République Algérienne Démocratique et Populaire
Ministère de L'enseignement Supérieur et de la Recherche Scientifique
Université A/Mira de Béjaia
Faculté des Sciences Exactes
Département Informatique

Mémoire de Fin de Cycle

En vue de l'obtention du diplôme Master en informatique

Option : Administration et Sécurité des réseaux

THÈME

Mise en œuvre d'une approche de composition de services web à base d'agents logiciels

Réalisé par :

Mr BELHOUD Fatah.
Mr BELMEHDI Rafik.

Devant le jury composé de :

Président :	Mr	. KHANOUCHE M. Essaid.
Examinateur :	Mr	. SAADI Mustapha.
Promoteur :	Mr	. FARAH Zoubeyr.

PROMOTION 2015/2016

Remerciements

NOUS remercions le bon Dieu le tout puissant de nous avoir accordées le courage et la patience pour mener à terme le présent mémoire.

Nous tenons, également, à exprimer notre sincère reconnaissance et notre profonde gratitude à notre encadreur Monsieur FARAH Zoubeyr qui a été d'une disponibilité et d'une attention exceptionnelles.

Nos remerciements sont aussi adressés aux membres du jury qui ont bien accepté de juger notre travail.

Enfin, nous tenons à remercier tous ceux qui ont contribué de près ou de loin à l'élaboration de ce travail.

F.R

Dédicaces

À mes très chers parents,
À mes sœurs et mon frère,
À ma précieuse famille,
À mes amis et collègues,
Et à toutes les personnes que j'ai connues et qui m'ont aidées, un grand MERCI à tous.

Rafik

À mes très chers parents,
À mes frères et sœurs, ma grand-mére,
À ma précieuse famille, mes oncles, tantes, cousins et cousines,
À mes amis et collègues,
Et à toutes les personnes que j'ai connues et qui m'ont aidées, un grand MERCI à tous.

Fatah

Table des matières

LISTE DES FIGURES

Liste des Acronymes

ACL	**A**gent **C**ommunication **L**anguage .
AMS	**A**gent **M**anagement **S**ystem .
AS	**A**gent **S**ervice .
ASCII	**A**merican **S**tandard **C**ode for **I**nformation **I**nterchange .
CORBA	**C**ommon **O**bject **R**equest **B**roker **A**rchitecture .
DF	**D**irectory **F**acilitator .
EDI	**E**lectronic **D**ata **I**nterchange .
FIPA	**F**oundation for **I**ntelligent **P**hysical **A**gents .
HTTP	**H**yperText **T**ransfer **P**rotocol .
IBM	**I**nternational **B**usiness **M**achines .
IP	**I**nternet **P**rotocol .
JADE	**J**ava **A**gent **DE**velopment **F**ramework .
JDK	**J**ava **D**evelopment **K**it .
JSP	**J**avaServer **P**ages .
KQML	**K**nowledge **Q**uery And **M**anipulation **L**anguage .
MadKit	**M**ulti **a**gents **D**evelopement kit .
OMG	**O**bject **M**anagement **G**roup .
OSI	**O**pen **S**ystems **I**nterconnection .
OWL-S	**W**eb **O**ntology **L**anguage .
POP	**P**ost **O**ffice **P**rotocol .
PROLOG	**PRO**grammation en **LOG**ique .
RADL	**R**eticular **A**gent **D**efinition **L**anguage .
RDF	**R**esource **D**escription **F**ramework .
RPC	**R**emote **P**rocedure **C**all .
SGML	**S**tandard **G**eneralized **M**arkup **L**anguage .
SMA	**S**ystème **M**ulti **A**gents .
SMTP	**S**imple **M**ail **T**ransfer **P**rotocol .
SD	**S**ervice **D**escription .
SOA	**S**ervice **O**riented **A**rchitecture .
SOAP	**S**imple **O**bject **A**ccess **P**rotocol .
SQL	**S**tructured **Q**uery **L**anguage .
TCP	**T**ransmission **C**ontrol **P**rotocol .
UBR	**U**DDI **B**usiness **R**egistry .
UDDI	**U**niversal **D**escription **D**iscovery and **I**ntegration .
UML	**U**nified **M**odeling **L**anguage .
W3C	**W**orld **W**ide **W**eb **C**onsortium .
WSDL	**W**eb **S**ervices **D**escription **L**anguage .
WS-I	**W**eb **S**ervices **I**nteroperability .
WSIG	**W**eb **S**ervice **I**ntegration **G**ateway .
XML	**E**xtensible **M**arkup **L**anguage .

Introdution Générale

De plus en plus, la compétitivité de l'entreprise dépend de sa capacité à générer, à traiter et à analyser rapidement et efficacement l'information pour prendre des décisions pertinentes, que ce soit dans sa relation avec ses clients, ses fournisseurs, ses collaborateurs ou ses partenaires. L'entreprise est incessamment exposée à des défis des autres entreprises concurrentes auxquels elle doit continuellement réagir. En effet, elle doit répondre aux enjeux concurrentiels du marché en améliorant la qualité de son offre, en proposant de nouvelles fonctionnalités pour étendre son activité commerciale, et en s'ouvrant sur les systèmes d'information de ses partenaires pour répondre au besoin du partage instantané de l'information.

Grâce au développement accéléré des technologies de l'information, l'émergence des services web [1] introduit un nouveau paradigme pour permettre l'échange d'informations à travers l'Internet et aussi le développement, le déploiement et l'intégration d'applications sur le web.

Les services web individuels restent limités par leurs capacités, l'entreprise est amenée à composer un ensemble de services afin de créer des services plus complexes. Cette composition se présente comme un paradigme fondamental de la technologie des services web.

Les systèmes multi agents [2] sont également destinés au développement d'applications distribuées, toutefois, dans une perspective différente aux services web mais complémentaire, les système multi agents offrent un grand intérêt et promettent d'avoir un impact important sur la technologie de services web.
Dans ce travail, nous étudions la problématique de composition de services web à base d'agents logiciels en utilisant les architectures de JADE et WSIG [3] afin de rendre cette composition dynamique et automatique et permettre des compositions complexes de services web, pour cela nous donnons comme exemple d'un

prototype sur un service voyage et qui sera tester sur les plateformes JADE et WSIG.

Ce mémoire est organisé en trois chapitres

Dans le ***chapitre 1***, nous donnons quelques généralités sur l'architecture orientée service SOA et les services web et leurs fonctionnements, ensuite nous définissons le web sémantique et les services web sémantique, ainsi que les principes de composition de services web.

Dans le ***chapitre 2***, nous définissons quelques concepts sur les agents logiciels ainsi que les systèmes multi agents, ensuite nous présentons la communication entre agents et les différentes plateformes orientées agent, enfin nous expliquons l'intégration des services web avec des agents.

Concernant le ***chapitre 3***, il sera consacré à la réalisation de notre prototype sur le " service voyage ", nous commençons par présenter les architectures de base de JADE et WSIG, ensuite nous montrons les différents diagrammes de séquences et de classes appropriés à notre application, et nous terminons par les différentes étapes de configuration et quelques captures de notre application.

Enfin, notre travail se termine par une conclusion générale résumant les grands points qui ont été abordés ainsi que les perspectives futures de notre travail.

1

Concepts sur les services web

1.1 Introduction

Dans ce chapitre, nous présentons une étude sur l'architecture orientée service SOA, les services web et leur architecture et leurs fonctionnements, ensuite nous poursuivons avec des notions sur le web sémantique et les services web sémantiques, nous terminons par la composition de services web et ses types.

1.2 L'architecture SOA

1.2.1 Définition

Le terme SOA (Service Oriented Architecture) [4] signifie architecture orienté services, qui est un ensemble de services qui communiquent les uns avec les autres. La communication peut impliquer soit le passage de données simple ou il pourrait impliquer deux ou plusieurs services de coordination d'une certaine activité.

SOA est un moyen logique de la conception d'un système logiciel pour fournir des services soit à des applications de l'utilisateur final ou d'autres services distribués dans un réseau grâce à des interfaces publiées et découvrables. La SOA définit une interaction entre agents comme un échange de messages entre les demandeurs de services (clients) et les fournisseurs de services. Les clients sont des agents logiciels qui demandent l'exécution d'un service. Les fournisseurs sont des agents logiciels qui fournissent le service. Les agents peuvent être simultanément deux clients et fournisseurs de services. Les fournisseurs sont responsables de la publication d'une description du service qu'ils fournissent. Les clients doivent être en mesure de trouver la description des services dont ils ont besoin et doivent être en mesure de se lier à eux.

1.2.2 Les principes d'une architecture SOA

Une architecture orientée service repose sur quelques principes :

- La notion de service, c'est-à-dire une fonction encapsulée dans un composant que l'on peut interroger à l'aide d'une requête composée d'un ou plusieurs paramètres et fournissant une ou plusieurs réponses.
- La description du service, consiste à décrire les paramètres d'entrée du service et le format et le type des données retournées.
- La publication, consiste à publier dans un registre les services disponibles aux utilisateurs.
- La découverte des services, qui recouvre la possibilité de rechercher un service parmi ceux qui ont été publiés.
- L'invocation, représentant la connexion et l'interaction du client avec le service.

1.3 Les services web

1.3.1 Définition

Selon la définition du W3C (World Wide Web Consortium), un service Web est un logiciel conçu pour soutenir l'interaction interopérable entre machines sur un réseau. Il dispose d'une interface décrite dans un format (WSDL)exploitable par des machines. D'autres systèmes interagissent avec le service Web d'une manière prescrite par sa description en utilisant des messages SOAP, qui sont généralement transmis en utilisant HTTP avec une sérialisation XML en conjonction avec d'autres normes liées au Web [1].

1.3.2 Historique et Origine

Les Web services sont nés de l'effort de plusieurs organisations qui ont partagé un intérêt commun en développant et en maintenant un " marché électronique". Celles-ci souhaitaient pouvoir communiquer plus simplement et sans avoir à se concerter sur chacune de leurs transactions pour pouvoir interpréter leurs différentes données. Elles souhaitaient supprimer l'isolement de leur système informatique avec les autres, c'est ainsi que naquit en 1975 l'EDI (Electronic Data Interchange) [5], Il s'agit d'un format standard permettant l'échange d'un certain type de données.
Vers la fin des années 80, l'évidence fut que l'âge des systèmes informatiques isolés touchait à son terme tandis que différents ordinateurs, de tailles, de capacités et de formes variées, apparaissaient au sein d'une même organisation, pour exploiter au mieux et au plus bénéfique la puissance d'analyse de ces ordinateurs, de nouvelles technologies apparut telles que CORBA (Common Object Request Broker Architecture) [6].CORBA, est une architecture logicielle, destinés aux concepteurs et aux développeurs qui veulent produire des applications conformes aux normes OMG (Object Management Group), l'avantage de la conformité est, en général, pour être en mesure de produire des applications interopérables qui sont basées sur des objets distribués. L'architecture CORBA assure l'achemine-

ment des appels entre les processus. Les applications et les composants CORBA associent typage statique et dynamique, ainsi, chaque composant est présenté statiquement par une interface mais les composants qui utilisent celui-ci doivent vérifier dynamiquement que l'interface est effectivement implantée.
A peu près à la même époque, un regroupement d'organisations recherchant une façon de structurer et d'échanger des documents XML, créa un protocole appelé SOAP (Simple Object Access Protocol) qui permet la transmission de messages entre applications distantes. Le transfert se fait le plus générale grâce au protocole HTTP, mais peut tout aussi bien se faire par d'autres protocoles, comme SMTP (Simple Mail Transfer Protocol). Le SOAP a été conçu à partir des concepts qui avaient produit, entre autres, des technologies comme CORBA et EDI, en lui ajoutant les composants XML et HTTP de telle façon que les applications puissent interagir entre elles, même à travers les firewalls des entreprises.
Aujourd'hui, les services web provoquent un intérêt certain auprès des architectes et des décideurs. Dès à présent, les Web Services sont sortis du champ des échanges interentreprises pour s'adapter à celui du référencement et de la mise à disposition des ressources de l'entreprise. Par ailleurs, ce modèle n'échappe pas à des problèmes de performance : les données sont transférées en ASCII dans une encapsulation XML elle-même intégrée dans une enveloppe SOAP puis HTTP... Le problème du choix de la bonne granularité du service, commun à toutes les architectures distribuées, se présente dans le cas des Web Services de manière plus aiguë encore. Même s'ils n'ont pas acquis la maturité nécessaire à leur industrialisation, les services web se présentent comme la solution appropriée aux problématiques d'échange de données et d'intégration d'applications.

1.3.3 Architecture et fonctionnement des services web

De plus en plus d'entreprises ont besoin de rendre leurs applications accessibles sur le web. Les motivations sont multiples : élargir l'audience des utilisateurs, vendre des services en ligne, faire communiquer des applications existantes ou supporter des interfaces de types différents.

L'OSI et le W3C sont les comités de coordination responsables de l'architecture et de standardisation des services web. Pour améliorer l'interopérabilité entre les réalisations de service Web, l'organisation WS-I a développé une série de profils pour faire évoluer les futures normes impliquées. L'aspect le plus important des Web Services est qu'ils reposent sur plusieurs standards qui permettent la structuration des architectures. Elle contient entre autres XML et SOAP pour le formatage et le codage des données, WSDL pour la description des services web et UDDI pour la recherche des services web nécessaire au bon fonctionnement des applications. Une des raisons principales pour lesquelles les services web sont employés semble être qu'ils se fondent sur le Internet et HTTP pour fonctionner.

1.3.3.1 XML

XML (Extensible Markup Language) [7], est un standard pour la définition des données dans un format de texte simple et flexible dérivé de SGML (Standard Generalized Markup Language).
XML est conçu pour relever les défis du web, où il joue un rôle important dans l'échange d'une grande variété de données sur le Web.
Les documents XML sont constitués d'unités de stockage appelées entités , qui contiennent des données composées de caractères , dont certains forment les caractères des données, et dont certains forment le balisage.
XML permet de définir un format d'échange selon les besoins de l'utilisateur et offre des mécanismes pour vérifier la validité du document produit. Il est donc essentiel pour le receveur d'un document XML de pouvoir extraire les données du document. Cette opération est possible à l'aide d'un outil appelé analyseur ou parseur (en anglais parser). Le parseur permet d'une part d'extraire les données d'un document XML (parsing) ainsi que de vérifier éventuellement la validité du document.

1.3.3.2 SOAP

SOAP (Simple Object Access Protocol)[8] est un protocole léger destiné à l'échange d'informations structurées dans un environnement décentralisé et distribué, il est moins lourd à mettre en œuvre que d'autres protocoles et c'est pour cela qu'il est de plus en plus adopté. Il utilise des technologies XML pour définir un cadre de messagerie extensible fournissant une construction de message qui peut être échangé par l'intermédiaire du protocole HTTP mais aussi SMTP et POP sous forme de texte structuré.
Deux grands objectifs de conception pour SOAP sont la simplicité et l'extensibilité. SOAP tente d'atteindre ces objectifs en omettant, dans le cadre de la messagerie, les caractéristiques qui sont souvent trouvées dans les systèmes distribués. Ces caractéristiques comprennent, la fiabilité, sécurité, corrélation et routage.
Par rapport à tous les autres protocoles comme RPC, SOAP est interopérable, ainsi il est indépendant des plates-formes et langages de programmation. L'autre avantage réside dans le déploiement des applications. Pour communiquer entre deux sociétés via Internet, il est très souvent désagréable d'utiliser autre chose que HTTP ou POP/SMTP à cause des Firewalls, qui doivent êtres reconfigurés engendrant ainsi des trous de sécurité. De plus, cela implique de longues négociations entre administrateurs réseaux.

1.3.3.3 UDDI

UDDI (Universal Description, Discovery and Integration) [9] est une norme pour la description et la découverte des services web.
Le but de UDDI est de faciliter la découverte de services à la fois au moment de la conception et lors de l'exécution. Dans des scénarios typiques de services web, les fournisseurs de services veulent publier leurs descriptions de service dans un registre, et les demandeurs de services aux deux temps de conception ou d'exécution veulent interroger le registre pour des descriptions de service. Il existe deux principaux types de registres UDDI : publique et privé. Le registre public est appelé UBR (UDDI Business Registry) ou le registre des entreprises. L'UBR est

hébergé par un petit nombre d'entreprises (comme IBM et Microsoft), et chaque entreprise héberge un nœud UDDI qui est répliqué avec les autres nœuds. Cela signifie que vous pouvez trouver les données publiées sur le site hébergé par IBM lors de la recherche à travers le site hébergé par Microsoft. Bien que le concept d'un registre public a été racoleuse quand UDDI a d'abord été annoncé, ce type de registre n'a pas été largement utilisé. Un registre public ne fournit pas le niveau de confiance qui est nécessaire pour permettre à un demandeur de service pour sélectionner et utiliser un fournisseur de services répertoriés dans le registre. La plupart de l'intérêt dans UDDI a été axé sur les registres privés. Un registre privé peut être hébergé sur Internet ou sur un intranet, où une seule entreprise peut héberger un registre UDDI pour les services web qu'elle utilise en interne.

1.3.3.4 WSDL

WSDL (Web Services Description Language) [10] fournit un modèle et un format XML pour la description des services web.
WSDL décrit un service web en deux étapes fondamentales : l'un abstrait et l'autre concret. Dans chaque étape, la description utilise un certain nombre de constructions de promouvoir la réutilisabilité de la description et de la conception des préoccupations distinctes et indépendantes.
À un niveau abstrait, WSDL décrit un service web en fonction des messages qu'il envoie et reçoit ; Les messages, syntaxe et sémantique indispensables à un service web, sont toujours abstraits. Au niveau concret, une liaison précise les détails de transport et le format pour une ou plusieurs interfaces. Les ports, adresse réseau à laquelle le service web peut être invoqué, sont toujours concrets.
WSDL joue un rôle important dans l'interopérabilité des services web. Il est défini selon une sémantique totalement indépendante du modèle de programmation de l'application. En effet, il sépare clairement la définition abstraite du service (en termes de messages à échanger avec le service), de ses mécanismes de liaison tels que la définition des protocoles applicatifs.
Nous pouvons Voir la figure 1.1, l'architecture d'un service web fonctionne de la

manière suivante :

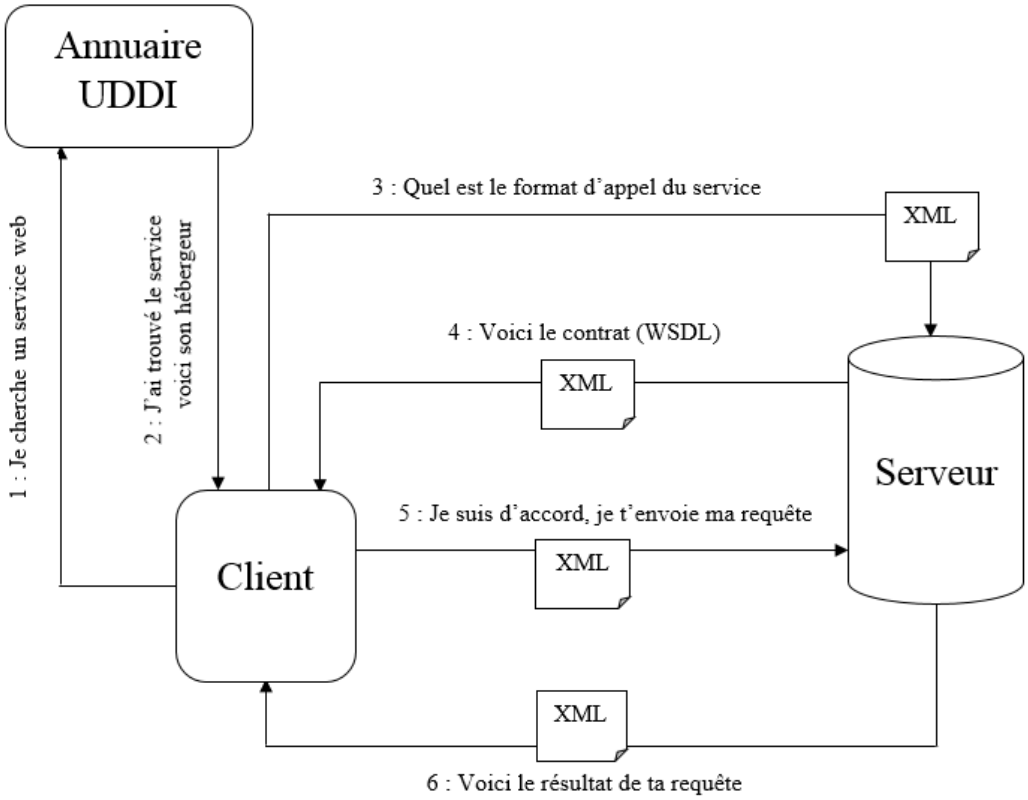

FIG. 1.1 – *L'architecture d'un service web.*

1. Le client envoie une requête à l'annuaire de Service pour trouver le service Web dont il a besoin.
2. L'annuaire cherche pour le client, trouve le service Web approprié et renvoie une réponse au client en lui indiquant quel serveur détient ce qu'il recherche.
3. Le client envoie une deuxième requête au serveur pour obtenir le contrat de normalisation de ses données.
4. Le serveur envoie sa réponse sous la forme établie par WSDL en langage XML.
5. Le client peut maintenant rédiger sa requête pour traiter les données dont il a besoin.
6. Le serveur fait les calculs nécessaires suite à la requête du client, et renvoie sa réponse sous la même forme normalisée.

1.4 Le web sémantique

1.4.1 Définition

Selon le W3C, " le Web sémantique [11] fournit un modèle qui permet aux données d'être partagées et réutilisées entre plusieurs applications, entreprises et groupes d'utilisateurs ". L'expression a été inventée par Tim Berners-Lee (inventeur du Web et directeur du W3C), qui supervise le développement des technologies communes du Web sémantique. Il le définit comme " une toile de données qui peuvent être traitées directement et indirectement par des machines pour aider leurs utilisateurs à créer de nouvelles connaissances ". Pour y parvenir, le Web sémantique met en œuvre le Web des données qui consiste à lier et structurer l'information sur Internet pour accéder simplement à la connaissance qu'elle contient déjà.

1.4.2 Les ontologies

1.4.2.1 Définition

Ainsi, les ontologies offrent un remède aux contraintes du web classique. Pour (Blois et Escobar et Choren) une ontologie [12] est définie comme "la science ou l'étude de l'être", qui est un ensemble de termes et de connaissances, y compris le vocabulaire, la sémantique des interconnexions, et des simples règles d'inférence et de logique pour certains domaines populaires. Par exemple, l'ontologie d'une recette de cuisine comprend les ingrédients, la façon de mélanger et de combiner les ingrédients, les attentes des produits qui seront mangés ou bus, et ainsi de suite. Les ontologies sont utilisées par les personnes, les bases de données et les applications qui doivent partager l'information du domaine (la médecine, l'imagerie, la réparation automobile, etc.). Une importante réalisation est le développement d'une nouvelle génération de langages web, qui permettent la création d'ontologies de n'importe quel domaine, et leurs instanciations dans la description de sites web spécifiques. Les ontologies sont explicitement mentionnées dans un langage formel, où plusieurs ont été proposés dans la littérature. Cependant,

les travaux les plus communs tournent autour du langage RDF [13] et OWL-S [14] :

RDF ou Resource Description Framework représente la première étape vers un langage web pour les ontologies. C'est un modèle de données permettant de décrire les ressources sur le web. Plusieurs outils ont été développés pour le RDF, et un nombre considérable d'applications du web sémantique utilisent ce langage, avant qu'un autre langage plus expressif soit développé.
OWL-S vise à surmonter les limitations d'expressivité du RDF, en fournissant des moyens de décrire les relations entre les classes (disjonction, union, intersection), la cardinalité et les caractéristiques des propriétés (la transitivité, la symétrie), l'égalité, etc. L'OWL-S fournit des composants pour encoder des connaissances en forme d'ontologie, en utilisant le langage XML.

Plusieurs propriétés sont attendues du langage OWL-S :

- Découverte automatique tout en respectant certaines contraintes spécifiées par le client.
- Invocation automatique par un programme informatique ou un agent, en utilisant seulement une description déclarative du service, contrairement à l'agent qui a été préprogrammé pour être en mesure d'appeler ce service.
- Composition et interopérabilité automatique des services web pour effectuer des tâches complexes, en se basant sur une description à haut niveau de la tâche à faire.

Avec l'OWL-S, les informations nécessaires pour choisir et composer des services seront encodées aux sites web de ces services. Ainsi, un logiciel peut être écrit pour manipuler ces représentations avec une spécification des objectifs de la tâche.

1.5 Services web sémantique

Une limitation partagée par les standards de description basées sur le langage XML, c'est leur déficit d'exprimer la sémantique des informations. Par exemple, deux descriptions identiques en XML peuvent avoir une signification totalement différente, et dépendront du contexte où elles sont utilisées. Par exemple, un demandeur de services ne sait pas quels sont les services fournis à un moment donné. Dans ce cas, il contacte directement le fournisseur pour récupérer la liste des services. En plus, les fournisseurs et les demandeurs possèdent des connaissances différentes du même service proposé. En effet, pour utiliser un service web, un agent logiciel doit avoir une description interprétable et les moyens pour accéder à ce service. Ainsi, une mise en correspondance de services web est nécessaire pour une découverte efficace, et a besoin d'un modèle de données (metadata) bien riche et flexible, qui n'est pas actuellement supporté par l'UDDI.
Le web sémantique est centré autour des métadonnées, des ontologies et des agents logiciels. En effet, les services web sémantique [15] forment une synergie entre le web sémantique et les services web, et ont le potentiel d'apporter une valeur ajoutée par la découverte autonome, et l'assemblage des services web afin d'accomplir une tâche de domaine.

Parmi les ressources web les plus importantes, sont celles qui offrent des "services". Par service, les auteurs (Nandigam, Gudivada et Kalavala) signifient des sites web qui ne se contentent pas seulement de fournir des informations statiques, mais de permettre d'effectuer une certaine action, comme par exemple la vente d'un produit ou le contrôle d'un périphérique physique. Le web sémantique devrait permettre aux utilisateurs de localiser, sélectionner, utiliser, composer, et de contrôler ces services web automatiquement. Un objectif important est d'établir un cadre dans lequel des descriptions sémantiques sont faites et partagées. C'est pour cela que les sites web devraient être en mesure d'employer une ontologie afin de déclarer et de décrire les services, dits services web sémantique [15].

1.6 La composition des services web

1.6.1 Définition

La composition de services [14] est considérée comme l'une des motivations les plus importantes du paradigme SOA, en effet, La composition est peut être définie comme le processus de combinaison et d'exécution de services en vue d'accomplir un objectif donné.

1.6.2 Les types de composition des services web

La plupart des travaux portant sur la composition de services web reconnaissent deux types de composition : l'orchestration et la chorégraphie de services. Cependant, l'orchestration et la chorégraphie sont des moyens de concevoir la composition. Afin de choisir l'un ou l'autre de ces types de composition (orchestration ou chorégraphie), le concepteur de systèmes doit prendre en compte différents paramètres. Les paragraphes ci-dessous, décrivent chacun de ces types.

1. Orchestration :

   L'orchestration [16] est un ensemble de processus exécutés dans un ordre prédéfini afin de répondre à un but. Ce type de composition permet de centraliser l'invocation des services web.

   Un moteur d'exécution est un service Web jouant le rôle de chef d'orchestre, il gère l'enchaînement des services web par une logique de contrôle.

   Pour concevoir une orchestration de services web, il faut décrire les interactions entre le moteur d'exécution et les services web. Ces interactions correspondent aux appels, effectués par le moteur, d'action(s) proposée(s) par les services web composants.

   L'orchestration de services web consiste en la programmation d'un moteur qui appelle un ensemble de services web selon un processus prédéfini. Ce moteur définit le processus dans son ensemble et appel les services web selon l'ordre des tâches d'exécution [17].

   En d'autres termes, l'orchestration de services web exige de définir l'enchaî-

nement des services web selon un canevas prédéfini, et de les exécuter selon un script d'orchestration. Ces derniers (le canevas et le script) décrivent les interactions entre services web en identifiant les messages, et en spécifiant la logique et les séquences d'invocation. Le module exécutant le script d'orchestration de services web est appelé un moteur d'orchestration. Ce moteur d'orchestration est une entité logicielle qui joue le rôle d'intermédiaire entre les services en les appelant suivant le script d'orchestration.

2. Choregraphie

   La chorégraphie [16] permet de décrire la composition comme un moyen d'atteindre un but commun en utilisant un ensemble de services web. Pour concevoir une chorégraphie, les interactions entre les différents services doivent être décrites. La logique de contrôle est supervisée par chacun des services intervenant dans la composition. L'exécution du processus est alors distribuée.

   La description de chaque service Web intervenant dans la chorégraphie inclut la description de sa participation dans le processus. De ce fait, ces services peuvent collaborer à l'aide de messages échangés. Chaque service Web peut communiquer avec un autre service Web par l'intermédiaire d'échange de messages.

   La chorégraphie est aussi appelée composition dynamique. En effet, l'exécution n'est pas régie de manière statique comme dans une composition de type orchestration. Dans une chorégraphie, à chaque pas de l'exécution (i.e. à chaque étape de la composition), un service Web choisit le service Web qui lui succède et implémente ainsi une partie de la chorégraphie. La composition de type chorégraphie n'est pas connue, ni décrite à l'avance [17].

Sur la base des spécifications natifs de deux modèles, on peut dire que dans le modèle d'orchestration il existe un élément central coordinateur (chef d'orchestre) qui appel les services web selon l'ordre des tâches d'exécution, mais dans le modèle chorégraphie, chaque partie (service web) est autonome et dans lequel aucune partie est maître sur toute autre (pas d'élément central).

1.7 Conclusion

Dans ce chapitre, nous avons présenté l'architecture orientée service SOA, ses principes ensuite les services web, leur architecture et leurs fonctionnements, ensuite nous avons parlé sur quelques notions du web sémantique et les services web sémantiques, enfin, nous avons terminé par définir la composition des services web et ses types.

2

Les Agents logiciels et l'intégration avec les services web

2.1 Introduction

Dans ce chapitre, nous présentons une étude sur les agents et systèmes multi agents, nous présentons ensuite la communication entre agents et les différentes plateformes d'agents. Finalement, nous expliquons l'intégration des services web avec les agents, et le but de cette intégration.

2.2 Les agents logiciels

Le terme " agent " ou agent logiciel, a trouvé sa place dans un certain nombre de technologies et a été largement utilisé, par exemple, dans l'intelligence artificielle, bases de données et systèmes d'exploitation. Bien qu'il n'y ait pas de définition unique d'un agent (voir, par exemple, Genesereth et Ketchpel, 1994 ; Wooldridge et Jennings, 1995 ; Russell et Norvig, 2003), toutes les définitions

conviennent qu'un agent est essentiellement un composant logiciel spécial qui dispose d'une autonomie qui fournit une interface interopérable à un système arbitraire et / ou se comporte comme un agent humain, travaillant pour certains clients dans la poursuite de son propre programme. Des agents peuvent interagir les uns avec les autres à la fois indirectement (en agissant sur l'environnement) ou directement (par l'intermédiaire de la communication et négociation). Les agents peuvent décider de coopérer pour le bénéfice mutuel ou peuvent concourir pour servir leurs propres intérêts [18].

2.3 Les Systèmes Multi Agents (SMA)

2.3.1 Définition

En informatique, un système multi agents (SMA) est un système composé d'un ensemble d'agents, où ces agents sont organisés et travaillent ensemble pour résoudre un problème commun, les possibilités de chaque entité (agent) prise séparément ne permettant pas de le résoudre. Chaque agent possède donc des connaissances et savoir-faire limités, ce qui l'oblige à interagir avec d'autres pour mener à bien le projet commun. Les SMA mettent en œuvre des entités aux interactions complexes, comme la coopération, la coordination ou la négociation. Ces trois types d'interactions peuvent brièvement se définir de la manière suivante [2] :

- la coopération : lorsque les agents travaillent ensemble vers un but commun.
- la coordination : consiste à organiser les activités liées à la résolution d'un problème de manière à éviter les interactions néfastes et à exploiter les interactions bénéfiques.
- la négociation : a pour objet la gestion des conflits entre agents, c'est à dire qu'elle consiste à aboutir à un accord acceptable par l'ensemble des agents impliqués.

Ces interactions sophistiquées ne sont possibles que par l'intégration au sein des agents de mécanismes et protocoles de communication de haut niveau (actes de langage). Enfin, les notions de société (également appelée collectivité), d'organisation et les relations entre entités que celles-ci induisent, jouent un rôle crucial dans un SMA.

2.3.2 Les Approches multi agents

En fonction de la taille d'un agent, de sa complexité, de ses connaissances et de son raisonnement, nous pouvons classer les approches multi agents en trois grandes catégories : cognitive, réactive et hybride [19].

– L'approche cognitive : les concepteurs des SMA cognitifs s'inspirent du comportement humain pour définir la structure et le raisonnement d'un agent cognitif. Pour pouvoir anticiper ou expliquer ses actions et celles des autres agents, un agent cognitif doit posséder des connaissances sur les autres agents. Généralement, les systèmes multi agents cognitifs sont composés d'un petit nombre d'agents de grandes granularités, où ils possèdent un minimum de connaissances et des comportements de haut niveau leur permettant de s'organiser, de se regrouper, de coopérer, d'apprendre à coopérer en se servant de leurs expériences, et également de prévoir les résultats de leurs comportements.

– L'approche réactive : les concepteurs de cette approche se sont inspirés des phénomènes biologiques, et tentent de les reproduire par la suite. Les agents sont dotés d'un modèle de comportement du style automate selon un modèle stimulus-réponse. Contrairement aux agents cognitifs, les sociétés d'agents réactifs sont composées d'un nombre considérable d'agents de faible granularité. En général, un agent réactif ignore ses expériences passées car il ne possède pas de processus de raisonnement sophistiqués qui lui permettent de planifier ou d'apprendre. Ces agents sont très rapides dans la prise de décision car ils sont dotés d'un minimum de connaissances et de modèles de raisonnement à base de règles.

Pour être fidèle aux modèles biologiques, les concepteurs d'agents réactifs évitent d'utiliser toute communication directe entre agents. Les agents ont alors recours à l'observation des changements qui affectent l'environnement à travers ses différents états, ce qui assure une transmission indirecte des connaissances entre agents

– L'approche hybride : pour surmonter les faiblesses de chacune des deux approches précédentes (la difficulté de mise en œuvre de l'approche cognitive et le manque de modèles formels dans l'approche réactive). Le postulat de cette approche est de maintenir une certaine réactivité de l'agent en le dotant de composants réactifs et de rendre les autres composants cognitifs pour garantir un raisonnement de qualité. Les fondateurs de cette approche argumentent son intérêt par les avantages qu'elle procure, notamment :
  – un agent hybride a une structure modulaire, ce qui est concrètement recommandé dans le développement de tout processus artificiel pour garantir l'évolution et la maintenance du système.
  – les capacités de traitement d'un agent peuvent être améliorées car ses différents composants peuvent fonctionner simultanément.
  – le composant réactif de l'agent devient plus performant car l'organisation des connaissances d'un agent en partitions permet à chacun des composants réactifs ou cognitifs de manipuler partiellement ou totalement cette connaissance.

2.4 La communication entre agents

Les agents peuvent interagir soit en accomplissant des actions linguistiques (en communiquant entre eux), soit en accomplissant des actions non-linguistiques qui modifient leur environnement. En communiquant, les agents peuvent échanger des informations et coordonner leurs activités. Dans les SMA, deux protocoles principaux ont été utilisés pour supporter la communication entre agent :

– Le langage de communication KQML (Knowledge Query and Manipula-

tion Language)[20] : est un langage extérieur de haut niveau pour les agents, orienté sur l'échange des messages, indépendant de la syntaxe et de l'ontologie du contenu des messages.
Le langage KQML est indépendant aussi du mécanisme de transport (TCP/IP, email, CORBA etc.) et du langage utilisé pour coder le contenu des messages (ex : PROLOG, SQL, etc.).
Le langage KQML spécifie le format des messages échangés par les agents, sans être concerné par le format de l'information transportée. Un message KQML peut être vu comme un objet, défini par l'information clé, la 'performative', (la classe) et un nombre .

– Le langage de communication ACL : Ces dernières années, KQML semble perdre du terrain au profit d'un autre langage plus riche sémantiquement ACL. Un langage mis en avant par la FIPA qui s'occupe de standardiser les communications entre agents. ACL est basé également sur la théorie du langage et a bénéficié grandement des résultats de recherche de KQML. Si toutefois, les deux langages se rapprochent au niveau des actes du langage, il n'en est rien au niveau de la sémantique et il semble qu'un grand soin a été apporté au niveau de ACL. Dans notre système, nous proposons l'utilisation du langage ACL pour formuler les messages échangés entre les différents agents.

2.5 Les plateformes d'agents

Il existe une multitude de plateformes multi agents dédiées à différents modèles d'agent. Les plateformes fournissent une couche d'abstraction permettant de facilement implémenter les concepts des systèmes multi agents. D'un autre côté, elle permet aussi le déploiement de ces systèmes. Ainsi, elles constituent un réceptacle au sein duquel les agents peuvent s'exécuter et évoluer. En effet, les plates-formes sont un environnement permettant de gérer le cycle de vie des agents et dans lequel les agents ont accès à certains services. Comme le choix

d'une plateforme d'agent a une grande influence sur la conception et la mise en œuvre des agents, FIPA a produit les normes qui décrivent comment une plateforme d'agent devrait être. Ces normes existent pour assurer une conception uniforme des agents indépendamment de la plateforme.

2.5.1 La plateforme ZEUS

Zeus [21] est une plateforme dédiée pour la construction rapide d'applications à base d'agents collaboratifs. Elle se prête bien aux systèmes économiques qui utilisent des applications de planification ou d'ordonnancement. Pour implémenter les agents collaboratifs, Zeus se base principalement sur les concepts agents, buts, tâches (que les agents doivent réaliser pour atteindre leurs buts) et faits (qui représentent les croyances des agents). Un agent dans Zeus est constitué en trois couches : la couche de définition, qui contient les capacités de raisonnement et des algorithmes d'apprentissage, la couche organisationnelle, qui contient la base de connaissances et des accointances de l'agent, et la couche de coordination, qui définit les interactions avec les autres agents. Zeus propose aussi un ensemble d'agents utilitaires (serveur de nommage et facilitateur) pour faciliter la recherche d'agents.
Zeus fournit un environnement de développement d'agents grâce à un ensemble de bibliothèques Java que les développeurs peuvent réutiliser pour créer leurs agents.

2.5.2 La plateforme JADE

La plateforme JADE (Java Agent DEvelopment framework) [18]est certainement celle qui est la plus utilisée par la communauté des systèmes multi agents. JADE permet de développer et d'exécuter des applications distribuées basées sur le concept d'agents et d'agents mobiles. Elle est compatible avec le standard FIPA. Les agents dans JADE sont implémentés selon 6 propriétés :

– Autonomie : les agents ont leurs propres threads de contrôle qui leur permet de contrôler leurs actions, de prendre leurs propres décisions afin de réaliser

leurs buts mais aussi de contrôler leur cycle de vie.

- Réactivité : les agents peuvent percevoir les évènements de leur environnement et réagissent en fonction de ces évènements.
- Aspects sociaux : les agents exhibent des aspects sociaux qui leur permettent de communiquer et d'interagir entre eux. La communication se fait à travers le passage de messages asynchrones. La communication est considérée comme un type d'actions et peuvent de ce fait intégrer un plan d'actions. Les messages ont une sémantique et une structure définis par le standard FIPA.
- Dynamicité : les agents ont la possibilité de découvrir dynamiquement d'autres agents et de communiquer avec eux.
- Offre de service : chaque agent offre un ensemble de services, il peut enregistrer ses services et les modifier. Il a aussi la possibilité de chercher des agents qui offrent les services dont il a besoin.
- Mobilité : les agents dans JADE ont la possibilité de se déplacer. Les agents sont implémentés dans des conteneurs et ils peuvent se déplacer.

JADE assure la sécurité en offrant aux applications des systèmes d'authentification qui vérifient les droits d'accès des agents. L'implémentation de JADE est basée sur Java. La plateforme peut être répartie sur un ensemble de machines et configurée à distance. La configuration du système peut évoluer dynamiquement puisque la plateforme supporte la mobilité des agents .
Dans le chapitre 3 nous allons utiliser la plateforme JADE.

2.5.3 La plateforme MADKIT

La plateforme MadKit (Multi-Agents Developement kit) [22] est développée à l'Université de Montpellier II. Bien qu'elle puisse supporter le développement de divers systèmes, elle semble bien adaptée pour les applications de simulation. La plate forme MADKIT est basée sur les concepts agent, groupe et rôle. Ces concepts sont appliqués de la manière suivante :

- L'agent : il décrit comme une entité autonome communicante qui joue des

rôles au sein de différents groupes. La faible sémantique associée à l'agent est volontaire l'objectif étant de laisser le choix au concepteur pour choisir l'architecture interne appropriée à son domaine applicatif.

- Le groupe : chaque agent peut être membre d'un ou plusieurs groupes. Il sert à identifier la structure organisationnelle d'un système multi agent usuel.
- Le rôle : il est considéré comme une représentation abstraite d'une fonction ou d'un service. Chaque agent peut avoir plusieurs rôles et un même rôle peut être tenu par plusieurs agents. Les rôles sont locaux aux groupes.

MadKit fournit une API permettant la construction d'agent en spécialisant une classe d'agent abstraite. Les agents sont lancés par le noyau de MadKit, qui propose notamment les services de gestion des groupes et de communication. L'échange des messages se fait à travers le rôle que l'agent est en train de jouer .

2.5.4 La plateforme AgentBuilder

AgentBuilder [23] est une suite intégrée d'outils permettant de construire des agents intelligents. Cette plate-forme est adaptée pour tous types de systèmes. L'élaboration du comportement des agents se fait à partir du modèle BDI. KQML est utilisé comme langage de communication entre les agents. AgentBuilder est composé d'une interface graphique et d'un langage orienté agent permettant de définir des croyances, des engagements et des actions. Il permet également de définir des ontologies et des protocoles de communications inter-agents. Les agents sont décrits avec le langage Radl (Reticular Agent Definition Language), qui permet de définir les règles du comportement de l'agent. Les règles se déclenchent en fonction de certaines conditions et sont associées à des actions. Les conditions portent sur les messages reçus par l'agent tandis que les actions correspondent à l'invocation de méthodes Java. Il est aussi possible de décrire des protocoles définissant les messages acceptés et émis par l'agent.
AgentBuilder propose l'utilisation de la méthode OMT durant la phase d'analyse. OMT permet de spécifier des objets du domaine et les opérations qu'ils peuvent effectuer. A partir de cette spécification une ontologie du domaine est

créée. Des outils graphiques sont fournis pour effectuer ces tâches. Durant la phase de conception, les agents sont identifiés et des fonctionnalités leur sont assignées. Leurs rôles et leurs caractéristiques sont définis ainsi que les protocoles d'interaction auxquels ils participent. Durant la phase de développement, le comportement de l'agent est défini ainsi que ses croyances, intentions et capacités .

2.6 Intégration de services web et agents

La composition des services web représente un axe de recherche émergeant. Mais avec le développement des technologies du Web sémantique, les techniques de composition sont devenues essentiellement sémantiques. Ainsi, des agents logiciels peuvent être développés afin de rendre la composition des services web dynamique et automatique.
De nombreux chercheurs dans le domaine affirment que les services web seuls sont considérés comme passifs, tandis que les agents peuvent fournir des alertes et des mises à jour lorsque de nouvelles informations deviennent disponibles. Malheureusement, ces technologies ont été à l'origine développées séparément avec différentes normes et caractéristiques. En conséquence, leur intégration devient importante dans ce contexte.
Notre idée est d'exploiter les capacités proactives d'interaction des agents afin d'améliorer la composition des services web.
Avec ce paradigme, les composants logiciels où chacun représente un service et un agent en collaboration, vont interagir pour fournir des services unis. Ceci correspond bien avec la prédiction de (Huhns and Singh) "Les agents deviendront une partie essentielle de la plupart des applications web, servant comme une colle qui permet à un système aussi grand que le web d'être maniable et viable" [24].

Les services web sont de plus en plus utilisés pour fournir des comportements actifs sur internet, et promettent aux utilisateurs finaux des avantages qui ont été

associés, dans des travaux antérieurs, avec des systèmes multi agents. Il est donc naturel de considérer quels types de relations existent ou doivent exister entre les agents et les services web. Nous argumentons qu'ils sont des composants distincts, bien qu'ils puissent partager des buts communs. Dans la littérature, l'idée de composer des services web à bases d'agents logiciels a reçu une attention considérable.

L'intégration des services web et agents logiciels apporte un avantage évident : relier les domaines d'application en permettant à un service web d'appeler un service d'agent et vice versa. Toutefois, cette interconnexion est plus qu'une simple invocation inter-domaines ; elle permettra également des compositions complexes de services d'agent et de services web à créer, gérés et administrés par des agents contrôleurs. Plusieurs arguments ont été mis en place pour soutenir ce cas, y compris celles qui sont faites par Laukkanen et Helin (2003) [25] et W3C (2004), qui sont tous pris en charge par la déclaration sans équivoque dans la spécification Web Services Architecture de la (W3C) qui exprime clairement l'idée que les agents logiciels sont les programmes en cours d'exécution qui animent les services web à la fois pour les mettre en œuvre et d'y accéder en tant que ressources informatiques qui agissent au nom d'une personne ou d'une organisation.
Fournir un moyen d'interopération entre les agents de JADE et de services web est l'objectif du WSIG, qui est conçu pour encapsuler les fonctionnalités requises pour relier les deux domaines, tout en assurant une intervention humaine minimale et de l'interruption de service. Du point de vue des agents, les services web sont des entités de calcul qui peuvent être appelés à effectuer une ou plusieurs compositions [18].

2.7 Conclusion

Dans ce chapitre, nous avons présenté une étude sur les agents et systèmes multi agents, ensuite nous avons décrit la communication entre agents et aussi les différentes plateformes d'agents.

Au final, nous avons expliqué l'intégration des services web avec les agents, nous avons expliqué que le but de cette intégration c'est de rendre la composition des services web dynamique et automatique.

3

Mise en œuvre

3.1 Introduction

Dans ce chapitre nous commençons par définir les architectures de JADE et WSIG sur lesquels nous sommes basés pour mettre en œuvre notre prototype " Service voyage ", qui combine entre les services web et les agents logiciels afin de composer des services web, ensuite nous présentons le cas d'étude de notre prototype, où nous allons montrer les différents diagrammes de séquences et de classes des agents constituant le prototype, enfin nous terminons par les configurations nécessaires de notre prototype.

3.2 Architecture de base de JADE et WSIG

Dans cette section nous allons présenter la mise en œuvre de l'architecture logicielle, où nous présentons la plateforme JADE pour la mise en œuvre des agents, ainsi que le WSIG, qui est un composant de JADE.

3.2.1 JADE

Le Java Agent DEvelopment framework (JADE) [3] est un middleware écrit en Java et conforme aux spécifications de la FIPA. Cet environnement simplifie le développement d'agent en fournissant les services de base définis par la FIPA, ainsi qu'un ensemble d'outils pour le déploiement. La plateforme JADE peut être répartie sur un ensemble de machines et configurée à distance, grâce à un mécanisme de migration d'agent au sein de la même plateforme.
La plateforme JADE contient :

- Un environnement d'exécution, où les agents JADE peuvent évoluer. Il doit être actif sur un hôte donné et ainsi un ou plusieurs agents peuvent être exécutés sur cet hôte.
- Une librairie de classes que les programmeurs utilisent pour développer leurs agents.
- Une suite d'outils graphiques qui permettent l'administration et la supervision des activités des agents en exécution. la figure 3.1 nous montre l'architecture de JADE :

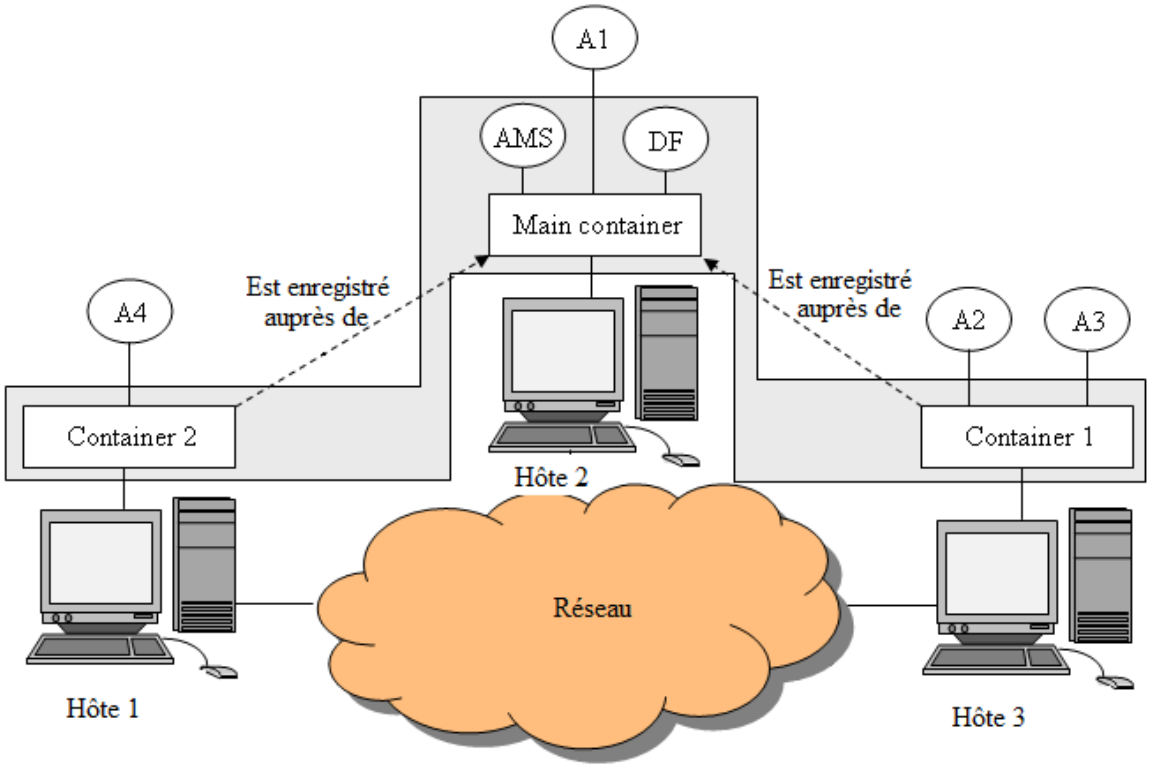

FIG. 3.1 – *Architecture de JADE*[26]

3.2.1.1 Les conteneurs

Chaque instance d'exécution de l'environnement JADE est appelé " Container " (ou conteneur) car il peut contenir plusieurs agents. L'ensemble des conteneurs actifs est appelé une plateforme. Le " Main Container " est l'unique conteneur principal qui doit être toujours actif dans une plateforme et tous les autres conteneurs s'enregistrent auprès de lui, dès qu'ils commencent.

3.2.1.2 AMS and DF

Le conteneur principal " Main Container " contient deux agents spéciaux :

– L'AMS (Agent Management System) qui fournit un service de nommage où il assure que chaque agent dans la plateforme possède un nom unique. Il représente l'autorité dans la plateforme (c'est possible de créer ou de tuer des agents dans des conteneurs distants en le demandant à l'AMS).
– Le DF (Directory Facilitator), offre le service de Pages Jaunes au moyen duquel un agent peut trouver d'autres agents fournissant les services dont il a besoin dans le but d'atteindre son objectif.

Le DF permet aux agents de publier des descriptions d'un ou de plusieurs services qu'ils fournissent afin que d'autres agents peuvent facilement les découvrir et les exploiter. En effet, n'importe quel agent peut à la fois enregistrer (publier) ou rechercher (découvrir) des services. Le DF est conforme avec les spécifications de la FIPA, où chaque plateforme conforme à cette norme devrait avoir un DF par défaut. En effet, un agent qui désire publier un ou plusieurs services, doit fournir au DF une description qui inclut son AID (Identifiant de l'agent), une liste des services fournis et, éventuellement, la liste des langues et des ontologies que d'autres agents doivent utiliser pour interagir avec l'agent concerné. Le DFAgentDescription, ServiceDescription et les classes de propriétés inclus dans le jade.domain.FIPAAgent représentent ces abstractions. Nous avons interfacé le DF avec La librairie JUDDI [27] , qui offre une interface afin d'interagir avec l'UDDI du système. En effet, il existe plusieurs technologies afin d'implémenter les services web, mais dans notre travail, nous avons utilisé les protocoles

standards (UDDI, SOAP et WSDL) pour leurs larges utilisations dans le monde industriel. En effet, un dispositif clairement utile de JADE serait des moyens de rechercher et d'appeler des services directement comme des agents d'application. Le WSIG (Web Service Integration Gateway) satisfait ces besoins en fournissant les moyens d'enregistrer des services dans l'environnement JADE [26].

3.2.1.3 Création des agents

Afin de créer un nouveau type d'agents, il est nécessaire de définir une classe héritant de la classe Agent de JADE.

Cette classe doit implémenter la méthode setup() qui contient l'initialisation de l'agent. Il est possible de spécifier des arguments lors de la création de l'agent. Ces arguments peuvent être obtenus dans la classe Agent à l'aide de la méthode getArguments.

JADE propose différents agents prédéfinis :

- DUMMY AGENT : permet aux utilisateurs d'interagir avec des agents JADE d'une manière personnalisée. L'interface graphique permet de composer et envoyer des messages ACL et maintient une liste de tous les messages envoyés et ACL reçu. Cette liste peut être consultée par l'utilisateur et chaque message peut être consulté en détail ou même édité.
- SNIFFER AGENT : lorsque l'utilisateur décide de renifler un agent ou un groupe d'agents, l'agent Sniffer permet de visualiser l'enchainement des messages entre les agents dans son interface graphique. L'utilisateur peut visualiser chaque message et l'enregistrer sur le disque. L'utilisateur peut également enregistrer tous les messages suivis et les recharger à partir d'un fichier unique pour une analyse ultérieure.
- INTROSPECTOR AGENT : cet outil permet de surveiller et de contrôler le cycle de vie d'un agent en cours d'exécution et de ses messages échangés, à la fois la file d'attente des messages envoyés et reçus. Il permet également de surveiller la file d'attente de comportements [28].

3.2.1.4 Les comportements (Behaviours) du JADE

Pour qu'un agent JADE exécute une tâche, nous avons tout d'abord besoin de définir ces tâches. Les tâches dans JADE (appelées des behaviours ou des comportements) sont des instances de la classe jade.core.behaviours. Pour qu'un agent exécute une tâche on doit lui l'attribuer par la méthode addBehaviour(Behaviour b) de la classe jade.core.Agent. Chaque Behaviour doit implémenter au moins les deux méthodes :

- action() : qui désigne les opérations à exécuter par le Behaviour.
- done() : qui exprime si le Behaviour a terminé son exécution ou pas.

Il existe deux autres méthodes dont l'implémentation n'est pas obligatoire mais qui peuvent être très utiles :

- onStart() : appelée juste avant l'exécution de ma méthode action().
- onEnd() : appelée juste après la retournement de true par la méthode done().

Des fois on a besoin de savoir quel est le propriétaire d'un Behaviour, et cela peut être connu par le membre myAgent du Behaviour en question.

JADE alloue un thread par agent, pour cela un agent exécute un Behaviour à la fois. L'agent peut exécuter plusieurs Behaviours simultanément en choisissant un bon mécanisme de passation d'un Behaviour à un autre (c'est le programmeur qui se charge ce mécanisme et non pas JADE).

1. Les Behaviours simples

   JADE offre trois types de Behaviours simple. Ces Behaviours sont :

   - One Shot Behaviour

     Un One Shot Behaviour est une instance de la classe jade.core.behaviours.OneShotBehaviour. Il a la particularité d'exécuter sa tâche une et une seule fois puis il se termine. La classe OneShotBehaviour implémente la méthode done() et elle retourne toujours true.

   - Cyclic Behaviour

     Un Cyclic Behaviour est une instance de la classe jade.core.behaviours.CyclicBehaviour. Comme son nom l'indique un Cyclic Behaviour exécute

sa tâche d'une manière répétitive. La classe CyclicBehaviour implémente la méthode done() qui retourne toujours false.

– Generic Behaviour

Un Generic Behaviour est une instance de la classe jade.core.behaviours.Behaviour.Le Generic Behaviour vient entre le Oneshot Behaviour et le Cyclic Behaviour de faite qu'il n'implémente pas la méthode done() et laisse son implémentation au programmeur, donc il peut planifier la terminaison de son Behaviour selon ces besoins.

2. Les Behaviours composés

Il est possible de composer des comportements :

– SequentielBehaviour

La logique suivie dans les SequentielBehaviour est simple et intuitive. Le Behaviour commence par exécuter le premier sousBehaviour et lorsque celui-là termine son exécution (sa méthode done() retourne true), il passe au prochain Behaviour, et ainsi de suite.

– FSMBehaviour

FSMBehaviour est une sorte de Behaviour qui implémente un automate à états finis dont chaque état correspond à l'exécution d'un sousBehaviour.

– ParallelBehaviour

Le parallelBehaviour permet d'exécuter plusieurs Behaviours en parallèle. Par parallèle on comprend qu'après avoir invoqué la méthode action() d'un sousBehaviour, on pointe sur le suivant sans attendre que le premier termine son exécution [18].

3.2.1.5 La communication entre agents dans JADE

Pour que plusieurs agents JADE arrivent à collaborer, ils doivent s'échanger des messages. Chaque agent JADE possède une sorte de boîte aux lettres qui contient les messages qui lui sont envoyés par les autres agents. Ces boîtes aux lettres sont sous forme d'une liste qui contient les messages selon l'ordre chro-

nologique de leur arrivée. - Format d'un message JADE
Les agents JADE utilisent des messages conformes aux spécifications de la FIPA (FIPAACL). Les messages JADE sont des instances de la classe ACLMessage du package jade.lang.acl. Ces messages sont composés en général de :

- L'émetteur du message : un champ rempli automatiquement lors de l'envoi d'un message.
- L'ensemble des récepteurs du message : un message peut être envoyé à plusieurs agents simultanément.
- L'acte de communication : qui représente le but de l'envoi du message en cours (informer l'agent récepteur, appel d'offre, réponse à une requête, ...)
- Le contenu du message.
- Un ensemble de champs facultatifs, comme la langue utilisée, l'ontologie, le timeOut, l'adresse de réponse...

3.2.2 WSIG

Le WSIG [3] est un composant de la plateforme JADE, qui tente de traduire les messages utilisés par les services web, à des messages compréhensibles par les agents logiciels. Ainsi, il supporte la pile des protocoles standards des services web (WSDL, SOAP et UDDI).

3.2.2.1 Architecture de WSIG

En effet, comme nous pouvons Voir dans la figure 3.2, le WSIG est une application web composée de deux éléments principaux, le "WSIG servlet" et le "WSIG agent".

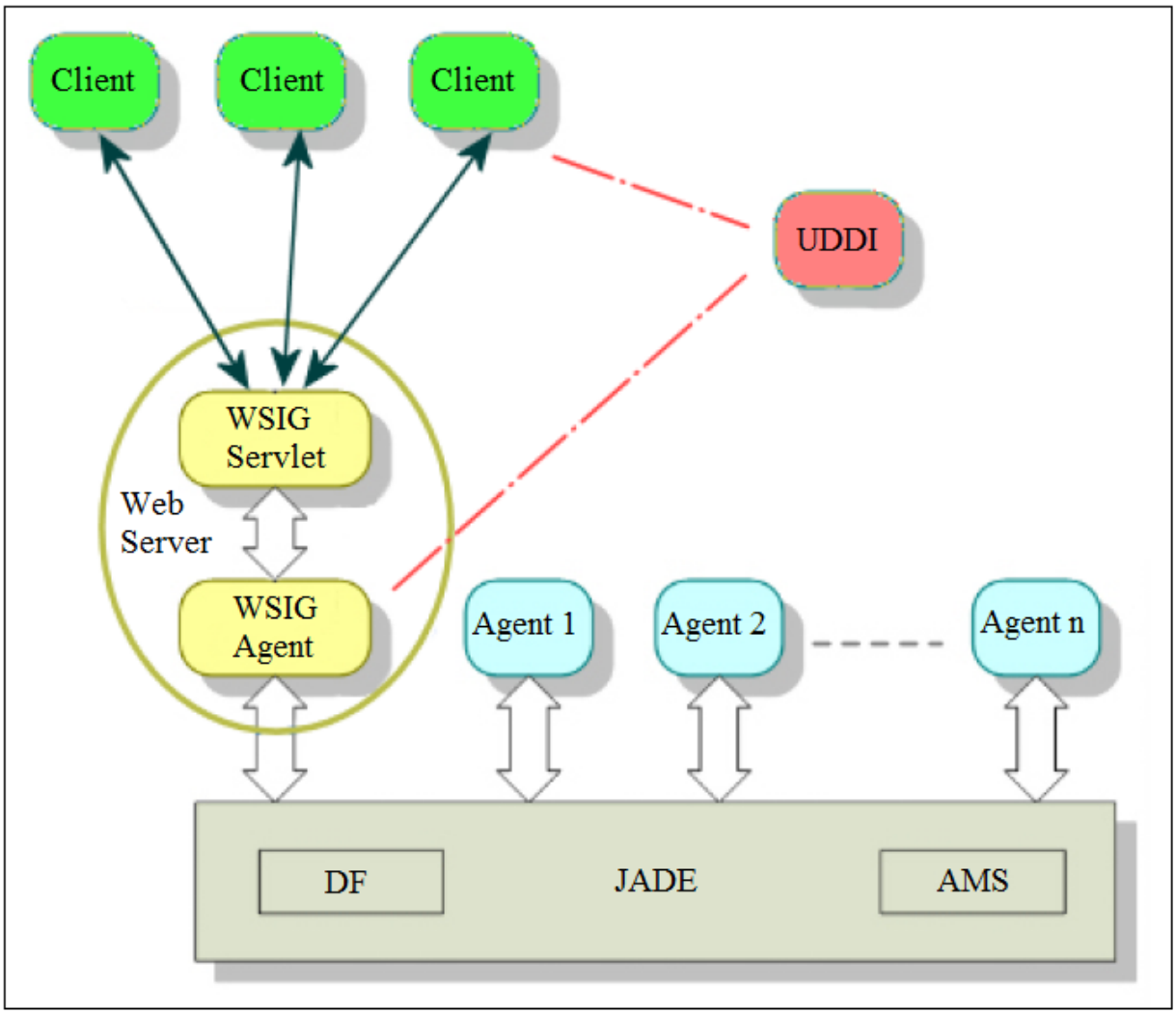

FIG. 3.2 – *Architecture de WSIG* [29]

Le "WSIG servlet" représente la facette vers le monde d'Internet. Il est responsable de :

- Collecter les demandes d'entrées HTTP / SOAP.
- Extraire le message SOAP.
- Préparer l'action d'agent correspondant et le transmettre au "WSIG Agent". Une fois l'action est faite :
  - Convertir le résultat de l'action à un message SOAP.
  - Préparer les protocoles HTTP / SOAP pour la réponse à envoyer au client.

Le "WSIG agent" est la passerelle entre le web et le monde des agents. Il est responsable de :

- Transférer des actions reçues par le "WSIG servlet" aux agents qui sont en mesure de les exécuter, et puis collecter les réponses.
- L'enregistrement dans le DF du JADE afin de recevoir des notifications sur les enregistrements des agents dans le système.
- Créer le fichier WSDL correspondant à chaque service d'agent enregistré par le DF, et publier le service dans l'UDDI, si nécessaire [29].

3.2.2.2 Les opérations de WSIG

Lorsque le WSIG est lancé, un nouvel agent est créé dans le conteneur spécifié. Cet agent est appelé 'WSIG Gateway Agent' ou l'agent passerelle WSIG, qui traitera tous les processus relatifs à la publication des services d'agent comme des services Web et des services Web comme des services d'agent. Le WSIG s'enregistre automatiquement avec la plateforme DF au démarrage.

- Référentiels DF et UDDI

  Le WSIG utilise deux référentiels, DF de l'hébergement de la plateforme JADE et un répertoire UDDI explicitement créé pour être utilisé avec le WSIG.

  Les actions typiques associés à ces deux référentiels sont enregistrer, désenregistrer, modifier et recherche.

  Dans le cas DF où sa visibilité est limitée à la plateforme JADE, ces actions

sont disponibles aux agents JADE seulement.

Le UDDI est plus ou moins le cas miroir de celui-ci où ces actions sont visibles à l'extérieur (de la plateforme JADE), les services Web et les clients de services Web. L'exception dans le cas UDDI est qu'il est également visible à l'Agent passerelle WSIG, mais pas directement à tout autre agent JADE.

Si un service d'agent est radié depuis la plateforme DF, l'agent de passerelle WSIG veillera à ce que le tModel correspondant est retiré du référentiel UDDI. Cela garantit que les deux référentiels restent cohérents.

– Publication de services d'agent de JADE comme services web

  Le WSIG permet à un agent de publier une description de service en tant que service Web. Ce processus est décrit par la figure 3.4.

  L'agent envoie sa description de service à DF pour qu'il sera enregistré dans le DF. Toutefois, la demande d'enregistrement sera automatiquement transmise à l'agent passerelle WSIG qui va faire la traduction de l'ACL en WSDL et créer le tMODEL pour UDDI, ensuite le WSIG va enregistrer le WSDL et le tMODEL dans l'UDDI et enfin il crée un stub pour le AS sur le serveur, voir la figure 3.3.

Quelques mots clés utilisés :

– SD : "Service Description" ou la description du service.
– AS : "Agent Service"
– ACL : "Agent Communication Language", ou le langage de communication utilisé par les agents.
– DF : "Directory Facilitator", ou l'annuaire des agents.
– tModel : Une entrée ou un identifiant du service web.
– stub : Lien entre le service et le système qui l'invoque.

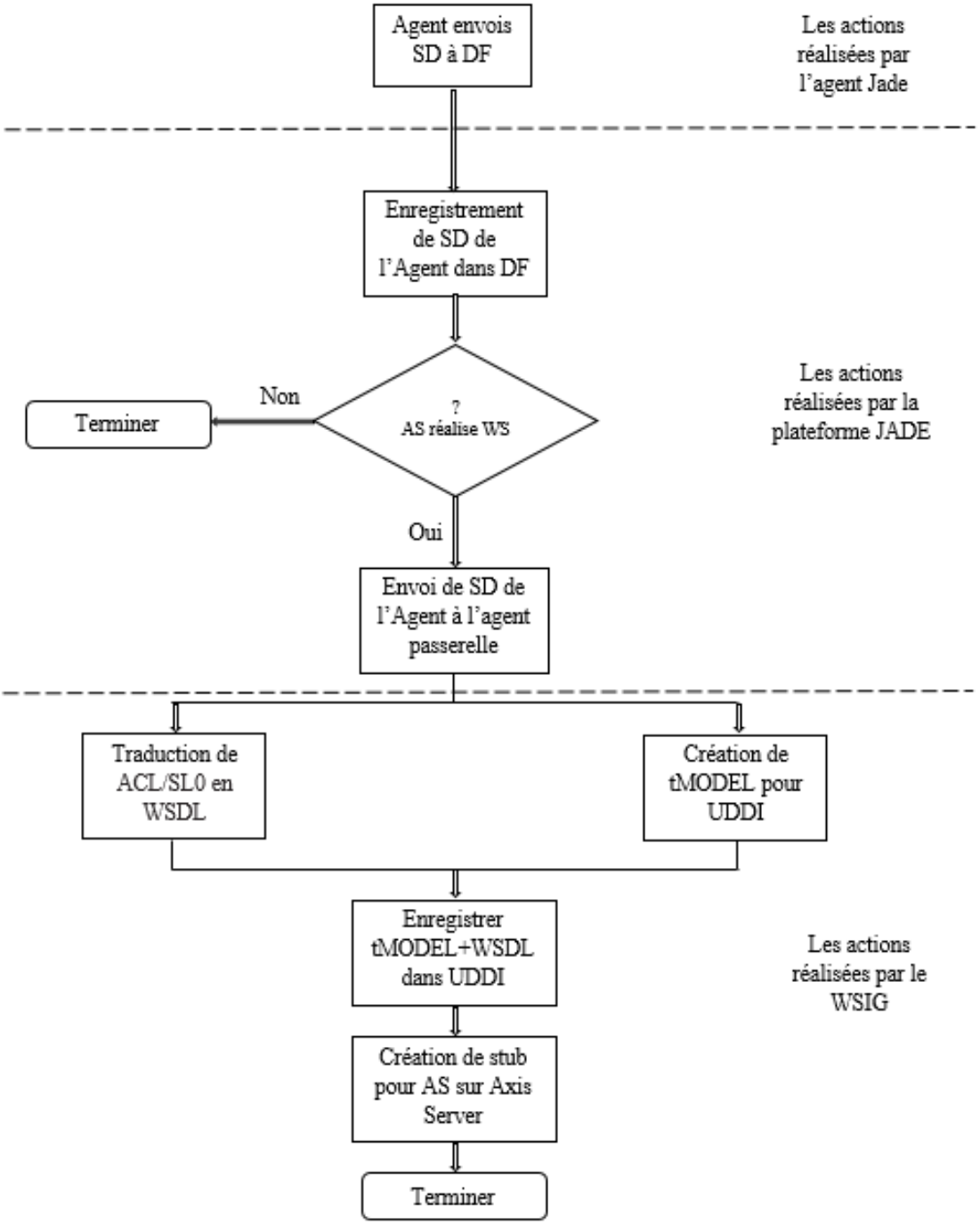

FIG. 3.3 – *Enregistrement de service d'agent avec WSIG* [18]

Les agents JADE publient leurs services dans le répertoire DF (Directory Facilitator) fournissant une structure appelée DF-Agent-Description qui est défini par la spécification FIPA. Un DF-Agent-description comprend un ou plusieurs service-description chacun décrit en fait un service fourni par l'agent d'enregistrement. Un service-description précise généralement, entre autres, une ou plusieurs ontologies qui doivent être connues afin d'accéder au service publié. Les actions de l'agent d'enregistrement qui sont en fait capables d'être exécuter sont celles définies dans les ontologies spécifiées.

Afin d'exposer un service d'agent en tant que service Web, il suffit de définir la propriété 'wsig' à 'true' dans les propriétés du service-description à DF au moment de l'inscription comme ci-dessous :

```
ServiceDescription sd = new ServiceDescription() ;
sd.addProperties(new Property("wsig", "true")) ;
```

– Aperçu de la conversion du DF Service-description en WSDL
  Chaque service-description, qui inclut la propriété wsig définie sur true sera mappé à un WSDL. Toutes les actions définies dans les ontologies spécifiées dans le service-description seront mappés à des opérations WSDL comme le montre la figure 3.4.

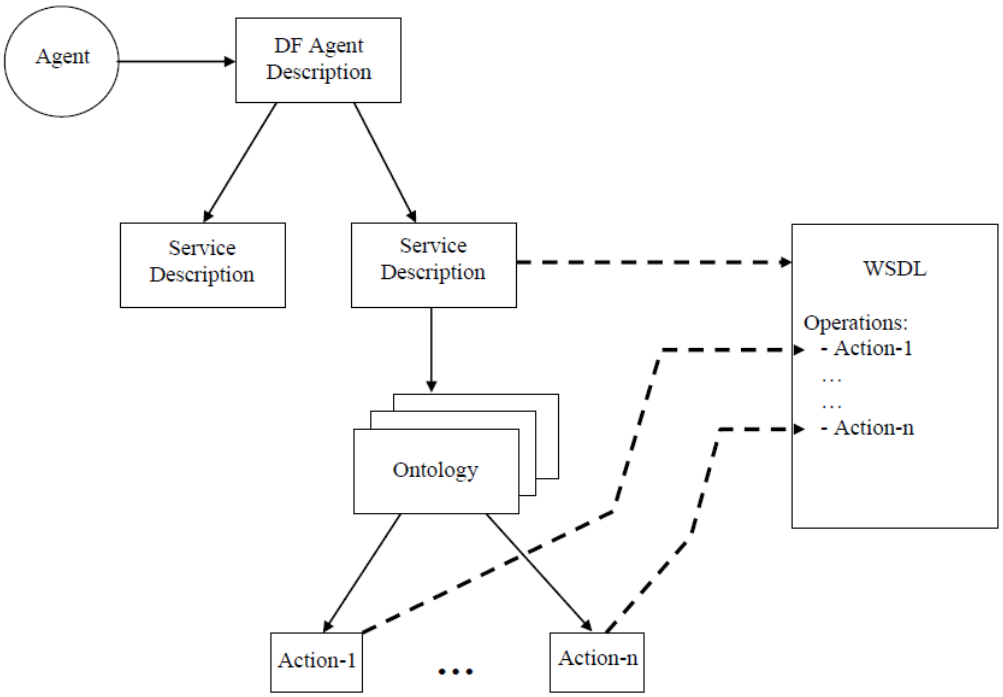

FIG. 3.4 – *Correspondance entre DF Service-Description et WSDL*[29]

3.3 Prototype

Cette section présente un exemple simple, pour illustrer les différentes étapes de développement de notre prototype basé sur des agents avec JADE. Le scénario considéré dans cet exemple comprend trois agents : un agent qui offre le service vol, un autre le service hébergement et le dernier offre le service transport pour le compte des clients, chacun de ces agents retourne un résultat (le cout du service) et propose.

Chaque agent quand il reçoit une requête (demande de service) de la part d'un client, il exécute son code et lui renvoie le résultat. Quand un client choisisse un service composé par exemple (vol et hébergement), l'agent qui offre le service vol reçoit une demande du client, et quand il termine son exécution il envoie un message contenant son résultat à l'agent du service hébergement, ce dernier envoie le résultat final (son résultat + le résultat de du service vol) au client.

Dans la figure 3.5,la figure 3.6 et la figure 3.7 ,nous pouvons voir des exemples

où les services-descriptions des agents " VolAgent ", " HebergementAgent ", " TransportAgent " qui ont chacun d'eux la propriété Wsig définie sur " true " seront mappés à des WSDL.
Les actions " vol ", " hebergement ", " transport " définies respectivement dans les ontologies " VolOntology ", " HebergementOntology ", " TransportOntology " spécifiée dans les services-descriptions appropriés à chacun d'eux seront mappés à des opérations WSDL.

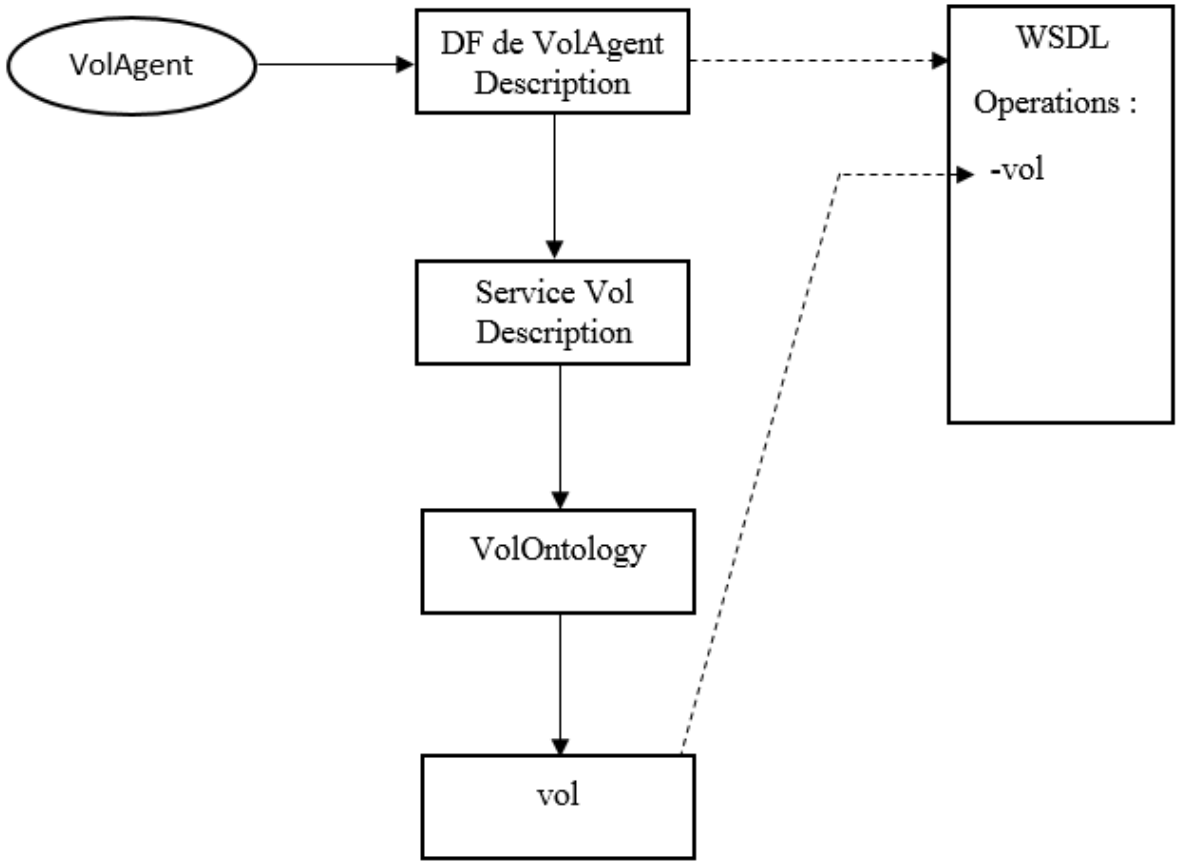

FIG. 3.5 – *Enregistrement du DF de l'agent VolAgent dans le WSDL*

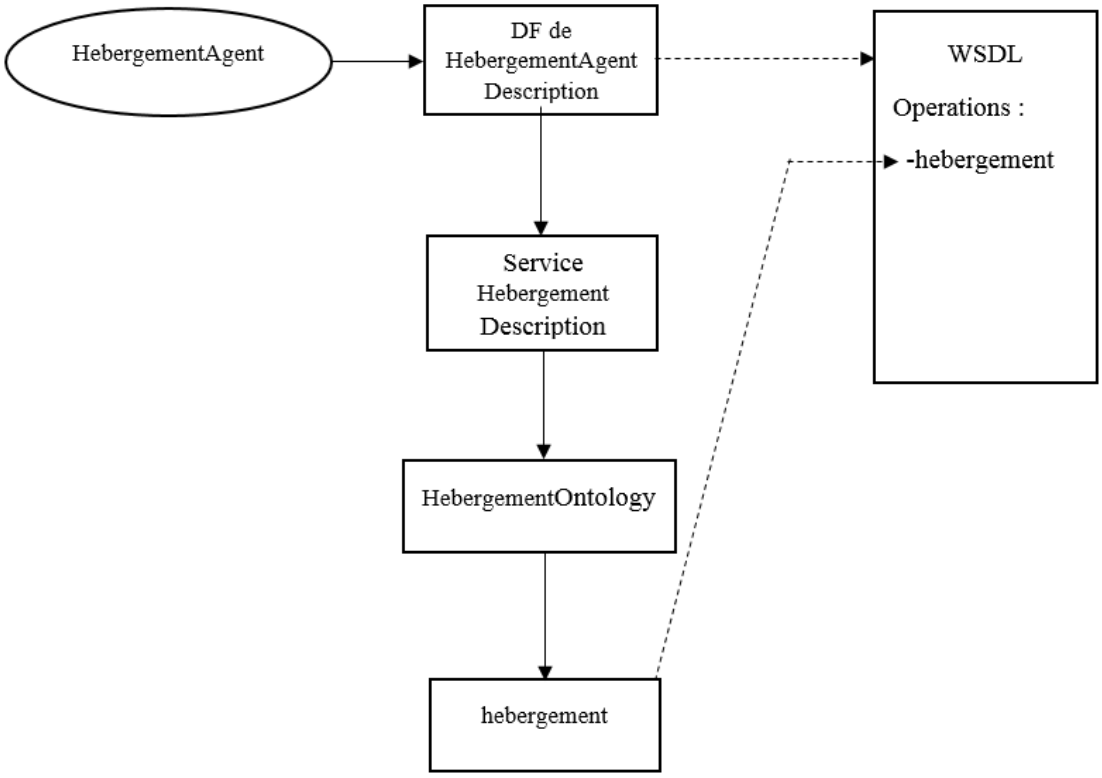

FIG. 3.6 – *Enregistrement du DF de l'agent HebergementAgent dans le WSDL*

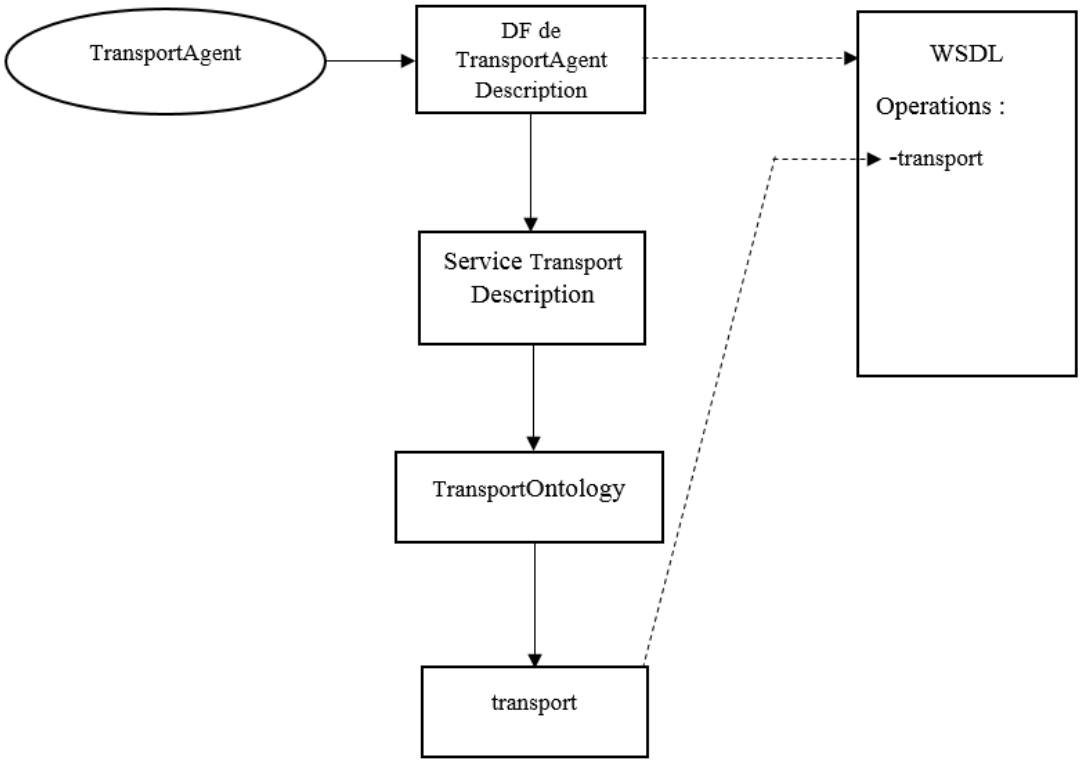

FIG. 3.7 – *Enregistrement du DF de l'agent TransportAgent dans le WSDL*

3.3.1 Les diagrammes des agents

Dans cette partie, nous avons utiliser uml(Unified Modeling Language) [30] pour présenter les diagrammes de séquences et de classes des agents qui constituent notre cas d'étude.

3.3.1.1 Diagrammes de séquences

Dans la figure 3.8, nous présentons le diagramme de séquence de l'enregistrement du service " VolAgent " comme service web par le WSIG. Au départ il faut créer l'agent " VolAgent " (créer ses différents comportements et actions) via le WSIG, ensuite afin d'enregistrer cet agent comme service web d'agent, " VolAgent " envoie un message ACL (demande d'enregistrement) à WSIG, ce dernier va traduire le message ACL en WSDL et enregistre l'agent " VolAgent " comme service web " Service Vol ".

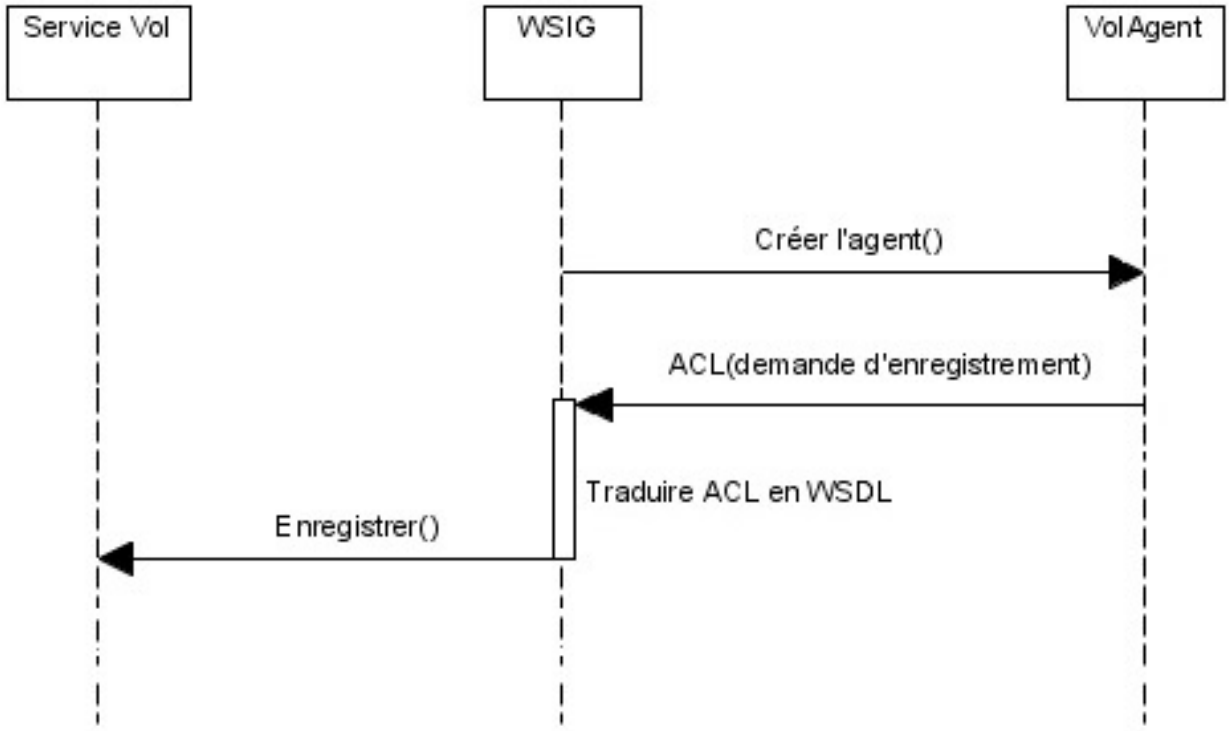

FIG. 3.8 – *Diagramme de séquence d'enregistrement de l'agent VolAgent comme Service Vol*

Dans la figure 3.9, nous trouvons le diagramme de séquence du mécanisme de composition de services. Au début un client demande un service qui se compose de 3 services web via le WSIG, ce dernier vérifie si la composition des trois services existe, si oui alors le WSIG interroge le premier service de la composition, ce dernier envoie un message contenant son résultat au deuxième service de la composition , et ce dernier ce dernier envoie un message contenant son résultat (le résultat des deux services) au troisième et dernier service de la composition qui va par la suite envoie son résultat (le résultat des trois services) au WSIG pour que ce dernier envoie la réponse à la requète du client.

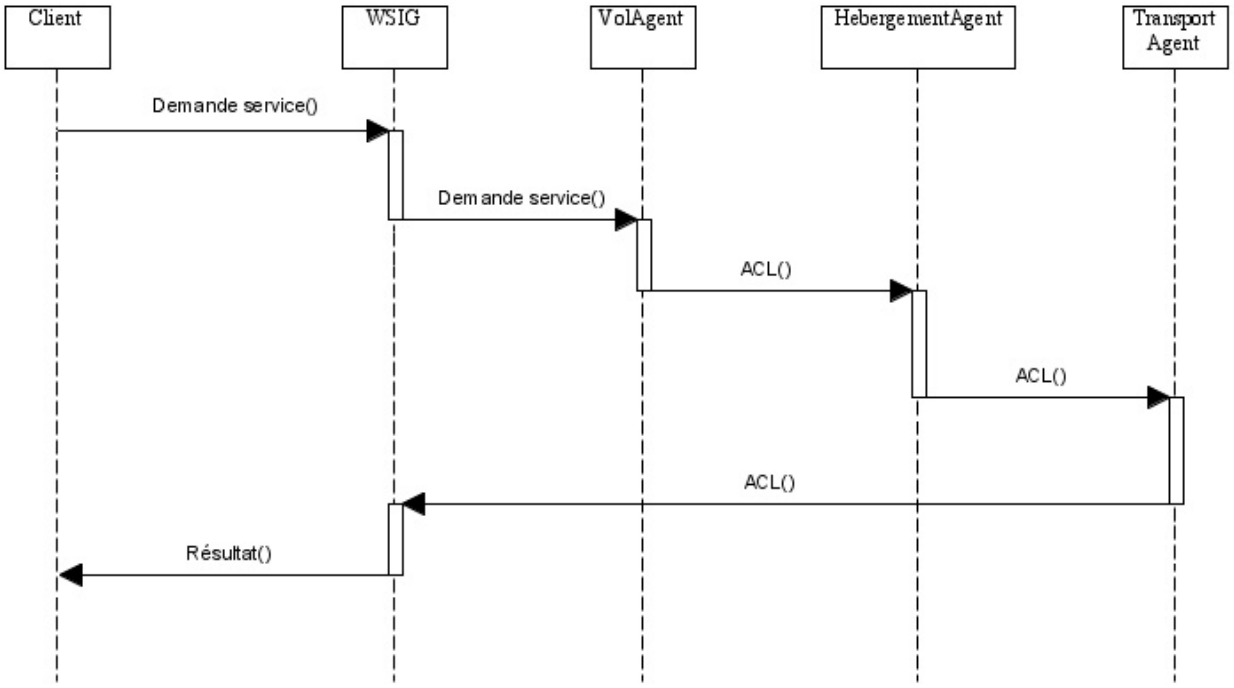

FIG. 3.9 – *Diagramme de séquence de composition de trois services de notre prototype*

3.3.1.2 Diagramme de classe

Dans la figure 3.10, nous présentons le diagramme de classes d'agents " VolAgent ", " HebergementAgent " et " TransportAgent " où ces différents agents héritent de la classe Agent du JADE, et chaque agent à ses propres actions à faire.

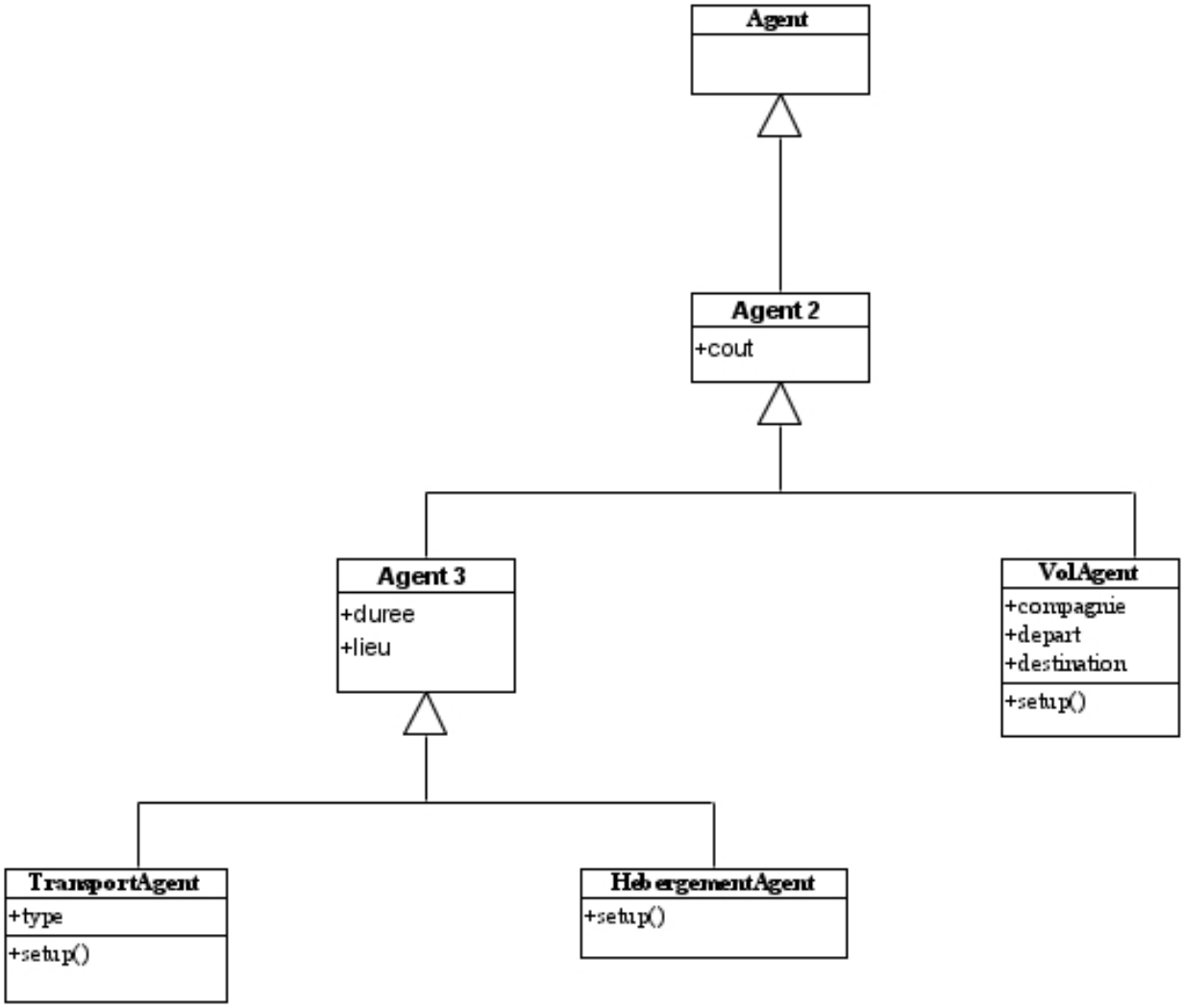

FIG. 3.10 – *Diagramme de classe des agents*

3.4 Configuration du prototype

3.4.1 Outils d'installation

Pour réaliser notre prototype (service voyage), nous avons utilisé les principaux outils suivant :

- JAVA (JDK 1.7) : Le Java Development Kit (JDK) est un environnement de développement pour la construction des applications, des applets et des composants en utilisant le langage de programmation Java. La JDK inclut des outils utiles pour développer et tester des programmes écrits dans le langage de programmation Java et fonctionnant sur la plateforme Java [31].
- JADE (4.4.0) : JADE est un middleware qui facilite le développement de systèmes multi-agents [32].
- WSIG add-on (4.2) : Il permet d'exposer des services d'agent enregistré avec le DF en tant que services web [33].
- Le serveur Apache Tomcat 7.0 :
  Apache Tomcat est un open-source qui a été écrit en langage Java, est le plus populaire de facto Java Web application serveur, un standard pour les développeurs Web en utilisant JSP Servlets [34].
- Eclipse JEE Mars 2 : Est un outil qui permet aux développeurs Java de créer et de réaliser des logiciels et applications web, en s'appuyant principalement sur Java [35].

3.4.2 Etapes de mise en œuvre

Afin de mettre en œuvre notre prototype (service voyage) un ensemble d'étapes doit être suivis :

1. La création des agents :
   la figure 3.11, nous montre une capture d'un code qui illustre comment créer un agent (exemple : VolAgent), en effet afin de créer l'agent VolAgent, il est nécessaire de définir la classe VolAgent héritant de la classe Agent de

JADE.

Cette classe implémente la méthode setup() qui contient l'initialisation de l'agent.

```
package com.tilab.wsig.examples;
import java.util.Date;

public class VolAgent extends Agent {

    public static final String WSIG_FLAGV = "wsig";

    private Logger log = Logger.getLogger(VolAgent.class.getName());
    public static AID myAID = null;
    private SLCodec codec = new SLCodec();
    private Date startDate;


    protected void setup() {
        log.info("A VolAgent is starting...");
        log.info("Agent name: "+getLocalName());
```

FIG. 3.11 – *Code de création de l'agent VolAgent*

la figure 3.12 ,nous montre la plateforme des agents nécessaires au prototype qui sont " VolAgent ", " HebergementAgent ", " TransportAgent ", où chaque agent est enregistré dans un conteneur, et tous ces conteneurs sont enregistrés auprès du conteneur principal qui est le " Main Container ". On distingue aussi l'agent " ControlWSIG-Container " qui est essentiel pour intégrer les agents qui se trouvent dans le " Main Container ", dans la plateforme WSIG.

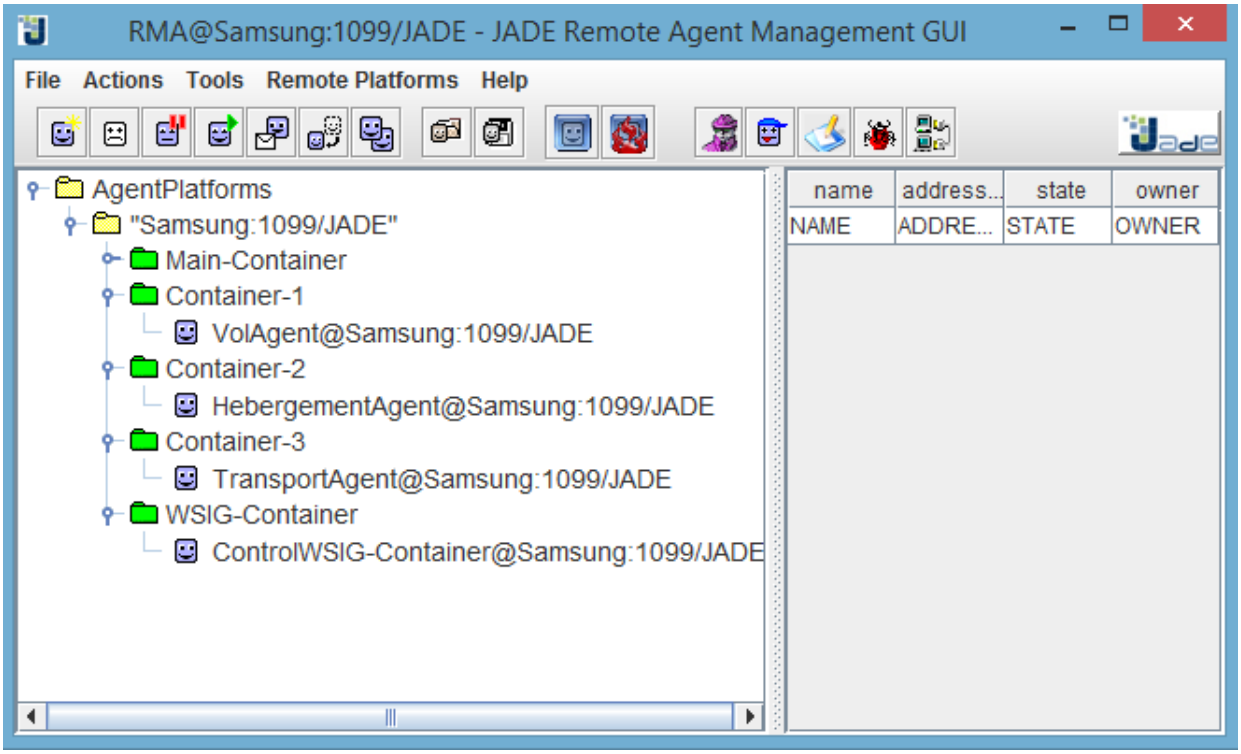

FIG. 3.12 – *La plateforme des agents*

2. La création des ontologies :

   la figure 3.13 ,nous montre une capture d'un code qui illustre la création d'une ontologie (exemple : VolOntology), pour créer cette ontologie il faut définir la classe VolOntology héritant de la classe Ontology de JADE.

```
package com.tilab.wsig.examples;

import jade.content.onto.BasicOntology;

public class VolOntology extends Ontology implements VolVocabulary
{

    private final static Ontology theInstance = new VolOntology();

    public final static Ontology getInstance()
    {
        return theInstance;
    }

    public VolOntology() {
        super(ONTOLOGY_NAME, BasicOntology.getInstance());

        try {
            add(new AgentActionSchema(VOL), Vol.class);
           AgentActionSchema as = (AgentActionSchema) getSchema(VOL);
            as.add(DEPART, (PrimitiveSchema) getSchema(BasicOntology.STRING));
            as.add(DESTINANTION, (PrimitiveSchema) getSchema(BasicOntology.STRING));
            as.add(COMPAGNIE, (PrimitiveSchema) getSchema(BasicOntology.STRING));
            as.add(COMPOSITION, (PrimitiveSchema) getSchema(BasicOntology.STRING));
            as.add(COUT, (PrimitiveSchema) getSchema(BasicOntology.FLOAT));
            as.setResult((PrimitiveSchema)getSchema(BasicOntology.FLOAT));
            as = (AgentActionSchema) getSchema(GETAGENTINFO);
            as.setResult((ConceptSchema)getSchema(AGENTINFO));
```

FIG. 3.13 – *Code de création de l'ontologie VolOntology*

3. Enregistrement du DF de chaque Agent dans des fichiers WSDL, comme le montre la figure 3.14, nous pouvons voir un exemple d'enregistrement du DF de l'agent VolAgent qui sera mappé par la suite à un fichier WSDL.

```
// DF registration
try
{
    DFService.register(this, dfadv);
}
catch (Exception e)
{
    log.error("Problem during DF registration", e);
    doDelete();
}
```

FIG. 3.14 – *Code d'enregistrement du DF de l'agent VolAgent*

4. WSIG assure la cohérence du système en liant un agent à son service correspondant, par le biais du WSIG. Par exemple, un service web 'Vol' est géré par un agent 'VolAgent', assurant son intégration et son invocation dans l'interface physique du système. Ainsi, le WSIG expose les services fournis par les agents, et les publie dans le DF comme étant des services web. Le processus implique la création d'un fichier WSDL pour chaque description de service enregistrée dans le DF, et éventuellement la publication des services exposés dans le répertoire UDDI. Dans la figure 3.15 et la figure 3.16, nous pouvons voir des captures d'écrans de l'interface du WSIG, avec les services enregistrés et leurs opérations, ainsi que les agents qui gèrent ces services.

5. L'envoi de messages entre les agents :

   o L'envoi d'un message :Pour envoyer un message, il suffit de remplir les champs nécessaires (l'ensemble des récepteurs et le contenu du message et l'acte de communication) d'un message JADE, puis d'appeler la méthode send() de la classe Agent, voir la figure 3.17

   Ce message est envoyé à l'agent appelé " HebergementAgent " pour lui dire " hello "

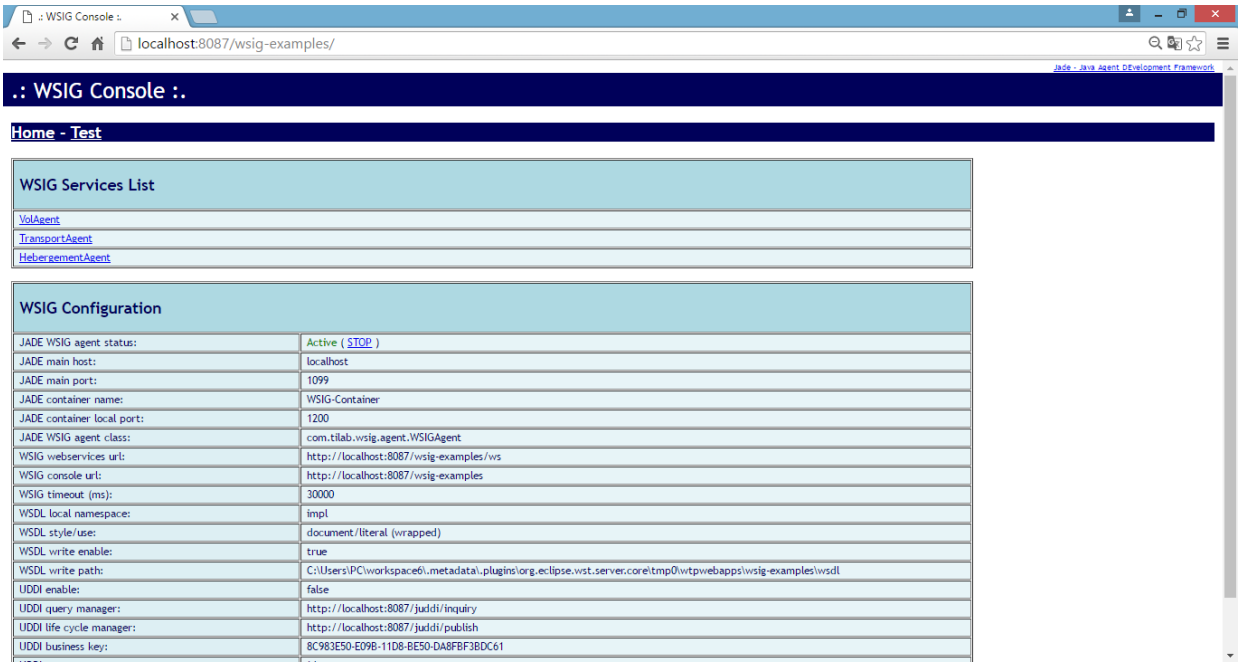

FIG. 3.15 – *Interface de WSIG*

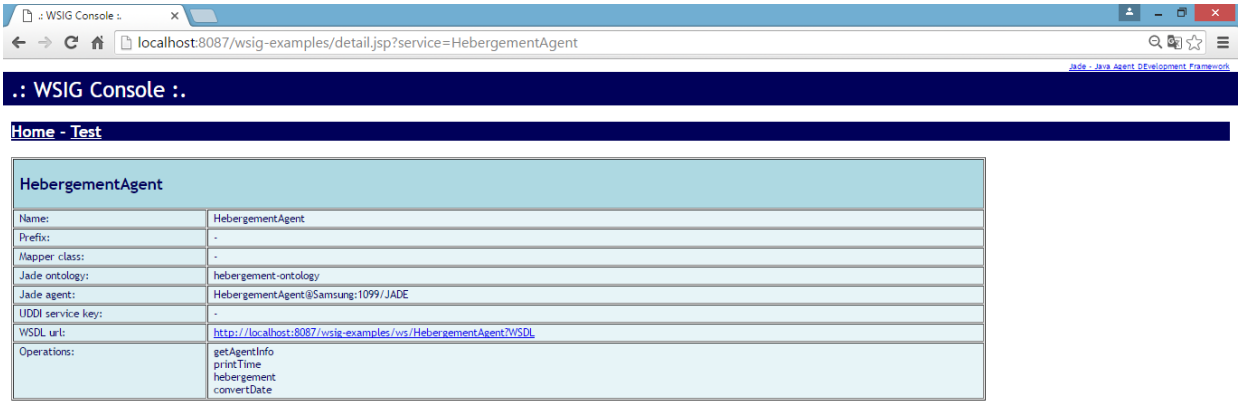

FIG. 3.16 – *Exemple d'un service enregistré dans WSIG*

```
ACLMessage message = new ACLMessage(ACLMessage.INFORM);
  message.addReceiver(new AID("TransportAgent", AID.ISLOCALNAME));
  message.setContent("hello");
  send(message);
```

FIG. 3.17 – *Envoi d'un message ACL*

o La réception d'un message :La réception d'un message est aussi simple que l'envoi, il suffit d'appeler la méthode reveive() de la classe Agent, voir la figure 3.18

```
ACLMessage messageRecu = receive();
```

FIG. 3.18 – *La réception d'un message ACL*

3.4.3 Aperçu de notre prototype

Le client choisit le service qu'il souhaite effectuer, dans la figure 3.19, le client choisit le " service Vol et le service Hebergement et le service Transport " qui est composé de trois services différents, ensuite le client va valider les différents choix qu'il veut, au final, le client aura le résultat final de ses requêtes où il sera affiché dans une fenêtre comme le montre la figure 3.20.

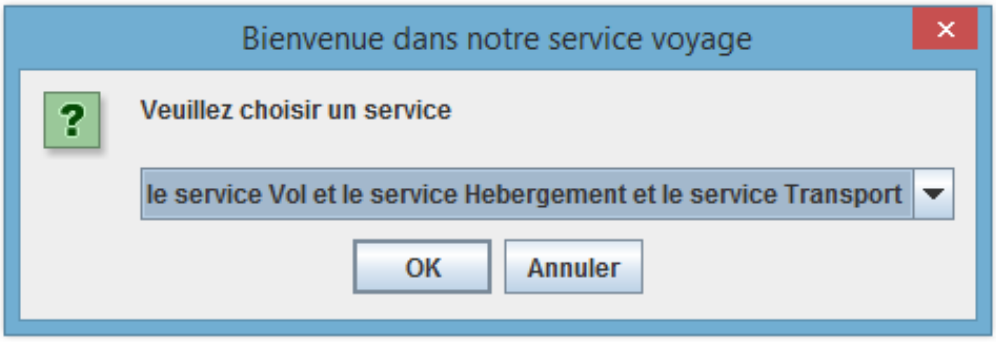

FIG. 3.19 – *Interface du service voyage*

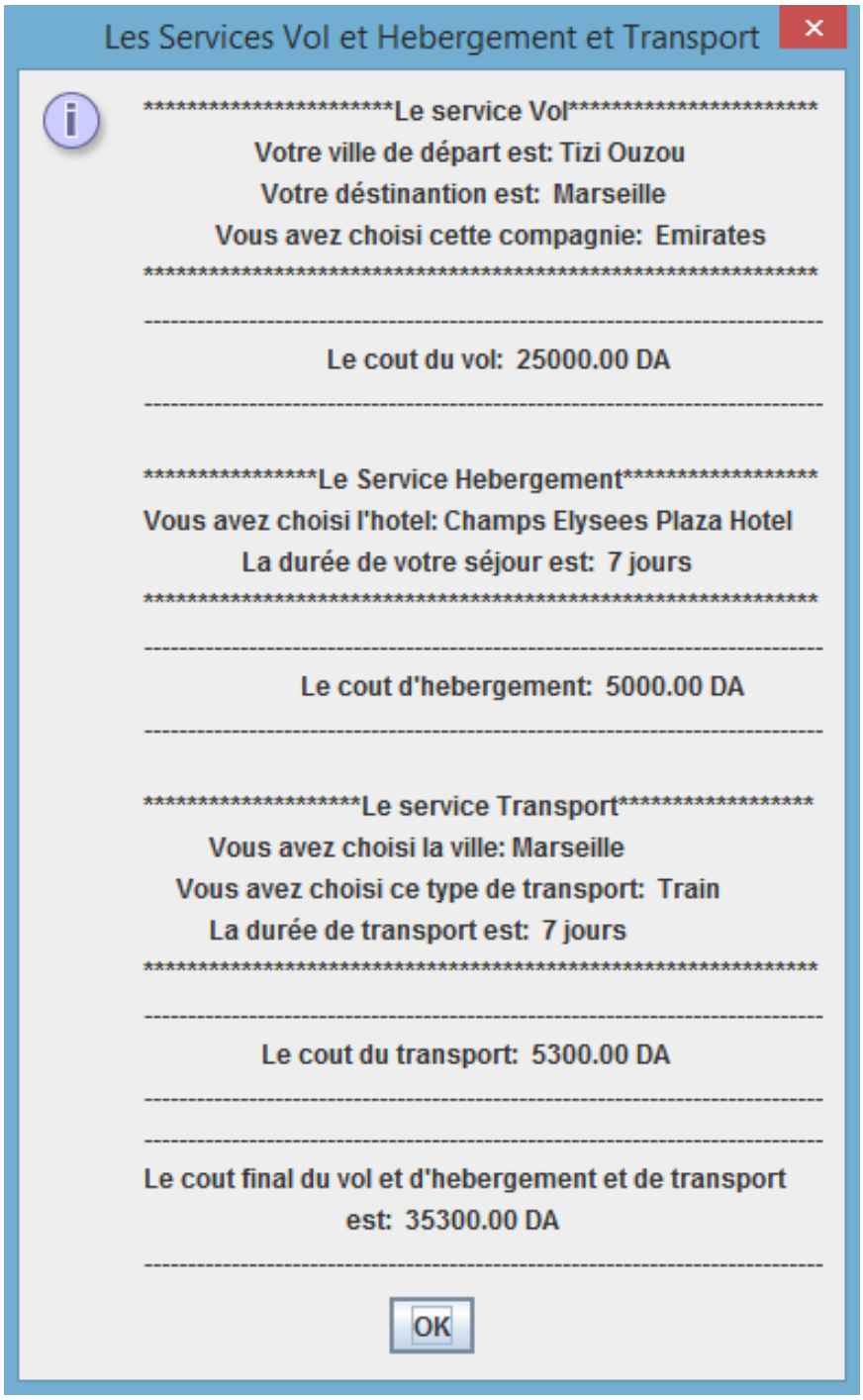

FIG. 3.20 – *Affichage du résultat des requêtes de client*

3.5 Conclusion

Dans ce chapitre, nous avons présenté les architectures de JADE et WSIG sur lesquels nous sommes basés pour mettre en œuvre notre prototype, ensuite nous avons présenté notre prototype, les différents diagrammes de séquences et de classes approprié au prototype, enfin, nous avons terminé par les différentes étapes de configuration et quelques captures de notre prototype.

Conclusion Générale et Perspectives

Avec la croissance rapide d'Internet, la demande d'être connecté et d'avoir accès à des documents et à des services web a énormément augmenté.

La composition des services web représente un axe de recherche. Mais avec le développement des technologies du Web sémantique, les techniques de compositions sont devenues essentiellement sémantiques. Cette sémantique est apportée grâce aux ontologies une des technologies importantes du Web sémantique. Ainsi, des agents logiciels peuvent être développés afin de raisonner sur ces ontologies rendant la composition des services web dynamique et automatique.

Dans ce travail, nous avons présenté les services web et les agents logiciels, ensuite nous avons mis en œuvre une approche de composition de services web basée sur les agents logiciels. Par rapport aux autres approches (Workflow [36], Transformation de graphes [37], Réseaux de petri [38]), cette approche offre des mécanismes efficaces et puissants pour améliorer la composition et la rendre dynamique et automatique.

Comme perspectives, nous souhaitons définir des fichiers de chorégraphie que chaque agent pourra télécharger et se comporter conformément à ce fichier de chorégraphie.

Bibliographie

[1] Booth .D, Haas .H, McCabe .F, Newcomer .E, Champion .M, Ferris .C, Orchard .D, Web Services Architecture (2004), https ://www.w3.org/TR/ws-arch.

[2] Jennings .N, Sycara .K, Wooldridge .M, A Roadmap of Agent Research and Development in Autonomous Agents and Multi-Agent Systems, Kluwer Academic Publishers, Boston, 1998.

[3] http ://jade.tilab.com/

[4] Cavedon .L, Maamar .Z, David Martin .D, Benatallah .B, Extending Web Services Technologies The Use of Multi-Agent Approaches, Springer Science and Business Media, Inc, 2005.

[5] http ://www.edibasics.com/what-is-edi/

[6] http ://www.corba.org/faq.htm .

[7] Bray .T, Paoli .J, Sperberg-McQueen .C .M, Maler .E, Yergeau .F, Extensible Markup Language (XML) 1.0 (Fifth Edition), 2008, https ://www.w3.org/TR/REC-xml/.

[8] Gudgin .M, Hadley .M, Mendelsohn .N., Moreau .J, Frystyk Nielsen .H, Karmarkar .A, Lafon .Y, SOAP Version 1.2 Part 1 : Messaging Framework (Second Edition), 2007, https ://www.w3.org/TR/2007/REC-soap12-part1-20070427/.

[9] Graham .S, Davis .D, Simeonov .S, Daniels .G, Brittenham .P, Nakamura .Y, Fremantle .P, König .D, Zentner .C, Building Web Services with Java

MAKING SENSE OF XML, SOAP, WSDL, AND UDDI, Sams Publishing, 2005.

[10] Chinnici .R, Gudgin .M, Moreau .J, Schlimmer .J, Weerawarana .S, Web Services Description Language (WSDL) Version 2.0 Part 1 : Core Language, 2003, https ://www.w3.org/TR/2003/WD-wsdl20-20031110/.

[11] https ://www.w3.org/2001/sw/.

[12] Blois .M, Escobar .M, Choren .R, Using agents and ontologies for application development on the semantic web, Journal of the Brazilian Computer Society, 2007.

[13] https ://www.w3.org/RDF/.

[14]] Martin .D, Burstein .M, Hobbs .J, Lassila .O, McDermott .D, McIlraith .S, Narayanan .S, Paolucci .M, Parsia .B, Payne .T, Sirin .E, Srinivasan .N, Sycara .K, OWL-S : Semantic Markup for Web Services, 2004, https ://www.w3.org/Submission/OWL-S/.

[15]] Nandigam .J, Gudivada .V, Kalavala .M, Semantic web services. Journal of Computing Sciences in Colleges, 2005.

[16] Barros .A, Dumas .M, Oaks .P , Standards for Web Service Choreography and Orchestration : Status and Perspectives, 2005.

[17] Peltz .C , Web Services Orchestration and Choreography. IEEE Computer, 2003.

[18] Bellifemine .F, Caire .G, Greenwood .D, Developing Multi-Agent Systems with JADE. John Wiley & Sons Ltd, 2007.

[19] Khezami .N, Vers un collecticiel basé sur un formalisme multi-agent destiné à la téléportation collaborative via Internet. PhD thesis, IBISC (Informatique, Biologie Intégrative et Système Complexes) Laboratory- University of Evry Val d'Essonne -Evry, France, 2005.

[20] Finin .T, Fritzson .R, McKay .D, McEntire .R, KQML as an Agent Communication Language University of Maryland Baltimore County, 1994.

[21] Nwama .H .S, Ndumu .D .T, Lee .L .C, Collis .J .C, Zeus : A toolkit for building distributed multi-agent systems. Applied Artificial Intelligence Journal, 1999.

[22] Gutknecht .O, Ferber .J, The madkit agent platform architecture. In Revised Papers from the InternationalWorkshop on Infrastructure for Multi-Agent Systems. SpringerVerlag. London, UK, 2001.

[23] AgentBuilder, an integrated toolkit for constructing multi-agent systems. Acronymics, Inc, 2004.

[24] Huhns .M, Singh .M, Service-oriented computing : Key concepts and principles, IEEE Internet Computing (2005).

[25] Laukkanen .M, and Helin .H, Composing Workflows of Semantic Web Services. In Proceedings of the 1st International Workshop on Web Services and Agent Based Engineering, Sydney, Australia, July 2003, http ://leap.crm-paris.com.

[26] Caire .G, Jade Tutorial, Jade Programming for Beginners. 2009 Telecom Italia S.p.A., Last update : 30 June 2009.

[27] http ://ws.apache.org/juddi.

[28] Bellifemine .F, Caire .G, Trucco .T, Rimassa G, Mungenast .R, Jade administrator's guide. JADE Board, 2005.

[29] Board .J ,Jade web services integration gateway (wsig) Guide. 2008 Telecom Italia, Last update : 12-Mar-2013.

[30] http ://www.uml.org/.

[31] http ://www.oracle.com/technetwork/java/javase/downloads/jdk7-downloads-1880260.html.

[32] http ://jade.tilab.com/download/jade/license/jade-download/.

[33] http ://jade.tilab.com/download/add-ons/.

[34] https ://tomcat.apache.org/index.html.

[35] http ://www.eclipse.org/downloads/packages/eclipse-ide-java-ee-developers/mars2.

[36] https ://www.doc.ic.ac.uk/ ecl1/papers/2002edocworkflow.pdf

[37] http ://www.univ-constantine2.dz/files/Theses/Informatique/Doctorat/Bachtarzi-Faycal.pdf

[38] http ://www.univ-constantine2.dz/files/Theses/Informatique/Doctorat/Chemaa-Sofiane.pdf

Résumé

De nos jours, les services web sont devenus très utilisés notamment par les entreprises pour rendre accessible leurs métiers et leurs données via le Web. La composition des services web est un sujet de recherche émergeant et qui suscite l'intérêt des chercheurs, elle assure l'interopérabilité des applications distribuées, et offre la possibilité de traitement de problèmes complexes même avec des services simples existants tout en coopérant entre eux pour créer des services ayant une valeur ajoutée.

La réalisation de ces buts nécessite l'automatisation de la phase de composition, dans cette optique, nous proposons une méthode de composition basée sur les agents logiciels. Par rapport aux autres approches, cette approche offre des mécanismes efficaces et puissants pour améliorer la composition et la rendre dynamique et automatique. Afin d'illustrer cette approche nous avons mis en œuvre un prototype "Service voyage " à base de la plateforme jade étendu avec le WSIG.

Les résultats obtenus par cette fusion entre les deux technologies services web et agents logiciels sont satisfaisants, en effet la composition des services web est rendue plus dynamique et automatique.

Mots clés **: Agents logiciels, composition de services web, web sémantique, services web sémantique, systèmes multi agents, chorégraphie.**

Abstract

Nowadays, web services have become very used especially by companies to make available their business and data via the Web. The composition of web services is an area of emerging research that is of interest to researchers, it ensures the interoperability of distributed applications, and offers the possibility of handling complex problems even with simple existing services while cooperating to create services with added value.

Achieving these goals requires automating the composition phase, in this context, we propose a method of composition based on software agents. Compared to other approaches, this approach provides efficient and powerful mechanisms to improve the composition and make it dynamic and automatic. To illustrate this approach, we have implemented a prototype "Service Voyage" based on the extended platform with jade WSIG.

The results obtained by the merger between the two web services technologies and software agents are satisfactory, in fact the composition of web services is made more dynamic and automatic.

Keywords **: software agents, web services composition, semantic web, semantic web services, multi agents systems, choreography.**