République Algérienne Démocratique et Populaire
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

Université Djilali BOUNAAMA - Khemis Miliana
Faculté des Sciences et de la Technologie
Département de Mathématiques et d'Informatique

Mémoire de fin d'étude

En vue de l'obtention du diplôme Master en Informatique

Spécialité :

Génie Logiciel et Systèmes Distribués

Thème

Conception et implémentation d'un Cloud Crawler Agent

Présenté par :

M^r MIEN Ange Habathé Yannick
M^lle OUAISSA Asmaa

Devant le jury composé de :

Examinateur 1 : *M^r* Hadj Salah Eddin, MAA *(Université de Khemis Miliana)*
Examinateur 2 : *M^r* Boukadoum Omar, MAA *(Université de Khemis Miliana)*
Encadreur : *M^r Haniche* Fayçal, MAA *(Université de Khemis Miliana)*

Année universitaire 2019/2020

Dédicace

Je dédie ce travail :

À ma famille. Ils sont ma motivation, ne cessant de contribuer à ma réussite par leurs innombrables sacrifices, leurs soutiens et leurs prières.

Ma mère qui par sa tendresse, son soutien moral, mon père qui par ses sacrifices m'ont poussé à faire de mes études une de mes priorités et l'excellence une quête.

À mes frères : Mathurin, Léon et mon petit frère Elisé ainsi qu'à ma sœur Evelyne, qui sont ma force et l'amour qui nous lie est un réconfort.

À mon mentor Caleb Caleb Kabayo, qui est pour moi plus qu'un frère. Ses conseils et ses encouragements permanents m'ont été salutaire durant mon cursus, je me rappellerai toujours de cette phrase qu'il m'a dite une fois : « Ta destiné t'imposera une discipline. »

Merci d'être toujours là pour moi.

MIEN Ange Habathé Yannick

Dédicace

Je dédie ce modeste travail, fruit de notre savoir à toutes les personnes qui ont contribué de près ou de loin à sa réussite.

J'ai l'immense plaisir de dédier ce travail à :

Ceux que j'aime le plus au monde, mes très chers et affectueux parents qui m'ont fait voir la lumière et arriver à ce jour par leur soutien quotidien ainsi que leurs encouragements pour être toujours debout pour dépasser les situations difficiles et qui font tout pour que je sois dans de bonnes conditions morale et physique.

A mon frère MOHAMED OUSSAMA

A ma sœur FATMA ZOHRA et son mari FATHI

Au trésor de ma vie, mon petit neveu le prince ZEYD

A mon binôme YANNICK qui a beaucoup contribué et aidé durant mes études universitaires.

Mes chères amies : SARAH, IMAN, FOUZIA et NIHAD.

Ainsi qu'à toutes mes camarades et à toutes les personnes qui ont partagé de bons moments de sourire et de joie.

OUAISSA Asmaa

Remerciements

Tout d'abord nous remercions le bon Dieu tout-puissant pour son aide dans l'accomplissement de ce travail.

Cette œuvre n'aurait pas pu être possible sans le soutien inconditionnel de nombreuses personnes :

Notamment notre directeur de mémoire monsieur HANICHE Fayçal à qui nous exprimons nos remerciements pour son encadrement et pour sa disponibilité à diriger ce travail.

Nos remerciements à toute l'équipe pédagogique du département Mathématiques-Informatique de l'Université de khemis Miliana, à tout l'ensemble de nos professeurs qui ont participé à notre formation tout le long de ce cursus de master, ainsi qu'à tout nos collègues de classe promotion 2020.

Nous remercions tous nos amis, nos familles et toute personne qui d'une manière ou d'une autres a eu à apporter son plus dans la réalisation de ce travail.

Merci pour votre support et vos encouragements.

MIEN Ange Habathé Yannick et OUAISSA Asmaa

Résumé

Le cloud computing est apparu comme un nouveau paradigme informatique fournissant des services sous forme d'infrastructures, de plates-formes et d'applications flexibles, économique et cet à la demande. Certains de ces services sont déployé sous forme d'API, de web application ou même de service web.

Le manque d'utilisation de langages de description normalisés, et le manque d'un certain nombre de caractéristiques uniques des services cloud, est un défi quant à ce qui s'agit de la recherche de services cloud, et arriver à fournir un ensemble d'éléments rendant la composition d'un service répondant à la demande du client.

L'objectif de notre présent travail, dans lequel nous avons présenté notre système de cloud crawler agent, a été de découvrir des services déployés sur internet en utilisant la technique des robots d'indexation (le crawling) utilisée par les moteurs de recherche pour l'indexation des pages. Les descriptions des services sont crawlées et les données nécessaires sont extraites, catégorisées et par la suite stockées dans une base de données de service. Pour valider notre approche qui utilise une architecture basée sur le paradigme des systèmes multiagents, nous avons utilisé la plateforme JADE. Environ 300 services Cloud de Google ont été collectés ainsi que certains services web, dont les données ont été stockées dans la base de données de service. Cette base de données pourra plus tard être utilisée par d'autres agents pour assurer la composition de service.

Mots clés : Cloud computing, Services cloud, Systèmes multi-agent, composition de service, crawler, robots d'indexation.

Abstract

Cloud computing has emerged as a new IT paradigm providing services in the form of infrastructure, platforms and applications that are flexible, cost-effective and on demand. Some of these services are deployed as APIs, web applications or even web services.

The lack of use of standardized description languages, and the lack of a number of unique features of cloud services, is a challenge when it comes to searching for cloud services, and be able to provide a set of elements making the composition of a service responsive to the client's request.

The objective of our present work, in which we presented our agent cloud crawler system, was to discover services deployed on the internet by using the technique of indexing robots (crawling) used by search engines for indexing pages. The service descriptions are crawled and the necessary data is extracted, categorized and later stored in a service database.

To validate our approach that uses a paradigm-based architecture of multi-agent systems, we used the JADE platform. About 300 Google cloud services were collected as well as some web services, whose data were stored in the service database. This database may later be used by other agents to ensure service mix.

Key words : Cloud computing, Cloud service, multi agent systems, service composition, crawler, crawler-agent.

Table des matières

Table des figures **vi**

Liste des abréviations **vii**

Introduction générale **1**

1 Cloud computing **2**
1.1 Introduction 2
1.2 Qu'est-ce que le Cloud computing ? 2
1.3 Émergence du Cloud computing 3
1.3.1 Les modèles de calcul 4
1.3.2 Les Bases technologiques 4
1.4 Le modèle de cloud computing 6
1.4.1 Les caractéristiques du cloud 6
1.4.2 Les modèles de Service 7
1.4.3 Les modèles de déploiement 9
1.5 Quelques acteurs du cloud computing 10
1.5.1 Cloud customer 10
1.5.2 Service provider 10
1.5.3 End-user (utilisateur final) 10
1.5.4 Infrastructure provider (Fournisseur de service) 10
1.6 Cloud computing : Avantages, limites et Challenges 11
1.6.1 Avantages 11
1.6.2 Limites et Challenges 11
1.7 Conclusion 12

2 Services Web, Services Cloud **13**
2.1 Introduction 13
2.2 Services Web 13
2.3 Architecture des services web 14
2.3.1 Langage de desciption WSDL 15
2.3.2 UDDI 16

2.3.3 SOAP 16
2.4 Service web sémantique 16
2.5 Services Cloud 17
2.6 Relation entre Services web et Services Cloud 17
2.7 Les méthodes de livraison de Services Cloud 18
2.8 Langages de description de services 19
2.8.1 WSDL 19
2.8.2 WSDL-S 19
2.8.3 OWL-S 19
2.8.4 L'OpenAPI 20
2.9 Les propriétés d'un Service 21
2.9.1 Propriétés fonctionnelles 22
2.9.2 Propriétés non-fonctionnelles 22
2.10 Sélection des services web 23
2.11 La découverte des services 24
2.11.1 La découverte des Services Web 24
2.11.2 La découverte des Services Cloud 24
2.11.3 Les différentes approches de découvertes 25
2.11.4 Les mécanismes de découvertes de services 25
2.12 Composition de service 26
2.12.1 Cycle de vie de la composition 26
2.13 Quelques travaux d'approche sur la découverte des services 27
2.13.1 Vue Architecturale 27
2.13.2 La vue matchmaking 28
2.13.3 Moteur de recherche pour la découverte des services cloud 28
2.13.4 La composition de service comme un système multi-agent 29
2.14 Conclusion 30

3 Agents, Système multi-agents et Agents Crawler 31
3.1 Introduction 31
3.2 Qu'est-ce qu'un agent ? 31
3.3 Les propriétés d'un agent 32
3.4 Types des agents 33
3.4.1 Agents cognitifs 33
3.4.2 Agents réactifs 33
3.4.3 Agents hybrides 33
3.5 Définition d'un système multi-agents 33
3.6 Les caractéristiques d'un système multi-agent 35
3.7 L'environnement dans un système multi-agents 35

3.8 Communication entre les agents 36
3.8.1 Communication par envoi des messages 37
3.8.2 Communication par partage d'information 37
3.8.3 Les langages de communication 38
3.9 Interaction entre les agents 39
3.10 Les Crawlers 40
3.10.1 Définition de Web Crawler 40
3.10.2 Fonctionnement d'un Web Crawler 40
3.10.3 Les différentes techniques de crawling 41
3.10.4 Cloud Crawler 42
3.11 Conclusion 42

4 Conception du système Crawler Agent 43
4.1 Introduction 43
4.2 Objectif visé par notre approche 43
4.3 Scénario du processus de déroulement 44
4.4 Architecture du Cloud crawler agent proposé 44
4.5 Diagramme de communication entre agents 46
4.6 Ontologie de communication entre agents 48
4.7 La structure de l'agent de stockage 49
4.8 Catégorisation des Services 50
4.9 La Base de Données de Service 52
4.10 Processus de mise à jour 54
4.11 Conclusion 54

5 Implémentation du Crawler Agent 55
5.1 Introduction 55
5.2 Outils de développement 55
5.2.1 Environnement Intellij , le langage Java et les bibliothèques tiers . . 55
5.2.2 La plateforme JADE 56
5.2.3 Git et Github 58
5.2.4 Système de gestion de base de données MySQL 59
5.3 Réalisation 59
5.3.1 Capture d'écran de l'interface graphique de JADE avec les agents en cours d'exécution 59
5.4 Quelques bout de code de notre implémentation 60
5.4.1 Capture d'écran des services crawlés 63
5.4.2 Quelques données de description de services crawlé 64
5.4.3 Vue de quelques échanges de messages entre agents 66
5.4.4 Schéma Relationnel de la base de donnée 67

5.5 Comparaison de notre approche avec d'autres approches 68
5.6 Conclusion 70

Conclusion **71**

Bibliographie **75**

Table des figures

1.1 Symbole représentatif sous forme de nuage du Cloud computing. 3
1.2 Architecture traditionnelle vs virtualisation, source [vmware.com] 5
1.3 Le cloud framework du NIST (source : NIST) 6
1.4 Modèle de service en synthèse - source : IONOS. [1] 9

2.1 Notions de base sur les services Web 14
2.2 Structure générale d'un document WSDL 15
2.3 Exemple de structure d'un document YAML OpenAPI 21
2.4 Les propriétés non-fonctionnelles de service 23
2.5 Services Cloud en WSDL/WADL vs Services Cloud non en WSDL/WADL 25
2.6 Cycle de vie de la composition de service 27
2.7 Architecture du système multi-agents 29

3.1 l'architecture interne d'un agent 32
3.2 Système multi-agents 34
3.3 Représentation classique d'un agent et de son environnement 36
3.4 Communication par partage d'informations 38
3.5 structure d'un message FIPA-ACL 39
3.6 exemple message FIFA-ACL 39
3.7 Processus de Crawling 41

4.1 Algorithme du scénario de crawling 44
4.2 Architecture - cloud crawler agent proposé 45
4.3 Diagramme de communication entre agents dans le cas de la découverte . . 47
4.4 Diagramme de communication entre agents dans le cas de la suppression . 47
4.5 Ontologie de communication entre agents 49
4.6 Schéma de la structure de l'agent de stockage 50
4.7 Modèle de notre approche sémantique pour la catégorisation des services . 51
4.8 Diagramme de classe de la base de services 54

5.1 GUI de JADE RMA (Remote Monitoring Agent) 57
5.2 Git et GitHub 58

5.3 Les différents agents du système en cours d'exécution 60
5.4 Liste des services Google crawlé 1 63
5.5 Liste des services Google crawlé 2 64
5.6 Description Service Books API 1 64
5.7 Description Service Books API 2 65
5.8 Description Service Books API 3 65
5.9 quelques messages échangés entre agents 67
5.10 Schéma relationnel de la base de donnée 68

Liste des abréviations

NIST The National Institute of Standards and Technology
PC Personal Computer
W3C Le World Wide Web Consortium
SOA Service Oriented Architecture (Architecture Orienté Service)
WSDL Web Service Definition Language
SOAP Simple OBject Acces Protocol
XML eXtensible Markup Language
UDDI Universal Description Discovery and Integration
API Application Programming Interface (interface de programmation d'application)
IaaS Infrastructure as a service
PaaS Platform as a service
SaaS Software as a service
CCIF Cloud Computing Interoperability Forum
AAA Authentification, Autorization, Accounting
SLA Service Level Agreement
REST Representational state transfer
HTTP Hyper Text Transfert Protocol
OWL-S Ontology Web Language for Service
QoS Quality of Service
USDL Unified Service Description Language
WADL Web Application Description Language
CCSC Cloud Computing Service Composition
SMA Système Multi-Agent
FIPA Foundation for Intelligent Physical Agents
ACL Agent Communication Language
URL Uniform Ressource Locator
JADE Java Agent DEvelopment
GNU GPL GNU General Public License

Introduction générale

Dans un contexte Covid-19, d'après *Canalys*[1], un cabinet d'étude américain, le secteur du Cloud computing qui est en quelques sorte cette forme de fournir des services informatiques via Internet a enregistré un chiffre d'affaires global de 31 milliards de dollars durant le premier trimestre de l'année 2020, une hausse de 34% en un an due à une forte demande en services. Le terme cloud est devenu une métaphore de l'informatique moderne où tout est un service, dans l'objectif d'offrir une innovation plus rapide, des ressources flexibles et des économies d'échelle, le cloud est devenus presque incontournable de nos jours.

Le paradigme du cloud computing repose sur plusieurs technologies comme les services web, etant un modèle de livraison de services se basant sur des infrastructures virtuelles mises à disposition par des fournisseurs de service auprès desquels les clients font leur demandes. Dans la plupart des cas, un seul service ne peut pas répondre à une requête du client, les services doivent être composés en un service composé. Il est évident que pour fournir ce dit service composé au milieu d'une panoplie de service disponible sur Internet, disposé d'une base de données ou aller piocher ces services serait un avantage.

L'objectif de notre travail est de réaliser un crawler-agent, avec le crawl qui est une technique utilisé par les moteurs de recherche pour découvrir de nouvelles pages et les indexer. Les crawlers, qui sont des agents logiciel sont également utilisés dans d'autres cas de figure notamment dans la sécurité pour tester si un site est vulnérable ou non. Quant à notre crawler, il fait partie d'un système multi-agents dédié à la composition de services-cloud atomique. Le but de cet agent est de découvrir les services cloud disponibles pour les ajouter à sa base de données pour qu'elles soient utilisés directement et facilement par les autres agents de ce système.

Notre mémoire est organisé en cinq chapitres :

— **Le 1er chapitre : « Le cloud computing »**. Il est ici de présenter le paradigme du cloud computing, ses modèles, ses avantages et limites.

— **Le 2ème chapitre : « Service web, Service Cloud »**. Ce chapitre introduit les principes de la technologie des services web et des services cloud, avec leurs environnements (définitions, concepts, langages, découverte, etc.), ainsi qu'une étude

1. https ://www.canalys.com/

comparative.

— **Le 3ème chapitre : « Agents, Système multi-agents et Agents Crawler »**. Ici nous avons défini les Agents et les Systèmes multi-agents. Nous avons aussi discuté sur les crawlers (web crawler et cloud crawler).

— **Le 4ème chapitre : « Conception du système Crawler Agent »**. Ce chapitre est consacré à la conception détaillée de notre système, en présentant notre approche ainsi qu'une architecture du système.

— **Le 5ème chapitre : « Implémentation du Crawler Agent »**. Ce chapitre est consacré à l'expérimentation et à l'évaluation de notre système. Nous y présentons notre réalisation ainsi que les outils de développement utilisés pour la réalisation du Cloud Crawler-Agent

Chapitre 1

Cloud computing

1.1 Introduction

Ayant émergé comme un nouveau paradigme pour l'accès à la demande à une immense variété de ressources informatiques, le Cloud computing abrégé en Cloud (« le Nuage »), ou l'informatique en nuage ou encore l'infonuagique (au Québec) offre une alternative à l'utilisation des ressources locales. Il est considéré comme étant la prochaine étape dans l'évolution de l'internet.

On peut être amené à se demander : Qu'est-ce que le Cloud computing ? Comment est-il apparu ? Quels sont ses objectifs ? En quoi consiste-t-il ?

Cette partie essayera de formuler une définition au Cloud computing et donner une vue globale de ses aspects, ses objectifs ainsi que tout ce qui est en relation avec ce paradigme.

1.2 Qu'est-ce que le Cloud computing ?

Depuis son apparition jusqu'à nos jours, plusieurs propositions de définitions ont été faites. Mais le Cloud Computing étant un paradigme en évolution, toutes ces propositions sont sujets à des redéfinitions à la suite de travaux de recherches et de discussions. Ainsi arriver à trouver une définition standard acceptée par tous est chose difficile, car plusieurs de ces définitions existantes ne se focalisent que sur seulement un ou quelques aspects du Cloud computing.

D'après le chercheur en cyber sécurité Vic (J.R.) Winkler[2], associé principal chez Booz Allen Hamilton, fournissant des consultation technique auprès des clients du gouvernement américain, dans son célèbre livre *Securing the cloud*, il définit le cloud computing comme étant un ensemble de technologies de l'information et composants informatiques (matériels, logiciels, réseaux et services) nécessaires pour permettre le développement et la livraison de services informatiques via Internet ou un réseau privé c'est à dire particulier[2].

La définition suivante, donnée par le National Institute of Standards and Technology

(NIST), l'Institut National des normes et de la technologie des États-Unis, est le plus largement accepté par la communauté, car couvrant un large aspect du Cloud Computing[3] :

Le Cloud computing est un modèle Informatique qui permet un accès facile et à la demande au travers du réseau à un ensemble partagé de ressources informatiques configurables (serveurs, stockage, applications et services) qui peuvent être rapidement provisionnées et libérées par un minimum d'efforts de gestion ou d'interaction avec le fournisseur du service.

Le Cloud est donc un modèle où on va *consommer* des ressources informatiques comme n'importe qu'elle utilité. Dans notre quotidien nous consommons de l'eau, de l'électricité, du gaz, avec le Cloud computing nous pouvons consommer : de la puissance de calcul, des serveurs, de l'espace de stockage ou des applications. Consommer tout type de ressource informatique sans en posséder l'infrastructure technique qui est chez les fournisseurs Cloud, et ces ressources informatique vous pouvez y accéder grâce à un réseau étendu voir internet.

Le terme "Cloud", ainsi que le symbole en forme de nuage, sont utilisés de façon métaphorique pour représenter en fait une abstraction cachant la complexité des infrastructures d'internet, de télécommunication et des processus à l'intérieur de ces infrastructures.[4]

FIGURE 1.1 – Symbole représentatif sous forme de nuage du Cloud computing.

1.3 Émergence du Cloud computing

Le Cloud computing en soi est plus une évolution d'usage qu'une révolution technologique. Car son existence s'est fait après la réalisation de certains progrès techniques sur lesquels il se base ainsi que sur certains paradigmes. Dans cette partie on fait un saut dans l'histoire de l'informatique, pour essayer de montrer les technologies et paradigmes qui ont permis l'émergence du Cloud.

1.3.1 Les modèles de calcul

L'histoire de l'informatique remonte à l'invention de la première machine à calculer. Et depuis après on a assisté à un bouleversement avec l'apparition de plusieurs technologies de modèle de calcul. Le Cloud computing est considéré comme la 6ème phase de cette évolution des paradigmes de calcul (computing paradigms).[5]

1. **Le Mainframe Computing (l'informatique du mainframe) :** Le mainframe (Ordinateur Central) est une grande machine sur lequel plusieurs utilisateurs se connectent via un terminal.
2. **Le PC Computing (L'informatique des PCs) :** Après le mainframe, ce petit ordinateur mais assez puissant pour satisfaire les besoins des utilisateurs fait son apparition. Il s'agit du PC [1].
3. **Le Network Computing (L'informatique des Réseaux) :** Ces PCs sont connectés entre eux et aussi avec un autre type spécialisé d'ordinateurs appelés serveurs pour créer un réseau.
4. **Internet Computing (L'informatique d'Internet) :** Consécutivement, les réseaux sont liés entre eux pour construire un réseau mondiale appelée l'Internet. Cependant, Internet a réalisé sa véritable percée avec l'invention du W3C par Tim Berners-Lee en 1989.
5. **Le Grid Computing :** Le Grid Computing permet l'agrégation des ressources distribuées pour y accéder d'une manière transparente. Ils cherchent dans la majeure partie des cas à partager des ressources de calcul et de stockage réparties entre différents domaines administratifs. Leur principal objectif est d'accélérer une large gamme d'applications scientifiques telles que la modélisation climatique, la conception de médicaments et l'analyse des protéines [6].
6. **Cloud Computing :** Le Cloud computing quand à lui, fournit des ressources partagées sur Internet ou un réseau de manière simple et évolutive.

1.3.2 Les Bases technologiques

La virtualisation, les services Web et l'Architecture orientée services qui ne sont pas de nouveaux concepts ; mais cependant, ils représentent les technologies de base pour le Cloud computing.

1. **La virtualisation :** La virtualisation est une technologie qui permet à de multiples Machines virtuelles (machines invitées) de s'exécuter sur une seule machine physique (machine hôte) et de partager les ressources de cette dernière machine physique. Avec la figure 1.2 nous pouvons voir l'opposition entre l'ancienne architecture avec

1. PC : Sigle signifiant « Personal Computer » ou « ordinateur personnel » en français.

une relation « un serveur physique, un système d'exploitation » et l'architecture virtualisé avec plusieurs systèmes d'exploitation fonctionnant sur une seule machine physique. L'avantage du Cloud computing est la possibilité de virtualiser et de partager des ressources entre différentes applications avec l'objectif d'une meilleure utilisation du serveur.[5]

2. **Services Web et Architecture orientée services (SOA) :** Les Services Cloud sont généralement conçus comme des services Web, qui suivent les normes, les standards de l'industrie, y compris WSDL (Web Services Description, qui est entre autre une grammaire XML permettant de décrire un service web), SOAP (Simple Object Access Protocol, qui est un protocole d'échange d'information structurée dans l'implémentation de services web bâti sur XML), et UDDI (Universal Description Discovery and Integration, qui est un annuaire de services fondé sur XML et plus particulièrement destiné aux services Web). SOA est un modèle de conception d'architecture logicielle largement adopté, basé sur la philosophie de conception de nouveaux services, composés de services existants réutilisés, pour être utilisé dans le cadre de processus métier afin d'exécuter des activités commerciales [7]. Dans une architecture SOA, un service est une fonction qui est bien défini, autonome et ne dépend pas du contexte ou état des autres services.

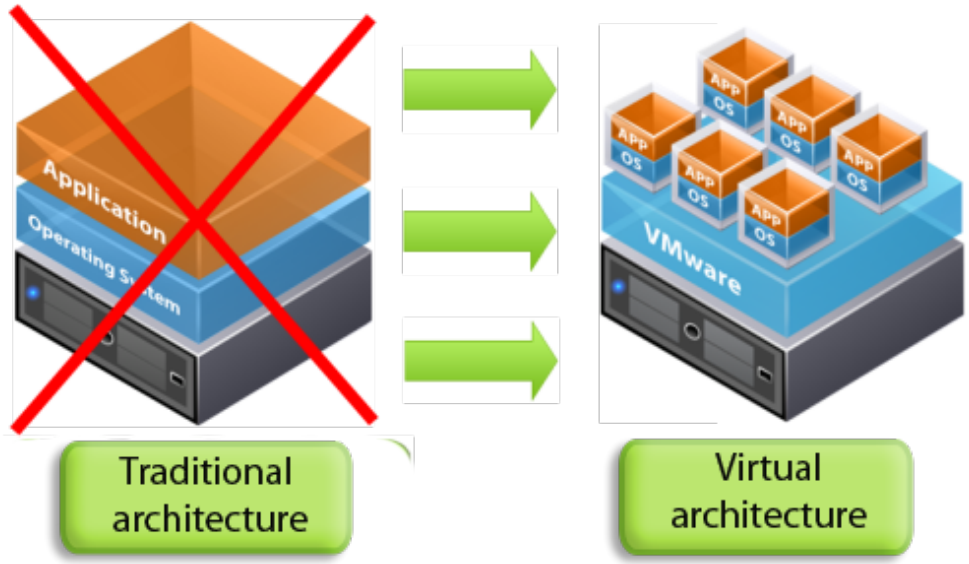

FIGURE 1.2 – Architecture traditionnelle vs virtualisation, source [vmware.com]

La virtualisation a été parmi les premières pierres vers le Cloud Computing ; elle ainsi que l'évolution constante des débits internet depuis son apparition, la prolifération des appareils mobiles intelligents, sans oublier les services web que nous allons voir un peu en profondeur par la suite, ont rendu l'existence du Cloud possible.

1.4 Le modèle de cloud computing

Le National Institute of Standards and Technology propose un Framework simple de définition des entités fonctionnelles du Cloud, représenté dans la figure 1.3. Dans ce modèle de Cloud on retrouve : [3][8]

— Quatre (4) Caractéristiques essentielles (Essential characteristics)

— Trois (3) modèles de Service (Service Models)

— Quatre (4) modèles de déploiement (Deployment Models).

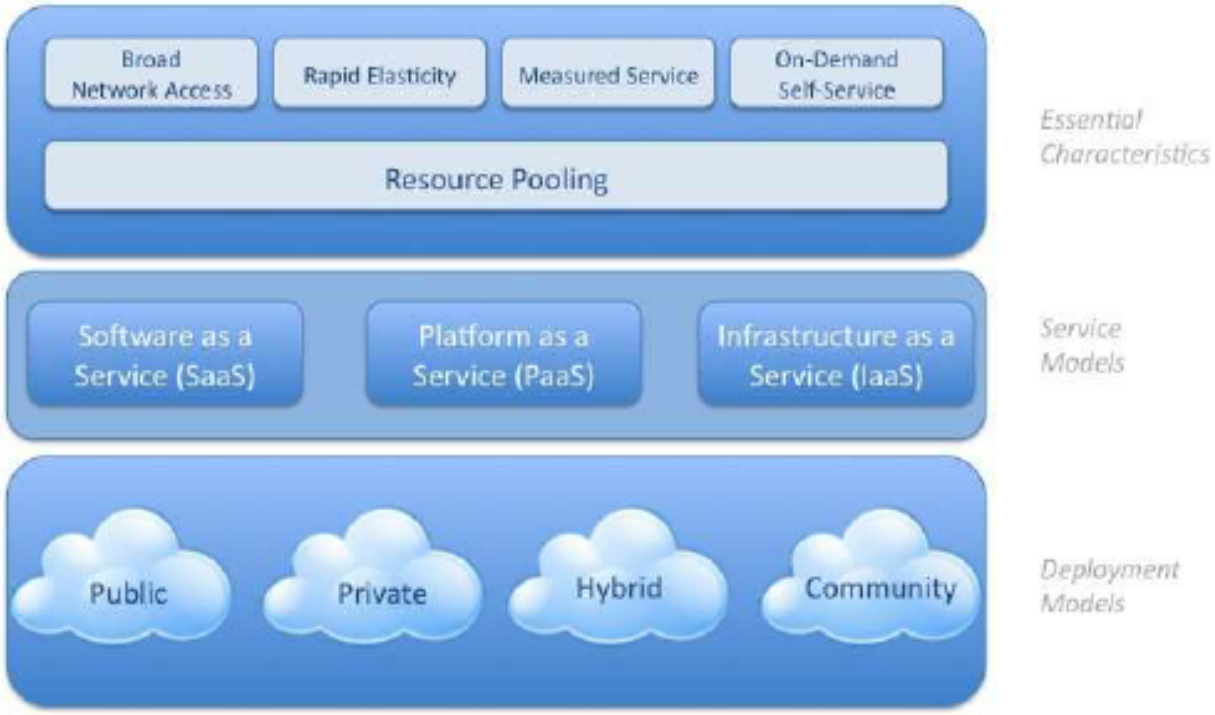

FIGURE 1.3 – Le cloud framework du NIST (source : NIST)

1.4.1 Les caractéristiques du cloud

De manière globale le cloud computing tel que nous le connaissons aujourd'hui contient cinq (5) caractéristiques essentielles.

1. **Service à la demande (On-demand self-service)**

   La première caractéristique est qu'avant tout un service Cloud est un service à la demande. Cela veut dire qu'un consommateur peut demander et recevoir l'accès à une offre de service, sans qu'un administrateur ou une sorte de personnel de support ne doive répondre manuellement à la demande. Les processus de demande et les processus d'exécution sont tous automatisés. Le service est fourni au client au moment de son besoin et de sa demande.

2. **Accès à travers le réseau (Broad network access)**

   Cette deuxième caractéristique consiste au fait que les ressources du Cloud Computing sont accessibles via le réseau (Internet par exemple) en utilisant des techniques standardisées. Ce qui donne la possibilité de s'en servir en utilisant un téléphone portable, une tablette ou bien un PC. Un cadre basé en Algérie par exemple peut remplir ses fonctions pendant les voyages d'affaires, en accédant aux ressources en ligne de son entreprise hébergées en France via la connexion Internet au Burkina-Faso.

3. **Ressource Pooling (Ressources mutualisées)**

   Les ressources informatiques du fournisseur Cloud sont regroupées pour servir plusieurs consommateurs. Ces différentes ressources physiques et virtuelles sont dynamiquement attribuées et réaffectés en fonction de la demande des consommateurs. Le même service et les mêmes ressources sont consommés par plusieurs clients.

4. **Élasticité rapide (Rapid elasticity)**

   Les capacités peuvent être provisionnées et libérées de manière élastique, dans certains cas automatiquement, pour évoluer rapidement vers l'extérieur et vers l'intérieur en fonction de la demande. Un service évolutif avec une certaine souplesse dans sa modification et son déploiement.

5. **Service mesuré et facturation à l'usage (Pay as you go)**

   Le client va payer ce qu'il a consommé selon le type et le temps d'utilisation du service.

1.4.2 Les modèles de Service

Les services du Cloud sont catégorisés en fonction de la couche technique fournie. Il y a 3 modèles d'utilisation, ou modèles de livraison.

1. **IaaS (Infrastructure as a Service)**

   C'est une forme de Cloud Computing offrant des ressources informatiques au sein d'un environnement virtualisé où le fournisseur peut contrôler l'augmentation, la réduction et l'accès aux ressources de l'infrastructure. Les IaaS peuvent être des serveurs, des réseaux, de l'espace de stockage ou des espaces au sein de Data Center. Un fournisseur IaaS sur le Cloud met en œuvre un système de payement basé sur le taux de consommation des ressources physiques et virtuelles, qui peuvent évoluer selon la demande du consommateur. C'est-à-dire que le client achète quand il veut, la quantité de ressources dont il a besoin, et il ne paye que ce qu'il utilise [9]. Dans le IaaS les utlisateurs du Cloud accède à l'infrasctructure via des API. IaaS représente le niveau le plus bas de modèle de service du Cloud Computing.

   — Amazon offre EC2 et AWS ;

— Microsoft offre Azure.

Vue les cas d'usage, on voit que la population cible du IaaS sont les exploitants informatiques.

2. **PaaS (Platform as a Service)**

   Le PaaS qui est le niveau intermédiaire du modèle de service offre un environnement de développement d'applications comme service à ses utilisateurs afin qu'ils puissent développer et maintenir leurs applications et leurs utilitaires spécifiques au Cloud. Le PaaS est déférent du SaaS car le SaaS est une application développée et déployée et le PaaS fourni une plateforme ou un terrain de développement, et aussi les outils et infrastructure de développement qui doivent être fourni par le fournisseur [10].

   — Microsoft offres Microsoft Azure Platform

   — Google offre Google App Engine

   Les développeurs sont la population cible.

3. **SaaS (Software as a Service)**

   Avec le SaaS, les fournisseurs offrent un abonnement annuel ou mensuel à un logiciel installé, stocké et exécuté sur le Cloud et gérés par ces fournisseurs qui s'occupent de faire la mise à jour de ces logiciels. Les utilisateurs peuvent accéder à des applications et des services SaaS depuis n'importe quel endroit en utilisant n'importe quel ordinateur ou appareil mobile connecté à internet car les données et les paramètres sont stockées sur le Cloud [9]. Le SaaS est le niveau supérieur du modèle de service. On y accède généralement par un agent logiciel ou le navigateur Web.

   — Google Apps (messagerie et bureautique)

   — Sales Force CRM (Customer Relationship Management)

   — Microsoft offre Office 365 (outils collaboratifs)

   Les utilisateurs finaux sont la population cible du SaaS.

4. **Synthèse :** Ci-dessous on peut voir de manière schématique ces différents services avec le niveau d'abstraction, montrant les couches que chaque acteur : utilisateur et fournisseur doivent gérer.

   Ainsi on peut voir que par exemple en situation interne (On-Premises) qui est le mode standard tout est hébergé et géré en interne. Plus nous nous éloignons de ce mode moins on a des choses à gérer , on le remarque avec le SaaS où tout est géré par le fournisseur et en tant qu'utilisateur on se contente d'utiliser le logiciel ou l'application fournie.

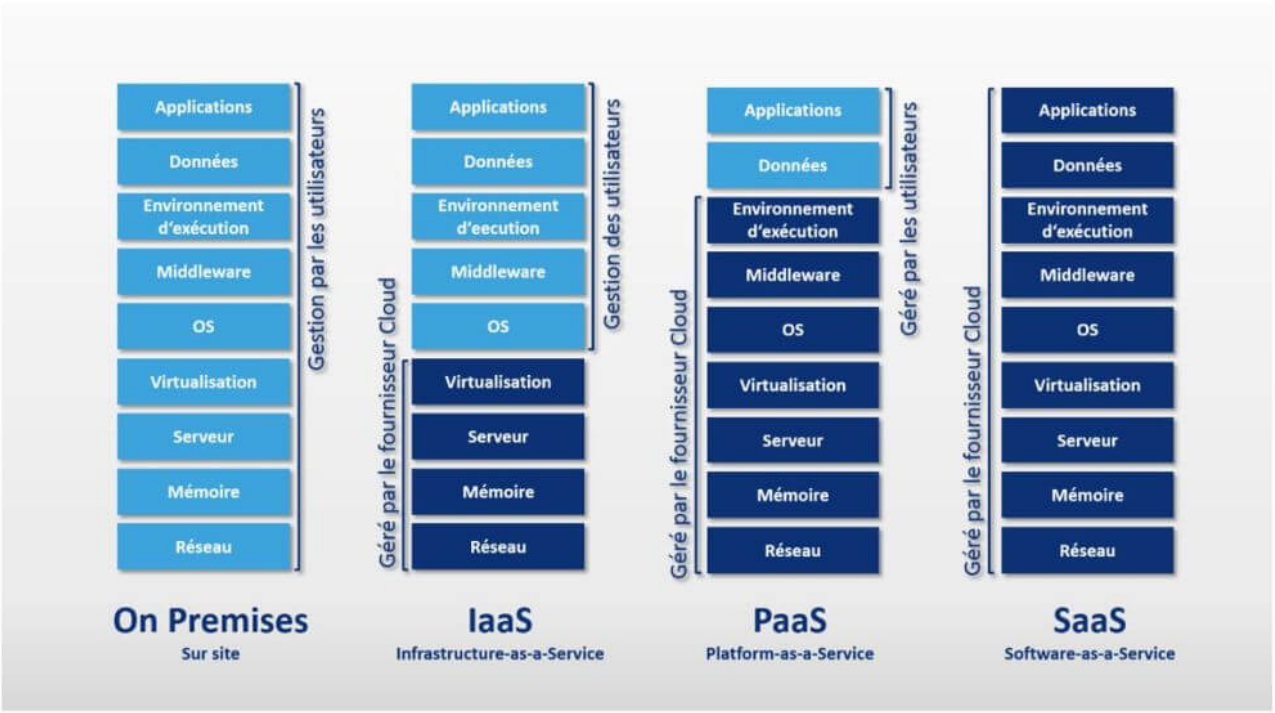

FIGURE 1.4 – Modèle de service en synthèse - source : IONOS. [1]

1.4.3 Les modèles de déploiement

Selon le nombre de clients à servir et le modèle commercial on peut énumérer 4 types de modèles de déploiement de manière générale :

1. **Cloud public :** L'infrastructure Cloud [2] est mise à disposition pour une utilisation ouverte par le grand public ou à de grands groupes industriels. Il peut appartenir, être géré et être exploité par une entreprise de commerce, par une organisation académique, ou par une organisation gouvernementale. Il est détenu par le fournisseur Cloud.
2. **Cloud privé :** Dans le Cloud privé contrairement au précèdent l'infrastructure est mise à disposition pour une utilisation exclusive par une seule organisation ou entreprise et avec plusieurs consommateurs. Le Cloud privé peut s'agir d'un Cloud interne ou entièrement dédié, accessible, hébergé chez un tiers, et mutualisé entre les différentes entités d'une seule et même entreprise.
3. **Cloud communautaire :** L'infrastructure du Cloud communautaire est mise à disposition pour une utilisation exclusive par une communauté spécifique de consommateurs qui partagent les mêmes domaines d'intérêts. Il peut appartenir, être géré et être exploité par une organisation ou plus dans la communauté, par un tiers ou

2. Une infrastructure Cloud est la collection de matériels et de logiciels permettant le Cloud computing. Il peut être considéré comme contenant à la fois une couche physique comprenant les ressources matérielles (serveur, stockage, réseau etc.) et une couche d'abstraction constituée du logiciel déployé, qui manifeste les caractéristiques essentielles du cloud.

par une combinaison d'eux. Avec ce modèle, on peut réaliser à la fois des économies et une réduction de la quantité de trafic vu que les sociétés utilisant ce modèle ont des besoins communs [8].

4. **Cloud hybride :** Dans ce modèle il s'agit d'une combinaison de deux ou plusieurs Cloud public, privé et communautaire amenés à coopérer et à partager entre eux des applications et données [8].

1.5 Quelques acteurs du cloud computing

1.5.1 Cloud customer

Le Cloud customer (client Cloud) est celui qui demande le déploiement d'un service auprès d'un prestataire de services : le service provider, pour satisfaire ses clients les utilisateurs finaux (End-users). Il est le propriétaire du service déployé et Il facture à ses clients l'utilisation de ses services et paie le prestataire pour la construction, le déploiement et la maintenance des services.[4].

1.5.2 Service provider

Un fournisseur de service ou service provider peut être une personne, une organisation ou une entité chargée de fournir un service disponible pour les consommateurs du Cloud. Un fournisseur de Cloud crée le logiciel, la plate-forme, gère l'infrastructure technique requise pour fournir les services, et assure la sécurité et la confidentialité des services [3]. Le fournisseur de services prend le contrat du Cloud customer pour construire un service répondant aux besoins de ce dernier.

1.5.3 End-user (utilisateur final)

L'utilisateur final est un utilisateur d'un service (complexe), qui interagit réellement avec le service et utilise ses fonctionnalités. L'utilisateur final paie le cloud consumer qui possède le service [4].

1.5.4 Infrastructure provider (Fournisseur de service)

Le fournisseur de service est le propriétaire de l'infrastructure physique du cloud. Ce fournisseur peut également faire des contrat avec d'autres fournisseurs tel que les fournisseurs de réseau dans le but d'utiliser leurs services [4].

1.6 Cloud computing : Avantages, limites et Challenges

Comme Chaque technologies le Cloud computing possède à lui aussi des avantages ainsi que des limites. Ces bénéfices sont multiples et différents, et cela selon la population concernée [11][12].

1.6.1 Avantages

1. Le Cloud permet aux entreprises de faire des économies, car pas d'investissement préalable dans du matériel, ainsi que la baisse des coûts de maintenance et d'exploitation.
2. Pour les petites entreprises, le Cloud permet de lancer un service sans aucun investissement capitalistique en hardware et peu en software.
3. Possibilité d'avoir accès à des services parfois coûteux à moindre prix et de manière évolutive.
4. Maîtrise des coûts : payant qu'à l'usage, le coût est proportionnel et prévisible dans le temps.
5. Plus grande flexibilité en accédant aux services de n'importe où.
6. Le produit peut s'adapter à tout moment aux besoins de l'entreprise.
7. La mutualisation des ressources permet de disposer de capacités illimitées en matière de stockage et de bande passante.

1.6.2 Limites et Challenges

Défis de Sécurité

Dans le Cloud computing la question de la sécurité est l'une des plus gros défis et la plus grande préoccupation : [8]

— Au niveau de l'Infrastructure Cloud : les préoccupations sont liées à la virtualisation, le stockage et les vulnérabilités du réseau. On peut également y inclure les aspects physiques de sécurité du centre de données.

— Au niveau des Données : les préoccupations autour de l'intégrité, conservation, disponibilité, confidentialité des données et des utilisateurs.

— Au niveau de l'Accès : la préoccupation autour de l'accès au Cloud (Contrôle d'authentification, d'autorisation et d'accès ou AAA [3]), chiffrement de la communication, et la gestion de l'identité de l'utilisateur.

3. AAA (Authentication, Authorization, Accounting) : Les protocoles AAA sont des protocoles de sécurité qui permettent de contrôler l'accès au réseau et/ou application. l'Accounting pour la traçabilité - source SUPINFO International University

Verrouillage des données et standardisation

L'une des préoccupations majeure des utilisateurs du Cloud computing est que leurs données sont en quelques sortes verrouillées par certains fournisseurs. En effet, dans leurs formes présentes les infrastructures et les plateformes du Cloud computing n'emploient pas de méthodes standards de stockage des données des utilisateurs ainsi que de leurs applications, ce qui rend difficile et souvent impossible le déplacement des données et applications d'un utilisateur de chez un fournisseur qui ne répond pas à ses exigences vers un autre. Par conséquent, il n y a pas d'interopérabilité [4] et les données utilisateur ne sont pas portables. Actuellement, chaque offre de Cloud a sa propre façon d'interagir avec les clients / applications / utilisateurs du Cloud. Dans ce sens, il y a des efforts pour créer des normes ouvertes pour le Cloud computing par des initiatives comme le Cloud Computing Interoperability Forum (CCIF) qui est un forum formé de plusieurs organisations telle que Intel, Sun, Cisco dans cette lancée. Mais il reste toujours des travaux à faire dans ce sens[6].

En plus de ces Challenges on peut citer comme limites :

1. Un problème de dépendance : L'entreprise dépend de son prestataire, tant au niveau technique que de la qualité de service. Certains départements informatiques sont inquiets car les fournisseurs de Cloud computing ont un contrôle total des plateformes.
2. Un problème d'accès : Nécessité d'une connexion efficace et rapide. En cas d'infrastructure Cloud en panne, le client est en incapacité d'avoir accès à ses données.
3. Un problème d'opacité et de contrôle de donnée : on ne peut pas savoir exactement où nos données se trouve, géographiquement.

1.7 Conclusion

Le Cloud computing est un nouveau paradigme informatique qui offre une énorme quantité de calcul, de stockage et ressources en masses. Les Individus (Ex. Scientifiques) et les entreprises (par exemple, les jeunes entreprises) peuvent avoir accès à ces ressources en payant juste pour ce qui est vraiment nécessaire. Il est avant tout un modèle économique qui utilise les technologies existantes, car côté technique il n y a vraiment rien de nouveau. Ce qui est nouveau ce sont les concepts sur lesquels s'appuie ce modèle du Cloud à savoir ses différentes caractéristiques, comme défini par le NIST, ses modèles de déploiement ainsi que ses modèles de services. Tout dans le Cloud est fourni sous forme de service. Ce point sur les services (Web et Cloud) sera l'objet de notre prochain chapitre.

4. Interopérabilité : possibilité de communication entre deux ou plusieurs systèmes, appareils ou éléments informatiques

Chapitre 2

Services Web, Services Cloud

2.1 Introduction

Les services d'une manière générale sont des éléments de calcul auto-descriptifs et indépendants de la plate-forme qui prennent en charge une composition rapide et peu coûteuse des applications distribuées. Ils exécutent des fonctions qui peuvent aller de simples demandes à des processus métier complexes. D'après le livre *Web Services, SOA and Cloud computing* de Dauglas K .Barry et David Dick.

Un service est un logiciel qui remplit une fonction informatique et possède un type de système informatique sous-jacent, bien que non requis, le Cloud computing peut fournir ce système informatique sous-jacent. [13]

De nos jours, accéder à des services informatiques, plus précisément des services de qualité [1] de n'importe où et à tout moment, est un besoin croissant. Ces services hébergés sont de plus en plus riches et complexes.

Dans cette partie nous avons présenté les services Web et les services Cloud avec leurs environnements (définitions, concepts, langages, découverte, etc.), ainsi qu'une étude comparative.

2.2 Services Web

Un service Web est un composant d'unité logicielle modulaire, auto-descriptif, autonome et accessible sur le Web pouvant être publié par un fournisseur de services et invoqué par les demandeurs de services à travers Internet [14]. Dans la définition de service donnée par [13] mentionnée plus haut, les services Web sont définis comme un moyen de connecter des services entre eux. Les services web sont décrits comme une technologie de connexions. On peut les voir comme un mécanisme de communication entre les applications distantes

1. ISO/IEC 9126 propose 6 caractéristiques de qualité du produit logiciel : Capacité fonctionnelle, Fiabilité, Facilité d'utilisation, Rendement, Maintenabilité, Portabilité.

à travers les réseaux internet indépendant de tout langage de programmation et de toute plate-forme d'exécution [15].

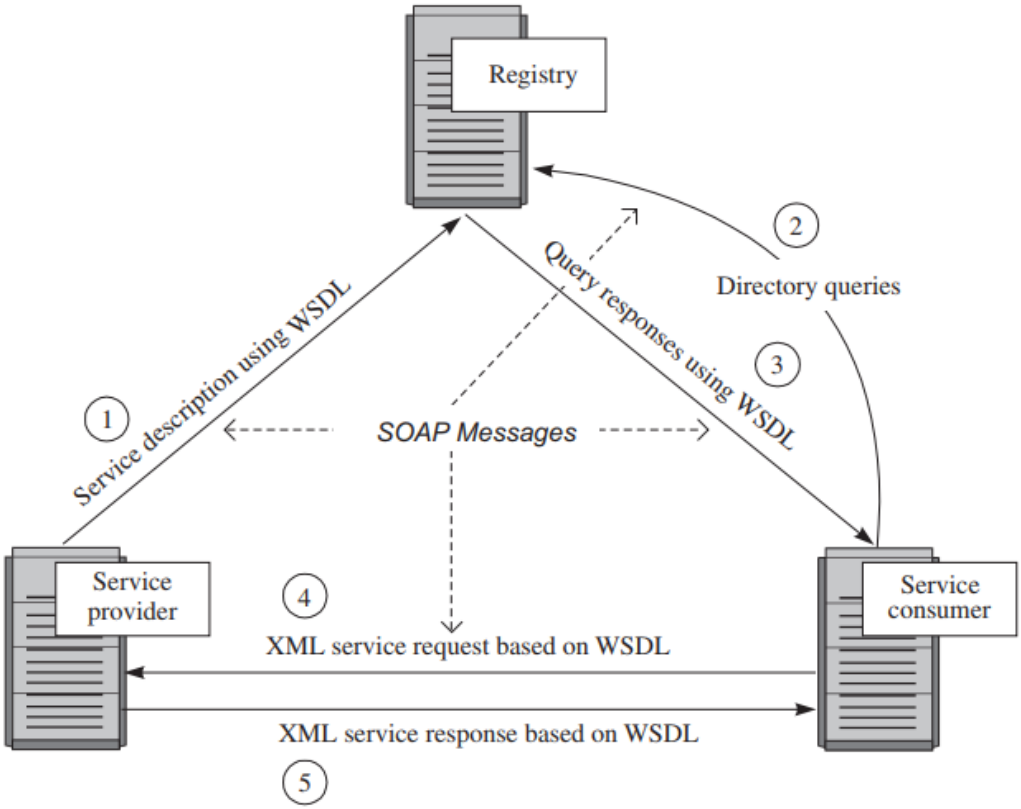

FIGURE 2.1 – Notions de base sur les services Web [13]

1) Un fournisseur de service décris son service utilisant un langage de description de service, comme WSDL. Cette définition est publiée dans un registre de service, tel que l'UDDI (Universal Description Discovery and Integration) ;
2) Un consommateur de service envoie une ou plusieurs requêtes au registre pour localiser un service et déterminer comment communiquer avec ce service ;
3) Une partie du fichier de description fourni par le fournisseur de services est transmise au consommateur de service. Cela indique au consommateur de service quelle sont les demandes et les réponses pour le fournisseur de services ;
4) Le consommateur de service (service consumer) utilise le fichier de description de service pour envoyer une requête au fournisseur de service ;
5) Le fournisseur de services fournit la réponse attendu au consommateur de services.

2.3 Architecture des services web

Les services Web communiquent via un ensemble de technologies fondamentales qui partagent une architecture commune, comme pour le World Wide Web. Les technologies

de bases utilisées par les services Web sont WSDL, SOAP et UDDI, HTTP, XML et bien d'autres.

2.3.1 Langage de desciption WSDL

WSDL (Web Services Description Language) est un langage formel de description standard des services web basé sur le XML [2] pour que différents langages d'implémentation puissent le comprendre et aussi être compatible avec les différentes plateformes utilisées. Un fichier WSDL permet de décrire les détails sur la fonctionnalité (Méthodes, Paramètres) et la localisation d'un service Web (URI, Port utilisés, Protocole d'invocation) [16]. Autrement dit il indique comment utiliser le service Web et comment interagir avec lui, il est en quelques sortes l'interface présentée aux utilisateurs.

Le WSDL offre une description sur deux niveaux. Le niveau abstrait qui est utilisé principalement lors du processus de sélection et le niveau concret qui est utilisé lors de l'invocation des opérations du service web [17].

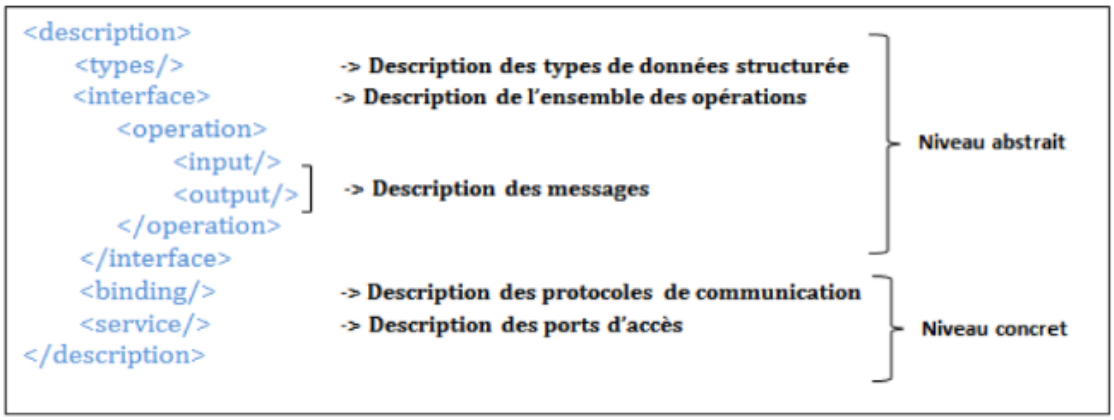

FIGURE 2.2 – Structure générale d'un document WSDL
[17]

— La partie abstraite : Décrit les messages et les opérations disponibles au travers de balises : type pour les types de données qui donnent des indications sur les structures de données des paramètres utilisés par le service, message qui encapsule les données véhiculées pour l'opération invoquée, et la balise operation.

— La partie concrête : Décrit le protocole et le type d'encodage à utiliser pour les messages nécessaires pour invoquer un service Web particulier.

2. XML (eXtensible Markup Language) est un modèle universel de représentation et d'échange de données. Il s'agit d'un texte au format simple, flexible et indépendant de tout fabricant. De plus, il donne une structure aux documents et aux données

2.3.2 UDDI

Un client ne peut pas utiliser un service dont il ignore l'existence. Ainsi le registre UDDI (Universal Description, Discovery and Integration) est une base de données virtuelle des services Web existants qui permet la publication et la localisation de services web. D'une part, il permet aux fournisseurs de services d'enregistrer des services Web sous un format normalisé et, d'autre part, il se concentre sur le processus de découverte de services Web répondant aux besoins des services en SOA [16].Il est destiné à servir de moyen de «découverte» des services Web décrits à l'aide de WSDL [13]. Le registre se compose de trois parties [17] :

1. Une partie comprenant des descriptions générales sur les fournisseurs de services (nom, coordonnées, la description etc.) appelée pages blanche.
2. Une deuxième partie (appelée pages jaunes) comportant des descriptions détaillées sur les fournisseurs de services catalogués dans les pages blanches de façon à classer les entreprises et les services par secteurs d'activité.
3. La troisième partie appelée pages vertes qui fournissent des informations techniques précises sur les services fournis (adresses web des services et les moyens d'y accéder). Ces informations incluent la description du service, du processus de son utilisation et des protocoles utilisées pour son invocation.

2.3.3 SOAP

SOAP (Simple object Access Protocol) est un protocole standard de communication qui définit un ensemble de règles pour structurer des messages, principalement pour exécuter des dialogues requête-réponse. Il est basé sur XML et sur le protocole standard HTTP et assure l'interopérabilité entre les composants indépendamment des mécanismes de transport, des systèmes d'exploitation et des langages de programmation.

2.4 Service web sémantique

Le concept des services Web sémantiques, est le fruit de la convergence du domaine des services web avec le Web sémantique. L'objectif premier du Web sémantique est de définir et lier les ressources du Web afin de simplifier leur utilisation, leur découverte, leur intégration et leur réutilisation dans le plus grand nombre d'applications [Berners-Lee et al. 2001]. Le Web sémantique doit fournir l'accès à ces ressources par l'intermédiaire de descriptions sémantiques exploitables et compréhensibles par des machines. Cette description repose sur des ontologies. La sémantique ainsi exprimée permettra l'automatisation des fonctionnalités suivantes qui sont nécessaires pour une collaboration interentreprises efficace [Patrick Kellert et Farouk Toumani Les web services sémantiques Juillet 2003] :

1. Processus de description et de publication des services ;
2. Découverte des services ;
3. Sélection des services ;
4. Composition des services ;
5. Fourniture et administration des services ;
6. Négociation des contrats.

2.5 Services Cloud

L'appellation Services Cloud fait référence aux matériels et aux logiciels accessibles via Internet et hébergés par des centres de données (data centers). Ces services Cloud sont utilisables par les utilisateurs sur une base de paiement à l'utilisation [14] qui est défini dans Le service-level agreement (SLA) ou « entente de niveau de service » qui est un accord négocié entre deux parties, dont l'une est le client et l'autre, le prestataire de service. C'est un document qui définit la qualité de service.

2.6 Relation entre Services web et Services Cloud

Les similitudes entre les services Web et les services Cloud sont une préoccupation dans le monde de la recherche. Certains sont catégorique sur le fait qu'ils ne sont pas les mêmes, d'autres visent plus le fait qu'on peut les confondre. Certains autres évoquent le fait qu'ils dépendent de plus en plus les uns des autres si les utilisateurs souhaitent bénéficier de la pleine vitesse et de la facilité d'utilisation qu'offrent ces services. Mais ces préoccupations ne nuisent en rien à leur utilisation.

Les services Web et les services Cloud ne sont pas identiques. Les entreprises peuvent disposer de services Web sans infrastructures Cloud. Les services Web font référence aux applications Web qui permettent aux clients d'interagir avec les logiciels via Internet de manière standardisée. Cependant, les services Cloud offrent un environnement pour les données, les logiciels, les plates-formes et les éléments d'infrastructure. [16].

Contrairement aux services Web qui utilisent des langages standard tels que le WSDL et le UDDI pour publier leurs services dans les registres des services pour la découverte , la majorité des services Cloud accessibles au public ne sont pas basés sur des normes de description, ce qui rend la découverte des services Cloud difficile [18]. Un service web déployé dans le Cloud est généralement appelé service Cloud, bien que les services web s'appuient sur un ensemble de standard permettant l'interopérabilité, tant dis qu'il existe des services dans le Cloud ne permettant pas une interopérabilité, car ne respectant pas un standard bien défini, et sont appelé services Cloud.

Cependant, certaines entreprises optent pour faire évoluer les services Web en services Cloud. C'est à dire que de nos jours plusieurs entreprises déplacent leurs applications ou services Web vers une infrastructure basée sur le Cloud pour moderniser leurs entreprises. Pour cela le service web doit présenter certains critères tels que [16] :

— Prise en charge de la technologie de virtualisation ;

— Prise en charge de l'hébergement multi client (exigences / besoins différents des clients) : cela signifie une seule instance de logiciel pour servir plusieurs entreprises en adaptant leurs exigences via la configuration en même temps ;

— Une infrastructure riche.

Aussi il nous faut noter que la souplesse des services web a poussé les entreprises à adopter les services web comme Interface aux services Cloud. La combinaison entre services web et Cloud computing permet alors d'appliquer les techniques de classifications, découverte et composition des services web sur les services Cloud, mais en prenant en considération les caractéristiques qui différencient une application Cloud d'un service web. [9]

Dans la suite de notre étude cette différence qui existe entre ces deux concepts n'a pas été négligée. Là où les besoins de distinctions se présentent entre Service Web et Cloud, une précision a été faite.

2.7 Les méthodes de livraison de Services Cloud

Comme dit précédemment certains fournisseurs de services Cloud livrent leur services sous forme de :

— **Service Web** : Adopter les services web comme Interface aux services Cloud.

— **API Rest ou Restful API** : Une API REST est une interface de programmation d'application (Application Programming Interface) qui utilise des requêtes HTTP pour POST, GET, PUT et DELETE [3] les données. Avec l'utilisation croissante du Cloud, les APIs sont utilisées par les fournisseurs cloud pour exposer et organiser l'accès aux services. L'architecture REST est un choix logique pour créer des API qui permettent aux utilisateurs de se connecter, de gérer et d'interagir avec les services cloud de manière flexible dans un environnement distribué.[19] Les API RESTful sont utilisées par des sites tels qu'Amazon, Google, LinkedIn et Twitter. Egalement Étant donné que de nombreux fournisseurs différents développent leurs propres interfaces, cela oblige coté développeur (les clients) ou agents logiciels de prendre en charge plusieurs APIs, ce qui veut dire que la modification d'une applications, nécessite des efforts supplémentaires de programmation [11]. Et il y a plus de risques d'erreurs lorsque trop d'API sont prises en charge.

3. POST, GET, PUT, DELETE sont des méthodes HTTP, qui correspondent au CRUD (Create pour créer, Read pour lire, Update pour mise à jour et Delete pour supprimer)

2.8 Langages de description de services

Les modèles de description de service sont généralement repartis en deux catégories qui sont : description à base syntaxique et sémantique.

2.8.1 WSDL

Comme vue dans la section sur l'Architecture des Services Web (section 2.3), l'un des langages les plus importants pour la description des services web est WSDL. Il se concentre principalement sur les aspects techniques tels que l'interface et les protocoles d'interaction, les messages et les modèles d'échange de messages. L'emplacement du service est également présenté. Bien que les attributs non fonctionnels ne soient pas définis dans WSDL[20]. Le WSDL dispose d'un manque de supports sémantiques pour indiquer le sens et les contraintes sémantiques des données impliquées dans les services Web, qui peuvent générer des ambiguïtés lors du processus de découverte de services.

2.8.2 WSDL-S

WSDL-S (Web Service Semantics) est une extension du WSDL traditionnel en ajoutant des aspects sémantiques. Il est le résultat de l'annotation sémantique de fichiers WSDL. D'après le W3C ces aspects sémantiques sont conservés en dehors des documents WSDL et sont référencés à partir du document WSDL via des éléments d'extensibilité WSDL.

2.8.3 OWL-S

OWL-S (Ontology Web Language for Service) est un langage de description de services qui enrichit les descriptions de services Web basées sur WSDL avec des informations sémantiques provenant d'ontologies OWL. OWL-S permet de décrire les services web de façon non-ambiguë et interprétable par des programmes [21]. OWL-S est aussi une ontologie OWL particulière qui définit une ontologie supérieure pour la description, l'invocation et la composition des services Web. La structure suivante résume la structure de l'ontologie supérieure.

— **Que fait le service** : Cette information est donnée dans le **Service Profile**. Il est utilisé à la fois par les fournisseurs pour publier leurs services et par les clients pour spécifier leurs besoins pour déterminer si le service répond aux besoins nécessaires. Service Profile est utilisé pour publier le service en décrivant ses propriétés fonctionnelles (par exemple, entrée, sortie, pré- condition, et les effets) et les propriétés non-fonctionnelles (par exemple, la confiance de service, la fiabilité, l'objet, le coût, etc.). [22]

— **Comment utilise-t-on le service** : cette information est fournie dans le **Service Model**. Il est utilisé pour décrire essentiellement le fonctionnement (modèle) d'un service composite. Il décrit comment demander le service et, en outre, il explique ce qui se passe lorsque le service est exécuté. OWL-S modélise les services en tant que processus défini par ses entrées/sorties. [17]

— **Comment accède-t-on au service** : cette information est donnée dans le **Service Grounding**. Celui-ci explique comment accéder concrètement au service et spécifie les détails concernant les protocoles de communication, les formats de messages et les adresses physiques, des détails spécifiques au service ou la façon de contacter le service. Ce type d'informations est particulièrement utile pour l'invocation automatique de services. [22]

2.8.4 L'OpenAPI

L'OpenAPI appelé autrefois Swagger est la plus grande référence dans le monde de la documentation d'API. Elle a été choisi par le projet Open API Initiative (OAI) [4] pour être la norme des documentations d'API en accord avec Google, IBM, Microsoft. L'OpenAPI est un ensemble de spécification qui permet d'écrire facilement la documentation d'une API REST/RESTful en JSON ou YAML [5], afin que les ressources et les utilisations soit facilement compréhensible par un humain comme par une machine.[23]

Il est doté de la capacité à piloter le cycle de vie entier de l'API, d'assurer une prise en charge des liens entre différentes opérations et des scénarios pour la lecture de l'API par un programme, avec également une meilleure description des paramètres.

Selon le GitHub du projet, La spécification OpenAPI (OAS) définit une description d'interface standard, indépendante du langage de programmation pour les API HTTP, qui permet aux humains et aux ordinateurs de découvrir et de comprendre les capacités d'un service sans avoir besoin d'accéder au code source, à une documentation supplémentaire ou à l'inspection du trafic réseau. [6]. Plusieurs étiquettes sont définit par la spécification qui permettent entre autres de :

— Définir les informations générales sur l'API : description, termes d'utilisation, licence, contact, etc. ;

— Fournir la liste des services qui seront offerts, avec pour chacun, comment les appeler et la structure de la réponse qui est retournée

— Définir le chemin pour consommer l'API ;

— Les méthodes d'authentification

4. La OpenAPI Initiative est un projet créé par la Fondation Linux afin d'améliorer les API. La version 3 d'Open API à vue le jour en 2017

5. Variante de JSON utilisant l'indentation à la place des accolades

6. https ://github.com/OAI/OpenAPI-Specification

— etc.

```yaml
swagger: '2.0'
info:
  version: Version de votre API
  title: Nom
  description: Description de l'API
  termsOfService: Termes d'utilisation
  contact:
    name: Nom personne de contact
    url: Site Web
    email: Adesse mail
  license:
    name: Nom de la licence
    url: Site Web ou tenir les informations sur la licence
basePath: "/"
paths:
  "Chemin de l'API":
    get:
      tags:
      - Nom du Tag
      summary: Resume de la methode qui est exposee
      description: Description de la m thode exposee
      operationId: Nom de la methode exposee
      consumes: []
      produces:
      - application/json
      responses:
        '200':
          description: Description de la reponse
          schema:
            "$ref": "#/definitions/ExempleDefinition"
definitions:
  ExempleDefinition:
    type: string
```

FIGURE 2.3 – Exemple de structure d'un document YAML OpenAPI

Étant donné que certains fournisseurs proposent leurs services Cloud au moyen de l'architecture REST, c'est à dire sous forme d'API REST ou Restful, ces APIs sont décrites et documentés en utilisant l'OpenAPI ou des variantes basées de l'OpenAPI.

2.9 Les propriétés d'un Service

D'après Youakim Badr et al. [24] on peut décrire un service Web entièrement par : Ses propriétés fonctionnelles et Ses propriétés non-fonctionnelles. Et chaque propriété non-fonctionnelle associée à un service peut être divisé en fonction de la qualité de service ou lié au contexte.

2.9.1 Propriétés fonctionnelles

Dans ce type il s'agit des propriétés relatives à la fonctionnalité du service. Ces propriétés sont ceux qui sont généralement décrites dans la description de service en termes d'opérations et reflètent le fonctionnement du service, ce sont les opérations que le service peut fournir.

2.9.2 Propriétés non-fonctionnelles

QoS (Quality of Service)

Les propriétés liées à la QoS (Qualité de Service) représentent un aspect très important des caractéristiques non fonctionnelles d'un service, et ses valeurs permettent la sélection des services pertinents aux demandes des utilisateurs. La QoS peut être divisée en deux catégories principales [24] :

1. La partie Exécution : qui inclut les paramètres de performance qui caractérisent l'interaction avec le service.
2. La partie Sécurité : qui est liée à la capacité d'un service donné à fournir des mécanismes de sécurité appropriés en tenant compte du chiffrement de l'authentification et du contrôle d'accès.

Propriétés liées au contexte

Dans cette partie nous avons les propriétés commerciales qui sont très pertinentes pour différencier deux services ayant les mêmes caractéristiques fonctionnelles et ceux liées à l'environnement.

1. les propriétés commerciales (Coût, Réputation, Mode de paiement etc.)
2. Les propriétés liées à l'environnement à savoir la localisation service et le temporel.

Les qualités de services (QoS) et le prix des services seront nécessaires et joueront un rôle plus important dans la recherche de services et la composition des services pour les services de Cloud computing.

QoS Propriétés	Location	Temps de réponse	le temps écoulé depuis la soumission d'une demande au moment où le la réponse est reçue
		Accessibilité	représente le degré que Le service Web est en mesure de répondre à une demande
		Conformité	Représente la mesure dans laquelle Le document WSDL suit WSDL spécification
		La réussite	Représente le nombre de demander des messages qui ont été répondus.
		Disponibilité	Représente le pourcentage de temps que fonctionne un service
	Sécurité	Chiffrement	La capacité d'un service Web à maintenir le cryptage des messages
		Authentification	la capacité d'un Web service pour offrir des mécanismes sert à l'identification de la partie qui a invoqué le service
		Contrôle d'accès	Si le service Web fournit des moyens de contrôle d'accès pour limiter l'invocation de l'opération et l'accès à information aux partis autorisés
Contexte Propriétés	Propriétés de commerce	Cout	Représente l'argent qu'un consommateur d'un service Web doit payer pour l'utiliser.
		Réputation	mesure la réputation de Services Web basés sur les commentaires des utilisateurs.
		Organisation arrangement	Inclut les préférences et l'historique (en cours partenariats)
		Méthode de paiement	Représente les méthodes de paiement acceptées par un service Web, c'est-à-dire banque de transfert, carte Visa etc.
		Surveillance	nécessaire pour un certain nombre d'objectifs, y compris l'optimisation des performances, vérification de l'état, débogage et dépannage.
	Propriétés d'environnement	Location	L'emplacement de service web.
		Temporel	Est-ce que le service est permanant ou temporel.

FIGURE 2.4 – Les propriétés non-fonctionnelles de service
[15]

Il faut noter que plusieurs services pour un besoin fonctionnel donné, peuvent assurent les mêmes taches. D'où l'importance des propriétés non-fonctionnelles qui permettrons de répondre à la question : *Quel est le service le plus performant entre plusieurs services équivalents ?*

2.10 Sélection des services web

Une fois le service Web décrit par son fournisseur, pour qu'il puisse être utilisé (et réutilisé) par le plus grand nombre de clients, il doit être publié dans un registre public. Avec la sélection des services, on cherche à choisir le meilleur fournisseur d'un service web ou cloud, étant donne un ensemble de fournisseurs de ce service qui est le plus adéquat selon un besoin. La sélection se base principalement sur la description de service qui est une interface de service et qui identifie un ensemble d'opérations et un ensemble de

propriétés pour qualifier le service. Dans la littérature, les objectifs de la sélection des services se classent généralement en deux types[25] :

— **Sélection du meilleur service :** le résultat de ce type de sélection est généralement le meilleur service alternatif ou un classement de service selon les exigences du consommateur demandant le service. Les résultats de la meilleure sélection de services sont généralement calculés pour les consommateurs finaux. Dans certains cas, les résultats de la sélection peuvent également être pris comme entrée pour la composition des services exécutée par les fournisseurs d'application cloud.

— **Sélection d'une composition optimale de services :** dans ce type de sélection de service, le système doit généralement sélectionner un groupe de services qui seront composés afin de répondre aux exigences en matière de fonctionnalités complexes.

2.11 La découverte des services

La découverte de service est un domaine de recherche vraiment active, particulièrement dans les Services Web durant la décennie passée. Les solutions pour une découverte automatique effective sont vraiment limités et les défis ont besoin d'être reconsidérer [26].

La découverte de service est nécessaire pour prendre en charge toute l'infrastructure de service telle que le Cloud Computing. Dans un environnement en nuage la découverte est rendue difficile par le nombre potentiellement important de services hétérogenèse disponibles ne respectant pas un standard donnée.

2.11.1 La découverte des Services Web

Les services Web sont découvertes simplement en collectant les interfaces des documents (les fichiers WSDL ou USDL [7]) en cherchant dans les registres UDDI.

2.11.2 La découverte des Services Cloud

Avec le Cloud computing la découverte des services est un peu plus complexe, et cela pour certaines raisons :

— Les Services Cloud sont offertes à différents niveaux (SaaS, PaaS, IaaS).

— Le manque de Standards pour la description et la publication des services cloud.

Par exemple d'après les résultats obtenus par Talal H. Noor et al. [18] en 2013, dans 5.883 services cloud découvertes, seulement 106 services cloud valides étaient implémentés en utilisant les standards WSDL et WADL [8].

7. Unified Service Description Language, La mission de l'USDL est de définir un langage pour décrire les parties générales et génériques des services techniques et commerciaux afin de permettre aux services

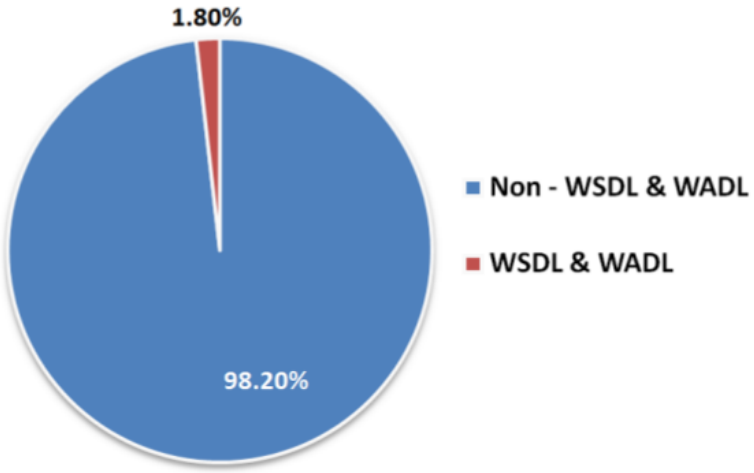

FIGURE 2.5 – Services Cloud en WSDL/WADL vs Services Cloud non en WSDL/WADL [18]

2.11.3 Les différentes approches de découvertes

Comme approches de découvertes, les plus utilisés sont : les approches basées sur les registres, qui sont des pôles centralisés où les informations d'un service peuvent être publier par les fournisseurs, et dans lesquels seul le propriétaire du registre a le droit de décider de quel type, quand et de quelle manière une information est placée sur le registre (exemple de la technologie UDDI) [9]. Les approches basées sur les index, contrairement aux registres ne sont pas dans la centralisation de la publication des informations relatives aux services web, et toute personne peut créer son propre système de publication de services, et y inclure toutes informations qu'elle désire sur les services qu'elle veut publier et par la suite vérifier et mettre à jours ces informations.

2.11.4 Les mécanismes de découvertes de services

Comme approche de mécanismes de découverte, on peut citer : la découverte syntaxique, celle sémantique et la découverte dynamique.

1. La découverte syntaxique est généralement basé sur la comparaison entre les mots clés de la requête du client avec ceux extraits de la description des services (Exemple WSDL).[9]. L'inconvénient majeur avec cette approche est que l'aspect sémantique de la requête n'est pas vraiment pris en compte
2. La découverte sémantique de services consiste à exploiter les différents concepts qu'on trouve dans une ontologie afin de calculer le degré de correspondance entre les mots clé de la requête et les mots clés de la description.

de devenir échangeables et consommables.

8. Le WADL (Web Application Description Language) est un langage informatique basé sur XML qui permet de décrire des applications REST.

3. On entend par découverte dynamique la possibilité de localiser automatiquement un service qui répond à des besoins particuliers. Différentes approches dynamiques ont été proposées dans la littérature.

2.12 Composition de service

Lorsque l'exigence de l'utilisateur est complexe, un service unique ne peut pas atteindre l'objectif. Par conséquent, au lieu de développer de nouveaux services complexes, il est préférable de combiner des services déjà existants de manière à pouvoir satisfaire toutes les exigences fonctionnelles et non fonctionnelles souhaitées par la requête complexe. Il existe deux types de services (atomique et composite).

— Un service atomique est une fonction autonome bien défini qui ne dépend pas du contexte ou de l'état des autres services.

— Un service composite est un assemblage de services atomiques ou d'autres services composites. Par exemple le Service composite *« Planifier un Voyage »* peut être composé de services individuel comme : *« Réserver billet d'avion », « Réservé Hôtel », « Réservé Taxi »*

La composition de services est le processus qui consiste à la collecte d'un ensemble des services pour satisfaire une demande de l'utilisateur [15]. Dans le Cloud computing, on appelle ce processus par lequel un service est construit à partir d'un ensemble de services sélectionnés, Cloud Computing Service Composition (CCSC). Le CCSC est tout un ensemble de stratégies ou de mécanismes pour composer des services sélectionnés dans un seul service afin de maximiser la satisfaction du client.

2.12.1 Cycle de vie de la composition

Le cycle de vie de la composition des services comme indiqué sur la figure 2.6 les différentes activités consistent à [27] :

1. Découvrir les services susceptibles de coopérer pour répondre à la requête traitée ;
2. Vérifier leur composabilité et générer les plans (solutions) de composition possibles ;
3. Sélectionner le ou les meilleurs plans qui répondent aux exigences et préférences du client en termes de propriétés non-fonctionnelles ;
4. Exécuter le service composite.
5. Publier les services composites

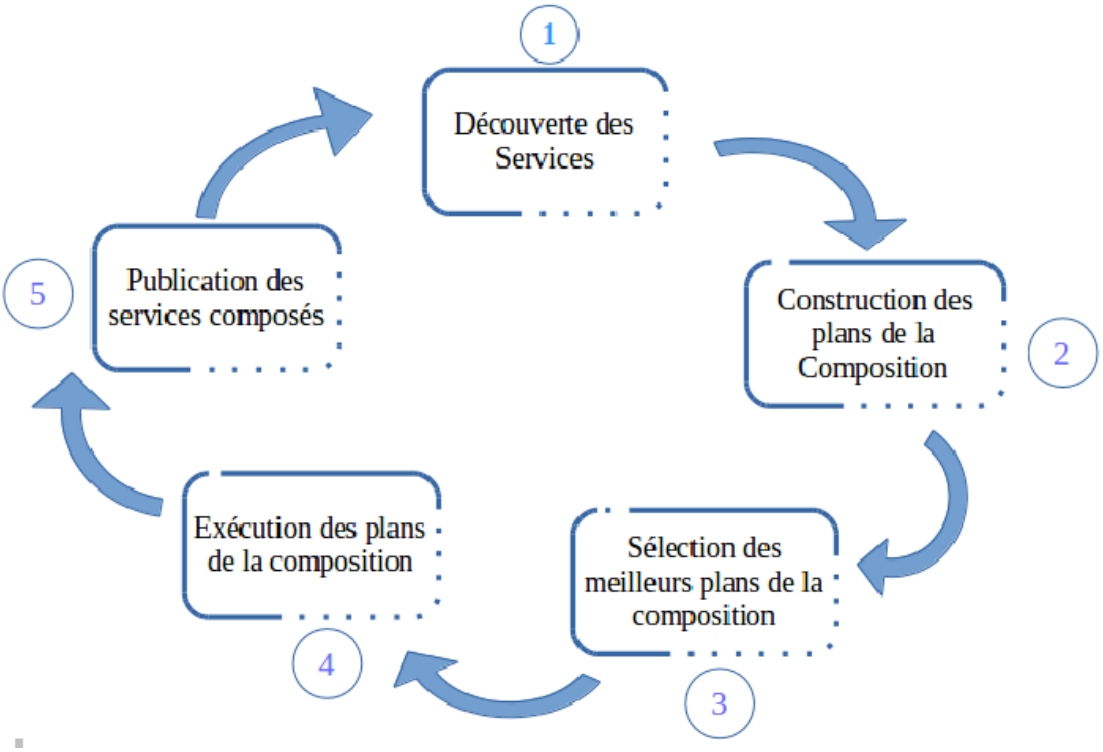

FIGURE 2.6 – Cycle de vie de la composition de service

2.13 Quelques travaux d'approche sur la découverte des services

Dans cette section nous présentons quelques travaux qui sont liés à la découverte, voir la composition des services. Ces travaux sont ceux qu'on a considérés pertinents pour notre recherche. La découverte et la sélection des services cloud sont considérées comme une tâche difficile pour différentes raisons. Les fournisseurs de cloud publient souvent les descriptions de leurs services, les politiques de tarification et les règles d'accord de niveau de service (SLA) sur leurs portails dans des formats divers et hétérogènes. Par conséquent, la plupart des services cloud disponibles sont fournis avec un format non standard (par exemple, documentation HTML et descriptions textuelles) [28].

De nombreux efforts de recherche ont été faits pour faciliter le processus de découverte et de sélection des services cloud. Ils peuvent être présentés sous deux points de vue différents, à savoir une vue architecturale et une vue matchmaking (jumelage ou de correspondance)

2.13.1 Vue Architecturale

A ce niveau l'architecture peut être réalisée en proposant un registre de services cloud utilisant une plateforme cloud broker.

Foued Jrad et al. [29] dans leur travail ont proposé une architecutre pour faciliter la

capacité des utilisateurs à trouver les services Cloud les plus adaptés en tenant compte de leurs exigences SLA fonctionnelles et non fonctionnelles. Pour cela un framework de courtier de services multiCloud agissant en tant que médiateur entre les consommateurs et plusieurs fournisseurs de Cloud pour automatiser la sélection et le déploiement des services a été conçu. Le framework contient des composants pour la prise de décision, la surveillance et la gestion des SLA, ainsi qu'une couche d'interopérabilité pour interagir avec des Clouds hétérogènes.

Les auteurs (Asma Musabah Alkalbaniand Kim, 2019)[30] proposent un module de collecte pour extraire les données du Web et les rendre disponibles dans différents formats de fichiers. Le module de récolte utilise un algorithme d'apprentissage de la structure HTML d'une page Web. Ce travail nécessite que l'utilisateur détermine des paramètres de contrôle spécifiques tels que l'URL de la page Web ciblée et les informations requises à partir de la page Web ciblée. De plus, les ensembles de données collectés manquent d'informations de service principales telles que la description et les opérations des services

2.13.2 La vue matchmaking

D'un autre côté, la vue de matchmaking est divisée en une vue syntaxique et une vue sémantique. Les approches basées sur la syntaxe sont généralement basées sur la description WSDL des services cloud.

Dans leur travaux Al-Masri et Mahmoud [31] ont procédé à la collecte des documents WSDL en explorant les registres commerciaux UDDI (UBR). Les services Web sont découverts en collectant simplement les documents d'interface (par exemple, les fichiers WSDL) et en recherchant les sous-répertoires des entreprises UDDI.

Certains chercheurs proposent d'utiliser des techniques d'ontologie pour la découverte de services cloud. Par exemple, [32] propose un système de découverte de services cloud (CSDS) qui exploite des techniques d'ontologie pour trouver des services cloud plus proches des besoins des consommateurs de services. En particulier, les auteurs proposent une ontologie du cloud où les agents sont utilisés pour effectuer plusieurs méthodes de raisonnement telles que le raisonnement par similitude, le raisonnement équivalent et le raisonnement numérique.

2.13.3 Moteur de recherche pour la découverte des services cloud

Dans l'article [18], les auteurs ont proposé un « Crawler Engine for Cloud Services Discovery »,un moteur de recherche de services cloud au travers de système de crawler qui explore les moteurs de recherche pour collecter les informations de service cloud disponibles sur le Web. Explorer le Web pour retrouver les services cloud en utilisant les moteurs de recherche existants, les API (Google, Yahoo, Baidu). Le moteur a été utilisé pour collecter les métadonnées de 5 883 services cloud valides via les moteurs de recherche

après avoir analysé plus d'un demi-million de liens possibles. Leur réalisation a été faite en se basant sur plusieurs questions d'entrée autour de la découverte de service cloud : *« Comment identifier si un service sur le web est un service cloud ? Combien de service sont actuellement disponible sur le web et qui sont les fournisseurs ? Quels types de fournisseurs de services sont sur le web ? Dans quelle mesure les normes du service-oriented computing contribuent elle au cloud ? »*. Leurs résultats fournissent des informations supplémentaires sur l'amélioration de la découverte des services cloud.

2.13.4 La composition de service comme un système multi-agent

J. Octavio Gutierrez-Garcia et Kwang-Mong Sim dans leur travail intitulé « Agent-based Service Composition in Cloud Computing » [**?**] ont proposé un système multi-agents pour la composition de services dans le cloud computing dont l'architecture est décrite dans la figure 2.7.

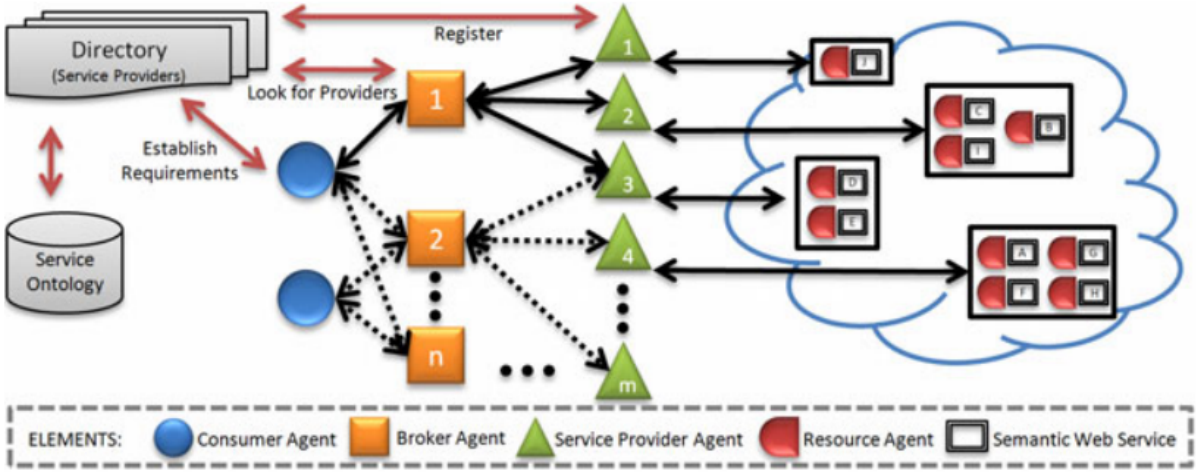

FIGURE 2.7 – Architecture du système multi-agents

— **Service ontology** est une représentation formelle des services et des exigences atomiques que ces services peuvent satisfaire.

— **Directory** (Annuaire) est une liste d'agents fournisseurs des services disponibles. Chaque entrée est associée à un fournisseur de services qui contient son nom, son emplacement (par exemple, l'adresse URI), et des capacités.

— **Semantic web service (SWS)** est un service Web dont la définition est mappée à une ontologie, qui caractérise la fonction et de son domaine d'application.

— **Consumer agent (CA)** : cet agent est chargé de soumettre les intérêts d'un consommateur au broker agent.

— **Resource agent (RA)** orchestre un service web sémantique. En outre, un RA contient une référence à une tache atomique, qui peut être remplie par ses SWS.

— **Service provider agent (SPA)** : cet agent gère un ensemble de RA. Sa fonction principale est de coordonner et synchroniser les RA.

— **Broker agent (BA)** accepte les demandes des agents des consommateurs CA et crée un seul service virtualisé par le biais de services composants déployés sur un ou plusieurs Clouds. Un BA utilise les exigences des consommateurs à la recherche de SPA, ensuite il démarre le processus de composition en adoptant le protocole de contrat net bien connu pour la sélection des services et l'allocation des tâches.

2.14 Conclusion

Le Cloud computing est récemment apparu comme la nouvelle génération de l'informatique distribuée, où son objectif est de concrétiser l'idée de fournir des ressources et des services informatique à la demande par internet. Dans ce chapitre nous avons eu à voir ces différentes technologies et approches qui ont fait du Cloud computing une possibilité. Les différentes services publiés dans le Cloud, ont pour objectif principale de satisfaire les besoins du client et pour cela il est important pour le consommateur de services que ça soit une personne physique ou un agent logiciel d'arriver à retrouver ces dites services, les comprendre dans le but de les utiliser. Nous avons eu à évoquer le problème liés à la description des services disponibles dans le Cloud due au manque de standards, qui rend le processus de découvertes, sélection, voir composition complexe, et nous avons également eu à faire ressortir quelques travaux existant dans ce sens. Parmi ces différentes approches il y a une solution évoquée qui est l'utilisation des systèmes multi-agents, c'est à dire l'utilisation des agents logiciels pour rendre la composition de service dans le Cloud possible. Dans le chapitre suivant allons voir en quoi consiste un système multi-agent, et comprendre ce concept de long en large, ensuite voir la technologie de crawler agent, dans le but de pouvoir élaborer notre solution qui il faut rappeler est la conception et l'implémentions d'un cloud crawler agent.

Chapitre 3

Agents, Système multi-agents et Agents Crawler

3.1 Introduction

Le domaine des systèmes multi-agents (SMAs) n'est pas récent, il est actuellement un champ de recherche très actif. Cette discipline est à la connexion de plusieurs domaines en particulier de l'intelligence artificielle, des systèmes informatiques distribues et du génie logiciel. 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 tournent autour de la coopération, de la concurrence ou de la coexistence entre ces agents [33].

3.2 Qu'est-ce qu'un agent ?

Le concept d'agent a été l'objet d'études pour plusieurs décennies dans différentes disciplines. Il a été non seulement utilisé dans les systèmes à base de connaissances, la robotique, le langage naturel et d'autres domaines de l'intelligence artificielle, mais aussi dans des disciplines comme la philosophie et la psychologie. La notion d'agent est centrale en IA. Son importance n'a cessé de croître depuis les années 80. Dans la littérature, on trouve une multitude de définitions d'agents. Elles se ressemblent toutes, mais différent selon le type d'application pour laquelle est conçu l'agent. Voici les premières définitions de l'agent selon Ferber [34].

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

part des ressources et des compétences dont il dispose, et d'autre part de ses propres représentations et des communications qu'il 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.

Pnueli a donné une définition qui représente l'architecture interne d'un agent [35] : « *Un agent est un processus cyclique composé de trois phases : perception, délibération et action.* »

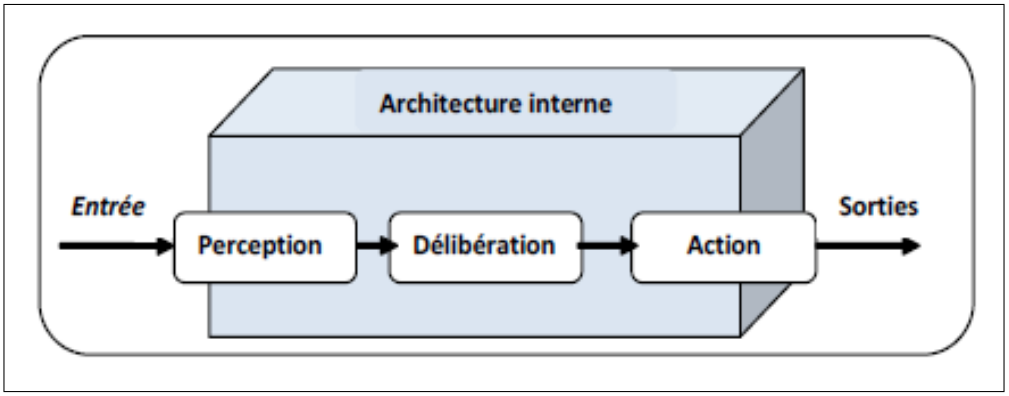

FIGURE 3.1 – l'architecture interne d'un agent

Récemment, Jennings,Sycara et Wooldridge ont proposé la définition suivante pour un agent [36] :

Un agent est un système informatique, situé dans un environnement, et qui agit d'une façon autonome et flexible pour atteindre les objectifs pour lesquels il a été conçu.

3.3 Les propriétés d'un agent

En partent des définitions et Ferber on peut distinguer les propriétés suivantes [36] :

- **Situé :** l'agent est capable de percevoir son environnement et y agir de façon limité ;
- **Autonome :** l'agent prend uniquement que les décisions sur son comportement et ne contrôle que ses propres actions ainsi que son état interne ;
- **Réactif :** être capable de percevoir son environnement et de réagir dans le temps ;
- **Proactif :** l'agent doit exhiber un comportement proactif et opportuniste, tout en étant capable de prendre l'initiative au bon moment.
- **Social :** l'agent est capable d'interagir pour atteindre ses buts et pour aider d'autres agents dans leurs activités ;
- **Intelligent.**

— **Capable de répandre à temps :** la capacité de percevoir son environnement et de réagir dans le temps.

Si un agent est capable d'agir de manière réactive, proactive et sociale alors il est un agent Flexible. Ce qui veut dire que l'agent a des réactions aux changements de l'environnement, et qu'il s'adapte aux ressources disponibles.

3.4 Types des agents

De manière générale, on distingue trois types d'agents. A savoir : les agents cognitifs, les agents réactifs et les agents Hybrides.

3.4.1 Agents cognitifs

C'est le premier modèle d'agents qui a été proposé. Les agents cognitifs ont des représentations explicites sur leur environnement, sur les autres agents avec lesquels ils communiquent et sur eux-mêmes. Ils savent tenir compte de leur passé et Ils sont souvent capables de planifier leurs actions et réagissent en fonction de leurs connaissances, leurs buts, leurs échanges d'informations avec les autres agents et de la perception de l'environnement.

3.4.2 Agents réactifs

Ils ont de très simples composantes et un comportement du type « stimulus - réponse ». Ils réagissent à leur perception de l'environnement et agissent en fonction de cette perception et ses buts interne. Ils ne sont pas capables de tenir en compte des actions passées.

3.4.3 Agents hybrides

Les agents hybrides sont conçus pour combiner des capacités réactives à des capacités cognitives où cette combinaison apporte plus d'avantages et moins d'inconvénients.

3.5 Définition d'un système multi-agents

Au premier abord, un système multi-agent (SMA) peux être considérer comme un système constitué d'un ensemble d'agents en interaction dans un environnement commun afin de réaliser leurs buts. Néanmoins, et pareille au concept « agent », il n'existe pas de définition acceptée en unanimité dans la littérature.

Wooldridge et Jennings [37], présente un SMA comme étant un ensemble d'agents en interaction afin de réaliser leurs buts ou d'accomplir leurs tâches. Les interactions peuvent

être directes par l'intermédiaire des communications, comme elles peuvent être indirectes via l'action et la perception de l'environnement. Les interactions peuvent être mises en oeuvre dans un but de :

— Coopération entre les agents, lorsqu'ils ont des buts communs.

— Coordination, pour éviter les conflits et tirer le maximum de profit de leurs interactions afin de réaliser leurs buts.

— Compétition, lorsque les agents ont des buts antagonistes.

[Demazeau, 1995] propose une décomposition d'un SMA en quatre dimensions qui correspondent aux quatre voyelles A, E, I et O, et qui est développée dans [Demazeau, 2001] [38] :

— **Agent (A) :** définition des modèles ou des architectures des composants du système.

— **Environnement (E) :** milieu dans lequel sont plongés les agents, est composé d'objets perçus et manipulés par les agents.

— **Interactions (I) :** ensemble des infrastructures, langages et protocoles d'interactions entre agents.

— **Organisation (O) :** structure des agents en groupes, hiérarchies, relations, etc.

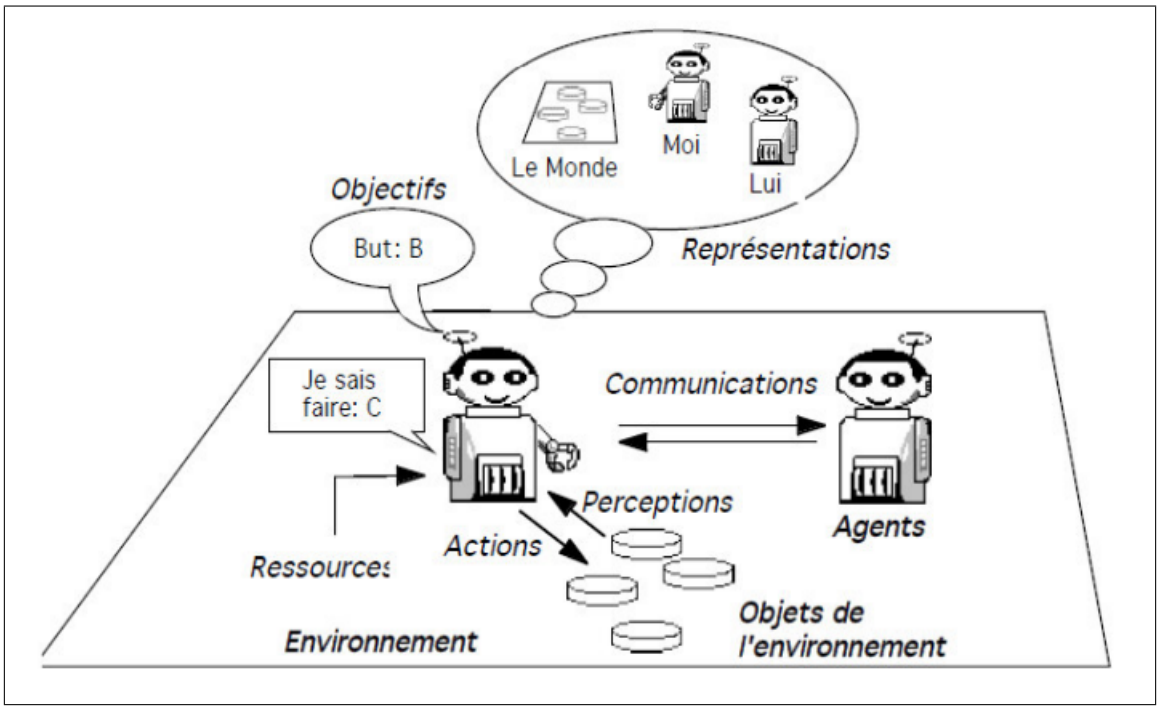

FIGURE 3.2 – Système multi-agents

Les entités en interaction avec un système multi-agent forme ce qu'on appelle *l'Environnement*, et leur ensemble à savoir SMA et Environnement est appelé *Monde*.

3.6 Les caractéristiques d'un système multi-agent

Les SMAs sont célèbres pour leur faculté à représenter et à s'adapter aux problèmes réels et de nature complexe. Ils le sont également grâce aux performances et caractéristiques que l'on y trouve [9] :

— **Fiabilité :** la distribution des tâches sur des agents spécifiques, facilite la résolution des défaillances d'un système rapidement et efficacement.

— **Extensibilité :** l'Indépendance des agents les uns vis à vis des autres facilite la modification de leurs comportements.

— **Robustesse :** la coopération entre agents permet au système de faire face à des situations incertaines et d'augmenter la tolérance aux fautes, vu que l'échec d'un agent n'a pas d'influence sur le fonctionnement des autres.

— **Maintenabilité :** L'inter-indépendance entre agents permet également de maintenir chaque agent séparément des autres sans affecter le fonctionnement global du système.

— **Evolutivité et Flexibilité :** la faculté d'auto-adaptabilité permet aux développeurs d'ajouter ou de supprimer de nouvelles contraintes (ou même de nouveaux agents) au SMA sans pour autant altérer le mécanisme global du système.

— **Efficacité :** la capacité de communication entre agents permet de développer des systèmes de calcul distribués très puissants et très efficaces.

— **Réduction des coûts :** un système centralisé est toujours gourmand en temps de développement et de maintenance, avec les SMA on peut décentraliser la gestion et l'exécution des tâches, en créant des sous-systèmes représentés par des agents spécifiques, ce qui réduit les coûts d'extension et de maintenance.

3.7 L'environnement dans un système multi-agents

Un agent ne peut exister sans environnement. C'est grâce à lui que les agents peuvent coexister et interagir. L'environnement doit pouvoir être perçu par les agents et ces derniers doivent pouvoir agir dessus et interagir au travers. Donc un environnement peut être considéré comme la représentation du monde dans lequel les agents se situent. Il est modifiable par des agents, soit de façon globale, soit en faisant la distinction entre objets passifs (soumis aux actions des agents) et entités actives (les agents) [39].

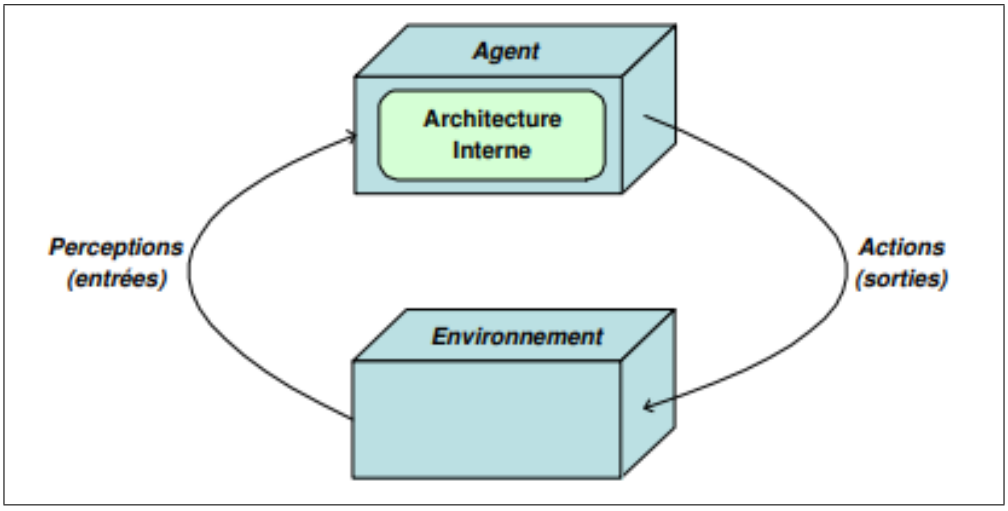

FIGURE 3.3 – Représentation classique d'un agent et de son environnement

Russell et Norvig distinguent pour un environnement les propriétés suivantes [39] :

— **Accessible/Inaccessible :** un environnement est dit accessible si les agents ont accès à l'intégralité de son état. L'accessibilité de l'environnement n'est pas une propriété souhaitable pour un SMA où la portée des actions et de la perception est locale.

— **Déterministe/Indéterministe :** un environnement est déterministe si un changement de son état est déterminé uniquement par son état courant et les actions des agents. Un environnement indéterministe pourra produire un résultat différent pour une même action.

— **Statique/Dynamique :** un environnement est dit statique s'il ne change d'état que sous l'effet des actions des agents. Au contraire, un environnement dynamique possède ses propres processus d'évolution qui peuvent modifier son état sans intervention des agents.

— **Discret/Continu :** l'environnement est discret lorsque le nombre de perceptions et d'actions possibles est limité. Si ce n'est pas le cas, l'environnement est alors continu.

3.8 Communication entre les agents

La communication désigne l'ensemble des processus physiques et psychologiques par lesquels s'effectue l'opération de mise en relation d'un agent (l'émetteur) avec un ou plusieurs agents (les récepteurs), dans l'intention d'attendre certains objectifs. La communication est caractérisée par l'échange d'information entre deux agents. Les processus physiques désignent les mécanismes d'exécution des actions (ex. l'envoi et la réception de messages), les processus psychologiques se rapportent aux transformations opérées par les

communications sur les buts et les croyances des agents. En général, les agents communiquent lorsqu'ils sont face à un problème qu'ils ne savent pas résoudre (soit par manque de compétences ou des ressources, voir informations), lorsqu'il est nécessaire de coordonner leurs actions, ou encore lorsqu'il y a un conflit entre plusieurs agents et que le conflit ne peut pas être résolu de façon déterministe. Les communications peuvent être diffusées à l'ensemble des agents ou à des agents particuliers (des agents susceptibles d'être intéressée par le message).

Il existe 3 modes de communication.

3.8.1 Communication par envoi des messages

Dans cette approche la communication se fait à l'aide des messages qui sont transférés entre les agents qui veulent communiqués. Malheureusement ce type de communication soufre de plusieurs inconvénients, le transfert de messages est coûteux en ressources de communication et c'est une approche difficile à mettre en œuvre dans des applications réelles, car il y a en général un fort degré d'incertitude au sujet des états actuel et futurs du monde. Le transport des messages est un point important dans les SMA : les agents "dialoguent" uniquement par échange de message, donc les performances du transport de messages influencent directement les performances du SMA[39].

3.8.2 Communication par partage d'information

C'est historiquement, le premier modèle de communication qui est apparu au début des années 60 [Newell, 62]. En intelligence artificielle la technique du tableau noir (blackboard) est très utilisée pour spécifier une mémoire partagée par divers systèmes. Ce mécanisme est constitué généralement de trois éléments principaux [39] :

1. Les *Sources de Connaissances* qui est le résultat du partage des connaissances du domaine ;
2. Le *blackboard* qui contient une description de l'état de la résolution sous formes d'entités appelées souvent *faits* ;
3. Le *Contrôle* qui permet de choisir, parmi les sources de connaissances dont la partie condition est vérifiée, celle dont la partie action sera exécutée, créant ainsi de nouvelles hypothèses dans le blackboard et permettant à la résolution de se poursuivre.

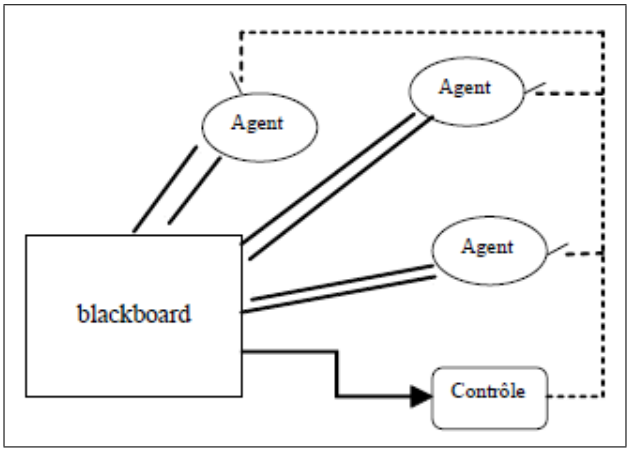

FIGURE 3.4 – Communication par partage d'informations

3.8.3 Les langages de communication

Il existe deux principaux langages de communication KQML (Knowledge Query Manipulation Language) développe en 1993 par le consortium DARPA [1]-KSE (Knowledge Sharing Effort), et ACL (Agent Communication Language) proposé en 1997 par la FIPA [2] (Foundation for Intelligent Physical Agents), qui ont été abordés afin de normaliser la communication entre agents.

KQML et ACL se distinguent au niveau de la sémantique des actes du langage utilisé. En effet, le langage ACL fortement inspiré des travaux de KQML, propose un langage auquel s'ajoute la définition de protocoles d'interactions.

Un message en FIPA ACL est constitué de champs obligatoires et facultatifs.

Exemple : L'agent A veut informer l'agent B du temps qu'il fera demain, selon ses prévisions :

1. Defense Advanced Research Projects Agency (DARPA) est une agence du département américain de la Défense responsable du développement de technologies émergentes destinées à l'armée
2. FIPA est un organisme de standardisation qui développe et définisse des normes de logiciels informatiques pour les agents et les systèmes basés sur des agents

performative :	type de l'acte communicatif
sender :	emetteur du message
receiver :	destinataire(s) du message
reply-to :	destinataire de la réponse au message
content :	contenu du message
language :	type de langage utilisé
encoding :	type de codage du message
ontology :	ontologie sur laquelle est basée le message
protocol :	type de protocole utilisé
conversation-id :	identifiant de la conversation
reply-with :	type de réponse souhaitée
in-reply-to :	nom de la requête
reply-by :	type de réponse

FIGURE 3.5 – structure d'un message FIPA-ACL

```
(inform
        :sender A
        :receiver B
        :content temps (demain, pleuvoir)
        :language Prolog
)
```

FIGURE 3.6 – exemple message FIFA-ACL

Lorsque deux agents vont communiquer, leur objectif est d'échanger des informations mais avant tout de se comprendre. Donc, quel que soit le langage ou la forme de communication, l'importance réside dans la possibilité pour un agent de pouvoir comprendre les autres agents. Dans la majorité des cas, la communication se fait par envoi de messages et parfois par envoi de signaux ou stimuli dans l'environnement.

3.9 Interaction entre les agents

La notion d'interaction est au centre de la problématique des SMA. Jacques Ferber [Ferber, 1995] définit l'interaction comme « Un ensemble de comportements résultant du regroupement d'agents qui doivent agir pour satisfaire leurs objectifs en tenant compte des contraintes provenant des ressources plus ou moins limitées dont ils disposent et de leurs compétences individuelles »[34].

3.10 Les Crawlers

Le WWW a une structure graphique dans le sens où les liens présents dans une page web peuvent être utilisés pour ouvrir d'autres pages. Internet étant comme un graphe orienté, la rechercher d'une information peut être résumée comme un processus de traversée d'un graphe orienté. C'est ainsi que les crawlers, ou robot d'indexation, beaucoup utilisé par les moteurs de recherches sont fait pour parcourir Internet généralement à partir d'une page web et cela à plusieurs fins.

3.10.1 Définition de Web Crawler

Crawler, Spider ou bien Robot d'indexation en Français est un programme / logiciel ou un script programmé qui parcourt le World Wide Web de manière systématique et automatisée, essentiellement utilisés pour créer une réplique de toutes les pages visitées qui sont ensuite traitées par un moteur de recherche qui indexera les pages téléchargées qui aident à la recherche rapide [40].

Un robot d'exploration est un composant essentiel du moteur de recherche qui télécharge les pages Web sur www en suivant les hyperliens dans les pages.

3.10.2 Fonctionnement d'un Web Crawler

Le fonctionnement d'un robot d'exploration Web peut être discuté en ces étapes suivantes [40] :

1. Selectionner une ou plusieurs URLs de départ ;
2. Ajouter l'URL à la frontière [3] (la Queue) ;
3. Choisir l'URL de la frontière ;
4. Récuper la page web correspondant à cette URL. Effectuer le traitement souhaité) ;
5. Analyser cette page Web pour trouver de nouveaux liens URLs ;
6. Ajout de toutes les URLs nouvellement trouvées dans la frontière ;
7. Passer à l'étape 3 et répéter jusqu'à ce que la frontière soit vide.

Les crawlers sont généralement actifs en permanence selon le principe de programmation et les objectifs recherchés, en visitant les pages en fonction des instructions qui leur sont données.

Nous avons comme exemple Googlebot qui est le robot d'indexation conçu pour le moteur de recherche du géant Google .Il se présente comme un grand programme avec la capacité d'analyser le web ainsi que les ressource d'un site internet en vue d'indexer les

3. En anglais frontier : c'est la liste des URLs à visiter, autrement dit à crawler.

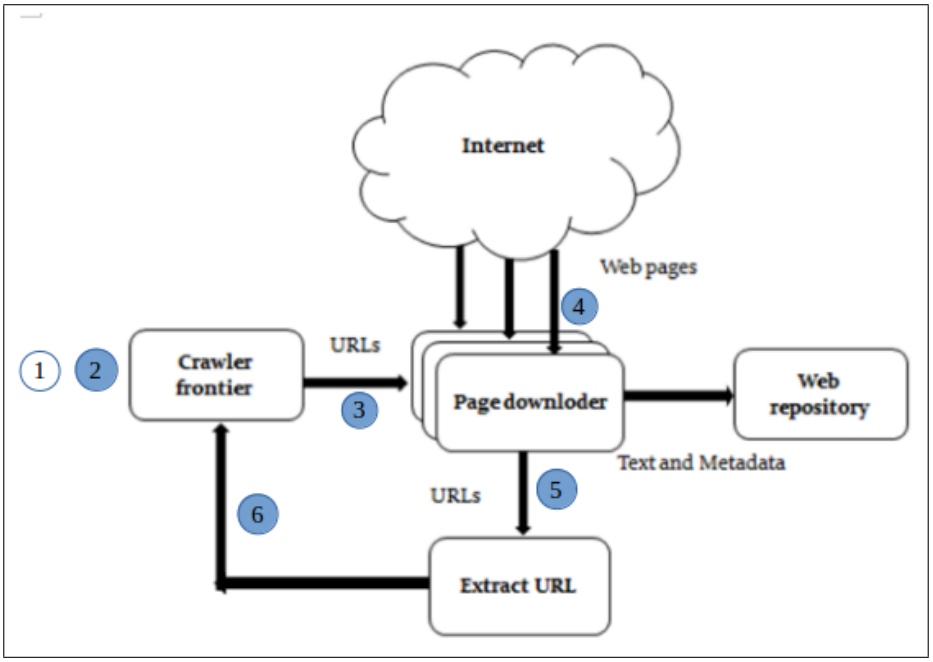

FIGURE 3.7 – Processus de Crawling

pages et cela de façon automatique. Son fonctionnement est basé sur un nombre conséquent d'algorithmes et sur des programmes informatiques qui déterminent les pages Web à explorer, à quelle fréquence, et le contenu à extraire de chaque site.

3.10.3 Les différentes techniques de crawling

Comme techniques d'exploration utilisées par les crawlers on peut citer [40] :

— **Le crawling à usage générale :** Dans ce cas de figure, un crawler à usage générale procède en collectant autant de pages que possible à partir d'un ensemble d'URL. En cela, le robot d'exploration est capable de récupérer un grand nombre de pages à partir d'emplacements différents. Cette technique peut s'avérer gourmande en bande passante du réseau, car elle récupère toutes les pages.

— **Le Crawling ciblé :** un crawler ciblé est conçu pour collecter des documents uniquement sur un sujet spécifique, en explorant uniquement les régions pertinentes du Web ce qui peut réduire le trafic réseau et les téléchargements. L'Objectif est de rechercher de manière sélective des pages appropriées à un ensemble prédéfini de sujets.

— **Le Crawling distribué :** dans l'analyse distribuée, plusieurs processus sont utilisés pour explorer et télécharger des pages à partir du Web.

3.10.4 Cloud Crawler

En appliquant les techniques de web crawling sur le cloud, nous avons ce que appel le Service crawler. Il est utilisé pour naviguer sur le Web et récupérer des documents contenant des informations sur la description des services déployé dans le cloud. Donc parlant de Cloud crawler, Il est question d'utiliser les techniques du web crawler mais dans l'objectif de parcourir internet à la recherche de la description des services.

Dans le domaine de la découverte et la composition des services cloud, dans certains travaux, comme [18], les techniques de web crawling ont été utilisé pour mettre en place un moteur de recherche de service cloud (Cloud Service Crawler Engine) utilisé pour collecter les métadonnées de services cloud valides après avoir analysé plus d'un demi-million de liens possibles cela en parcourant les pages web de service en analysant les informations dans le contenu Html. Mais la plus part de ces travaux sont beaucoup plus orienté agent humain, c'est à dire que ces services sont en majeure partie des cas compréhensible que par les humains. Nous dans notre travail l'objectif est de se focaliser sur les services compréhensibles et invocable par un agent logiciel.

3.11 Conclusion

Dans ce chapitre nous avons eu à évoquer la notion d'agent, de système multi-agent et d'agent crawler. Ses agents étant des entités pro-actives dotées d'un comportement intelligent et pouvant communiquer, oeuvrant à coordonner leurs buts et leurs plans d'action pour résoudre un problème donné. Avec le modèle de cloud computing qui est un modèle de fourniture de services, on peut, en utilisant des techniques de crawling web en parcourant le cloud, arriver à récupérer les descriptions des services publiés via un système d'agents logiciels et tenir une base de données facilement accessible dans le but d'assurer une éventuelle composition de service. Le chapitre suivant fera l'objet de proposer une conception à ce dit système.

Chapitre 4

Conception du système Crawler Agent

4.1 Introduction

Après avoir vu et étudié les concepts de base en ce qui concerne le cloud computing, les services déployés dans le cloud et effectué une études des solutions de compositions disponible notamment celle à bases des agents logiciels, nous allons dans ce chapitre présenter notre approche de crawling au travers d'un Système d'agents crawler. Nous avons opter pour la conception d'un Système d'agents crawler car l'utilisation d'un agent unique est inutile, du fait que cette activité est un peut compliqué et nécessite une décomposition en plusieurs rôles. [41].

4.2 Objectif visé par notre approche

Notre objectif est de développer un crawler agent pour la mise en place d'un système de découverte de services, qui vas parcourir le cloud à la recherche des services cloud disponible, dans le but de tenir une base de donnée des services crawlés. Cette base de données pourra par la suite être utilisées par d'autres agents participant à la composition de services, au bénéfice des utilisateurs, et procéder à une sélection et composition des services pour répondre à la demande de l'utilisateur. L'intérêt de ce travail est de proposer un mécanisme permettant de faciliter le processus de composition de service, en fournissant une base de données des services . Les différentes descriptions de services collectés doivent être compris par un agent logiciel *(machine-readable)*.La base de données fournira les informations, les attributs fonctionnelle, et non-fonctionnelle des services, permettant au processus de composition de ne plus parcourir tout le cloud à nouveau pour découvrir les services, limitant ainsi le temps de découverte, et évitant des tâches supplémentaires aux agents de découvertes.

D'abord il nous faut noter que notre agent fait partie en réalité d'un système Multi-Agent dédié à la composition de services-Cloud atomiques pour construire de nouveaux

services composés répondant efficacement aux besoins des utilisateurs finaux.

4.3 Scénario du processus de déroulement

Le scénario du processus de déroulement est représenté par l'algorithme sur la figure 4.1 qui est défini en 4 étapes essentiels.

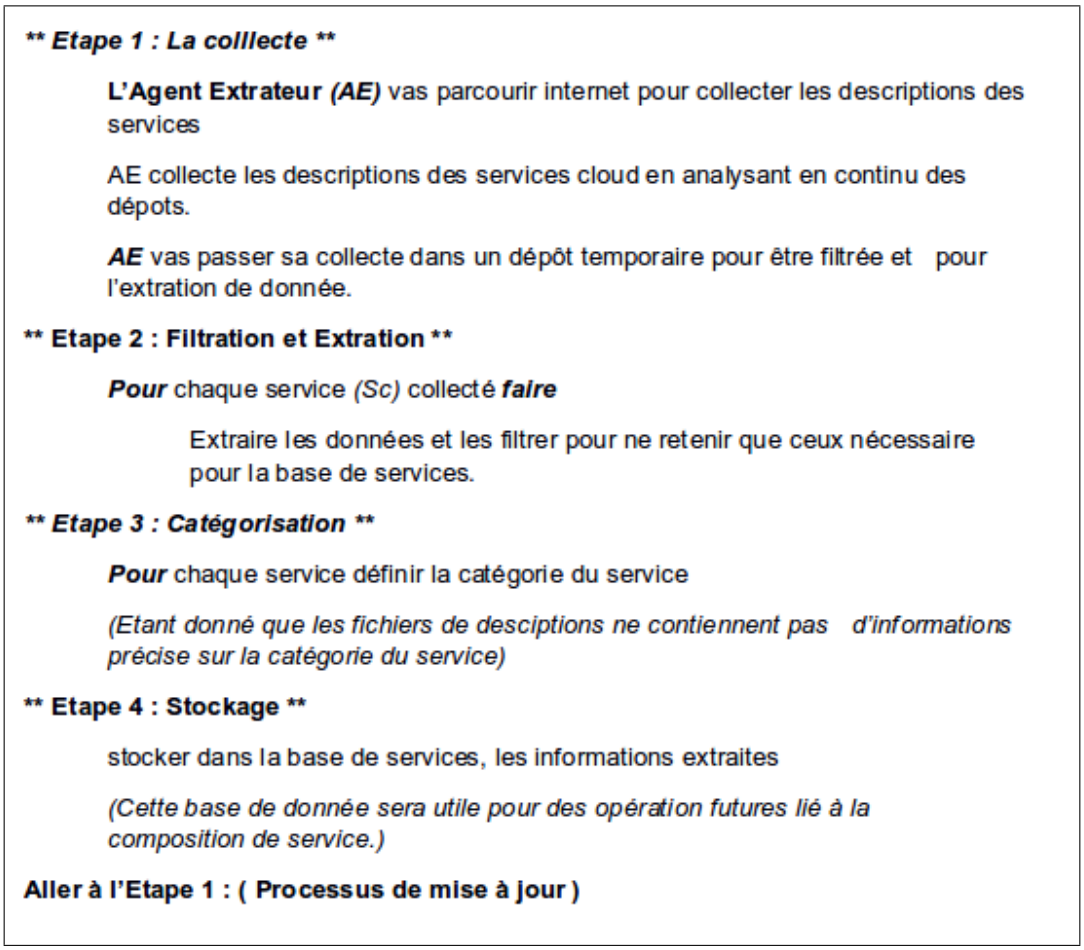

FIGURE 4.1 – Algorithme du scénario de crawling

Le processus de mise à jour se base sur l'accès périodique aux dépôt visités en vérifiant si une quelconque modification dans le fichier de description a eu lieu.

4.4 Architecture du Cloud crawler agent proposé

L'architecture du Cloud Crawler (robot d'exploration Cloud) proposé est illustrée à la figure 4.2. Le Cloud crawler agent en lui même est un systèmes composée de plusieurs agents. D'où nous pouvons parlé d'un système Cloud crawler.

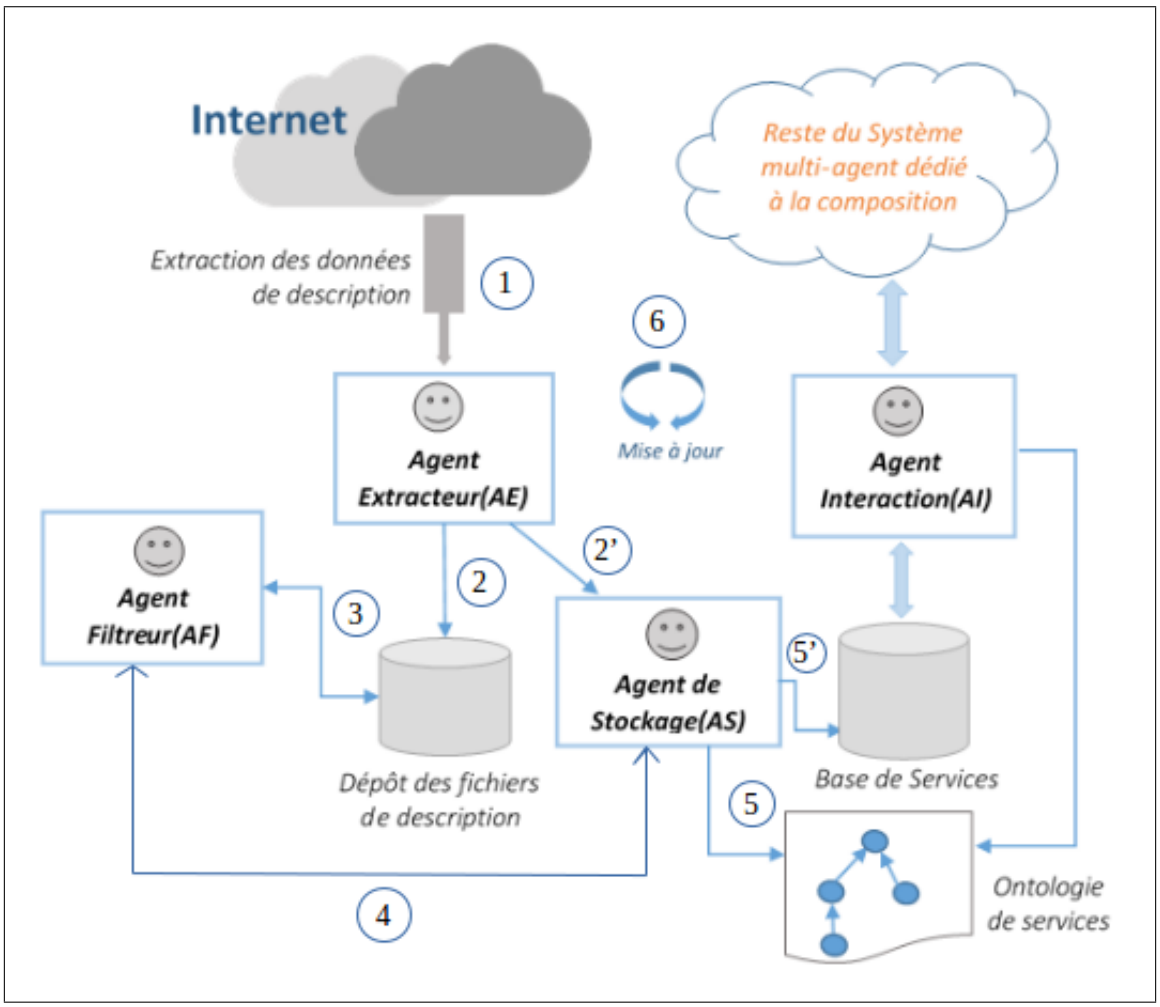

FIGURE 4.2 – Architecture - cloud crawler agent proposé

— **Agent Extracteur (AE) :** Cet agent se chargera de parcourir le web, à la recherche des descriptions des différentes services sur le cloud lisible par la machine. Due au manque d'utilisation de standard, pour la description des services disponibles. Nous aurons plusieurs instances d'AE. C'est à dire Un agent AEi explore un site de cloud données et extrait les descriptions.

Exemple : Un AEi qui sera assigné au service cloud de Google à l'aide d'un API d'extraction. AEj qui utilisera un API d'extraction pour extraire la description des services ou API proposées par Microsoft. Un AEk qui vas parcourir internet pour récupérer les fichiers OWL-S . Pour cela dans l'architecture nous avons une une base d'agents avec des hôtes qui leur sont assignés.

— **Agent Filtreur (AF) :** Cet agent est chargé de filtrer les différentes données collectés, et y retirer les informations importante et laisser les informations non nécessaires. Cet agent va se baser sur une liste de paramètres utile prédéfinie, récupérer les informations utiles. Ces données filtrées seront catégorisées et stockées.

— **Agent de Stockage (AS) :** Les données des services cloud sont stockées dans la

base de services correspondantes au travers de l'agent de Stockage. Une couche sémantique à savoir la catégorie à laquelle appartienne un service donnée, est ajoutée. Ainsi par exemple les services dédiés au stockage seront catégorisé dans ce sens.

— **Agent d'Interaction (AI) :** Cet agent servira d'interface avec les autres agents du Systèmes multi-Agent dédié à la composition de service d'avoir accès à la base de données des services dans le but de réaliser la composition. AI a une compréhension de la base de donnée, et définit un protocole de communication claire avec les autres agents dédiés à la composition. L'objectif recherché par ce agent est d'assurer l'interopérabilité de notre système.

— **L'ontologie de Service :** L'ontologie de Services qui est utilisée dans notre système représente les relations entre les services Cloud pour faciliter le reasoning, c'est-à-dire une correspondance lors de la recherche d'un service dans la base de données. Ainsi avec cette ontologie cloud, la similitude entre les services cloud sera déterminée. Il s'agit d'une ontologie, mis à jour par *l'agent de stockage* et consultée par *l'agent d'interaction*, afin de retrouver facilement dans la base de données un service demandé .

  Cette ontologie contient des données qui rendent le contenu de la base de données significatif et permet d'affiner le processus de recherche qui met en correspondance une demande de service et les services disponibles.

4.5 Diagramme de communication entre agents

Le diagramme de communication entre agents défini les interactions existantes entre les agents ainsi que les messages qui en résultent. Nous avons deux cas de figures.

— **En cas de découverte comme décrit dans la figure 4.3 :**

1. Découvrir les nouveaux services via l'Extracteur.
2. Enregistrer le fichier dans le dépôt, qui est en fait un dossier de fichiers. Lors de l'enregistrement l'extracteur va vérifier si le fichier crawlé existe déjà dans le dépôt, sinon, c'est qu'il s'agit d'un nouveau service, si oui il vérifie si son contenu a été modifié. Si le contenu a été modifier alors c'est que le service a été mis à jour, sinon le fichier n'est pas sauvegardé. Nous avons choisi l'url à laquelle le fichier a été extrait comme nom du fichier de description et comme id du service, ainsi nous nous assurons d'avoir un nom et un id unique.
3. Notifier le filtreur de la disponibilité des fichiers de description avec l'id des services.
4. Filtrer les nouveaux services, en se passant de certaines informations non nécessaires (le filtreur récupère les données significatives).

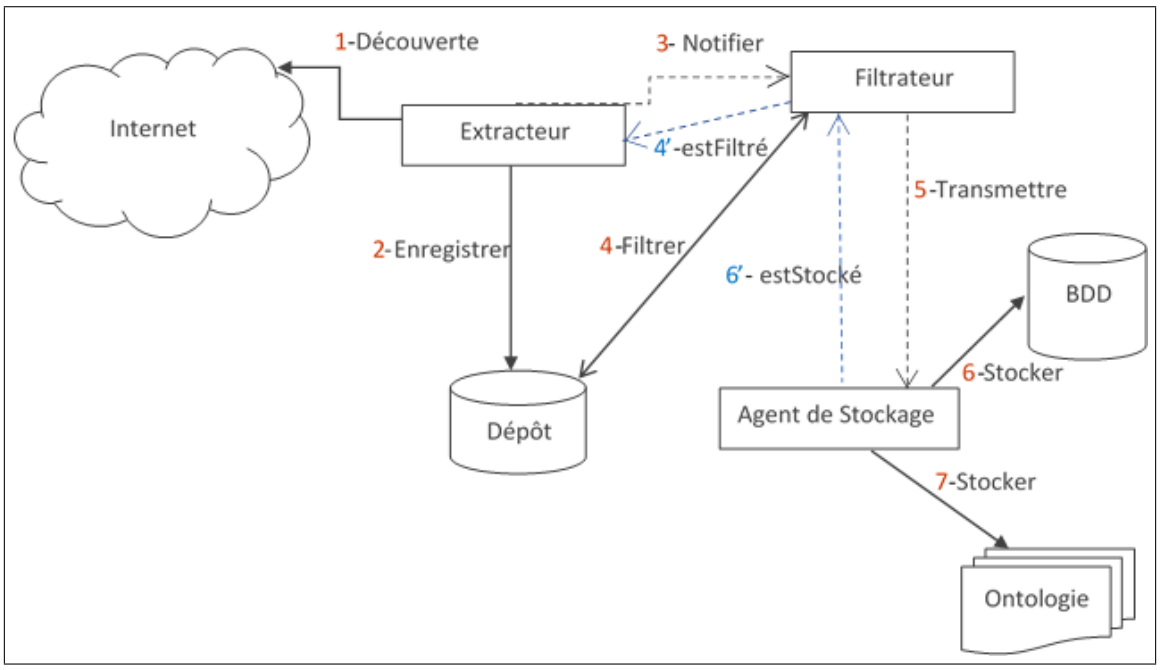

FIGURE 4.3 – Diagramme de communication entre agents dans le cas de la découverte

5. Transmission de ces données à l'agent de stockage pour être stocké.
6. Stocker les données du service dans la base de données.
7. Stocker les données du service dans l'ontologie.

— **En cas de suppression comme décrit dans la figure 4.4 :**

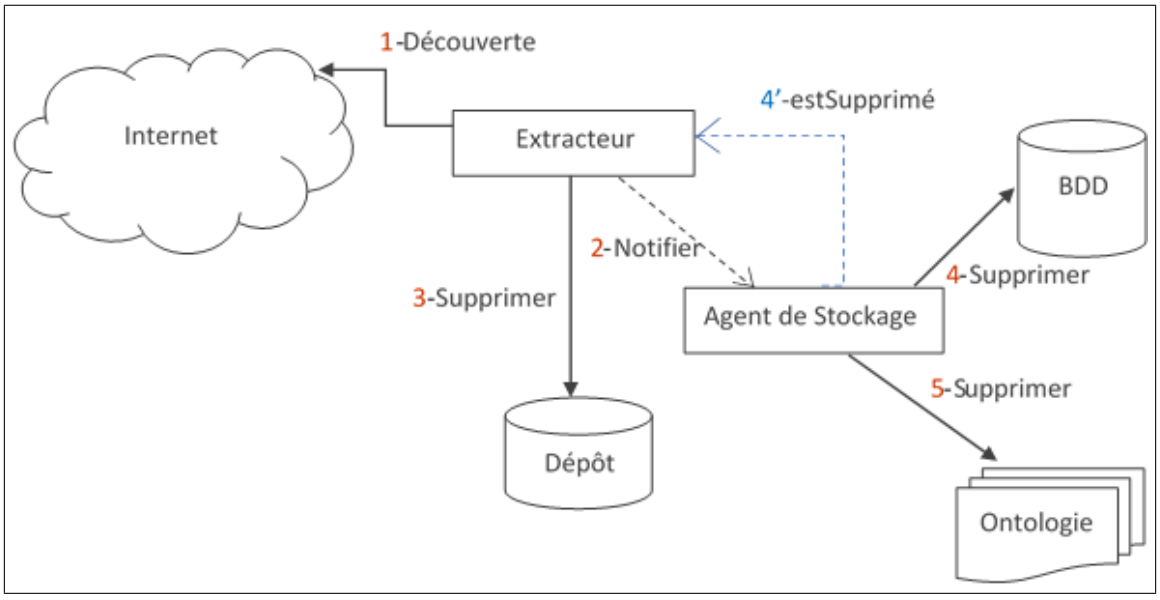

FIGURE 4.4 – Diagramme de communication entre agents dans le cas de la suppression

1. L'extracteur va se rendre sur les zones des services. Selon la technique des crawler, un crawler dispose d'une liste des liens visités (frontières). En se basant sur cette liste, si l'agent extracteur trouve un lien cassé (broken link) cela veut dire que le service a été supprimé.
2. S'il trouve un fichier de description supprimé, l'agent de stockage est notifié.
3. Le fichier est supprimé dans le dépôt.
4. Les données de ce dit service sont retirées de la base de données.
5. Les données du service sont retirées de l'ontologie.

4.6 Ontologie de communication entre agents

La communication inter-agent est fondamentale. Les agents communiquent entre eux au travers d'une conversation FIPA. Cette conversation, qui est composée d'une série de messages est basée sur une ontologie qui garantit qu'une expression a la même signification pour tous les agents [42]. L'ontologie proposée dans la figure 4.5 contient les données qui rendent le contenu du message significatif pour les agents du système.

Les messages sont tous relative au même concept qui est le service. Par exemple quand l'AE demande au filtreur de filtrer un nouveau service ajouté au dépôt, il envoie l'ID de ce service sous forme d'un message ACL. Par exemple *request(Filtrer(service_id))*.

Dans le stockage le filtreur transmet à l'agent de stockage l'id de service qui est l'URL, son type, le protocole utilisé, qui sont des données génériques présentes dans tous les fichiers de description, la liste des paramètres liés au service (fournisseur, coût, etc.), ainsi que les différentes opérations liés à ce service. Par exemple : supposons un service à stocker avec ses données suivantes : l'URL, le type, le protocole, le coût, le fournisseur etc. Le message que l'agent de stockage vas recevoir est de la forme :

Stocker(Service_id, type, protocole, [liste_des_données]) avec liste_des_données qui est composée des données du service, leur type et leur valeur.

Stocker(accesscontextmanager.googleapis.com, RESTful API, REST, [(nom, fournisseur, coût,...) ;(string, string, int,...) ;(Accesscontext Manager, Google, 30,...)]

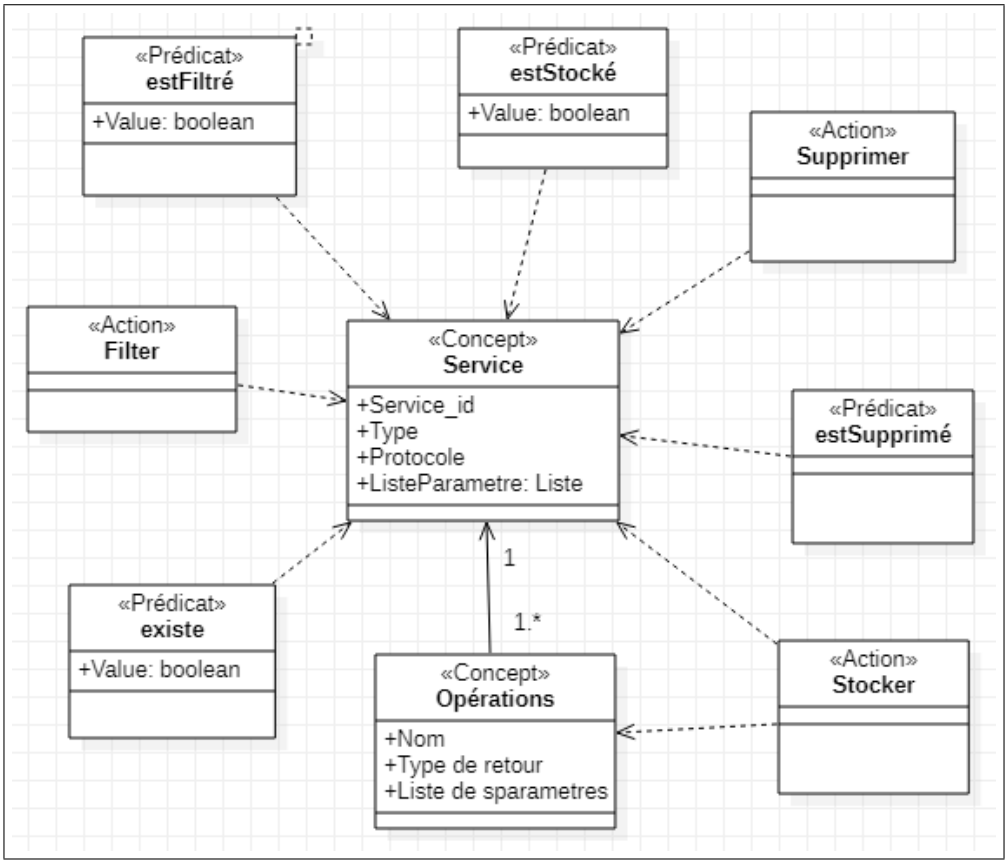

FIGURE 4.5 – Ontologie de communication entre agents

4.7 La structure de l'agent de stockage

Dans la figure 4.6 nous avons présenté la structure de l'agent de stockage.

— L'agent de Stockage reçoit les messages via son interface de communication.

— Ces messages sont analysés au niveau de l'analyseur d'évènement. Par rapport au type de message reçu une action (comportement) est exécutée.

— Si le message reçu est une requête avec nom d'action = stocker envoyé par le filtreur, le comportement 1 est exécuté. Si le message reçu provient de l'extracteur signalant la suppression d'un service qui n'existe plus alors le comportement 2 est exécuté.

— Avec le comportement 1, le module de catégorisation est utilisé (voir section 4.8), qui consulte WordNet (WordNet et WordNet Domain) pour déterminer le domaine d'appartenance d'un service. Le résultat est ajouté dans la base de données de service ainsi que les données relatives au service, et dans l'ontologie.

— Avec le comportement 2, qui est exécuté lors de la réception de l'id d'un service à supprimer via l'agent d'extraction, ce dit service est supprimé.

— Dans les deux cas le 3ème comportement est exécuté à la fin pour répondre aux autres agents pour les notifié de la réalisation de l'action demandé.

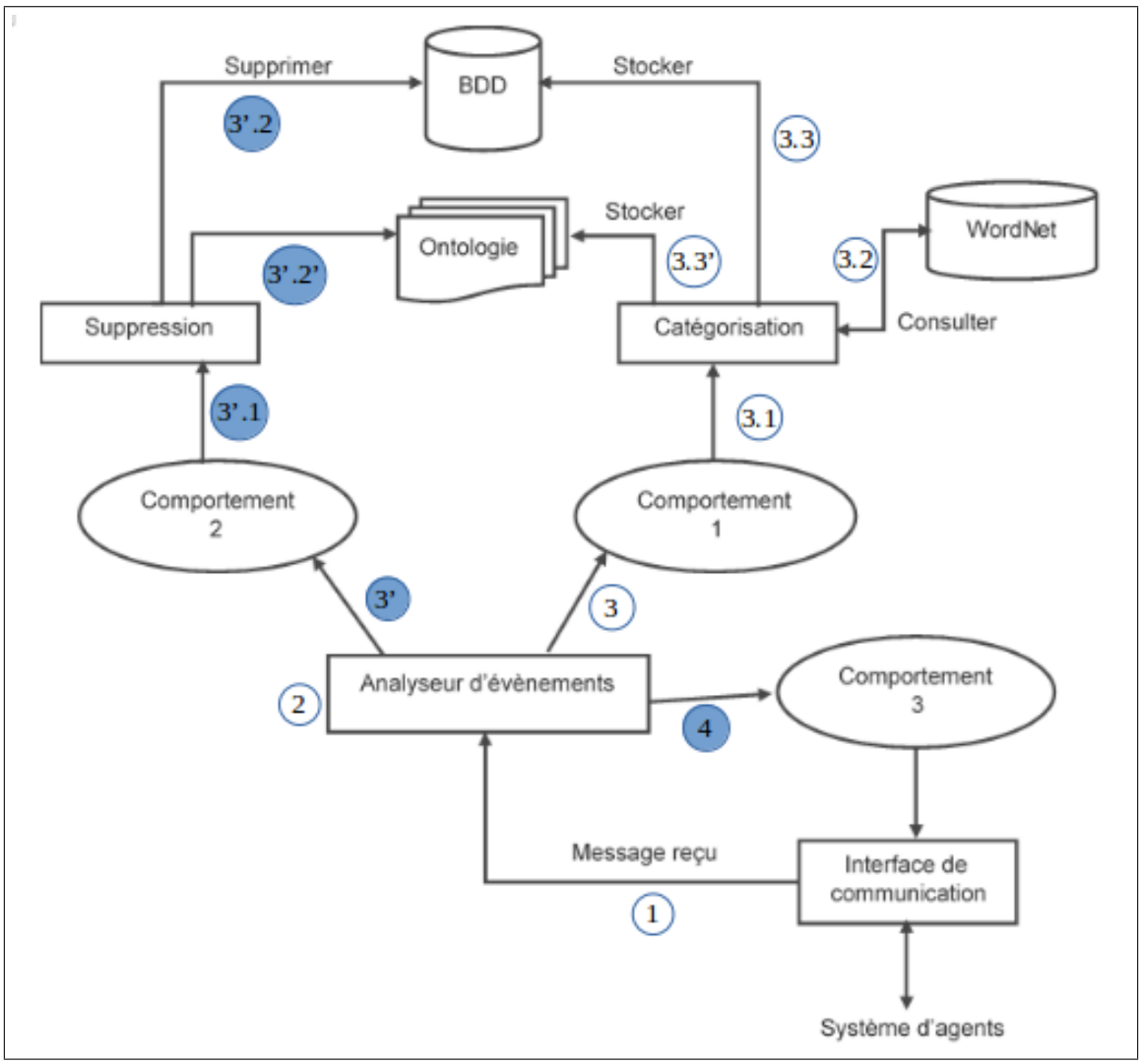

FIGURE 4.6 – Schéma de la structure de l'agent de stockage

4.8 Catégorisation des Services

La catégorisation des services est basée sur la technique de catégorisation de textes, qui permet de chercher une liaison fonctionnelle entre un ensemble de textes et un ensemble de catégories [43]. Dans notre approche pour la conception nous avons procédé pour une catégorisation des services en fonction de leur domaine en utilisant WordNet.

WordNet est une base de données lexicale qui a pour but de répertorier, classifier et mettre en relation de diverses manières le contenu sémantique et lexical de la langue anglaise, mais il existe aussi ses variantes dans les autres langues. WordNet organise des synsets (synonym sets ou ensemble de synonymes) de noms et de verbes sous forme d'hypernymes et d'hyponymes (hypernyms and hyponyms) [44]. En effet entre les mots, il y a parfois des rapports de hiérarchie. On parle d'hyperonymie quand un terme général ou terme générique englobe plusieurs termes spécifiques, appelés hyponymes. *Exemple : «Fruit» est un hyperonyme, qui englobe «orange», «banane », etc. «Banane» est un hy-*

ponyme de «fruit», tout comme «orange». Banane» et «orange» sont des cohyponymes car ils partagent le même hyperonyme. L'hyponymie est représentée dans WordNet par le symbole *«@»*, qui est interprété par *«is-a» ou «same-as» («est-un» ou «est-une-sorte-de»).*

Étant donné un service en entrée, les principales étapes de la catégorisation comprennent comme indiqué dans la figure 4.7 : une étape de prétraitement, une étape de désambiguïsation lexicale et enfin la catégorisation.

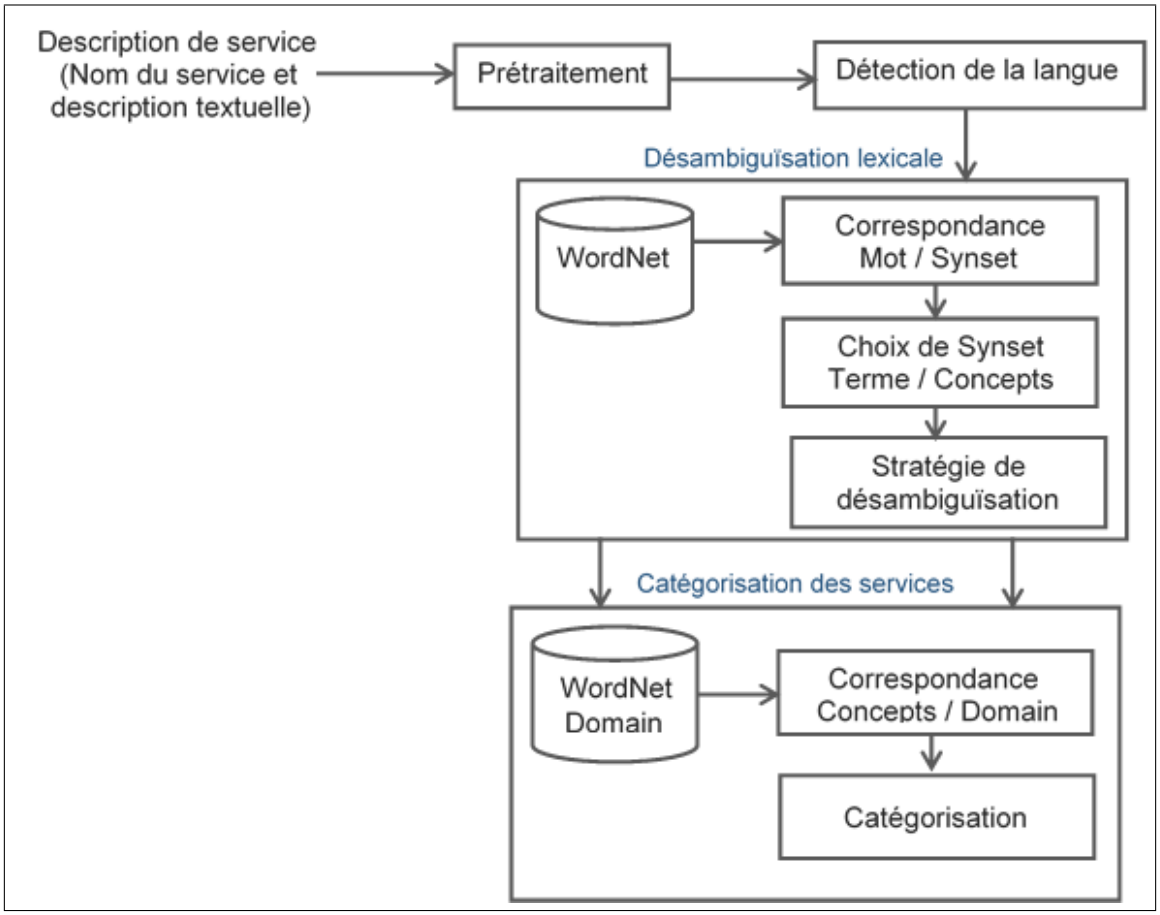

FIGURE 4.7 – Modèle de notre approche sémantique pour la catégorisation des services

— **La phase de prétraitement** consiste à nettoyer les données d'entrée : Suppression des ponctuations, les caractères numérique, et les caractères spéciaux, et est suivie par une étape de détermination de la langue utilisée. Déterminer la langue est importante vue que WordNet est de base fait pour la langue anglaise, sa variante française est WordNet libre du français (WOLF).

— **La phase de désambiguïsation lexicale** est la tâche qui consiste à définir le sens d'un mot ambigu dans un contexte, soit Word Sense Disambiguation (WSD). Dans cette phase, tout d'abord une correspondance est faite, entre les mots et les synset de WordNet (mapping Word/synset), ensuite il les termes sont cartographier en concepts, qui est un processus de correspondance des termes en concepts (associer

les termes à leurs concepts). Le problème de la désambiguïsation de sens est relatif au fait que l'attribution de termes aux concepts est ambiguë. Par conséquent, un mot peut avoir plusieurs significations et donc un mot peut-être mappé en plusieurs concepts il faut alors déterminer dans ce cas quel sens est utilisé. Pour cela des stratégies existent, comme la méthode du *First Concept* (Premier concept) qui va consister à ne considérer que le sens le plus souvent utilisé du mot comme le concept le plus approprié. Cette stratégie est basée sur l'hypothèse que l'ontologie utilisée via WordNet renvoie une liste ordonnée de concepts dans lesquels les significations les plus courantes sont listées avant les moins courantes, stratégie proposé dans [45].

— **Dans la phase de catégorisation des services**, il est question en premier lieu de faire la correspondance concepts/Domaine avec WordNet Domain [1]. Wordnet Domain est une ressource lexicale créée de manière semi-automatique avec une extension de la portée de Wordnet avec des étiquettes de domaine. Dans ce travail, nous avons appliqué les domaines extraits de Wordnet Domain pour classer les services par domaine. L'étape de catégorisation consiste à assigner une catégorie a un service donnée.

Une catégorisation plus détaillée peut également être faite par les opérations des services, en utilisant les inputs et les outputs. Par exemple si on a un service X *NomPaysParCode* qui donne le nom d'un Pays via le code téléphonique, et un service Y *ListVilleParNomPays* qui donne la liste des villes d'un pays à travers le nom du pays. Le output du service X peut correspondre à l'Input du service Y. Il y a ainsi une correspondance entre ces deux opérations.

4.9 La Base de Données de Service

Les différents documents de description crawlé (à savoir explorer), après leur analyse sont décomposés et les principaux éléments de ses documents sont traités pour les stocker respectivement dans la base de données.

Pour une découverte et une composition des meilleurs services il est nécessaire d'avoir une description plus ou moins complète des services Cloud. Pour notre système, nous proposons une description des services avec :

Chaque service, fourni par un fournisseur *(Service_provider)* et qui est catégorisé dans un domaine *(Domain)* spécifique. Un service est composé d'opérations *(Operation)*, et chaque opération nécessite des paramètres *(OperationParameter)*. Chaque service possède des paramètres *(ServiceParameter)* qui sont entre autre : le nom du service, sa description, le coût etc. ainsi que leur valeur.

1. http ://wndomains.fbk.eu/

Comme données fonctionnels qui nous permettent de décrire ce que fais un service, nous avons :

— **Description :** est la description textuelle de la fonctionnalités du service.

— **Operation :** les différentes opérations permisent par le service.

— **OperationParameter :** les différents paramètres nécessaires pour l'invocation d'une méthode donnée du service (les inputs).

— **Response :** les réponses éventuelles fournies par le service, qui dépendent de la méthode invoquée.

— **Protocol :** le type de protocole utilisé. A savoir le protocole utilisé pour l'invocation du dit service

Comme paramètres non fonctionnels qui définissent le niveau de qualité offerte par un service, ainsi que les paramètres liés à l'aspect business, nous avons :

— **Response_time :** temps de réponse.

— **Authentication :** le système d'authentification s'il y a.

— **Provider_name :** nom du fournisseur du service cloud.

— **Provider_domain :** adresse ou domaine du fournisseur.

— **Location :** emplacement du service.

— **Payment :** défini le type de payement pour l'utilisation. *Free* pour utilisation gratuite, *Pay per use* pour payement à l'utilisation, *Dynamic* pour un payement dynamique.

— **Price :** coût du service.

Vue la limite d'utilisation de standards dans la description par les fournisseurs de service, et aussi le fait que certaines informations sont non diffusées au grand public, les données inscrites dans les descriptions ne sont pas les mêmes d'un fournisseur à l'autre, et d'un langage à l'autre, mais un certains nombre d'élément générique apparaisse dans la plus part.

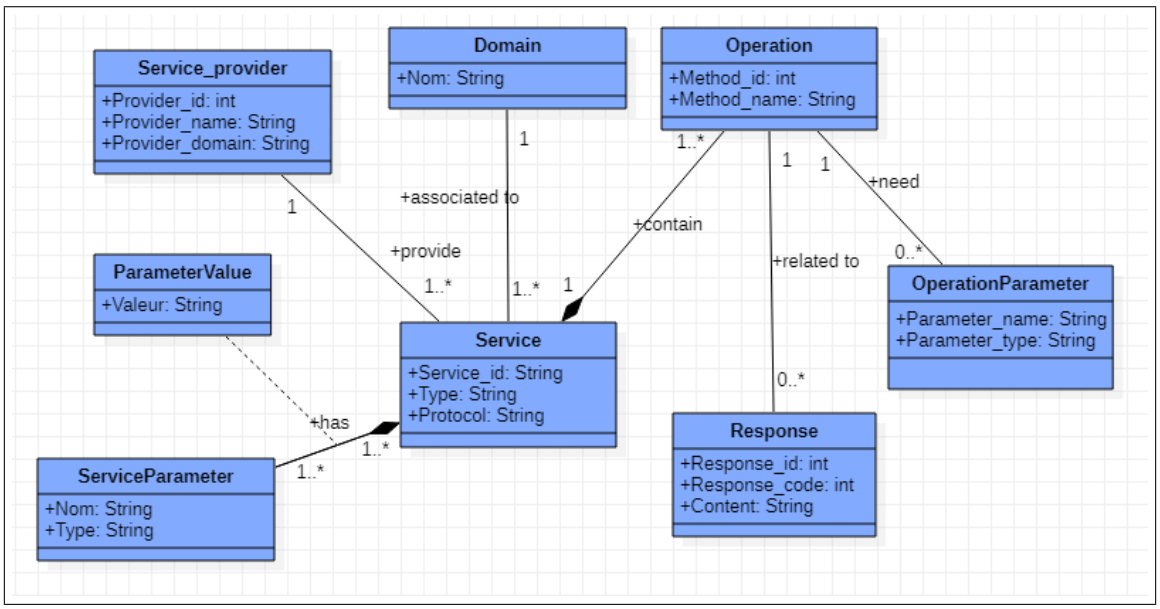

FIGURE 4.8 – Diagramme de classe de la base de services

4.10 Processus de mise à jour

Pour tenir à jour la base de service : L'agent extracteur se rend sur les zones crawlées de manière périodique. *Par exemple chaque 12h.* Également, si un service à été supprimé par le fournisseur, celui ci prévient l'agent de stockage par un message qui à son tour va supprimer le service de la base de services comme vue dans la figure 4.4.

4.11 Conclusion

Tout au long de ce chapitre nous avons eu à faire ressortir l'approche proposé pour l'exploration des services cloud, qui est en fait un système d'agents faisant partie d'une architecture de système multi-agents dédié à la composition des services cloud. Dans le chapitre prochain il sera question de voir les notions techniques, les outils utilisés et l'implémentation de notre crawler système.

Chapitre 5

Implémentation du Crawler Agent

5.1 Introduction

Dans l'optique de tester et de valider notre système de crawler proposé dans le chapitre précèdent, pour son implémentation nous avons eu a utiliser plusieurs outilsde développement. Nous présentons dans ce chapitre les outils et les technologies que nous avons utilisés pour la conception et l'exécution du système ainsi que détailler l'aspect implémentation.

5.2 Outils de développement

Comme matériel utilisé : un Ordinateur avec le Système d'exploitation Linux Mint, possédant un processeur Intel Core I3. En ce qui concerne le coté logiciel, nous avons utilisé l'environnement de développement Intellij avec le langage Java, avec lequel avons eu recours à certains outils et bibliothèques externes en Java. En ce qui concerne les systèmes multi-agent plusieurs outils existe de nos jours pour leur mises en place regroupé en deux groupes : les plateformes multi-agent de simulation ainsi que les plateformes multi-agent de gestion et de contrôle. Pour notre réalisation nous avons eu recours à cette deuxième catégorie d'outil, en optant pour la plateforme JADE.

5.2.1 Environnement Intellij , le langage Java et les bibliothèques tiers

1. **Intellij :** Intellij IDEA « IntelliJ » est un environnement de développement intégré (IDE) pour Java. Il a été développé par l'entreprise JetBrains [1] et est doté d'une assistance intelligente au codage alliée au design ergonomique, ce qui rendent le développement aussi productif qu'agréable. Il est également doté d'outils cruciaux, tels que les systèmes de contrôle de version intégré qui sont indispensables dans

1. JetBrains est une entreprise informatique éditant des logiciels pour développeurs de logiciels

un travail en équipe. Java, Kotlin, Groovy et Scala sont des langages supporté par l'IDE.

2. **Le langage de programmation Java :** Java est un langage de programmation orienté objet, développé par Sun Microsystems [2]. Il permet de créer des logiciels compatibles avec plusieurs environnements (Windows, Linux, etc.). Il est exploité pour développer et fournir des applications distribuées, mobiles et imbriquées.
3. **Crawler4j :** Crawler4j [3] est un Web crawler open source implémenté en Java qui fournit une interface simple pour explorer le Web. Nous avons eu à étendre crawler4j en l'adaptant dans le but d'extraire automatiquement la description de certains services cloud.

5.2.2 La plateforme JADE

JADE (JAVA Agent DEvelopment) est une plateforme de programmation libre distribué par TILab (Telecom Italia Lab) en Open Source avec une licence LGPL développé en Java. JADE a pour but de simplifier le développement des systemes multi-agents tout en fournissant un ensemble complet de services et d'agents conformes aux spécifications FIPA [4] qui représentent un ensemble de normes destinées à promouvoir l'interopérabilité d'agents hétérogènes et des services qu'ils peuvent représenter.Le framework est doté de nombreuses fonctionnalités qui facilitent le développement d'un système distribué.

JADE contient :

1. **Un runtime Environnement :** qui est l'environnement ou les agents peuvent vivre. Dans ce environnement les agents vivent dans ce qu'on appelle des conteneurs (*Containers)* et une plate-forme est un ensemble de conteneurs. Mais il existe un conteneur principal appelé *main containers*, qui est le premier conteneur à être lancé et tous les autres conteneurs doivent se joindre à ce conteneur principal en s'enregistrant auprès de lui. Lorsque le conteneur principal est lancé, deux agents spéciaux sont automatiquement instanciés et démarrés par JADE, dont les rôles sont définis par le standard de gestion des agents FIPA. L'Agent Management System (AMS) qui est l'agent qui supervise l'ensemble de la plateforme, et le Directory Facilitator (DF) qui est l'agent qui met en œuvre le service des pages jaunes, utilisé par tout agent souhaitant enregistrer leurs services ou rechercher d'autres services disponibles.
2. **Une librairie de classes ainsi qu'une documentation :** qui sont utilisés par les développeur pour coder leurs agents.

2. Racheté par Oracle Corporation en 2009, Sun Microsystems était un constructeur d'ordinateurs et un éditeur de logiciels américain.
3. https ://github.com/yasserg/crawler4j
4. organisme intégré en 2005 à l'IEEE (Institute of Electrical and Electronics Engineers)

3. **Une suite d'outils graphiques :** qui facilitent la gestion et la supervision de la plateforme des agents.

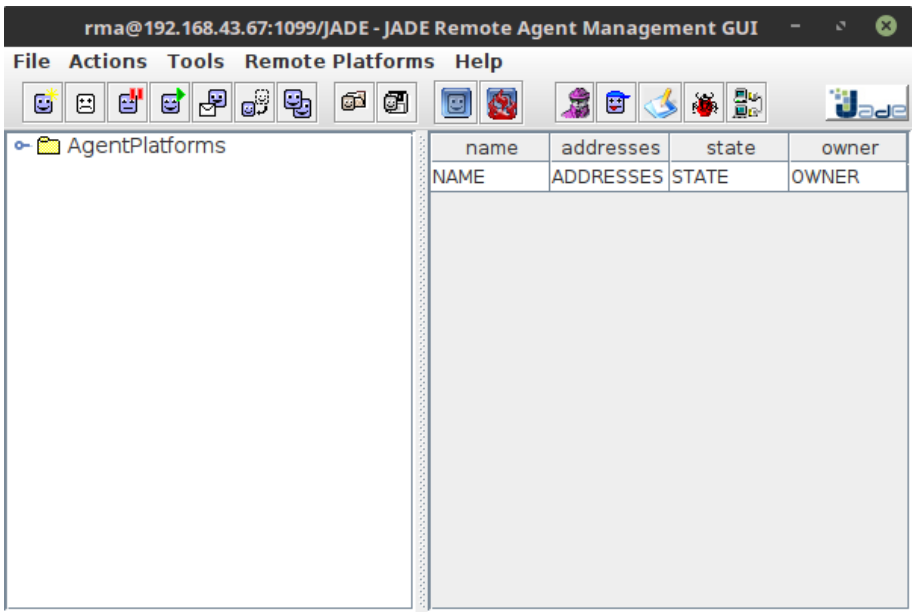

FIGURE 5.1 – GUI de JADE RMA (Remote Monitoring Agent)

En JADE un agent hérite de la classe *Agent* et est doté d'un comportement (*Behavior)*, qui est en quelques sortes les actions que celui-ci doit mener dans son environnement. Pour un comportement (classe héritant de la classe *Behavior* ou de ses sous-classe) nous avons :

1. **La méthode *action()*** qui définit les actions à exécuter par l'agent.
2. **La méthode *done()*** qui spécifie si l'exécution du comportement est fini et s'il doit être retiré de la file des comportements de l'agent en retournant un booléen.

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

Les agents JADE utilisent des messages conformes aux spécifications de la FIPA (FIPA ACL). 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, ...

Un agent JADE peut ainsi envoyer, recevoir, attendre un message d'un autre agent ou encore choisir un message dans sa boite aux lettres.

5.2.3 Git et Github

Git est un logiciel de gestion de versions initialement sorti en 2005. Le logiciel a été créé par Linus Torvald, auteur du Kernel Linux. La gestion de versions permet aux développeurs de suivre les différentes versions du logiciel en cours de développement. Différentes versions (ou branches) du logiciel peuvent être développées simultanément [46]. Git est open source, et parmis les autres logiciels de gestion de versions, est l'un des plus utilisé par les développeurs. L'intérêt d'utiliser un logiciel de versions comme Git est que pour toutes les modifications et contributions apportées à une solution logicielle, il est plus facile de maintenir un journal simultané des fonctionnalités ajoutées, ainsi permet de garder une trace des modifications et des contributions de chaque développeur. l'un des principaux avantages du contrôle de version est la possibilité de développer de manière indépendante. Une équipe de développeurs n'a donc pas besoin de travailler ensemble au même moment au même endroit. Dans un travail en équipe sur un code, une fois qu'un des développeurs a terminé sa tâche, il peut procéder par intégrer sa contribution à la base de code principale pour que d'autres développeurs puissent l'acquérir et travailler à partir sur ça.

GitHub [5] est un outil pour héberger du code open source collaboratif construit sur le système de contrôle de version Git , c'est un référentiel Web pour les projets logiciels et serait la plus grande communauté open source au monde, hébergeant à la fois du code et la documentation de ce code. On peut considérer GitHub comme étant un référentiel central et Git un outil qui vous permet de créer un référentiel local.

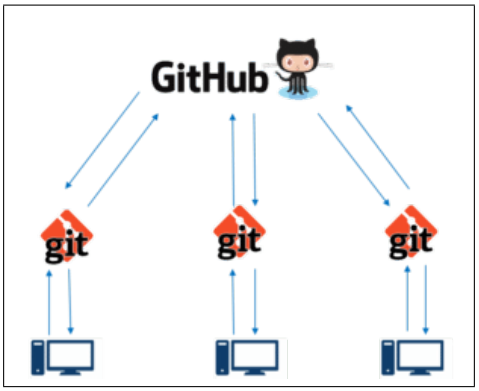

FIGURE 5.2 – Git et GitHub

5. https ://github.com/

Nous avons créé un dépôt Github dans lequel nous avons eu à travailler sur le code en utilisant le logiciel de gestion de versions décentralisé Git intégré dans l'environnement Intellij.

5.2.4 Système de gestion de base de données MySQL

MySQL est un système de gestion de base de données SQL open source sous licence GNU GPL [6] populaire qui est développé, distribué et prit en charge par Oracle Corporation. MySQL gère une collection structurée de données. Une base de données MySQL vous aide à ajouter, accéder et traiter les données stockées dans la base de données [47]. MySQL stocke les données dans des tables séparées. Les structures de la base de données sont organisées en fichiers physiques optimisés pour la vitesse. Le modèle logique, avec des objets tels que des bases de données, des tables, des vues, des lignes et des colonnes, offre un environnement de programmation flexible. La partie SQL de «MySQL» signifie «Structured Query Language», qui est le langage normalisé le plus couramment utilisé pour accéder aux bases de données.

5.3 Réalisation

Pour la réalisation de notre système, nous avons dû paramétrer le crawler en ne se focaliser que sur les services proposées par google par le biai de API Discovery Service pour l'extraction en explorant le cloud de Google.

L'API Discovery [7] fournit en effet une liste des APIs Google et des métadonnées c'est à dire les "Documents de découverte" qui sont lisibles par un agent logiciel, pour chaque API. L'API vous permet d'effectuer des tâches telles que : lister toutes les API Google, ainsi que toutes les méthodes pour une API particulière (y compris les paramètres, les chemins d'URL, etc.). C'est un outils essentiellement destiné aux développeurs pour interagir avec les API cloud de Google.

Nous avons aussi eu à crawler un dataset (OWLS-SLR [8]) de services décrit en OWL-S pour récupérer la description des services.

5.3.1 Capture d'écran de l'interface graphique de JADE avec les agents en cours d'exécution

Dans la figure 5.3 nous pouvons voir via l'interface graphique de JADE l'ensemble des agents (*AgentExtracteur, AgentFiltreur, AgentInteraction, AgentStockage*) en cours

6. GNU GPL est une licence qui fixe les conditions légales de distribution d'un logiciel libre du projet GNU
7. https ://developers.google.com/discovery
8. http ://lpis.csd.auth.gr/systems/OWLS-SLR/datasets.html

d'exécution dans le conteneur principal.

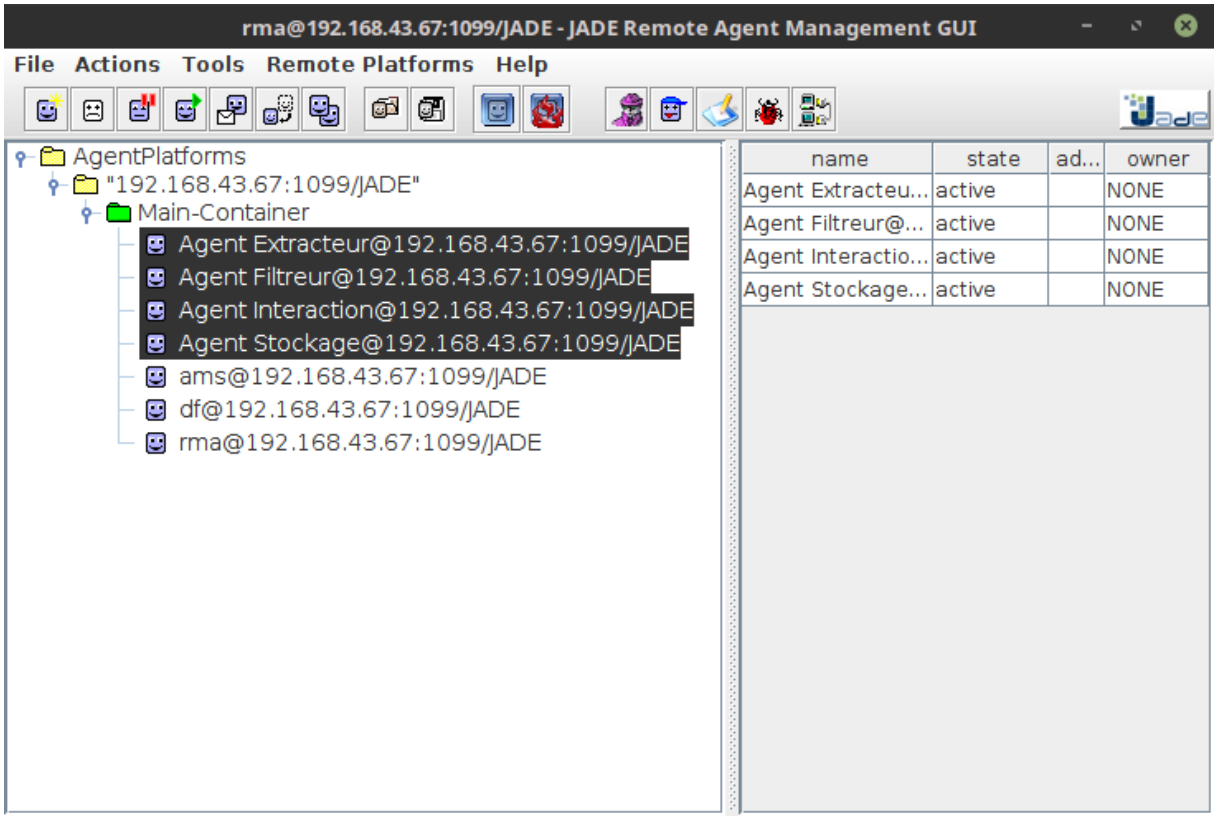

FIGURE 5.3 – Les différents agents du système en cours d'exécution

5.4 Quelques bout de code de notre implémentation

Listing 5.1 – Code java pour le lancement des agents

```
// Main.java
import jade*;

public class Main {
    public static void main(String[] args){
        Runtime runtime = Runtime.instance();
        Profile profile = new ProfileImpl();
        profile.setParameter(Profile.MAIN_HOST,"localhost");
        profile.setParameter(Profile.GUI,"true");
        ContainerController containerController =
            runtime.createMainContainer(profile);
        AgentController agentControllerAE;
        AgentController agentControllerAF;
        AgentController agentControllerAI;
```

```
        AgentController agentControllerAS;
        try {
            agentControllerAE = containerController.createNewAgent("Agent
                Extracteur","Agents.AgentExtracteur",null);
            agentControllerAF = containerController.createNewAgent("Agent
                Filtreur","Agents.AgentFiltreur",null);
            agentControllerAI = containerController.createNewAgent("Agent
                Interaction","Agents.AgentInteraction",null);
            agentControllerAS = containerController.createNewAgent("Agent
                Stockage","Agents.AgentStockage",null);
            agentControllerAE.start();
            agentControllerAF.start();
            agentControllerAI.start();
            agentControllerAS.start();
        } catch (StaleProxyException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
    }
}
```

Listing 5.2 – Code java pour Configurer le CrawlerControler

```
//CrawlerControler
public class CrawlerControler {
    CrawlController controller = null;
    public CrawlerControler(String seed) {
        CrawlConfig configuration = new CrawlConfig();
        File crawlStorage = new File("src/resources/frontiers");

        configuration.setCrawlStorageFolder(crawlStorage.getAbsolutePath());
        configuration.setPolitenessDelay(300);
        configuration.setMaxDepthOfCrawling(2);
        configuration.setMaxPagesToFetch(1);
        configuration.setResumableCrawling(false);
        PageFetcher pageFetcher = new PageFetcher(configuration);
        RobotstxtConfig robotstxtConfig = new RobotstxtConfig();
        RobotstxtServer robotstxtServer = new RobotstxtServer(robotstxtConfig,
            pageFetcher);
        try {
            controller = new CrawlController(configuration, pageFetcher,
                robotstxtServer);
```

```
        } catch (Exception e) {
            e.printStackTrace();
        }
        controller.addSeed(seed);
        configuration.setIncludeBinaryContentInCrawling(true);
    }
```

a classe Controler a un paramètre qui spécifie les valeurs de départ de l'analyse, le dossier dans lequel les données d'analyse intermédiaires doivent être stockées.

Pour ne pas imposer une charge énorme sur les serveurs et que notre crawler soit bloqué nous avons indiqué un temps d'attente de 300 millisecondes entre les requêtes avec **configuration.setPolitenessDelay(300) ;**

Nous lui passons une zone à crawler avec controler.addSeed(seed) ; Le parametre seed sera affecté dans lors de léexécution.

Listing 5.3 – Partie Code java du comportement de l'Agent d'extraction

```
//ExtractionBehavior

public class ExtractionBehavior extends TickerBehaviour {
    public ExtractionBehavior(Agent a, long period) {
        super(a, period);
    }
    @Override
    protected void onTick() {
        System.out.println("Start Extraction ... ");
        // Number of threads to use during crawling.
        int numberOfCrawlers = 1;
        CrawlerControler cc = new
            CrawlerControler("https://www.googleapis.com/discovery/v1/apis");
        CrawlController controller = cc.getControler();

        CrawlController.WebCrawlerFactory<Extraction> factory = () -> new
            Extraction();
        controller.start(factory, numberOfCrawlers);

    }

}
```

Le comportement de ce agent est un comportement TickerBehavior, qui permet d'exécuter une action de manière périodique (ici avec le paramètre period).

5.4.1 Capture d'écran des services crawlés

En faisant une extension de crawler4j [9] qui est un crawler web open source pour Java qui fournit une interface simple pour explorer le Web. et en utilisant le service Discovery de google, nous avons pu crawlé la liste de 300 Restful API de google déployé sur le cloud.

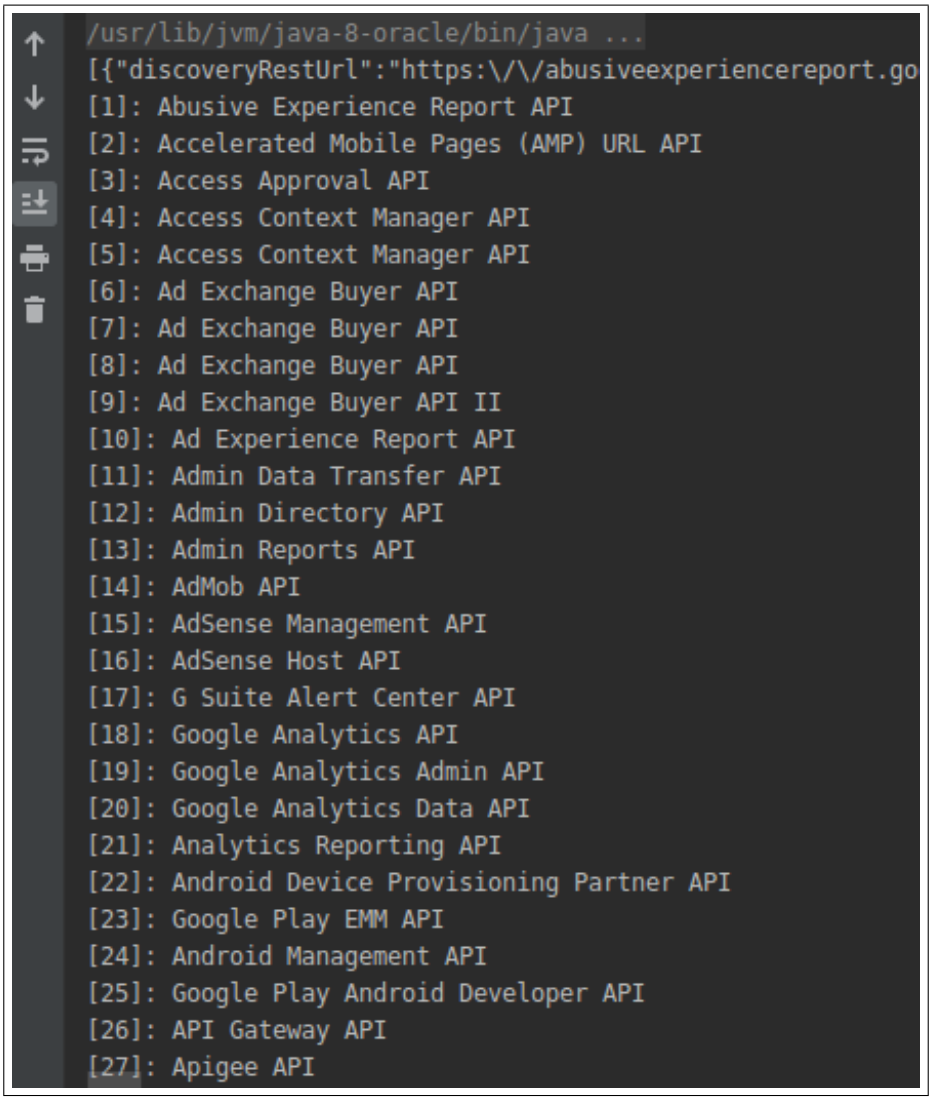

FIGURE 5.4 – Liste des services Google crawlé 1

9. https ://github.com/yasserg/crawler4j

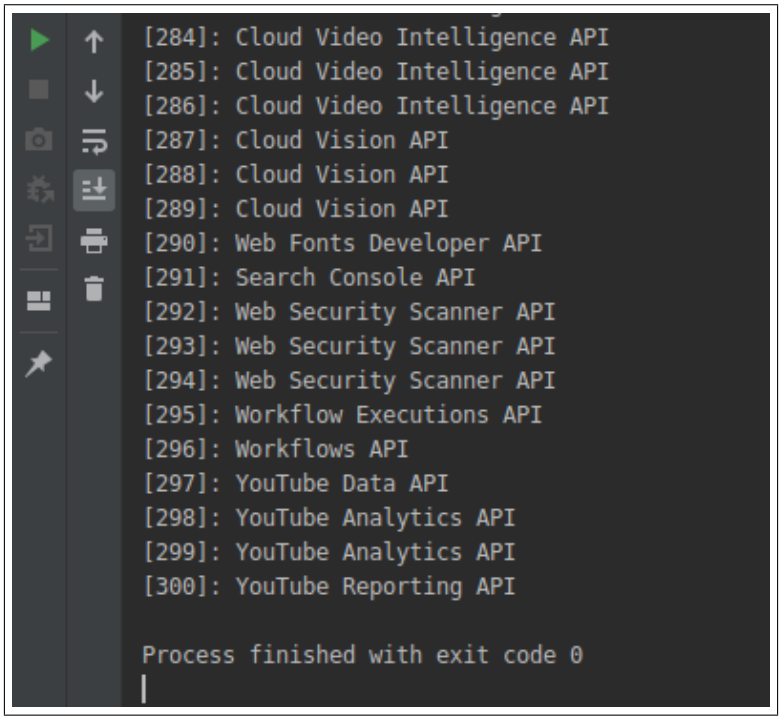

FIGURE 5.5 – Liste des services Google crawlé 2

5.4.2 Quelques données de description de services crawlé

Nous présentons ici quelques captures d'écrans du fichier de description du service Books API. L'API Google Books permet aux clients d'accéder au répertoire de Google Books (Google livre) qui est un outil de recherche intra-texte, et de consultation de livres en ligne, de constitution de collections personnelles, et de téléchargement d'ouvrages libres de droits.

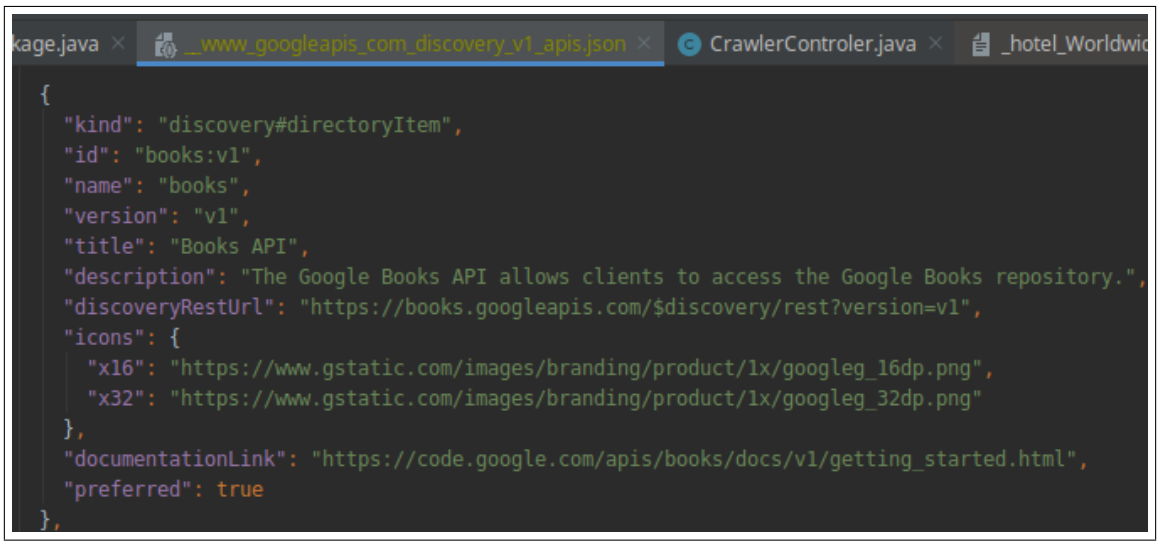

FIGURE 5.6 – Description Service Books API 1

Dans ce fichier de description, on peut voir des données comme *version*, qui est la version du service *description*, qui donne une petite description textuel du service, *discoveryRestUrl* qui est le lien pour les données liés aux opérations, les paramètres du service, on peut voir une portion du contenu dans la figure 5.7. Avec *ownerName* qui est le nom du fournisseur (dans notre cas Google) et le parametre *auth* qui defini la méthode d'authentification utilisé (ici la méthode oauth2, qui est en fait un protocole libre qui permet d'autoriser un site web, un logiciel ou une à utiliser l'API sécurisée d'un autre site web pour le compte d'un utilisateur).

Dans la figure 5.8 nous pouvons voir par exemple l'opération *list* avec ses paramètres dans la liste des méthodes *(methods)*.

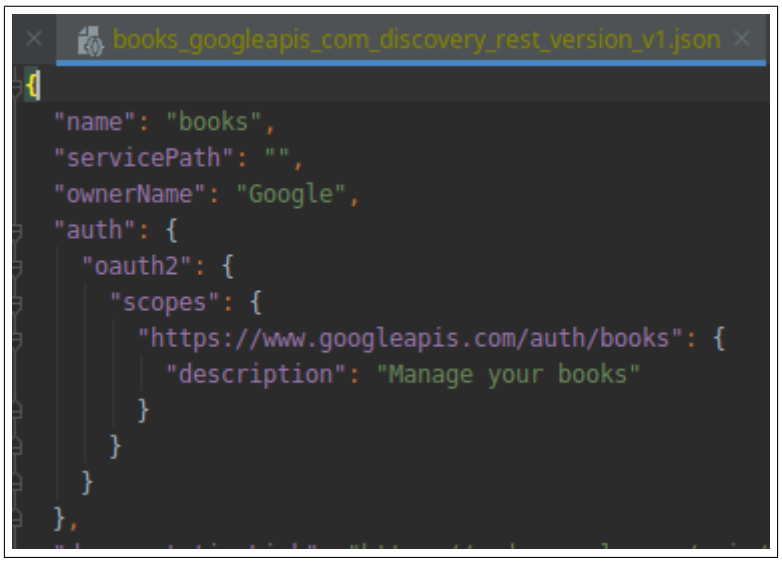

FIGURE 5.7 – Description Service Books API 2

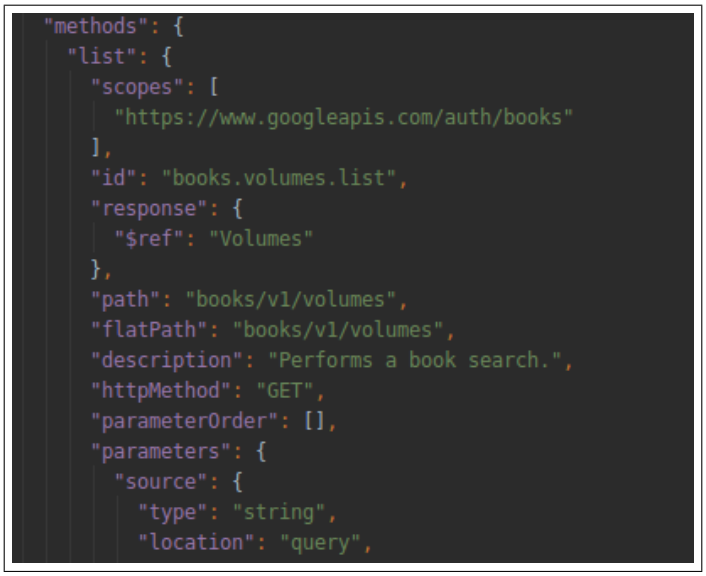

FIGURE 5.8 – Description Service Books API 3

Listing 5.4 – Fichier de description OWL du service ActivityCityService crawlé

```
<?xml version="1.0" encoding="WINDOWS-1252"?>
......
<service:Service rdf:ID="ACTIVITY_CITY_SERVICE">
<service:presents rdf:resource="#ACTIVITY_CITY_PROFILE"/>
<service:describedBy rdf:resource="#ACTIVITY_CITY_PROCESS_MODEL"/>
<service:supports rdf:resource="#ACTIVITY_CITY_GROUNDING"/>
</service:Service>
<profile:Profile rdf:ID="ACTIVITY_CITY_PROFILE">
<service:isPresentedBy rdf:resource="#ACTIVITY_CITY_SERVICE"/>
<profile:serviceName xml:lang="en">
Activity City Service
</profile:serviceName>
<profile:textDescription xml:lang="en">
This service returns city for a given activity.
</profile:textDescription>
<profile:hasInput rdf:resource="#_ACTIVITY"/>
<profile:hasOutput rdf:resource="#_CITY"/>
.....
<process:Input rdf:ID="_ACTIVITY">
<process:parameterType
   rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI">http://127.0.0.1/ontology/travel.ow
<rdfs:label></rdfs:label>
</process:Input>
<process:Output rdf:ID="_CITY">
<process:parameterType
   rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI">http://127.0.0.1/ontology/travel.ow
<rdfs:label></rdfs:label>
</process:Output>
<grounding:WsdlGrounding rdf:ID="ACTIVITY_CITY_GROUNDING">
<service:supportedBy rdf:resource="#ACTIVITY_CITY_SERVICE"/>
</grounding:WsdlGrounding>
....
```

5.4.3 Vue de quelques échanges de messages entre agents

La plateforme JADE dispose d'un sniffer, qui est une application Java créée pour suivre les messages échangés. Le sniffer est complètement intégré dans l'environnement Jade et est particulièrement utile lors du débogage des comportements des agents. Dans la figure 5.9 on peut voir quelques types de messages échangés (Request, Inform). Par

exemple l'Extracteur après avoir extrait un service envois un message de request adressé au Filtreur, contenant l'id du service.

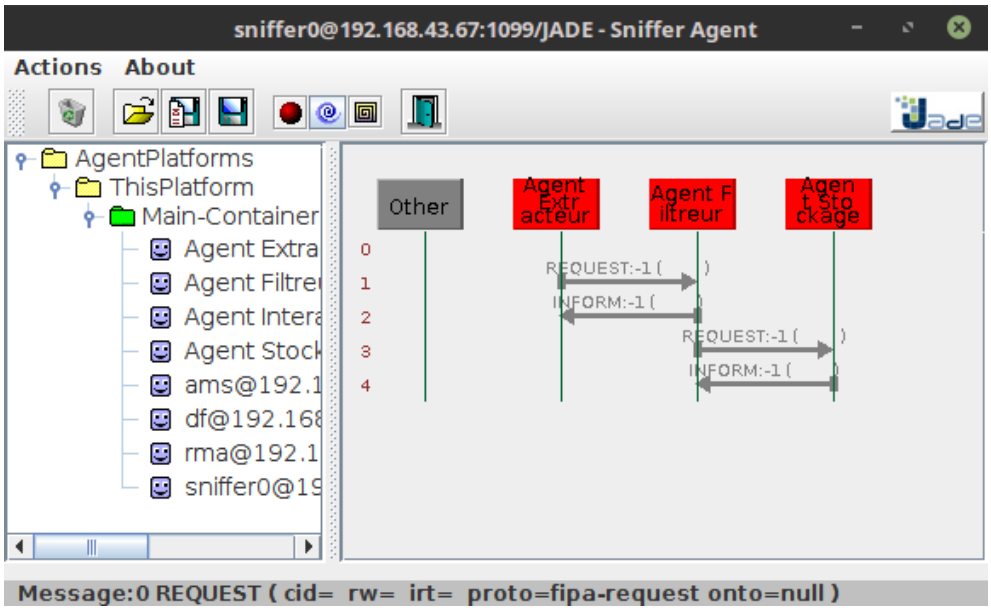

FIGURE 5.9 – quelques messages échangés entre agents

5.4.4 Schéma Relationnel de la base de donnée

La figure 5.10 présente la structure des tables utilisées dans la base de données et les relations entre elles. Le modèle relationnel représente la base de données comme un ensemble de relations, avec des attributs Le passage du diagramme de classe au Relationnel a été faite en appliquant la règle de transformation suivante :

— **Transformation des classes :** Chaque classe du diagramme UML devient une relation. La clé primaire de cette relation est l'une des attributs de la classe. Les attributs de la relation sont les mêmes attributs que la classe.

— **Transformation des associations :** Pour chaque association binaire de type *1 :N*, on ajoute à la relation côté N une clé étrangère vers la relation côté 1. Pour chaque association binaire de type *M :N*, on crée une nouvelle relation, composée de clés étrangères vers chaque relation associée, et dont la clé primaire est la concaténation de ces clés étrangères. Pour une association *1 :1* il faut la traiter comme une association *1 :N*, puis ajouter une contrainte UNIQUE sur la clé étrangère pour limiter la cardinalité maximale à 1.

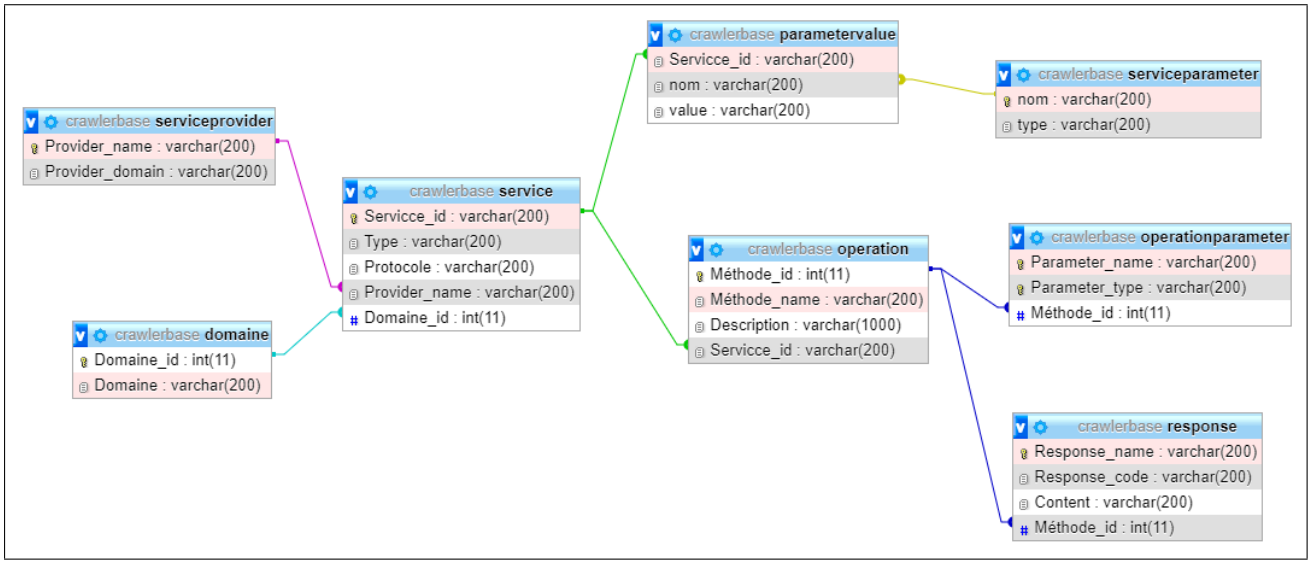

FIGURE 5.10 – Schéma relationnel de la base de donnée

5.5 Comparaison de notre approche avec d'autres approches

Nous comparons, dans le tableau suivant , notre approche avec quelques approches sur lesquelles nous avons penchées notre étude.

Référence	Principe et Techniques utilisées	Avantages	Inconvénient
[9],[15],[17],[31]	Mise en place d'une architecture dans laquelle les fournisseurs seront obligés de respecter un standard de description de service spécifié par le système (WSDL, OWL, etc.) , et un ensemble d'informations données sur les services et les fournisseurs.	Meilleurs résultats, car le système dispose d'informations homogènes sur les services facilitant la composition.	Manque d'utilisation des standards de description des services cloud par les fournisseurs, d'où la non efficacité de cette approche dans un environnement réel (c'est à dire Internet) Fournisseurs contraints à fournir des entrées définies par le système
[18],[32]	Se base sur les descriptions des services cloud contenues dans les pages web, les fichiers HTML des pages du services mis en place par les fournisseurs	Concentré sur la découverte de services cloud dans un environnement ouvert (par exemple Internet), et ne requiert pas un standard de description de service	Non compris directement par un agent logiciel, nécessitant une charge conséquente en traitement de langage naturel. Manque de données sur les services recueillis, rendant la composition de service non efficace.
[48]	Utilisation des tables de hachage distribuées pour une meilleure découverte et un meilleur équilibrage de la charge des services cloud.	Cette approche proposée est validée sur une plateforme publique de cloud computing (Amazon EC2)	Travail concentré sur un seul environnement fermé. (Amazon EC2)
Notre approche	Utiliser la technique du crawling pour récupérer les fichiers de descriptions comme présents dans le cloud (internet), dans les dépôts des fournisseurs, compréhensive par un agent logiciel.	Prise en compte de plusieurs standards de descriptions de services utilisés sur Internet (WSDL, OWL, OpenAPI, WADL) et compréhensible par un agent logiciel	Diversité de standards, d'où non homogénéité des données recueillis sur les services.

La particularité de notre approche est que la plupart des travaux pour le cloud crawling se basent sur les informations qui sont contenues dans les pages web, les fichiers HTML des fournisseurs comme dans [18]. Ces informations sont généralement faites pour être compris par un agent humain et non un agent logiciel.

Également vue le manque d'utilisation de standards pour décrire les services, plusieurs travaux ont orienté leurs recherches sur la mise en place d'une architecture systèmes dans lesquels les fournisseurs seront obligés de respecter un standard donné spécifié par le système, et un ensemble d'informations données sur les services et les fournisseurs, comme dans [9]. Notre travail était d'utiliser des fichiers de descriptions comme présents dans le cloud, dans les dépôts des fournisseurs, avec des fichiers qui peuvent être lus par une machine.

5.6 Conclusion

Dans ce dernier chapitre nous avons vu les outils utilisés pour l'implémentation du système crawler agent, qui a consisté à développer un système multiagent répondant à l'approche proposée. Nous avons prèsenté des extraits du code sources et un déroulement de l'exécution du système, également nous avons fait ressortir la particularité de notre approche avec certaines autres approches disponibles dans la littérature.

Conclusion générale

Les travaux de recherche que nous avons menée ont permis d'obtenir effectivement des résultats sur la collecte et le stockage des services cloud.

En effet l'un des points le plus important, pour bénéficier de la diversité et la qualité des services disponibles sur le cloud computing est la découverte, ainsi que la composition d'un service répondant de manière adéquate au besoin du client.

Le but de notre travail était la proposition d'un système en utilisant le paradigme d'agents et la technique de crawling afin de parcourir le cloud, à la collecte des descriptions des services, déployé sous forme de Web service, Web API, API REST. Plusieurs services existent mais de manière non centralisé, de manières "dispersés" sur le cloud.

L'objectif de notre approche était de crawler internet à l'aide de notre système de crawler agent, et de mettre à disposition une base de données contenant les informations sur les services, les caractéristiques fonctionnelles et non fonctionnelles, nécessaire pour une opération de composition.

Pour mettre en évidence notre approche proposée, nous avons utilisé la plateforme JADE, pour développer nos agents. Notre crawler à put crawler au moins 300 descriptions de services API REST, et une centaine de services Web dont les informations ont été extraits pour alimenter la base de données qui sera par la suite utilisée par des agents dédiés à la composition de service.

Les difficultés majeures rencontrées lors de ce travail, étaient entre autres le manque d'utilisation de langage de description normalisé par les fournisseurs de services cloud, le manque de certaines informations non diffusées dans la description d'un service par le fournisseur, et également le fait que certains serveurs bloquent le crawler, ce qui ralentit un peu le processus car il faut diminuer pour cela le nombre de requête envoyé aux serveurs. Comme perspective immédiate il serait question d'élargir le processus sur plusieurs fournisseurs cloud, ainsi que la prise en compte de plusieurs types de fichiers de description.

Bibliographie

[1] IONOS. Container-as-a-service : comparaison des offres caas, Mis en ligne le 24/06/19, consulté le 20/04/2020. https ://www.ionos.fr/digitalguide/serveur/know-how/caas-comparaison-des-offres-de-container-as-a-service/.

[2] Vic (J.R.) Winkler. *Securing the cloud : Cloud Computer Security Techniques and Tactics*. Elsevier, 2011.

[3] Michael Hogan and Annie Sokol. Nist cloud computing standards roadmap. *National Institute of Standards and Technology Special Publication 500-291*, page 113, 2013.

[4] Khanh Toan Tran. *Efficient complex service deployment in cloud infrastructure*. PhD thesis, Université d'Evry-Val d'Essonne. Laboratoire IBISC, 2013.

[5] Furht Borko and Escalante Armando. *Handbook of cloud computing*. Springer, 2010.

[6] Buyya Rajkumar, Broberg James, and Andrzej Goscinski. *Cloud Computing : Principles and Paradigms*. John Wiley and Sons, Inc, 2011.

[7] Ville Alkkiomäki. The role of service-oriented architecture as a part of the business model. *International Journal of Business Information Systems*, 21 :368–387, 02 2016.

[8] AIT ELMRABTI Almokhtar, ABOU EL KALAM Anas, and AIT OUAHMAN Abdellah. Les défis de sécurité dans le cloud computing : Problèmes et solutions de la sécurités en cloud computing. *IEEE*, pages 80–85, 2012.

[9] SAOULI Hamza. *Découverte de services web via le Cloud computing à base d'agents mobiles*. PhD thesis, Mohamed Khider, Biskra.

[10] Imrane Ashraf. An overview of service models of cloud computing. *Le beau journal*, page 5, Aug 2014.

[11] Judith Hurwitz, Robin Bloor, Marcia Kaufman, and Dr Fern Halper. *Cloud Computing for Dummies*. John Wiley and Sons, Inc, 2010.

[12] S. Marston, Z. Li, S. Bandyopadhyay, J. Zhang, and A. Ghalsasi. Cloud computing : The business perspective. *Elsevier*, pages 177–189, 2010.

[13] Douglas K. Barry and David Dick. *Web Services, Service-Oriented Architectures, and Cloud Computing*. Elsevier, 2013.

[14] Jiehan Zhou, Kumaripaba Athukorala, Ekaterina Gilman, Jukka Riekki, and Mika Ylianttila. Cloud architecture for dynamic service composition. *international Journal of Grid and High Performance Computing*, pages 1–14, 06 2012.

[15] BARKAT Abdelbasset. *Composition de service web dans le cloud computing.* PhD thesis, Université Mohamed KHIDER - BISKRA, 2018.

[16] Hassina Nacer, Kada Beghdad Bey, and Nabil Djebari. Migration from web services to cloud services. 05 2017.

[17] BEKKOUCHE Amina. *Vers une composition automatique des services web.* PhD thesis, Univeristé Abou Bekr Belkaid Tlemcen, 2018.

[18] Talal H. Noor, Quan Z. Sheng, Abdullah Alfazi, Anne H.H. Ngu, and Jeriel Law. A crawler engine for cloud services discovery on the world wide web. *IEEE 20th International Conference on Web Services CSCE*, 2013.

[19] Margaret Rouse. Restful api (rest api), publié en Septembre 2014 et consulté le 24 juillet 2020.

[20] Sun Le, Dong Hai, and Ashraf Jamshaid. Survey of service description languages and their issues in cloud computing. *2012 Eighth International Conference on Semantics, Knowledge and Grids*, pages 1–8, 2012.

[21] D. B. Claro, P. Albers, and J.-K Hao. Approaches of web services composition. *In International Conference on Enterprise Information Systems*, page 208–213, 2005.

[22] Nomane Ould Ahmed M'Barek and Samir Tata. Services web : revue des approches de description sémantique. *SIIE 2008 : Système d'Information et Intelligence Economique, Feb 2008, Hammamet Tunisie*, pages 1 – 7, 2008.

[23] FEUGA Loïc. Open api - les bases. https ://www.supinfo.com/articles/single/5965-open-api-bases, Publié le 06/10/2017.

[24] Youakim Badr, Ajith Abraham, Frédérique Biennier, and Crina Grosan. Enhancing web service selection by user preferences of non-functional features. *Fourth International Conference on Next Generation Web Services Practices (NWeSP 2008)*, pages 60–65, 2008.

[25] Lie Qu. *Credible Service Selection in Cloud Environments.* PhD thesis, Université MACQUARIE, 2016.

[26] Y. Wei and M. B. Blake. Service oriented computing and cloud computing. *IEEE*, pages 72–75, 2010.

[27] Hajar OMRANA. *Vers une composition dynamique des Services Web : une approche de composabilité offline.* PhD thesis, UNIVERSITÉ MOHAMMED V – AGDAL RABAT ECOLE MOHAMMADIA D'INGENIEURS (Centre d'Etudes Doctorales), 2014.

[28] Hamdi Gabsi, Rim Drira, Henda Hajjami, and Henda Ben Ghezala. Cloud services discovery and selection assistant. 05 2020.

[29] Jrad F., Tao J., Streit A., Knapper R., and C Flath. A utility based approach for customised cloudservice selection. *In International Journal of Compu-tational Science and Engineerin*, pages 32 – 44, 2005.

[30] W. H. Asma Musabah Alkalbani and J. Y. Kim. A centralised cloud services repository (ccsr) frame-work for optimal cloud service advertisement discovery from heterogenous web portals. *In IEEE Access*, page 128213 – 128223, 2019.

[31] E. Al-Masri and Q. Mahmoud. Investigating web services on the world wide web. *in Proc. of the 17th Int. Conf. on World Wide Web (WWW'08)*, 2008.

[32] Kwang Mong Sim. Agent-based cloud computing. *IEEE Transactions on services computing*, pages 1–14, 2011.

[33] Imed Jarras and Brahim Chaib-draa. Aperçu sur les systèmes multiagents. 01 2002.

[34] Jacques Ferber. *Les Systèmes Multi Agents : vers une intelligence collective*. 1995.

[35] John Tranier. *Vers une vision intégrale des systèmes multi-agents*. PhD thesis, UNIVERSITÉ Montpellier II, 2007.

[36] B. Moulin B. Chaib-draa, I. Jarras. Systèmes multiagents : Principes généraux et applications. 2001.

[37] Nicholas R. Jennings Michael Wooldridge. Intelligent agents : Theory and practice. 1995.

[38] BENHAMZA Karima. *Conception d'un système multi-agents adaptatif pour la résolution de problème*. PhD thesis, UNIVERSITÉ UNIVERSITE BADJI MOKHTAR-ANNABA, 2016.

[39] GHOUL Khalid. *Application des systèmes multi-agents pour des enchères Optimales au sens de Pareto*. PhD thesis, UNIVERSITÉ UNIVERSITE FERHAT ABBAS - SETIF, 2011.

[40] Mohammad Abu Kausar, Vijaypal Dhaka, and Sanjeev Singh. Web crawler : A review. *International Journal of Computer Applications*, 63 :31–36, 02 2013.

[41] Emmanuel ADAM. La programmation orientee agent - presentation et elements de modelisation. Cours UPHF/ISTV-LAMIH Universite Polytechnique des Hauts-De-France, Consulter le 11/8/2020.

[42] P. Mohamed Shakeel, S. Baskar, V. R. Sarma Dhulipala, Sukumar Mishra, and Mustafa Musa Jaber. Maintaining security and privacy in health care system using learning based deep-q-networks. *Journal of Medical Systems*, page 186, 08 2018.

[43] Sebastiani F. Machine learning in automated text categorization. *ACM Computing Surveys, vol. 34, no. 1*, pages 1–47, 2002.

[44] Elberrichi Zakaria, Rahmoun Abdelattif, and Bentaalah Mohamed Amine. Using wordnet for text categorization. *The International Arab Journal of Information Technology, Vol. 5, No. 1*, pages 16–24, 01 2008.

[45] McCarthy D., R. Koeling, J. Weeds, and J. Carroll. Finding pre-dominant senses in untagged text. *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics*, pages 280–287, 2004.

[46] F. F. Blauw. The use of git as version control in the south african software engineering classroom. In *2018 IST-Africa Week Conference (IST-Africa)*, pages Page 1 of 8 – Page 8 of 8, 2018.

[47] Binildas Christudas. *MySQL*, pages 877–884. 06 2019.

[48] R. Ranjan, L. Zhao, X. Wu, A. Liu, A. Quiroz, and M. Parashar. Cloud computing : Principles, systems and applications. pages 280–287, 2010.