des services Web

Chaque fois qu'un fournisseur veut publier un service Web, il faut qu'elle le rende disponible aux clients et cela par le biais du UDDI, l'enregistrement d'un service Web ce fait comme suit :

1. Couper en deux le fichier WSDL, en séparant la partie abstraite de la partie concrète.

2. Enregistrer la partie abstraite comme un technical model.
3. S'enregistrer comme une Business Entity en déterminant le nom de votre entreprise et une brève description de l'entreprise.
4. Enfin, enregistrer le BusinessService (partie concrète de document WSDL) associant le BusinessEntity avec le technicalModel.

Concernant la découverte des services Web, actuellement cette tache consiste a une recherche basée sur les mots clés au moyen de : *ftnd-binding, find_business, find_relatedBusinesses, find_service, find_tModel, get_bindingDetail, get_businessDetail, get_operationalInfo..etc*, par l'intermédiaire d'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 a ces besoins ou non.

1.6 Discussion

Malgré les bénéfices et avantages apportés par l'utilisation des services Web, cette technologie reste encore limité. Les services Web ne sont qu'un mécanisme de transfert de données d'un système à l'autre, ils n'apportent, en aucun cas, plus de valeurs à l'information déjà possédée. Ils permettent juste une meilleure diffusion auprès des clients et des fournisseurs.

L'utilisation d'XML dans l'architecture des services Web, permet de structurer la construction d'un document. Il ne permet pas de spécifier le sens à donner au document. Concernant la découverte des services Web dans le registre UDDI, nous avons vue que l'UDDI permet une recherche sous différentes entrées au registre. Les services peuvent être recherchés par exemple par nom, par emplacement ou par tModels. Cependant, cette recherche est restreinte à la détection de mots-clés et ne supporte aucune inférence des informations présentes dans l'UDDI. Ce problème est causé par l'absence de la sémantique dans les technologies de base des services Web. Pour le résoudre, les recherches faites sur le Web sémantique ont été exploitées dans le domaine des services Web.

1.7 Conclusion

Comme toute innovation, l'apparition des services donne lieu à un ensemble d'opportunités et d'applications nouvelles. Aujourd'hui, les services Web sont partout présents en nombres. Qu'il s'agisse de services de réservations en ligne ou bien de gestion de comptes bancaires et même d'applications métiers, tous ces services partagent en commun le fait d'être maintenant accessibles sous forme de services Web. Cette tendance ne semble pas devoir faiblir dans les prochaines années, bien au contraire. Ainsi, le monde du B to B (Business to Business) et plus globalement celui du commerce électronique souhaitent disposer d'interactions encore plus flexibles et automatisées entre les services Web. Cependant, cette volonté d'automatisation des interactions entre les services Web se heurte à un certain nombre de problèmes. En particulier, celui de posséder un mécanisme d'évaluation fiable et performant de l'interopérabilité effective d'un ensemble de services Web.

L'émergence de standards proposés par le W3C dans le cadre du Web Sémantique permet de proposer de nouvelles perspectives dans la résolution des problèmes (limites) des services Web. L'objet de prochain chapitre est de présenter l'architecture du projet Web sémantique ainsi que le principe de base des services Web sémantique.

Chapitre 2

Web sémantique et service Web

2.1 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 en tout genre (articles, sites personnels, audio, vidéo) mais plus ces ressources s'accumulent, plus la recherche d'une information précise s'avère difficile. Tim Berners-Lee [LEE 01], pionnier dans le domaine d'Internet, propose alors d'ajouter à toutes ces ressources une sémantique qui permettrait aux systèmes informatiques de comprendre le sens en accédant à des collections structurées d'informations et a des règles d'inférences qui peuvent être utilisé pour conduire des raisonnements automatisés : c'est la naissance du Web sémantique.

Avec l'apparition du Web sémantique, il est apparu très tôt que l'approche WSDL (la description syntaxique des services Web vue en premier chapitre), ne suffisait pas pour répondre au besoin d'ouverture imposé par Internet. Les travaux sur les services Web sémantiques (SWS) proposent de palier ce manque. Puisque nous nous intéressant à la vue sémantique des services Web, dans la première partie de ce chapitre nous étudierons l'évolution de Web actuel jusqu'à apparition de la notion du Web sémantique et surtout les ontologies. Par suite nous abordons le problème de découverte sémantique des services web ceci en élaborant une comparaison sur quelques approches existantes dans ce domaine. Et pour finir, dans la dernière partie de ce chapitre nous présenterons en détails l'algorithme de Paolucci sur le quel ce base notre approche.

2.2 Le Web traditionnel

2.2.1 Historique

Le premier réseau d'ordinateurs ARPANET, ancêtre d'Internet est une technologie qui repose sur l'envoi par paquets grâce au protocole TCP/IP, il a ensuite été mis dans le domaine public et les universitaires américains ont commencé à s'y intéresser. Tim Berners-Lee écrit, en septembre 1990, le premier navigateur Web basé sur l'hypertexte et le nomma "World Wide Web". Le Web a connu une grande évolution à tous les niveaux : conceptuel, technologique

et ses domaines d'application. Dans ce qui suit, nous allons présenter l'évolutions du Web jusqu'à apparition du Web sémantique (Web de demain).

2.2.2 Le langage HTML

Le Web traditionnel repose sur l'hypertexte, HTML (HyperText Markup Language). HTML est un langage de balises, conçu comme un langage spécifiant le contenu d'un document avec d'importantes extensions d'hypertexte. Il n'est pas conçu pour être le langage de traitement de texte d'impression conforme à la visualisation (WYSIWYG) telle que Word et WordPerfect. Ce choix a été fait parce qu'un même document HTML peut être affiché par de nombreux navigateurs de différentes capacités. Les balises du langage HTML sont délimitées par les symboles "<" et ">" ; et pour chaque élément il y a deux types de balises : des balises d'ouverture (<balise>) et des balises de fermeture (</balise>). La structure la plus courante d'un document HTML divise celui-ci en deux parties (figure 2.1). La première est l'en-tête qui est délimité par l'élément HEAD. La deuxième est le corps du document délimité par l'élément BODY. La première ligne définit la version de HTML utilisée.

```
<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN">
<HTML>
   <HEAD>
        <TITLE>Titre du Document</TITLE>
   </HEAD>
   <BODY>
     ... Le corps du document avec le balisage approprié ...
   </BODY>
</HTML>
```

Figure 2.1 – Structure d'un document HTML

2.3 Le Web structuré

2.3.1 Présentation

La naissance du Web structuré est due aux limites de langage HTML, ce dernier laisse l'interprétation des éléments marqués au navigateur ce qui donne une interprétation différente

suivant le navigateur utiliser. Des chercheurs ont donc développé le langage SGML, des détailles sur la syntaxe du SGML dans [SGML 95]. SGML a été très utilisé pour le traitement de la documentation technique dans l'industrie (aviation, automobile, etc.). Mais c'est un langage complexe, il y'a eu alors recours au développement d'un nouveau langage à savoir XML.

2.3.2 Le langage XML

Le langage XML (Extensible Markup Language) [XML], est un standard d'échange de données proposé par le W3C. Il a l'avantage de représenter les données, dans un format simple et auto-descriptif, sans se soucier de leur présentation. Du HTML, XML hérite l'utilisation des adresses Web (URL) pour pointer vers d'autres objets. De SGML, il hérite certains mécanismes d'adressage extrêmement puissants pour pointer par exemple vers des parties et des plages de documents. XML permet de représenter les données sous forme de documents contenant des éléments balisés imbriqués (figure 2.2). Les balises sont extensibles, c'est-à-dire, nous pouvons toujours ajouter des balises. Or, le document doit respecter une certaine syntaxe pour être bien formé. Pour cela, chaque balise ouverte doit être fermée, et elles doivent être correctement imbriquées. La *figure 2.2* montre un exemple d'un document XML bien formé.

```
<? xml version="1.0" encoding="ISO-8859-1"?>
   <POPULATION>
        <PERSON id_person="Adam">
            <NAME>Adam</NAME>
            <AGE>13</AGE>
        </PERSON>
 </POPULATION>
```

Figure 2.2 – Exemple d'un document XML bien formé

Un document XML est valide s'il est bien formé et respecte une DTD (Document Type Définition). La DTD est un ensemble de méta-donnée permettant de définir l'imbrication des éléments constituant un document, certains pouvant être obligatoire, d'autres optionnels,

tous répétitifs ou non [GARD 02]. La *figure 2.3* montre la DTD du document XML de l'exemple précédent.

```
<!DOCTYPE POPULATION [
<!ELEMENT POPULATION (PERSON+)>
<!ELEMENT PERSON (NAME, AGE)>
<!ELEMENT NAME (#PCDATA)>
<!ELEMENT AGE (#PCDATA)>
<!ATTLIST id_person CDATA #REQUIRED>
]>
```

Figure 2.3 – Exemple d'une DTD

Une alternative de la DTD est le schéma XML. Les schémas sont standardisés sous forme de recommandation du W3C depuis mai 2001. Malgré quelques critiques pointant surtout la complexité des schémas, ils tendent aujourd'hui à remplacer les DTD car ils permettent de définir des modèles beaucoup plus complets [GARD 02].

Toutefois dans le cadre du Web Sémantique où l'on cherche à donner du sens à l'information, on se rend rapidement compte de la limite de XML. En effet, l'exemple précèdent représente un document XML bien formé équivalent à celui-ci :

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<AAA>
    <BBB id_person="Adam">
        <CCC>Adam</CCC>
        <DDD>13<DDD>
    </BBB>
</AAA>
```

Figure 2.4 – Exemple d'un document XML

On comprend bien que l'on a défini une structure mais pas de sens. En d'autres termes XML traite la structure syntaxique des documents mais ne permet pas de spécifier ni la sémantique d'une balise, ni le typage de son contenu, ni des relations normalisées entre les balises. Le Web de demain doit permettre de définir la signification des balises, donc des informations.

2.4 Le Web sémantique

2.4.1 Naissance de Web sémantique

Tim Berners Lee, le créateur du World Wide Web, a proclamé que le Web sémantique est la prochaine évolution du Web. Les informations présentes sur le Web ont été initialement des documents enregistrés pour être consultés par des utilisateurs humains. Mais, les nouveaux systèmes conçus et développés pour fonctionner sur le Web tendent à décharger l'utilisateur de quelques, voir de beaucoup de tâches en déléguant celles-ci à des agents logiciels. Le Web Sémantique est une vision du futur Web dans lequel l'information est menée un sens explicite facilitant ainsi aux machines le traitement et l'intégration des informations sur le Web. Les recherches actuelles sur le Web Sémantique proposent de s'appuyer sur des techniques de représentation de connaissances (formalisme et raisonnement) pour munir l'information contenue dans les ressources Web d'une sémantique.

2.4.2 Architecture du Web sémantique

Le Web sémantique nécessite une architecture partagée, pour échanger des ressources sur Internet. Il ce compose de trois niveaux importants (figure 2.5) : Le niveau d'adressage, le niveau syntaxique et le niveau sémantique.

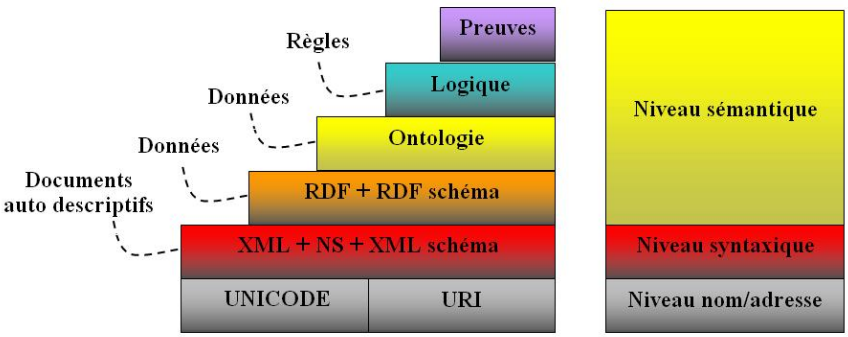

Figure 2.5 – Les niveaux du Web Sémantique

➢ **URI (W3C) :** *le concept d'URI (Universal Ressource Identifier) regroupe toutes les formes syntaxiques permettant de désigner une ressource Internet. L'URI la plus courante est l'URL (Uniform Ressource Locator). Une URL est une chaîne de caractère indiquant l'emplacement où se trouve une ressource.*

➢ **XML (W3C, 1998) :** *fournit une surface syntaxique pour les documents structurés mais ne fournit aucune contrainte sémantique sur le sens de ces documents.*

➢ **XML Schéma (W3C, 2001) :** *c'est un langage pour restreindre la structure des documents XML et aussi étendre XML avec des types de données.*

➢ **RDF (W3C, 2004) :** *RDF (Resource Description Framework),est un modèle de données pour les objets (" ressources ") et les relations entre eux, il fournit une sémantique simple pour ce modèle de données qui peuvent être représentés en XML.*

➢ **RDF Schéma :** *c'est un vocabulaire pour décrire les propriétés et les classes des ressources RDF.*

➢ **OWL :** (*Ontologie Web Language) ajoute plus de vocabulaire pour décrire les propriétés et les classes entre autres, les relations entre les classes, cardinalité, égalité, typage de propriétés plus riche, caractéristiques des propriétés et les hiérarchies des propriétés et des classes.*

2.4.3 Principe du Web sémantique

La vision du Web sémantique s'appuie sur l'utilisation des annotations et les métadonnées ainsi et surtout les ontologies :

2.4.3.1 Annotation et métadonnée sur le Web sémantique

Les Annotations et les métadonnées sont, des informations associées à une ressource du web, permettant son exploitation par des agents logiciel. Les métadonnées, sont des données sur des données, ils sont aussi des données et peuvent donc être structurées afin de décrire

une ressource quelconque. Par contre les annotations représentent des notes critiques ou explicatives accompagnant un texte [YANI 03].

Un des grands principes du Web sémantique est qu'il est nécessaire d'associer aux ressources du Web (sources d'informations et services) des informations exploitable par des agents logiciels afin de favoriser l'exploitation de ces ressources. Pour pouvoir exprimer les métadonnées et les annotations avec les informations sur le Web, beaucoup de langages ont été élaborés. Dans la suite nous présenterons quelques uns.

☞ Le langage RDF

➢ Présentation

RDF (Resource Description Framework) [RDF 99], est un standard de description de métadonnée qui permet d'affirmer des relations entre des " ressources ". Il est utiliser pour annoter des documents écrits dans des langages non structurés, ou comme une interface pour des documents écrits dans des langages ayant une sémantique équivalente.

➢ Modèle de données de RDF

Un document RDF est un ensemble de triplets de la forme <sujet, prédicat, objet> ou < ressource, propriété, Valeurs >, un peu comme <sujet, verbe, complément> d'une phrase très simple. Les éléments de ces triplets peuvent être des URIs (Universal Resource Identifiers), des littéraux ou des variables. Ces métadonnée permettent de décrire des ressources Web ou des relations entre ces ressources. RDF, est à la base exprimé sous la forme d'un graphe orienté mais on peut le sérialiser sous différentes syntaxes dont XML (On parle alors de "RDF/XML").

➢ Représentation RDF

RDF peut être représenté par un graphe, comme le montre la figure suivante :

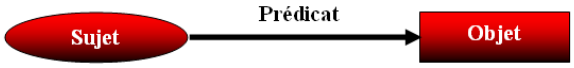

Figure 2.6 – Graphe RDF

Le même triplet sérialisé au format RDF/XML est comme suit :

```
<rdf:Description about=sujet>
        <Prédicat>objet</Prédicat>
<rdf:Description/>
```

Figure 2.7 – Document RDF/XML

Considérons l'exemple suivant, la page d'adresse ***http ://www.w3.org/Home/ALI*** a été écrite par ***Ali***. Cette phrase sera représenter en RDF comme le montre la figure suivante :

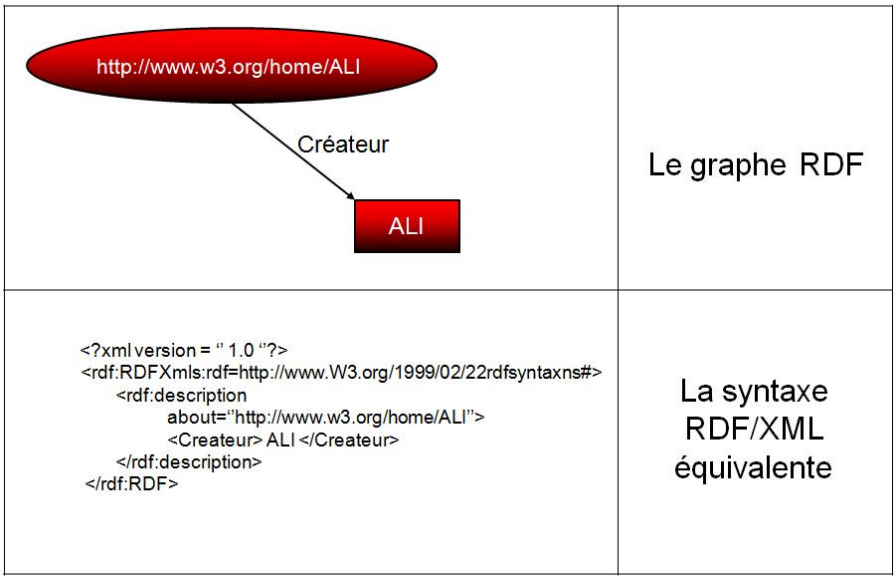

Figure 2.8 – Exemple de représentation RDF

Cette solution est difficile à lire, mais elle présente l'avantage d'être en XML et donc de profiter des avantages de ce dernier, notamment de l'interopérabilité et de la profusion d'outils. La syntaxe XML de RDF est améliorable quand elle est bien présentée.

➢ Les limites de RDF

Comme nous venons de le voir, RDF est un langage formel qui permet d'affirmer des relations entre des "ressources". Il sera utilisé pour annoter des documents écrits dans des langages non structurés. RDF est muni d'une syntaxe, et d'une sémantique. Quand les données de XML sont déclarées au format RDF, les applications peuvent comprendre une grande partie de la traduction des données. Il existe cependant un certain nombre de situations ou l'on est en droit d'être plus exigeant. Par exemple, certaines fois, nous avons besoin de connaître des informations sur les ressources identifiées, comme par exemple ce qu'elles représentent exactement : si nous avons une propriété représentant un auteur, nous pouvons en effet exiger que la valeur d'une telle propriété fasse référence à une personne (et non une voiture ou une maison). C'est pour cela que l'on associe à RDF le standard RDF Schéma (RDFS).

☞ Le langage RDFS

➢ Présentation

L'objectif de RDFS [RDFS 99], est la définition de modèles souvent appelés "vocabulaires spécifique à un domaine applicatif. Une application pourra alors, par simple analyse des types de données définis dans le modèle, connaître un certain nombre d'interprétations applicables aux annotations liées à ce modèle. Ce langage est basé sur RDF et permet de définir des notions de classes (en mettant en place des mécanismes d'héritage) et des contraintes sur des classes d'objets.

RDFS est doté d'un nombre minimal de concepts nécessaires à la définition d'un vocabulaire.

- **'rdfs :Class'** définit la notion de "classe" qui est un ensemble de plusieurs objets, par exemple la classe des "Enseignant".

- **'rdfs :Property'** définit la propriété particulière "est une sous-classe de" qui permet de définir qu'une classe est un sous-ensemble d'une autre classe : par exemple la classe "Enseignant" est sous-classe de la classe des "professions".
- **'rdfs :Resource'** définit la classe des "ressource" qui est la classe mère de toutes choses, tout est une ressource dans le web sémantique, sauf la notion de "literal". Toute classe est une sous-classe de la classe de ressource.
- **'rdfs :Literal'** définit la notion de "literal" qui est une valeur comme une chaîne de caractère ou des chiffres.
- **'rdfs :range'** définit la propriété "s'applique à la classe" (range) permettant ainsi de spécifier le champ d'application d'une propriété. Par exemple, la propriété "sont enseigner par" s'applique a la classe "Enseignant".
- **'rdfs :domain'** définit la propriété "est l'objet de la propriété" permettant ainsi de spécifier quelles sont les classes auxquelles ont peut affecter telle ou telle propriété. Par exemple, la classe des "Etudiant" peut être l'objet de la propriété "sont-enseigner-par".

➢ **Les limites de RDFS**

Les primitives offertes par RDFS sont insuffisantes pour la modélisation dans le Web Sémantique. Elles ne permettent pas, par exemple, de spécifier qu'une propriété ne peut prendre qu'une seule valeur (comme par exemple a-pour-mère), que deux classes n'ont aucun individu en commun (comme la classe HOMME et la classe FEMME), ou encore de définir une classe par un ensemble de contraintes sur ses individus (par exemple, la classe JEUNE-CADRE peut être définie comme l'ensemble des personnes dont la propriété AGE a une valeur inférieure à 30, et dont la propriété PROFESSION a une valeur 'cadre').

2.4.3.2 Les ontologie

Pour être susceptibles d'être exploitées automatiquement, les annotations et métadonnées doivent être entièrement explicites, c'est-à-dire suivre un modèle et être exprimées dans un vocabulaire clairement et formellement définis. Les ontologies, deuxième pilier du Web sémantique, sont le réceptacle de ces définitions.

➢ Définition

L'ontologie est un terme philosophique qui signifier, science ou théorie de l'être [ENCY 00]. Dans les milieux de l'intelligence artificielle, il semblerait que l'ontologie ait été abordée pour la première fois par *John McCarthy* 1980 [PSYC 04], puis, *Gruber* [GRUB 93] a proposé sa première définition sur l'ontologie :*" une ontologie est une spécification d'une conceptualisation "*. Il considère que toute représentation de connaissance est basée sur une conceptualisation. La conceptualisation représente la collection des objets, de concepts et des autres entités qui sont supposés exister dans un certain domaine d'intérêt ainsi que les relations qui les relient. Une représentation formelle de cette conceptualisation s'appelle une ontologie. Une ontologie possède une syntaxe qui est celle de langage utilisé par la machine, et une sémantique qui est celle de domaine de connaissance.

➢ Constituants d'une ontologie

Une ontologie se compose des éléments suivants [PSYC 04] : *Concept Relations ; Fonctions ; Axiomes ; et Instances.*

- ***Concepts,*** les concepts, aussi appelés classes de l'ontologie, correspondent aux abstractions pertinentes d'un segment de la réalité (le domaine du problème), retenues en fonction des objectifs qu'on se donne et de l'application envisagée pour l'ontologie.
- ***Relations,*** les relations traduisent les associations existant entre les concepts présents dans le segment analysé de la réalité. Ces relations incluent les associations suivantes : "Sous-classe-de (généralisation - spécialisation) ; Partie-de (agrégation ou composition) ; Associée-à ; Instance-de, .. etc.". Ces relations nous permettent d'apercevoir la structuration et l'interrelation des concepts, les uns par rapport aux autres. Les fonctions constituent des cas particuliers des relations, dans laquelle un élément de la relation, le nième est défini en fonction des n-1 éléments précédents.
- ***Axiomes,*** constituent des assertions, acceptées comme vraies, à propos des abstractions du domaine traduites par l'ontologie.
- ***Instances,*** constituent la définition extensionnelle de l'ontologie ; ces objets véhiculent les connaissances (statiques, factuelles) à propos du domaine du problème.

➢ Méthodologie de l'ingénierie ontologique

Fernandez et *Corcho*, ont fait une revue des différentes méthodologies de construction d'ontologies [FERN 99][CORC 03]. Toutes ces méthodologies, même si elles fournissent à l'utilisateur une aide pour la construction d'ontologies, reste insuffisantes et aucune d'elles n'est standardisé, Mais, elles se rejoignent sur un processus général de construction d'ontologie, qui repose sur un enchaînement de quatre étapes :

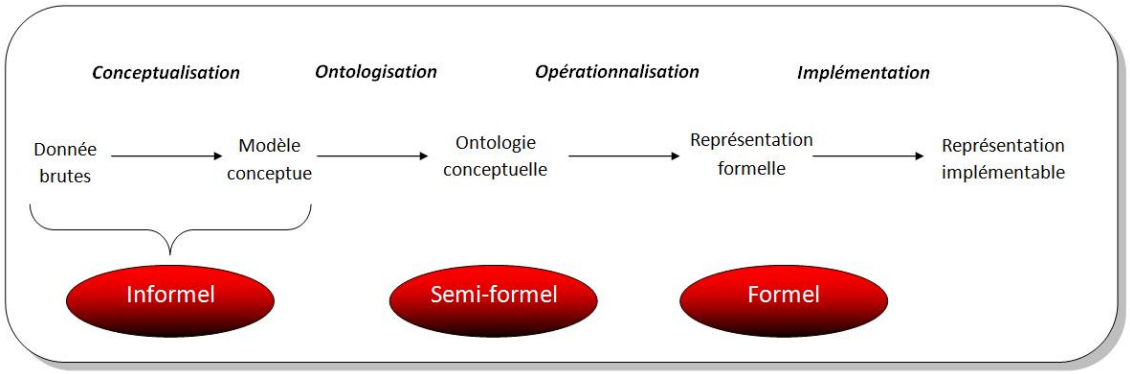

Figure 2.9 – Processus de construction d'ontologie

- ***La conceptualisation,*** consiste à, dégager a partir des données brutes les concepts et les relations entre ces concepts permettant de décrire de manière informelle les entités cognitives du domaine. Elle est réalisée par un expert du domaine assisté de l'ingénieur de la connaissance et aboutit à un model conceptuel.
- ***L'ontologisation,*** est la formalisation partielle, indépendante des contraintes d'opérationnalisation, du modèle conceptuel dont on précise la sémantique à l'aide d'axiomes et de contraintes sur les concepts et les relations. Elle est réalisée par un ingénieur de la connaissance aidé par l'expert du domaine et aboutir à une ontologie conceptuelle. Celle-ci est semi-formelle, certaines connaissances ne pouvent pas forcément être formalisées à ce niveau.
- ***L'opérationnalisation,*** est une formalisation de l'ontologie conceptuelle à l'aide d'un langage de représentation de connaissances. Elle est donc menée par un spécialiste du langage de représentation et par l'ingénieur de la connaissance. On obtient alors une représentation formelle des connaissances du domaine, basée sur l'ontologie. Le formalisme de représentation présente à priori des capacités expressives moindres que

celles du langage dans lequel l'ontologie est exprimée. L'opérationnalisation impose donc de réduire la richesse cognitive de l'ontologie.

- ***L'implémentation,*** est le passage du modèle formel de connaissances à son implémentation en machine. Cette étape peut imposer des contraintes lors de l'étape précédentes, par exemple si des éléments du modèle formel ne sont pas implémentés. Cette étape est réalisée par un informaticien connaissant les outils informatiques liés au langage de représentation utilisé.

➢ **Langage d'Ontologie Web (OWL)**

OWL (Ontologie Web Language), est un langage ontologique du Web qui est une extension de RDFS pour définir et instancier les ontologies du Web. OWL a été conçu pour être utilisé par les applications qui traitent le contenu de l'information au lieu de la présenter seulement aux êtres humains. OWL facilite grandement l'interopérabilité au niveau machine du contenu du Web plus que ce qui est déjà supporté par les langages XML, RDF et RDFS en fournissant du vocabulaire supplémentaire avec des sémantiques formelles. OWL possède des sous-langages de plus en plus expressifs *OWL Lite, OWL DL et OWL Full.*

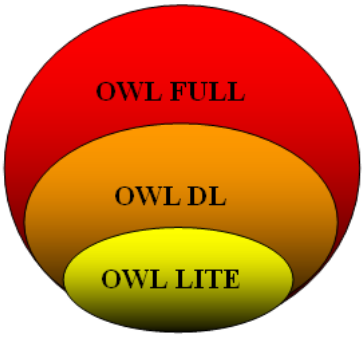

Figure 2.10 – Les trois niveaux du OWL

➦ **OWL Lite**

- Convient aux utilisateurs qui ont principalement besoin d'une hiérarchie de classification et de contraintes simples.
- Reprend tous les constructeurs de RDF (c'est-à-dire fournit des mécanismes permettant de définir un individu comme instance d'une classe, et de mettre des individus en relation).
- Utilise les mots-clés de RDFS (rdfs :subClassOf, rdfs :Property, rdfs :subPropertyOf, rdfs :range, rdfs :domain), avec la même sémantique,
- Supporte les contraintes de cardinalité, mais ne permet d'utiliser que les valeurs 0 ou 1.

➦ **OWL DL**

- Convient aux utilisateurs qui veulent le maximum d'expressivité.
- Reprend tous les constructeurs d'OWL LITE.
- Permet tout entier positif dans les contraintes de cardinalité.
- Tire son nom de sa correspondance avec les logiques de descriptions.

➦ **OWL FULL**

- Reprend tous les constructeurs d'OWL DL.
- Reprend tous les constructeurs de RDF Schéma.
- Permet d'utiliser une classe en position d'individu dans les constructeurs.

➤ **Exemple d'une représentation OWL**

La *figure 2.11* représente une déclaration d'une classe **Male** et une classe **Femelle**, sous classes de la classe **Animale**. La classe **Femelle** est disjointe avec la classe **Male**.

```
<OWL:Class rdf :ID="Male">
    <rdfs:subClassOf rdf:resource="#Animale"/>
</OWL:Class>
```

```
<OWL:Class rdf :ID="Femelle">
    <rdfs:subClassOf rdf:resource="#Animale"/>
    <OWL:disjointWith rdf:resource="#Male"/>
</OWL:Class>
```

Figure 2.11 – Exemple d'une représentation OWL

2.5 Les services Web sémantique

2.5.1 Présentation

Les services Web sémantiques se situent à l'intersection de deux domaines de recherche importants qui concernent les technologies de l'Internet : le Web sémantique et les services Web. Le Web sémantique s'intéresse principalement aux informations statiques disponibles sur Internet (pages Web) et à la manière de les décrire d'une façon compréhensible par les machines. D'autre part, les standards permettant la description des services Web tels que WSDL et UDDI qui utilisent des structures XML pour la description de la fonctionnalité des services et des données qu'ils échangent. De telles structures sont syntaxiques et engendrent une forme d'ambiguïté. Ainsi la même structure XML peut être interprétée différemment d'un utilisateur à un autre. Cette contrainte empêche l'utilisation des services Web par des agents logiciels. Le même problème s'était posé quelques années plus tôt pour les pages Web,

et avait engendré l'apparition des solutions pour l'annotation sémantique des pages Web. De nombreux travaux ont tentés d'adapter ces solutions pour l'annotation sémantique des services Web.
Un service Web sémantique est un service Web décrit en utilisant des annotations sémantiques dans un langage bien défini (exemple : les ontologies) qui permettent au service d'avoir une interface interprétable sans ambiguïté, facilitant l'automatisation de certaines taches telles que leurs découverte, sélection, invocation et leurs composition.

➦ ***Découverte automatique de services Web :*** *actuellement cette tache doit être réalisée par un utilisateur 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.*

➦ ***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 de services afin de délivrer les données nécessaires à l'exécution du service Web.*

➦ ***Composition automatique de services Web :*** *l'objectif que l'utilisateur veut atteindre nécessite souvent l'utilisation de plusieurs services Web. L'agent logiciel chargé d'atteindre cet objectif doit disposer de suffisamment de données afin de pouvoir sélectionner, composer et interopérer automatiquement ces services web.*

2.5.2 Langages de modélisation des services Web

Plusieurs langages de description des ontologies pour les services Web sémantique ont était élaborés, les plus connus sont OWL-S [OWLS 04] (que nous allons l'exploiter pour l'adaptation de notre approche), et le langage WSMO [WSMO 05].

2.5.2.1 Ontologie de Modélisation des Services Web (WSMO)

WSMO (Web Service Modeling Ontology) est une ontologie pour la description des services Web sémantiques aux moyens de :

- Propriétés non-fonctionnelles : telles que la performance, la fiabilité, la sécurité, la robustesse, et la scalabilité du service Web.
- les médiateurs utilisés par le service.
- la fonctionnalité : décrite en termes de pré-conditions, post-conditions, hypothèses et effets.
- les interfaces : Une interface de description d'un service Web est constituée d'informations telles que les erreurs gérées par le service, une description d'orchestration si le service fait appel à d'autres services, une description de la conversation du service appelée (échange de message), et des stratégies de compensations, utilisées dans le cas ou certaines opérations exécutées par le service doivent être annulées.
- les groundings : Le grounding spécifie les informations concrètes pour l'accès au service.

Il est clair que de nombreux aspects définis dans cette ontologie sont également couverts par OWL-S. Par exemple les pré-conditions, post-conditions, hypothèses et effets dans WSMO correspondent respectivement, aux entrées, sorties, pré-conditions et effets dans OWL-S. Un autre point commun est la description de la conversation d'un service dans les interfaces WSMO qui est également décrite dans le process Model de OWL-S.

2.5.2.2 Langage d'Ontologie Web - Service (OWL-S)

➢ Présentation

OWL-S est un langage de description des ontologies pour les services web. Il est basé sur le langage OWL. Le langage OWL est un langage basé sur XML, il a pour objectif d'ajouter des descriptions sémantiques aux services Web (en plus de leur description syntaxique WSDL). Il a été conçu pour faciliter l'automatisation des tâches relatives aux Web Services et plus particulièrement la découverte automatique des Web Services, leurs exécutions, ou bien encore leurs compositions et leurs interopérabilités.

➢ **Les principaux composants du langage OWL-S**

Le langage OWL-S organise la description d'un service en trois zones conceptuelles :*le Service Profile, Service Model et le Service Grounding* (figure 2.12).

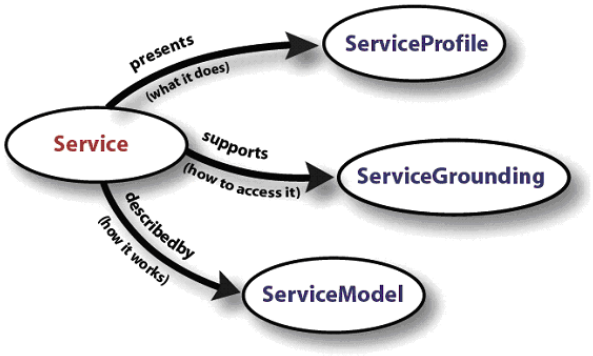

Figure 2.12 – Architecture OWL-S

Le langage OWL-S est architecturé autour de quatre entités dont la classe Service fournit le point d'entrée pour la description d'un service Web. Il n'existe qu'une seule instance de la classe Service par Web Service.

➦ ***Le Service Profile,*** pour la publication et la découverte des services. Il permet à des agents logiciels d'accéder à l'ensemble des informations nécessaires leur permettant d'évaluer si le Web Service correspond à leurs besoins.

➦ ***Le Service Model,*** permet de connaître le fonctionnement du service. Il peut être utilisé de quatre manières différentes :

- Effectuer une analyse plus complète du Web Service.
- Composer des descriptions de services.
- Coordonner les activités des différents participants à l'exécution de ce service.
- Surveiller l'exécution du service.

➦ ***Le Service Grounding,*** définit les modalités d'accès au service. Typiquement, le ServiceGrounding va spécifier un protocole de communication, les formats des messages ainsi

que les autres détails spécifiques pour la mise en oeuvre technique du service.

➢ **Ontologie de Service Profile**

La *figure 2.13* illustre la partie supérieure de l'ontologie de ServiceProfile. La figure est logiquement devisée en trois parties :

- Le bas de l'ontologie représente la description non-fonctionnelles de service, il constitue la définition de l'acteur (Actor) : il enregistre des informations sur le fournisseur du service.
- La partie centrale décrit les propriétés fonctionnelles comme : QualityRating qui est le taux attribué au service, GeographicRadius qui spécifies les contraintes géographiques de service..etc.
- La partie supérieure de la figure représente la description fonctionnelle du service, il décrit les capacités du service en termes d'inputs/outputs, preconditions et effects. Les inputs sont ce qui est requis par un service afin de produire l'output désirée. Les preconditions représentent les conditions dans le monde qui devrait être vrais pour la bonne exécution du service. L'exécution du service va avoir des résultats dans le monde, ces résultats sont décrient par la description effects.

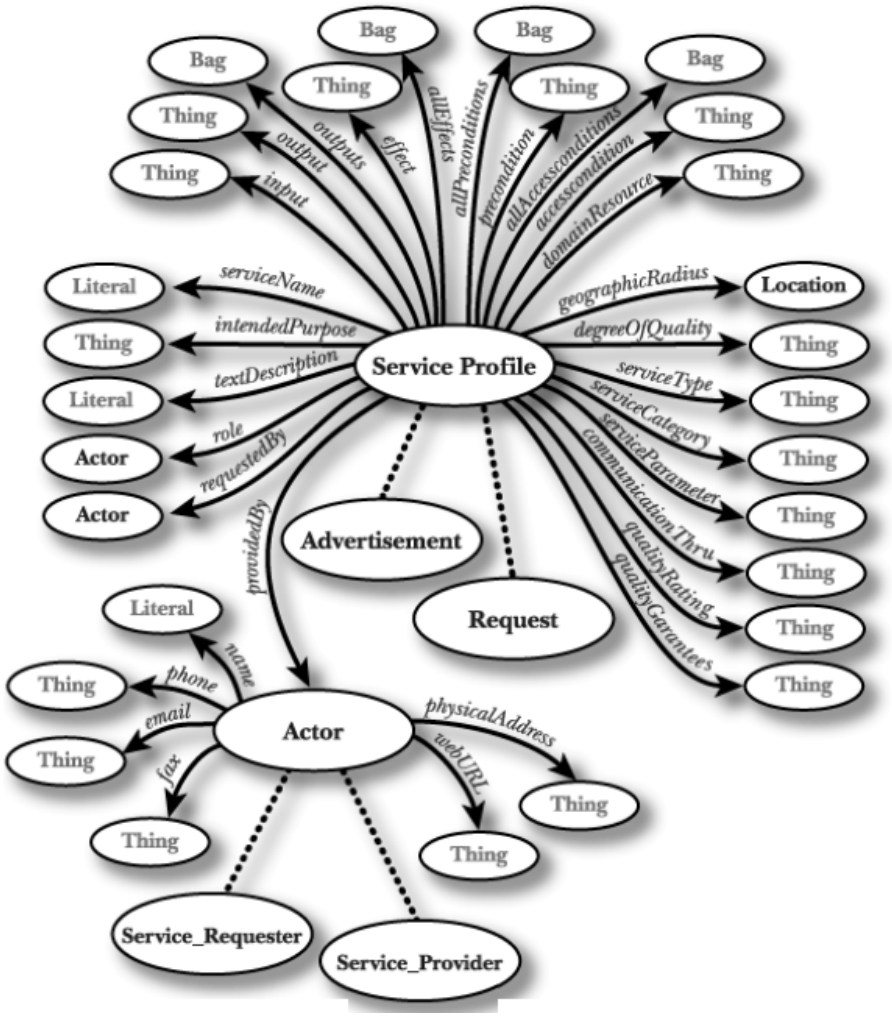

Figure 2.13 – Ontologie de ServiceProfile

2.6 La découverte dynamique des services web

L'un des problèmes les plus importants dans l'architecture des services Web est le problème de découverte. La découverte des services Web tel que l'ont défini *Trasour, Bartolini* et *Gonzalez* [TRAS 01] est le fait de mettre en correspondance deux entités tel que l'une offre ce que l'autre requiert. La correspondance permet ainsi de sélectionner le service approprié pour répondre à la requête de l'utilisateur. La découverte des services Web peut être syntaxique ou sémantique selon leurs descriptions impliquées. Les premières techniques de découverte de services Web se basaient sur les descriptions syntaxiques présentes dans les descriptions WSDL. Ce type de solution s'est avéré trop imprécis, et a engendré l'apparition d'autres

algorithmes de découverte basés sur des descriptions sémantiques. On outre, on entend par découverte dynamique la possibilité de localiser automatiquement un service web qui répond à des besoins particuliers.

2.7 Approches existantes

Différentes approches ont été proposées pour la résolution du problème de découverte sémantique des services web, toutes ces approches implantent en fait une découverte approximative car il n'est pas réaliste d'imaginer qu'il y a toujours un service qui correspond exactement aux besoins spécifiés, ces approches diffèrent par le langage de description de service utilisé (DAML-S, OWL-S, logique de description...etc.) et/ou par l'algorithme de découverte utilisé (l'algorithme de *Paolucci* [PAOL 02][SYCA 02][SYCA 03][PAOL 04]. Test de subsumption (*Gonzàlez-Castillo* et *Trastour* 2001 [GONZ 01][TRAS 01]), utilisation des filtres [SYCA 99], réécriture (*Benatallah* et *Alain*, 2002 [BENA 02])), et/ou par la méthode utilisée (ajouter une couche sémantique au dessus de WSDL et UDDI (*Verma* et *Miller*, 2003 [VERM 03])).

L'approche proposée par *Gonzàlez-Castillo et Trastour*, se base sur une description DAML-OIL des services, le principe est d'exploiter les mécanismes de raisonnement fournis par DAML-OIL pour supporter la découverte dynamique des services web, l'algorithme de correspondance repose sur les tests de subsumption entre concepts.

Paolucci, propose un algorithme de correspondance plus élaboré entre un service et une requête décrits en DAML-S. L'algorithme reconnaît différents degrés de correspondance qui sont déterminés par la distance minimale entre les concepts dans une ontologie globale.

Benatallah et Alain, Ont proposés une approche qui permet de résoudre de problème de découverte basé sur les capacités des services en utilisant la logique de description. Le principe est de réécrire les concepts en utilisant des terminologies. L'implémentation de cette approche a aboutie à la réalisation du projet MKBEEM (Multilingual Knowledge Based Eu-

ropean Electronic Marketplace).

Dans [SYCA 99], *Sycara* a proposé une approche dans lequel les services sont considérés comme des frames, et les proprietés *inputs/outputs, inConstraints* et *outConstraints* peuvent être utilisés pour décrire les attributs essentiels d'un service. Le processus de découverte dans cette approche réalise une découverte syntaxique et sémantique, il est formé d'un groupe de filtres indépendants les uns des autres qui limitent progressivement le nombre des services qui vont être découverts.

Par contre *Verma et Miller*, ont explorés la possibilité d'ajouter la sémantique au dessus de la description WSDL et UDDI, Cette approche consiste à impliquer la relation des concepts dans WSDL et l'ontologie DAML+OIL dans la description des services web pour que par suite produire une interface pour UDDI qui acceptera des requêtes basées sur des concepts ontologiques. Le faite que WSDL a été accepté comme standard industriel pour la description des services Web, cette approche est plus confiante et efficace pour les entreprises surtout avec la standardisation du langage SAWSDL (Semantic Annotations for WSDL and XML Schema) [SAWSDL 07][JACE 07] (extension de WSDL 2.0), que celles qui maigres de WSDL vers DAML-S (OWL-S) qui n'est pas encore standardisé ou que celle qui utilise la logique de description. En utilisant le standard WSDL les fournisseurs de services web ont une option pour décrire et publier leurs services avec, ou non la sémantique au dessus. Les auteurs ont met en considération une ontologie pour, les inputs/outputs et/ou les attributs fonctionnels des opérations ainsi que les effets.

2.8 Approche de *Paolucci*

2.8.1 Présentation de l'approche

Dans cette approche [PAOL 02], les informations considérées dans la description des services Web pour la publication sont les entrées/sorties (inputs/outputs) offertes par le service. La requête est également spécifiée sous forme d'un ensemble d'inputs/outputs requises. Dans l'algorithme proposé par les auteurs, un service web S peut répondre à la requête R si

les entrées requises par S sont équivalentes ou englobent (selon une ontologie partagée) celles offertes par R, et si les sorties offertes par S sont équivalentes ou englobent celles requises par R. Ce critère garanti que le service trouvé satisfait les besoins de la requête. Les auteurs se basent sur une description DAML-S des services Web, les entrées sorties sont ainsi définies en utilisant l'ontologie DAML+OIL publier sur le Web, les relations de subsomption peuvent donc être facilement déduites.

2.8.2 Adaptation avec le standard UDDI

L'annuaire UDDI, est une solution qui permet aux entreprises d'enregistrer leurs services web pour que ces derniers soient recherchés et invoquer facilement. Dans UDDI, les services web peuvent être cherché par : emplacement, par nom d'entreprise, par bindings ou par tModels ce qui implique que l'UDDI offre une pauvre recherche car il permet une recherche basée seulement sur les mots clés et par quelques valeurs associées aux tModels, il n'y a pas une forme d'inférences, c-à-d, qu'il ne peut produire aucun support pour trouver des services dans le biais de ce que ces services produit. Par exemple, un service de vente de voiture peut être publié comme "Concessionnaires de voitures neuves", mais une recherche sur UDDI pour un service " Concessionnaires d'automobiles " ne va pas identifier le service publier malgré que le service "Concessionnaires de voitures neuves" est sous-type de " Concessionnaires d'automobiles " selon l'inférence sur l'ontologie globale.

Le deuxième problème avec UDDI c'est qu'il peut produire beaucoup de résultats qui n'ont pas d'intérêt. Par exemple, lors de la recherche pour "Concessionnaire automobile", peut être que le client n'est pas intéressé par les concessionnaires qui n'acceptent pas des cartes de crédit comme méthode de paiement, ceci implique que pour résoudre ce problème il faut aussi mettre en confédération les inputs/outputs des services web afin de produire une recherche plus précise [SYCA 04].

L'approche de *Paolucci* résous ce problème en ajoutant une couche sémantique pour permettre une inférence sémantique, en se basant sur la découverte des capacités des services et par utilisation de DAML-S comme langage de description des capacités (inputs/outputs). L'adaptation de cette approche avec UDDI permettra aux services d'être recherchés par des mots clés UDDI et aussi par inférence sémantique.

2.8.3 Architecture de découverte combinée

2.8.3.1 Principe

La figure ci-dessous (figure 2.14) illustre l'architecture combinée DAML-S/UDDI, proposé par les auteurs dans [PAOL O2].

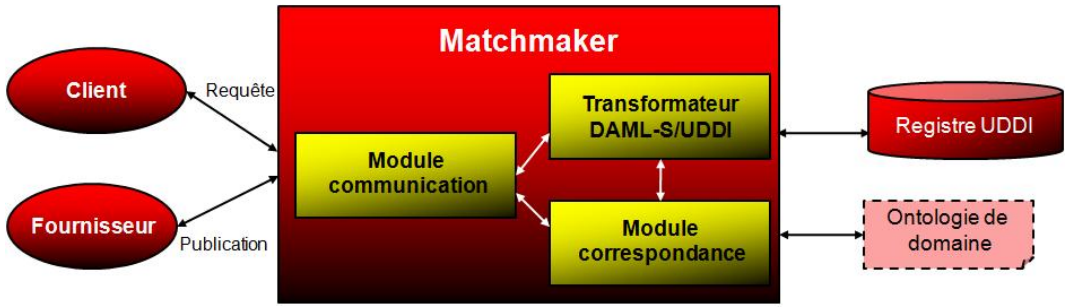

Figure 2.14 – Approche combinée DAML-S/UDDI.

Dans l'architecture combinée DAML-S/UDDI ci-dessus, la première tache de service web est de publier ces capacités avec le Matchmaker (le Matchmaker représente la couche ou l'algorithme injecter par les auteurs a l'architecture classique des services web afin d'aboutir a une nouvel architecture basé sur le raisonnement sémantique). La registration des capacités permet au service web d'être découvert et d'agir comme fournisseur [SYCA 03].

Le Matchmaker reçoit les publications des services et les requêtes dans le format DAML-S de l'extérieur a travers le module communication après avoir reconnue que le message est une publication, le module communication l'envoie au module transformateur qui construit une description UDDI a partir de la description DAML-S, ce dernier produit une clé de référence ID de service. Cet ID combiné avec la description des capacités publiées sont envoyé au module de correspondance, qui ce charger de sauvegarder les publications dans une base de donnée des publications en même temps il les indexent en utilisant les ontologies télécharger a partir du web et sauvegarder dans la base de donnée des ontologies, la registration des capacités permet au service web d'être découvert et d'agir comme fournisseur.

Si une requête envoyer au Matchmaker, le module communication l'envoie au module de correspondance qui établie une correspondance basé capacités, le résultat de correspondance c'est la publication des fournisseurs sélectionnés et une référence ID au service enregistrer dans l'UDDI.

2.9 Discussions

Les avantages d'intégration du Web sémantique dans la description des services Web sont nombreux. En plus de rendre l'interface du service accessible automatiquement par des machines et la description fonctionnelle des services ils permettent également, la description des propriétés non fonctionnelles telles que la qualité de service, les contraintes géographiques ainsi que les contraintes de sécurité, d'une manière uniforme compréhensible par tous.

Dans ce chapitre, nous avons vu que la modélisation avec des ontologies n'est pas tout à fait différente avec celle dans le contexte de la technologie de programmation. Malheureusement, des ontologies n'ont pas été largement acceptées en général dans le développement des logiciels. En effet, le Web sémantique est souvent critiqué lorsqu'il s'agit de la confiance. Comment faire confiance à une machine. Un service transactionnel interagit avec un humain. Comment la machine pourrait-elle prendre le rôle d'un humain ?

2.10 Conclusion

Dans ce chapitre nous avons vu que, le Web sémantique est le Web de demain dans lequel les ontologies jouent un rôle très important dans la représentation des connaissances. Avec l'apparition du Web sémantique, il est apparu très tôt que les approches WSDL et UDDI (présenter dans le premier chapitre) ne suffisait pas pour répondre au besoin d'ouverture imposé par Internet. En effet, les travaux sur les services Web sémantiques (SWS) proposent de palier ce manque en ajoutant une couche sémantique à la description de service affin d'automatiser le processus de découverte des services web.

Dans le prochain chapitre nous présenterons les systèmes multi-agents, ainsi que le principe de leurs interopérabilité avec les services Web.

Chapitre 3

Agent et service Web

3.1 Introduction

Les travaux en IA (Intelligence Artificielle) tentent de mettre en place un ensemble de théories et d'outils permettant la compréhension et le raisonnement sur les processus de résolution automatique des problèmes. Depuis son apparition, l'IA a été appliquée sur des entités cognitives destinées à résoudre des problèmes de plus en plus complexes. Face à la complexité des domaines d'application, l'IA a été confrontée à un certain nombre de limites matérielles et conceptuelles (dues essentiellement à la nature distribuée d'un certain nombre de problèmes). Avec le développement de l'informatique distribuée, l'IA a donné lieu à un nouveau domaine : l'IAD (Intelligence Artificielle Distribuée). Les systèmes multi-agents (SMA), constituent un paradigme visant à canaliser l'IAD dans un cadre abstrait et modulaire qui permet une approche conceptuelle simple pour appréhender des problèmes complexes. Avec les systèmes multi-agents, les défis théoriques engendrés par la distribution de l'expertise se sont concrétisés donnant lieu à plusieurs champs de recherche différents. Les SMAs constituent l'un des champs de recherche les plus actifs dans le domaine de l'informatique et ses champs d'application ne cessent de s'élargir.

Dans ce chapitre nous allons aborder la notion d'agents en présentant ses types ainsi que ses principales caractéristiques. Par suites nous définissant les systèmes multi-agents, enfin nous montrerons la relation qui existe entre les défirent technologies (Agents, Web service et Web sémantique).

3.2 Notion d'agent

3.2.1 C'est quoi un agent ?

Dans "Petit Robert" le mot agent vient du latin "agere", c-à-d : agir ou faire. Un agent est celui qui agit (opposé au patient qui subit l'action).

En informatique, un agent est l'équivalent d'un robot logiciel. C'est un programme qui accomplit des tâches à la manière d'un automate et en fonction de ce que lui a demandé son auteur.

Dans le contexte d'Internet, les agents intelligents sont liés au Web sémantique, dans lequel ils sont utilisés pour faire à la place des humains les recherches et les corrélations entre les résultats de ses recherches. Ceci se fait en fonction des règles prédéfinies. Ils sont aussi capables de dialoguer entre eux [WIKAG].

3.2.2 Définition

Le terme agent est utilisé de manière assez vague. Cependant *J. Ferber* [FERB 95] a pu dégager une définition minimale est presque commune : Il a défini un agent par, une entité physique ou virtuelle

- qui est capable d'agir dans un environnement,
- qui peut communiquer directement avec d'autres agents,
- qui est mue par un ensemble de tendances (sous la forme d'objectifs individuels ou d'une fonction de satisfaction, voire de survie, qu'elle cherche 'a optimiser),
- qui possède des ressources propres,
- qui est capable de percevoir (mais de manière limitée) son environnement,
- qui ne dispose que d'une représentation partielle de cet environnement (et éventuellement aucune),
- qui possède des compétences et offre des services,
- qui peut éventuellement se reproduire,
- dont le comportement tend à satisfaire ses objectifs, en tenant compte des ressources et des compétences dont elle dispose, et en fonction de sa perception, de ses représentations et des communications qu'elle reçoit.

3.2.3 Caractéristiques d'un agent

Un agent doit posséder un certain nombre de caractéristiques pour correspondre à la définition que nous avons adoptée.

- ***Autonomie,*** un agent autonome est capable d'agir seul, c'est-à-dire de prendre des décisions selon des critères qui lui sont propres et sans l'intervention d'un autre agent

ou d'un opérateur humain, ce qui le différencie de tous les concepts semblables, qu'ils s'appellent " objets ", " modules logiciels " ou " processus ".

- ***Interactif,*** capable de communiquer avec d'autre agents pour synchroniser leurs actions et pour résoudre des conflits (de ressources, de buts ou d'intérêts).
- ***Adaptatif,*** capable de modifier son comportement basé sur son expérience.
- ***Rationnel,*** capable de choisir une action en se basant sur les buts internes et des connaissances qu'une action va amener à ses buts.
- ***Coopératif,*** capable de coopérer avec d'autre agents pour atteindre un but commun.
- ***Intentionnel,*** l'agent possède des buts explicitement qui motivent son action, c'est-à-dire la volonté d'un agent d'atteindre un but ou d'effectuer une action.
- ***Intelligent,*** l'agent doit pouvoir interpréter les événements observés afin de prendre les décisions appropriées, de manière autonome.

3.3 Systèmes multi-agents

Les systèmes multi-agents sont l'une des dernières générations de systèmes informatiques intelligents. Emergeant de la recherche en intelligence artificielle distribuée dans les années quatre-vingt, les systèmes multi-agents constituent aujourd'hui une grande part de la recherche et du développement de l'intelligence artificielle et de la programmation distribuée.

3.3.1 Définition

Un SMA (Systèmes Multi-Agents) est une connexion d'entités (agents) faiblement couplées qui interagissent afin de résoudre des problèmes qui dépassent les capacités et les connaissances de chacun [SYCA 98]. Deux points importants sont exposés dans cette définition, le premier est que les entités ou les agents doivent être faiblement couplés. Le deuxième point est la distribution des capacités et des connaissances entre les entités face aux exigences des problèmes traités par le système. La distribution des outils de résolution est essentielle, elle permet de réduire la complexité de la solution, La distribution des compétences face aux objectifs constitue le moteur principal qui pousse les entités à interagir pour une résolution coopérative des problèmes.

3.3.2 Composants des systèmes multi-agents

Selon Ferber [FERB 95], un système multi-agents, est un système composé des éléments suivants.

- Un environnement E, c'est-à-dire un espace disposant généralement d'une métrique.
- 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.
- Un ensemble A d'agents, qui sont des objets particuliers $(A\ O)$, lesquels représentent les entités actives du système.
- Un ensemble de relations R qui unissent des objets (et donc des agents) entre eux.
- Un ensemble d'opérations O_p permettant aux agents de A de percevoir, produire, consommer, transformer et manipuler des objets de O.
- 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.

3.3.3 La communication

3.3.3.1 Présentation

Les systèmes multi-agents constituent une approche de résolution distribuée des problèmes. Dans un SMA, l'expertise nécessaire pour le traitement des problèmes dans le domaine d'application est distribuée entre les agents du système. Les agents communiquent et interagissent pour synchroniser leurs actions, pour résoudre des conflits (de ressources, de buts ou d'intérêts) et aussi pour s'aider mutuellement.

Les systèmes multi-agents utilisent le langage ACL (Agent Communication Language) [ACL 02] pour la communication (l'échange d'information, de connaissances ou encore de services) entre agents. C'est la communication entre agents qui fait qu'un groupe d'agents forme un SMA. 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, .. etc.). Le vocabulaire commun entre les agents concerne les définitions précisées dans une ontologie (figure 3.1).

Figure 3.1 – Modèle de langages de communication entre agents

3.3.3.2 Modèles de communication entre agents

On distingue essentiellement deux modèles de communication entre agents.

- Communication par envoi de messages.
- Et Communication par partage d'information.

➢ Communication par envoi de messages

Ce type de communication est caractérisé par le fait que les agents possèdent une représentation propre et locale de l'environnement qui l'entoure .Chaque agents va alors interroger directement les autres agents sur cet environnement ou leur envoyer des informations sur sa propre perception des choses. La transmission de l'information est effectuée sous deux modes.

- *Mode point à point,* l'agent émetteur du message connaît et précise d'adresse de/des agent(s) destinataire(s). Ce type de communication qui est généralement le plus employé dans les systèmes a agents cognitifs.
- *Mode par diffusion,* le message est envoyé à tous les agents du système. Ce type de transmission est très utilisé dans les systèmes dynamiques ainsi que les systèmes d'agent réactif.

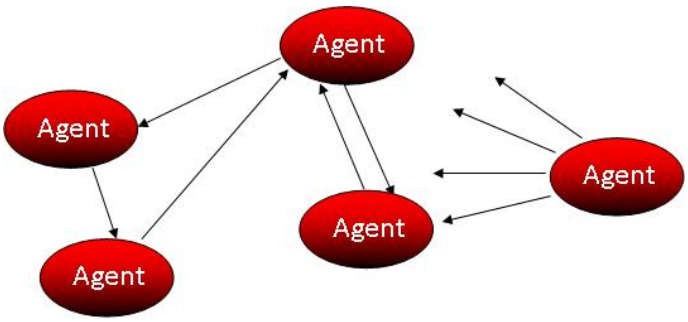

Figure 3.2 – Communication par envoie de messages

➢ **Communication par partage d'information**

Elle s'effectue via un tableau noir (blackboard). Ce dernier fournit un environnement pour échanger les données, qui sont écrites dessus par les agents participants. N'importe quel agent peut obtenir l'accès aux informations et aux messages signalés sur le tableau noir. Ce type de communication est utilisé quand il y a recouvrement des domaines d'expertise de chaque agent. Il suppose également que les agents ne possèdent qu'une connaissance limitée sur les domaines d'activité des autres agents. Ce type de communication est l'un des plus utilisés dans la conception des systèmes multi-agents.

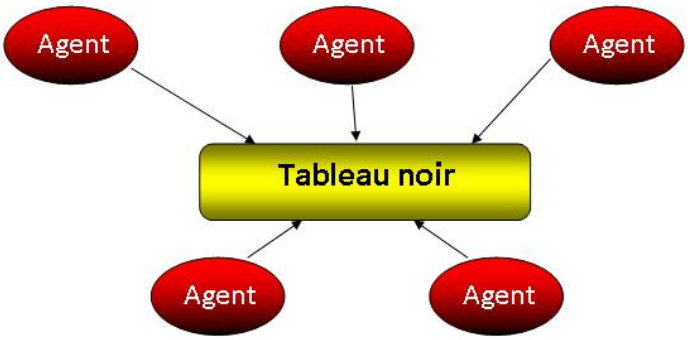

Figure 3.3 – Communication par partage d'informations

L'architecture d'un système à base de tableau noir comprend 03 sous systèmes.

- Les sources de connaissance (agents).
- La zone partagée " blackboard " qui contient les résultats partiels des problèmes en cours de résolution, les hypothèses qui spécifient à un moment donné quels sont les sources de connaissances qui peuvent contribuer à l'avancement (progression) de la résolution du problème, et d'une manière générale toutes informations que s'échangent les agents.
- Un dispositif de contrôle qui gère les conflits d'accès entre agents.

3.3.4 Les modes d'interaction inter-agents

Un système multi-agents se distingue d'une collection d'agents indépendants qui s'interagissent entre eux par l'intermédiaire d'un autre agent ou même en agissant sur leur environnement. De même, les agents peuvent désirer des ressources que les autres agents possèdent. Pour pouvoir travailler ensemble, les agents ont besoin de communiquer. Il existe différents protocoles de communication.

3.3.4.1 La coordination

La coordination est la propriété du système d'agents accomplissant une activité dans un environnement partagé. L'objectif est d'avoir un système plus cohérent. Le degré de coordination peut être envisagé. Le but de la coordination est de gérer les interactions et les activités des agents de manière à atténuer les effets de l'interaction négatifs. Il s'agit de trouver parmi un ensemble de comportement d'agents ceux qui permettent de satisfaire les objectifs les plus importants.

3.3.4.2 La coopération

L'agent est capable de collaborer avec d'autres agents pour accomplir un but commun. Ils partagent la réussite ou l'échec. (Le terme "Collaboration" est utilisé en tant que synonyme de "Coopération"). La coopération entre différents agents est un problème difficile. Les agents ont une connaissance plus ou moins précise des autres agents de système. Ils doivent se représenter les compétences des autres agents, mais également les taches que ces derniers sont entrain de réaliser.

3.3.4.3 La négociation

La négociation est une forme de coordination entre des agents compétitifs, i.e. des agents concurrents dont les buts s'opposent (conflits). Chacun des agents communique avec les autres pour atteindre son propre but. Le mécanisme favori pour la résolution des conflits et la coordination, inspiré du modèle des humains, est la négociation. Dans le cas des agents intelligents et dans le système multi-agents, la négociation est une composante de base de l'interaction surtout parce que les agents sont autonomes, il n'y a pas de solution imposée a l'avance et les agents doivent arriver à trouver des solutions dynamiquement, pendant qu'ils résolvent les problèmes.

3.3.5 L'interopérabilité des SMA

Dans les SMAs, trois niveaux d'interopérabilité peuvent être distingués : le niveau technologique, le niveau opérationnel et le niveau sémantique [MELL 04].

3.3.5.1 L'interopérabilité technique

L'interopérabilité technique a été largement abordée dans le cadre de l'informatique distribuée. C'est le niveau le plus bas de l'interopérabilité. A ce niveau, se pose le problème de la possibilité d'interagir entre les entités implémentées dans des systèmes hétérogènes utilisant des langages hétérogènes et surtout des systèmes de typage hétérogènes. L'intégration des applications utilisant des langages et des modèles différents constitue le premier obstacle pour faire interagir des applications. A ce niveau, la standardisation est l'approche adoptée pour résoudre le problème de l'hétérogénéité technique. L'utilisation d'XML et Java joue actuellement un rôle important dans les standards émissent par la FIPA afin d'assurer l'interopérabilité technique.

3.3.5.2 L'interopérabilité opérationnelle

Chaque agent ou plus généralement entité logicielle est implémenté suivant un modèle opérationnel, que ce soit un objet ou un processus, une approche RPC ou orientée messages. A ce stade, l'interopérabilité concerne le mode de fonctionnement des entités logicielles. Deux entités sont interopérables au niveau opérationnel quand elles sont interopérables au niveau technique et qu'elles peuvent communiquer selon leur modèle opérationnel mutuel. L'interopérabilité au niveau opérationnel permet aux entités de gérer et d'exécuter toutes les interactions possibles des deux entités.

3.3.5.3 L'interopérabilité sémantique

C'est un niveau d'interopérabilité spécifique aux systèmes multi-agents. En effet, deux entités interopérables au niveau opérationnel peuvent interagir sans pour autant comprendre le sens et mettre à profit leurs interactions. Au niveau sémantique, les entités interagissent en échangeant le sens et peuvent alors se rendre service. Deux systèmes sont sémantiquement interopérables s'ils sont interopérables au niveau opérationnel et s'ils partagent la même référence sémantique sur l'objet de l'interaction Au niveau opérationnel, les entités sont capables de capter le fonctionnement d'un protocole mais sans interopérabilité sémantique, les entités sont incapables d'assurer son bon déroulement en vérifiant les contraintes fonctionnelles et en construisant les messages avec le contenu conforme.

3.3.6 Langage de communication entre agents ACL

Depuis quelques années, la recherche été basée sur trouvé un moyen pour permettre aux programmes informatiques de s'échanger des informations et des connaissances. Les techniques suivantes ont déjà été utilisées, indépendamment des notions d'agent : RPC (Remote Procédure Call), RMI (Remote Method Invocation), CORBA (Common Object Resource Broker Architecture). Cependant, ces techniques sont limitées car elles restent spécifique au cadre de la programmation impérative ou procédurale : à chaque instant, le programme sait quelle procédure appeler. En ce sens, elles sont mal adaptées aux systèmes multi-agents. C'est pourquoi plus récemment, sont apparus les langages de communication entre agents (KIF, KQML, ACL).

La compréhension entre agents passe par différents niveaux.

- Tout d'abord, il faut que les agents parlent le même langage, ou bien qu'il existe des traducteurs bijectifs d'un langage dans un autre. On est ici au niveau syntaxique.
- Mais il faut également que les mêmes objets, les mêmes concepts aient la même signification pour tous les agents. Il est donc nécessaire que tous les agents partagent une ontologie commune, c'est à dire une définition des concepts associée aux relations qui existent entre eux.
- Enfin, les agents doivent être capables de dialoguer entre eux, pas au niveau du simple envoi de messages, mais à un niveau supérieur : ils doivent pouvoir s'informer, se poser des questions, rechercher d'autres agents, etc. Pour ce faire, les agents ne peuvent pas se contenter d'appeler une procédure (laquelle appelleraient-ils... ?). ils doivent spécifier un état désiré dans un langage déclaratif, c'est ici qu'intervient l'ACL.

En 1999, la FIPA (Foundation for Intelligent Physical Agents) a défini ACL, pour répondre principalement à une critique assez vigoureuse contre KQML (Foundation for Intelligent Physical Agents) [KQML 92], ses performatifs sont trop nombreux (37 performatifs), donc redondants, et de plus ils ne couvrent pas en entier le champ des performatifs envisageables. L'ACL ressemble beaucoup à KQML, mais il ne comporte que 22 performatifs. De plus, il est défini de façon beaucoup moins ambiguë : la sémantique des performatifs est décrite dans un langage spécialisé, le langage SL (Semantic Language).

La figure suivante représente un message au format ACL.

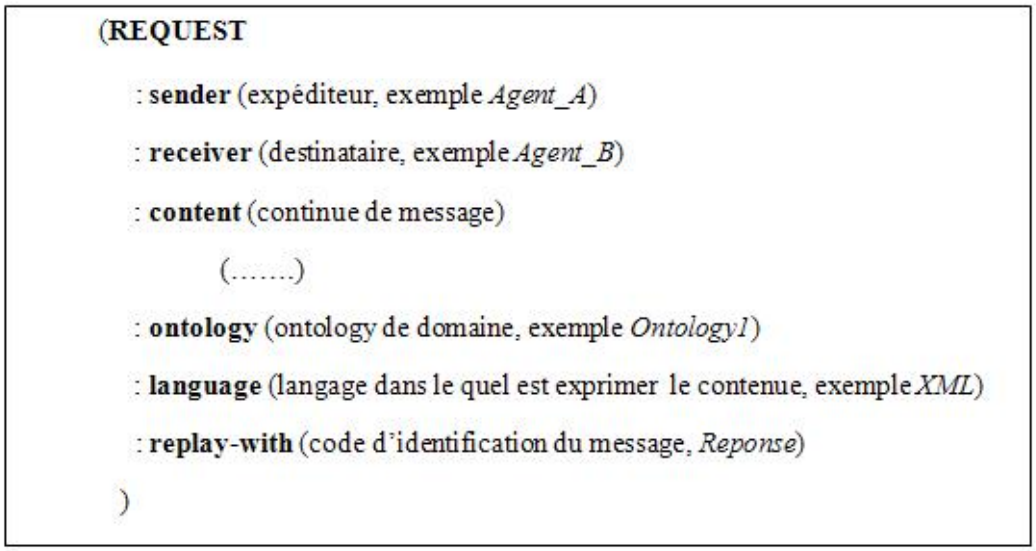

Figure 3.4 – Une requête au format ACL

Dans la figure ci-dessus, l'*AgentA* envoie a lAgentB un message ACL. Ce message a été émis avec la performative *Request*, la structure standard du message apparaît clairement. Elle se compose des champs expéditeur (:sender), destinataire (:receiver), contexte (:ontology), message (:content) et d'un certain nombre de champs métalinguistiques, en particulier le champ (:language) qui spécifie le langage du contenu du message. Le champ (:ontology) sert quant à lui, à préciser l'ontologie décrivant la structure de la conversation entre agents.

3.3.7 Avantages et objectifs des SMA

Les SMA sont des systèmes idéaux pour représenter des problèmes qui possèdent de multiples méthodes de résolution, de multiples perspectives. D'après [JARR 02] et [BOUD 07] l'approche SMA 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 SMA sont [KAZA 96] :

- l'analyse théorique et expérimentale des mécanismes.
- La résolution de programmes distribués.

3.3.8 ACL dans le contexte du web

La *figure 3.5* ci-dessous montre la superposition des protocoles et langages de communication entre agents. Les langages de communication entre agents (comme KQML ou l'ACL de la FIPA) s'appuient sur les protocoles de communication classiques d'Internet : SMTP (Simple Mail Transfer Protocol) ou HTTP (Hypertext Transfer Protocol), qui reposent eux-mêmes sur les couches TCP (Transmission Control Protocol) et IP (Internet Protocol). Jusqu'en 1999, les langages de communication entre agents avaient été développés de façon complètement indépendante d'Internet. Notamment, aucun lien n'avait été fait avec les technologies du Web sémantique comme XML et/ou RDF. Les choses ont beaucoup changé depuis, notamment avec les travaux du W3C.

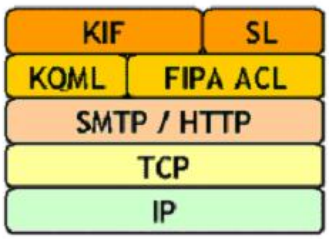

Figure 3.5 – Protocoles et langages de communications entre agents

➢ **ACL et le Web sémantique** La préoccupation initiale des ACL, qui était de permettre la communication entre agents hétérogènes, peut quasiment s'identifier aujourd'hui à la vision du web sémantique . Même si le vocable d'agent est peu utilisé, il s'agit bien de faire collaborer entre eux des programmes autonomes et hétérogènes. L'initiative la plus importante à cet égard est sans doute OWL. A ce jour, les langages définis sont les suivants.

- *OWL*, Les langages de communication entre agents vus précédemment KQML et l'ACL, permettent aux agents de communiquer dans des langages standardisés (ils sont interopérables), mais rien ne garantit qu'ils comprennent les données qu'ils s'envoient, OWL permet de représenter des ontologies. Le formalisme logique est une variante des logiques de description et permet une forme limitée d'inférence, essentiellement basée sur des tests de subsomption entre classes. OWL est en quelque sorte le successeur de KIF.
- *OWL-S*, définit une ontologie pour les services web. Le langage permet en outre la description des actions que peut accomplir chaque agent, dans les termes d'une ontologie DAML+OIL, et la décomposition de tâches complexes en sous-tâches élémentaires. OWL-S n'est pas à proprement parler un ACL, car il ne s'agit pas véritablement d'un langage, mais il permet à chaque agent d'offrir un certain nombre d'opérations aux autres agents, et de comprendre la sémantique des opérations proposées par ses pairs, grâce à des ontologies de référence.

3.4 Les SMA et les services Web

On assiste aujourd'hui à un rapprochement entre les recherches en SMA et les services Web. Ce rapprochement se manifeste sous différents aspects. Il y a essentiellement deux aspects.

- D'une part l'utilisation des agents en tant que cadre conceptuel pour mettre en place des services Web sophistiqués, ou encore des outils de planification, découverte et composition des services Web, ce qui est le cas de notre cadre d'étude, i. e, utiliser les SMAs comme environnement pour la découverte des services Web
- D'autre part l'utilisation des services Web comme un cadre architectural et/ou technologique pour mettre en place des SMAs ou des agents accessibles à travers Internet. L'objectif principal est de rendre les capacités des agents accessibles à travers des services Web.

Le tableau suivant représente une comparaison entre les différentes technologies (services Web et SMA)

Environnement	SMA	Web Services
Description de service	Description d'agent	WSDL
Registration	DF/AMS (détailler en bas)	UDDI
Communication	ACL	SOAP
Langage Sémantique	SL	-
Interaction	IP	WS-BPEL WS-CDL
Autre capacités	Négotiation	-

Figure 3.6 – Comparaison entre les services web et les SMAs

La technologie des services web actuel présente quelques limites, nous allons citer quelques unes.

- Les services web sont passifs jusqu'à ce qu'ils soient invoqués.
- Un service web a seulement connaissance de lui-même, mais pas de ses applications ou utilisateurs clients.
- Un service web n'est pas adaptable, et il n'est pas capable de bénéficier des nouvelles capacités de l'environnement afin d'apporter des services améliorés.
- L'autonomie, l'adaptation et la coopération, points faibles des services web, sont par ailleurs des domaines qui ont largement été explorés par la recherche dans les systèmes multi-agents.

les SMA peuvent etre considérer comme solution aux limites actuelles que présente des services web car, les services web sont souvent des interfaces de programmes orientés objet (développés via J2EE ou .NET) traités pour générer des descriptions WSDL afin de pouvoir communiquer avec des messages SOAP. Ils bénéficient alors des caractéristiques d'abstraction du paradigme orienté objet (encapsulation, héritage, passage de messages, etc). L'une des majeures évolutions consiste à passer d'une approche orienté objet des services vers une approche orienté agents. Les services tiendraient alors compte du contexte d'ouverture et de distribution de l'environnement, bénéficieraient des caractéristiques de proactivité, flexiblité

et d'adaptativité des agents, et deviendraient de plus réellement interactifs. En outre, comme les agents peuvent potentiellement participer à des interactions complexes tout en maintenant la coordination avec les autres agents, les services pourraient être dotés des pré-requis pour la modélisation d'une collaboration dynamique entre services[YASM 07].

3.5 Conclusion

Les services Web, les systèmes multi-agents ainsi que le Web sémantique offrent un cadre de travail pour assurer l'interopérabilité entre les applications. Nous avons présenté dans ce chapitre la technologie des agents, sa définition, ses caractéristiques ainsi que les SMA et le principe d'interopérabilité dans les SMAs et les services Web afin de donner une idée sur la réalisation d'une approche hybride entre SMA et WS (Web Service). En d'autres termes, cela consiste à intégrer les agents en tant que cadre conceptuel pour mettre en place des services Web sophistiqués (outils de découverte des services Web).
Dans le prochain chapitre nous présenterons notre approche qui propose intégration des services Web dans un environnement de système multi-agents, afin d'aboutir à une automatisation de la tache de découverte des services web sémantique par le bais des systemes multi-agents.

Chapitre 4

Approche basée agents pour la découverte des services web sémantique

4.1 Introduction

Les techniques actuelles de découverte des services web, effectuent seulement une recherche syntaxique basée sur mots clés avec une intervention humaine dans presque chaque étape de cycle de vie des services web. L'apparition de web sémantique a muni plusieurs chercheurs à l'exploiter pour palier aux limites actuelles de découverte des services web dans l'UDDI affin de rendre possible la recherche automatiquement des services Web en ce basant sur leurs capacités fonctionnelles et sémantique tel que leurs inputs, outputs, preconditions, effects, ...etc. Le web sémantique peut être exploité par un ensemble d'outils et d'applications affin d'optimiser au maximum la plateforme web actuel. Par exemple les systèmes multi-agents.

Dans ce chapitre nous allons présenter notre contribution qui est inspiré à partir de la recommandation du W3C dans [BOOT 04] qui a définie l'architecture d'un service web comme suit :*"Un service web est vu comme une notion abstraite qui doit être implémentée par une entité concrète".* C'est-à-dire nous avons proposé une architecture à base d'agents pour mettre en œuvre une nouvelle plateforme basée sur le web sémantique pour effectuer une découverte automatique des capacités des services web. Dans notre approche l'agent est une entité logicielle qui envoie et reçoit des messages, alors que les services web représente l'ensemble de fonctionnalités offertes.

4.2 Architecture du système

Dans cette section nous proposons l'architecture générale de système de découverte sémantique des services web à base d'agents. Les raisons pour les quelles on a choisie d'utiliser une plateforme multi-agents dans notre système sont les suivants :

- La nature distribuée du cycle de vie des services web.
- Les agents permettent de partager des connaissances (description sémantique).
- Les agents dialoguent entre eux et négocient la fourniture des services (publier, rechercher, exécuter et offrir l'information).

- L'interopérabilité, la coordination et la coopération permettent aux agents d'automatiser la résolution des problèmes distribués et complexes (découverte des services web en se basant sur des descriptions sémantiques OWL-S).
- Les agents peuvent travailler en parallèle lorsque l'environnement matériel et logiciel le permet (La modularité). C'est-à-dire, la publication et la découverte ainsi que l'exécution d'un service peuvent être réalisées simultanément d'une façon modulaire.
- La réutilisabilité, c'est-à-dire l'intégration facile des nouveau composants agents dans un système multi-agents (ajout des nouveau rôles).

La *figure 4.1* représente l'architecture globale de notre système proposé.

Notre architecture comporte deux catégories d'agents : les agents de système (Agent-annuaire, Agent-publication, Agent-découverte) ainsi que les agents acteurs (Agent-fournisseur et l'Agent-Client).

En plus des agents nous avons utilisés dans notre nouvelle architecture deux types de répertoire : le répertoire des agents et leurs services (RAS) et le répertoire des correspondances (RC). Le RAS représente la boite noire dans notre plateforme multi-agents où presque tous les agents de notre système partage leurs informations, ces informations sont relatives aux services web publiés par les fournisseurs ainsi que les identificateurs des agents fournissant ces services. Le RC est utilisé pour sauvegarder les correspondances entre les requêtes et les publications.

Les ontologies jouent un rôle clé dans notre architecture en fournissant des vocabulaires qui peuvent être utilisés par les applications afin de comprendre l'information partagée. Nous avons utilisé OWL (qui permet le raisonnement sur les concepts) pour décrire l'ontologie globale du domaine, nous avons aussi utilisé la description avec OWL-S car il supporte notre besoin pour la représentation sémantique des services web et puisque il est basé sur OWL, il permet de définir des relations entre les concepts. Cela signifie que nous pouvons utiliser OWL-S pour mettre en œuvre une découverte des fonctionnalités en utilisant un moteur d'inférence pour calculer les correspondances entre ces concepts.

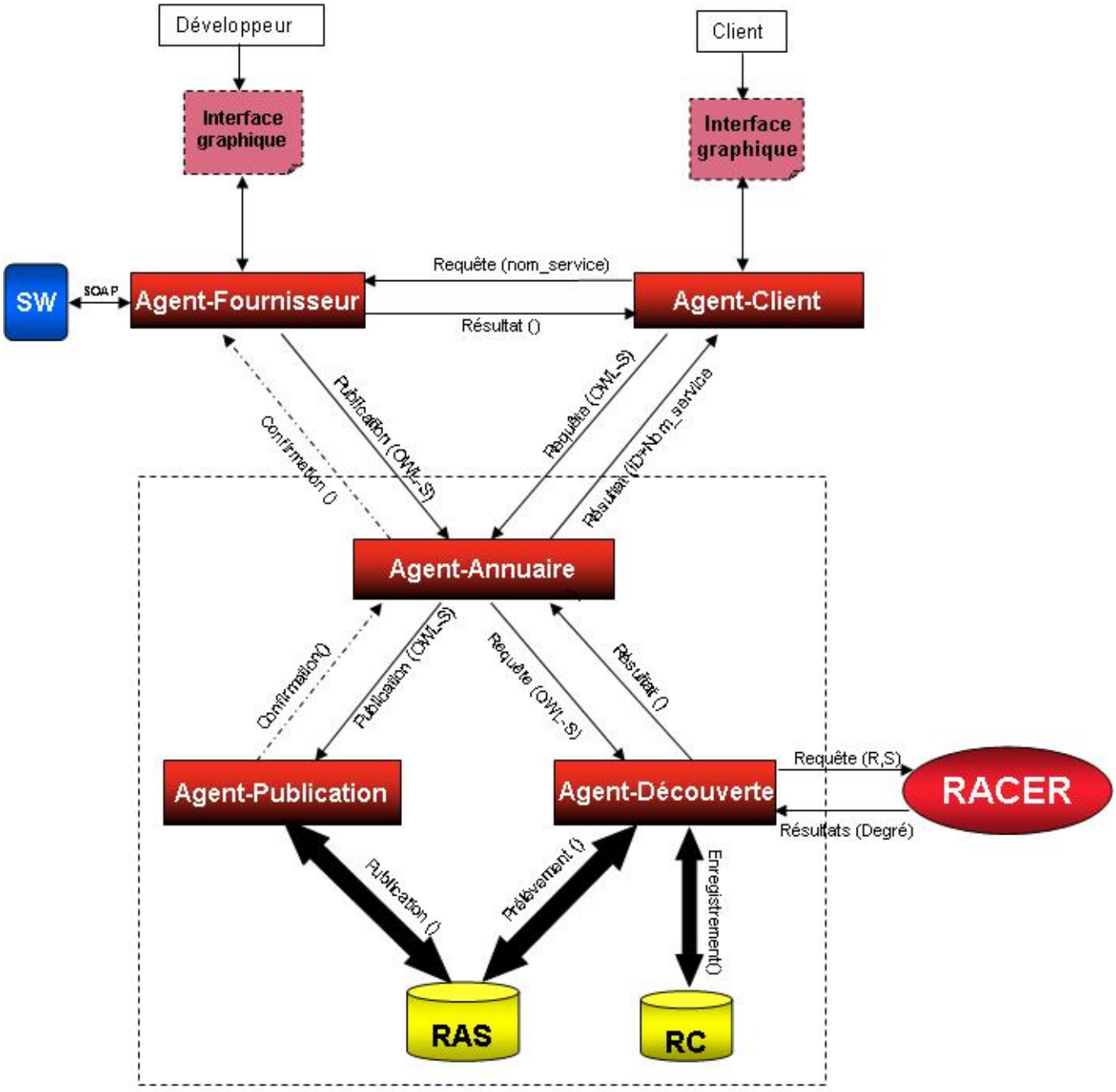

Figure 4.1 – Architecture globale du système

Puisque dans notre approche nous nous intéressons à l'adaptation de l'architecture des services web dans un système multi-agents afin d'automatiser la tache de découverte des services web, nous représenterons seulement un service web par les propriétés inputs et outputs (qui représentent ses attributs fonctionnels).

4.3 Modèle d'interaction entre agents

Le terme modèle d'interaction est un synonyme pour le terme protocole d'interaction utilisée dans la spécification FIPA. Un concepteur d'un système multi-agents a la responsabilité de rendre les agents suffisamment conscients de la signification des messages, des objectifs, des croyances et d'autres attitudes mentales que l'agent possède. En d'autres termes, à tout moment, l'agent doit savoir quelle est sa prochaine étape, selon le message qu'il reçoit des autres agents.

En général, un agent peut engager un dialogue avec d'autres agents, tout simplement en suivant le modèle d'interaction. Dans ce qui suit nous allons présenter les scénarios des modèles d'interaction envisagés pour notre architecture. Nous avons proposé deux types de scénarios, ces derniers représentent les différentes interactions entre les agents de notre système afin d'aboutir au final a une découverte sémantiques et automatique des services Web.

4.3.1 Scénario de publication

La phase de publication dans notre système proposé se déroule comme suit.

1. L'Agent-Fournisseur présentera une interface graphique sous forme d'un formulaire au fournisseur de service afin que ce dernier puisse saisir les informations relatives au service qu'il veut publier. L'Agent-Fournisseur génère et envoie la description OWL-S du service vers l'Agent-Annuaire sous une requête ACL (comme contenue de la requête).
2. L'Agent-Annuaire effectue une reconnaissance de type de la requête reçue.
3. Après avoir su que cette requête représente une publication il la remet à l'Agent-Publication.
4. L'Agent-Publication ce chargera de décryptage de la requête.

5. Le décryptage est réaliser affin de savoir s'il s'agit d'une nouvelle publication à enregistrer ou d'une modification d'une publication existante ou aussi d'une demande de suppression d'une publication existante.
6. Une fois l'action est exécuté un message de confirmation de l'action réalisé sera transmis à l'Agent-Annuaire.
7. Ce dernier renvoie la confirmation vers sa destination (l'Agent-Fournisseur).

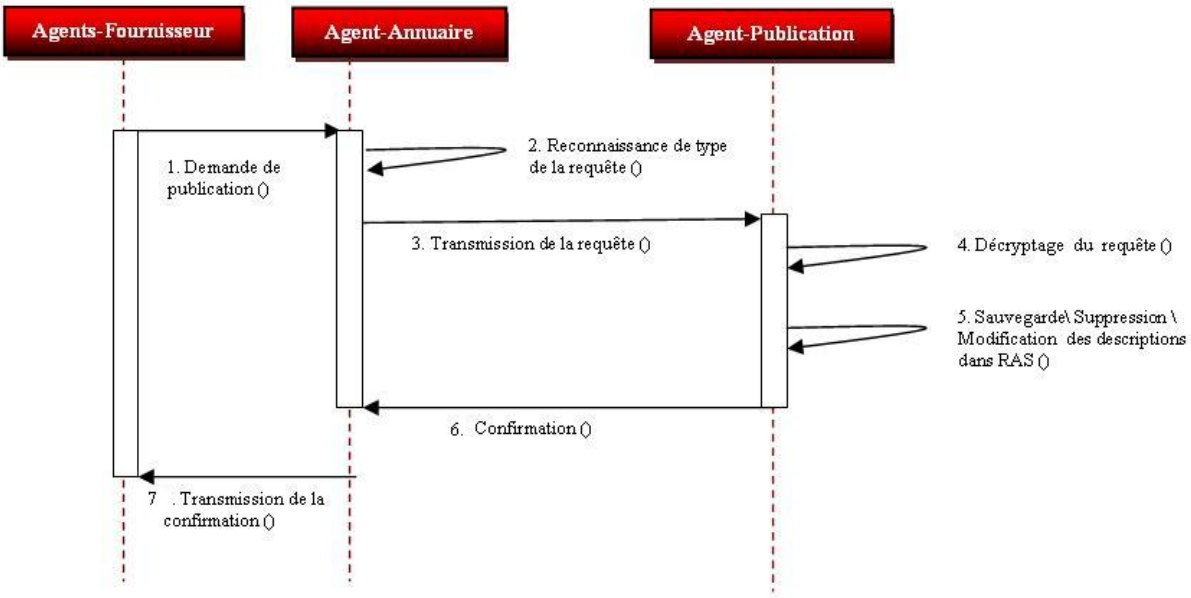

Figure 4.2 – Scénario de publication

4.3.2 Scénario de découverte

La phase de découverte sémantique des services Web se déroule comme suit.

1. Un client, active l'Agent-Client, ce dernier présentera une interface graphique sous forme d'un formulaire au client pour que ce dernier puisse décrire les différentes informations concernant les services qu'il veut chercher (nom du service recherché ainsi que ces inputs et outputs). Ce formulaire va être converti par l'Agent-Client en un fichier OWL-S (similaire au format de la publication pour faciliter la correspondance entre le service et la requête) qui représentera la description sémantique de service a recherché. L'Agent-Client envoie ensuite la description de service recherché à l'Agent-Annuaire.

2. Ce dernier effectue une reconnaissance de types de la requête.
3. Après savoir qu'il s'agit d'une requête de recherche d'un service il là remit a l'Agent-Découverte qui ce chargera de la recherche sémantique de service Web souhaité.
4. L'Agent-Découverte prélève les descriptions OWL-S ainsi que les identifiants de leurs agents fournisseurs, publiés dans le RAS.
5. Ensuite, il les compare avec la requête de l'utilisateur à l'aide d'un moteur d'inférence en lui envoyant les concepts de la requête et la publication.
6. Le moteur d'inférence retourne le résultats de correspondance a l'Agent-Découverte.
7. Ce dernier sauvegarder les résultats de correspondance dans le registre des correspondances RC.
8. Une fois le processus de correspondance effectuer, l'Agent-Découverte procède au processus de sélection de la meilleur découverte pour choisir le service le plus approprié a la demande de l'utilisateur.
9. Une fois le service sélectionné l'Agent-Découverte envoie l'identifiant de l'Agent-Fournisseur ainsi que le nom de service recherché à l'Agent-Annuaire.
10. Ce dernier retransmis le résultat obtenue au client de service.
11. L'Agent-Client et après avoir reçue les informations nécessaires et validation du résultat par le client il contacte l'Agent-Fournisseur pour l'invocation du service trouvé.
12. Enfin de scénario l'Agent-Fournisseur et après négociation des contrainte d'exécution du service il invoque le service demander via le protocole SOAP, et envoie a l'Agent-Client le résultat de l'exécution du service web.

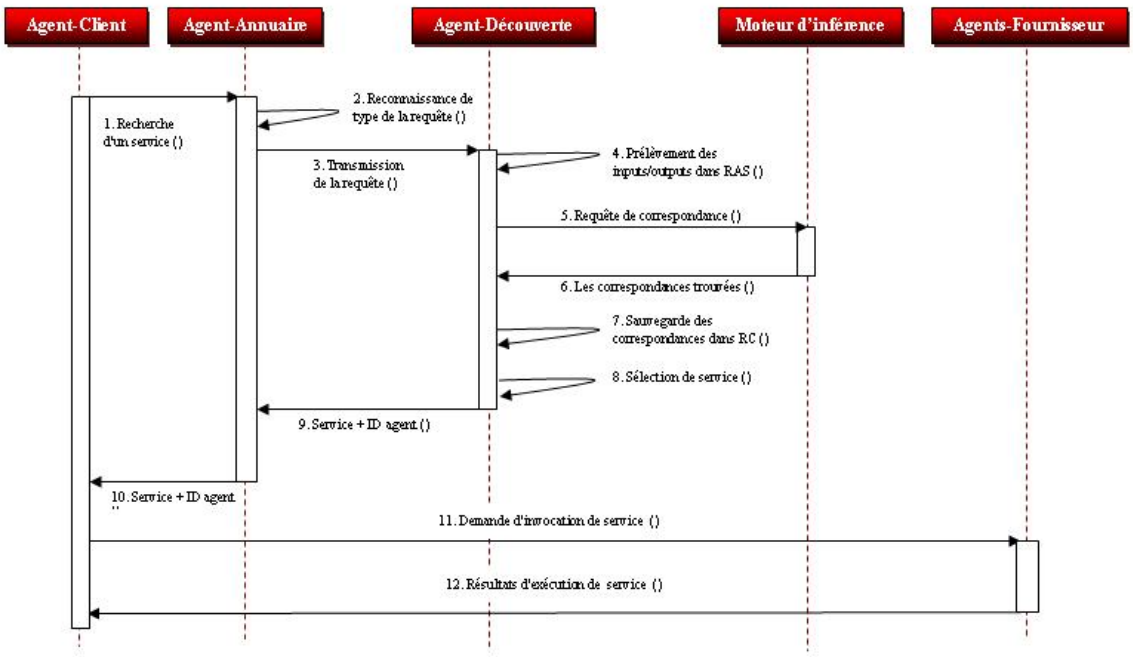

Figure 4.3 – Scénario de découverte

4.4 Description des composants de l'architecture du système

Nous allons présenter dans cette section l'architecture interne des différents agents participant dans notre système ainsi que leurs systèmes de fonctionnement et les types des messages échangés entre eux.

4.4.1 Agent Client

L'Agent-Client fait partie des agents acteurs de notre système car c'est l'agent responsable de déclanchement de processus de découverte d'un service web. L'Agent-Client représente l'unité intermédiaire entre un utilisateur demandeur de service et l'unité responsable de la recherche des services web, il représente aussi l'unité intermédiaire entre un demandeur de service et l'Agent-Fournisseur propriétaire de service recherché. Il comporte trois mo-

dules internes (le module présentation, générateur OWL-S et un module de communication inter-agents). L'Agent-Client assure les suivant.

- Il offre une interface HTML sous forme d'un formulaire par le biais de module de présentation à l'utilisateur afin que ce dernier puisse saisir les paramètres fonctionnels (nom de service, input, output, precondition, effect,. . .. . .etc.) nécessaire pour la description de service web a recherché.
- Après le remplissage du formulaire l'Agent-Client fait appel au composant Générateur OWL-S pour traduire les différentes informations saisie par l'utilisateur en description ontologique représentant toutes ces informations, et génère le résultat de traduction sous forme d'un fichier OWL-S représentant la demande de l'utilisateur comme requête R qui va être envoyé via le module de communication comme contenue d'un message ACL vers l'Agent-Annuaire.
- Une fois le service recherché est trouvé, c'est à dire, une fois que l'Agents-Annuaire trouve le service S correspondant a la demande R de l'utilisateur, il envoie a l'Agent-Client un message ACL comportant des informations sur l'identité du l'agent fournissant le service souhaité ainsi que le nom exacte de service S publier par le fournisseur affin que ce dernier procède a l'invocation de service web via le protocole SOAP et envoie le résultat d'exécution de service a l'Agents-Client.

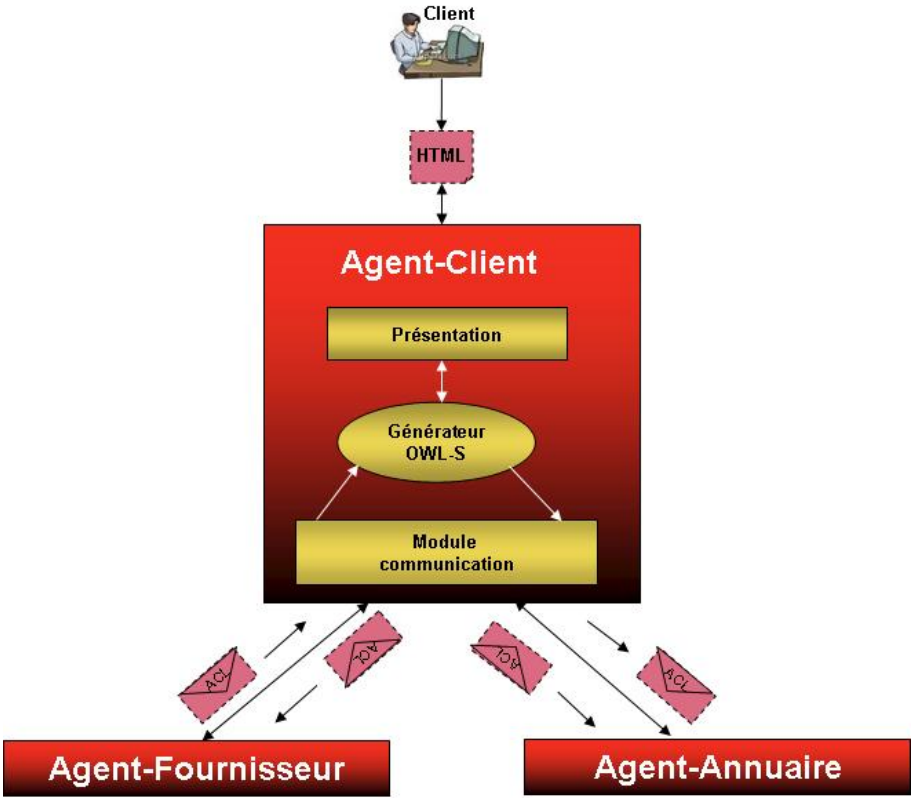

Figure 4.4 – Architecture de l'Agent-Client

4.4.2 Agent Fournisseur

Comme l'Agent-Client, l'Agent-Fournisseur fait partie de la catégorie des agents acteurs de notre système. Il est le responsable de déclanchement du scénario de la publication des services web. L'Agent-Fournisseur comporte quatre modules internes (le module présentation, générateur OWL-S, communication et une API d'invocation des services web). Il assure les rôles suivant.

- Il offre une interface HTML (par le biais de module de présentation) au fournisseur (développeur de l'application à publier) pour saisir ou charger la description OWL-S du service S à publier ou supprimer une de ses publications.
- Le deuxième rôle de l'Agent-Fournisseur est de communiquer avec l'Agent-Annuaire via le module de communication pour publié ses services S en lui envoyant la description OWL-S de service comme contenue du message ACL après avoir sauvegarder une copie de sa description dans sa base de données interne.

- Le troisième rôle de l'Agent-Fournisseur est l'invocation du service web à travers l'API (Application Programming Interface) d'invocation. Cela ce produit lorsque l'Agent-Client le contacte pour invocation d'un service. L'Agent-Fournisseur prélève la partie Grounding de la description de service demandé dans sa base de données, afin de procéder à l'invocation de ce dernier et retourner le résultat de l'exécution du service vers l'Agent-Client par le biais de module de communication.
- Un autre rôle de l'Agent-Fournisseur et d'offrir au fournisseur le choix de modifier ou supprimer une publication en envoyant a l'Agent-Annuaire une requête contenant le nom du service S à supprimer ou a modifier. Dans le cas d'une modification notre système propose de changer totalement l'ancienne publication par la nouvelle description.

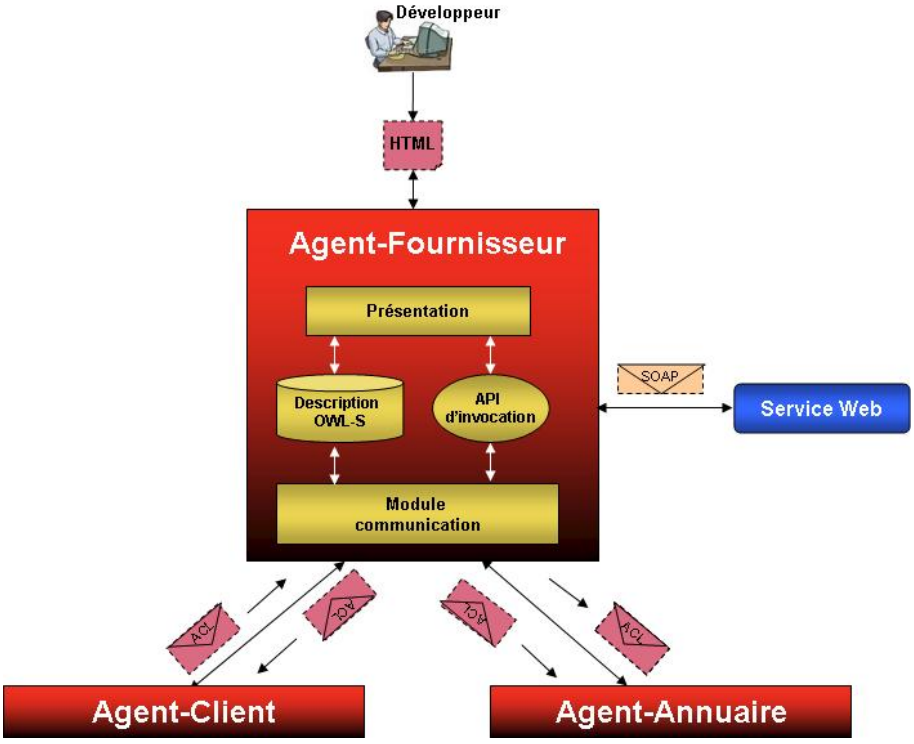

Figure 4.5 – Architecture de l'Agent-Fournisseur

4.4.3 Agent Annuaire

Si on compare avec l'architecture actuelle des services web, l'Agent-Annuaire est similaire à l'API de communication avec le registre UDDI. Dans notre architecture multi-agents,

l'Agent-Annuaire joue le rôle de pivot de notre plateforme (figure 4.6), il gère la transaction des messages entre la plateforme multi-agents interne de découverte et publication des services web et les agents acteurs de notre système.

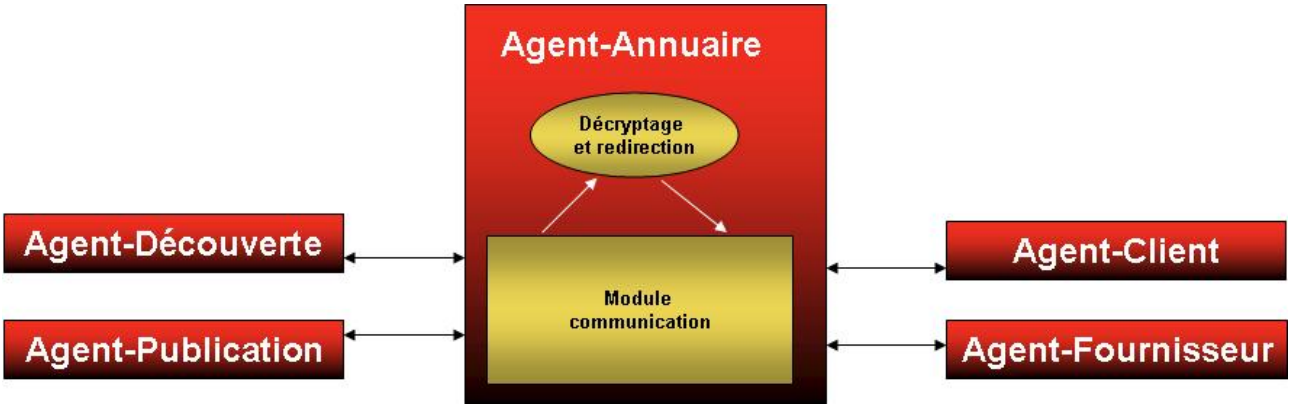

Figure 4.6 – Architecture de l'Agent-Annuaire

l'Agent-Annuaire comporte deux modules internes (le module décryptage et le module communication inter-agents). Il assure les rôles suivants.

- Quand une requête arrive à l'Agent-Annuaire il là décrypte. S'il s'agit d'une requête de publication, il l'envoie à l'Agent-Publication, dis que ce dernier exécute cette requête il renvoie un message de confirmation de l'exécution de la requête envoyé à l'Agent-Annuaire (via le module de communication) qui a son tour renvoie cette confirmation a l'Agent-Fournisseur.
- Si le message arrivé représente une requête R de découverte d'un service web, l'Agent-Annuaire là renvoie vers l'Agent-Découverte qui ce chargera de la recherche et la sélection du service souhaité.
- Une fois le service trouvé l'Agent-Découverte envoie le résultat de la recherche (ID Agent-Fournisseur plus le nom de service S trouvé), vers l'Agent-Annuaire, ce dernier et après savoir que c'est le résultat de découverte il le renvoie a l'Agent-Client l'originaire de la requête R.

Nous pouvons représenté les interactions de l'Agent-Annuaire dans le digramme UML suivant :

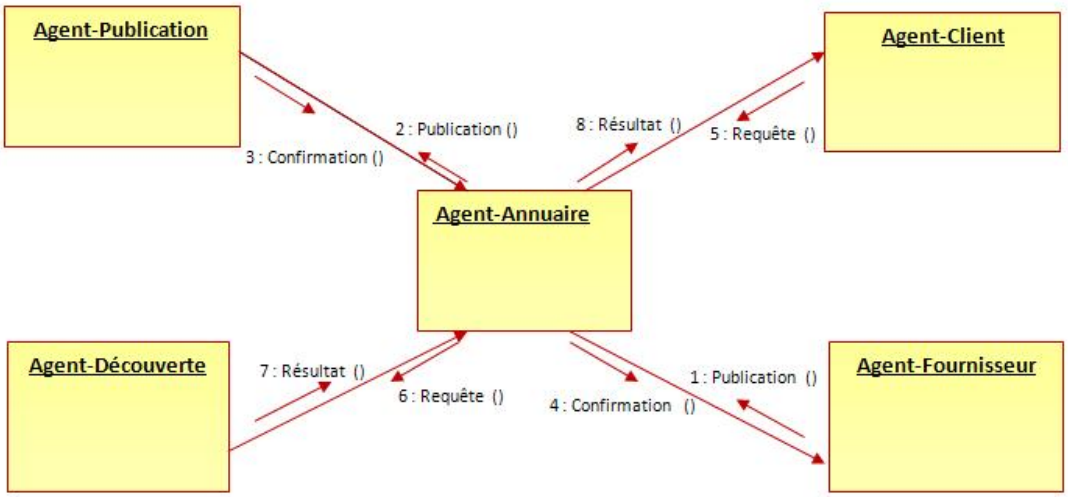

Figure 4.7 – Diagramme UML Pour les interactions de l'Agent-Annuaire

4.4.4 Agent Publication

L'Agent-Publication représente l'unité responsable de l'enregistrement des publications (Identifiant des agents fournisseurs ainsi que la description sémantique de leurs services web) dans notre système. L'architecture de l'Agent-Publication est présentée dans la *figure 4.8*.

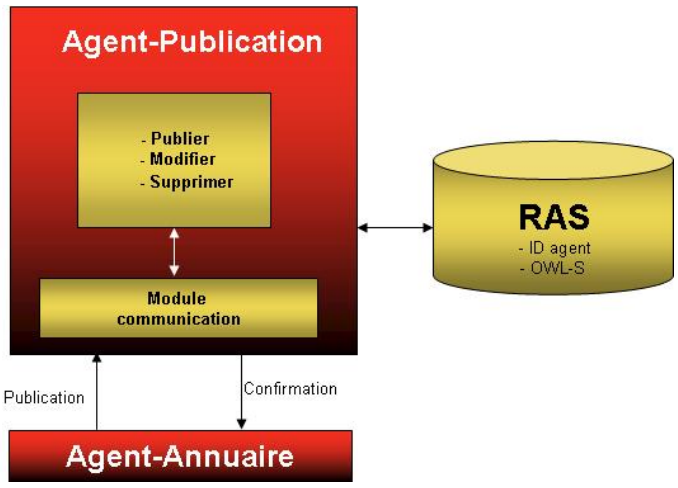

Figure 4.8 – Architecture de l'Agent-Publication

Comme illustrer dans la *figure 4.8*, l'Agent-Publication utilise le registre appeler RAS (Registre des Agents et leurs Services) affin de sauvegarder les informations concernant une publication donnée, ces informations représente l'identifiant de l'Agent-Fournisseur du service ainsi que la description ontologique OWL-S du service S offert par ce fournisseur. L'Agent-Publication offre les rôles suivants :

- Quand l'Agent-Publication reçoit une requête de la part de l'Agent-Annuaire via le module communication, et s'il s'agit d'une nouvelle publication il enregistre (publie) dans le RAS, l'identifiant de l'Agent-Fournisseur ainsi que la description OWL-S du service S qu'il offre. Une fois le service S publié, l'Agent-Publication envoie une requête de confirmation de publication à l'Agent-Annuaire.
- Dans le cas ou il s'agit d'une requête de modification l'Agent-Publication recherche dans le RAS l'identifiant du l'Agent-Fournisseur qui cherche à modifier la description de son service et remplace son ancienne description par la nouvelle description envoyer dans la requête de l'Agent-Annuaire. Une fois l'opération terminé, il envoie un message de confirmation vers l'Agent-Annuaire.
- Dans le cas ou il s'agit d'une suppression, l'Agent-Publication recherche dans RAS l'identifiant de l'Agent-Fournisseur afin de le supprimer avec la description de son service S publié.

4.4.5 Agent Découverte

L'Agent-Découverte est l'agent responsable de la tache de recherche et sélection des services web souhaité dans notre système. Comme présenter dans le chapitre -2-, il existe plusieurs travaux sur la découverte sémantique des services web [PAOL 02][SYCA 02][SYCA 03][PAOL 04] [GONZ 01][TRAS 01] [BENA 02] [VERM 03]. On peut dire que, malgré que l'approche de Paolucci dans [PAOL 02] ce limite a l'utilisation des attributs fonctionnels inputs/outputs pour l'adaptation de son système de découverte sémantique des servies web, elle reste l'approche la plus simple pour résoudre ce problème, vue que les auteurs se basent sur le langage de modélisation DAML-S qui offre une description plus riche, simple et détaillée sur les services web et aussi le fait que les auteurs ont intégrer leur solution au dessus des standards connu tel que UDDI et WSDL, ceci nous mène à exploiter le principe de *Paolucci*,

afin de l'adapter à notre approche pour réalisé une plateforme à base de système multi-agents afin de résoudre le problème de découverte dynamique des services web. L'architecture de l'Agent-Découverte est présentée dans la *figure 4.9*.

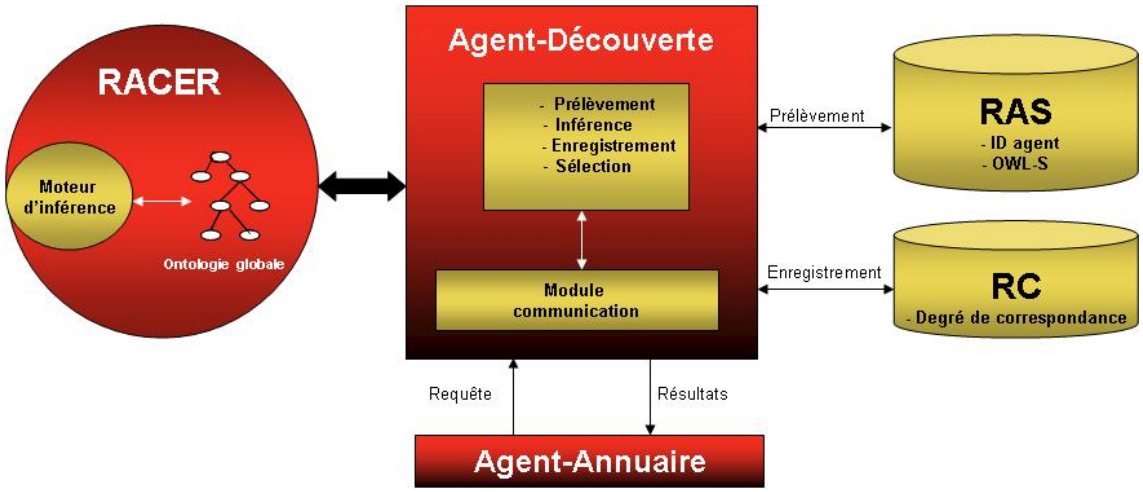

Figure 4.9 – Architecture de l'Agent-Découverte

Comme illustrer dans la figure ci-dessus, l'Agent-Découverte utilise un moteur d'inférence sur les descriptions ontologique (RACER) plus un registre de correspondance (RC).

4.4.5.1 Moteur d'inférence

Les techniques de recherche actuels des services web sont basées sur la comparaison de chaîne de caractère (recherche par mots clés) qui est, comme mentionné dans les chapitres précédents, peut causer des problèmes de performance et produire des résultats incomplets. Ainsi, l'utilisation d'un outil d'inférence sur des descriptions ontologiques pour palier aux problèmes précédents, est la partie atout dans notre système d'automatisation de la tache de découverte des services web. Il existe plusieurs outils d'inférence sur les descriptions ontologiques (RACER [RACER], PELLET [PELL 04], . . .etc.) mais l'outil RACER a montré sa performance dans plusieurs domaines d'applications des ontologies. Dans le scénario de découverte de notre système, l'Agent-Découverte fait appel au moteur d'inférence qui exploite les ontologies des services (les descriptions OWL-S) ainsi que l'ontologie globale de domaine (nous supposons que l'Agent-Découverte envoie au RACER l'URL de l'ontologie globale du

domaine pour chaque teste de correspondance) pour calculer le degré de correspondance entre la requête et le service.

Définition 4.1. *Soit in_R et out_R, les entrées offertes et les sorties requises par une requête R, et soit in_S et out_S, les entrées requises et les sorties offertes par un service web S. Un service web S peut répondre à la requête R si les entrées requises par R sont équivalentes ou englobent (selon l'ontologie globale de domaine) celles offertes par S, et si les sorties offertes par S sont équivalentes ou englobent celles requises par R [PAOL 02].*

Dans [PAOL 02] les auteurs ont défini quatre niveaux de correspondances (figure 4.10), ces niveaux dépondent de la relation entre les concepts associer a chaque entrée/sorties de la requête R et la publication S.

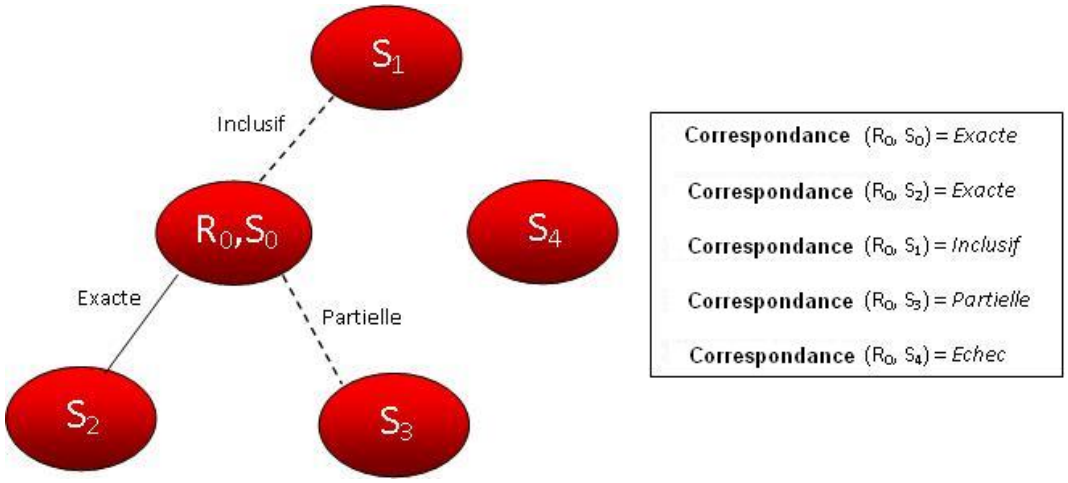

Figure 4.10 – Niveaux de correspondances

➢ La correspondance exacte

Le RACER reconnais une correspondance exacte entre le service S et la requête R, lorsque out_R=out_S. Un tel service répond exactement aux besoins de la requête. Aussi la correspondance peut être exacte, lorsque out_R est une sous-classe directe de out_S.

➢ La correspondance inclusive

Le RACER reconnais une correspondance inclusif (Plugin), lorsque out_S super-classe-de out_R. Cette règle reconnaît qu'il y a une faible relation entre out_R et out_S, dans ce cas par

exemple, on peut s'attendre à ce que un service qui a comme output *véhicule* offre un certain type de *voitures*, mais nous ne pouvons pas attendre qu'il offre chaque types de sa sous-classe directe.

➢ La correspondance partielle

Dans le cas de correspondance partielle (subsumes), la requête est plus générique que le service trouvé. En d'autres termes : out_R est une super-classe-de out_R. Dans ce cas, le service S peut être retenu, mais ce service peut ou ne peut pas satisfaire la requête, il ce peut qu'un autre service peut être recherché pour compléter le manque (composition des services Web).

➢ Cas d'échec

Si aucune des relations de correspondance précédentes n'a été identifiée on dit que le service S et la requête R sont incompatibles. En d'autre terme, le RACER reconnaît un échec (Fail).

L'algorithme suivant représente (Algorithme 1) l'algorithme qu'utilise le RACER pour calculer le degré des correspondances entre les couples des concepts associés à la requête et la publication.

Algorithm 1 DegréDeCorrespondance (out_R,out_S)

if out_S=out_R **then**
  return exact
else if out_R sous-classe-de out_S **then**
  return exact
else if out_S super-classe-de out_R **then**
  return inclusif
else if out_R super-classe-de out_S **then**
  return partiel
else
  return echec
end if

4.4.5.2 Le registre de correspondances RC

Le registre RC est utiliser par l'Agent-Découverte pour sauvegarder les correspondances trouvées par le RACER entre les couples (S, R).

4.4.5.3 Fonctionnement de l'Agent-Découverte

Le processus de découverte de l'Agent-Découverte est inspiré des algorithmes proposés dans [PAOL 02]. Le principal algorithme (Algorithme 2) utiliser par l'Agent-Découverte est celui de correspondance entre les sorties des services publiés dans le registre RAS et les sorties de la requête R, l'algorithme de correspondance entre les entrées des services et le entrée de la requête et le même sauf que l'ordre entre le service et la requête est inversé.

Algorithm 2 CorrespondanceDesOutput($outputR$, $outputsS$)

for all out_S **in** $outputsS$ **do**

$DegreCorrespondance \longleftarrow Max$(DegreDeCorrespondance($out_R$, out_S))

end for

if $DegreCorrespondance \neq echec$ **then**

Enregistrer(S, ID)

end if

Dis que l'Agent-Annuaire envoie la requête de recherche a l'Agent-Découverte ce dernier prélève la sortie de la requête R, puis il prélève dans RAS une publication S (ses concepts de sorties) ainsi que le ID de l'Agent-Fournisseur, après, il envoie chaque couple (outputR,outputS) au moteur d'inférence (RACER) qui exploite l'ontologie globale OWL du domaine pour trouver ou non une correspondance entre la sortie de la requête R et la sortie de la publication S et retourner le résultats de l'inférence (le degré de l'inférence) a l'Agent-Découverte. Dans le cas où le RACER reconnais une correspondance il retourne un degré de correspondance différent de Echec, alors la publication actuel S est retenue par l'Agent-Découverte qui prélevé le nom du service S et l'enregistre dans le registre RC avec l'identifiant de l'Agent-Fournisseur du service S.

Après avoir sauvegardé les correspondances trouvés, l'Agent-Découverte utilise l'algorithme de trie (Algorithme 3), pour trier les correspondances trouvées sachant que la cor-

respondance exact est la préférer, en deuxième niveau viens la correspondance inclusif et en troisième niveau, la correspondance partiel. Le principale critère de trie est de sélectionner la correspondance avec le haut degré dans les outputs. La correspondance des inputs est utilisée seulement en deuxième lieu pour séparer les correspondances des outputs avec le même degré.

Algorithm 3 TrieCorrespondance(*corresp*1, *corresp*2)

if *corresp*1.*output* > *corresp*2.*output* **then**
  corresp1 > corresp2
else if *corresp*1.*output* = *corresp*2.*output* **and** *corresp*1.*input* > *corresp*2.*input* **then**
  corresp1 > corresp2
else if *corresp*1.*output* = *corresp*2.*output* **and** *corresp*1.*input* = *corresp*2.*input* **then**
  corresp1 = corresp2
end if

4.5 conclusion

Dans ce chapitre, nous avons présenté notre architecture basée agents pour automatiser la tache de découverte des services web. Nous avons aussi présenté l'architecture interne de chaque agent utilisé ainsi que leur principe de fonctionnement, plus les déférents modes d'interaction entre ses agents. Nous avons présenté deux catégories d'interaction ou scénario, un scénario pour la publication et l'autre pour la découverte des services web. Nous avons utilisé le langage OWL-S pour la description sémantique des services web ainsi que OWL pour la description de l'ontologie globale commune de domaine des agents. Enfin, nous pouvant conclure que le comportement des agents dans notre système permet d'approcher de plus près de celui de l'homme en essayant de localiser automatiquement les services web souhaités.

Chapitre 5

Etude de cas

5.1 Introduction

Dans le chapitre précédent nous avons proposé un modèle de découverte sémantique des services web dans un environnement multi-agents. Afin de montrer la validité et fiabilité du modèle proposé, nous allons faire une étude de cas, d'où nous allons appliquer notre modèle sur l'exemple de vente des téléphones portables sur le web.

5.2 Description du problème

Considèrant le scénario suivant : soient VENDEUR1, VENDEUR2 et VENDEUR3 trois directeurs pour des magasins de ventes des téléphones. Chaqu'un d'entre eux veut créer une application de vente sur le Web la fonction principale de ces applications c'est qu'elles peuvent donner les téléphones qui peuvent êtres vendue avec un tel ou tel prix. Pour cela ils font appel a leurs informaticiens pour procéder au développement de ces applications et enfin les faire publier sur le Web comme des services web. Pour cela, et après avoir développer les applications souhaités les informaticiens doivent installer chez eux un agent locale (Fournisseur ou Client). Dans le cas des vendeurs, ils installent des agents Fournisseurs, ces dernier s'occuperont de processus de publication des services web.
Considèrant maintenant le scénario suivant : ALI est un jeune informaticien qui habite à Alger et désire acheter un téléphone portable NOKIA série N (N96 sinon N73 sinon N71, . . .etc.), cela dépond du prix de ces téléphones portables par ce qu'il n'a qu'une petite somme d'argents et il ne connais pas leurs prix. C'est pour cela qu'il décide de consulter Internet affin de chercher si il trouve des fournisseurs (services web) qui offre la possibilité de vérifier quel portable NOKIA série N il peut acheter avec l'argent qu'il possède. Pour cela il installa l'agent locale (Agent Client) chez lui et commençera sa recherche.

5.3 Exigence de la solution envisagée

La description d'un service web est décomposée en trois éléments : le profile de service <ServiceProfile>, le modèle de service <ServiceModel>, et les connaissances de service

<ServiceGrounding>. Pour notre travail nous utilisons que la partie <ServiceProfile> car elle contient les informations nécessaires pour la découverte des services web.

Dans notre travail nous avons modélisé les services web publiés ainsi que la requête du client avec la description ontologique OWL-S qui est un langage de description de service web comme présenter dans les chapitres précédents. Nous allons aussi utiliser le langage OWL pour la description de l'ontologie globale utilisé (Telephone.owl). Nous allons aussi utiliser RACER pour l'inférence entre les concepts ontologique des couples (service, requête). Pour le développement des agents du système nous avons choisie la plateforme JADE comme environnement de notre solution proposé.

5.4 La solution proposée

Comme nous avons expliqué plus haut, notre problématique c'est que ALI veut acheter un téléphone portable en utilisant les services web, notre système va permettre aux vendeurs de publiés leurs services web et a ALI de recherché le service désirer en effectuant une recherche automatique de service le plus précis qui correspond a sa demande. Pour illustrer cela nous allons présenter les différentes étapes de scénario proposé avec notre système sachant que notre modèle basé agents proposé résous le problème de publication, découverte et invocation des services web (ServiceProfile, ServiceModel, ServiceGrounding). Dans notre cas d'étude nous avons utilisé que la partie profile d'un service web puisque c'est suffisant pour démontrer notre approche basée agents pour la découverte automatique (sémantique) des services web.

Le déroulement de notre solution est comme suit :

➢ Premièrement, l'informaticien de premier vendeur va exécuter son agent local (Agent-Fournisseur), ce dernier va lui présenter une interface comme formulaire pour que le vendeur puissent saisir ses cordonnées, cette interface est représenté dans la figure suivante *figure 5.1.*

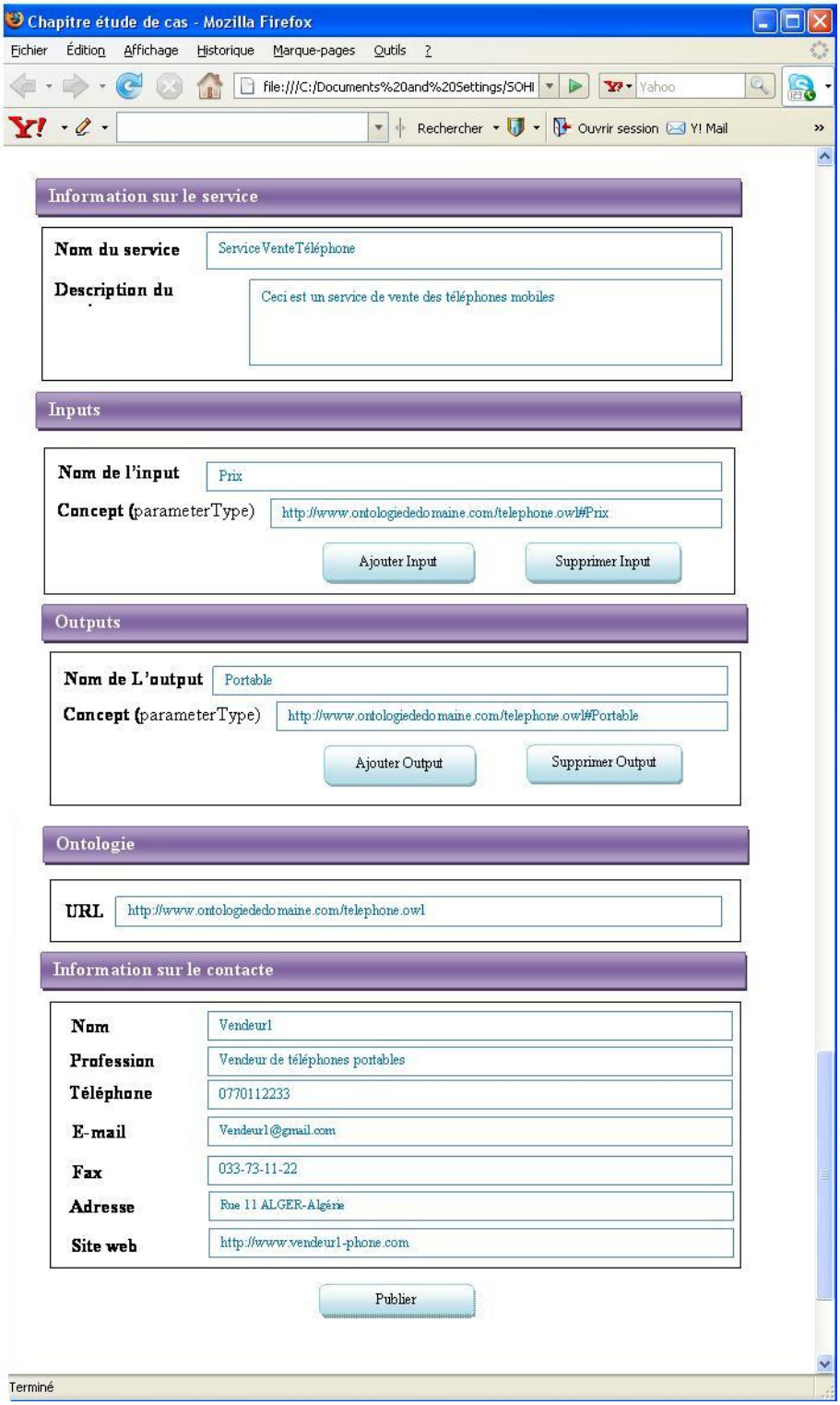

Figure 5.1 – Formulaire présenté par l'Agent-Fournisseur

Comme présenter dans le *chapitre -2-* (ontologie de service profile) la partie en haut du formulaire représente les informations fonctionnelles sur les quels est basée notre approche c'est-à-dire les capacités d'un service web (inputs/outputs) Les inputs sont ce qui est requis par un service afin de produire l'output désirée, ainsi que les informations sur le service et l'ontologie de domaine globale utilisée.

Concernant le bas du formulaire, il représente les informations non-fonctionnelles il constitue la définition de l'acteur (Actor) : il enregistre des informations sur le fournisseur du service.

Enfin, une fois le premier vendeur a saisie ces informations l'agent-Fournisseur traduit les informations saisies en fichier OWL-S en utilisant son outils intégrer OWLSAPI, le fichier générer est représenter dans la *figure 5.2*, ce fichier est nommé par le nom du service publier c'est-à-dire dans notre exemple *ServiceVenteTelephone.owl*.

```
    ----------------------------------------

<owl:Ontology about="">
   <owl:imports rdf:resource= "&service;" />
   <owl:imports rdf:resource= "&profile;" />
   <owl:imports rdf:resource= "&process;" />
   <owl:imports rdf:resource= "&actor;" />
   <owl:imports rdf:resource= "&addParam;" />
  <owl:imports rdf:resource= "http://www.ontologiededomaine.com/telephone.owl"/>
 </owl:Ontology>

<profile:Profile>
 <profile:serviceName>ServiceVenteTelephone</profile:serviceName>
     <profile:textDescription> Ceci un service de vente des téléphones mobiles
     </profile:textDescription>

<profile:contactInformation>
      <actor:Actor rdf:ID="VENDEUR1">
        <actor:name>VENDEUR1</actor:name>
        <actor:title>vendeur de téléphones portables</actor:title>
        <actor:phone>0770112233</actor:phone>
        <actor:fax>vendeur1@gmail.com </actor:fax>
        <actor:email>033-73-11-22</actor:email>
        <actor:physicalAddress>Rue 11 ALGER-Algérie</actor:physicalAddress>
        <actor:webURL>http://www.vendeur1-phone.com</actor:webURL>
      </actor:Actor>

</profile:contactInformation>

 <profile:hasInput>
  <process:Input rdf:ID="prix-input">
    <process:parameterType>http://www.ontologiededomaine.com/telephone.owl#prix
    </process:parameter Type>
  </process:Input>
 </profile:hasInput>


. <profile:hasOutput>
   <process:UnConditionalOutput rdf:ID="portable-output">
     <process:parameterType>http://www.ontologiededomaine.com/telephone.owl#portable
     </process:parameterType>
    </process:UnConditionalOutput>
  </profile:hasOutput>

</profile:Profile>
```

Figure 5.2 – La description OWL-S du service proposé par Vendeur1

Cette description montre que les inputs attendus par le service sont des instances du concept *Prix* par contre les outputs du service générés sont des instances du concept *portable*.

➢ Considèront maintenant un autre vendeur de téléphones (Vendeur2) qui lui aussi fait appel à son développeur afin de développer et publier son service web, alors, après l'installation de l'application Agent-Fournisseurs localement, ce dernier lui propose la même interface présenter dans la *figure 5.1*, a la fin de la saisie des informations sur le service à publier et en appuyant sur le bouton *Publier*, l'Agent-Fournisseur génère automatiquement la description OWL-S correspondante dans le fichier *ServiceTelephonePortable.owl* (Figure 5.3).

```
    ----------------------------------------

<owl:Ontology about="">
   <owl:imports rdf:resource= "&service;" />
   <owl:imports rdf:resource= "&profile;" />
   <owl:imports rdf:resource= "&process;" />
   <owl:imports rdf:resource= "&actor;" />
   <owl:imports rdf:resource= "&addParam;" />
   <owl:imports rdf:resource= "http://www.ontologiededomaine.com/telephone.owl"/>
</owl:Ontology>

<profile:Profile>
    <profile:serviceName> ServiceTelephonePortable </profile:serviceName>
    <profile:textDescription> Ceci un deuxième service de vente des téléphones mobiles
</profile:textDescription>

<profile:contactInformation>
      <actor:Actor rdf:ID="VENDEUR2">
        <actor:name>VENDEUR2</actor:name>
        <actor:title> vente des téléphones portables et leurs accessoires </actor:title>
        <actor:phone>0770223344  </actor:phone>
        <actor:fax>VENDEUR2@gmail.com </actor:fax>
        <actor:email>033-73-22-33</actor:email>
        <actor:physicalAddress>Rue 12 ALGER-Algérie</actor:physicalAddress>
        <actor:webURL>http://www.vendeur2-phone.com</actor:webURL>
      </actor:Actor>

</profile:contactInformation>

 <profile:hasInput>
     <process:Input rdf:ID="prix-input">
       <process:parameterType>http://www.ontologiededomaine.com/telephone.owl#prix
       </process:parameter Type>
     </process:Input>
 </profile:hasInput>

   <profile:hasOutput>
     <process:UnConditionalOutput rdf:ID="serieN-output">
        <process:parameterType>http://www.ontologiededomaine.com/telephone.owl#serieN
        </process:parameterType>
      </process:UnConditionalOutput>
   </profile:hasOutput>

   </profile:Profile>
```

Figure 5.3 – La description OWL-S du service proposé par Vendeur2

Cette description montre que les inputs attendus par le service sont des instances du concept *Prix* comme définit dans l'ontologie *Telephone*. Par contre les outputs du service générés sont des instances du concept *SerieN*.

➢ Ensuite considèront un troisième service publié par le Vendeur3, la même chose que les exemples précédents, une fois que les informations concernant le service web à publier son saisie l'Agent-Fournisseur générera la description OWL-S correspondante dans le fichier *ServiceTelephone.owl* présenter dans la figure suivante (figure 5.4).

```
-----------------------------------------
<owl:Ontology about="">
    <owl:imports rdf:resource= "&service;" />
    <owl:imports rdf:resource= "&profile;" />
    <owl:imports rdf:resource= "&process;" />
    <owl:imports rdf:resource= "&actor;" />
    <owl:imports rdf:resource= "&addParam;" />
    <owl:imports rdf:resource= "http://www.ontologiededomaine.com/telephone.owl"/>
</owl:Ontology>

<profile:Profile>
  <profile:serviceName>ServiceTelephone</profile:serviceName>
    <profile:textDescription> Ceci un troisième service de vente des téléphones
</profile:textDescription>
 <profile:contactInformation>
       <actor:Actor rdf:ID="VENDEUR3">
         <actor:name>VENDEUR3</actor:name>
         <actor:title>vendeur de téléphones </actor:title>
         <actor:phone>0770334455</actor:phone>
         <actor:fax>vendeur3@gmail.com </actor:fax>
         <actor:email>033-73-33-44</actor:email>
         <actor:physicalAddress>Rue 13 ALGER-Algérie</actor:physicalAddress>
         <actor:webURL>http://www.vendeur3-phone.com</actor:webURL>
       </actor:Actor>
 </profile:contactInformation>

  <profile:hasInput>
      <process:Input rdf:ID="prix-input">
        <process:parameterType>http://www.ontologiededomaine.com/telephone.owl#prix
        </process:parameter Type>
      </process:Input>
  </profile:hasInput>

  <profile:hasOutput>
      <process:UnConditionalOutput rdf:ID="fixe-output">
          <process:parameterType>http://www.ontologiededomaine.com/telephone.owl#fixe
          </process:parameterType>
       </process:UnConditionalOutput>
    </profile:hasOutput>

 </profile:Profile>
```

Figure 5.4 – La description OWL-S du service proposé par Vendeur3

Cette description montre que les inputs attendus par le service sont des instances du concept *Prix* comme définit dans l'ontologie *Telephone*, par contre les outputs du service générés sont des instances du concept *fixe*.

➢ Comme présenter dans notre modèle proposé dans le chapitre précédent chaque Agent-Fournisseur après la génération de la description OWL-S, il envoie cette publication encapsuler dans une requête ACL comme contenue de message vers l'Agent-Annuaire qui se situe dans un serveur d'annuaires de services web. Une fois que l'Agent-Annuaire sait que c'est une publication il la renvoie vers l'Agent-Publication qui se charge de sauvegarder ces publication dans le registre RAS (IDagent+OWL-S). Une fois la sauvegarde réalisée il envoie une confirmation vers l'Agent-Annuaire qui à son tour renvoie cette confirmation vers chaque Agent-Fournisseur.

➢ Maintenant intervient *ALI*, il installa chez lui l'Agent-Client qui lui propose une interface similaire a celle de *figure 5.1* sauf que au lieu du champ **information sur le service**, il y aura un champ **information sur la requête**, ensuite l'agent-Client ce chargera de la génération de la description correspondante de la requête dans le format OWL-S dans le fichier *AchatTelephone.owl* (figure 5.5).

```
-----------------------------------------

<owl:Ontology about="">
  <owl:imports rdf:resource= "&service;" />
  <owl:imports rdf:resource= "&profile;" />
  <owl:imports rdf:resource= "&process;" />
  <owl:imports rdf:resource= "&actor;" />
  <owl:imports rdf:resource= "&addParam;" />
  <owl:imports rdf:resource= "http://www.ontologiededomaine.com/telephone.owl"/>
</owl:Ontology>

<profile:Profile>
   <profile:serviceName>AchatTelephone </profile:serviceName>
   <profile:textDescription> Ceci est une requête d'achat de téléphones</profile:textDescription>

<profile:contactInformation>
      <actor:Actor rdf:ID="ALI">
        <actor:name>ALI</actor:name>
        <actor:title>Informaticien</actor:title>
        <actor:phone>0778565846</actor:phone>
        <actor:fax>ALI@gmail.com </actor:fax>
        <actor:email>033-73-66-66</actor:email>
        <actor:physicalAddress>Dali Brahim ALGER-Algérie</actor:physicalAddress>
        <actor:webURL>http://www.ALI.com</actor:webURL>
      </actor:Actor>

</profile:contactInformation>
 <profile:hasInput>
      <process:Input rdf:ID="prix-input">
        <process:parameterType>http://www.ontologiededomaine.com/telephone.owl#prix
        </process:parameter Type>
      </process:Input>
 </profile:hasInput>
   <profile:hasOutput>
     <process:UnConditionalOutput rdf:ID="nokia-output">
         <process:parameterType>http://www.ontologiededomaine.com/telephone.owl#nokia
         </process:parameterType>
      </process:UnConditionalOutput>
   </profile:hasOutput>
</profile:Profile>
```

Figure 5.5 – La description OWL-S de la requête de client

La requête montre que le client recherche un service AchatTelephone qui doit avoir comme inputs une instance de concept *Prix* et générer comme outputs des instances de concept *Nokia* c'est-à-dire que ALI recherche un vendeur de téléphone portable qui vend exactement les produit Nokia (instance de concept Nokia). Après la génération de fichier *AchatTelephone.owl* l'Agent-Client le renvoie automatiquement vers l'Agent-Annuaire pour procéder à la recherche de service souhaité.

➢ L'Agent-Annuaire et après avoir su qu'il s'agit d'une requête de recherche il la redirige vers l'Agent-Découverte, ce dernier va prélever la première description OWL-S présente dans le registre **RAS** c'est-à-dire le fichier *ServiceVenteTelephone.owl* ainsi que la description OWL-S de la requête envoyer par le client c'est-à-dire le fichier *AchatTelephone.owl*. La correspondance entre chaque publication du service et la requête demande la correspondance entre leurs inputs et outputs, respectivement. Pour faciliter le travail les inputs de tous les services publiés et la requête sont égaux pour avoir une correspondance exacte (c'est-à-dire ils ont tous comme input le concept *Prix*). Ensuite l'Agent-Découverte va exécuter l'algorithme de correspondance entre les outputs de service et la requête (Algorithme4).

Algorithm 4 CorrespondanceDesOutput(*outputR*, *outputsS*)

for all out_S **in** *outputsS* **do**

$DegreCorrespondance \longleftarrow Max(\text{DegreDeCorrespondance}(out_R, out_S))$

end for

if $DegreCorrespondance \neq echec$ **then**

Enregistrer(S, ID)

end if

L'algorithme de correspondance fait appel à la fonction *DegréDeCorrespondance()* présenter dans ***Algorithme5*** pour savoir qu'elle est le degré de correspondance entre la première publication et la requête de client.

Algorithm 5 DegréDeCorrespondance (out_R,out_S)

if out_S=out_R **then**
  return exact
else if out_R sous-classe-de out_S **then**
  return exact
else if out_S super-classe-de out_R **then**
  return inclusif
else if out_R super-classe-de out_S **then**
  return partiel
else
  return echec
end if

Pour cela l'Agent-Découverte renvoie l'output de premier service c'est-à-dire le concept *portable* ainsi que l'output de la requête qui est *nokia* ainsi que l'URL de l'ontologie de domaine *Telephone.owl* présenté dans la *figure 5.6* vers le *RACER* qui reconnaît une correspondance **EXACT** entre les concepts Portable et nokia.

```
    ----------------------------
<owl:Ontology rdf:about=""/>
    ----------------------------
<owl:Class rdf:ID="Telephone"/>
<owl:Class rdf:ID="Option"/>
<owl:Class rdf:ID="Prix"/>
<owl:Class rdf:ID="Portable">
    <rdfs:subClassOf rdf:resource="#Telephone"/>
</owl:Class>
<owl:Class rdf:ID="Fixe">
    <rdfs:subClassOf rdf:resource="#Telephone"/>
</owl:Class>
<owl:Class rdf:ID="Audio">
    <rdfs:subClassOf rdf:resource="#Option"/>
</owl:Class>
<owl:Class rdf:ID="Video">
    <rdfs:subClassOf rdf:resource="#Option"/>
</owl:Class>
<owl:Class rdf:ID="Sumsung">
    <rdfs:subClassOf rdf:resource="#Portable"/>
</owl:Class>
<owl:Class rdf:ID="Nokia">
    <rdfs:subClassOf rdf:resource="#Portable"/>
</owl:Class>
<owl:Class rdf:ID="SonnyEricsson">
    <rdfs:subClassOf rdf:resource="#Portable"/>
</owl:Class>
<owl:Class rdf:ID="SerieW">
    <rdfs:subClassOf rdf:resource="#Samsung"/>
</owl:Class>
<owl:Class rdf:ID="SerieK">
    <rdfs:subClassOf rdf:resource="#Samsung"/>
</owl:Class>
<owl:Class rdf:ID="SerieN">
    <rdfs:subClassOf rdf:resource="#Nokia"/>
</owl:Class>
<owl:Class rdf:ID="SerieE">
    <rdfs:subClassOf rdf:resource="#Nokia"/>
</owl:Class>
<SerieN rdf:ID="N96"/>
<SerieE rdf:ID="E61i"/>
<SerieE rdf:ID="E73"/>
<SerieE rdf:ID="E55"/>
<SerieN rdf:ID="N93i"/>
<SerieN rdf:ID="N90"/>
    ----------------------------------------
<!--Created with Protege (with OWL Plugin 3.2.1, Build 365)  http://protege.stanford.edu  -->
```

Figure 5.6 – La description OWL de l'ontologie Telephone.owl

Après avoir reçue le dégré de correspondance entre la requête et le premier service qui est déférente d'échec, l'Agent-découverte enregistre cette première correspondance dans le registre de correspondance **RC**.

➢ Ensuite l'Agent-Découverte va prélever le deuxième service *ServiceTelephonePortable.owl*, comme dans l'étape précédente, il envoie au *RACER* le paramètre output de la publication avec le paramètre output de la requête c'est-à-dire *serieN* et *nokia* respectivement ansi que l'URL de l'ontologie de domaine. Comme illustrer dans la *figure 5.8* la classe *serieN* est une sous-classe de *nokia*. Ceci veut dire que le *RACER* va reconnaitre une correspondance **Partiel**, qui va être enregistré dans le registre RC.

➢ La même chose pour le troisième service, l'Agent-Découverte va prélever les concepts d'outputs *Fixe* et *nokia* ainsi que l'URL de l'ontologie de domaine et les envoyer au RACER qui reconnait un **échec** de correspondance car comme montré dans la *figure 5.8* il n'existe aucune relation de **subsomption** entre les deux classes *Fixe* et *nokia*. Le tableau suivant représente les correspondances trouvées par l'Agent-Découverte entre la requête du Client et les trois services publiés.

Couples de concepts	**Output_Requete**	**Output_Publication**	**Correspondance**
(Req, Ser1)	Nokia	Portable	Exact
(Req, Ser2)	Nokia	SerieN	Partielle
(Req, Ser3)	Nokia	Fixe	Echec

Figure 5.7 – Les correspondances trouvées par l'Agent-Découverte

➢ Une fois que tous les services publiés ont été analysés, l'Agent-découverte exécute la fonction de trie de correspondance présenter dans l'**Algorithme6**. Cette fonction réalise un trie entre toutes les correspondances enregistrer dans le registre RC, dans notre cas on a deux correspondance : la première avec le premier service qui possède une correspondance **EXACT** et l'autre avec le deuxième service qui possède une correspondance **PARTIEL**. Donc selon l'algorithme de trie le premier service sera élue. C'est-à-dire que le premier service sera choisi ***ServiceVenteTelephone***.

Algorithm 6 TrieCorrespondance(*corresp*1, *corresp*2)

if *corresp*1.*output* > *corresp*2.*output* **then**

corresp1 > corresp2

else if *corresp*1.*output* = *corresp*2.*output* **and** *corresp*1.*input* > *corresp*2.*input* **then**

corresp1 > corresp2

else if *corresp*1.*output* = *corresp*2.*output* **and** *corresp*1.*input* = *corresp*2.*input* **then**

corresp1 = corresp2

end if

➢ Une fois le service le plus adéquat a la demande de client est trouvé l'Agent-Découverte envoie le résultat vers l'Agent-Annuaire qui ce chargera de la livraison de résultats vers l'Agent-Client.

La figure suivante représente l'ontologie globale OWL (*Telephone.owl*) utilisée dans notre exemple.

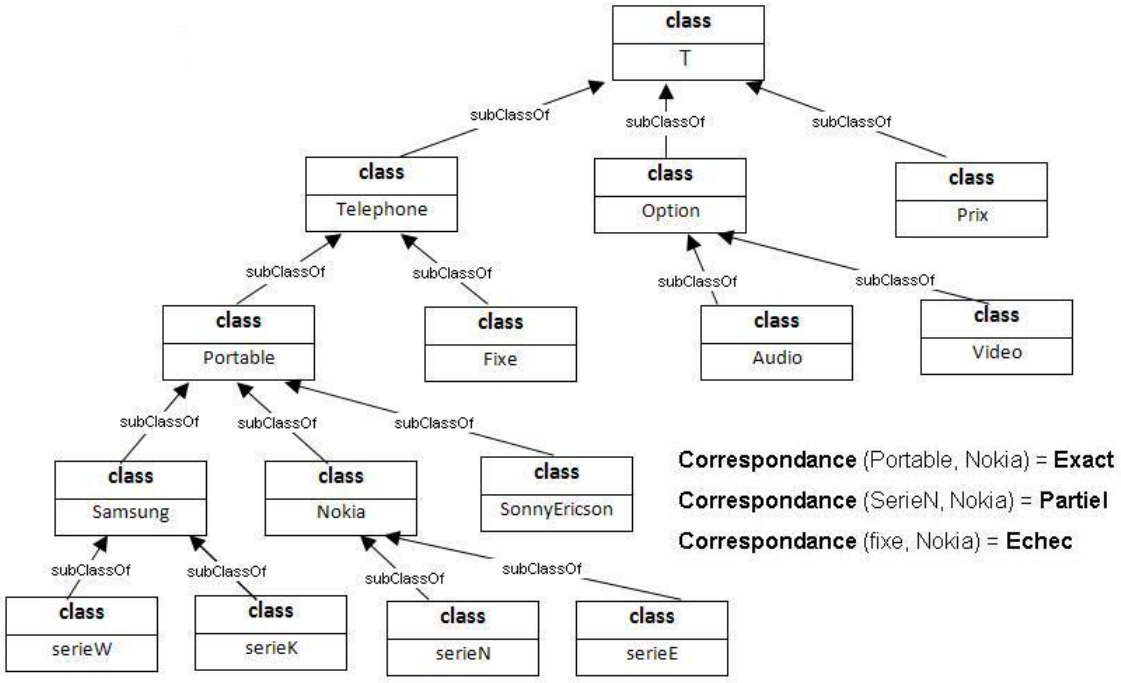

Figure 5.8 – Le graphe d'ontologie OWL de l'exemple telephone.owl

5.5 Outils de programmation

Pour implémenter un système tel que celui que nous avons proposé on a besoin d'un langage tel *JAVA*.

5.5.1 Pourquoi JAVA ?

Né en 1995, Java est un langage orienté objets permettant d'écrire de façon simple et claire des programmes portables sur la majorité des plates-formes. De plus, il bénéficie d'une importante bibliothèque de " classes " avec lesquelles l'utilisateur pourra composer des interfaces graphiques, animer une page HTML par une " applet ", créer des applications " multi-threads " ou encore communiquer en réseau. Le Langage Java, concepts et pratique s'adresse à l'étudiant, l'ingénieur ou le programmeur qui, connaissant déjà la syntaxe du langage C, désire acquérir la maîtrise de Java. Sont traités en particulier les classes, l'héritage, les interfaces, les exceptions, les paquetages, les règles de visibilités, les interfaces graphiques, les flux de données, les applets, les threads et la communication en réseau [CHAR 06].
Dans notre cas nous avons besoin d'implémenter un système multi-agents. Une plateforme multi-agent 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 des SMA ainsi créé. Parmi les plateformes fournies comme logiciels libres, il y a quelques plateformes plus connues pour avoir été utilisées dans le développement de plusieurs applications : JADE, Zeus, MadKit, AgentBuilder, Jack, JAFMAS, AgentTool, DECAF, RMIT,...etc. Mais le plus connue c'est JADE (Java Agent DEvelopment framework).

5.5.2 La plateforme JADE

La plate-forme JADE est entièrement implémentée en JAVA, et répond aux spécifications FIPA (Foundation for Intelligent Physical Agents). JADE essaye d'optimiser les performances d'un système d'agent distribué. JADE est une plateforme multi-agent créé par le laboratoire TILAB. Il est entièrement implémenté en JAVA. Il permet le développement de systèmes multi-agents et d'applications conformes aux normes FIPA (Foundation for Intelli-

gent Physical Agents). Il fourni des classes qui implémentent " JESS " 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 plateforme.

5.5.3 Editeur d'ontologie

5.5.3.1 Editeur OWL

Il existe plusieurs éditeurs d'ontologie gratuit sur le web, on trouve : Protégé, SWOOP, KMgen, SemanticWorks . . .etc. mais reste toujours Protégé le plus connu et le plus utilisé des éditeurs d'ontologie. Open-source, il est 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.

Dans notre étude de cas nous avons utilisé *Protégé 3.2.1* comme présenter dans la *figure 5.9* pour la construction de l'ontologie globale de domaine et générer enfin la description OWL dans le fichier *Portable.owl* (figure 5.6).

5.5.3.2 Editeur OWL-S

Il existe plusieurs outils pour la création des descriptions OWL-S, il y a l'API OWL-S qui peut être intégrer a protégé. Il y a aussi OWLSeditor. Pour la création des descriptions des services web et requête nous avons utilisé l'OWLSAPI téléchargeable depuis [OWLSAPI] et intégrable sur la plateforme JAVA.

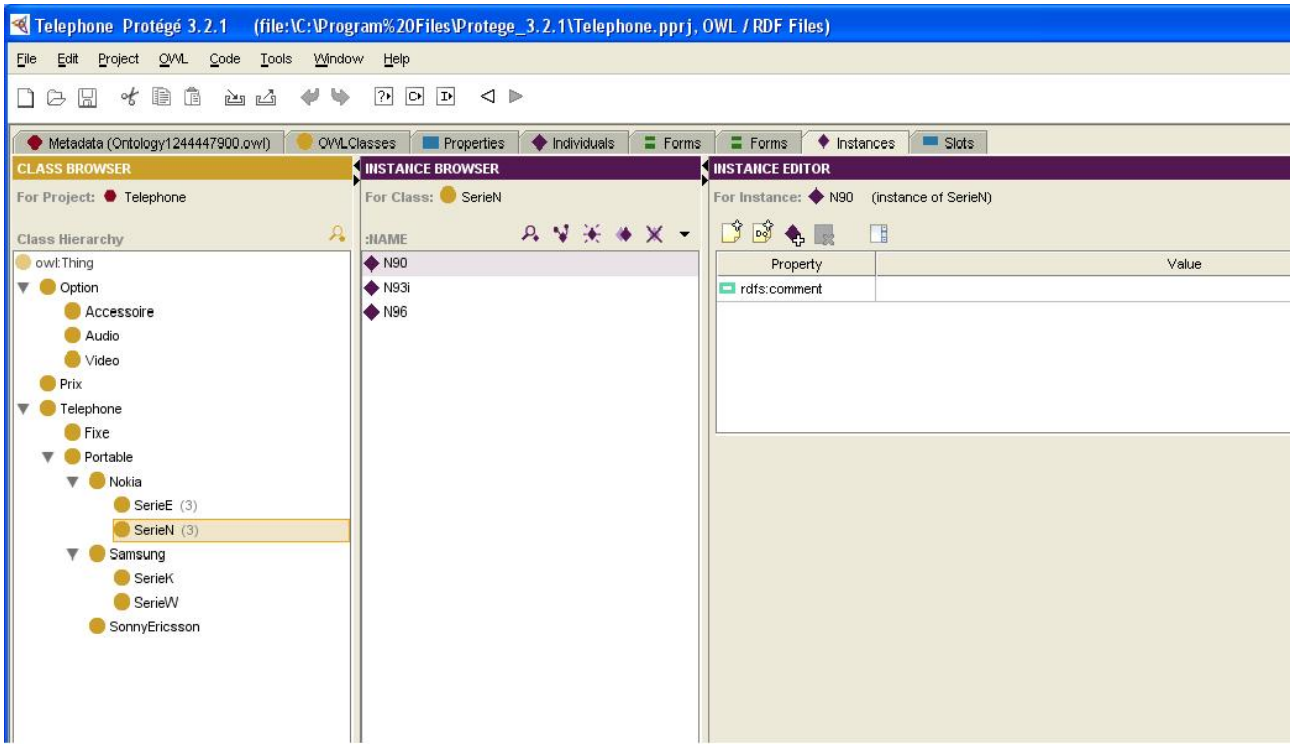

Figure 5.9 – Construction d'ontologie avec Protégé

5.5.3.3 Inférence sur l'ontologie

Pour l'inférence et les testes de subsomption entre les concepts sur l'ontologie de domaine Telephone.owl on a choisi l'outils RACER téléchargeable sur [RACE] et intégrable dans la plateforme JAVA.

5.6 conclusion

Dans ce chapitre nous avons appliqué notre modèle sur un cas fréquent qui est la vente de téléphone sur le Web. Nous avons focalisé sur le point essentiel de la problématique qui est la recherche des services web souhaité. Comme nous le sachant tous que actuellement si une personne se lance dans la recherche d'un service web sur internet, cette recherche s'effectue vraiment manuellement avec l'intervention complète de client par contre avec notre solution proposée qui est une nouvelle vue de la problématique ou on a intégré des agents et les concepts découverte sémantique a base d'ontologies afin d'automatiser cette tache. Aussi pour avoir une précision dans la recherche c'est-à-dire rechercher que les services les plus

adéquats à la demande de client. Dans notre exemple étudié nous avons montré un cas de requête et services différents mais qui donne des correspondances *Exact* ou *Partiel* en utilisant des informations ontologiques.

Conclusion générale

Conclusion

L'utilisation des technologies du Web sémantique et système multi-agents constitue une voie prometteuse permettant de mieux exploiter les services web en automatisant, autant que possible, les différentes tâches liées au cycle de vie d'un service web (publication, recherche, sélection et composition).

Dans ce mémoire, nous avons présenté les différentes technologies nécessaires pour l'adaptation d'un modèle pour la découverte sémantique des services web à base d'agents, on a présenté les ontologies et le web sémantique ainsi que les systèmes multi-agents.

Nous avons vue que la découverte actuelle des services web est purement syntaxique et ne permet pas d'exploiter toutes les capacités (dans la publication) des services dans l'UDDI. C'est de ces deux problèmes que nous avons inspiré notre nouveau modèle.

Nous avons proposé un modèle basé sur les systèmes multi-agents où chaque agent a un rôle bien précis, mais qui interagissent entre eux pour aboutir à une découverte sémantique qui exploite toutes les capacités (input, output, precondition, effect) dans une description d'un service web (publication) et d'automatiser la tache de la découverte de ces services.

La recherche des services web sémantique est réalisée en suivant une approche basée sur les niveaux de subsomption entre les paramètres du service et la requête par rapport a une ontologie globale de concepts.

Afin de démontrer notre travail, nous avons fait une étude de cas où nous avons appliqué notre modèle sur un exemple de vente des téléphones mobiles sur le Web.

Perspectives

Nous proposant certaines perspectives à savoir :

- Ajouter un nouvel agent nommé Agent-Sélection pour la sélection des services web, le rôle de cet agent est de choisir le meilleur fournisseur d'un service web, étant donné un ensemble de fournisseurs de services qui peuvent avoir le même degré de correspondance avec la requête de l'utilisateur, il faut savoir gérer cette problématique. Il y a une proposition de base sur la Qualité de Service (i.e. QdS en anglais Quality of Service QoS) qu'il faut exploiter pour la rendre comme principale rôle de l'Agent-Sélection.
- Proposer un prototype complet plus complexe basé sur les agents pour aboutir à une découverte optimale des services web en se basant sur toutes les partie fonctionnelles et non-fonctionnelles dans un service profile, et pour quoi pas intégrer le ServiceGrounding dans la tache découverte des services web afin d'avoir une mobilité d'invocation des services web.
- Il faux penser a l'extensibilité de notre modèle multi-agent proposé et le rendre ouvert à d'autres types de système (SMA ou modèle de service web), c'est-à-dire rendre possible la réplication des donnée entre plusieurs Agent-Annuaires, Il faut aussi penser à l'adaptation du modèle que nous avons proposé à la plateforme actuelle des services web, en rendant possible la communication entre systèmes hétérogènes.

Bibliographie

[ACL 02] FIPA ACL Message Structure Specification, 03/12.2002. http://www.fipa.org/specs/fipa00061/SC00061G.html

[ALEX 02] Alex Mustiere, *introduction aux services Web*, 06 juin 2002.

[ALLI 03] JM Alliot, *Qu'est-ce que le Middleware*, 13 mars 2003.

[BENA 02] Boualem Benatallah, Mohand-Said Hacid, Alain Leger, Christophe Rey, Farouk Toumani. *On Automating Web Services Discovery*. 2002.

[BOOT 04] D. Booth, H. Haas, F. McCabe, E. Newcomer, M. Champion, C. Ferris, and D. Orchard. *Web Services Architecture*. Technical report, W3C Working Group Note 11, 2004. http://www.w3.org/TR/ws-arch/#id2260073

[BOUD 07] A. Boudina, L. Tayeb, *L'intelligence Artificielle Distribuée et les Systèmes Multi-Agents*, février 2007. htttp://opera.inrialpes.fr/people/Tayeb.Lemlouma/Papers/IAD_Presentation.pdf

[CABR 04] L. F. Cabrera, C. Kurt, D. Box, *Introduction à l'architecture de services Web et ses spécifications*. WS. Version 2.0. Octobre 2004.

[CHAR 06] Irène Charon. Le langage Java : concepts et pratique -le JDK 5.0. janvier 2006.

[CORC 03] O. Corcho, M. Fernandez Lopez, A. Gomez-Pérez. *Methodologies, tools and language for building ontologies*. Data and knowledge Engineering, 46 :41-64. 2003.

[ENCY 00] Encyclopaedia Universalis. Dictionnaire de la philosophie. Paris : A. Michel, Encyclopaedia Universalisc 2000.

[ERIC 02] Eric Newcomer, *Understanding Web Services XML, WSDL, SOAP and UDDI*, 2002.

[FERB 95] J. Ferber, *Les Systèmes Multi-Agents*. Vers une intelligence collective. Interéditions, 1995.

[FERN 99] M. Fernandez Lopez. *Overview of methodologies for building ontologies*. IJCAI-99 Workshop on Ontologies and problem-solving Methods : Lessons learned and future Trends. 1999.

[FIPA 00] FIPA Abstract Architecture Specification. Foundation for Intelligent Physical Agents, 2000. http://www.fipa.org/specs/fipa00001/

[JARR 02] I. Jarras et B. Chaib-draa, *Aperçu sur les systèmes multiagents*, Série scientifique, Centre interuniversitaire de recherche en analyse des organisations (CIRNO), Université de Montréal, 2002.

[JEFF 06] Jeffrey Hasan with Mauricio Duran, *Expert Service Oriented Architecture in C# 2005*, Second Edition, 2006.

[GUDG 00] M. Gudgin, M. Hadley, N. Mendelsohn, J. Moreau, and H. Nielsen. *Simple object access protocol (soap)*. Technical report, World Wide Web Consortium, may 2000. http://www.w3.org/TR/SOAP/

[GONZ 01] Javier Gonzalez-Castillo, David Trastour, Claudio Bartolini. *Description Logics for Matchmaking of Services*. Trusted E-Services Laboratory, HP Laboratories Bristol, HPL-2001-265, October 30th, 2001.

[GARD 02] G. Gardarin, *XML : Des bases de données aux services Web*, Paris 2002.

[GRUB 93] T. Gruber. *A translation approach to portable ontology specifications*. Knowledge acquisition, 5(2), 199-220. 1993.

[HEAT 01] K. Heather. *Web services conceptual architecture (wsca 1.0)*, may 2001. http://www-306.ibm.com/software/solutions/webservices/pdf/WSCA.pdf

[JACE 07] KOPECKY Jacek, VITVAR Tomas, BOURNEZ Carine, FARRELL Joel (4). *SAWSDL : Semantic annotations for WSDL and XML schema*. 2007.

[KAZA 96] O. Kazar, *Conception et réalisation d'un modèle de réseau sémantique*, Mémoire de magister, pp.6-45, Université de Constantine, 1996.

[KQML 92] T. Finin, R. Fritzson, and D. McKay. An overview of KQML : *A Knowledge Query and Manipulation Language*. Technical report, University of Maryland Baltimore Country, 1992.

[LEE 01] Berners-Lee, J. Hendler, and O. Lassila. *The semantic web*. Scientific American 284(5),34-43. 2001.

[LYEL 03] M. Lyell, L. Rosen, M. Casagni-Simkins and D. Norris. *On software agents and web services : Usage and design concepts and issues*. In Workshop on Web services And Agent-based ingineering, Melbourne, Australia, 2003.

[MELL 04] Tarek MELLITI, *Interopérabilité des services Web complexes*. Application aux systèmes multi-agents, Doctorat de l'Université Paris IX Dauphine le 8 décembre 2004.

[OWLS 04] OWLS, *Semantic Markup for Web Services*, 22 November 2004. http://www.w3.org/Submission/OWL-S/

[OWLSAPI] http://projects.semwebcentral.org/frs/?group_id=32

[PELL 04] Pellet : *The Open Source OWL DL Reasoner*, 2004. http://clarkparsia.com/pellet

[PAOL 02] M. Paolucci, T. Kawamura, T.R. Payne and K. Sycara, *Semantic Matching of Web Services Capabilities*. In : International Semantic Web Conference (ISWC), 9 - 12 June, Sardinia, Italy. 2002.

[PAOL 04] M. Paolucci, D. Martin, S. McIlraith, M. Burnstein, D. McDermott, D. McGuinness,B. Parsia, T.R. Payne, M. Sabou, M. Solanki,N. Srinivasan and K. Sycara, *Bringing Semantics to Web Services : The OWL-S Approach*. In : First International Workshop on Semantic Web Services and Web Process Composition (SWSWPC 2004), San Diego, CA.

[PSYC 04] V. psyche, O. Mendes, J. bourdeau. *Apport de l'ingénierie ontologique aux environnements de formation a distance*. Revue SRTICEF, Volume 10, 2003, ISSN : 1764-7223, mis el ligne le 05/02/2004.

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

[RACE] http://www.daml.ri.cmu.edu/matchmaker/download/racer-1-7-12-windows.zip

[RAHE] Rahee Ghurbhurn, *Introduction aux Web Services*, Master Web Intelligence.

[RDF 99] RDF, *Resource Description Framework*, 1999. http://www.w3.org/TR/REC-rdf-syntax/

[RDFS 99] RDFS, *Resource Description Framework Schéma*, 1999. http://www.w3.org/TR/rdf-schema/

[SYCA 02] M. Paolucci, T. Kawamura, T.R. Payne and K. Sycara, *Importing the Semantic Web in UDDI*. In : Web Services, E-Business and Semantic Web Workshop, CAiSE 2002, May 2002. Toronto, Canada.

[SYCA 03] Katia Sycara, Massimo Paolucci, Anupriya Ankolekar, Naveen Srinivasan, Automated discovery, *interaction and composition of Semantic Web services*. Web Semantics : Science, Services and Agents on the World Wide Web 1 (2003) 27-46 Carnegie Mellon University, Pittsburgh, Pennsylvania, USA.

[SAWSDL 07] SAWSDL : *Semantic Annotations for WSDL Working Group*, 2007. http://www.w3.org/2002/ws/sawsdl/

[SGML 95] SGML, *Standard Generalized Markup Language*, 1995. http://www.w3.org/MarkUp/SGML/

[SYCA 98] K. Sycara. *Multiagent Systems*. AI Magazine 19(2), 1998.

[SYCA 04] Naveen Srinivasan, Massimo Paolucci , Katia Sycara. *Adding OWL-S to UDDI, implementation and throughput*, 2004.

[TRAS 01] David Trastour, Claudio Bartolini, Javier Gonzalez-Castillo. *A Semantic Web Approach to Service Description for Matchmaking of Services*. Trusted E-Services Laboratory, July 30th , 2001.

[UDDI 00] Universal Description, Discovery and Integration (uddi). The UDDI Technical White Paper, 2000. http://www.uddi.org

[VERM 03] Kaarthik Sivashanmugam, Kunal Verma, Amit Sheth and John Miller, *Adding Semantics to Web Services Standards*, Proceedings of the first IEEE International Conference on Web Services (ICWS'03), Las Vegas, Nevada (June 2003) pp. 395-401.

[WSMO 05] WSMO, *Web Service Modeling Ontology*, 2005. http://www.w3.org/Submission/WSMO/

[WIKAG] http://fr.wikipedia.org/wiki/Agent_(informatique)

[XML] XML, *eXtended Markup Language*. http://www.w3.org/XML/

[YASM 07] Yasmine CHARIF, *Chorégraphie dynamique de services basée sur la coordination d'agents introspectifs*. Thèse de doctorat. UNIVERSITE PIERRE ET MARIE CURIE, 2007.

[YANI 03] Yannick Prié. *Annotations et métadonnées pour le Web sémantique*. LIRIS - Université Lyon 1. Journée Web Sémantique et SHS. Action spécifique WS - 7 mai 2003.