DEMOCRATIQUE ET POPULAIRE
MINISTERE DE L'ENSEINEMENT SUPERIEUR ET DE LA RECHERCHE SCIENTIFIQUE

UNIVERSITE FERHAT ABBAS DE SETIF
UFAS, ALGERIE

MEMOIRE

Présentée à la faculté des sciences
Département D'informatique

Pour l'obtention du diplôme de

MAGISTER

Option : Ingénierie Des Systèmes Informatique

Présentée par : CHAIB AOUATEF

Composition automatique et semi- automatique des services Web sémantique Par l'utilisation des techniques de planification distribuées «Les Systèmes Multi Agents »

Soutenue le : 18/06/2011 **Devant la commission d'examen :**

Mr. M. TOUAHRIA	**Maitre de conférences U.F.A SETIF**	**Président**
Mr. A.KHABABA	**Maitre de conférences U.F.A SETIF**	**Rapporteur**
Mr. A.MOUSSAOU	**Maitre de conférences U.F.A SETIF**	**Examinateur**

REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE
MINISTERE DE L'ENSEINEMENT SUPERIEUR ET DE LA RECHERCHE SCIENTIFIQUE

UNIVERSITE FERHAT ABBAS DE SETIF
UFAS, ALGERIE

MEMOIRE

Présentée à la faculté des sciences
Département D'informatique

Pour l'obtention du diplôme de

MAGISTER

Option : Ingénierie Des Systèmes Informatique

Présentée par : CHAIB AOUATEF

Composition automatique et semi- automatique des services Web sémantique Par l'utilisation des techniques de planification distribuées «Les Systèmes Multi Agents »

Soutenue le : 18/06/2011 **Devant la commission d'examen :**

Mr. M. TOUAHRIA	**Maitre de conférences U.F.A SETIF**	**Président**
Mr. A.KHABABA	**Maitre de conférences U.F.A SETIF**	**Rapporteur**
Mr. A.MOUSSAOUI	**Maitre de conférences U.F.A SETIF**	**Examinateur**

Remerciements

Tout d'abord, je tiens à remercier le bon Dieu pour m'avoir illuminée et menée jusqu'ici et qui ma donné la santé, la volonté et le courage pour accomplir ce travail.

Mes sincères remerciements à mon encadreur Mr **KHABABA**, je lui suis particulièrement reconnaissante pour sa disponibilité, sa compétence, son soutien, ses conseils judicieux et la confiance dont il m'a fait part lors de la réalisation de ce travail.

Je remercie aussi Mme **Boussebough** pour son aide très précieux qu'elle m'a apporté tout au long de ma préparation de ce mémoire.

Je souhaite adresser mes remerciements au membre de jury qui ont accepté la tâche délicate de rapporter ce mémoire et qui ont eu la patience de juger ce travail.

Je remercie également tous les enseignants qui ont contribué à ma formation durant toutes ces années d'études.

A mes amis et mes collèges de l'´ecole doctorale promotion 2008/2009.

Mes profonds et mes plus grands remerciements vont aux membres de ma famille pour leur patience pour me soutenir, m'encourager, me faire confiance et vouloir toujours que je vole plus haut. Je tiens à remercier ma mère, la personne qui m'a donner le courage et le soutien pour faire ce travail, mon père, l'homme qui ma soutenu tout au long de mes études et sans lui je n'aurais jamais atteint ce niveau d'étude, mon mari, pour son apport moral et son encouragement ainsi que ma sœur RAYENNE et mon frère MANAR EL ISLAM.

A tous ce qui m'ont aidé de prêt ou de loin à la réalisation de ce modeste travail.

AOUATEF ;

Table des Matières

Deuxième partie - Etat de l'art des services Web et de la composition de services Web

Chapitre 3 - Les services Web

Troisième partie – Un système multi agent pour la composition des services Web (MAS4WSC)

Chapitre 4 – La conception du système

Chapitre 5 – L'implémentation du système

Liste des figures

Liste des tableaux

Introduction Générale

L'évolution des réseaux informatiques combinée à celle des systèmes informatiques a permis la mise en œuvre de systèmes distribués de plus en plus performants et le développement d'applications de toute sorte s'appuyant sur ces infrastructures. Les systèmes informatiques distribués recouvrent un champ très vaste dont font partie les grilles de calcul, le stockage d'informations dans des réseaux de pairs, le déploiement d'applications web, etc. L'évolution de plateformes à composants distribués (ex. les EJB de Sun ou encore .NET de Microsoft) a donné naissance à un nouveau mode de développement logiciel appelé SOC.

SOC, acronyme de Service-Oriented Computing, est le paradigme de computing qui utilise des services comme étant les éléments fondamentaux pour la réalisation des applications, cette collection des services communiquent entre eux via des protocoles et technologies web standardisés (XML[1], HTTP[2], FTP[3], SMTP[4]…). L'utilisation de ces technologies et protocoles standards à pour but d'atteindre un objectif majeur qui est la réalisation d'applications distribuées en réutilisant des services ou blocks existants dans des architectures et plateformes qui sont souvent hétérogènes. Un des plus importants secteurs qui bénéficie des applications SOC est E-biseness.

Dans ce contexte, les applications sont construites à partir de services individuels exposant leurs fonctionnalités à travers des registres et s'abstrayant complètement de leur implémentation sous-jacente. Leurs interfaces publiées peuvent alors être recherchées et invoquées par des utilisateurs ou d'autres services. L'avantage de SOC est qu'il permet de créer des systèmes d'information en agrégeant des services individuels. En effet, il permet de créer des systèmes d'information inter- et intra-entreprises de manière relativement aisée en « serviçant »les systèmes existants, d'accomplir interopérabilité et transparence entre ces systèmes au fil de l'eau, de sélectionner dynamiquement des services sur la base de critères de qualité de services personnalisés par chacun et d'utiliser efficacement les ressources de grilles.

Les besoins de SOC pour la satisfaction des précédents cas d'usage peuvent être plus facilement satisfaits à travers une architecture répondant aux

---

[1] Extensible Markup Language. http://www.w3.org/XML/1999/XML-in-10-points.fr.html
[2] HyperText Transfer Protocol. http://www.w3.org/Protocols/
[3] File Transfer Protocol
[4] Simple Mail Transfer Protocol

propriétés requises. Une telle architecture est appelée SOA ou architecture orientée services.

La notion de SOA[5] définit un modèle d'interactions au niveau logiciel mettant en œuvre des connexions entre des composants logiciels «fournisseurs» à l'attention de composants logiciels «consommateurs». Cette approche a largement été adoptée pour le développement de systèmes de traitement de l'information favorisant par là même, la distribution de fonctionnalités indépendantes, leur réutilisabilité et leur intégration.

Afin de mettre en œuvre une SOA, il est essentiel d'avoir une compréhension claire de ce qu'est un service web.

À l'origine, la technologie des services web a été initiée par IBM et Microsoft, puis en partie normalisée sous l'égide du W3C3, l'organisme chargé de standardiser les évolutions du web. Elle est maintenant acceptée par l'ensemble des acteurs de l'industrie informatique, faisant des services web une technologie révolutionnaire.

Les web services sont une généralisation des RPC[6] indépendamment d'une plate-forme et d'un modèle de programmation. Ils permettent par le biais d'Internet d'appeler des procédures distantes afin d'agir sur le serveur et/ou pour en recevoir des informations.

Les Web Services sont des applications modulaires basées sur Internet qui exécutent des tâches spécifiques et qui respectent un format spécifique ou encore sont une interface qui décrit une collection d'opérations accessibles au travers d'un réseau avec des messages standardisés basé sur XML. Ils ont pour vocation de favoriser une architecture orientée services, intégrant des systèmes hétérogènes complexes, fortement distribués et pouvant coopérer sans recourir à une intégration spécifique et coûteuse. L'avantage majeur des services web, par rapport aux autres approches de systèmes distribués tel que RMI[7], réside dans leur support des pare-feux, mais surtout dans leur articulation autour d'XML (standard promulgué par le W3C), ce qui leur procure l'avantage d'être non propriétaire et ainsi multiplateforme.

L'architecture des services web repose sur les principes et standards suivants :

[5] Architecture Orientée Services
[6] Remote Procedure Call
[7] Remote Method Invocation

- Un service web est identifié par une chaîne de caractère appelée URI (Unified Resource Identifier), dont la syntaxe respecte une norme d'Internet mise en place par le W3C.
- L'interface publique du service et les liaisons sont définies et décrites en XML, généralement en WSDL (Web Service Description Language). Un document WSDL est une description basée sur XML indiquant le protocole de communication et le format de messages requis pour communiquer avec un service.
- La définition du service peut alors être découverte par d'autres systèmes logiciels au travers de registres tels qu'UDDI (Universal Description Discovery and Integration). Ce modèle, acronyme de, est une technologie d'annuaire basée sur XML permettant de localiser sur le réseau un service web recherché.
- Les applications et systèmes peuvent alors interagir avec le service web d'une manière prescrite par les protocoles Internet, usuellement SOAP (Simple Object Access Protocol) qui est un protocole permettant l'échange d'informations structurées dans un environnement décentralisé et distribué. Il a été conçu indépendamment de tout modèle de programmation et autre sémantique spécifique d'implémentation.

Des recherches faites sur les services Web par rapport à SOA couvrent plusieurs problèmes intéressants : la Sélection des services Web, la Découverte et Découverte dynamique des services Web, la Composition des services Web… etc. Notre étude est basée sur la notion de composition des services web.

La composition des services web est l'un des défis des SOA, elle permet de créer de nouveaux services personnalisés, plus riches et plus intéressants aussi bien pour des applications, d'autres services ou plus communément pour des utilisateurs humains. En effet, si une application ou un client requièrent des fonctionnalités, et qu'aucun service n'est seul apte à les fournir, il devrait être possible de combiner ou de composer des services existants afin de répondre aux besoins de cette application ou de ce client.

La composition des services web a pour objectif de déterminer une combinaison de services en fonction d'une requête d'un client. Du coté du client, cette composition semblera un unique service. La composition sera transparente au client, même si cette composition sera la combinaison de plusieurs services web.

L'approche actuelle de la composition de services Web consiste à définir des processus métier, i.e., des enchaînements réutilisables de services. Cette approche est la plus utilisée par le monde industriel. Une autre approche envisageable est la composition « dynamique ». Les services Web sont composés

dynamiquement en fonction de leurs fonctionnalités, de leur disponibilité et des aléas pouvant survenir au moment du processus de composition.

Nous nous positionnons, dans cette thèse, sur l'étudie des mécanismes nécessaires à la mise en œuvre de la composition dynamique de services Web et plus particulièrement de proposer un modèle de composition s'appuyant sur la planification multi-agent. En effet, les services web partagent avec les systèmes multi agents plusieurs propriétés dont les méthodologies agents tiennent explicitement compte dans leur ébauche de solutions. De telles propriétés sont: **[19]**

- La distribution géographique et/ou logique d'entités, données ou informations hétérogènes ;
- La complexité du problème global à résoudre, celui-ci n'étant manipulable que par des stratégies heuristiques utilisant des données ou connaissances locales ;
- La flexibilité des interactions il n'y a pas d'affectation a priori des tâches et les mécanismes de résolution de problèmes ne sont pas préalablement assignés à telle ou telle autre entité ;
- L'environnement dynamique nécessitant, pour la résolution de problèmes, des entités réactives et adaptatives ;
- L'ouverture il n'est pas possible de donner une spécification complète du problème à résoudre ni de définir une fonction d'utilité globale.

Les systèmes multi agent ne sont pas qu'un simple groupe d'agents dans un environnement commun, mais une réelle organisation avec des règles sociales et des interactions permettant la coopération et la collaboration pour la résolution de problèmes que les systèmes centralisés ne pourraient pas résoudre.

Alors les systèmes multi agents semblent être particulièrement bien adaptés à la modélisation des problématiques (de distribution, d'ouverture, de flexibilité et de collaboration des services) sous-tendues par la composition dynamique de services. C'est la raison pour laquelle l'objet de notre étude va porter sur l'étude de solutions SMA pour la modélisation des problématiques de composition des services.

Nous proposons un système multi agents (SMA) centré utilisateur dédié à la composition des services web sémantique, les agents permettent d'utiliser les propriétés sémantiques des services en vu de les mettre en correspondance avec les concepts de la requête de l'utilisateur et sont chargés de coordonner leurs compétences afin de réaliser la tâche soumise. Le système proposé s'appuie sur le protocole des réseaux contractuels. Dans ce protocole, les agents peuvent prendre

deux rôles: *gestionnaire* ou *contractant.* L'agent qui doit exécuter une tâche donnée (le gestionnaire) commence tout d'abord par décomposer cette tâche en plusieurs sous-tâches. Il doit ensuite annoncer les différentes sous-tâches au reste des agents de l'environnement. Les agents qui reçoivent une annonce de tâche à accomplir peuvent ensuite faire une proposition devant refléter leur capacité à remplir cette tâche. Le gestionnaire rassemble ensuite toutes les propositions qu'il a reçues et alloue la tâche à l'agent ayant fait la meilleure proposition. **[7]**

Cette thèse est scindée en 3 parties majeures :

La Première partie est consacrée à un état de l'art relatif aux domaines abordés dans cette thèse. Nous présentons dans le **Chapitre 1** la définition et les concepts des services Web. Leur architecture et leurs standards. Nous détaillons aussi la notion des services web sémantique. Finalement nous donnons les axes de recherche sur les services Web à savoir la sélection, la découverte, la découverte dynamique et la composition de services Web.

Dans le **Chapitre 2**, nous présentons un état de l'art de la composition des services Web sémantiques. Nous définissons tout d'abord les différentes catégories de composition des services. Nous abordant en détail la composition automatique et dynamique des services web sémantique en donnant son cadre générale et expliquant ses différentes techniques

La Deuxième partie porte sur une étude complète des systèmes multi agents. Dans le **Chapitre 3** nous présentons en détaille le concept d'agent et le système multi agent. Ensuite on entame le noyau de notre chapitre qui est la planification distribuée dans un système multi agents. Ensuite en présente les protocoles d'interaction les plus utilisés en détaillons le protocole des réseaux contractuels. En fin nous terminerons notre chapitre par quelque environnement du développement des systèmes multi agents.

La Troisième partie est le cœur de notre travail. Elle est consacrée au système que nous avons développé pour la composition de services Web sémantiques. Nous présentons notre approche proposée : un système multi agent dédié à la composition automatique des services web, où pour chaque service est associé un agent. Les agents doivent coordonner leurs compétences afin de proposer un plan de composition. Le protocole de coordination utilisé dans notre système est le protocole des réseaux contractuel (contract net protocole) c-à-dire que les agents échangent leurs propositions sous forme d'appel d'offre. Les agents peuvent trouver plusieurs plan de composition qui peuvent satisfaire la requête utilisateur, alors le meilleur plan doit être choisi et retourner à l'utilisateur. Le plan optimal est caractérisé par une qualité de service élevée.

Dans le **Chapitre 4**, nous expliquons le principe de notre approche. Nous définissons tout d'abord plus précisément l'architecture de notre système, ainsi que son fonctionnement et l'intégration de chaque agent.

Le **Chapitre 5** présente la mise en œuvre de notre architecture dans un environnement de composition de SWS, en détaillant les algorithmes proposés dans le système développé. Pour cela, nous avons choisi la plate forme JADE pour la mise en œuvre de notre système car : JADE est un logiciel libre, elle offre plusieurs fonctionnalités et fournir une large gamme de bibliothèques, elle est aussi riche en documentation. En plus le protocole d'interaction utilisé dans notre système est le protocole des réseaux contractuels (ContractNet Protocol) a été normalisé par l'organisation FIPA et implémenté dans la plate-forme JADE. [45]

Ce chapitre est illustré par une étude de cas, afin de démontrer que le système proposé peut être appliqué à un domaine quelconque. Dans notre travail, nous avons appliqué ce système à une "Agence de voyage" pour effectuer les différentes réservations, en détaillant chaque étape de la composition.

Pour finir, une conclusion générale reprendra nos contributions apportées dans cette thèse, et en analysera ses limites. Nous proposerons enfin quelques perspectives de recherche envisagées.

Partie I

Etat de l'art des services Web et de la composition de services Web sémantiques

☞ *Chapitre 1*

Les services Web et les services Web sémantiques

1.1 Introduction

Les architectures à base de services SOA [8] ont introduit une nouvelle organisation des applications, basée sur le concept de services, composants logiciels auto-descriptifs, faiblement couplés et interagissant. L'une des implémentations des SOA, les services Web (Web Services) sont devenus incontournables pour la réalisation d'applications distribuées sur Internet

Les services Web prennent leurs racines dans l'informatique distribuée et dans l'avènement du Web. La technologie des services Web a pour objectif d'uniformiser la présentation des services offerts par une entreprise et d'en rendre l'accès transparent pour tout type de plate-forme, au travers d'un certain nombre de standards d'interopérabilité. Cette notion de services Web désigne essentiellement une application mise à disposition sur Internet par un fournisseur de service, et accessible par les clients au travers de protocoles Internet standards.

Nous présentons, dans un premier temps, le concept des services Web, leur architecture et leurs standards. Dans la suite nous introduisons la notion de service Web sémantique on présentant quelques approches de la description sémantique des services web. Finalement nous donnons les axes de recherche sur les services web.

1.2 Les services web

1.2.1 Présentation

Les services web sont une généralisation des RPC indépendamment d'une plate-forme et d'un modèle de programmation. Ils permettent par le biais d'Internet d'appeler des procédures distantes afin d'agir sur le serveur et/ou pour en recevoir des informations. **[1]**

Plusieurs définitions ont été proposées dans la littérature. Cependant, celle proposée par le World Wide Web Consortium (W3C) : **[2]**

[8] **S**ervice **O**riented **A**rchitecture. Http : //www.w3.org/TR/ws-arch/

"A Web service is a software system designed to support interoperable machine-to-machine interaction over a network. It has an interface described in a machine-processable format (specifically WSDL). Other systems interact with the Web service in a manner prescribed by its description using SOAP messages, typically conveyed using HTTP with an XML serialisation in conjunction with other Web-related standards"

Cette définition met en valeur les avantages principaux d'un service Web, à savoir :

- Son interface décrite d'une manière interprétable par les machines, qui permet aux applications clientes d'accéder aux services de manière automatique.
- Son utilisation de langages et protocoles indépendants des plateformes d'implantation, qui renforcent l'interopérabilité entre services.
- Son utilisation des normes actuelles du Web, qui permettent la réalisation des interactions faiblement couplées et favorisent aussi l'interopérabilité.

Autour de cette définition, la **pile standard de protocoles des services Web** fournit les fonctionnalités requises par les SOA figure 1.1.

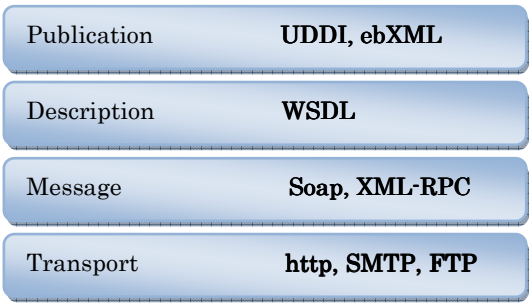


FIGURE 1.1 – Couches de fonctionnalités et protocoles/langages correspondants

1.2.2 Architecture des services web

Les architectures en informatique proposent des schémas directifs et une vision sur le fonctionnement et la dynamique des systèmes. Pour promouvoir l'interopérabilité et l'extensibilité du paradigme des services Web, une architecture de référence est nécessaire afin de préserver les objectifs initiaux visés par les services Web lors des évolutions technologiques successives.

Selon le" Web Service architecture working group" du W3C, SOA fournit un modèle conceptuel et le contexte pour comprendre les Web services et les relations entre les différents acteurs du modèle. En effet, cette architecture comporte six types d'acteurs figure 1.2 : **[8]**

- Des fournisseurs de Web services responsables des services, ce qui peut être parfois assimile à des serveurs de Web services,
- Un client de Web service, qui émet des requêtes
- Un annuaire ou registre de Web services, devant en principe fournir les informations nécessaires à la localisation des Web services,
- Des services (un service peut être implémenté par plusieurs programmes ou composants différent, en ce sens, un Web service est un Web service sémantique, car il doit spécifier ce qu'il fait.
- Les ressources, c'est à dire l'URL ou l'URI du Web service,
- Les messages que les Web services s'échangent, et dont le contenu aujourd'hui n'a pas de sémantique.

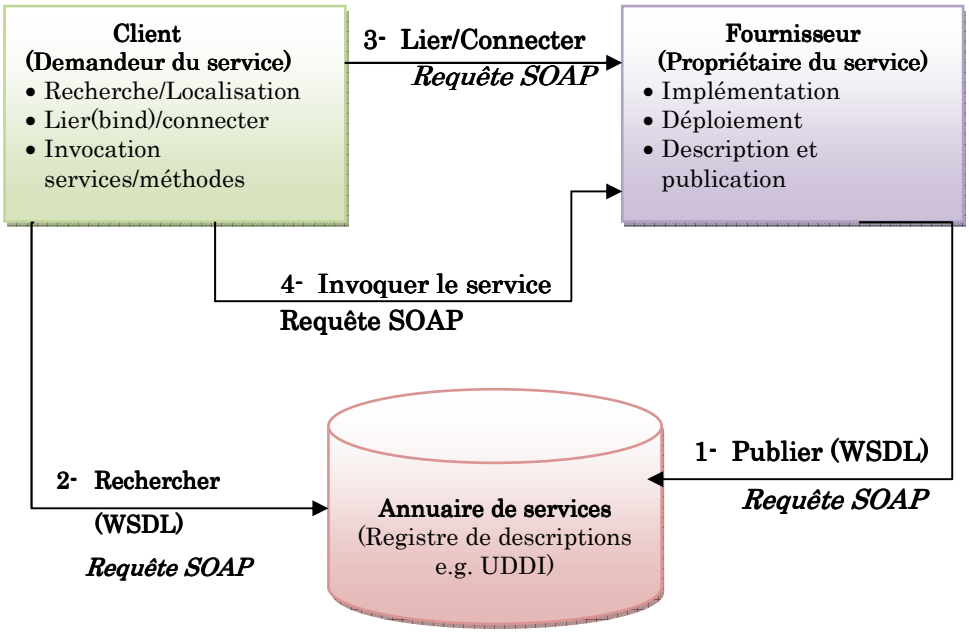


FIGURE 1.2 – Architecture de référence des web services.

1.2.3 Les standards en jeu dans les services web

De cette architecture SOA, nous suivons les multiples facettes que les Web services possèdent. Aujourd'hui il existe trois spécifications, respectivement le protocole d'échange des messages, la description fonctionnelle des Web services,

les registres de Web. Ils sont considères comme des standards car ils ont été recommandes par le W3C

Simple Object Access Protocol (SOAP)

Pour les architectures des services web, le protocole de base de communication entre deux parties est le protocole Simple Object Access Protocol (SOAP)... Puisque SOAP est le protocole de base de transmission de messages pour des services web, les autres protocoles sont construits au dessus des spécifications de SOAP ou ont des attachements à SOAP. [7]

Le protocole SOAP est un protocole de communication basé sur XML qui permet aux services Web d'échanger des informations dans un environnement décentralisé et distribué tout en s'affranchissant des plateformes et des langages de programmation utilisés. Il définit un ensemble de règles pour structurer les messages envoyés.

SOAP définit deux types de massages, Requête (Request) et Réponse (Response), permettant aux demandeurs de demander un service via une procédure d'accès à distance du service sollicité et aux fournisseurs de répondre à de telles requêtes

Un message de SOAP est un document formé en XML contient 3 éléments : *Envelope* , un élément facultatif en-tête (*Header*), et un élément obligatoire le corps (Body), comme montre la figure 1.3.

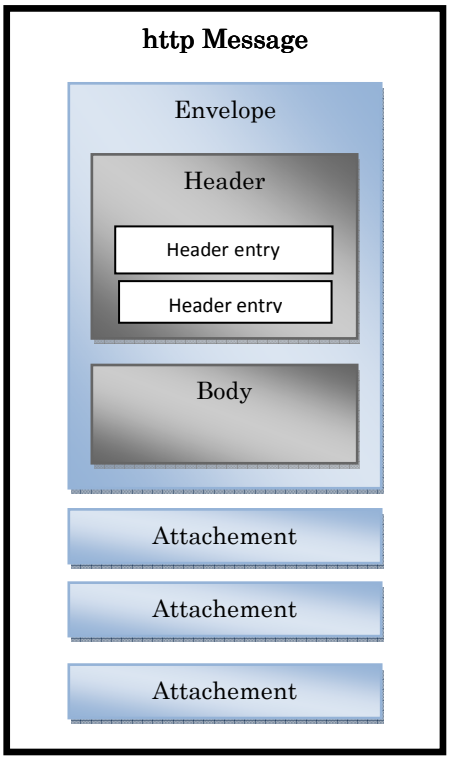


FIGURE 1.3 – Structure d'un message SOAP

- ***Enveloppe (Envelope)*** : définit les divers namespaces qui sont utilisés par le reste du message SOAP.
- ***L'entête (Header)*** : est un élément optionnel pour la diffusion des informations auxiliaires d'authentification, transaction et payement. Si l'entête est présent il doit être le premier enfant de l'enveloppe.
- ***Le corps (Body) :*** est l'élément principale du message, quand SOAP est utilisé pour exécuter un RPC, le corps contient un élément simple contenant le nom de la méthode et ces arguments ainsi de l'adresse cible du service.

Pour la réponse SOAP est retournée sous forme d'un document XML dans une réponse standard http. Le document XML retourné à la même structure que la requête SOAP sauf que le corps contient le résultat encodé de la méthode. **[7]**

La figure 1.4.a montre une requête SOAP qui invoque une méthode distante nommée
GetLastTradePrice et nous lui passons le paramètre "DEF" dont la définition se situe à l'adresse "http://example.org/quotes".

```
<SOAP-ENV:Envelope
  xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
    SOAP-
ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">

        <SOAP-ENV:Body>
        <m:GetLastTradePrice xmlns:m="http://example.org/quotes">
        <symbol> DEF </symbol>
        </m:GetLastTradePrice>
        </SOAP-ENV:Body>
```

FIGURE 1.4.a – Une requête SOAP

La réponse SOAP contient le prix retourné 34.5 voir la figure.1.4.b.

```
<SOAP-ENV:Envelope
xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/>

        <SOAP-ENV:Body>
        <m:GetLastTradePriceResponse xmlns:m="http://example.org/quotes">
        <Price>34.5</Price>
        </m:GetLastTradePriceResponse>
        </SOAP-ENV:Body>


</SOAP-ENV:Envelope>
```

FIGURE 1.4.b – Une réponse SOAP

Universal Description, Discovery and Integration (UDDI)

Il fait l'objet d'une spécification et permet la publication et la recherche et l'accès aux services web par l'Internet. Les requêtes opérées des services UDDI sont encapsulées dans une enveloppe SOAP. Ces requêtes sont destinées à la recherche ou à la publication des services. [1] L'UDDI contient les références permettant de retrouver les services, et non les services eux-mêmes, ce qui semble logique pour un annuaire. Un annuaire UDDI peut être public ou prive. Les premiers sont héberges par des sociétés comme IBM ou Microsoft. Ils sont moins développes que les annuaires prives car ils ne sont pas suffisamment sécurises. Les annuaires prives peuvent être héberges par une société quelconque sur un réseau prive ou sur internet. [8]

UDDI offre une manière uniforme de définir des annuaires de services Web en trois parties :

- ***les pages blanches :*** contiennent des informations et des identifiants sur les entreprises fournissant les services (nom de business, adresse, identifiant unique,…)
- ***les pages jaunes*** : décrivent les services métiers,
- ***les pages vertes :*** contiennent les données techniques (implémentation, business process, . . .) [1], [8]

Le registre UDDI est composé de quatre types d'informations Il est basé sur XML :

- ***BusinessEntity :*** Cette structure décrit l'entreprise, une ou plusieurs catégories auxquelles elle appartient (il existe plusieurs taxonomies), une description et l'adresse électronique des contacts qu'elle met à la disposition des clients pour assurer une assistance ou fournir des renseignements. Elle est en relation avec une ou plusieurs définitions des services ("businessServices") fournis par l'entreprise. Les champs "categoryBag" et "identifierBag" permettent de distinguer les entreprises suivant certains critères et permettent de faciliter la recherche dans le registre UDDI.
- ***BusinessService :*** Cette structure est décrite au moyen d'une entrée "serviceKey" identifiant le service publié par une entité d'affaire "businessEntity". Elle fait référence à une catégorie de service pour permettre une recherche en fonction d'un type de service particulier. Enfin, elle pointe vers une liste de liens ("bindingTemplates") décrivant les différents accès au service.

- ***BindingTemplate :*** Une entreprise peut proposer ses services par différents moyens de communication ou protocoles. Pour chacune des méthodes, elle présente un enregistrement de type "bindingTemplate". Ces derniers pointent vers une structure "tModel" et vers un point d'accès au service sur le web via HTTP, FTP, SMTP...
- ***tModel*** :Cette dernière structure permet de mettre en relation les trois précédentes avec des classifications préétablies. C'est un index utilisé pour classer les "businessEntity", les "businessService" et les "bindingTemplate". Cet index permet de rechercher les services en rapport avec des schémas de classification (taxonomies) et pointe vers un URL dans lequel se trouve une description ou des instructions particulières concernant un protocole.

L'interface de programmation pour les applications API[9] dialoguant avec UDDI comprend des méthodes pour interroger le registre ou pour le modifier. Les APIs sont fournies pour localiser les fournisseurs de services, les services, bindings et tModels. La publication des APIs est incluse pour la création et suppression des données UDDI dans l'annuaire de services. Des APIs des UDDIs sont basées sur le SOAP.

La figure1.6.a montre la découverte d'un service auprès de fournisseur de services "GetQuote Company " dans l'annuaire de services via l'utilisation d'un message SOAP.

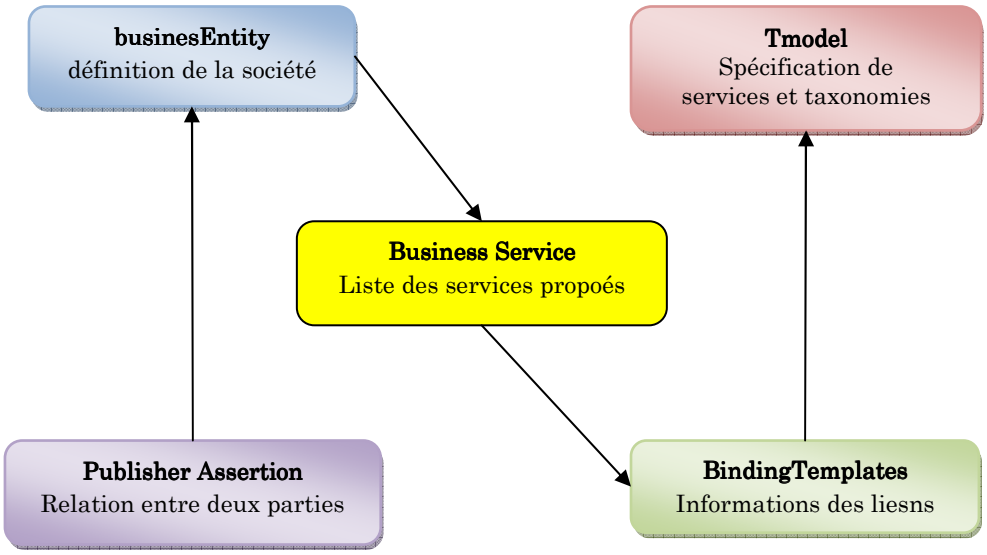


FIGURE 1.5 – Architecture de UDDI

[9] Application Program Interface sont des interfaces publiques qui permettent a des applications de communiquer

```
< Envelope xmlns:s="http://schemas.xmlsoap.org/soap/envelope/">
< Body>
  <find_business xmlns="urn:uddi-org:api" generic="1.0">
    <name>GetQuote Company</name>
  </find business>
</Body>
</Envelope>
```

FIGURE 1.6.a – Message SOAP pour la découverte d'un service

La réponse de cette requête sont les services "*stock quote*" et "Portfolio Management" **figure.1.6.b**.

```
<businessInfo businessKey="0076B468-EB27-42E5-AC09-9955CFF462A3">
      <name>GetQuote Company</name>
       <description xml:lang="en">Features portoflio managers, stock quote
and other
            stock market related services</description>
       <serviceInfos>
            <serviceInfo businessKey="0076B468-42E5-AC09-9955CFF462A3"
                 serviceKey="1FFE1F71-B788-09AF7FF151A4">
       <name>Stock Quote</name>
       </serviceInfo>
        <serviceInfo businessKey="0076B468-EB27-42E5-AC09-
9955CFF462A3"
          serviceKey="8BF2F51F-8ED4-43FE-B665-38D8205D1333">
       <name>Portfolio Management    </name>
       </serviceInfo>
```

FIGURE 1.6.b –Réponse SOAP de la requête de découverte

Web Service Description Language (WSDL)

C'est un langage de description des services web basé sur XML publié par le W3C. Il permet à une application d'obtenir des informations sur les services proposés et sur la façon de les invoquer. La description des web services autorise l'automatisation des transactions. [1] Cette description est faite dans un fichier de description WSDL qui se compose de deux parties : interface et implémentation de service web. [6]

La spécification WSDL est définie selon une sémantique totalement indépendante du modèle de programmation de l'application. Elle sépare clairement la définition abstraite du service (échange de messages) de ses mécanismes de liaison (définition des protocoles applicatifs) ce qui est un point important pour assurer l'interopérabilité des services.

Pour qu'une application puisse utiliser un service web, l'interface de ce dernier doit être décrite et spécifié avec précision dans un document WSDL, ce document définit un service web comme une collection de point finaux dans un réseau (network endpoints) ou *ports*. La définition abstraite du endpoints et message est séparée de leur format de données dans le réseau. [6]

Un document de WSDL emploie les éléments suivants dans la définition des services figure 1.7:

- ***Type*** : un récipient pour la définition de types de données.
- ***Message*** : Une définition abstraite de données échangées
- ***Operation*** : Une description abstraite de l'action supportée par le service.
- ***Port Type*** : un ensemble abstrait d'opérations supportées par un ou plusieurs endpoints.
- ***Binding*** : une spécification concrète de format de données et protocole pour un *port type* particulier.
- ***Port*** : un endpoint définit comme une combinaison de l'adresse réseau de service et binding.
- ***Service*** : collection d'endpoints

WSDL permet de décrire un service du point de vue technique : les ports, les types de messages échangés... mais ne donne aucune information sur le rôle que doit jouer le service d'un point de vue fonctionnel : le service Web n'a pas de description sémantique. L'idée de base de service web sémantique est d'ajouter de

la sémantique aux Web services dont le but de rendre ces derniers accessible et utilisable par les programmes et agents logiciel. Des Web services sémantiques seraient intuitivement des programmes dont les effets sur leur environnement seraient connus et dont les données manipulées posséderaient aussi une sémantique. [15]

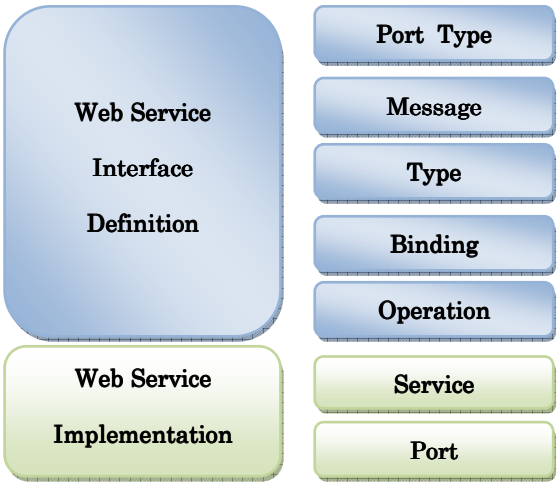


FIGURE 1.7–Architecture de WSDL

1.3 Service web sémantique

Les services web sémantiques se situent à la convergence de deux domaines de recherche importants qui concernent les technologies de l'Internet, à savoir le web sémantique et les services web.

1.3.1 Web sémantique

Le Web actuel 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.

Le but du web sémantique est de développer un web dont le contenu s'adresse, au moins pour partie, aux machines, afin qu'elles puissent aider les utilisateurs humains. Si l'on cherche à préciser, un tel web doit doter ses ressources (documents, service…) d'annotations dont le but n'est pas d'assurer l'affichage des documents mais l'appréhension de son contenu par divers outils logiciels grâce à la représentation *sémantique* de leurs contenus. [13][14] Depuis le début

des années 2000, de nombreux axes de recherches concernant Internet se sont tournés vers la nouvelle génération du web « le Web sémantique ».

Le Web sémantique désigne un ensemble de technologies visant à rendre le contenu des ressources du World Wide Web accessible et utilisable par les programmes et agents logiciels, grâce à un système de métadonnées formelles, utilisant notamment la famille de langages développés par le W3C[10].

Le principe du web sémantique, comme celui du web, est son ouverture : la possibilité de faire référence à des ressources distantes, non maîtrisées et la possibilité d'enrichir ces ressources sans être entravé par des barrières physiques, techniques ou même réglementaires. Pour hériter de ces particularités du web, le web sémantique tirera parti d'un espace d'adressage global des ressources permettant d'ajouter en permanence de l'information à ce web et d'y avoir accès sans entrave. Il offrira des langages permettant de partager et d'échanger la description de ces ressources. Une application du web sémantique est donc une application qui se nourrit des descriptions des ressources du web et qui, en retour, produit de telles descriptions.

Le web sémantique doit donc être une infrastructure juxtaposant au web actuel des documents structurés par des langages pour exprimer la connaissance, pour décrire les relations entre les connaissances, pour décrire les conditions d'utilisation et pour décrire les garanties et les modes de payement, et des dispositifs permettant de trouver les ressources. **[14]**

Le web sémantique est une extension du web, en rajoutant une sémantique au web, et le web service est une application logicielle autonome syntaxiquement compréhensible par n'importe quel système ou application car son interface est conçue par des langages standards comme (XML, Etc.) pour rendre un accès dynamique au web. L'intersection de ces deux (Web sémantique et service web) engendre « le service web sémantique »

1.3.2 Service web sémantique

Le service web sémantique est le résultat du l'évolution syntaxique du web service avec le web sémantique

De manière générale, l'objectif visé par la notion de web services sémantiques est de créer un web sémantique de services dont les propriétés, les capacités, les interfaces et les effets sont décrits de manière non ambiguë et exploitable par des machines et ce en utilisant les couches techniques sans pour autant en être

[10] World Wide Web Consortium

conceptuellement dépendants. La sémantique ainsi exprimée permettra l'automatisation des fonctionnalités suivantes qui sont nécessaires pour une collaboration interentreprises efficace: [11]

- Processus de description et de publication des services;
- Découverte des services;
- Sélection des services;
- Composition des services;
- Fourniture et administration des services ;
- Négociation des contrats.

De nombreux langages ou extensions de langages, étudiés dans La section suivante, ont pour objectif de résoudre les problèmes de sémantique rencontrés par les services Web. Ces langages intègrent la sémantique des données dans les descriptions des services Web, qui deviennent des services Web sémantiques.

1.3.3 Description sémantique des services web

Pour atteindre l'interopérabilité sémantique, c'est-a-dire pour être en mesure de prendre en charge la signification des données qu'ils échangent, les services Web doivent être capables :

- D'interpréter correctement la sémantique des données qu'ils envoient et reçoivent,
- De décrire les fonctionnalités qu'ils fournissent en utilisant une sémantique explicite et compréhensible par les machines.

De nombreux langages et architectures sont proposés afin de décrire les services Web sémantiques, ces langages sont divisés en deux approches [9], La première approche consiste à développer un langage complet qui décrit les services Web ainsi que leur sémantique d'un seul bloc. La deuxième approche consiste à annoter les langages existants avec l'information sémantique. L'avantage principal de ce genre de solutions réside dans la facilité pour les fournisseurs de services d'adapter leurs descriptions existantes aux annotations proposées.

1.3.3.1 Langages de description sémantique

Web Ontology Language for Web services OWL-S

OWL-S est une ontologie et un langage pour les web services développé dans le cadre du projet DAML[11]. OWL-S se base sur OWL qui permet de décrire des ontologies. **[12]**

Ainsi OWL-S est une ontologie OWL particulière. OWL étant le successeur de DAML+OIL. L'ontologie OWL-S a pour objectif de décrire de façon non-ambigüe les web services de façon à ce qu'un agent logiciel puisse exploiter automatiquement ces informations. Il est compatible avec des formats de description syntaxiques tels que WSDL. **[9]** Les bénéfices de l'emploi d'une ontologie sont multiples, mais le premier qui peut venir à l'esprit concerne l'amélioration de la recherche de services par rapport aux solutions existantes telles UDDI. OWL-S s'applique à renseigner les Web Services de façon à pouvoir exécuter les tâches suivantes :

1. *La découverte automatique de Web Services* : L'objectif est ici de pouvoir découvrir dynamiquement et automatiquement des services pour un besoin donné.

2. *L'invocation automatique des Web Service*s : On cherche ici à obtenir les opérations proposées par un service (nom, paramètres, retour) ainsi que les relations qui les unissent dans le but de réaliser une transaction complète.

3. *Composition et interopérabilité de Web Services* : Cette tâche implique la sélection, composition et interopérabilité automatique de services web pour résoudre certaines tâches complexe nécessitant l'utilisation de plusieurs services web. Pour atteindre cet objectif automatiquement sans l'intervention de l'être humain. OWLS fourni les informations nécessaires pour la sélection et composition de services web encodées dans des sites web, ces informations sont interprétables par des machines.

4. *Surveillance automatisée de l'exécution de Web Services* : L'exécution d'une opération d'un service n'est pas forcément immédiate et/ou de courte durée. Aussi l'utilisateur va vouloir surveiller la progression de cette exécution et obtenir un retour du succès ou de l''echec d'une opération. Cette perspective est évoquée par les auteurs de la spécification d'OWL-S, mais ils ne se sont pas encore penchés dessus.

OWL-S partitionne la description sémantique d'un service web en 3 parties figure1.8 *ServiceProfile* : exprime ce que le service propose ainsi que les pré-

[11] http://www.daml.org/.

requis que son emploi imposent, *ServiceModel* : exprime le fonctionnement du service, *ServiceGrounding:* exprime comment le service est utilisé. **[15]**

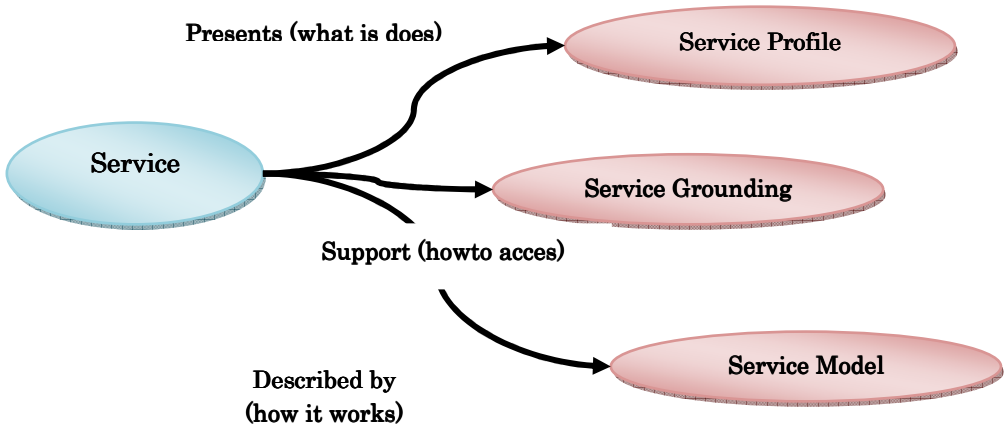


FIGURE 1.8–Les modules d'OWL-S

ServiceProfile : Analogue au service page-jaune de l'UDDI **[15]**, il décrit ce que fait le service. Un système recherchant un service examinerait la première fois le «profile» pour voir si le service fournit ce qui est nécessaire. Le « profile» contient les informations suivantes : **[12]**

- *Le fournisseur du service* : Cette information précise les données nécessaires pour identifier et contacter le fournisseur du service.
- *La description fonctionnelle* : Elle spécifie ce que l'on peut attendre du service en termes d'entrées attendues et de résultats produits en sortie. Elle mentionne également les pré-conditions et les effets du service.
- *Propriétés additionnelles* : Plusieurs propriétés sont utilisées pour qualifier le service. La catégorie du service, la qualité de service, une indication de temps de service, une restriction d'usage géographique,. . .

ServiceModel : Donne les conditions nécessaires à l'exécution d'un service ainsi que ces conséquences. Il inclut des informations concernant les entrées, sorties, pré conditions et effets d'un service. Dans le cas d'un service complexe (composé de plusieurs étapes dans le temps), le modèle de processus détaille la décomposition du service en composants plus simples en en explicitant les liaisons et structures de contrôle permettant l'enchaînement de ces composants.

ServiceGounding : Mapping de OWL-S vers WSDL **[15]**, il spécifie la manière, pour un agent, d'accéder au service concerné. Typiquement, elle spécifie les protocoles de communication à utiliser ainsi que les éléments spécifiques au

service comme, par exemple, les ports à utiliser. Le Service Grounding doit définir, pour chaque type abstrait spécifié dans le Service Model, une façon non ambigüe d'échanger des données de ce type avec le service.

La figure1.9 montre le lien entre la description OWL-S et la spécification WSDL : **[15]**

- Un processus atomique d'OWL-S correspond à une operation WSDL
- Un input ou un output d'un porcessus atomique correspond à un message dans le WSDL
- Les types de message d'un processus atomique correspond à des types abstrait de WSDL

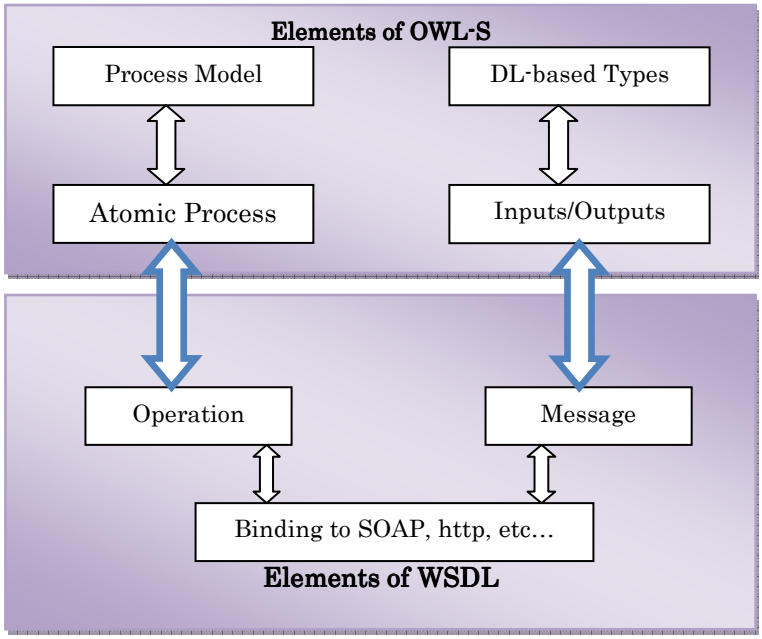


FIGURE 1.9–Lien entre WSDL el OWL-S

Web Service Modeling Ontology WSMO

WSMO [12] est un framework des services web sémantique qui affine et étend le framework WSMF[13] en une méta-ontologie des services web sémantiques. WSMO est accompagné par un langage formel WSML [14] qui permet d'écrire des annotations des services web en fonction du modèle conceptuel. **[16]**

Feier et all **[17]** ont donnés les concepts principaux du WSMO à savoir :

[12] http ://www.wsmo.org/
[13] Web Service Modeling Framework
[14] Web Service Modeling Language

Conformité du Web : WSOM hérite le concept URI [15]pour une identification unique de ressources comme un principe essentiel pour la conception du Web.

Utilisation des Ontologies : les ontologies sont utilisées comme un modèle de données dans WSMO, signifiant que toutes les descriptions de ressources à savoir toutes les données durant l'utilisation de service sont basées sur des ontologies. L'utilisation des ontologies augmente le traitement automatique de l'information ainsi qu'une meilleure interopérabilité entre services et applications.

Découplage strict : chaque ressource WSMO est spécifiée indépendamment par rapport aux autres ressources. Qui est motivé par la nature ouverte et distribuée du web.

Centralité de la médiation : la médiation adresse la manipulation des environnements ouverts hétérogènes, et afin de complémenter le principe de *découplage strict*, WSMO donne une importance de la médiation pour obtenir un déploiement réussi de service web par la mise on œuvre de la médiation en premier ordre de l'architecture du framework WSMO.

Séparation de rôle d'ontologies (Ontological Role Separation) : les demandes des utilisateurs sont formulées indépendamment et dans un contexte différent pour les services web disponibles. L'épistémologie fondamentale du WSMO met une différence entre ce désirent les utilisateurs avec les services web disponibles.

L'architecture de WSMO est organisée autour de quatre éléments principaux :**[9], [16]**

- **Les Ontologies** fournissent la terminologie de référence aux autres éléments de WSMO, afin de spécifier le vocabulaire du domaine de connaissance d'une manière interprétable par les machines.
- **Les services Web** sont définis comme des < entités computationnelles > qui fournissent une fonctionnalité. Une description est associée chaque service, dans le but de décrire sa fonctionnalité, son interface, et ses détails internes.
- **Les buts (Goals)** servent à décrire les souhaits des utilisateurs en termes de fonctionnalités requises. Les objectifs sont une vue orientée utilisateur du processus d'utilisation des services Web, ils sont une entité à part entière dans le modèle WSMO. Un objectif décrit la fonctionnalité, les entrées/sorties, les pré conditions et post conditions d'un service Web.
- **Les Médiateurs (Mediators)** WSMO inclut les médiateurs comme des composants centraux de son architecture, ils sont utilisés pour résoudre de nombreux problèmes d'incompatibilité, telles que les incompatibilités de données dans le cas où les services Web utilisent différentes terminologies,

[15] Universal Ressource Identifier

les incompatibilités de processus dans le cas de la combinaison de services Web, et les incompatibilités de protocoles lors de l'établissement des communications. Les hétérogénéités rencontrées sont gérés par les types de médiation suivants :

- ✓ La **médiation de données** résout les incompatibilités de représentation des données.
- ✓ La **médiation de processus** est relative à la logique applicative de la composition.
- ✓ La **médiation de protocoles** adapte les différents protocoles de communication utilisés.

Les langages de description sémantique définissent une nouvelle façon de décrire les services web souffrent de quelques limitations importantes. Tout d'abord, ne sont pas alignée sur les normes de services Web existants, par exemple si le grounding model de OWL-S utilise WSDL binding, le profil model de OWL-S reproduit la description incarné dans le reste du WSDL. Deuxièmement, il suppose que tout le monde utilise OWL pour représenter les ontologies mai qui n'est pas toujours le cas. Pour surmonter ces limitations, une nouvelle approche est proposée et qui consiste à une évolution compatible des normes de services Web existants

Contrairement aux langages précédents, plusieurs travaux proposent d'étendre les normes existantes par une annotation, soit en utilisant les éléments d'extensibilités prévus à cet effet, soit en modifiant les fonctionnalités initiales des normes.

1.3.3.2 Annotation des langages existants

L'avantage principal de ce genre de solutions réside dans la facilité pour les fournisseurs de services d'adapter leurs descriptions existantes aux annotations proposées.

Annotation de WSDL

La norme WSDL opère au niveau syntaxique et n'a pas de sémantique pour représenter les besoins et les capacités des services web.

L'annotation de WSDL est basé sur l'analyse de la description WSDL, trois types d'éléments peuvent faire augmenter leur sémantique à travers l'annotation

sémantique avec les concepts d'ontologies qui sont : *Opération, Messages, Préconditions et Effets*. Tous ces éléments existent dans le fichier WSDL.

Les avantages d'ajouter de la sémantique au WSDL sont : **[17]**

- Les utilisateurs et d'une manière compatible vers le haut décrivent les niveaux sémantiques et opérationnels dans WSDL qui est un langage reconnu par les développeurs
- Une description sémantique indépendamment au langage de représentation sémantique utilisé, permettant aux développeurs de choisir le langage sémantique adéquat est préféré.

Une extension du WSDL qui utilise un mécanisme d'annotation sémantique aux services web est le WSDL-S qui est un langage de description sémantique développé par IBM et l'université de Georgia, ce langage augmente l'expressivité et la description WSDL en utilisant les mêmes concepts semblables à l'OWL-S.

WSDL-S

Proposé par R.Akkiraju & all dans **[17]** qui se base sur l'ajout des annotations sémantique aux documents de description WSDL. Cette approche est guidée par les principes suivants :

- *Développement d'une approche basée sur les standards utilisés dans les services web* : les standards utilisés dans les services web (http, FTP, SOAP, WSDL, UDDI..) sont rapidement devenus les technologies préférées pour l'intégration et interopérabilité d'application, car ils présentent une forte interopérabilité entre application et systèmes hétérogènes. Et comme la majorité des entreprises et organisations mettent leurs investissements sur les projets d'intégrations basées sur les services web. R.Akkiraju et all croient que Le développement d'une approche basée sur l'ajout sémantique au service web revient fortement à utiliser ou faire une extension aux standards existants, dans leurs cas ou ils ont utilisés WSDL.
- *Le mécanisme d'annotation sémantique de services web doit être obligatoirement indépendant aux langages de représentations sémantiques :* il y a plusieurs langages de représentation sémantique parmi lesquelles en peut citer OWL, WSMO, UML[16]. Chaque langage offre différents niveaux d'expression sémantique et supports de développement. Donc il est fortement recommandé de développer une approche qui n'est pas attaché à un langage de représentation sémantique particulier, cette approche qui utilise un mécanisme de représentation sémantique séparée au langage

[16] Unified Modeling Language. Il s'agit d'un langage de modélisation de 3ème génération. Voir http://www.uml.org/

sémantique utilisé offre une grande flexibilité permettant aux développeurs de sélectionner le langage de représentation sémantique favorisé pour eux.

- *Le mécanisme d'annotation sémantique de services web permet une association de multiples annotations écrites par différents langages de représentation sémantique :* nous avons vu précédemment qu'il y on a plusieurs langages de représentation sémantique. Les fournisseurs de services web ont le choix de langage sémantique utilisé pour l'annotation sémantique de leurs services web. Donc le mécanisme d'annotation sémantique associé aux services web doit présenter une possibilité d'annotation sémantique de différents langages sémantiques.
- *Supporter l'annotation sémantique des services web dont leurs types de données décrits avec l'XML schéma* : une pratique courante dans les services web basée sur intégration est la réutilisation des interfaces décrites en XML, la définition des documents d'affaires (business documents) utilisant XML schéma est une pratique répandue et réussie. XML schéma est une définition importante de formats de données dans le futur, les auteurs croyons que l'annotation sémantique des entrées et sorties de service web doit supporter l'annotation XML schémas.
- *Fournir l'appui pour des riches traçant des mécanismes entre types schéma de services web et ontologies* : il est important de faire l'annotation XML schémas dans les descriptions de services web, l'attention devrait être donnée au problème de tracé (mapping) les types complexes de XML schémas aux concepts d'ontologie.

SAWSDL

Est un ensemble d'attributs d'extension permettant de décrire la sémantique des éléments contenus dans les documents WSDL. L'objectif de SAWSDL est de définir comment une annotation doit être réalisée, tout en laissant le choix du langage utilisé pour la description sémantique. SAWSDL fournit les mécanismes permettant d'attacher des concepts décrits dans des ontologies aux annotations des descriptions WSDL. Cette proposition étend WSDL-S, elle peut être considérée comme une continuité de ce langage. Elle vise à apporter la valeur ajoutée de la sémantique non seulement lors de l'invocation des services Web, mais aussi durant la phase de découverte. Trois attributs d'extensibilité sont définis par défaut : l'attribut "modelReference " permet l'association entre un composant WSDL et un concept d'une ontologie, et les attributs "liftingSchemaMapping" et "lowering-SchemaMapping" sont ajoutés aux définitions de types pour spécifier les correspondances entre les éléments du schéma des données et l'information sémantique de l'ontologie.

Annotation de l'UDDI

Cette approche vise à améliorer la publication et la découverte de services Web dans les registres UDDI. Ils étendent les descriptions UDDI avec l'information sémantique nécessaire pour décrire les capacités des services Web, en utilisant le langage DAML-S, prédécesseur d'OWL-S. Ils soutiennent que la découverte des fonctionnalités de services Web ne peut être effectuée qu'au niveau sémantique. En effet, sans sémantique explicite, deux descriptions équivalentes peuvent être interprétées différemment, selon les circonstances de leur utilisation. Ainsi, l'automatisation de la découverte des services Web passe par la description explicite de leurs capacités. Les auteurs décrivent une méthode pour effectuer la traduction de la représentation DAML-S vers la représentation UDDI. De plus, ils proposent un algorithme de correspondance sémantique des fonctionnalités offertes par les services Web, sur la base de leur représentation UDDI modifiée. Ainsi, la découverte de services Web tire avantage des descriptions sémantiques qui ont été insérées dans le standard UDDI.

1.4 Problématique de recherche sur les services Web

Des recherches faites sur les services Web par rapport à SOA couvrent plusieurs problèmes intéressants : la Sélection des services Web, la Découverte et Découverte dynamique des services Web, la Composition des services Web… etc.

1.4.1 Sélection des servies Web

Avec la sélection des services web, on cherche à choisir le meilleur fournisseur d'un service web, étant donné un ensemble de fournisseurs de ce service. Un nouveau modèle de registre et découverte de services web a été définit basé sur la Qualité de Service (i.e. QdS en anglais Quality of Service - QoS) . La plate-forme de cette proposition est constituée de quatre rôles: Fournisseur de services web; Consommateur de services Web; Certificateur de la Qualité de Service et le nouvel registre. Le fournisseur du service offre le service Web en publiant ce dernier dans le nouveau registre; le consommateur a besoin du service web offert par le fournisseur ; le nouvel registre UDDI est un lieu de stockage de services web enregistrés avec facilités de recherche. Le certificateur QdS sert à vérifier les revendications de qualité de service pour un service Web. Le nouveau registre est différent du actuel modèle UDDI en ayant information sur la description fonctionnelle du service Web, et sur la qualité de service enregistré dans le stockage. La consultation pourrait être faite selon la description fonctionnelle du service Web désirable, avec la qualité exigée de service attribue comme des contraintes de consultation.

1.4.2 Découverte de services Web

Dans le contexte d'une application qui a besoin d'exécuter une fonctionnalité implémentée comme un service web par plusieurs fournisseurs, la découverte fait référence au processus de recherche des services web implémentant la fonctionnalité souhaitée. Les registres UDDI sont des entités qui servent d'appui à la découverte de services web pour les applications client. De cette façon une application interroge un registre UDDI pour les fournisseurs d'un service web.

1.4.3 Découverte dynamique

La découverte dynamique d'un service Web s'effectue d'une façon différente. Au lieu de s'enregistrer dans un registre connu, les services Web découverts dynamiquement annoncent explicitement leur arrivée sur le réseau et leur départ. WS-Discovery définit des protocoles pour annoncer et découvrir des services Web via des messages de multi diffusion.
Lorsqu'un service Web se connecte à un réseau, il annonce son arrivée en émettant un message "Hello". Lorsqu'un service quitte un réseau, WS-Discovery précise qu'un message "Bye" doit être envoyé au réseau ou au proxy de découverte. Ce message informe les autres services du réseau que le service Web concerné n'est plus disponible.

La découverte dynamique étend aussi l'architecture de services Web à des équipements, comme les imprimantes et les systèmes de stockage qui peuvent être intégrés dans un système sous la forme de services Web, sans avoir besoin d'outils ou de protocoles spécialisés.

1.4.4 Composition de Services Web

«Les composants sont destinés à être composés». La réutilisation est un avantage important de l'approche par composants. Les services Web, tels qu'ils sont présentés, sont conceptuellement limités à des fonctionnalités relativement simples qui sont modélisées par une collection d'opérations. Toutefois, pour certains types d'application, il est nécessaire de combiner un ensemble de services Web WSDL (services Web basiques) en services Web plus complexes (services Web agrégés ou composites) afin de répondre à des exigences plus complexes.

La composition des services web nécessite la réalisation des étapes suivantes : **[9]**

1. **La découverte des services Web** pouvant répondre aux besoins de la composition se fait généralement de manière manuelle par envoi de requêtes aux annuaires UDDI. De nombreux travaux visent à automatiser cette étape
2. **L'organisation des interactions** entre les services Web composés est distinguée par les termes chorégraphie et orchestration. La chorégraphie gère les échanges de messages avec un unique service Web. L'orchestration, quant à elle, organise les interactions entre plusieurs services Web. Une composition est associée à une spécification pour gérer les échanges de messages et mettre en place les structures de contrôle nécessaires (boucles itératives, opérateurs conditionnels et gestion des exceptions).
3. **L'exécution de la composition** est l'étape d'invocation effective des services Web participants à une composition. Une spécification de composition est hébergée par un serveur et son exécution est prise en charge par un moteur de composition, la surveillance de l'exécution par le moteur de composition permet de gérer certaines erreurs dynamiquement.

La composition de services constitue l'un des champs de recherche les plus actifs dans le domaine des services Web sur lequel nous nous concentrons dans notre travail.

1.5 Conclusion

Les services Web se présentent aujourd'hui comme un support crédible permettant à des applications d'exposer leurs fonctionnalités au travers d'interfaces standardisées et de plus en plus éprouvées. Les services Web ont pour vocation de favoriser une architecture orientée services, intégrant des systèmes hétérogènes complexes, fortement distribués et pouvant coopérer sans recourir à une intégration spécifique et coûteuse. La souplesse introduite par les services Web permet d'envisager de nouvelles formes de collaboration entre applications distantes.

Les standards des services web (SOAP, WSDL, UDDI) sont de bas niveau sans véritable sémantique. Ils permettent de créer des composants logiciels distribués, de décrire leur interface et de les utiliser indépendamment de la plate-forme sur laquelle ils sont implémentés. La recherche dans ce domaine est très active et s'intéresse entre autres à la composition, l'orchestration, la sécurité, la sémantique des services web…etc.

L'idée de base de service web sémantique est d'ajouter de la sémantique aux Web services dont le but de rendre ces derniers accessible et utilisable par les programmes et agents logiciel, La sémantique ainsi exprimée permettra l'automatisation de la découverte, la sélection et la composition des services web et qui sont les grands axes de recherche dans les services web.

Composer des services c'est créer de nouvelles fonctionnalités en combinant des fonctionnalités offertes par d'autres services existants Le chapitre 2 est dédié à cet axe,
La composition des services web.

☞ *Chapitre 2*

La composition des services Web

2.1 Introduction

« Les composants sont destinés à être composés. »

Un des défis des SOA est l'intégration de services ou de systèmes distribués pour l'approvisionnement de nouveaux services personnalisés, plus riches et plus intéressants aussi bien pour des applications, d'autres services ou plus communément pour des utilisateurs humains. En effet, si une application ou un client requièrent des fonctionnalités, et qu'aucun service n'est seul apte à les fournir, il devrait être possible de combiner ou de composer des services existants afin de répondre aux besoins de cette application ou de ce client. C'est ce que l'on appelle la composition de services.

La composition des services web (Service composition) peut être définit par la création d'un service composite en combinant des services composants disponible, elle est utilisée dans la situation où la requête d'un client ne peut pas être satisfaite par un seul service web mais par la combinaison de plusieurs services. En termes de génie logiciel, la composition ***automatique*** des services web augmente significativement la puissance des SOAs.

Dans ce chapitre en va étudier la composition automatique, tous d'abord en va donner un exemple motivant l'utilisation de la composition des services web, ensuite on va donner les différentes catégories de composition des services web. Finalement en va détailler la composition automatique des services web on expliquant les différentes techniques de cette composition.

2.2 Exemple motivant

Typiquement, le client localise au premier le service web qui peut server sa requête *r* à partir de l'UDDI, puis il obtient son API à partir de la spécification WSDL, le client envoi sa requête au service web via le standard SOAP et il reçoit la réponse. Le problème se pose lorsque un service web ne suffi pas pour répondre à la requête r alors comment composer plusieurs services pour satisfaire r ?

Considérons qu'on a 4 services web :

1. **findHotel** : reçoit en entrée le nom, la ville, et des informations d'état de l'hôtel, il retourne l'adresse et le code postale de l'hôtel

2. **findRestaurant** : reçoit en entrée le code postale et le repas préféré, il retourne le nom, le numéro du téléphone et l'adresse de plus proche restaurant

3. **guideRestaurant**: reçoit en entrée l'emplacement actuelle et le repas préféré, il retourne l'adresse et la catégorie du plus proche restaurant

4. **findDirection** : reçoit en entrée le point de départ et la destination, il retourne en détail le chemin vers la destination

Considérons le deux requêtes suivantes:

r1 : trouver l'adresse de l'hôtel "Cirta"
r2: trouver un restaurant traditionnel le plus proche de l'hôtel «Cirta » et donner son chemin.

Pour satisfaire ***r1*** il suffit d'invoquer le service **findHotel**(«Cirta », «Constantine »,«3 étoiles »).
Mais aucun service parmi les quatre peut satisfaire ***r2*** tous seul. Les deux services web **findRestaurant** et **guideRestaurant** peuvent trouver un restaurant algérien proche de l'hôtel «Cirta » mais ne peuvent pas donner son chemin. Le service web **findDirection** peut donner le chemin d'aller d'un emplacement à un autre mais ne peut trouver aucun restaurant. Alors il faut combiner plusieurs services pour satisfaire ***r2*** . Alors il faut tous d'abord invoquer **guideRestaurant** pour obtenir le plus proche restaurant, puis invoquer **findDirection** pour trouver le chemin.

1.3 Catégories de la composition de services

La composition de services reste un problème très complexe. Cette complexité tient au fait que les solutions de composition de services doivent tenir compte du nombre croissant de services déployés sur le web, de leur mise à jour continuelle et de leur hétérogénéité. En conséquence, elles nécessitent de traiter les problématiques de la coordination des services, leurs transactions, leur exécution et leur modèles de conversation (ou d'interaction). Ces dernières années, plusieurs communautés se sont intéressées à la problématique de la composition

de services. Des solutions ont notamment été proposées dans les communautés des bases de données, du web sémantique, et plus récemment dans celles des agents.

Les solutions proposées peuvent être classifiées selon deux axes: [19]

1.3.1 Selon que la sélection des services et la gestion du flot soient faites ou non a priori, une approche sera dite ***statique*** ou ***dynamique***.

1.3.1.1 Composition statique

On dit qu'une **composition** est **statique** lorsqu'elle prend place à l'étape de conception, au moment où l'architecture et la conception du système logiciel sont planifiées. Les composants (ou services) qui seront utilisés sont préalablement choisis et reliés, et la gestion du flot est effectuée a priori, l'agrégation des différents services Web se fait de manière statique avant exécution. Les approches statiques de composition de services sont celles adoptées par l'industrie, elles s'inspirent de la gestion de processus métiers quant à la description des données et des flots de contrôle pour le processus de composition.

1.3.2.2 Composition dynamique

Par opposition, une **composition** de services est dite **dynamique** si les services sont sélectionnés et composés à la volée en fonction des besoins formulés par l'utilisateur.

Une approche dynamique pour la composition de services offre le potentiel de réaliser des applications flexibles et adaptables en sélectionnant et en combinant les services de manière appropriée sur la base de la requête et du contexte de l'utilisateur. Ce type de composition peut engendrer de nombreuses applications utiles qui n'ont pas été prévues à l'étape de conception. Par conséquent, la composition dynamique de services est propice dans un environnement tel que le web où les composants disponibles sont dynamiques et les attentes des utilisateurs variables et personnalisées. C'est précisément ce type d'approche qui fait l'objet de notre travail.

1.3.2 En fonction du degré de participation de l'utilisateur dans la définition du schéma de composition, ces propositions *sont* ***manuelles, semi-automatiques*** ou ***automatiques***.

1.3.2.1 Composition manuelle

Les techniques de composition manuelles reposent sur un expert qui se charge de définir et générer, graphiquement ou moyennant un éditeur de texte, des scripts de workflow qui sont ensuite soumis à un moteur d'exécution de workflows. Elle est la plus adaptable aux besoins de l'utilisateur, car il va tout définir à son goût depuis le début, en générale les utilisateurs sont soutenus avec des interfaces graphiques GUI[17]pour faciliter la composition, le service composite est vulnérable au changement, la correction est manuelle si le service composite est non valable ou lorsque un nouveau service apparaît. La plupart d'entre elles sont entièrement statiques puisque les services à composer sont préalablement sélectionnés, et le flot de composition défini a priori. Elle est inappropriée dans le cas ou le problème de composition de services web est un problème avec une échelle large (Large-scale WSC problem).

1.3.2.2 Semi-automatique

Dans les approches semi-automatiques, des outils sont développées afin de proposer à l'utilisateur des suggestions sémantiques quant à la sélection des services à composer (ex. en fonction de leurs entrées/sorties, annotations, etc.) durant le processus de composition.

Les techniques de composition semi-automatiques sont un pas en avant en comparaison avec la composition manuelle, dans le sens qu'ils font des suggestions sémantiques pour aider à la sélection des services Web dans le processus de composition. **[10]**

Durant le processus de composition, le framework propose à l'utilisateur de sélectionner des buts, des médiateurs et des opérateurs de flot de contrôle. Une fois que le service composé est défini, l'outil instancie les opérateurs de contrôle ainsi que le workflow. **[19]**

1.3.2.3 Automatique

Les techniques de la composition automatique automatisent entièrement le processus de composition en le traitant généralement comme un problème de planification. Dans ce cas, le moteur de composition y est toujours dynamique. La composition totalement automatisée prend en charge tout le processus de composition et le réalise automatiquement, sans qu'aucune intervention de l'utilisateur ne soit requise. L'approche de la composition automatique peut résister au changement dynamique des services web car elle utilise des agents logiciels qui sont chargent de mettre la connexion entre les différents services

[17] Graphique User Interface

web impliqués dans le problème de composition, sachant que ces agents ont la capacité de savoir si deux services peuvent se connecter ou non via l'utilisation des correspondances syntaxiques ou sémantiques entre les paramètres des services web (Syntactic matching or semantic matching).

Le cadre de notre travail se situe dans ce type de composition, dans la suite de ce chapitre en va étudier en détail la composition automatique.

2.4 La composition automatique

2.4.1 Un framework pour la composition automatique des services web

Nous allons vous présenter un modèle général (general framework) pour la composition automatique de services web. [18] Ce framework est sous une forme abstraite à haute niveau qui ne prendre pas en considération des langages particuliers, plateformes ou méthodes utilisées dans le processus de composition.

Dans ce framework figure 2.1, le système de composition automatique de services web a deux participants qui sont le fournisseur de service (Provider) et l'utilisateur ou le consommateur de service (Service requester).le fournisseur de service propose les services web pour utilisation et le consommateur de service utilise le service proposé par le fournisseur de service. Le système contient aussi les composants suivants : Translateur (Translator), générateur de processus (Process Generator), évaluateur (Evaluator) et un moteur d'exécution et découverte de service (Execution engine and service repository).

Le translateur est un médiateur qui se charge de faire la translation entre les langages utilisés par les participants (Fournisseurs et utilisateurs de services) et le langage interne utilisé par le générateur de processus. Pour chaque requête fourni par l'utilisateur le générateur de processus maître ou génère des plan pour composer les services disponible dans l'annuaire de services afin de répondre à la requête d'utilisateur. Si le générateur de processus trouve plusieurs plans permettant la résolution de la requête de l'utilisateur, l'évaluateur se charge de choisir le meilleur plan pour l'exécution et finalement le moteur d'exécution et de découverte exécute le plan choisi et retourne le résultat d'exécution à l'utilisateur.

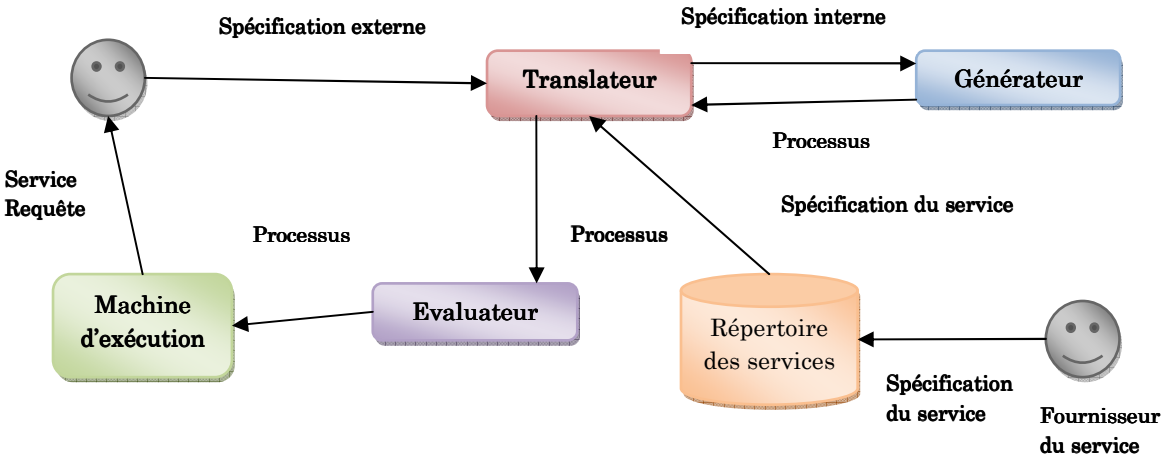


FIGURE 2.1 – Un framework général pour la composition automatique des services

Plus précisément, le processus de composition automatique de services Web contient les phases suivantes :

2.4.1.1 Présentation du service

Premièrement les fournisseurs de services publient leurs atomiques services dans un emplacement global centralisé. Pour publier un service Web (Advertise) Il y'a plusieurs langages universels permettant la publication des services web tel que **UDDI**, **DAML-S [23].** Pour bien présenter et publier un service web, le service doit contenir les attributs essentiels suivants *Signature*, *états* et les *valeurs non fonctionnelles*. La signature est représentée par les entrées, sorties et exceptions des services Web (Inputs, Outputs and exceptions), ils fournissent des informations sur les informations transmises durant le processus d'exécution d'un service web. Les états sont spécifiés par les pré-conditions et post-conditions qui modélisent la transformation du service web d'un état à un autre état. Les valeurs non fonctionnelles sont les attributs qui sont utilisées pour l'évaluation d'un service à savoir la description du coût, qualité et temps de réponse pour un service web (la qualité du service).

2.4.1.2 Traduction des langages

Plusieurs méthodes de composition de service différent entre les langages de spécification interne et externe du service web. Les langages externes sont utilisés pour faciliter l'accès aux utilisateurs dans le sens où ces utilisateurs peuvent exprimer comment ou quels sont les services utilisés d'une manière relativement simple. Les langages externes de spécification de services Web sont souvent différents des langages internes utilisés par le processus générateur de composition de services, car le processus générateur de composition a besoin des langages formels et précis comme par exemple les langages de programmation

logiques (Logical Programming Languages). En revanche de ceux des langages externes utilisés par les utilisateurs qui sont généralement des langages standards (e.g : WSDL, DAML-S..). Donc les composants de la traduction entre les langages externes de spécification de service web et les langages internes utilisés dans le processus de composition automatique de services web sont devenus obligatoires à développer.

2.4.1.3 Génération du modèle de processus de composition

L'utilisateur ou le consommateur (Service Repuester) de services peut exprimer son besoin à l'aide d'un langage de spécification de services. Le processus générateur se charge de répondre à la requête fournie par l'utilisateur par la découverte et composition de services atomiques publiés par les fournisseurs. Le processus générateur prend les fonctionnalités de services comme *entrées* et génère le modèle du processus (process model) qui décrit le service composite. Le modèle du processus contient un ensemble de services atomiques sélectionnés d'une façon permettant un échange de flots de contrôle et flots de données entre ces services.

2.4.1.4 Evaluation du service composite

Parfois on trouve plusieurs services qui ont des fonctionnalités similaires. Donc il est possible que le processus générateur génère plusieurs plans de service composite répond à la requête utilisateur. Dans ce cas le module évaluateur des services composites se charge de sélectionner le meilleur plan du service composite en utilisant généralement les informations et attributs non fonctionnels des services web.

2.4.1.5 Exécution du service composite

Après la sélection du meilleur plan du processus composite de services, on exécute le service composite. L'exécution du service composite suit une séquence de message défini par le modèle du processus de service composite. Le flux de données du service composite est défini par l'échange des actions ou de données entre un service et un autre service impliqué dans le processus de composition. Cet échange est traduit par le transfert des sorties (outputs) d'un service au entrées (inputs) d'un autre service web.

2.4.2 Techniques de la composition automatique

Les différentes approches existantes pour la composition de services Web peuvent être regroupées en deux courants. La première technique de workflow se base sur le fait qu'un service Web composite est définit par un ensemble de services atomiques et par la façon dont ils communiquent entre eux. Cette

définition est de même type que la manière dont est défini un processus métier. Dans le second courant, la composition est vue comme la génération automatique d'un plan d'exécution des services Web. Les approches envisagées sont des approches du domaine de l'intelligence artificielle et plus particulièrement de la planification. Ces approches sont basées sur le fait qu'un service Web peut être spécifié par ses préconditions et ses effets.

2.4.2.1 Les techniques de workflow

La notion de processus collaboratif ou workflow se définit par l'exécution automatisée d'un ensemble de tâches nécessaire à l'exécution d'une procédure d'entreprise ou processus métier tel que la réservation d'un billet d'avion ou la planification de congés au sein d'une entreprise, le « Workflow Managment Coalition » définit Le concept de Workflow comme suit: **[20]**

"Workflow is concerned with the automation of procedures where documents, information or tasks are passed between participants according to a defined set of rules to achieve, or contribute to, an overall business goal"

D'une certaine façon, un service composite est similaire à un processus métier. Un service composite inclut un ensemble de services atomiques ainsi que les contrôles et échanges de données entre ces services. De la même façon, un processus métier est composé d'un ensemble d'activités élémentaires structurées, ainsi que l'ordre d'exécution entre elles.

Dans les méthodes et les techniques de WorkFlow de composition, nous devrions distinguer la génération statique et dynamique de déroulement des opérations dans le processus de composition. L'approche statique signifie que le demandeur ou l'utilisateur devrait établir un modèle de processus abstrait avant que la planification de composition commence. Le modèle de processus abstrait inclut un ensemble de tâches et de leur dépendance de données. Chaque tâche contient une requête qui est employée pour rechercher le vrai service web atomique pour accomplir la tâche. Dans ce cas, seulement le choix et l'attache des services web atomiques est fait automatiquement par programme. La méthode statique le plus généralement utilisée est d'indiquer le modèle de processus sous un graphe. D'autre part, la composition dynamique crée le modèle de processus et choisit des services atomiques automatiquement. Ceci exige du demandeur d'indiquer plusieurs contraintes, y compris la dépendance entre services, la préférence de l'utilisateur et ainsi de suite figure 2.2.

L'orchestration et la chorégraphie sont des compositions de services web permettant de créer un processus métier (workflow).

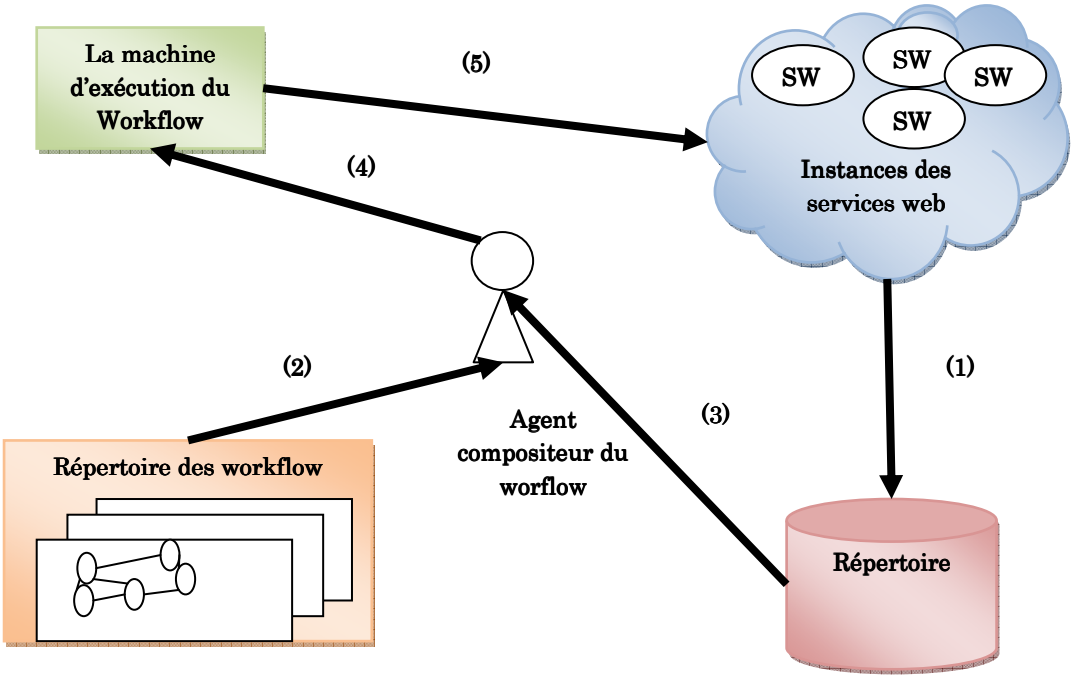


FIGURE 2.2 – Composition dynamique du workflow

Orchestration

L'orchestration décrit, du point de vue d'un service, les interactions de celui-ci ainsi que les étapes internes (ex. transformations de données, invocations à des modules internes) entre ses interactions. **[21]**

L'orchestration décrit l'interaction des services au niveau de messages, incluant le logique métier et l'ordre d'exécution des interactions. Les services web n'ont pas de connaissance (et n'ont pas besoin de l'avoir) d´être mêlées dans une composition et d'être partie d'un processus métier. Seulement le coordinateur de l'orchestration a besoin de cette connaissance.

La figure 2.3. Montre le workflow dans l'orchestration des services web. Un coordinateur prend le control de tous les services web impliqués et coordonne l'exécution des différentes opérations des services web qui participent dans le processus.

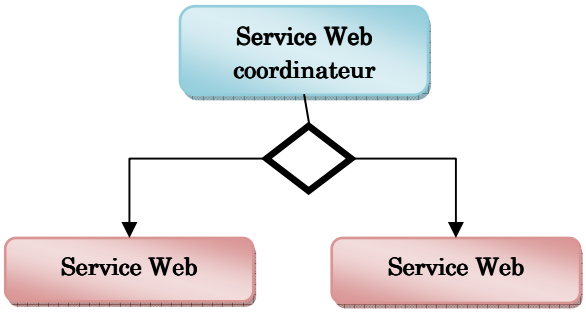


FIGURE 2.3 – L'orchestration des services Web

Chorégraphie

La chorégraphie décrit la collaboration entre une collection de services dont le but est d'atteindre un objectif donné. L'accomplissement de ce but commun se fait alors par des échanges ordonnés de messages. **[21]**

Contrairement à l'orchestration, la chorégraphie n'a pas un coordinateur central. Chaque service web mêlée dans la chorégraphie connaît exactement quand ses opérations doivent être exécutées et avec qui l'interaction doit avoir lieu. La chorégraphie est un effort de collaboration dans lequel chaque participant du processus décrit l'itération qui l'appartient. Elle trace la séquence des messages qui peut impliquer plusieurs Services Web.

Un modèle de chorégraphie n'est pas exécutable. Il constitue en fait un accord entre des parties, où celles-ci spécifient comment leur collaboration doit se dérouler. Cette convention peut être établie par une consultation commune ou conçue par un comité de standardisation. **[10], [21]**

La collaboration dans la chorégraphie des services web peut être représentée par la figure 2.4.

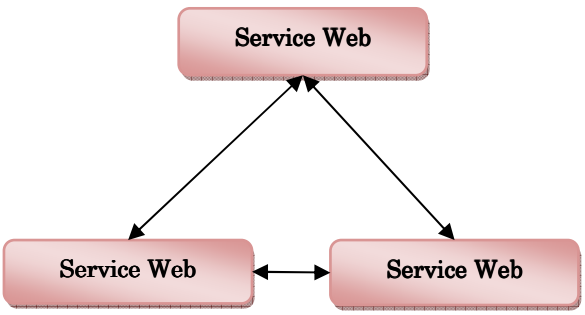


FIGURE 2.4 – La chorégraphie des services Web

L'orchestration et la chorégraphie sont toutes deux des manières de composition de service web agrégé. La différence principale entre ces deux

manières, est que dans la chorégraphie, chaque service connaît les autres services qui interagissent, alors que dans le cas de l'orchestration, le service qui joue le rôle de chef d'orchestre est le seul qui connaît les services en interaction.

Les principaux langages d'orchestration et/ou de chorégraphie de services web sont :

- XLANG - XML Business Process Language (Microsoft)
- BPML - Business Process Modeling Language (BPMI)
- WSFL - Web Service Flow Langage (IBM)
- WSCL - Web Service Conversation Language (Hewlett-Packard)
- WSCI - Web Service Choregraphy Interface (SUN)
- BPEL4WS - Business Process Execution Language for Web Services (IBM, Microsoft, BEA)

2.4.2.2 Technique de l'intelligence artificielle IA

Plusieurs recherches ont étés proposées pour résoudre le problème de composition automatique de services web via l'utilisation des techniques d'intelligence artificielle et plus précisément les méthodes de planification du l'intelligence artificielle (AI Planning).

La composition dynamique de services Web par des techniques d'intelligence artificielle, et plus particulièrement par des techniques de planification, est la voie qui semble la plus prometteuse. En effet, les concepts d'OWL-S sont très fortement inspirés de ceux de la planification : les prés conditions présentent les conditions qui doivent être satisfaites pour garantir la bonne exécution d'un service Web. Les effets sont le résultat du succès de l'exécution d'un service. Dans ce qui suit, nous présentons quelques axes de recherche actuels concernant la composition par planification et par d'autres techniques d'intelligence artificielle. En générale un problème de planification est décrit comme suit: [18]

Problème de cinq-uplets (S, S_0, G, A, T), tel que :

- S est l'ensemble de tous les états possible du système,
- $S_0 \subset S$ est l'état initial du système,
- $G \subset S$ est l'état final (Goal) pour lequel cet état est recherché par le système de planification.
- A est l'ensemble d'actions effectuées par le planificateur (Planner) permettant le changement entre les différents états du système et
- $T \subseteq S \times A \times S$ la relation (fonction) de translation qui définit les pré-conditions et effets pour l'exécution de chaque action.

Dans le terme de services web, S_0 et **G** sont l'état initial et l'état final (Goal) spécifiés dans les requêtes fait par l'utilisateur (Web service requesters). **A** est l'ensemble de services web disponibles. **T** dénote la fonction de changement d'état pour chaque service.

DAML-S (appelé OWL-S dans ces versions récentes) est le seul langage de service web qui mettre une connexion direct avec les méthodes de planification (AI planning). Le changement d'état est produit par l'exécution de service web qui est spécifié par les propriétés pré-conditions et effets du ServiceProfile dans DAML-S, les pré-conditions présentent les conditions logiques qui doivent être vérifiées à priori avant que le service concerné est exécuté et les effets sont les résultats de l'exécution de service web concerné.

Calcul des situation

McIlraith et al **[24]** proposent d'adapter et d'étendre le langage Golog pour la construction automatique de services Web. Golog est un langage de programmation logique qui permet de faire du calcul situationnel (i.e., langage logique qui sert `a représenter des changements ou évolutions en terme de situations, d'actions et d'objets). Le problème de la composition de services Web est abordé de la façon suivante : la requête de l'utilisateur et les contraintes des services sont représentées en termes de prédicats du premier ordre, les services sont transformés en actions (primitives ou complexes) dans le même langage, puis, à l'aide de règles de déduction et de contraintes, des modèles sont ainsi générés et sont instanciés à l'exécution à partir des préférences utilisateur.

Théorème prouvant

Rao et Al **[22]** ont introduit une méthode pour la composition automatique de services web sémantiques en utilisant la *Linear logic theorem proving.* La méthode utilise le langage de service web sémantique (DAML-S) pour la représentation externe de services web. Intérieurement, les services sont représentés par les axiomes de l'extralogical et les preuves dans la logique linéaire.

Waltinger **[23]** a élaboré une approche pour la composition de services par preuve de théorèmes. Cette approche est basée sur la déduction automatique et la synthèse de programmes. Les services disponibles et les requêtes utilisateur sont traduits dans un langage du premier ordre. Puis des preuves sont produites à partir d'un prouveur de théorèmes. La composition est obtenue `a partir de preuves particulières.

PDDL

PDDL (Planning Domain Definition Language) fait partie des langages reconnus et normalisés dans le domaine de planification, développé par la communauté model –based planning community comme un langage standard pour la planification des compétitions [25]. La communauté scientifique travaillant sur les planificateurs a très vite relevé la ressemblance existant entre DAML-S et PDDL dans leurs représentations. La planification des tâches dans PDDL est divisée en deux : *Domain file* pour les prédicats et les actions et *Problem file* pour les objets et l'état initial et but.

PDDL est destiné à exprimer la physique d'un domaine, c'est-à-dire quels sont les prédicats existants, quelles sont les actions possibles, quelle est la structure des compositions d'actions et quels sont les effets des actions.

La résolution d'un problème de composition automatique peut se faire par un mapping de la description DAML-S en PDDL, et utiliser un planificateur déjà existant.

HTN

HTN (Hierarchical Task Network)planning est une méthode de planification par décomposition de tâche. Un système basé sur HTN décompose itérativement la tâche désiré en sous tâches jusqu'à l'obtention des tâches atomiques qui peuvent être exécutés par des opérations atomiques. **[26]**

Dans **[27]**, Wu et al préconisent l'utilisation du planificateur SHOP2 pour la composition automatique de services Web à partir de leur description sémantique. SHOP2 est un planificateur HTN (Hierachical Task Network). D'après les auteurs, le principe de décomposition d'une tâche en sous-tâches dans la planification hiérarchique est très similaire au concept de décomposition de processus composites dans OWL-S.

Sirin et Parsia **[28]** sont allés plus loin en intégrant un raisonneur sur les langages de descriptions dans le planificateur SHOP2. La planification HTN est très puissante pour les domaines o`u la connaissance est complète et détaillée, ce qui n'est pas, d'après Klush et al, le cas avec les services Web.

Multi agents

Les services web partagent avec les systèmes multi agent plusieurs propriétés dont les méthodologies agents tiennent explicitement compte dans leur ébauche de solutions. De telles propriétés sont: **[19]**

- **La distribution** géographique et/ou logique d'entités, données ou informations hétérogènes ;
- **La complexité** du problème global à résoudre, celui-ci n'étant manipulable que par des stratégies heuristiques utilisant des données ou connaissances locales ;
- **La flexibilité** des interactions il n'y a pas d'affectation a priori des tâches et les mécanismes de résolution de problèmes ne sont pas préalablement assignés à telle ou telle autre entité ;
- **L'environnement dynamique** nécessitant, pour la résolution de problèmes, des entités réactives et adaptatives ;
- **L'ouverture** il n'est pas possible de donner une spécification complète du problème à résoudre ni de définir une fonction d'utilité globale.

	Agents	**Services Web**
Méthodes de communication	Messages ACL et KQML (asynchrone)	SOAP et RPC (synchrone)
Avantages	Autonomie Mobilité Comportement intelligent Comportement dynamique	Indépendances vis à vis des plateformes et des langages Basé sur XML Facilité de localisation
Inconvénients	Dépendance du conteneur Le non support de XML Invocable uniquement par un agent	Services statiques

Tableau 2.1-Un tableau comparative entre les agents et les services Web

CASCOM **[29]** son objectif principal est d'implémenter, tester et valider une infrastructure pour les applications business basées sur le web sémantique. Un

système multi agent est utilisé pour la sélection, l'ordonnancement et l'exécution des services.

Yasmin Cherif [19] propose dans sa thèse un modèle de coordination multi agent pour une chorégraphie dynamique des services. Dans ce modèle de coordination la collaboration de capacités du service est modelée par des agents introspectifs capables de raisonner sur leurs services et de coordonner avec les autres agents pour surmonter ses limitations et rependre au besoin de l'utilisateur.

2.5 La composition vue comme un problème de planification multi agents

Composer des services web est de trouver un ordre d'exécution de ces services, ordonner des web services est très similaire à un problème de planification automatique. La planification a en effet pour rôle de trouver une séquence d'actions, ou plan, pour, à partir d'un état initial, arriver à un état but, exprimé par l'utilisateur. Le principe général de la planification est le suivant : étant donné une situation initiale, une description des buts et un ensemble d'actions possibles alors trouver une séquence d'actions (un plan) qui transforme la situation initiale en une situation (finale) dans laquelle les conditions de but final sont valides, alors l'action de l'agent repose sur deux étapes : la première sert à trouver une séquence d'actions (plan) dont l'exécution permet d'achever le but désiré ou ce qu'on appelle la construction de plan ; alors que la deuxième étape consiste à exécuter ce plan.

La planification représente le raisonnement menant à une série d'actions pour compléter un but. Les actions représentent les opérations des services, l'entrée est le but du client et les services disponibles, la sortie est un service composite sous la forme d'un plan.

2.6 Conclusion

La composition des services web est un point crucial qui a un grand impact sur plusieurs domaines de recherches. Beaucoup d'efforts ont été fourni afin de permettre une composition utilisable et acceptable de services Web.

Nous avons vu qu'il existe différente catégories de composition des services web qui sont classées selon deux axes : Selon que la sélection des services et la gestion du flot soient faites ou non a priori, (***statique*** ou ***dynamique)***, ou selon le degré de participation de l'utilisateur (***manuelles, semi-automatiques*** ou ***automatiques)***.

Comme notre travail se situe autour de la composition automatique, Nous avons présentés un modèle général (general framework) pour la composition automatique de services web ce modèle contient cinq phases : phase de présentation et description, phase de translation, phase de génération, phase d'évaluation et la phase d'exécution.

Dans la composition automatique il existe deux grands courants de recherche. Certains proposent de composer les services en utilisant des techniques de génie logiciel : c'est l'orchestration et la chorégraphie. D'autres envisagent une approche plus dynamique : la composition de services en utilisant des techniques d'intelligence artificielle : calcul situationnel, système multi-agent ou encore la planification.

Comme en a vu dans ce chapitre, la composition automatique des services web ressemble à un problème de planification automatique, alors la composition des services web peut être réalisée par un système multi agent le chapitre 3 introduit la notion des systèmes multi agents.

Partie II

Etat de l'art de la planification distribuée à l'aide des systèmes multi agent

☞ *Chapitre 3*

La planification distribuée

3.1 Introduction

A la différence de l'Intelligence Artificielle classique (IA) qui modélise le comportement intelligent d'un seul agent, l'intelligence artificielle distribuée (IAD) s'intéresse à des comportements intelligents qui sont le produit de l'activité coopérative de plusieurs agents. Le passage du comportement individuel aux comportements collectifs est considéré non seulement comme une extension mais aussi comme un enrichissement de l'IA, d'où émergent de nouvelles propriétés et de nouveaux comportements.

Un des axes fondamentaux dans la recherche en IAD sont les systèmes multi-agents (SMA). **[33]**Il s'agit de faire coopérer un ensemble d'agents dotés d'un comportement intelligent et de coordonner leurs buts et leurs plans d'actions pour la résolution d'un problème.

Dans ce chapitre on va tout d'abord savoir qu'est ce qu'un agent, l'architecture, les types et le fonctionnement des agents, qu'est ce qu'un système multi agents et les avantages qu'il offre. Ensuite on entame le noyau de notre chapitre qui est la planification distribuée dans un système multi agents. Ensuite en présente les protocoles d'interaction les plus utilisés en détaillons le protocole des réseaux contractuels. En fin nous terminerons notre chapitre par quelque environnement du développement des systèmes multi agents

3.2 Système multi agent SMA

Le thème des systèmes multi-agents (SMA), s'il n'est pas récent, est actuellement un champ de recherche très actif. C'est une discipline qui s'intéresse aux comportements collectifs produits par les interactions de plusieurs entités autonomes et flexibles appelées **agents,** que ces interactions tournant autour de la coopération, de la concurrence ou de la coexistence entre ces agents.

3.2.1 Le concept d'agent

Le concept d'agent a été l'objet d'études pour plusieurs décennies dans différentes disciplines. Il a été non seulement utilisé dans les systèmes à base de connaissances, la robotique, le langage naturel et d'autres domaines de l'intelligence artificielle, mais aussi dans des disciplines comme la philosophie et la psychologie.

3.2.1.1 Définition

Dans la littérature, on trouve une multitude de définitions d'agents. Elles se ressemblent toutes, mais diffèrent selon le type d'application pour laquelle est conçu l'agent, voici les premières définitions de l'agent dû à Ferber : **[30]**

«Un agent une entité informatique qui se trouve dans un système informatique ouvert , qui peut communiquer avec d'autres agents, est mue par un ensemble d'objectifs propres, possède des ressources propres, ne dispose que d'une représentation partielle des autres agents, possède des compétences (services) qu'elle peut offrir aux autres agents, a un comportement tendant à satisfaire ses objectifs, en tenant compte d'une part des ressources et des compétences dont elle dispose, et d'autre part de ses propres représentations et des communications qu'elle reçoit.. »

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

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

- **L'autonomie** : L'agent travaille sans intervention directe, jusqu'à un point défini par l'utilisateur. L'autonomie d'un agent peut aller du simple lancement d'une sauvegarde, à la négociation du prix d'un produit choisi par son mandataire. 'Le moteur d'un agent est lui-même'. **[30]**
- **La réactivité** : L'agent percevra son environnement (qui peut être l'utilisateur au travers d'une interface graphique, d'un ensemble d'agents,…) en répondant dans les temps impartis aux changements qui surviennent sur cet environnement [31].
- **L'interactivité** : L'agent doit pouvoir exercer des actions sur son environnement et réciproquement. L'interactivité d'un Agent de sauvegarde sera sa possibilité de sauvegarder un certain nombre de fichiers par une action exercée au travers du système d'exploitation. Ce dernier pourra informer l'Agent d'une permission non accordée (exemple : un fichier dont l'utilisateur n'a pas le droit de lecture). **[31]**
- **La flexibilité** : qui se résume par la capacité de répondre à temps, d'être proactif et en particulier Social ; l'agent doit être capable d'interagir avec les

autres agents quand la situation l'exige afin de compléter ses tâches ou aider ces agents à les accomplir. **[30]**

- **L'adaptabilité** : Un agent adaptatif est un agent capable de contrôler ses aptitudes (communicationnelles, comportementales, etc.) selon l'agent avec qui il interagit. Un agent adaptatif est un agent d'un haut niveau de flexibilité. **[30]**
- **Social :** Capacité à interagir pour atteindre ces buts pour aider d'autres agents dans leurs activités.

3.2.1.2 Architecture d'un agent

La figure 3.1 décrit l'architecture globale d'un agent. **[32]** C'est une synthèse des architectures d'agents décrites dans la littérature

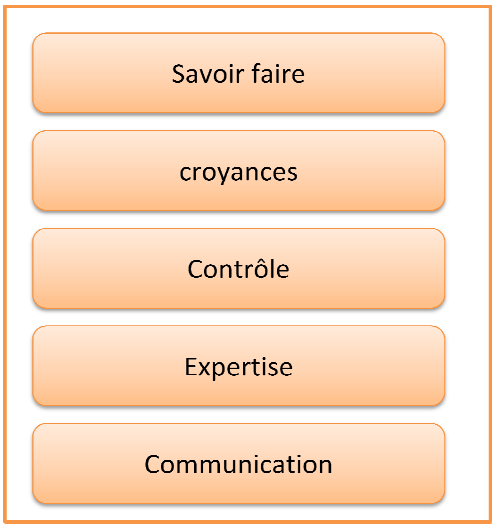


FIGURE 3.1 – L'architecture d'un agent

- **Savoir-faire** : Le savoir-faire est une interface permettant la déclaration des connaissances et des compétences de l'agent. Il permet la sélection des agents à solliciter pour une tâche donnée. Il n'est pas nécessaire mais il est très utile pour améliorer les performances du système, quel que soit le mode de coopération utilisé.

- **Croyances** : Dans un univers multi-agents, chaque agent possède des connaissances sur lui-même et sur les autres. Ces connaissances ne sont nécessairement objectives, on parle alors de croyances d'un agent.

- **Contrôle** : La connaissance de contrôle dans un agent est représentée par les buts, les intentions, les plans et les tâches qu'il possède.

- **Expertise** : C'est la connaissance sur la résolution de problème. Pour un système expert utilisant le formalisme de règle, par exemple, cette connaissance correspond à sa base de règles.
- **Communication** : L'agent doit posséder un protocole de communication lui permettant d'interagir avec les autres agents pour une bonne Coopération et bonne coordination d'actions.

3.2.1.3 Fonctionnement d'un agent

La structure générale d'un agent, avec les composantes principales et les fonctions ainsi définies, sont représentées dans la figure3.2. **[30]** Les ellipses représentent les processus mis en œuvre lors du fonctionnement de l'agent.

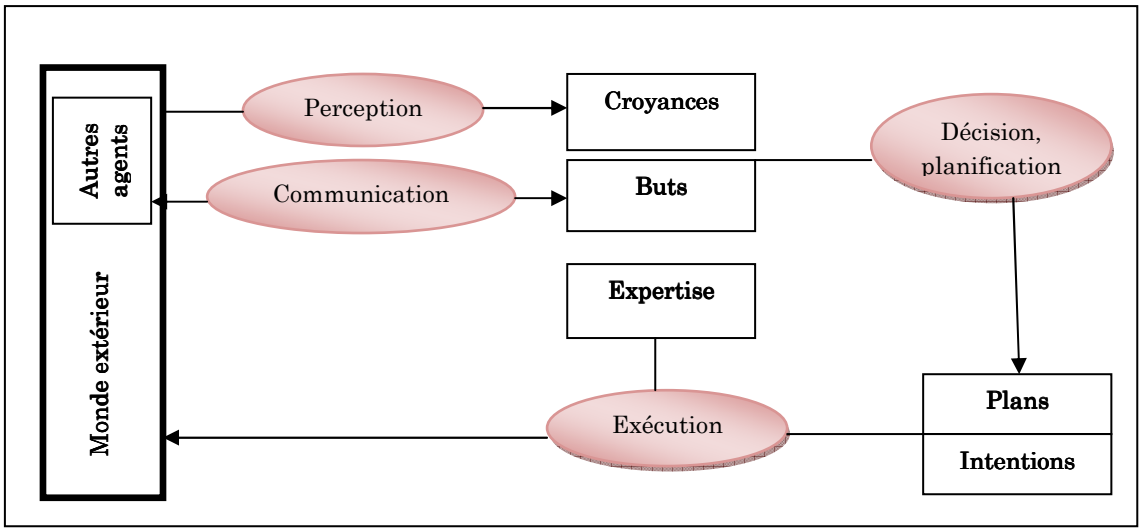


FIGURE 3.2 – Les composants principales d'un agent

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

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

3.2.1.4 Types d'agent

- **Agents réactifs vs agents cognitifs :** La granularité[18] des agents impliqués dans une application varie selon deux écoles coexistantes aujourd'hui : l'école cognitive et l'école réactive. Suivant le type d'agent utilisé, on parlera de systèmes cognitifs ou de systèmes réactifs.

- **Les agents réactifs :** n'ont de représentation ni d'eux ni de leur environnement. Leur comportement est régi par des réponses immédiates à des *stimuli* fournis par l'environnement et par leurs buts internes. Ces agents sont les plut simples à réaliser et permettent notamment d'observer l'émergence de comportement collectifs.

- **Les agents cognitifs :** possèdent une représentation, plus ou moins limitée, d'eux-mêmes et de leur environnement. Ils sont souvent capables de planifier leurs actions et permettent de simuler des comportements plus complexes. Souvent, les agents ne sont ni purement réactifs ni purement cognitifs, mais situés à un niveau intermédiaire entre ces deux extrêmes. Les systèmes d'agents cognitifs sont fondés sur la coopération d'agents capables à eux seuls d'effectuer des opérations complexes. Les agents d'un système cognitif sont des entités intelligentes, par contre les systèmes réactifs sont basés sur l'émergence de comportement intelligent à partir de l'interaction de nombreux comportements élémentaires (simples), il n'est pas nécessaire que les agents soient intelligents individuellement pour que le système ait un comportement global intelligent. **[33]**

Systèmes d'agents cognitifs	Système d'agents réactifs
• Représentation explicite de l'environnement • Peut tenir compte de son passé • Agents complexes • Petit nombre d'agents	• Pas de représentation explicite • Pas de mémoire de son histoire • Fonctionnement stimulus/réponse • Grand nombre d'agents

Tableau3.1 – Les agents cognitifs vs réactifs

[18] On entend par granularité le degré de détail des connaissances de l'agent. La granularité exprime la complexité des fonctionnalités d'un agent.

- **Agents hybrides :** Les agents hybrides encapsulent dans une seule entité les deux classes d'agents. Cette classe d'agents est motivée dans le cas d'applications où la combinaison de philosophies d'agents apporte plus d'avantages et moins d'inconvénients.

3.2.2 Système multi agent SMA

Les systèmes multi agent ne sont pas qu'un simple groupe d'agents dans un environnement commun, mais une réelle organisation avec des règles sociales et des interactions permettant la coopération et la collaboration pour la résolution de problèmes que les systèmes centralisés ne pourraient pas résoudre.

Plus précisément, on appelle un système multi-agents, un système composé d'un **environnement** c'est-à-dire un espace disposant généralement d'une métrique, un ensemble d'**objets** situés dans l'espace, ils sont passifs, ils peuvent être perçus, détruits, créés et modifiés par les agents, un ensemble d'**agents** qui sont l'entité un ensemble de **relation**s actives du système qui unissent les objets entre eux, un ensemble d'**opérations** permettant aux agents de percevoir, de détruire, de créer, de transformer et de manipuler les objets et un ensemble d'opérateurs chargés de représenter l'application de ces opérations et la réaction du monde à cette tentative de modification (les lois de l'univers). **[30]**

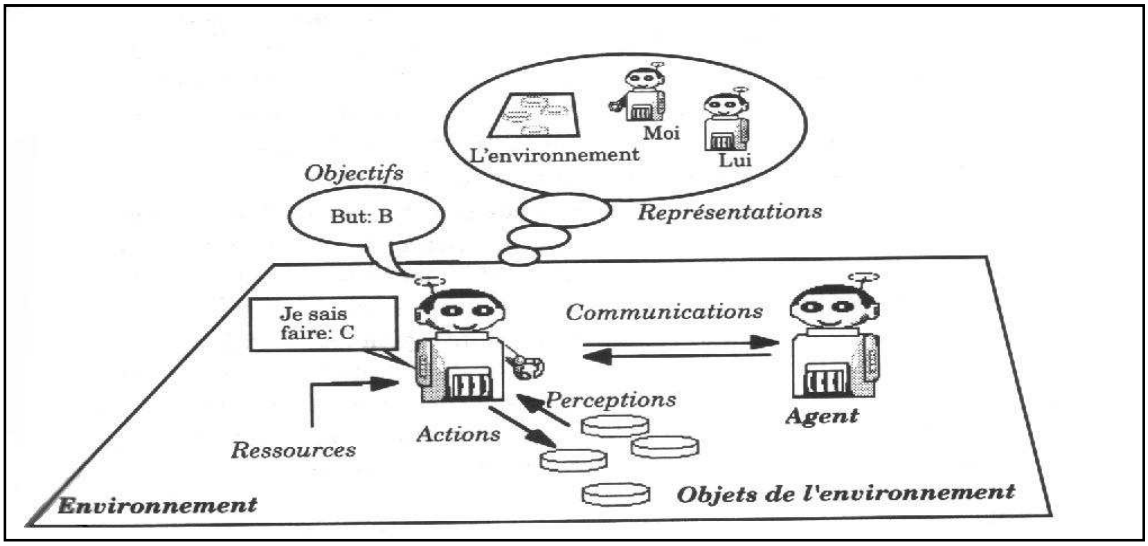


FIGURE 3.3 – Système multi agents

3.3 La planification distribuée dans un environnement multi agents

La planification multi-agent est une extension du domaine de la planification en intelligence artificielle aux systèmes multi-agent. En planification classique on cherche, par exemple, à contrôler les mouvements d'un robot. Alors qu'en planification multi agent on s'intéresse à la coordination des mouvements entre chaque robot : la planification est distribuée.

Chaib draa dans **[33]** définit la planification comme suit :

« Planifier c'est concevoir une série d'actions liées les unes aux autres en vue de réaliser un but. »

Le principe général de la planification est le suivant : étant donné une situation initiale, une description des buts et un ensemble d'actions possibles alors trouver une séquence d'actions (un plan) qui transforme la situation initiale en une situation (finale) dans laquelle les conditions de but final sont valides, alors l'action de l'agent repose sur deux étapes : la première sert à trouver une séquence d'actions (plan) dont l'exécution permet d'achever le but désiré ou ce qu'on appelle la construction de plan ; alors que la deuxième étape consiste à exécuter ce plan. La coordination d'action basant sur la planification revient à une décomposition en trois étapes : la construction des plans, la synchronisation/la coordination des plans et l'exécution des plans. Dans le cadre de la planification collective, dans un contexte multi-agent, l'objectif est d'avoir des agents capables de collaborer pour réaliser un but commun.

Dans la planification multi-agents, un plan multi-agents (ou plusieurs plans individuels équivalents à un seul qui soit cohérent) est créé en analysant la manière dont les actions des agents doivent être organisées dans le temps et l'espace de manière à réaliser les objectifs. Le plan généré sera exécuté par plusieurs agents dans un environnement commun. Donc, la planification contribue à la coordination, dans la mesure où lorsque les agents adoptent un plan "bien fait", ils agissent de manière coordonnée. La coordination est une problématique centrale des systèmes multi-agent dans la mesure où les agents doivent partager un même environnement et sont dans l'obligation de mettre en commun leurs compétences pour réaliser les objectifs fixés.

Le terme de planification distribuée est toutefois ambigu car il n'explicite pas ce qui est distribué. En effet, les plans peuvent être construits de manière centralisée puis distribués aux agents, ou bien chaque agent peut construire localement son propre plan puis le coordonner de manière distribuée. Dans le premier cas, seule l'exécution du plan est distribuée. En revanche, dans le second, la synthèse de plans, le processus de coordination ainsi que l'exécution sont réalisés de manière complètement distribuée. Une classification des planifications est donnée ci après. **[36]**

- **Planification multi-agent centralisée et plans distribués :** (Un planificateur, plusieurs exécutants) La coordination par planification centralisée repose toujours sur l'existence d'un agent coordinateur. Cet agent centralise l'ensemble des plans des agents du système et résout les conflits potentiels entre leurs activités en introduisant des actions de synchronisation. L'agent coordinateur peut : soit planifier pour l'ensemble des agents et, dans ce cas, il doit décomposer le plan global en sous-plans synchronisés pouvant ˆêtre exécutés par les agents ; soit chaque agent peut planifier localement et, dans ce cas, le rôle de l'agent coordinateur se limite à la synchronisation des plans reçus [37]. L'avantage de cette planification est quelle est Intéressant pour la résolution de conflits et la convergence vers une solution globale mais désavantages liés à la centralisation du contrôle : goulet d'étranglement, charge de communication et défaillance.

- **Planification multi-agent distribuée**

  Dans une approche de planification distribuée, les activités de planification sont réparties au sein d'un groupe d'agents. Cette approche est utilisée quand un seul agent ne peut pas avoir une vue globale des activités du groupe. Il existe deux approches : **[40], [41]**

  - ✓ **Planification distribuée avec plan centralisé**

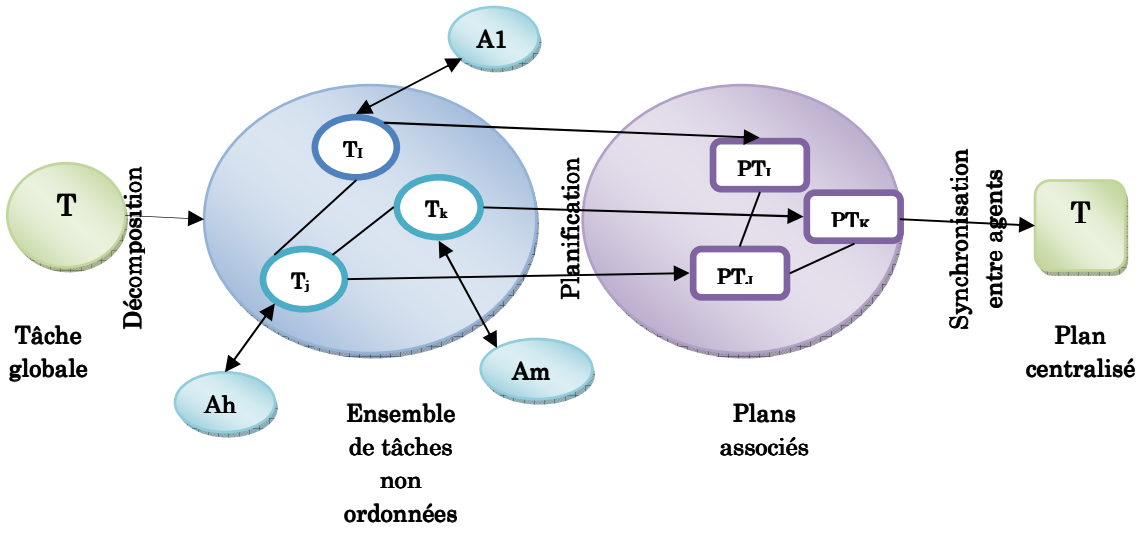


FIGURE 3.4 – Planification distribuée avec plan centralisé

Le processus de planification est distribué entre différents agents qui coopèrent et communiquent en partageant des objectifs et des représentations dont l'objectif principale est de construire un plan cohérent. Chaque agent génère sa portion de sous plans, les plans partiels

des agents sont ensuite synchronisées en un seul plan (partage de résultats au préalable). Le problème dans cette approche est la possibilité d'incompatibilité des buts et des intentions, de connaissance incohérentes, de représentation des plans partiels différente d'un agent à l'autre,…

- ✓ **Planification distribuée avec plan distribué**
  Chaque agent produit un plan indépendamment des autres ou en étant dirigé par un but commun avec les autres, les solutions possibles dans cette approche sont : **[41]** La fusion de plans, Planification itérative, Planification négociée.

3.4 Protocoles d'interactions

Un système multi agents est un ensemble organisé d'agents interactifs. L'interaction est ainsi un concept clé des systèmes multi agents. Pour définir les interactions entre agent, différents protocoles d'interaction ont été proposés. Les protocoles d'interaction sont des descriptions de patterns standards d'interaction entre deux ou plusieurs agents. La majorité des traveaus existants, notamment les traveaux de la FIPA, décrit en effet les séquences de messages échangés et leurs contenus(performative, émetteur, receveur…). Voici une présentation des protocoles les plus utilisés.

3.4.1 Les enchères

L'initiateur veut vendre un objet au plus grand prix, les participants veulent l'acheter au plus petit prix. Un protocole centralisé inclut un initiateur et plusieurs participants. L'initiateur annonce un objet pour la vente, les participants font des offres. Cela peut être fait plusieurs fois en fonction du type d'enchère. L'initiateur choisi le gagnant. On a trois Types d'enchère

- Enchère avec valeur privée : la valeur d'un agent pour un objet dépend seulement de ces préférences privées
- Enchère avec valeur commune : la valeur de l'objet dépend de l'évaluation des autres
- Enchère avec valeur corrélée : la valeur de l'objet dépend des évaluations internes et externes

Il existe plusieurs protocoles d'enchère : enchère anglaise, Enchère premier-prix offre-cachée, Enchère hollandaise…

3.4.2 Négociation basée sur des heuristiques

Des modèles de négociation informels, produit une solution bonne mais pas optimale, il y' a pas de médiateur central les actes de paroles sont privés entre les agents négociants, son problème principal est la prise des décisions heuristique pendant la négociation.

Un ***objet de négociation*** (**NO**) est l'intervalle des issues sur lesquelles un accord doit être trouvé. L'objet de la négociation peut être une action que l'agent négociateur *A* demande à un autre agent *B* d'effectuer, un service que l'agent *A* demande à *B*, ou l'offre d'un service qu'A veut effectuer pour *B* si *B* accepte les conditions de *A*.

3.4.3 Les réseaux contractuels (Contractnet)

Le protocole *Contract-Net* [39] est le protocole d'interaction le plus utilisés et le plus complets dans les systèmes multi agents. Il repose sue un mécanisme d'allocation de tâches régi par le protocole d'appel d'offres qui est utilisé dans les organisations humaines. Le modèle est basé sur une communication par envoi de messages.

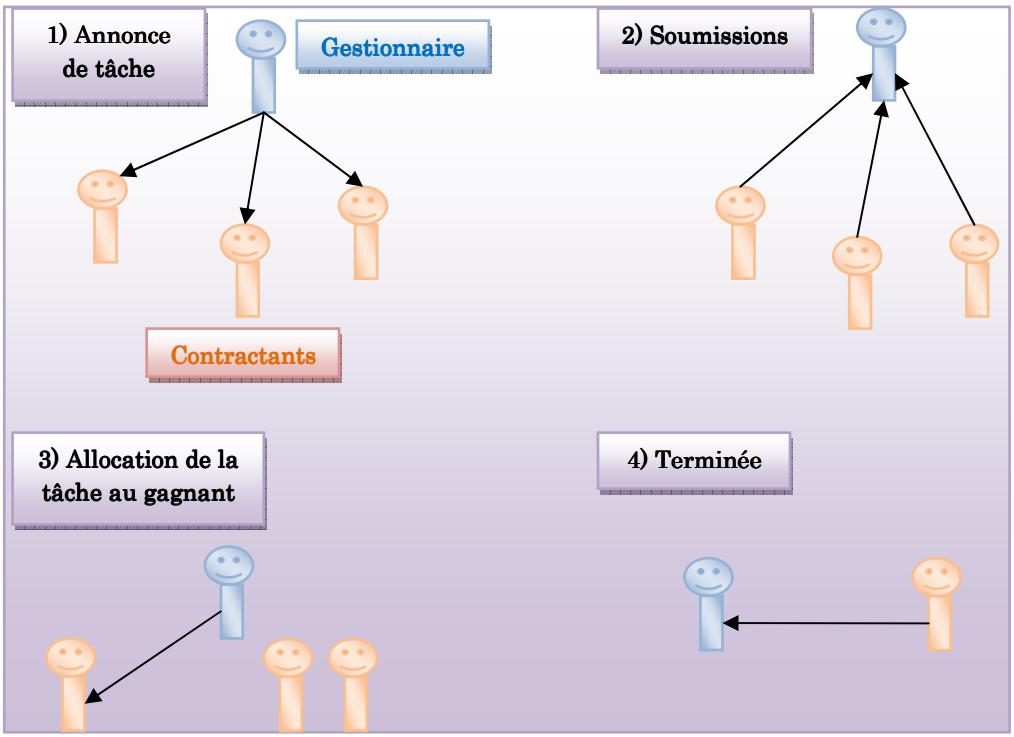


FIGURE 3.5 – Le protocole des réseaux contractuels

Dans ce protocole, les agents peuvent prendre deux rôles: *gestionnaire* ou *contractant.* L'agent qui doit exécuter une tâche donnée (le gestionnaire) commence tout d'abord par décomposer cette tâche en plusieurs sous-tâches. Il doit ensuite annoncer les différentes sous-tâches au reste des agents de l'environnement. Les agents qui reçoivent une annonce de tâche à accomplir

peuvent ensuite faire une proposition devant refléter leur capacité à remplir cette tâche. Le gestionnaire rassemble ensuite toutes les propositions qu'il a reçues et alloue la tâche à l'agent ayant fait la meilleure proposition. **[32]**

Voici l'algorithme du *Contract Net* : **[40]**

Étant donné une tâche, un gestionnaire, un groupe de (n - 1) soumissionnaires :

1. Le gestionnaire envoie un message de type « *annonce-tâche* » à un groupe d'agents (ou fait un broadcast).
2. Chaque agent évalue la tâche annoncée à l'aide d'une fonction locale « *évalue_annonce* »
3. L'évaluation précédente permet à certains agents de soumettre une proposition à l'aide d'une « *soumission-tâche* » au gestionnaire.
4. Si une proposition est jugée satisfaisante (à l'aide de la fonction « *évalue_soumission »*), alors le gestionnaire envoie un message de type « *acceptation* » à celui dont la proposition est retenue. Il envoie également un message de type « *refus*» aux autres agents dont les propositions n'ont pas été retenues.
5. Le gestionnaire peut mettre fin à la période d'acceptation de proposition si le temps d'expiration est dépassé.
6. Si aucune proposition n'a été retenue, alors le gestionnaire fait parvenir à tous les agents non retenus un message de type « *refus* » pour indiquer le rejet de chacune des propositions. Il peut alors se retirer de la négociation, retenir la proposition la plus acceptable, redémarrer un nouvel appel d'offre (nouveau *« annonce-tâche »*) ou prolonger le temps d'expiration de la période d'acceptation de proposition.
7. L'agent ayant obtenu le contrat, remet un rapport d'exécution lorsque la tâche est complétée.

ContractNet étendu

Le modèle Contract Net étendu **[40]** est basé sur une décomposition décentralisée de tâches. L'agent gestionnaire fractionne une tâche en plusieurs sous-tâches et les annonce à un agent ou à un groupe d'agents soumissionnaires. Mais contrairement au Contract Net original où une tâche ne peut être qu'accordée ou rejetée, dans le "Contract Net" étendu, une tâche peut être accordée temporairement, rejetée temporairement, accordée définitivement et rejetée définitivement par les agents.

Dans le Contract Net étendu, L'agent gestionnaire annonce une commande un groupe d'agents. Il sectionne le meilleur agent soumissionnaire à qui il envoie un accord temporaire. Tous les autres agents reçoivent un rejet temporaire. Si la meilleure soumission ne couvre pas la totalité de la commande, la partie restante de la commande est ré annoncée par le gestionnaire jusqu'à couvrir sa totalité. Au terme de ce processus les accords et des rejets temporaires sont transformés par des accords et des rejets définitifs. Quand un agent reçoit un accord temporaire, il crée une copie de son ancien plan, ainsi il sera en mesure de le récupérer en cas de rejet définitif.

3.5 Communication entre agents

En communiquant, les agents peuvent échanger des informations et coordonner leurs activités. Dans les SMA deux stratégies principales ont été utilisées pour supporter la communication entre agents: les agents peuvent échanger des messages directement ou ils peuvent accéder à une base de données partagées (appelée tableau noir ou "blackboard") dans laquelle les informations sont postées. Les communications sont à la base des interactions et de l'organisation sociale d'un SMA. **[34]**

3.5.1 Modes de communication

3.5.1.1 Communication par partage d'informations

Les composants ne sont pas en liaison directe mais communiquent via une structure de données partagée, où on trouve les connaissances relatives à la résolution (état courant du problème) qui évolue durant le processus d'exécution. Cette manière de communiquer est l'une des plus utilisées dans la conception des systèmes multi-experts. L'exemple parfait d'utilisation de ce mode de communication est l'architecture de blackboard.

3.5.1.2 Échange d'informations grâce à un tableau noir

En intelligence artificielle la technique du tableau noir ("blackboard") est très utilisée pour spécifier une mémoire partagée par divers systèmes. Dans un SMA utilisant un tableau noir, les agents peuvent écrire des messages, insérer des résultats partiels de leurs calculs et obtenir de l'information. Le tableau noir est en général partitionné en plusieurs niveaux qui sont spécifiques à l'application.

3.5.1.3 Les langages de communication

L'intérêt des langages de communication est de faciliter l'échange et l'interprétation des messages et l'interopérabilité entre les agents. Ces langages se focalisent essentiellement sur la manière de décrire exhaustivement des actes de communication d'un point de vue syntaxique et sémantique. **[43]** Les langages de communication entre les agents doivent satisfaire certaines propriétés :

- Une forme déclarative,
- Une syntaxe simple et lisible,
- Faire la distinction entre le langage de communication et le langage de contenu. Le langage de communication décrit les actes de langage et le langage du contenu permet d'exprimer les informations contenues dans le message,
- Avoir une sémantique du langage ayant un fondement théorique et une description formelle,
- Avoir une implémentation efficace, en vitesse et en bande passante.

Le premier langage qui a été introduit est KQML[19]. A l'origine, KQML a été développé pour échanger des informations et des connaissances entre des systèmes à base de connaissances. Il a été ensuite repris dans pour décrire les messages échangés entre les agents. Le deuxième langage dit FIPA-ACL [20] est proposé dans le cadre d'un travail de standardisation mené au sein l'organisation c'est une extension du langage KQML.

FIPA-ACL est fondé sur vingt et un actes communicatifs, exprimés par des performatifs, qui peuvent être groupés selon leurs fonctionnalités de la façon suivante :

- Passage d'information : *Inform, Inform-if, Inform-ref, Confirm, Disconfirm,*
- Réquisition d'information : *Query-if, Query-ref, Subscribe,*
- Négociation : *Accept-proposal, Cfp, Propose, Reject-proposal,*
- Distribution de tâches (ou exécution d'une action) : *Request, Request-when, Requestwhenever, Agree, Cancel, Refuse,*
- Manipulation des erreurs : *Failure, Not-understood.*

Les langages de communication, sont pleinement utilisés pour la spécification des protocoles d'interaction. Les protocoles d'interaction sont utilisés par les agents pour régir leurs interactions. **[43]**

3.6 Environnements du développement des systèmes multi agents

[19] Knowledge Query and Manipulation Language
[20] FIPA Agent Communication Language

Afin de réaliser une opérationnalisé plus accessible des systèmes multi-agents, des travaux ont tenté de réutiliser des architectures et des langages existants pour construire des environnements de développement de ces systèmes. Les environnements de développement ou les plates-formes multi-agents sont nécessaires pour renforcer le succès de la technologie multi agents. Les plates-formes multi agents permettent aux développeurs de concevoir et réaliser leurs applications sans perdre de temps à réaliser des fonctions de base pour la et l'interaction entre agents et éliminent, dans la plupart des cas, la nécessité d'être familier avec les différents concepts théoriques des systèmes multi agents.

Parmi les plates-formes fournies comme logiciels libres, il y quelques plates-formes plus connues pour avoir été utilisées dans le développement de plusieurs applications :

- **Swarm :** est une plate-forme générique, constituée d'une bibliothèque logicielle (en Objective C, avec une interface Java) qui permet de développer des simulations à base d'agents.
- **MadKit :** écrit en JAVA; Permet l'utilisation de différents langages de communication (KQML, XML ...); C'est une plate-forme générique sans spécification concernant le comportement des agents, mais proposant un modèle conceptuel d'organisation d'agents. Basé sur 3 concepts principaux: agents, groups, et rôles.
- **Zeus :** pour les agents cognitifs, Ses concepts s'appuient sur les notions d'agents ; de buts ; de tâches (que doit réaliser l'agent pour atteindre son but); et de faits (ce que l'considère comme vrai).
- **Jade :** (**J**ava **A**gent **D**évE**E**loppement framework) est un logiciel distribué par TILab en Open Source avec une licence LGPL (Lesser GPL). Jade a pour but de simplifier le développement des systèmes multi agents tout en fournissant un ensemble complet de services et d'agents conformes aux spécifications FIPA, dans un environnement java. C'est pour ces raisons notre choix s'est porté sur cette plateforme.

3.7 Conclusion

Les agents et systèmes multi agents ont beaucoup été étudiés dans la littérature, ils ne sont pas un simple groupe d'agents dans un environnement commun, mais une réelle organisation avec des règles sociales et des interactions permettant la coopération et la collaboration pour la résolution de problèmes que les systèmes centralisés ne pourraient pas résoudre.

Les systèmes multi agents semblent être particulièrement bien adaptés à la modélisation des problématiques (de distribution, d'ouverture, de flexibilité et de collaboration des services) sous-tendues par la composition dynamique de services. C'est la raison pour laquelle l'objet de notre étude va porter sur l'étude de solutions SMA pour la modélisation des problématiques de composition des services.

Chapitre4.

Partie III

Un Système multi agent pour la composition des services Web (MAS4WSC)

☞ *Chapitre 4*

La conception du système

4.1 Introduction

L'étude des travaux existants a montré qu'à ce jour différentes dimensions de composition des services web font l'objet d'étude. Cependant, ce problème de composition automatique de services Web est par nature très difficile. En effet les données sont instables et le Web étant dynamique.

Dans le chapitre 2, nous avons étudié les différentes techniques de composition de services Web qui ont été proposées à ce jour. Les solutions proposées peuvent être classifiées selon deux axes :

- ✓ Selon que la sélection des services et la gestion du flot soient faites ou non a priori, une approche sera dite ***statique*** (orchestration et chorégraphie) qui utilise des canevas prédéfinis ; ou ***dynamique***. qui permet de définir un enchaînement de services Web en réponse à un objectif précis fixé par l'utilisateur.

- ✓ En fonction du degré de participation de l'utilisateur dans la définition du schéma de composition, ces propositions *sont* ***manuelles, semi-automatiques*** ou ***automatiques***.

La composition automatique des services web a pour but la réduction de la complexité, elle est essentiel car elle permet d'intégrer les facteurs aussi important que le passage en échelle, la réduction des coûts, la prise en compte les critères de qualités…

Les travaux existants sur la composition dynamique et automatique de services Web ont mis en avant le fait que la description syntaxique (WSDL) des services n'est pas suffisante et ont proposé des solutions basées sur une description sémantique (OWL-S). Les services Web sont considérés sous l'angle de leurs IOPE (Inputs, Outputs, Preconditions, Effects), qui permettrons d'utiliser les capacités sémantiques des services notés en vue, de les mettre en correspondance avec les concepts de la requête de l'utilisateur.

Parmi les techniques d'intelligence artificielle qui ont été appliquées à la composition de services, la planification classique est une solution prometteuse du fait de la correspondance entre la notion de processus atomique et composite

d'OWL-S et la notion d'opérateur et de méthode de la planification, ainsi que la possibilité de spécifier un service Web par ses pré conditions et ses effets. Cependant, la composition par planification classique ne traite pas l'aspect distribué des services Web et a donc ses limites. Dans la planification distribuée, le domaine de planification est réparti sur l'ensemble des agents. Chaque agent est capable de réaliser un certain nombre d'actions, c'est la mise en commun et l'ordonnancement des compétences de chaque agent, dans le but de résoudre un problème donné.

Notre principale contribution est de proposer un système multi agents pour la composition dynamique et automatique des services web sémantiques (MAS4SWSC), où pour chaque service est associé un agent. Les agents doivent coordonner leurs compétences afin de proposer un plan de composition. Le protocole de coordination utilisé dans notre système est le protocole des réseaux contractuel (contract net protocole) c-à-dire que les agents échangent leurs propositions sous forme d'appel d'offre. Les agents peuvent trouver plusieurs plans de composition qui peuvent satisfaire la requête utilisateur, alors le meilleur plan doit être choisi et retourner à l'utilisateur. Le plan optimal est caractérisé par une qualité de service élevée.

Dans ce chapitre en va tous d'abord donner la définition du problème le principe général de notre contribution ensuite l'architecture générale de notre système, suivi par l'architecture détaille du système ainsi que l'architecture interne de chaque agent.

4.2 Architecture du système

4.2.1 Présentation générale du système proposé

La composition des services Web a pour but de produire une description spécifiant une séquence d'appels à des services ainsi que la façon dont ces services sont liés entre eux dans le but de résoudre un objectif donné. Elle nécessite la réalisation des étapes suivantes:

- ✓ **La découverte des services Web** pouvant répondre aux besoins de la composition.
- ✓ **L'organisation des interactions** entre les services Web, la composition est effectuée en utilisant la description (syntaxique ou sémantique) des services sélectionnés ;
- ✓ **L'exécution de la composition** est l'étape d'invocation effective des services Web participants à une composition.

Dans notre étude nous ne traitons pas le premier point de l'algorithme de composition, i.e., trouver l'ordre d'exécution des services Web présélectionnés à partir de leur description sémantique pour réaliser un but fixé par l'utilisateur puis exécuté cette composition.

L'architecture générale du système est illustrée dans la figure4.1. Les services web sont enregistré dans une base de données du système, une fois le système est lancé, l'agent initiateur initialise chaque agent avec une description sémantique d'un service web. Lorsque l'utilisateur envoi sa requête au système via une interface, le processus de composition génère plusieurs plans de composition, le module d'évaluation choisi le meilleur plan selon certain critère et le résultat ensuite est retourné à l'utilisateur.

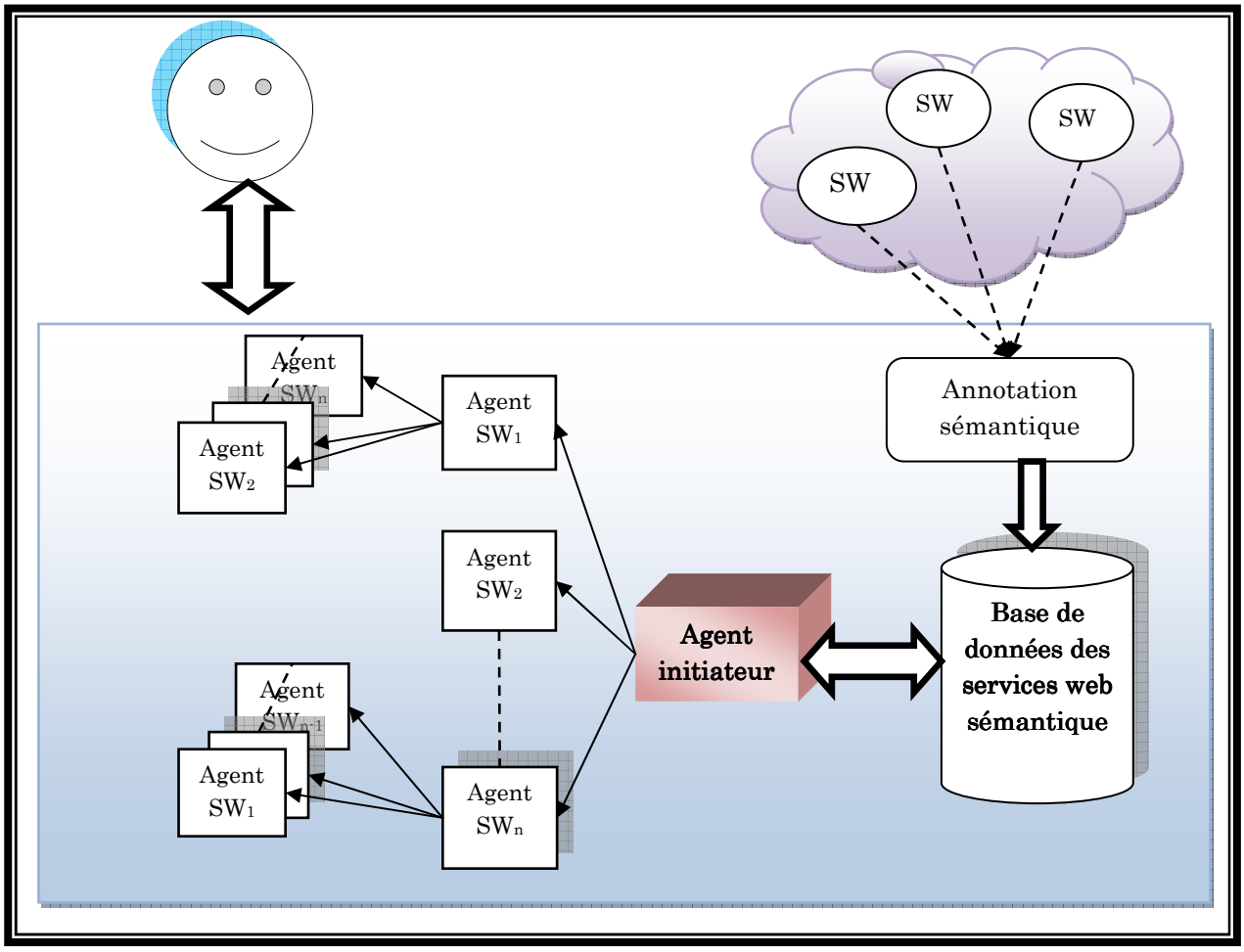


FIGURE 4.1 – Système de composition des services web sémantique

Afin de permettre une prise en charge automatique de la composition, il est essentiel de définir formellement la sémantique des services et des données qu'ils échangent par le biais d'ontologies. En ce qui concerne les processus décrits par

les services, OWL-S [12] une ontologie basée sur OWL permet de décrire ces services en terme de flux de contrôle, c'est à dire l'ordre dans lequel ils sont exécutés ainsi qu'en terme de flux de données, c'est à dire comment leurs entrées et leurs sorties sont liées. D'autre part les préconditions, permettant l'exécution d'un service si son environnement est dans un certain état, et les effets décrivant les changements qu'apporte un service à l'état de l'environnement, y sont aussi décrites.

On a choisi le langage OWL-S car il est plus adéquat aux problèmes d'interopérabilité et la composition des services web par sa description et sa représentation aux services web.

Une description OWL-S est composée de :

- ServiceProfile: contient les informations générales comme le nom du service, la description, Inputs et Outputs
- ServiceModel: décrit comment le service fonctionne
- ServiceGrounding: indique comment chercher le service

Parmi ces trois module d'information d'OWL-S on a utilisé que deux module le service profile (Inputs,Outputs) et service grounding (le lien au service) car ces deux nous aident plus dans la recherche et la composition des services web.

Nous proposons dans le système, une base de données contenant des services web annotés. Chaque service lui sera associé un agent SW lors de l'initialisation du système

Le processus de composition et d'évaluation est un système multi agents (SMA) centré utilisateur dans lequel les agents sont les représentants des services et sont chargés de coordonner leurs compétences afin de réaliser une tâche soumise par l'utilisateur. Il s'appuie sur le protocole des réseaux contractuels (contract net protocol) c-à-dire que les agents échangent leurs propositions sous forme d'appel d'offre. Dans ce protocole, les agents peuvent prendre deux rôles: *gestionnaire* ou *contractant.* L'agent qui doit exécuter une tâche donnée (le gestionnaire) commence tout d'abord par décomposer cette tâche en plusieurs sous-tâches. Il doit ensuite annoncer les différentes sous-tâches au reste des agents de l'environnement. Les agents qui reçoivent une annonce de tâche à accomplir peuvent ensuite faire une proposition devant refléter leur capacité à remplir cette tâche. Le gestionnaire rassemble ensuite toutes les propositions qu'il a reçues et alloue la tâche à l'agent ayant fait la meilleure proposition. **[32]**

Alors notre système est constitué de 2 types d'agents :

- ***Un agent initiateur*** : joue le rôle de gestionnaire, il initialise le processus de composition des services web en soumettant aux autres agents du système le but à réaliser.
- ***Des agents services web*** : peuvent prendre 2 rôles gestionnaire ou contractant, ils représentent les services web dont le but est de simuler l'exécution du service. Cet agent est une entité autonome qui contient les Inputs, les outputs et la qualité de service (le temps d'exécution) du service web et qui est capable de communiquer avec les autres agents dans le but de co-construire un plan d'exécution des services Web.

4.2.2 Architecture détaillé du système

Dans notre système une requête utilisateur est lancée sous forme « état initial, état final » à partir de l'agent initiateur vers l'ensemble des agents services web, ces derniers comparent leurs inputs avec l'état initial de la requête. Ceux qui peuvent contribuer à la réalisation de la requête, vont à leur tour envoyer un appel d'offre à tous les agents du système. Cet appel est composé de leurs inputs, les agents récepteurs vont répondre selon leurs compétences (ce qui correspond à leurs inputs), et ceux qui vont contribuer à la composition vont lancer à leur appel. Le processus d'appel d'offres s'achève une fois l'état final de la requête est rencontré, c'est à dire certaines compositions commencent à apparaitre. A cet instant, démarre la construction progressive des plans et ce en retour arrière de l'appelé vers l'appelant. Chaque agent de niveau **n** remet son plan à l'agent de niveau **n-1** qui évalue à son niveau la composition. Le processus de construction/ évaluation est réitéré jusqu'à atteindre l'agent initiateur.

Un arbre et/ou est alors construit dont le sommet est l'agent initiateur et les feuilles les agents ayant pour input l'input de la requête. Les chemins de l'arbre représentent des plans de composition. L'agent initiateur choisira alors la meilleure composition correspondant au plan ayant la meilleure évaluation.

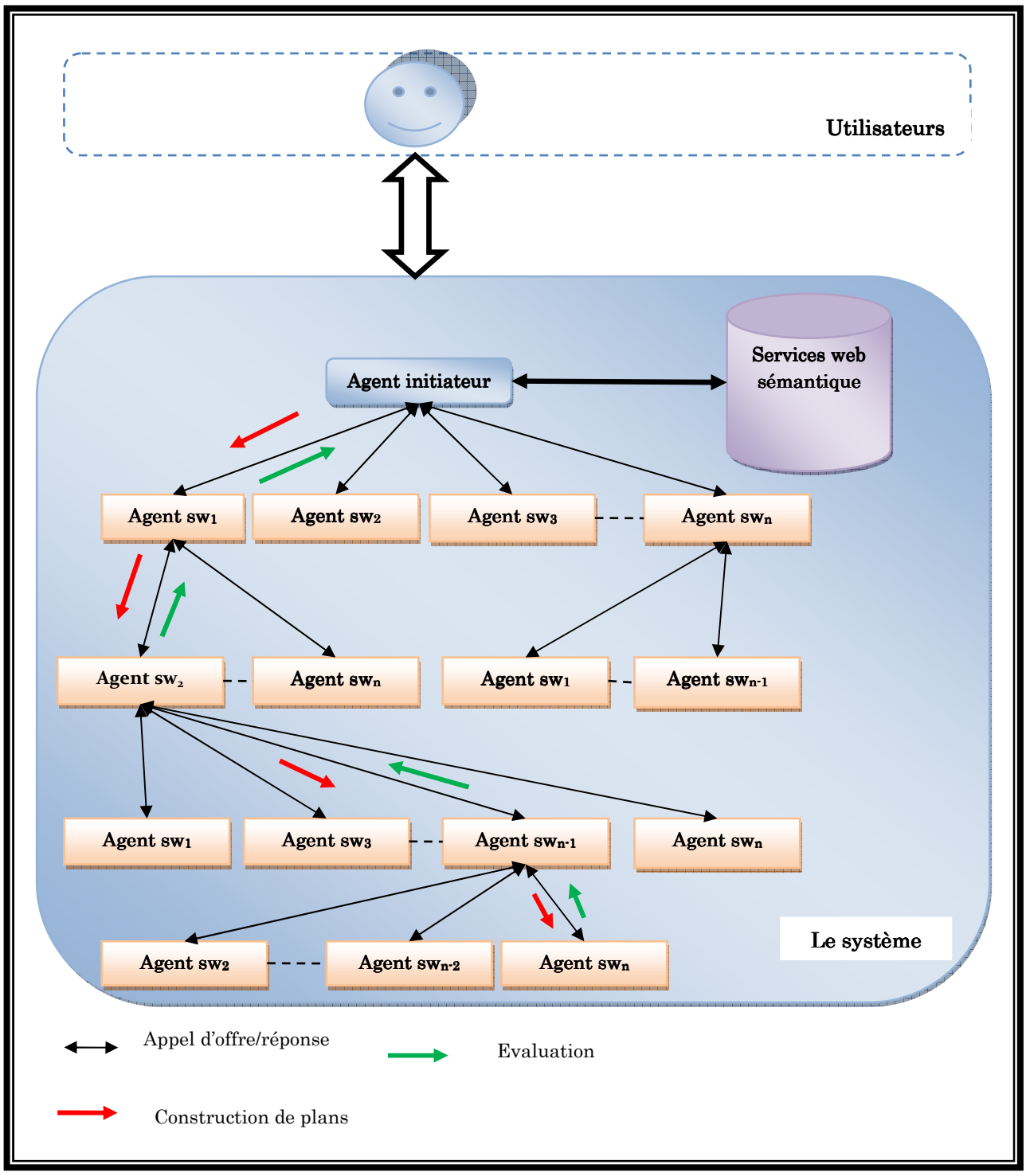


FIGURE 4.2 – Le processus de composition et d'évaluation

Chaque agent du système est exprimé par un ensemble de composants qui travaillent ensemble pour satisfaire les buts de l'agent.

4.2.2.1 Architecture interne de l'Agent initiateur

L'agent initiateur est le premier gestionnaire du système, il possède principalement plusieurs composants figure 4.3.

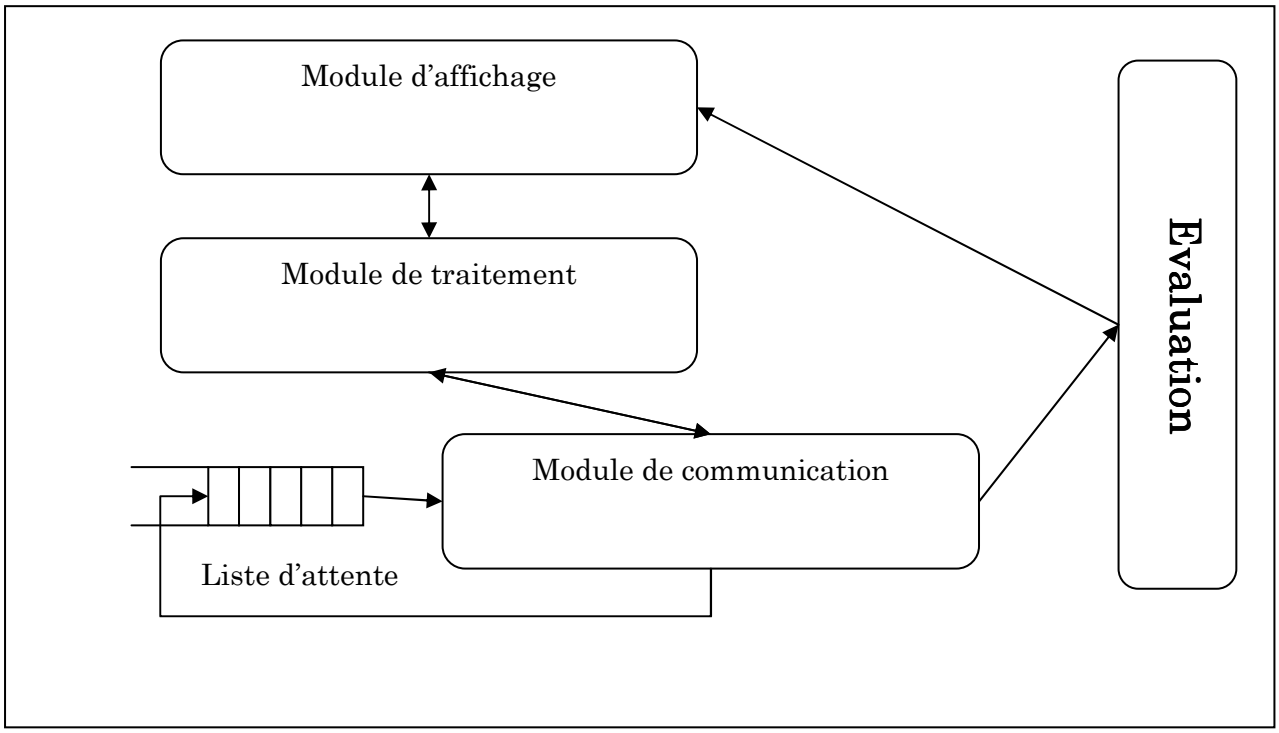


FIGURE 4.3 – Architecture interne de l'agent initiateur

Module affichage

Ce module offre à ses utilisateurs un ensemble d'interfaces conviviales pour qu'ils puissent l'interroger.

1. Affichage de l'interface d'acquisition de la requête : pour permettre aux utilisateurs d'interroger le système. lorsque l'utilisateur pose sa requête dans l'interface du système, cet agent prend la requête et il l'envoi au module de traitement.

2. Affichage des résultats : lorsque le traitement de la requête se termine, cet agent doit visualiser le résultat à l'utilisateur.

Module de communication

1. Distribution de l'inputs/outputs de la requête : la requête doit être traité, alors cet agent broadcaste les inputs/outputs de cette dernière (le but à réaliser) aux agents service web pour qu'ils puissent faire leurs traitement.
2. Réception des réponses : après la distribution de la requête utilisateur aux agents service web ; ces derniers traite l'appel d'offre et ensuite ils envoient leurs réponses au gestionnaire. Ce module reçoit ces réponses.

3. Acquisition des compositions avec leurs évaluations: à la fin de processus de composition, les agents services web de niveau1 qui ont pu satisfirent la requête envoi les plans de la composition avec leurs évaluations à l'agent initiateur.

Module de traitement

1. Initialisation des agents service web : au démarrage du système, l'agent initiateur lance tout d'abord tous les agents services web en utilisant la base de données qui contient la description sémantique des services web pour associer à chaque agent la description sémantique d'un service web concernant le nom, les Inputs, les Outputs et le temps d'exécution du service web.

2. Traitement de la requête : ce module considère la requête comme étant un service web, c'est-à-dire que l'agent extrait les inputs et les outputs de la requête.

Module évaluation

Responsable du choix de la meilleure composition (celle qu'elle a une évaluation minimum). Après la réception des compositions avec leurs évaluations ce module choisis la meilleure composition selon une fonction de choix (CHOIX).

Soit c_i un plan de composition, avec $c_i \in C$

CHOIX (C)= Min (TempEx(c_i)) / TempEx(c_i) est le temps de l'exécution du plan de composition c_i

La liste d'attente

Cette liste contient les identités des agents qui ont acceptés l'appel d'offre, c à dire les agents qui ont les mêmes Outputs de la requête.

Cette liste d'attente assure aussi qu'à un moment donné il y a qu'un seul gestionnaire du système.

4.2.2.2 Architecture interne de l'Agent service web

Au moment de l'initialisation du système, les agents services web sont des contractants mais après les rôles des agents service web ne sont pas spécifiés, n'importe quel agent service web peut agir comme un gestionnaire et annoncer une tâche; n'importe quel autre agent peut être un contractant pour une annonce qu'on a faite. Cette flexibilité permet de nouvelles sous-requêtes : un contractant pour une requête utilisateur peut agir comme un gestionnaire en annonçant les sous-requêtes à d'autres agents. Les liens entre les gestionnaires et les contractants pour diverses sous-requêtes forment un réseau contractuel hiérarchique qui permet la division de la requête et la synthèse des résultats. La figure 4.4 montre l'architecture interne de l'agent service web.

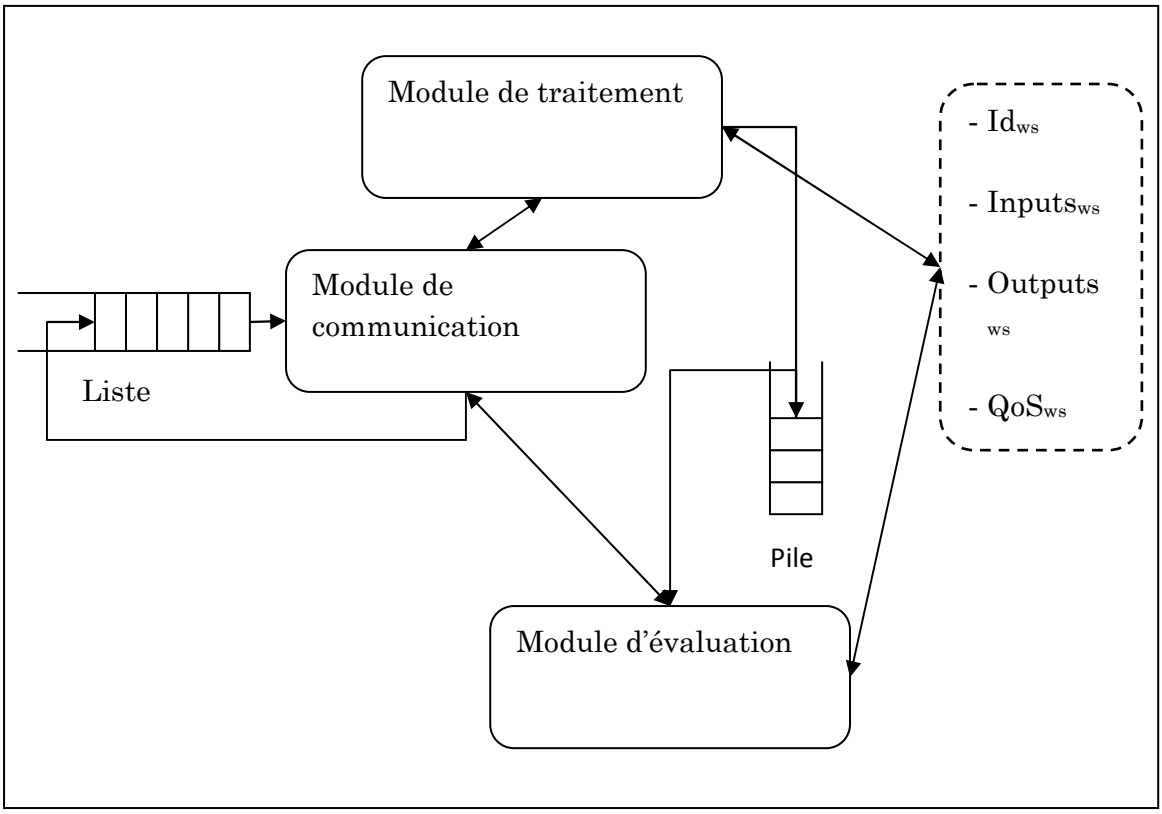


FIGURE 4.4 – Architecture interne de l'agent service Web

Module de communication

Si l'agent service web est un contractant alors ce module est responsable de :

1. La réception des appels d'offre : quand le gestionnaire envoi un appel d'offre, ce module est responsable de son acquisition.

2. L'envoi de la réponse : après le traitement de l'appel d'offre ce module envoi à son gestionnaire un message d'acceptation de l'appel ou un message de refus selon le traitement effectué.

3. L'envoi de la qualité de service : ce module prend le sommet de la pile (le gestionnaire actuel de cet agent contractant) et il l'envoi envoi la qualité de service de cet agent ou les plans de composition avec leurs évaluations.

Si l'agent service web est un gestionnaire alors :

1. L'envoi de l'appel d'offre : quand l'agent service web devient un agent gestionnaire il envoi ses Inputs aux autres agents services web (les contractants) par un appel d'offre.

2. Réception des réponses : les agents service web traitent l'appel d'offre et ensuite ils envoient leurs réponses au gestionnaire. Ce module reçoit ces réponses.

Module de traitement

Lorsque l'agent service web est un contractant, il reçoit un appel d'offre, ce module est responsable à son traitement. Le traitement est effectué comme suit :

Lors de la réception de l'appel d'offre ce module compare toujours les éléments de l'appel d'offre avec ses outputs,

Si *les éléments de l'appel d'offre appartiennent à l'ensemble des outputs du service web* alors

Ce module accepte le traitement

Il empile l'identité de son gestionnaire dans la pile des gestionnaires puis

Il compare ses inputs avec les inputs de la requête

Si *ces derniers apparaissent dans l'ensemble des inputs du service web* alors

Ce module donne l'ordre au module de communication d'envoyer la qualité service.

Sinon

Cet agent service web devient un gestionnaire

Fin Si

Sinon

Ce module refuse le traitement

Fin Si

Module d'évaluation

Quand l'agent service web reçoit la qualité de service d'un autre agent ou un plan de composition avec son évaluation, il choisit la meilleure évaluation par une fonction de choix CHOIX. Soit c_i un plan de composition reçu composé de un ou plusieurs services web, avec $c_i \in C$

CHOIX (C)= Min (TempEx(c_i)) / TempEx(c_i) est le temps de l'exécution du plan de composition c_i

Il enchaîne son identité au meilleur plan d'évaluation, puis il calcule sa propre évaluation selon une fonction d'évaluation EVAL par l'ajout de sa qualité de service à la meilleure évaluation reçu.

EVAL (SW)= Min (TempEx(c_i))+ QoS_{ws}

En fin ce module demande au module de communication d'envoyer le nouveau plan avec son évaluation à son gestionnaire.

Liste d'attente

Quand l'agent service web est un gestionnaire, cette liste contient les agents services web contractant qui ont acceptés le traitement de l'appel d'offre.

Cette liste assure qu'il existe qu'un seul gestionnaire du système à un moment donné.

Pile des gestionnaires

Quand l'agent service web est un contractant, cette pile contient les agents services web gestionnaire de cet agent.

Cette pile est utilisée pour que l'agent contractant puisse envoyer sa qualité de service ou sa propre évaluation des plans de composition avec les identités des services web à son gestionnaire.

4.2 Conclusion

Dans ce chapitre nous avons exposé notre modèle de composition de services Web par un système multi agent en utilisant le protocole des réseaux contractuel (Contract Net Protocol) comme un protocole d'interaction.

Nous avons décrit l'architecture générale de notre système, ainsi que l'architecture interne de chaque agent du système.

Au niveau de chapitre 5 on va essayer de décrire les éléments d'implémentation nécessaire pour le fonctionnement de notre système en restant le plus fidèle possible à sa spécification qui a été présentée dans ce chapitre.

☞ *Chapitre 5*

L'implémentation du système

5.1 Introduction

Après avoir décrit les spécifications de notre système dans le chapitre précédent, nous allons essayer dans ce chapitre de décrire les éléments d'implémentation de notre système : le langage de programmation utilisé, la plate forme utilisée qui permettent de traduire notre conception en une application réelle.

Nous terminerons ce chapitre par une étude de cas, afin de démontrer que notre système peut être appliqué à un domaine quelconque. Dans notre travail, nous avons appliqué ce système à une "Agence de voyage" pour effectuer les différentes réservations, en détaillant chaque étape du système pat cet exemple.

5.2 Algorithmes proposés

Cette partie couvre les algorithmes nécessaires pour la mise en œuvre (l'implémentation) de notre système. La première phase de notre système MAS4WSC concerne l'initialisation des agents services web par la description sémantique des services web correspondant (à chaque agent est associé un service web). La deuxième phase de MAS4WSC concerne la recherche des plans de composition qui peuvent satisfaire la requête. La troisième phase de MAS4WSC concerne la recherche du meilleur plan d'exécution par rapport à certains critères de qualité c'est-à-dire l'évaluation des plans de composition.

L'Algorithme général décrit le fonctionnement global de notre système. Comme notre système est composé de deux types d'agents : agent initiateur et agent service web, alors chaque agent est responsable de l'accomplissement de certaines tâches.

Avant de construire nos algorithmes, on doit poser quelques hypothèses :

- *Hypothèse 1* : Le nombre de services Web disponibles est limité noté par SWi. Soit n le nombre d'agent correspondant à chaque service Web.
- *Hypothèse 2* : La requête du client est considérée comme un service web c'est-à-dire que l'état initial de la requête est considéré comme Input et l'état final comme un Output.
- *Hypothèse 3* : Au plus il existe un seul gestionnaire du système à un moment donné.

5.2.1 Algorithme général

Entrée : $INPUT_{REQ}$, $OUTPUT_{REQ}$
Sortie : le meilleur plan de composition

Tableau : IDSW[n], QSP [k]
Liste : ListeAccept[m]
NbAccept :entier

Début

//l'agent initiateur commence à s'exécuter

BaseSWS.Open ;
NBSWS ←Initialisation (BaseSWS) ;
BaseSWS.Close ;

RecevoirReq () ;

prepareCfps () ; // une méthode du protocole ContractNet

EnvoyerCFP ($INPUT_{REQ}$) ;
EnvoyerCFP ($OUTPUT_{REQ}$) ;

//l'agent initiateur attend le résultat du CFP
//l'agent service web reçoit le CFP

Reçevoir (msg1) ;
Reçevoir (msg2) ;
IN ←msg_1.contenu ;
OUT ←msg_2.contenu ;
ID ←msg_1.destinateur ;

TraiterCFP (OUT, ID) ;

//l'initiateur reçoit les réponses

NbAccept ←0 ;

Si msg= « PROPOSAL » *alors*

ListeAccept[i] ← msg_1.destinateur ;
NbAccept++ ;
msg_3.contenu ←ACCEPT_PROPOSAL ;
msg_3.protocol ← Fipa ContractNet Protocol ;
msg_3.destinataire ← msg_1.destinateur ;
msg_3.envoyer ;

Fin Si

Pour (j=1, j=NbAccept, j++) *faire*

//l'agent service web de l'identité ListeAccept[j] devient le gestionnaire

TraitEval (IN) ;

Fin Pour

//choix du meilleur plan de composition

Pour (m=1, m= NbAccept, m++) *faire*

Recevoir (msgEval) ;
QSP[m] ← msgEval.contenu ;

Fin pour

Max ← QSP[1] ;

Pour (n=1, l= NbAccept, n++) *faire*

Si (QSP[n] >Max) *alors*

Max ← QSP[l] ;

Fin si

Fin pour

Fin

5.2.1.3 La fonction d'Initialisation

Définition de la fonction

Cette fonction est implémentée dans l'agent initiateur, elle sert à créer pour chaque service web de la base des services web sémantique un agent qui contient la description sémantique du service web correspondant.

Présentation de la fonction

Entrée : Base BaseSWS
Sortie : Entier NbSW, un ensemble d'agent
Début

NbSW ⟵0 ;

Tan que BaseSWS != vide *faire*

Créer un agent
Id agent ⟵ nomSW ;
IDSW[NbSW] ⟵nomSW ;
In agent ⟵InSW ;
Out agent ⟵OutSW ;
Qof agent ⟵TempExSW ;
NbSW++ ;

Fin tan que

Retourner (NbSW) ;

Fin

5.2.1.4 La procédure RecevoirReq

Définition de la procédure

Cette procédure affiche une interface simple et conviviale contient des champs pour la saisie de la requête utilisateur, elle analyse la requête et capture les Inputs et les Outputs de la requête.

5.2.1.5 La procédure EnvoyerCFP

Définition de la procédure

L'agent initiateur envoi (broadcaste) la requête aux agents service web du système pour effectuer leurs traitement au dessus, la requête utilisateur est envoyé par un message d'appel d'offre sous forme $INPUT_{REQ}$, $OUTPUT_{REQ}$.

Présentation de la procédure

Entrée : String MSG
Début
msg : message
msg.contenu ⟵ MSG ;

msg.protocol ← Fipa ContractNet Protocol ;
msg.type ← CFP;// Call For Proposal

Pour (i=1, i=NBSWS, i++) *faire*

msg.destinataire ←IDSW[i] ;

Fin pour

msg.envoyer ;

Fin

5.2.1.6 La procédure TraiterCFP

Définition de la procédure

Cette procédure est implémentée dans tous les agents service web, elle traite l'appel d'offre envoyé par l'initiateur. Elle vérifie si l'agent peut résoudre l'appel d'offre ou non.

Présentation de la procédure

Entrée : $OUTPUT_{REQ}$, ID_{des}
Début

msg : message
Pile :PileGes ;

prepareResponse() ;//une méthode du protocole ContractNet

Si $OUTPUT_{REQ} \in OUTPUT_{agent}$ alors

msg.contenu ←PROPOSAL ;
PileGes.empiler(ID_{des})

Si Non

msg.contenu ←REJECT;

Fin Si

msg.envoyre ;

Fin

5.2.1.7 La procédure TraitEval

Définition de la procédure

La procédure TraitEval est le cœur de notre système, elle est implémentée dans l'agent services web. Le résultat de cette procédure est le plan qui répond le mieux à la requête de l'utilisateur. La phase du traitement et d'évaluation est décentralisé c'est-à-dire qu'elle est répartie sur l'ensemble des agents services web.

La requête utilisateur est lancée sous forme « état initial, état final » à partir de l'agent initiateur vers l'ensemble des agents services web, ces derniers comparent leurs inputs avec l'état initial de la requête. Ceux qui peuvent contribuer à la réalisation de la requête, vont à leur tour envoyer un appel d'offre à tous les agents du système. Cet appel est composé de leurs inputs, (exécuter cette procédure) les agents récepteurs vont répondre selon leurs compétences (ce qui correspond à leurs inputs), et ceux qui vont contribuer à la composition vont lancer à leur appel. Le processus d'appel d'offres s'achève une fois l'état final de la requête est rencontré, c'est à dire certaines compositions commencent à apparaitre. A cet instant, démarre la construction progressive des plans et ce en retour arrière de l'appelé vers l'appelant. Chaque agent de niveau **n** remet son plan à l'agent de niveau **n-1** qui évalue à son niveau la composition.

Présentation de la procédure

Entrée : $INPUT_{REQ}$
Début
msg, msgEval : message

Si $INPUT_{REQ} \in INPUT_{agent}$ alors

msgEval.contenu ←Qos agent ;
msgEval.destinataire ←PileGes.dépiler() ;
msgEval.envoyer ;

Si Non

IN ←$INPUT_{agent}$;
prepareCfps () ;
msg.contenu ←$INPUT_{agent}$;
msg.protocol ← Fipa ContractNet Protocol ;

Pour (i=1, i=NBSWS, i++) *faire*

Si (IDSW[i] !=Id agent) *alors*
msg.destinataire ← IDSW[i] ;
Fin si

Fin pour

msg.envoyer ;// nouvel appel d'offre

//cet agent service web (le gestionnaire actuel du système) attend le résultat du CFP

$NbAccept_{agent}$ ⟵0 ;

Si msg= « PROPOSAL » *alors*

$ListeAccept_{agent}$[i] ⟵ msg.destinateur ;
$NbAccept_{agent}$++ ;
msg_1.contenu ⟵ACCEPT_PROPOSAL ;
msg_1.protocol ⟵ Fipa ContractNet Protocol ;
msg_1.destinataire ⟵ msg.destinateur ;
msg_1.envoyer ;

Fin Si

Pour (j=1, j=$NbAccept_{agent}$, j++) *faire*

//l'agent service web de l'identité $ListeAccept_{agent}$[j] devient le gestionnaire

TraitEval (IN) ;

Fin Pour

// L'évaluation des plans obtenus

Pour (k=1, k= $NbAccept_{agent}$, k++) *faire*

Recevoir(msgEval) ;
QoS[k] ⟵ Qos agent + msgEval.contenu ;

Fin pour

Max ⟵ QoS[1] ;

Pour (l=1, l= $NbAccept_{agent}$, l++) *faire*

Si (QoS[l] >Max) *alors*

Max ⟵ QoS[l] ;

Fin si

Fin pour

msgEval.contenu ←Max ;
msgEval.destinataire ←PileGes.dépiler() ;
msgEval.envoyer ;

Fin Si

Fin

5.3 L'intégration du système dans la plate forme JADE

5.3.1 Choix de la plate forme JADE

Afin de réaliser les idées et les concepts inclus dans ce système, il est pratique d'utiliser une plate forme multi-agents. Celle-ci est un inter logiciel qui permet aux développeurs de concevoir et de réaliser leurs applications sans perdre du temps à programmer des fonctions de base pour la création et l'interaction inter-agents. **[44]**

Parmi toutes les plates formes rencontrées dans la littérature telle que ZEUS, AgentTool, JINI, JADE, etc. nous avons choisi la plate forme JADE pour la mise en œuvre de notre système car :

- JADE est un logiciel libre : Une plate forme libre sera préférable à une plate forme propriétaire car elle permet aux utilisateurs d'accéder aux codes sources.
- JADE permet de développer les deux typas d'agents (réactif ou cognitif).
- JADE offre plusieurs fonctionnalités et fournir une large gamme de bibliothèques.
- JADE est riche en documentation cela facilite son utilisation.

En plus le protocole d'interaction utilisé dans notre système est le protocole des réseaux contractuels (ContractNet Protocol) a été normalisé par l'organisation FIPA et implémenté dans la plate-forme JADE. **[45]**

5.3.2 Les classe de JADE utilisée dans notre système

5.2.2.1 La classe Agent

Définie dans le package jade.core, représente une super classe commune pour tous les agents définis par l'utilisateur. De point de vue programmeur, la conséquence, et qu'un agent JADE est simplement une classe JAVA qui étend la clase de base " Agent ". Cela permet à l'agent d'hériter un comportement fondamental caché (qui traite toutes les tâches liées à la plate forme, telles que : l'enregistrement, la configuration, la gestion à distance, etc.), et un ensemble de méthodes qui peuvent être appelées pour implémenter les tâches spécifiques à l'agent.

5.2.2.2 La classe ContractNetInitiator (abstraite)

Implémente le protocole FIPA ContractNet du point de vue de l'agent initiateur. Elle dispose plusieurs méthodes : La méthode prepareCfps(), La méthode handlePropose()…**[45]**

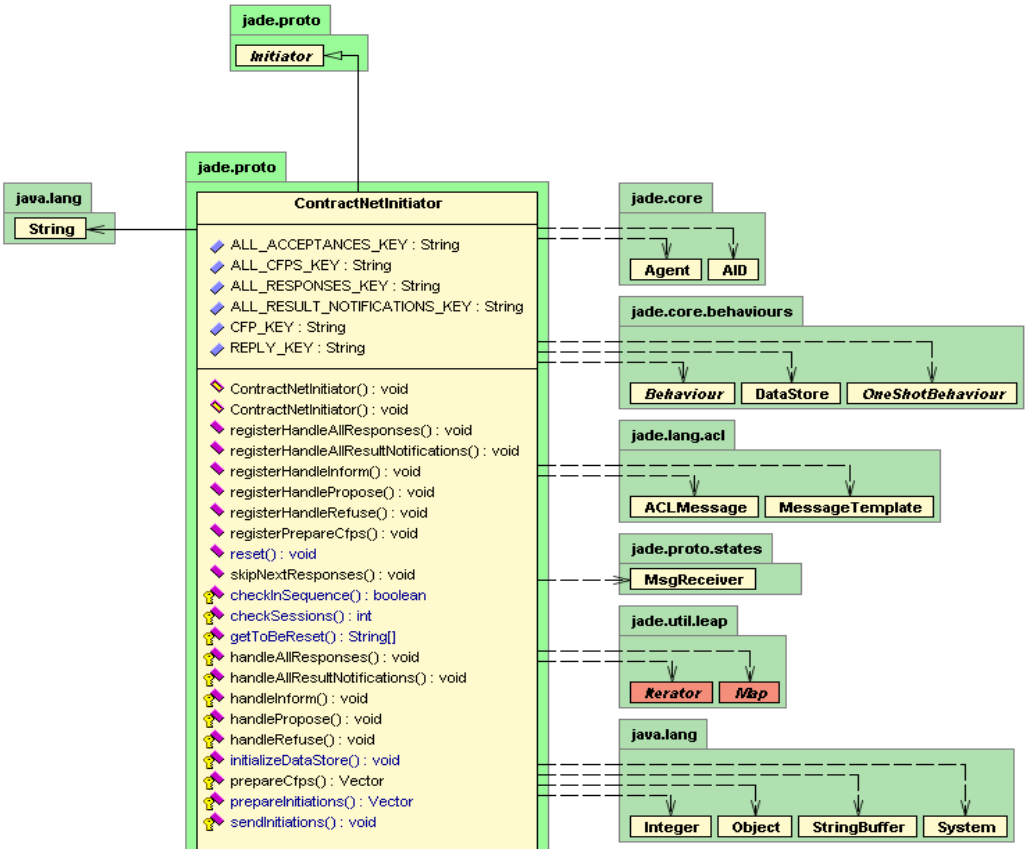


FIGURE 5.1 – Diagramme de classe UML de La classe ContractNetInitiator

5.2.2.3 La classe ContractNetResponder (abstraite)

Implémente le protocole FIPA ContractNet du point de vue d'un agent contractant (qui répond au message CFP).elle dispose aussi plusieurs méthodes : La méthode prepareResponse, La méthode handleRejectProposal ()... **[45]**

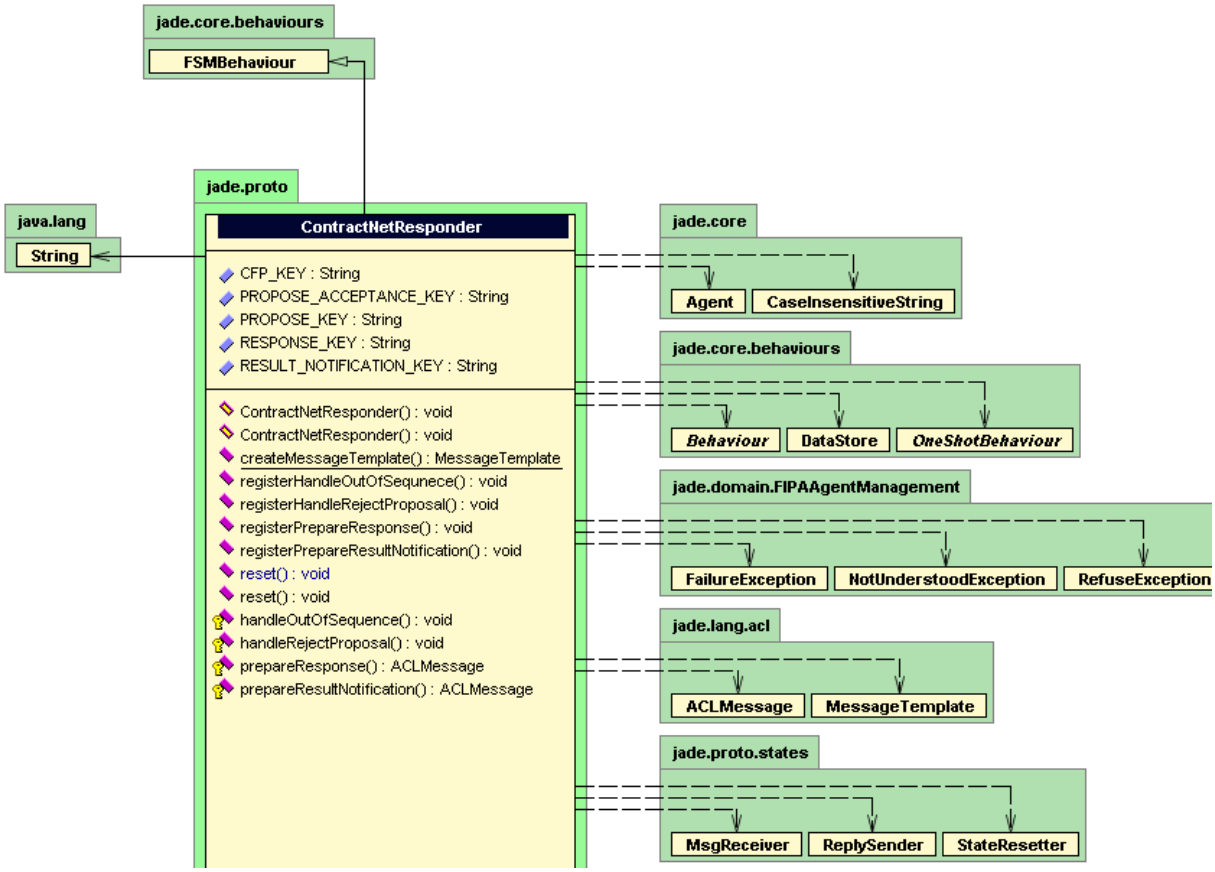


FIGURE 5.2 – Diagramme de classe UML de La classe ContractNetResponder

5.2.2.4 La classe Behaviour

Définie dans le package jade.core. JADE utilise l'abstraction de comportement (Behaviour) pour modéliser les tâches qu'un agent peut exécuter et les agents vont instancier leur comportement selon leurs besoins et leurs capacités. Tout objet de type Behaviour dispose deux méthodes action () et done ().

5.2.2.5 La classe cyclicBehaviour

Définie dans le package jade.core. La classe cyclicBehaviour est une classe abstraite qui hérite la classe abstraite Behaviour. Modélise un comportement qui s'exécute continuellement dans le temps. La méthode done () de cette classe retourne toujours la valeur " false ".

5.2.2.6 La classe ACLMessage

Définit dans le package jade.lang.acl, représente les messages qui peuvent être échangé par les agents. Le tableau 5.1 représente quelques champs d'un message ACL qui vont être remplis par un agent du système.

Champ	Signification
Performative	Type de l'acte de communication
Sender	Expéditeur du message
Receiver	Destination du message
Content	Contenu du message

Tableau 5.1 – Les champs d'un ACLMessage

5.2.2.7 La classe DFservice

Définie dans le package jade.domain. Nous avons utilisé trois méthodes implémentées par cette classe :

Register : permet d'enregistrer un agent dans les pages jeunes du DF.

Deregister : supprimer un agent des pages jeunes du DF.

Search : permet à un agent d'obtenir une liste de tous les agents inscris dans le DF qui peuvent réaliser la tâche qu'il cherche à exécuter.

5.3.3 L'implémentation des agents du système sous la plate forme JADE

Chaque agent implémenté dans la plateforme JADE doit suivre les étapes d'exécution illustrées dans la figure 5.1 : **[46]**

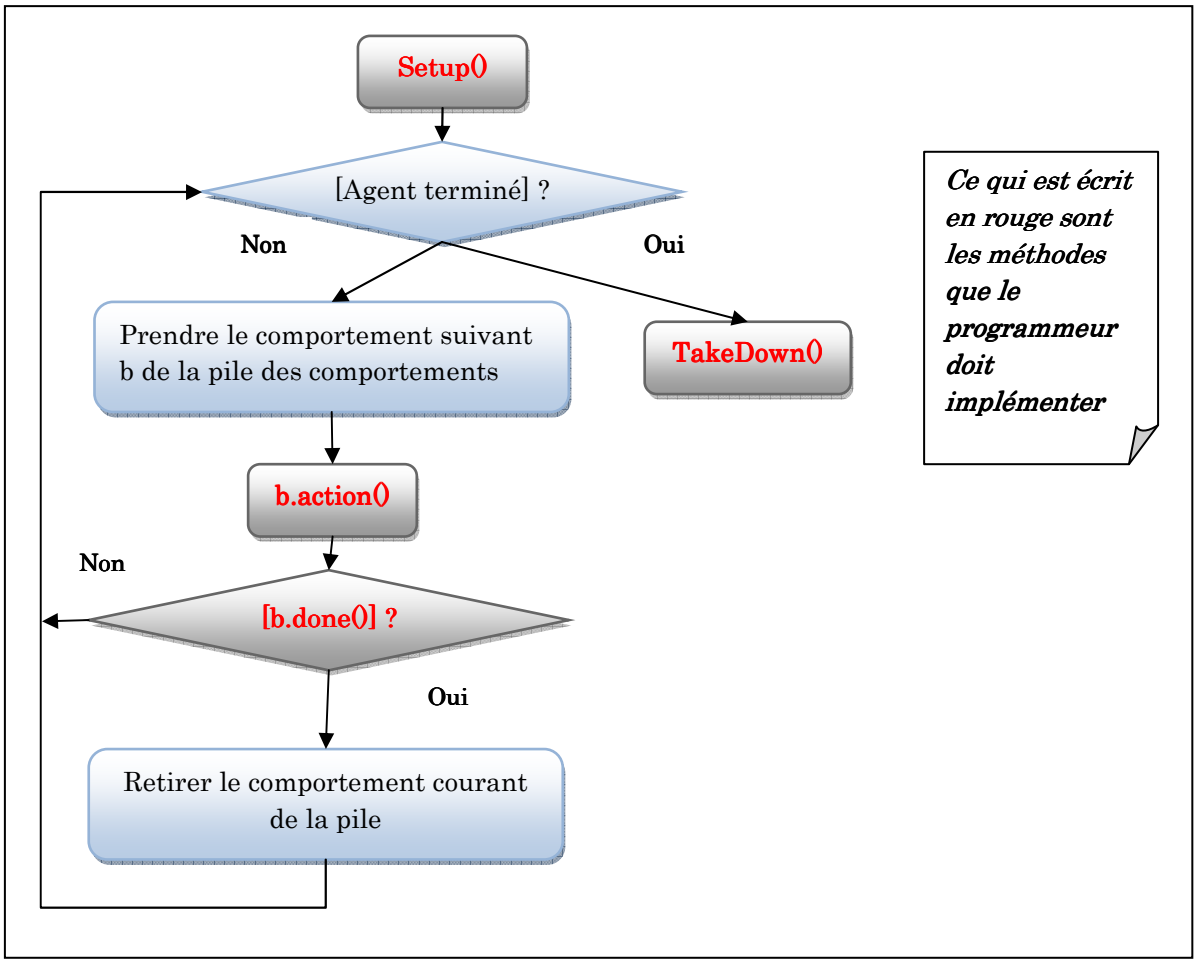


FIGURE 5.3 – Les phases d'exécution d'un agent sous JADE

5.3.3.1 Implémentation de l'agent initiateur

L'agent initiateur est implémenté par la classe **"InitiatorAgent "** qui hérite de la classe jade.core.Agent définit par JADE, cette héritage lui permet d'utiliser les fonctionnements de cette classe. La classe InitiatorAgent se compose des méthodes suivantes :

Setup () : c'est la méthode principale qui amorcera l'exécution du code de l'agent initiateur ; elle déclanche l'exécution de tous les autres méthodes.

Les méthodes de la classe DataStoreConnection et TableDataSet de JAVA pour manipuler la base des ervices web sémantiques telles que : *SetuserName (String), setfileName (String), open (), close, Setpassword (String),* etc.

createNowAgent : de la classe AgentContainer de JADE pour créer les agents service web.

addBehaviour : qui a comme paramètre un objet de la class Behaviour. Ce comportement crée une instance de la classe interface (affichage des interfaces).

La méthode *prepareCfps ()* et la méthode *handlePropose ()* de la classe *ContractNetInitiator*. La méthode prepareCfps () est la première invoquée. C'est l'appel d'offre ou proposition (message CFP). La méthode handlePropose () est appelée dés que le message PROPOSE a été reçu (ie. l'offrant fait une proposition). Elle prépare les messages d'acceptation ou de refus correspondant (message ACCEPT-PROPOSAL ou REJECT-PROPOSAL).

La méthode *EnvoyerCFP (String msg)* qui permet d'envoyer un appel d'offre aux agents services web.

La méthode *EvalPlan ()* qui permet de choisir le meilleur plan d'exécution (le meilleur service web composite) et afficher le résultat de traitement de la requête à l'utilisateur.

La figure 5.4 présente le diagramme de classe UML l'agent initiateur(InitatorAgent).

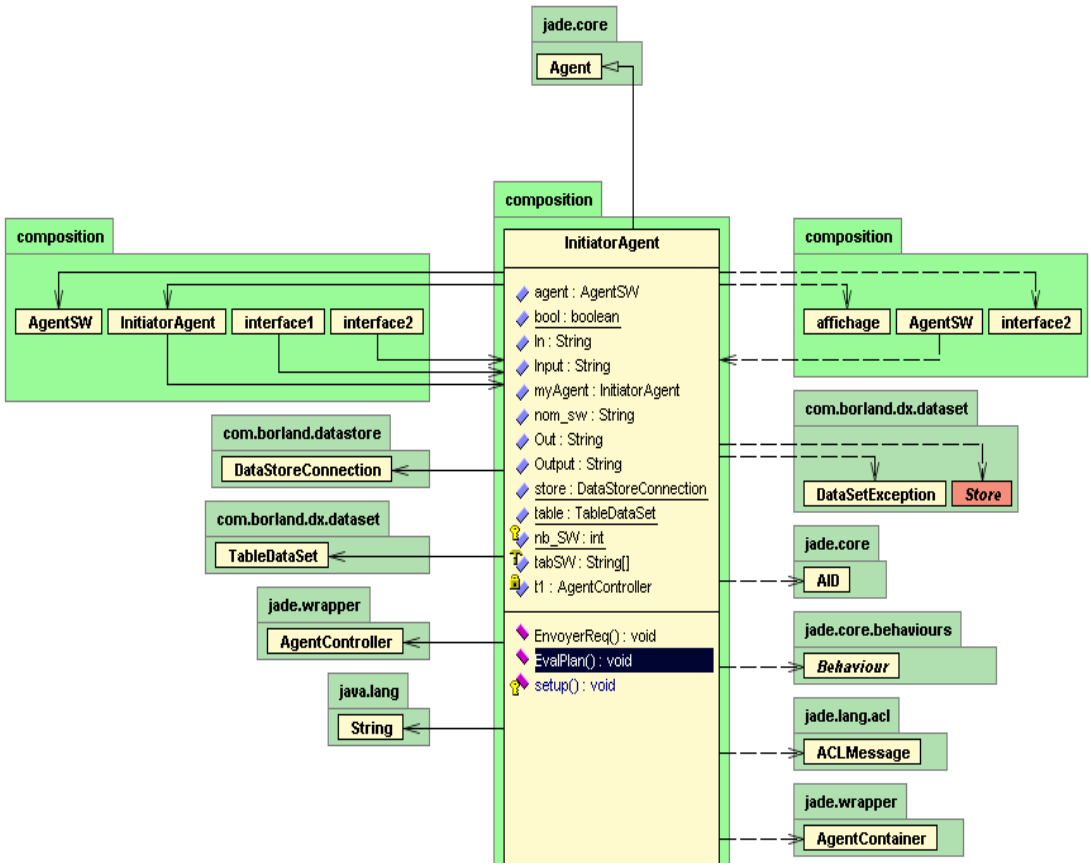


FIGURE 5.4 – Diagramme de classe UML de l'agent Initiateur

5.3.3.2 Implémentation de l'agent service Web

L'agent service web est implémenté par la classe **" AgentSW "** qui hérite la classe jade.core.Agent définit par JADE. Il est composé des méthodes suivantes :

Setup () : c'est la méthode principale pour l'exécution du code de l'agent vendeur.

teckDown () : cette méthode est responsable de la libération de tous les ressources louées par l'agent vendeur juste avant la terminaison de ce dernier.

La méthode DFServiceregister : utilisée pour enregistrer les caractéristiques de cet agent dans le DF.

La méthode *prepareResponse ()* de la classe ContractNetResponder est invoquée dès que l'appel d'offre de l'initiateur a été reçu. Cette méthode prépare le message de proposition PROPOSE (pour répondre au message CFP).

La méthode *ReçevoirCFP ()* affiche une interface pour la saisie de la requête utilisateur, elle analyse la requête et capture les Inputs et les Outputs de la requête.

La méthode *TraiterCFP ()* elle traite l'appel d'offre envoyé par l'initiateur.

La méthode *TraitEval ()* permet de construire et d'évaluer les différents plans de composition qui satisfont la requête utilisateur. Le diagramme de classe UML de l'agent service web (AgentSW) est décrit présenté dans la figure 5.5.

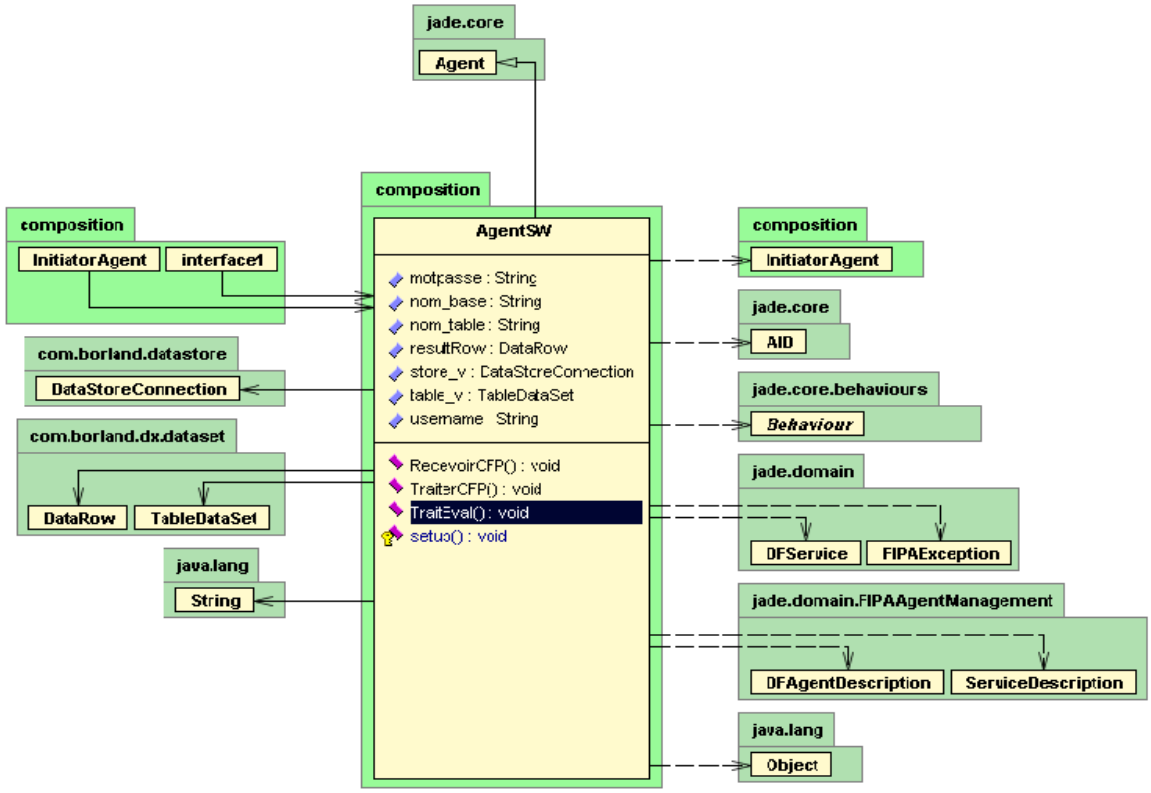


FIGURE 5.5 – Les Diagramme de classe UML de l'agent Service Web

5.4 Etude de cas : Organisation d'un voyage

Une agence de voyages fournit typiquement les services pour : la consultation, la réservation, le paiement et l'annulation de billets d'avion, de chambres d'hôtel et de locations de voiture. Afin de fournir ces services à ses clients, l'agence de voyages doit établir des liens avec d'autres entreprises : compagnies aériennes, compagnies de location de voiture set réseaux hôteliers. Une institution financière (une banque) est également nécessaire pour faciliter les transactions financières entre les clients et l'agence de voyages, ou entre l'agence de voyages et les autres partenaires.

Notre système "e-TravelOrganizer" organise des voyages des clients de cette agence de voyage, le client doit juste indiquer la ville de départ et la ville destination et le système se charge de l'organisation de ce voyage et donne la meilleure solution.

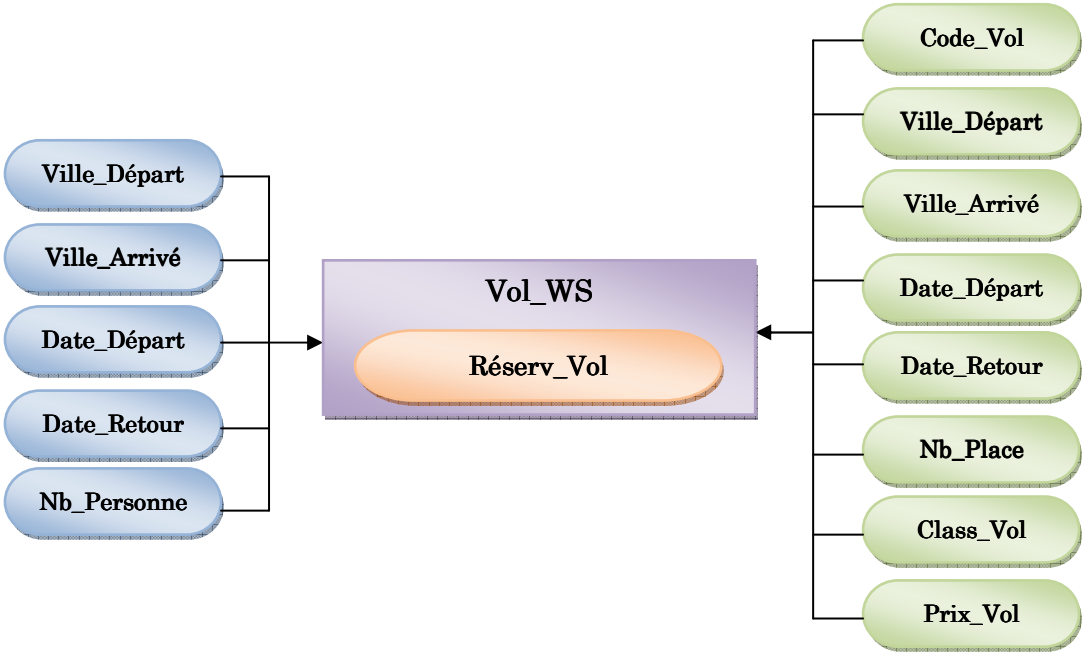
Le principe général du système peut se résumer ainsi :

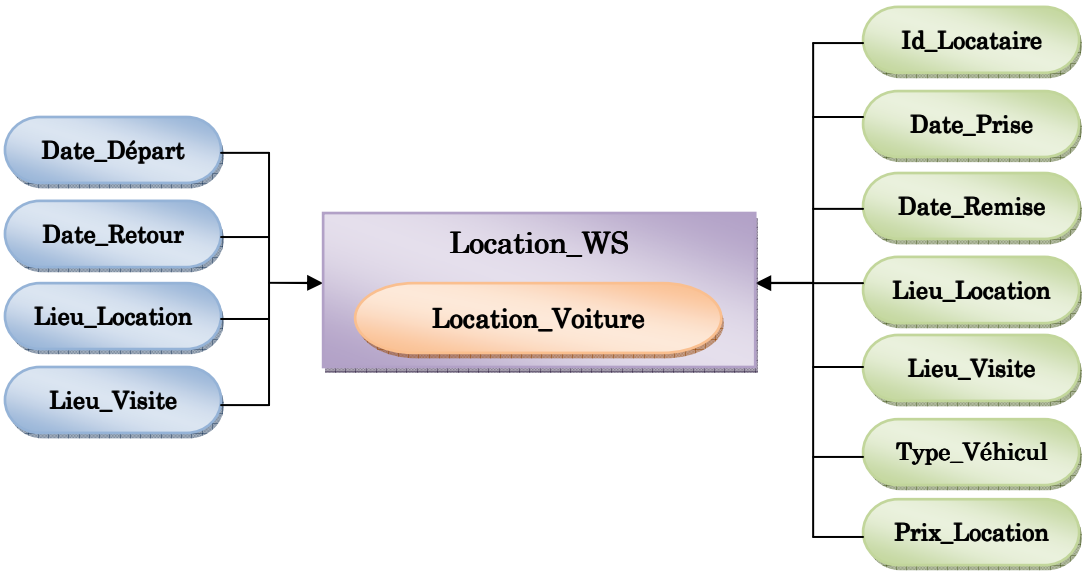
- La compagnie aérienne, la compagnie de location de voitures et le réseau hôtelier doivent fournir des services permettant à une agence de voyages de consulter, réserver, payer et d'annuler les vols, les voitures et les chambres disponibles respectivement.
- La banque doit fournir un service permettant à une entreprise d'effectuer une transaction financière à partir de données de payement (identités du débiteur et du créancier).

A cet effet cinq services web peuvent être proposés :

- Vol-WS : retourne les informations concernant les vols programmés selon la date sélectionnée et la ville de départ et d'arrivée.
- Location-WS: retourne la disponibilité et les informations d'un modèle de voiture auprès des agences de location de voiture selon la date sélectionnée et de la ville sélectionné.
- Hôtel-WS: permet d'afficher touts les noms, l'adresse, le nombre d'étoiles, le prix de la chambre…etc. des hôtels de la ville sélectionnés ainsi que la disponibilité de chambres vide selon la date précisée et le nombre de places.
- Activité-WS: retourne les activités disponibles d'une ville donnée.
- Bank-WS : permet au client de payer les différentes réservations qu'il sera amené à faire, puis additionne les prix générés par les services participant à la composition.
- Vol+Car-WS : offre les fonctionnalités du service Reserv-Vol-WS et du service Rent-a-Car-WS.

La figure 5.6 montre les services fournit par l'agence de notre exemple ainsi que les inputs/outputs correspond à chaque services.




Date_Départ
Date_Retour
Lieu_Location
Lieu_Visite
Location_WS
Location_Voiture
Id_Locataire
Date_Prise
Date_Remise
Lieu_Location
Lieu_Visite
Type_Véhicul
Prix_Location


Ville
Date_Entrée
Date_Sortie
Nb_Personne
Hôtel_WS
Réserv_Hôtel
Nom_Hôtel
Adr_Hôtel
Class_Hôtel
Type_Chambr
Prix_Hôtel

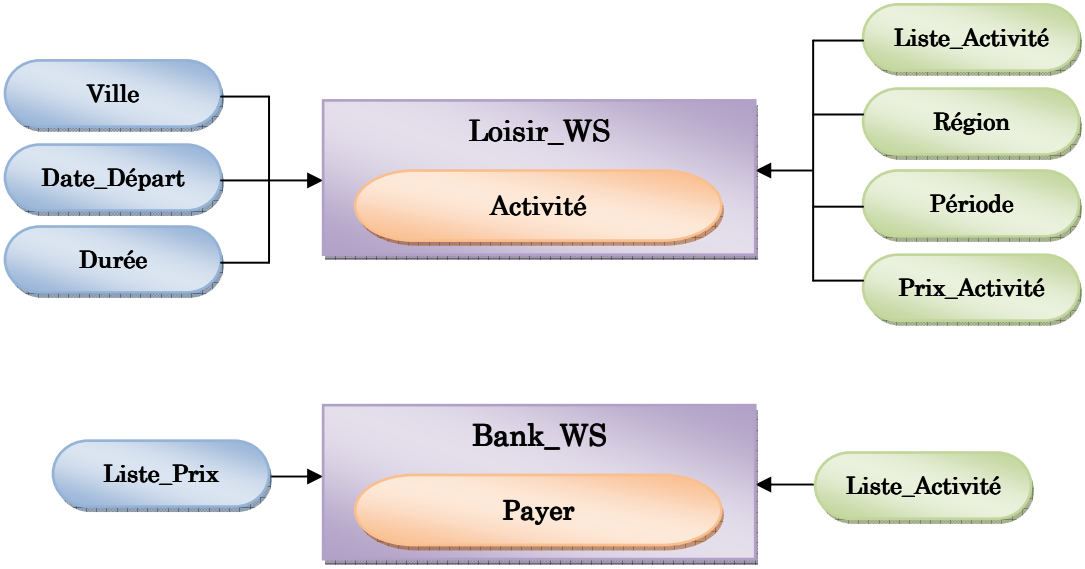

Ville
Date_Départ
Durée
Loisir_WS
Activité
Liste_Activité
Région
Période
Prix_Activité
Liste_Prix
Bank_WS
Payer
Liste_Activité

FIGURE 5.6 – Les services Web de l'agence

Un enseignant chercheur, habite à "Alger", doit se rendre à "Constantine" Samedi pour assister à un colloque. Il décide d'organiser son voyage par Internet en faisant appel au système "e-TravelOrganizer". Sa requête comprend la réservation de billet d'avion et la location d'un véhicule pour la durée du séjour, ainsi que le payement de ces derniers. La figure 5.7 montre la suite d'opérations ainsi que la liste d'Inputs/Outputs de la requête du client

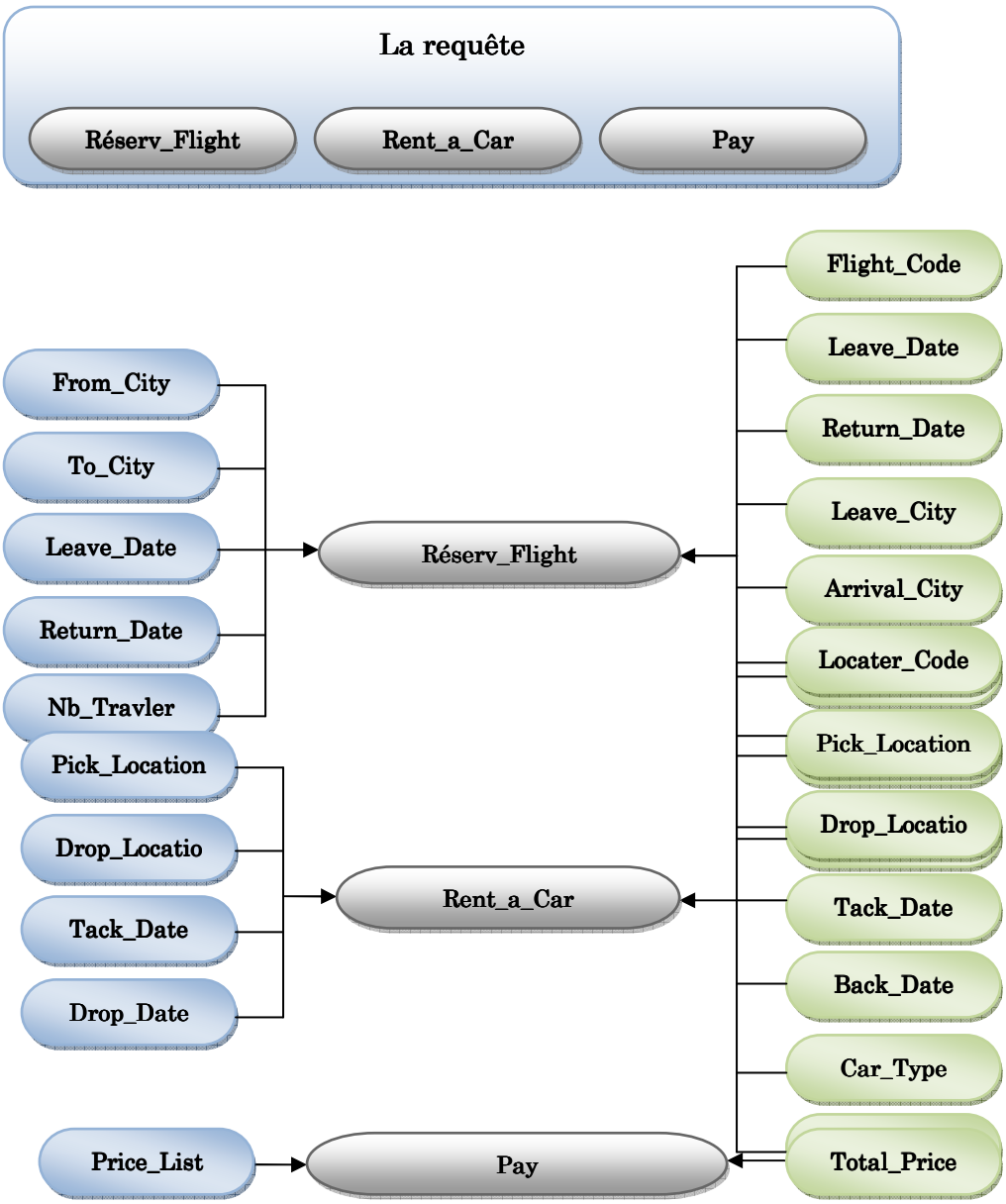


FIGURE 5.7 – La liste des opérations de la requête

Afin de satisfaire la demande du client, on propose la composition des services web montrée dans la figure 5.8.

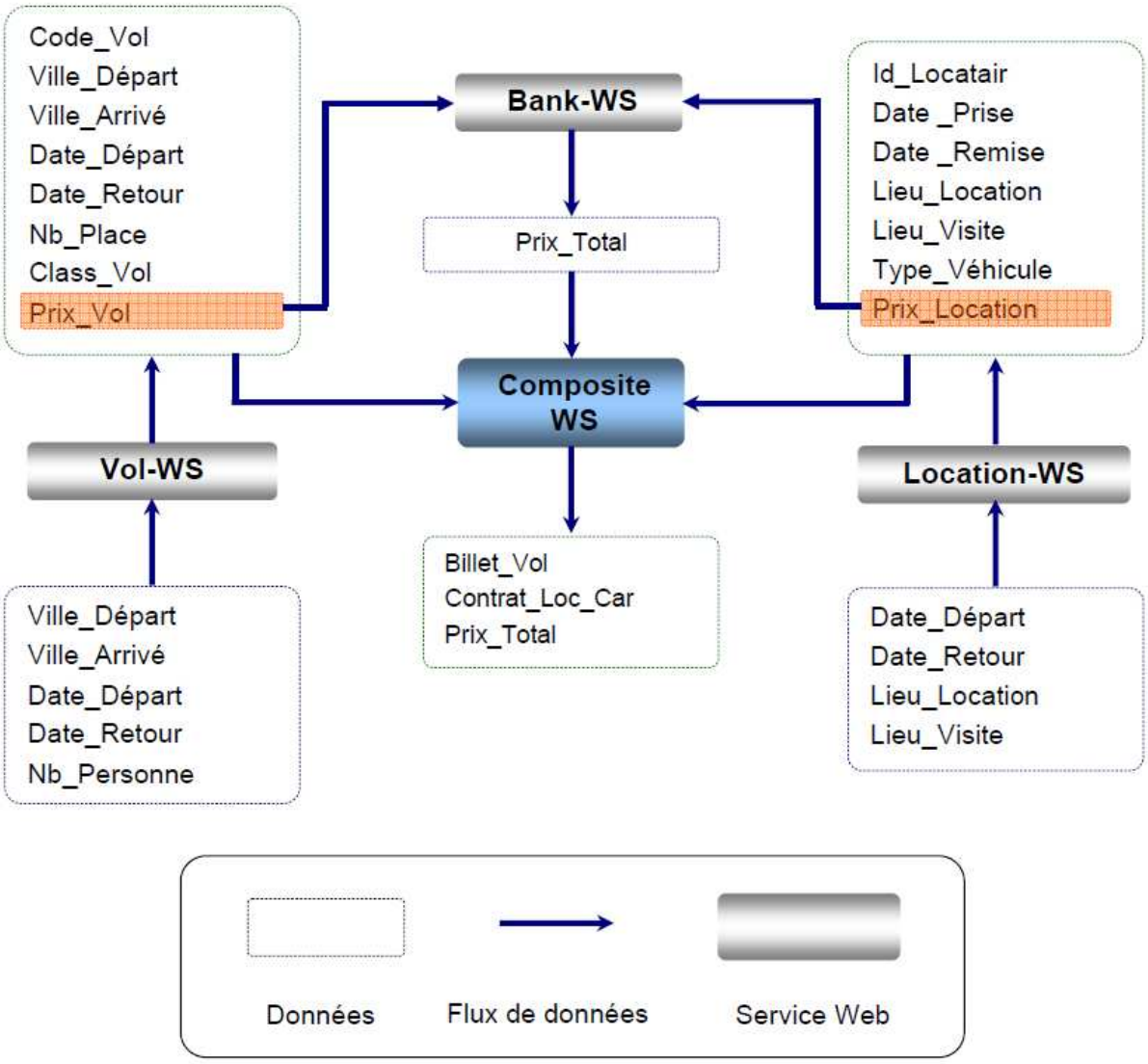


FIGURE 5.8 – Composition des services web

5.4.1 Implémentation de notre exemple

5.4.1.1 Les interfaces de notre système

La figure 5.9 montre l'interface de démarrage de notre système, elle explique brièvement ce qui fait le « e_TravelOrganizer », et comment l'utilisé.

La figure 5.10 illustre l'interface principale de notre Système (interface de la saisie de la requête)

FIGURE 5.9 – Interface de démarrage

FIGURE 5.10 – Interface de la saisie de la requête

5.4.1.2 Implémentation des agents

L'agent Initiateur

La partie du code suivante présente la procédure d'initialisation du système et la création des agents services web.

```
try {

  store.setFileName("BaseSWeb/BaseSW.jds");
  store.setUserName("initiateur");
  store.setPassword("base");
  table.setStoreName("TABLE-SW");
  table.setStore(store);
  table.open();
  table.first();

  while (table.inBounds()) {

    nom_sw = table.getString("ID_SW");
    tabSW[nb_SW] = nom_sw;

    try {
      // create agent t1 on the same container of the creator agent
     AgentContainer container = (AgentContainer) getContainerController(); //
container controller pour créer les nouveaux agents
      t1 = container.createNewAgent(nom_sw,
                          "composition.AgentSW", null);

      t1.start();

      // Thread.sleep(1000);// pour le sniffer

    }
    catch (Exception any) {
      any.printStackTrace();
    }

  }

}
catch (DataSetException dse) {
  dse.printStackTrace();
}
finally {
  try {
    store.close();
    table.close();
```

```
}
catch (DataSetException dse) {
  dse.printStackTrace();
}
```

	ID_SW	NOMBASE	INPUTS
1	Resrv_Vol	BaseSw1.jds	IN1
2	Loc_Voiture	BaseSw2.jds	IN2
3	Reserv_Hôtel	BaseSw3.jds	IN3
4	Activité	BaseSw4.jds	IN4
5	Pyer	BaseSw5.jds	IN5

FIGURE 5.11 – La base des services web sémantiques

Le code suivant présente la procédure qui envoi l'appel d'offre

```
ACLMessage msg1 = new ACLMessage(ACLMessage.CFP);
for(int i=0;i<nb_SW;i++){
  msg1.addReceiver(new AID(tabSW[i], AID.ISLOCALNAME));}
  msg1.setProtocol(FIPANames.InteractionProtocol.FIPA_CONTRACT_NET);
  msg1.setContent(Input);
  send(msg1);
  System.out.println("le message IN a été envoyé");
  ACLMessage msg2 = new ACLMessage(ACLMessage.CFP);
  for(int i=0;i<nb_SW;i++){
  msg2.addReceiver(new AID(tabSW[i], AID.ISLOCALNAME));}
  msg2.setProtocol(FIPANames.InteractionProtocol.FIPA_CONTRACT_NET);
  msg2.setContent(Input);
  send(msg2);
  System.out.println("le message OUT a été envoyé");
```

L'agent service Web

La partie du code ci-après montre comment un agent service Web crée s'enregistre dans le DF.

```
DFAgentDescription dfd = new DFAgentDescription();
dfd.setName(getAID());
ServiceDescription sc = new ServiceDescription();
sc.setName("Service Web");
sc.setType("Voyage");
dfd.addServices(sc);
InitiatorAgent.nb_SW++;
```

```
InitiatorAgent.table.next();

try {
  DFService.register(this, dfd);
}
catch (FIPAException fe) {
  fe.printStackTrace();
}
```

5.4.1.3 Résultats de la composition

La figure 5.11 est une capture d'écran de l'interface de l'agent Sniffer de JADE qui illustre les communications entre les agents du système en cours d'exécution. Cette capture d'écran résume les différents messages échangés entre l'agent initiateur et Cinq Agents services web du notre exemple.

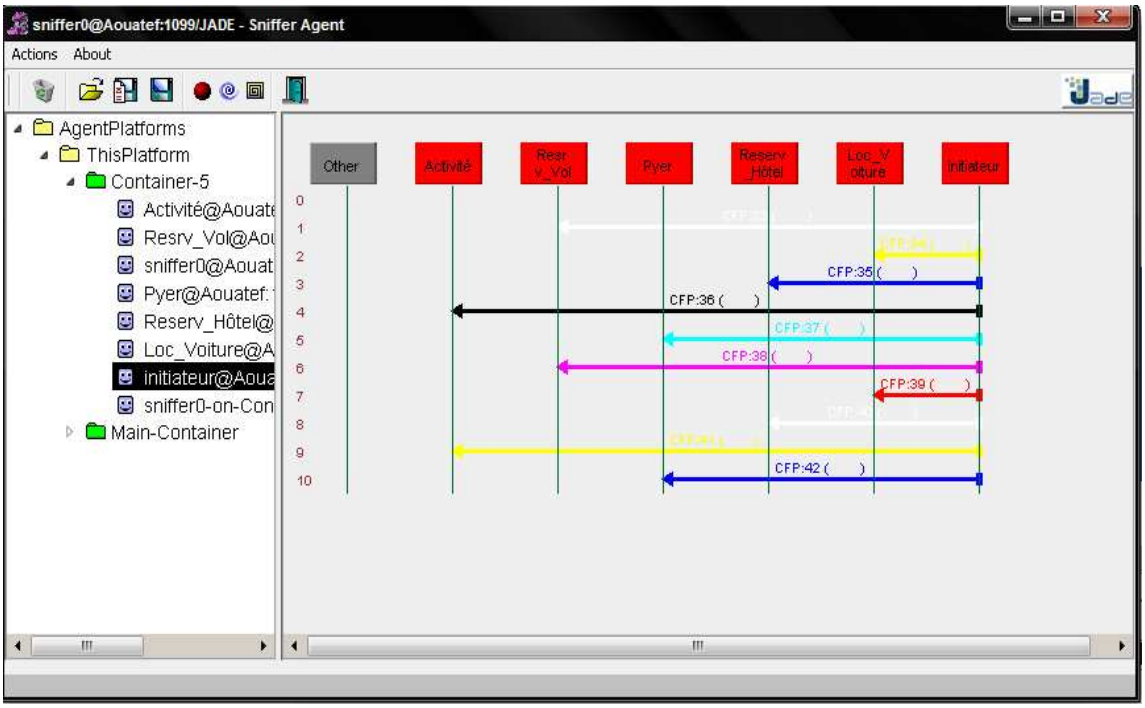


FIGURE 5.12 – L'interface de l'agent Sniffer

5.5 Conclusion

Dans ce chapitre, nous avons essayé de monter qu'il est possible d'implémenter les concepts proposés pour notre système et nous somme attelés à la représentation de toutes les classes et méthodes relatives aux agents : initiateur, vendeur et acheteur. L'utilisation de JADE comme plateforme de développement, nous a permis de profiter de la richesse de sa bibliothèque. En effet, JADE facilite l'implémentation des agents car elle fournit des services de contrôle très utiles pour le développeur.

Conclusion et perspectives

Conclusion générale

Nous avons étudié l'architecture des services Web, les différentes problématiques de recherche sur les services Web et nous nous sommes particulièrement intéressés aux travaux sur la composition dynamique de services Web.

La composition de services Web a pour but d'utiliser les compétences de plusieurs services afin de résoudre un problème qu'aucun ne saurait résoudre individuellement. Le résultat de cette composition est un enchaînement des services Web qui permet de définir la façon dont les données calculées par les uns sont consommées par les autres.

Différentes approches existantes pour la composition de services Web peuvent être regroupées en deux courants. Le premier est la technique de workflow se base sur le fait qu'un service Web composite est définit par un ensemble de services atomiques et par la façon dont ils communiquent entre eux. Dans le second courant, la composition est vue comme la génération automatique d'un plan d'exécution des services Web. Les approches envisagées sont des approches du domaine de l'intelligence artificielle.

Après avoir étudié les travaux existants de ces deux courants, nous avons proposé un modèle de composition de services Web qui s'appuie sur les systèmes multi agents. Un agent est une entité autonome, réelle ou abstraite, qui est capable d'agir sur elle-même et sur son environnement, qui, dans un univers multi-agent, peut communiquer avec d'autres agents, et dont le comportement est la conséquence de ses observations, de ses connaissances et des interactions avec les autres agents.

Les services web partagent avec les systèmes multi agent plusieurs propriétés dont les méthodologies agents tiennent explicitement compte dans leur ébauche de solutions. De telles propriétés sont: **La distribution, La complexité** du problème global à résoudre, **La flexibilité** des interactions, **L'environnement dynamique** et **L'ouverture**.

Dans notre modèle nous avons associé à chaque agent un service web. Les agents permettent d'utiliser les propriétés sémantiques des services en vu de les mettre en correspondance avec les concepts de la requête de l'utilisateur et sont chargés de coordonner leurs compétences afin de proposer un plan de composition. Le protocole d'interaction utilisé dans notre système est le protocole

des réseaux contractuel (contract net protocole) c-à-dire que les agents échangent leurs propositions sous forme d'appel d'offre. Les agents peuvent trouver plusieurs plans de composition qui peuvent satisfaire la requête utilisateur, alors le meilleur plan doit être choisi et retourner à l'utilisateur. Le plan optimal est caractérisé par une qualité de service élevée.

Perspectives :

Afin de compléter le modèle de composition de services Web que nous avons proposé, les améliorations suivantes sont envisageables :

La découverte des services web : nous souhaitons ajouter à ce système un module de découverte des services web au lieu d'une base de données des services web présélectionnés car nous considérons qu'une composition de services web doit s'engager dès la découverte de services jusqu'à l'interaction avec les utilisateurs et en plus le web est dynamiques, les changements sur les services web se font très rapidement.

Munir le système par une base de données qui mémorise les problèmes déjà résolus. Elle constitue alors une source de connaissance exploitable pour les raisonnements futurs.

Adopter une composition adaptative et multi critères, c'est-à-dire selon le besoin de l'utilisateur et selon plusieurs critères de qualité de service que l'utilisateur choisis.

Références bibliographiques

[1] Conservatoire National des Arts et des Métiers de Paris Centre Régional agréé de Nancy Examen Probatoire en Informatique Présenté par Franck Wynen Les « Web services » Novembre 2001

[2]Survey and Comparison of Planning Techniques for Web Services Composition K. S. May Chan1, Judith Bishop1 and Luciano Baresi2

[3]Understanding Web Services- XML, WSDL, SOAP and UDDI Eric Newcomer

[4]Planning and Composition of Web Services with Dynamic Constraints Using Situation Calculus Ken Nariai, Incheon Paik, Mitsuteru Shinozawa *University of Aizu*

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

[6]http://www.w3.org/2002/ws/desc/, Rapport consulté le 05/07/2008

[7]Java Web Services Architecture James McGovern Sameer Tyagi, Michael E. Stevens, Sunil Mathew

[8] Représentation et comparaison de Web services complexes avec des logiques dynamiques Laure Bourgois juin 2007

[9] Michaêl Mrissa, Médiation Sémantique Orientée Contexte pour la Composition de
Services Web Claude Bernard Lyon I Novembre 2007

[10]Composition semi-automatique de Services Web, Nerea Arenaza Février 2006

[11] Patrick Kellert et Farouk Toumani Les web services sémantiques
Juillet 2003

[12] Julien Ponge, Compatibilité et substitution dynamique des web services *Prise en compte de contraintes temporelles et de propriétés d'activation.* Université Blaise Pascal

[13] Jean Charlet, Philippe Laublet & Chantal Reynaud Web sémantique Rapport final décembre 2003

[14] Jérôme Euzenat1, Raphaël Troncy, Web sémantique et pratiques documentaires

[15]Evren Sirin, Bijan Parsia, and James Hendler. Composition-driven filtering and

selection of semantic web services.

[16] Cristina Feier, Dumitru Roman, Axel Polleres, John Domingue, Michael Stollberg1, and Dieter Fensel Towards Intelligent Web Services: The Web Service Modeling Ontology (WSMO)

[17]R.Akkiraju, J.Farrell, J.A.Miller, M.Nagarajan, A.Sheth , K.Verma , *Web Service Semantics – WSDL-S*, IBM Reshearche Report 2006, disponible sur www.iswc2006.semanticweb.org

[18] Jinghai Rao and Xiaomeng Su A Survey of Automated Web Service Composition Methods

[19] Yasmine CHARIF, Chorégraphie dynamique de services basée sur la coordination d'agents introspectifs UNIVERSITÉ PIERRE ET MARIE CURIE, décembre 2007

[20] Frédéric MONTAGUT, Processus Collaboratifs Ubiquitaires Architecture, Fiabilité et Sécurité ,Octobre 2007

[21] Demba COULIBALY UN LANGAGE ET UN ENVIRONNEMENT DE CONCEPTION ET DE DEVELOPPEMENT DE SERVICES WEB COMPLEXES, août 2006

[22] Jinghai Rao, Peep Kùngas, Mihhail Matskin, Composition of SemanticWeb Services using Linear Logic Theorem Proving

[23] Waltinger, R. Web agents cooperating deductively. In Proceeding of FAABS 2000 (April 2000), Greenbelt, Ed., vol. 1871 of Lecture Notes in Computer Sciences, Springer-Verlag, pp. 250–262.

[24] McIlraith, S., and Son, T. Adapting Golog for composition of semantic web services. In Proceedings of the 8th International Conference on Knowledge Representation and Reasoning (KR '02) (Toulouse, France, April 2002), Morgan Kaufmann Publishers, pp. 482–493.

[25] Franco Raimondi, Charles Pecheur, Guillaume Brat PDVer, a Tool to Verify PDDL Planning Domains

[26] Service Composition: State of the art and future challenges G. Kapitsaki, D.A. Kateros, I.E. Foukarakis, G. N. Prezerakos, D.I. Kaklamani and I.S. Venieris

[27] Wu, D., Parsia, B., Sirin, E., Hendler, J., and Nau, D. Automating DAML-S web services composition using SHOP2. In ISWC'03 (2003).

[28]Sirin, E., and Parsia, B. Planning for semantic web services. In Proceedings of Semantic Web Services Workshop at 3rd International Semantic Web Conference
(2004).

[29] Heikki Helin Ahti Syreeni CASCOM: Context-Aware Service Coordination in Mobile
Computing Environments

[30]J. Ferber. *Les systèmes multi-agents, vers une intelligence collective*. InterEditions, 1995.

[31] Stéphane Anglerot, Guillaume Bonnet, Guy Regnault. « Les agents intelligents sur Internet ».

[32] S. Labidi, W. Lejouad De l'Intelligence Artificielle Distribueé aux Systèmes Multi-
Aout 1993

[33]Chaib-Draa, B. (2003).http://www.damas.ift.ulaval.ca/ coursMAS/.

[34] Imed Jarras et Brahim Chaib-draa Aperçu sur les systèmes multiagents

[35] E. H. Durfee and V. Lesser. Negotiating task decomposition and allocation using partial global planning. In L. Gasser and M. Huhns, editors, *Distributed Artificial Intelligence Volume II*, pages 229-244. Pitman Publishing: London and Morgan Kaufmann: San Mateo, CA, 1989.

[36] Amal El Fallah-Seghrouchni Coordination & Planification

[37] Julien Guitton, Planification multi-agent pour la composition dynamique de services Web

[38]El-Fallah-Seghrouchni, A., and Haddad, S. A coordination algorithm for multi-agent planning. In Lecture Notes in Computer Science, vol. 1038. Springer Verlag Publisher, 1996, pp. 86–99.

[39] REID G. SMITH, MEMBER, IEEE The Contract Net Protocol: High-Level Communication and Control in a Distributed IEEE TRANSACTIONS ON COMPUTERS, VOL. C-29, NO. 12, DECEMBER 1980 1104

[40] Mourad Sassi, Brahim Chaib-draa, Stratégies de négociation entre agents dans le domaine du transport

[41] Olivier Boissieer, DEA communication et coopération dans les systèmes à agents Systèmes multi agents Planification multi agents, novembre 2000

[42] Michael C. Jaeger, Lars Engel, and Kurt Geihs, A Methodology for Developing OWL-S Descriptions

[43] Tarek Jarraya, Réutilisation des protocoles d'interaction et Démarche orientée modèles pour le développement multi-agents, décembre 2006.

[44]Sabrina BOUZIDI. « Un modèle de négociation basée sur la formation de coalitions pour l'allocation des taches dans les système d'information coopératifs ». *Thèse de magister mars 2006, pages 91. 92.*

[45]Bellifemine, F., Caire, G., Trucco, T., Rimassa, G.: Jade Programmer's Guide, (2003)
http : //sharon.cselt.it/projects/jade/

[46]Giovanni Caire (TLAB,formerly CSELT). « JADE PROGRAMMING FOR BEGINNERS ». JADE TUTORIAL , 04 Décembre 2003. JADE 3.1.

Résumé

Composer des services web est de trouver un ordre d'exécution de ces services, ordonner des web services est très similaire à un problème de planification automatique, alors la composition des services web peut être réalisée par un système multi agent, en effet les services web partagent avec les systèmes multi agent plusieurs propriétés dont les méthodologies agents tiennent explicitement compte dans leur ébauche de solutions. Dans ce contexte, nous proposons *MAS4WSC* un système multi agent pour la composition automatiquement des services web sémantiques (SWS).Pour chaque service est associé un agent, l'ensemble des agents coordonnent leurs compétences afin de réaliser une tâche soumise par l'utilisateur. Le système proposé s'appuie sur le protocole des réseaux contractuels « *Contract net Protocol* » c'est-à-dire que les agents échangent leurs propositions sous forme d'appel d'offre. Pour une requête utilisateur, plusieurs plans de composition peuvent être trouvés, alors il faut trouver le meilleur plan, la sélection du meilleur est en fonction de quelques critères de qualité (par exemple le temps d'exécution du plan).

Mots clé: service web sémantique, composition des services web, planification automatique, système multi agent, Contract Net Protocol.

Abstract

Composing web services is find an execution order of these services, Ordering web services is very similar to a problem of automatic planning, so the composition of web services can be implemented using a multi-agent system. Web services share with the multi-agent systems several properties whose agents methodologies explicitly take it into account in their draft options. In this context, we propose MAS4WSC an multi agent system for web services composition. For each service is associated an agent, all agents coordinate their skills to perform a task submitted by the user. The proposed system is based on Contract Net Protocol. The agents exchange their proposals in the form of tender. For a user query, multiple dial plans can be found, then we must find the best plan, the selection of the best one is based on some quality criteria (eg execution time of the plan).

Keywords : semantic web service, composition of Web services, automatic planning, multi agent system, Contract Net Protocol.

ملخـــــــص

تركيب خدمات الويب يعني العثور على ترتيب لتنفيذ هذه الخدمات، تركيب خدمات الويب يشبه لحد كبير مشكلة التخطيط التلقائي ، إذن يمكن تركيب خدمات الويب باستخدام نظام متعدد الوكلاء. في الواقع [illegible] . [illegible] *MAS4WSC* نظام متعدد الوكلاء مخصص للتركيب التلقائي لخدمات الويب الدلالية. أين يتم إسناد وكيل لكل خدمة ويب ، وجميع وكلاء النظام ينسقون مهاراتهم لتنفيذ المهمة المقدمة إليهم من قبل المستخدم. النظام المقترح يعتمد على بروتوكول شبكات العقود وهذا يعني أن الوكلاء يتبادلون مقترحاتهم على شكل مناقصة. لتحقيق المهمة المطلوبة من طرف المستخدم، يمكن العثور على عدة مخططات تركيب ، إذن يجب علينا أن نجد أفضل خطة، تقوم على أفضل وجه التحديد على بعض معايير الجودة.

كلمات مفتاحية:خدمات الويب الدلالية، تركيب خدمات الويب، التخطيط التلقائي، نظام متعدد الوكلاء بروتوكول شبكات العقود