MINISTERE DE L'ENSEIGNEMENT SUPERIEUR ET DE LA RECHERCHE SCIENTIFIQUE
UNIVERSITÉ DR TAHAR MOULAY DE SAIDA

MATCHMAKING DES SERVICES WEB SÉMANTIQUE

KOUIDER OMEIRI & ABDELHAK BELHACHEMI
FACULTÉ DE TECHNOLOGIE
DÉPARTEMENT DE L'INFORMATIQUE
RÉSEAUX INFORMATIQUE ET SYSTÈMES RÉPARTIS

MÉMOIRE PRÉSENTÉ EN VUE DE L'OBTENTION
DU DIPLÔME DE MASTER
JUIN 2016

ENCADRÈ PAR : M. KHATER MAAMAR

© KOUIDER OMEIRI & ABDELHAK BELHACHEMI, 2016.

UNIVERSITÉ DR TAHAR MOULAY DE SAIDA

DÉPARTEMENT DE L'INFORMATIQUE

Ce mémoire intitulé :

MATCHMAKING DES SERVICES WEB SÉMANTIQUE

présenté par : OMEIRI KOUIDER & BELHACHEMI ABDELHAK
en vue de l'obtention du diplôme de : Master

DÉDICACE

A nos parents
A nos familles
A nos frères, nos sœurs et leurs familles
A touts nos amis
A nos professeurs.

REMERCIEMENTS

Tout d'abord, nous remercions Dieu qui nous a donné la force et la vie pour accomplir cette tâche, qui semblait d'abord une tâche difficile.
Nous tenons à remercier tout d'abord, Mr. Maamar Khater, notre encadreur pour son encouragement de notre superviseur, son expérience, ses conseils et de sympathie qui nous a permis d'accomplir cette thèse.
Nos vifs remerciements vont également aux membre du jury pour l'intérêt qu'ils ont porté a notre recherche en acceptant d'examiner notre travail et de l'enrichir par leurs propositions Enfin, nous tenons également à remercier toutes les personnes qui ont participé de prés ou de loin à la réalisation de ce travail.

RÉSUMÉ

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

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

Mots clés : service Web, systèmes multi-agents, Web sémantique

ABSTRACT

Web services are emerging and promising technologies for the development, deployment and integration of Internet applications. They are based on three main bricks that are SOAP (Simple Object Access Protocol), WSDL (Web Service Description Language) and UDDI (Universal Description, Discovery and Integration). The language used behind these protocols is XML (eXtensible Markup Language), which makes Web services independent of platforms and programming languages. They have become very effective in the interoperability of systems. The need to introduce semantics in Web services is felt to automate the different phases of their life cycle, namely the discovery phase. The concept of semantic web services, is the result of convergence in the field of web services with the Semantic Web, in fact its ultimate goal is to make web services more accessible to the machine by automating tasks that facilitate their use. In this work, we study the problem of semantic discovery of services by providing a method that is based on agents.
Keywords : Web service, multi-agents system, semantic web

TABLE DES MATIÈRES

TABLE DES MATIÈRES 1

LISTE DES TABLEAUX 4

LISTE DES FIGURES 5

CHAPITRE 1 INTRODUTION GÉNÉRAL 8
1.1 Contexte 10
1.2 Problématique 11
1.3 Objectifs 11
1.4 Conclusion 14

CHAPITRE 2 LES SERVICES WEB 15
2.1 Introduction 17
2.2 Les services web 17
2.2.1 Définition 17
2.2.2 L'architecture orientée service 19
2.2.3 Architecture standard des services Web 19
2.3 Fonctionnement des services Web 20
2.3.1 Service provider service 20
2.3.2 Service requester programme client 20
2.3.3 Annuaire service registry 20
2.4 Les technologies standards 21
2.4.1 Le protocole SOAP 21
2.4.2 Le référentiel UDDI 22
2.4.3 Le langage de description WSDL 23
Histoire de WSDL 23
WSDL Eléments 23
2.5 Conclusion 24

CHAPITRE 3 LE WEB SÉMANTIQUE 26
3.1 Le Web sémantique 28
3.2 Historique du web sémantique 29
3.3 Objectifs du web sémantique 29

3.4 La pile du web sémantique . . . 30
3.4.1 URI (Uniform Resource Identifier) . . . 30
3.4.2 XML (eXtensible Markup Language) . . . 31
3.4.3 RDF (Resource Description Framework) . . . 31
Sérialisation de triplets RDF . . . 33
3.4.4 RDF Schéma (RDFS) . . . 34
3.4.5 OWL (Ontology Web Language) . . . 35
3.5 Ontologie . . . 36
3.5.1 définition D'ontologie . . . 36
3.5.2 cycle de vie . . . 37
3.5.3 types d'ontologie . . . 38
Ontologie de représentation de connaissances . . . 38
Ontologie de haut niveau / supérieure (Top-level / Upper-model) . . . 38
Ontologie Générique (Generic ontology) . . . 39
Ontologie du domaine (Domain ontology) . . . 39
Ontologie de Taches (Task ontology) . . . 39
Ontologie d'application (Application ontology) . . . 39
3.6 Composantes d'une ontologie . . . 39
3.7 Domaines d'applications des ontologies . . . 42
3.8 Conclusion . . . 43

CHAPITRE 4 LES SERVICES WEB SÉMANTIQUE . . . 44
4.1 Introduction . . . 46
4.2 Définition . . . 46
4.3 Approches proposées pour réalisation web sémantique . . . 46
4.3.1 approche basées sur langages sémantique . . . 46
L'approche OWL-S . . . 46
L'approche WSMO . . . 49
4.3.2 A base d'annotation sémantique . . . 50
L'approche USDL . . . 51
L'approche SAWSDL . . . 52
4.4 Conclusion . . . 56

CHAPITRE 5 LA DÉCOUVERTE DES SERVICES WEB . . . 58
5.1 Introduction . . . 60
5.2 Définition . . . 60
5.3 Critères de découverte . . . 60

5.3.1 Critère d'architecture . . . 60
5.3.2 Critère d'automatisation . . . 61
5.3.3 Critère de matchmaking . . . 61
5.4 Les approches de découverte . . . 62
5.4.1 Approches fonctionnelles . . . 62
Approches Syntaxiques . . . 62
Approches Sémantiques . . . 64
Approches Comportementales . . . 68
5.4.2 Approches Non Fonctionnelles . . . 68
Approches à Base de Réputation et Confiance . . . 69
5.5 Conclusion . . . 69

CHAPITRE 6 L'APPROCHE PRPOSÉE . . . 70
6.1 Introduction . . . 72
6.2 La démarche de notre approche . . . 72
6.3 L'approche SAWSDL Long path . . . 72
6.4 Les étapes de l'algorithme de matching . . . 73
6.5 L'algorithme de matchmaking . . . 73
6.6 La procédure de matching . . . 74
6.7 Le passage au graphe . . . 75
6.8 Le tri des résultats . . . 76
6.9 Discussion des Résultats . . . 77
6.9.1 Le temps de Calcule . . . 77
6.9.2 Les Tableaux des resultats . . . 79
6.10 Outils de travail . . . 80
6.10.1 Le langage java . . . 80
6.10.2 Eclipse . . . 80
6.10.3 Xampp . . . 81
6.10.4 Raisonneur Pellet (version) . . . 81
6.10.5 JavaFX . . . 82
6.11 Expérimentation . . . 83
6.12 Conclusion . . . 87

CHAPITRE 7 CONCLUSION GÉNÉRALE . . . 88
7.1 Conclusion générale et perspectives . . . 89

RÉFÉRENCES . . . 91

LISTE DES TABLEAUX

Tableau 6.1 poids des degrés de correspondance SAWSDL 72
Tableau 6.2 les outputs et les inputs du service requête 74
Tableau 6.3 les outputs et les inputs du service web 74

LISTE DES FIGURES

Figure 1.1 Matchmaking des services web 11
Figure 1.2 Partie de book ontology 12
Figure 2.1 Service Web - Modèle en couches[2] 18
Figure 2.2 Une Architecture de service Web basique[4] 19
Figure 2.3 Fonctionnement des services Web 20
Figure 2.4 La structure des messages SOAP 22
Figure 3.1 couches de web sémantique [5] 30
Figure 3.2 Représentation visuelle d'un graphe RDF [5] 32
Figure 3.3 Représentation graphique d'un ensemble de triplets 34
Figure 3.4 cycle de vie d'une ontologie 38
Figure 4.1 type d'information [8] 47
Figure 4.2 Principaux éléments de description dans USDL [14] 51
Figure 5.1 quelques approches de découverte [4] 62
Figure 6.1 le graphe de matching(Pondération1) 76
Figure 6.2 le graphe de matching(Pondération2) 76
Figure 6.3 Résultat pour book_price_service trier 77
Figure 6.4 la selection de fichier XML contient 95 service et le service requète 78
Figure 6.5 La résultat de 95 services / UserBookService / pondération 1 = 24 seconds 78
Figure 6.6 La résultat de 95 services / UserBookService / pondération 2 = 23 seconds 79
Figure 6.7 La list des services résultat de 95 services / UserBookService / pondération 1 = 10 services 79
Figure 6.8 La list des services résultat de 95 services / UserBookService / pondération 2 = 10 services 79
Figure 6.9 L'interface principale de notre application MSWS 83
Figure 6.10 L'entête d'application pour le lancement , le re-lancement et l'affichage d'etat de progression 83
Figure 6.11 Ici tu peux charger le fichier XML qui contient la list des service 84
Figure 6.12 Ici tu peux sélectionner la service de requête 84
Figure 6.13 Les option d'application , selection de pondération 85
Figure 6.14 Résultat de Matching entre les Inputs 85
Figure 6.15 Résultat de Matching entre les outputs 86
Figure 6.16 Resultat Final des service compatible avec le service requète 86
Figure 6.17 charts Bar pour la comparaison entre les Rounds 87

Figure 6.18 Message de confirmation pour quitter l'application 87

LISTE DES SIGLES ET ABRÉVIATIONS

W3C	World Wide Web Consortium
RDF	Resource Description Framework
RDFS	RDF Schema
SPARQL	SPARQL Protocol and RDF Query Language
OWL	Web Ontology Language
URI	Uniform Resource Identifier
SOA	Service Oriented Architecture
SOAP	Simple Object Access Protocol
UDDI	Universal Description, Discovery and Integration
WSDL	Web Service Description Language
IETF	Internet Engineering Task Force
OSI	Open Systems Interconnection

CHAPITRE 1

INTRODUTION GÉNÉRAL

LES SECTIONS

1.1 Contexte 10
1.2 Problématique 11
1.3 Objectifs 11
1.4 Conclusion 14

1.1 Contexte

L'intégration de l'informatique dans les entreprises a permet d'améliorer la circulation et le traitement des informations en utilisant des systèmes d'information -noté SI-.
Avec l'air du web et l'apparition de la notion de services Web, les entreprises sont convertis à intégrer les services web dans leurs SI. La technologie des services Web est en train de devenir une approche prometteuse d'intégration et d'interaction des applications intra et inter organisations. Dans ce contexte, l'efficacité et la fiabilité des SI à base de services Web deviennent un facteur critique dans la réussite des entreprises.
Les services Web sémantiques se situent à la convergence de deux domaines de recherche importants qui concernent les technologies de l'Internet : le Web sémantique et les Web services.
L'expression « web sémantique », due à Tim Berners-lee [berners-Lee et al., 2001][1] au sein du W3C, fait d'abord référence à la vision du web de demain comme un vaste espace d'échange de ressources entre êtres humains et machines permettant une exploitation, qualitativement supérieure, de grands volumes d'informations et de services variés. Le web sémantique s'intéresse principalement aux informations statiques disponibles sur le web et les moyens de les décrire de manière intelligible pour les machines. Cependant, la tâche principale des services web est d'assurer l'interopérabilité entre les applications via le web en vue de rendre le web plus dynamique. L'objectif visé par la notion de services Web sémantiques est de créer un Web sémantique de services dont les propriétés, les capacités, les interfaces et les effets sont décrits de manière non ambiguë et exploitable par des machines.
Les problèmes classiques connus dans le domaine des services web sont imposés implicitement dans la conception et la mise en œuvre d'un tel système, à savoir : la description de service Web, le processus de publication, le processus de découverte, le processus de composition, le processus de sélection, la substitution, et le processus d'adaptation.
Différentes approches ont été proposées dans la littérature pour répondre à ces besoins, ces approches diffèrent dans le langage et le modèle de description utilisé, on trouve des approches ontologiques qui s'appuient sur la description sémantique du service, et les approches comportementales qui reposent essentiellement sur la description fonctionnelle du service.[1]
Le besoin d'automatisation du processus de conception et de mise en œuvre des services web sémantiques rejoint les préoccupations à l'origine du web sémantique, à savoir comment décrire formellement les connaissances de manière à les rendre exploitables par des machines.
Parmi les solutions proposées de la problématique de la description sémantiques des services, SAWSDL se présente comme l'ontologie qui détient une large utilisation dans ce domaine. Notre projet de fin d'etude s'intéresse aux services web sémantiques à base SAWSDL, et vise à proposer une solution pour améliorer les performances du processus de matchmaking des services atomiques.

1.2 Problématique

Le mécanisme de matching fourni par UDDI et WSDL ne peut servir l'objectif de la découverte dynamique des services Web, le principe de ce mécanisme est basé sur la forme syntaxique de services Web et peut produire par conséquent de nombreux faux positifs et faux négatifs, ce qui réduit énormément les performances du système de découverte.
Pour surmonter ces limitations, le concept de la sémantique doit être introduit dans le processus de matching par l'utilisation d'ontologies.

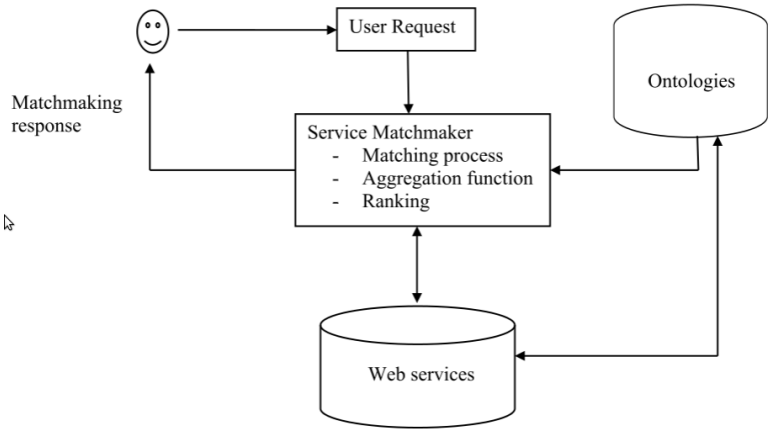


Figure 1.1 Matchmaking des services web

La découverte de services sémantiques est un domaine de recherche très actif, et elle est abordée dans un grand nombre de travaux de recherches. Nous considérons que le matchmaking est le processus de trouver des services web pertinents en se basant sur l'évaluation et l'agrégation des résultats des opérations de matching. Les résultats retournés doivent être rangés et classés par ordre de mérite.

1.3 Objectifs

Étant donné un besoin d'un client, qui peut être présenté sous la forme d'un ensemble de concepts d'entrées, de concepts de sorties (black-box matching), et éventuellement des descriptions sur le fonctionnement du service (glass-box matching). Le matchmaker doit créer des mécanismes qui comparent ces besoins avec l'ensemble des services publiés, et de retourner des résultats performants en termes de rappel, de précision et de temps d'exécution. Deux sous-problématiques ressortent de cette formalisation de la problématique de la découverte :

Découverte des services web sémantiques atomiques :

Le matching est basé seulement sur les propriétés (inputs/outputs) entre la requête et les services publiés. L'approche de référence pour cette catégorie de matching est celle proposée par [Paolucci et al., 2002]. L'expérimentation de cette approche a montré que les résultats du matchmaking sont moins performants dans certaines situations (l'ordre de concepts lors de l'opération de matching, retrait (ou non) du service après son matching). **Le premier scenario** : sans suppression des concepts du service publié après le matching.

Advertise 'A'

Input	publisher
Output	novel, price

Query 'Q'

Input	publisher
Output	romantic novel, science-fiction novel

liste-candidats=novel, price

dom(romantic novel, novel)=exact=1.

dom(romantic novel, price)=fail=0.

dom(science-fiction novel, novel)=exact=1.

dom(science-fiction novel, price)=fail=0.

Gdom=exact.

Le matchmaker retourne 'A' comme réponse correcte à la requête 'Q' avec un degré de correspondance « exacte », tandis que 'A' représente un faux positif en réalité (en général, c'est le cas où on trouve deux concepts ou plus de la requête qui sont en correspondance avec un seul concept du service 'A').

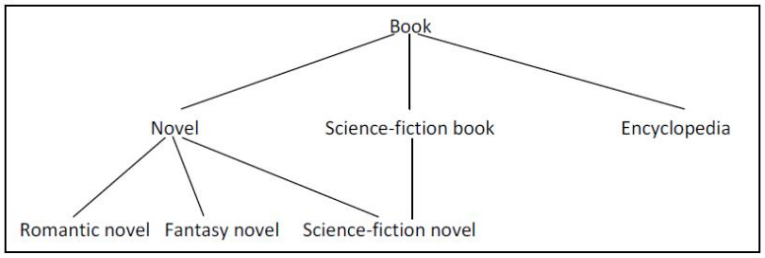


Figure 1.2 Partie de book ontology

Le deuxième scenario : avec suppression des concepts du service publié après le matching.

Advertise 'A'

Input	publisher
Output	novel, science-fiction book

Query 'Q'

Input	publisher
Output	science-fiction novel, romantic novel

list-candidat= novel, science-fiction book
dom(science-fiction novel, novel)=exact=1.
dom(science-fiction novel, science-fiction book)=1= exact.
Le concept 'Science-fiction novel' correspond au concept 'novel', et 'novel' est retiré de la liste des concepts candidats.
list-candidat= science-fiction book
dom(romantic novel, science-fiction book)=fail=0.
Le matchmaker ne retient pas le service 'A' comme bonne réponse à la requête 'Q' car il a trouvé un échec dans le processus de matching.
On remarque que 'A' représente un faux négatif (l'ordre des concepts de la requête influence sur le degré de correspondance et peut changer par conséquence le Gdom).
Nous intéressons à améliorer le processus de matchmaking proposé par l'approche de [Paolucci et al., 2002][1].

Organisation de la manuscrit

Le manuscrit est composé de manière générale d'une introduction générale, de cinq chapitres et d'une conclusion générale.
Le manuscrite est structuré en cinq chapitre et une conclusion générale :

- Le premier chapitre est consacré a un état de l'art sur l'architecture orientée service et plus particulièrement la technologie des services web
- Le deuxième chapitre porte sur les ontologies et les technologies du web sémantique.
- Le troisième chapitre on va étudier le mécanisme de couplage entre web sémantique et les web services nous décrivons les différents langages de descriptions sémantiques dédiés aux services web.
- Le quatrième chapitre est une présentation des différentes approches proposées pour la découverte de services web.
- Le cinquième chapitre illustre l'application de notre approche. Nous décrivons tout d'abord les différents outils techniques utilisés. Nous précisons ensuite l'implémentation des différents composantes inclus dans notre architecture et comment l'opération de découverte peut être réalisées.
- Enfin, le manuscrit se termine par une conclusion générale qui récapitule les travaux réalisés et propose quelque vision pour les travaux futurs.

1.4 Conclusion

Le web sémantique propose une nouvelle approche pour la réalisation de solution a service en utilisant un modéle plus riche d'informations. elle est définie par un ensemble de concepts reliés par des relation spécifiques.
La terminologie définie par une ontologie est ensuite utilisée pour la représentation des ressources ou de l'information que les services du domaine manipulent. les chapitres suivants presente les services web.

CHAPITRE 2

LES SERVICES WEB

LES SECTIONS

2.1 Introduction 17
2.2 Les services web 17
2.2.1 Définition 17
2.2.2 L'architecture orientée service 19
2.2.3 Architecture standard des services Web 19
2.3 Fonctionnement des services Web 20
2.3.1 Service provider service 20
2.3.2 Service requester programme client 20
2.3.3 Annuaire service registry 20
2.4 Les technologies standards 21
2.4.1 Le protocole SOAP 21
2.4.2 Le référentiel UDDI 22
2.4.3 Le langage de description WSDL 23
Histoire de WSDL 23
WSDL Eléments 23
2.5 Conclusion 24

2.1 Introduction

De nos jours, le Web n'est plus simplement un énorme entrepôt de texte et d'images, son évolution a fait qu'il est aussi un fournisseur de services. La notion de "service Web" désigne essentiellement une application mise à disposition sur Internet par un fournisseur de services, et accessible par les clients à travers des protocoles Internet standard. Par essence, les services Web sont des composants logiciels autonomes et auto-descriptifs et constituent par ce fait un nouveau paradigme pour l'intégration d'applications. Actuellement, les services Web sont mis en œuvre au travers de trois technologies standards : WSDL, UDDI et SOAP. Ces technologies facilitent la description, la découverte et la communication entre services. Cependant, cette infrastructure de base ne permet pas encore aux services Web de tenir leur promesse d'une gestion largement automatisée. Cette automatisation est pourtant essentielle pour faire face aux exigences de passage à l'échelle et de la volonté de réduire les coûts de développement et de maintenance des services. Fondamentalement, elle doit s'accommoder d'un moyen pour décrire les services Web d'une manière compréhensible par une machine.

2.2 Les services web

2.2.1 Définition

Un Service Web est un composant logiciel autonome qui reçoit des requêtes et qui renvoie des réponses au travers une interface standard bien définie. Les services Web sont indépendants de toute plateforme d'exécution et de tout langage de programmation. Plus précisément, un service Web est un composant logiciel autonome et unique (définit par une seule instance de service) qui permet de décrire un ensemble d'opérations accessibles via le Web. Il se base sur le standard XML pour décrire les appels de fonctions distantes et les données échangées en utilisant le protocole HTTP comme moyen de communication. Ce protocole est basé sur le principe de requêtes/réponses décrites avec des messages XML.
L'émergence des services Web a permis aux applications d'être vues comme un ensemble de services métiers bien structurés et correctement décrits, plutôt qu'un ensemble d'objets et de méthodes. Cela facilite, non seulement, les échanges entre les applications d'une même organisation mais aussi, les échanges entre les applications des organisations distantes et distribuées.
La granularité de cette approche (c.-à-d. le découpage d'une application en services) a un avantage très important en termes de maintenance et d'interopérabilité d'applications. En effet, elle permet de modifier plus facilement un service ou de le remplacer par un autre, réduisant ainsi la complexité d'une application. En d'autres termes, le développeur peut se focaliser sur un service indépendamment du reste de l'application.
Le fonctionnement d'un service Web repose sur un modèle en couches dont les trois couches prin-

cipales sont représentées dans la Figure 2.1.

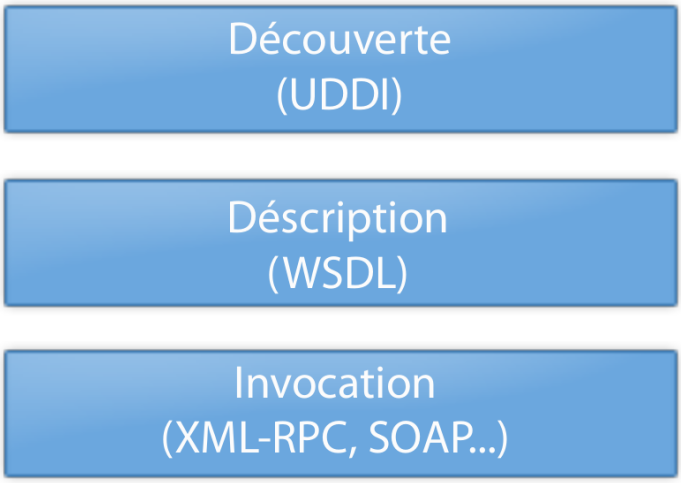


Figure 2.1 Service Web - Modèle en couches[2]

La couche découverte a pour objectif de rechercher et de localiser un service Web. Ceci est possible grâce au protocole standard de découverte UDDI [1]. La deuxième couche - Description - permet de décrire les interfaces des services Web (c.-à-d. les paramètres des fonctions, les types de données, etc.). Les services Web offrent un moyen de décrire leurs interfaces d'une manière suffisamment détaillée pour permettre à un utilisateur de créer une application cliente capable de communiquer avec eux. La description d'un service est généralement fournie par un document XML nommé WSDL [2] qui précise les méthodes pouvant être invoquées, leurs signatures et les points d'accès du service (URL, port, etc.). La dernière couche, dite couche d'invocation, décrit alors la structure des messages échangés. En effet, les services Web proposent aux utilisateurs des fonctionnalités pratiques grâce à un protocole standard basé sur XML, dans la plupart des cas, le protocole utilisé est SOAP [3] ou encore XML-RPC [4].
Les services web ont une architecture basée sur trois composants principaux qui répondent chacun à une question :

- **Echange** : comment échanger les messages entre les services web ?
- **Découverte** : comment identifier et localiser les services web ?
- **Description** : comment exposer les fonctions des services web ?

1. UDDI : Universal Description Discovery and Integration
2. WSDL : Web Service Description Language
3. SOAP : Simple Object Access Protocol
4. XML-RPC : XML Remote Procedure Call

Cette architecture est appelée une architecture orientée service (Service Oriented Architecture en Anglais - SOA).[3]

2.2.2 L'architecture orientée service

Service Oriented Architecture (SOA) est une architecture qui a vu le jour au début de l'année 2000, pour répondre aux besoins des entreprises en termes de décentralisation et de répartition des applications logicielles. A l'origine, celle ci a été essentiellement utilisée pour résoudre les problématiques d'interopérabilité entre les différentes technologies informatiques distribuées utilisées en entreprise. C'est une solution très efficace en termes de réutilisabilité, d'interopérabilité et de réduction de couplage entre les différentes entités implémentant leurs propres systèmes d'information. SOA est un modèle d'interaction applicative qui permet de décomposer une application en un ensemble de composants basiques (c.-à-d. services), de décrire le schéma d'interaction entre ces composants en utilisant un format d'échange bien défini, le plus souvent XML, et des couplages externes par l'intermédiaire d'une couche d'interopérabilité des interfaces, le plus souvent un service Web.

2.2.3 Architecture standard des services Web

Les services Web fournissent un moyen standard d'échange de données entre différentes applications logicielles fonctionnant sur une variété de plateformes et/ou de frameworks. L'architecture des services Web est une architecture qui identifie des éléments globaux nécessaires pour assurer l'interopérabilité entre les services. Ces éléments correspondent aux notions d'annuaire, de bus, de contrat et de protocole de communication. Cette architecture est illustrée dans la Figure 2.2

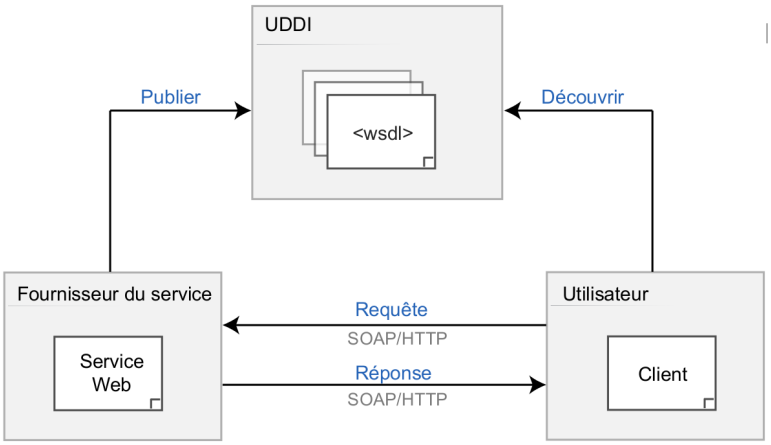


Figure 2.2 Une Architecture de service Web basique[4]

2.3 Fonctionnement des services Web

Le fonctionnement des services Web s'articule autour de trois acteurs principaux illustrés par le schéma suivant[4] Figure 2.3

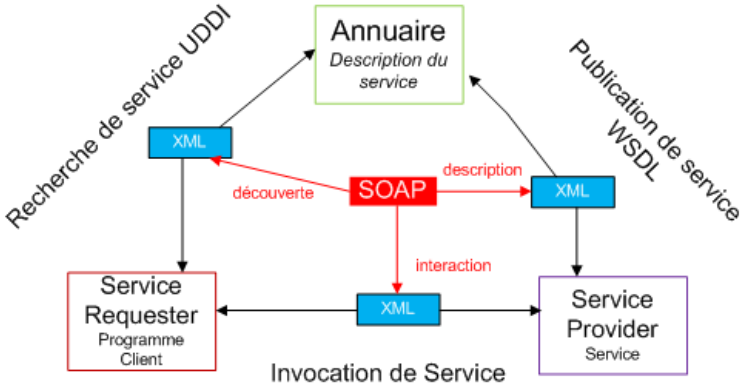


Figure 2.3 Fonctionnement des services Web

2.3.1 Service provider service

Le fournisseur de service met en application le service Web et le rend disponible sur Internet.

2.3.2 Service requester programme client

C'est n'importe quel consommateur du service Web. Le demandeur utilise un service Web existant en ouvrant une connexion réseau et en envoyant une demande en XML (REST, XML-RPC, SOAP).

2.3.3 Annuaire service registry

Le registre de service est un annuaire de services. Le registre fournit un endroit central où les programmeurs peuvent publier de nouveaux services ou en trouver. Les interactions entre ces trois acteurs suivent plusieurs étapes :

- **La publication du service :** le fournisseur diffuse les descriptions de ses services Web dans l'annuaire.
- **La recherche du service :** le client cherche un service particulier, il s'adresse à un annuaire qui va lui fournir les descriptions et les URL des services demandés afin de lui permettre de les invoquer.
- **L'invocation du service :** une fois que le client récupère l'URL et la description du service, il les utilise pour l'invoquer auprès du fournisseur de services.

2.4 Les technologies standards

2.4.1 Le protocole SOAP

La question relative à l'échange de données entre services web est gérée par le protocole :
Simple Object Access Protocol (SOAP) est un produit de Microsoft et IBM. Sa première version a été acceptée par le W3C en 2000. SOAP est un protocole de type requête/réponse fonctionnant sur le protocole de communication HTTP. C'est un mécanisme d'échange de messages XML par l'intermédiaire d'un protocole de communication.
SOAP est un protocole de l'architecture SOA qui assure la messagerie. Du fait qu'il est basé sur XML, il permet l'échange de données structurées indépendamment des langages de programmation ou des systèmes d'exploitation.
C'est une spécification XML qui définit un protocole d'échange de données structurées entre des applications dans un environnement distribué et hétérogène.
Un message SOAP est composé d'un élément *envelope* . L'envelope contient à son tour deux éléments fils : une entête (facultative) et un corps encore appelé *body* . L'entête contient des informations nécessaires aux applications pour interpréter la charge utile. La charge utile se trouve dans l'élément *body* . Le code XML suivant nous donne la structure générale d'un message SOAP. Dans ce code, nous avons présenté les balises principales d'un document SOAP. L'enveloppe est représentée par *env :Envelope* , l'en-tête est représentée par *env :Header* et le corps est dans la balise *env :Body* . Le nom de domaine est représenté par : *xmlns :env="http ://schemas.xmlsoap.org/soap/envelope/"* .

Structure générale d'un message SOAP

```
<?xml version="1.0" ?>
<env :Envelope xmlns :env="http://schemas.xmlsoap.org/soap/
   envelope/">
<env :Header/>
<env :Body ... >
...
</env :Body>
</env :Envelope >
```

La structure des messages SOAP se divise en quatre parties (Figure 2.4) :

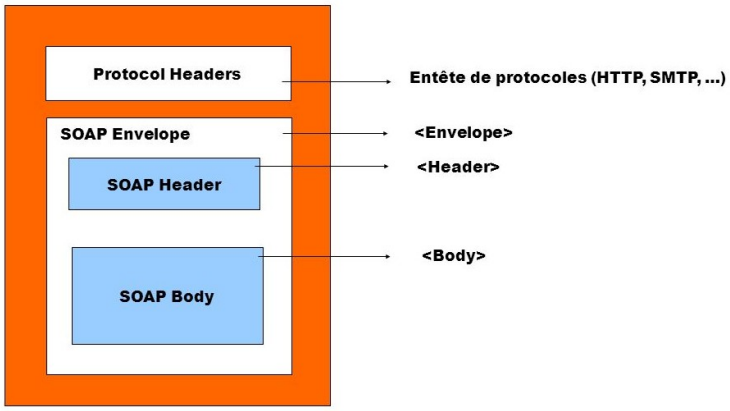


Figure 2.4 La structure des messages SOAP

- **Le http Header :** Le protocole http envoie une requête post. L'entête HTTP se trouve juste avant le message SOAP, et définit le destinataire du message, les règles d'encodage HTTP, etc. Le champ SOAP action peut être utilisé pour indique l'intention de la requête SOAP. Cette information peut être utilisée par un firewall pour filtrer les messages. Ce champ est obligatoire mais peut être vide si on n'indique pas l'intention de la requête.
- **L'enveloppe SOAP :** L'enveloppe contient l'espace de nommage définissant la version de SOAP utilisée, et les règles de sérialisation, et d'encodage.
- **Le header SOAP :** Cette partie du message est optionnelle. Elle sert a transmettre des informations nécessaire pour l'exécution de la requête SOAP aux intermédiaires qui recevront le message. On y précise généralement des informations liées aux transactions, à l'authentification, etc.

  Le header est composé d'un ou plusieurs champs : l'attribut actor, désignant le destinataire du header, l'attribut mustUnderstand, qui indique si le processus est optionnel.
- **Le body :** Le body SOAP contient toutes les informations que l'on veut transmettre à l'application distante. Le contenu du body est normalisé dans SOAP RPC, pour modéliser une requête et sa réponse. Le body de la requête contient l'identifiant de l'objet distant, le nom de la méthode a exécuter et les éventuels paramètres le body de la réponse contient le résultat de l'exécution de la requête.

2.4.2 Le référentiel UDDI

La question relative à l'identification d'un service web est gérée par le référentiel universel : *Universal Description, Discovery and Integration* (UDDI) est né d'une collaboration entre IBM [5] et ARIBA [6] , et ensuite définit par OISIS. Il normalise une solution d'annuaires distribués de services

5. IBM : International Business Team
6. ARIBA : http ://www.ariba.com/

Web permettant, d'une façon standard, à la fois de publier et d'explorer les services, soit pour une utilisation sur un réseau public, soit dans l'infrastructure interne d'une organisation. UDDI offre un mécanisme fondé sur des normes pour classer, cataloguer et gérer les services Web de tel sorte qu'ils peuvent être découverts et utilisés par d'autres applications.

2.4.3 Le langage de description WSDL

La question relative à la description d'un service web est gérée par un langage de description des services web :
Web Service Description Language (WSDL) est un langage se basant sur XML qui permet de décrire toutes les informations nécessaires pour invoquer un service Web, à savoir la description du contenu des messages, l'endroit où le service est disponible et le protocole de communication à utiliser pour communiquer avec le service. WSDL utilise la notation XML pour décrire les formats des messages des requêtes/réponses, ce qui le rend indépendant de tout langage de programmation et de toute plateforme (p. ex. système d'exploitation).
En résumé, WSDL est un contrat entre un client et un serveur sans dépendance particulière pour une plateforme ou un langage. Ce contrat fait état des spécifications d'interfaces définissant le service Web, en particulier les opérations qu'il réalise et le type de message échangé.

Histoire de WSDL

WSDL 1.1 a été présenté comme une note W3C par Ariba, IBM et Microsoft pour décrire services pour l'activité XML du W3C sur les protocoles XML dans Mars de 2001.
WSDL 1.1 n'a pas été approuvé par le Consortium World Wide Web (W3C), mais il a publié un projet pour la version 2.0 qui sera une recommandation (un fonctionnaire standard), et donc approuvé par le W3C
WSDL décompose les services Web en trois éléments spécifiques identifiables qui peuvent être combinés ou réutilisés une fois définies. Les trois principaux éléments de WSDL qui peuvent être définies séparément sont :

- Les types
- opérations
- binding

WSDL Eléments

Un document WSDL contient les éléments suivants :

- **Définition :** Il est l'élément racine de tous les documents WSDL. Il définit le nom du service web, déclare plusieurs espaces de noms utilisés dans le Reste du document, et contient tous

les éléments de service.

- **Types de données :** Les types de données à utiliser dans les messages sont sous la forme de XML schémas.
- **Message :** Il est une définition abstraite des données, sous la forme d'un message présenté soit comme un document entier ou comme arguments pour être mappé à une invocation de méthode.
- **Opération :** Il est la définition abstraite de l'opération pour un message , comme nommer un, file de messages , ou processus d'affaires de la méthode, qui va accepter et traiter le message.
- **Type de port :** Il est un ensemble abstrait des opérations mappé sur un ou plusieurs points d'extrémité, la définition de la collection d'opérations pour une liaison ; la collecte des opérations, comme il est résumé, peuvent être mises en correspondance avec plusieurs transports par le biais de Diverses liaisons.
- **Port :** Il est une combinaison d'une liaison et une adresse de réseau, en fournissant les Adresse cible de la communication de service.
- **Service :** Les services et les ports définissent l'emplacement du service Web. Le service contient le nom du service Web et une liste des ports. En plus de ces éléments principaux, la spécification WSDL définit également les éléments suivants éléments d'utilité :
- **Documentation :** Cet élément est utilisé pour fournir lisible par l'homme La documentation et peut être inclus à l'intérieur de tout autre élément de WSDL.
- **Importation :** Cet élément est utilisé pour importer d'autres documents WSDL ou XML Schémas.

2.5 Conclusion

Dans ce chapitre nous avons présenté le concept des services web comme étant la dernière technologie pour l'intégration et l'interopérabilité des systèmes réparties.
Il est nécessaire de faire le point sur la technologie des services Web. Les services Web est un terme qui décrit un ensemble de protocoles standards utilisés pour établir un domaine d'intégration des applications.
L'un des facteurs ayant contribué au succès des services Web est sans doute l'utilisation des standards Internet tels que XML et HTTP. En conséquence, tout système capable d'analyser du texte et de communiquer via un protocole de transport Internet standard peut communiquer avec un service Web. XML a engendré l'apparition de nouveaux protocoles tels que SOAP pour l'échange de messages, WSDL pour la description de services et UDDI pour la publication et la découverte de services. Ces protocoles reposent sur une architecture orientée services (SOA), et correspondent à des composants logiciels qui peuvent être combinés, grâce à un langage de composition, pour

former de nouveaux services plus élaborés.

CHAPITRE 3

LE WEB SÉMANTIQUE

LES SECTIONS

3.1 Le Web sémantique 28
3.2 Historique du web sémantique 29
3.3 Objectifs du web sémantique 29
3.4 La pile du web sémantique 30
3.4.1 URI (Uniform Resource Identifier) 30
3.4.2 XML (eXtensible Markup Language) 31
3.4.3 RDF (Resource Description Framework) 31
Sérialisation de triplets RDF 33
3.4.4 RDF Schéma (RDFS) 34
3.4.5 OWL (Ontology Web Language) 35
3.5 Ontologie 36
3.5.1 définition D'ontologie 36
3.5.2 cycle de vie 37
3.5.3 types d'ontologie 38
Ontologie de représentation de connaissances 38
Ontologie de haut niveau / supérieure (Top-level / Upper-model) 38
Ontologie Générique (Generic ontology) 39
Ontologie du domaine (Domain ontology) 39
Ontologie de Taches (Task ontology) 39
Ontologie d'application (Application ontology) 39
3.6 Composantes d'une ontologie 39
3.7 Domaines d'applications des ontologies 42
3.8 Conclusion 43

3.1 Le Web sémantique

Le web sémantique est un ensemble de normes pour le partage de données et la sémantique de ces données sur le Web pour une utilisation par les applications. Regardons cette définition une ou deux phrases à la fois, et puis nous allons examiner ces questions plus en détail.

"Un ensemble de normes"

Avant de Tim Berners-Lee a inventé le World Wide Web, systèmes hypertextes plus puissants étaient disponibles, mais il a construit sa spécifications autour simples qu'il a publiés comme normes publiques. Cela a permis aux gens de mettre en œuvre son système sur leur propre (qui est, à écrire leurs propres serveurs Web, les navigateurs Web, et en particulier les pages web), et son système a grandi pour devenir le plus grand système hypertexte. Berners-Lee a fondé le *World Wide Web Consortium* (W3C) pour superviser ces normes, et le web sémantique est également construit sur des normes W3C : le modèle de données *Resource Description Framework* (RDF), le langage d'interrogation *SPARQL Protocol and RDF Query Language* (SPARQL), et *RDF Schema* (RDFS) et *Web Ontology Language* (OWL) normes pour stocker les vocabulaires et les ontologies. Un produit ou un projet peut faire face à la sémantique, mais si elle ne pas utiliser ces normes, il ne peut pas se connecter et faire partie du web sémantique, pas plus que d'un système hypertexte de 1985 pourrait créer un lien vers une page sur le World Wide Web sans en utilisant les standards HTML ou HTTP. (Il ya ceux qui sont en désaccord sur ce dernier point.).

"pour le partage de données ... sur le Web pour une utilisation par les applications"

Le web original de Berners-Lee a été conçu pour fournir des documents lisibles(human-readable). Si vous voulez voler d'un aéroport à l'autre côté dimanche après-midi, vous pouvez aller à un site de la compagnie aérienne, remplir un formulaire de requête, puis lire les résultats de la requête hors de l'écran avec vos yeux. Sites de comparaisons d'avion ont des programmes qui extraient des pages web à partir de plusieurs sites de compagnies aériennes et d'extraire les informations dont ils ont besoin, dans un processus connu sous le nom "screen scraping : La capture de données d'écran", avant d'utiliser les données pour leurs propres pages Web. Avant d'écrire un tel programme, un développeur au site de comparaison aérienne doit analyser la structure HTML du site Web de chaque compagnie aérienne pour déterminer où le programme de "screen scraping" doit rechercher les données dont il a besoin. Si une compagnie aérienne redessine son site Web, le développeur doit mettre à jour son programme de "screen scraping" pour tenir compte de ces différences.
Berners-Lee est venu avec l'idée de données liées comme un ensemble de meilleures pratiques

pour le partage de données sur l'infrastructure Web de sorte que les applications peuvent plus facilement récupérer des données à partir de sites publics sans avoir besoin de "screen scraping : La capture de données d'écran", par exemple, de laisser votre programme de calendrier obtenir de l'information de vol à partir de plusieurs sites de compagnies aériennes dans un format commun, lisible par machine. Ces meilleures pratiques recommandent l'utilisation des *Uniform Resource Identifier* (URI) de nommer les choses et l'utilisation des normes telles que RDF et SPARQL. Ils fournissent d'excellentes lignes directrices pour la création d'une infrastructure pour le web sémantique.

3.2 Historique du web sémantique

En 1994, lors de la première conférence WWW à Genève, plus précisément au CERN, a lieu l'annonce de la création du W3C. C'est d'ailleurs à cette période que Tim Berners-Lee dresse les objectifs du W3C et montre les besoins d'ajouter de la sémantique au Web futur. Il montre alors en quoi les liens hypertextes ou, plus précisément, la façon dont on met en relation les documents sur le Web est trop limitée pour permettre aux machines de relier automatiquement les données contenues sur le Web à la réalité. Compte tenu de l'ambition d'un tel projet, cette idée suscite quelques résistances et controverses qui sont classiquement rencontrées dès qu'on aborde des problématiques liées au domaine de l'intelligence artificielle.
Après cette conférence, mise à part la mise en place des recommandations nécessaires à la construction des documents, le W3C nouvellement créé entame les premières réflexions sur la mise en place du Web sémantique. Ces réflexions aboutissent à la publication d'un premier draft de recommandations sur le Web sémantique en octobre 1997 et d'une seconde en avril 1998. Cette même année, Tim Berners-Lee publie un document sur les toutes premières moutures de ce qui sera plus tard appelé le Web sémantique. Ces moutures consistent à mettre en place les différentes technologies du Web sémantique. Dans ce document, il présente le Web sémantique comme une sorte d'extension du Web des documents, qui constitue une base de données à l'échelle mondiale, afin que toutes les machines puissent mieux lier les données du Web. Cette feuille de route se matérialise par une représentation graphique, le « layer cake », qui montre l'agencement des différentes briques technologiques du Web sémantique.

3.3 Objectifs du web sémantique

Un des principaux objectifs du Web sémantique est de permettre aux utilisateurs d'utiliser la totalité du potentiel du Web : ainsi, ils pourront trouver, partager et combiner des informations plus facilement. Aujourd'hui tout le monde est capable d'utiliser des forums, d'utiliser des réseaux sociaux, de chatter, de faire des recherches ou même d'acheter différents produits. Néanmoins, il serait mieux que la machine fasse tout ceci à la place de l'homme, car actuellement, les machines

ont besoin de l'homme pour effectuer ces tâches. La raison principale est que les pages Web actuelles sont conçues pour être lisibles par des êtres humains et non par des machines. Le Web sémantique a donc comme principal objectif que ces mêmes machines puissent réaliser seules toutes les tâches fastidieuses comme la recherche ou l'association d'informations et d'agir sur le Web lui-même.

3.4 La pile du web sémantique

A l'état actuel, le web sémantique est fondé sur des langages structurés en couches

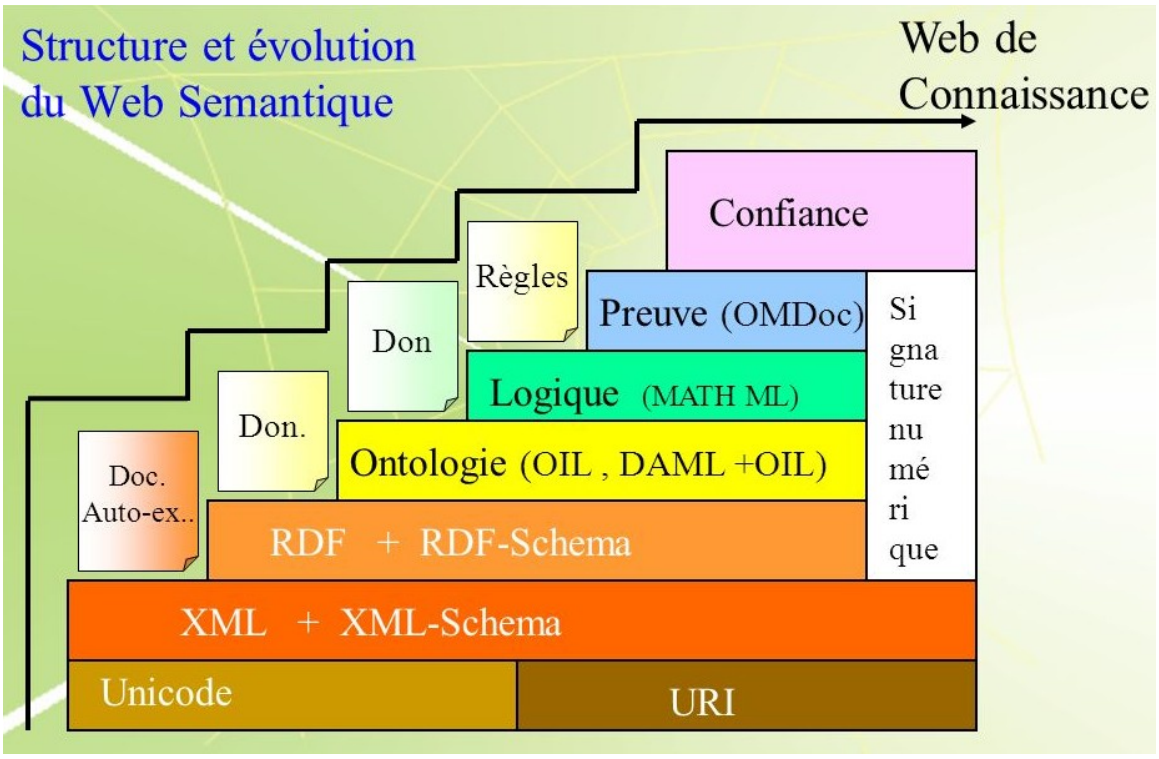


Figure 3.1 couches de web sémantique [5]

3.4.1 URI (Uniform Resource Identifier)

Une URI est un identificateur d'une ressource mais surtout d'un concept, c'est une forme d'étiquette. Définit par la RFC 2396, les URI(s) constituent une des notions fondamentales du web sémantique, elles sont composées d'URN et d'URN.L'URL est un identifiant qui permet de repérer de manière unique un objet sur le réseau. Les URLs peuvent être utilisées pour créer des URIs mais dans le cadre du web sémantique, l'URI n'envoie pas nécessairement vers un document, elle est utilisée comme identifiant d'un concept dans un fichier RDF, cela bien sur ne veut pas dire que L'URI par elle-même a un sens, c'est juste un mot. Le sens est donné de la même façon que dans le

langage naturel, c'est-à-dire par le contexte, les définitions ou les références. L'URN est une URI qui identifie une ressource indépendamment de son endroit physique.

3.4.2 XML (eXtensible Markup Language)

Le langage *eXtended Markup Language* (XML) est un format général de documents orienté texte. Il s'est imposé comme un standard incontournable de l'informatique. Il est aussi bien utilisé pour le stockage de documents que pour la transmission de données entre applications. Sa simplicité, sa flexibilité et ses possibilités d'extension ont permis de l'adapter à de multiples domaines allant des données géographiques au dessin vectoriel en passant par les échanges commerciaux. De nombreuses technologies se sont développées autour de XML et enrichissent ainsi son environnement. Le langage XML dérive de SGML [1] et de *Hyper Text Markup Language* (HTML) . Comme ces derniers, il s'agit d'un langage orienté texte et formé de balises qui permettent d'organiser les données de manière structurée.

Exemple de fichier XML valide

```
<Personne sex="homme"> <!-- personnes -->
<Nom>BELHACHEMI</Nom>
<Prenom>ABDELHAK</Prenom>
</Personne>
<Personne sex="homme"> <!-- personnes -->
<Nom>OMEIRI</Nom>
<Prenom>KOUIDER</Prenom>
</Personne>
```

3.4.3 RDF (Resource Description Framework)

Resource Description Framework (RDF) est un modèle de données générique basé sur la théorie des graphes, permettant de représenter de l'information avec des triplets de la forme (sujet, prédicat, objet). C'est le modèle standard pour l'échange de données sur le Web de données. Il facilite l'intégration des données de sources variées et permet une évolution des schémas de données au cours du temps sans que cela ne nécessite une modification au niveau des clients du Web de données. Il est en certains points semblables au modèle entité-relation. Un triplet est un ensemble de trois informations liées.

1. SGML : Standard Generalized Markup Language.

- **Le sujet** (la ressource au sujet de laquelle une information va être définie, représente par une URI).
- **Le prédicat** (le type d'information qui va être définie a propos du sujet, représente par une URI).
- **L'objet** (la valeur du prédicat du sujet, étant soit une URI ou un identifiant local, soit un littéral).

Pour mettre cela en relation avec le modèle de graphes RDF, un sommet représente une entité (ressource identifiée par une URL ou entité locale identifiée par un identifiant local), et une arrête représente un triplet RDF, c'est-a-dire le prédicat qui lie le sujet à l'objet. Un ensemble de triplets RDF constitue donc un graphe dirigé étiqueté (directed labelled graph).

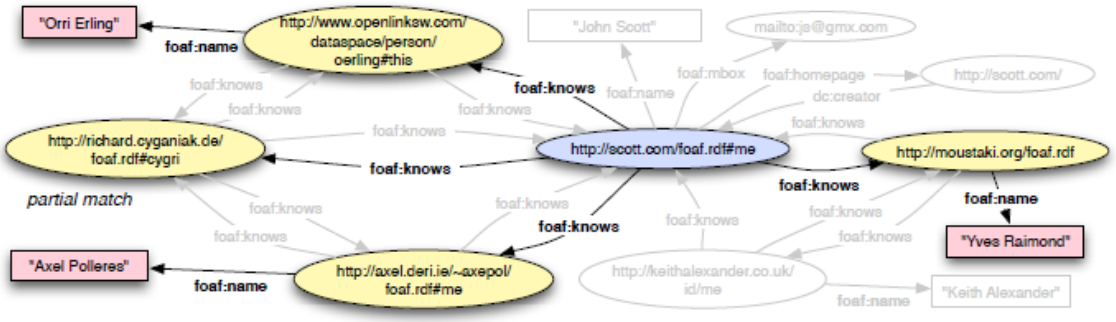


Figure 3.2 Représentation visuelle d'un graphe RDF [5]

Figure 3.2 est une représentation visuelle d'un graphe RDF. Ce type de représentation des triplets présente l'avantage d'être facile à interpréter car c'est un modèle mental relativement simple. Dans cette même figure, nous observons deux types de sommets :

- Premièrement les entités, représentées par un contour ovale. Les entités peuvent être de deux types :
  - soit une ressource, identifiée par une URI (ce qui est le cas de toutes les entités de la figure)
  - soit un nœud local, identifie par un identifiant local. Ces nœuds locaux sont lies a d'autres entités (par des arrêtes représentant des triplets RDF), mais ne sont pas adressables directement via une URI.
- Deuxièmement les littéraux, représentes par un contour rectangulaire. Il va de soi que les objets littéraux ne se trouvent qu'en extrémité de graphe. Il existe également des quadruplets (quads), une extension des triplets rajoutant le contexte (la provenance de l'information) comme quatrième information au triplet.

Sérialisation de triplets RDF

RDF définit une méthode de représentation de l'information sous forme de triplets. Apres avoir décrit le modèle conceptuel à la section précédente, il est utile de s'intéresser a la manière de sérialiser cette information. Pour rappel, la sérialisation est le processus d'encodage d'une information (comme un modèle ou un graphe RDF) physiquement sous forme d'octets ou de bits, permettant ainsi un stockage sur disque ou un transfert sur le réseau. An de procéder a la sérialisation de triplets RDF, il existe plusieurs syntaxes de sérialisation des graphes RDF : RDF/XML, N-Triple, Turtle, et Notation3 (N3). Cependant, il semble que RDF/XML et Turtle soient les plus utilises. Avant de détailler cela, il est nécessaire d'introduire la notion d'espace de noms (namespace). Une source A pourrait décrire une ressource dont elle choisira X comme identifiant. Une source B pourrait décrire une autre ressource, dont elle choisira également X comme identifiant. X est donc un identifiant unique d'une ressource au sein de sa source (i.e. unique au sein de A, et unique au sein de B), mais pas un identifiant unique sur le Web (puisque le X de A ne représente pas le même concept que le X de B). On dit alors que X est un identifiant au sein d'un espace de noms, en l'occurrence l'espace de noms de A ou celui de B. Lors de la sérialisation ou de l'écriture de triplets, que ce soit en RDF/XML ou en Turtle, il est en général préférable de définir à l' avance les différents espaces de noms qui vont être définis, et d'attribuer à chaque espace de noms un "nom local", beaucoup plus court, afin d'abréger l'écriture des triplets et de les rendre plus lisibles.

— **RDF/XML**

est une représentation d'un graphe RDF sous forme XML. Cette syntaxe est plus difficilement lisible par des humains, mais présente l'avantage d'être basée sur le standard XML, et donc facilement parsable par une machine. Un exemple de sérialisation en RDF/XML pourrait être le suivant, dont l'illustration est donnée à la Figure 3.3 :

Représentation d'un graphe RDF sous forme XML

```
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-
   ns#" xmlns:predicats="http://predicats.org/">
<rdf:Description rdf:about="http://ulb.ac.be/PierreGewelt">
<predicats:AnneeDeNaissance>1989</predicats:anneeDeNaissance>
<predicats:VilleNatale>Bruxelles</predicats:villeNatale>
</rdf:Description>
</rdf:RDF>
```

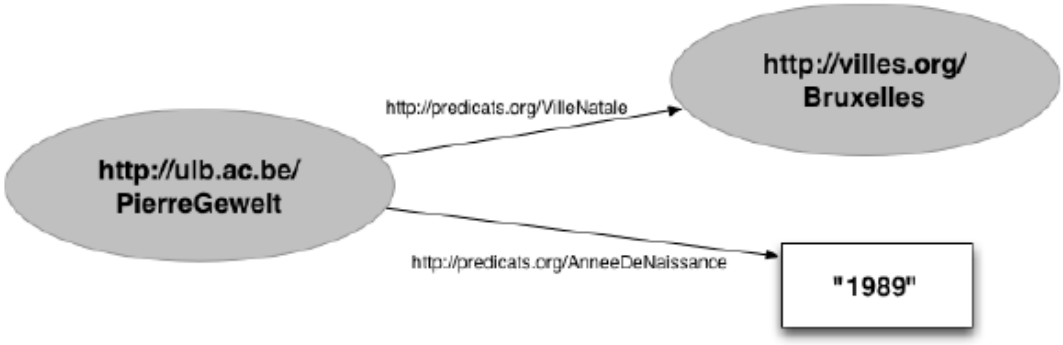


Figure 3.3 Représentation graphique d'un ensemble de triplets

— **Turtle**

Il existe trois syntaxes fort similaires ayant pour but d'être plus facilement lisibles par les humains que RDF/XML : N-Triple, Turtle et Notation3 (N3). N-Triple est un sous-ensemble de Turtle, qui est lui-même un sous-ensemble de Notation3. Turtle étant le plus utilisé, c'est lui qui va être décrit ici. Turtle a pour but d'être une sérialisation de graphes RDF beaucoup plus facile à écrire et à interpréter par un humain. Son écriture est beaucoup plus directe et plus proche du modèle conceptuel de triplets. La même information que celle décrite ci-dessus en RDF/XML s'écrit de cette manière en Turtle :
@prefix prédicats : <http ://predicats.org/>.
<http ://ulb.ac.be/PierreGewelt>predicats :AnneeDeNaissance "1989".
<http ://ulb.ac.be/PierreGewelt>predicats :VilleNatale"Bruxelles".

3.4.4 RDF Schéma (RDFS)

RDF Schéma (RDFS) est un langage permettant de représenter de simples vocabulaires RDF sur le Web de données. La sémantique des prédicats et les classes d'entités peuvent être définies grâce aux vocabulaires et aux ontologies. Ceux-ci laissent notamment la possibilité de créer des règles d'inférence, via un mécanisme de classes et de propriétés, permettant ainsi de générer une grande quantité d'information et de Triplets dérives des triplets existants. Cela permet d'augmenter les capacités de raisonnement que peuvent avoir les machines et les programmes traitant les données du Web sémantique.
RDFS permet notamment de définir des classes et sous-classes de ressources. Ainsi si l'on définit que B est une sous-classe de A, alors toute ressource X ayant pour triplet (X rdf :type B), c'est-à dire étant de type B, est automatiquement également de type A.
De même, il est possible de définir le type de donnée du sujet et de l'objet d'un prédicat. Par exemple, si l'on définit que le prédicat estMarieA a pour sujet une ressource de type Personne

et a pour objet également une ressource de type Personne, alors si l'on définit le triplet (Romeo estMarieA Juliette), le système peut automatiquement inférer que les ressources Romeo et Juliette sont toutes deux de type Personne.
Au dessus de la couche "vocabulaire RDF" qu'est RDFS, il existe des ontologies pour le Web sémantique comme expliqué précédemment, offrant encore plus de possibilités d'inférence de données. Le langage d'ontologies le plus connu est OWL et sera détaillé à la section suivante.

3.4.5 OWL (Ontology Web Language)

Web Ontology Language, dont l'acronyme est OWL, est un langage d'ontologie pour le Web sémantique, avec une sémantique formellement définie. OWL permet donc d'étendre les possibilités de raisonnement déjà offertes par RDFS. Elle exprime des classes et propriétés, stockées elles-mêmes sous forme de documents au format RDF par exemple, publiées donc sur le Web de données.
OWL existe lui-même en 3 sous-langages : OWL Full, OWL DL et OWL Lite.
En complément de RDFS, OWL 2 permet de définir des propriétés supplémentaires, comme déclarer des classes disjointes (e.g. aucune ressource représentant une personne ne peut appartenir à la fois à la classe Homme et à la classe Femme). Il est également possible de déclarer des propriétés spécifiques sur les prédicats, comme la transitivité, l'unicité, l'inverse ou la symétrie.
Par exemple, si l'on définit que le prédicat est UnAncetreDe est une propriété transitive, alors s'il existe un triplet (A est UnAncetreDe B) et un triplet (B est UnAncetreDe C), on peut automatiquement inférer le triplet (A est UnAncetreDe C). L'unicité ex- prime que si l'on définit que le prédicat aPourPereBiologique est une propriété unique, alors on garantit qu'aucune ressource n'aura deux pères (biologiques) déclares. Il est également possible d'exprimer des propriétés inverses, par exemple définir les prédicats estParentDe et estEnfantDe comme propriétés inverses l'une de l'autre. Quant à la symétrie, il est possible de déclarer un prédicat (e.g. estMarieA) comme symétrique, créant alors un lien à double sens entre le sujet et l'objet.
Les ontologies ont un pouvoir d'expression relativement fort, car comme nous l'avons vu, elles permettent de déclarer un ensemble de règles concernant les données, d'inférer de nouvelles données sur base des données existantes, etc. Cependant, cela en augmente grandement la complexité de calculs et la décidabilité. Au vu du nombre de données disponibles sur le Web de données, et puisque ce nombre est amené à grandir exponentiellement dans les années à venir, l'utilisation de RDFS est privilégiée pour le Web Sémantique, même si cela réduit les possibilités d'exploitation du potentiel du Web de données.

3.5 Ontologie

3.5.1 définition D'ontologie

Ontologie est une branche de la métaphysique qui s'intéresse à l'existence, à l'être en tant qu'être et aux catégories fondamentales de l'existant. En effet, ce terme est construit à partir des racines grecques « ontos » qui veut dire ce qui existe, l'être, l'existant, et « logos » qui veut dire l'étude, le discours, d'où sa traduction par « l'étude de l'être » et par extension de l'existence. Dans la philosophie classique, l'ontologie correspond à ce qu'Aristote appelait la Philosophie première (protè philosopha), c'est-à-dire la science de l'être en tant qu'être, par opposition aux philosophies secondes qui s'intéressaient, elles, à l'étude des manifestations de l'être (les existants). Ontologie : partie de la métaphysique qui s'attache à l'étude ou à la théorie de l'être dans son essence, indépendamment des phénomènes de son existence.

- Dans le cadre de l'intelligence artificielle, **Neeches** et ses collègues furent les premiers à proposer une définition à savoir : « Une ontologie définit les termes et les relations de base du vocabulaire d'un domaine ainsi que les règles qui indiquent comment combiner les termes et les relations de façon à pouvoir étendre le vocabulaire » [6]
- En 1993, **Gruber** propose la définition suivante : « Spécification explicite d'une conceptualisation ». [6] Cette définition a été modifiée légèrement par **Borst** comme « spécification formelle d'une conceptualisation partagée ». [6]
- Ces deux dernières définitions sont regroupées dans celle de **Studer** comme « spécification formelle et explicite d'une conceptualisation partagée ». [6]
  - Formelle : l'ontologie doit être lisible par une machine, ce qui exclut le langage naturel.
  - Explicite : la définition explicite des concepts utilisés et des contraintes de leurs utilisations.
  - Conceptualisation : le modèle abstrait d'un phénomène du monde réel par identification des concepts clefs de ce phénomène.
  - Partagée : l'ontologie n'est pas la propriété d'un individu, mais elle représente un consensus accepté par une communauté d'utilisateurs.
- Pour Guarino et Giaretta « une ontologie est une spécification rendant partiellement compte d'une conceptualisation ». Swartout et ses collègues la définissent comme suit : « une ontologie est un ensemble de termes structurés de façon hiérarchique, conçue afin de décrire un domaine et qui peut servir de charpente à une base de connaissances ».[6]
- même notion est également développée par Gomez comme : « une ontologie fournit les moyens de décrire de façon explicite la conceptualisation des connaissances représentées dans une base de La connaissances ».
- **GRUBER** : « In the context of computer and information sciences, an ontology defines a set

> of representational primitives with which to model a Domain of knowledge or discourse. The representational primitives are typically classes (or sets), attributes (or properties), and Relationship (or relations among class members)». [6]

3.5.2 cycle de vie

Puisque les ontologies sont destinées à être utilisées comme des composants logiciels dans des systèmes répondant à des objectifs opérationnels différents, leur développement doit s'appuyer sur les mêmes principes que ceux appliqués en génie logiciel. Ainsi, les ontologies doivent être considérées comme des objets techniques évolutifs et possédants un cycle de vie qui nécessite d'être précisé. Dans ce contexte, les activités liées aux ontologies sont, d'une part, des activités de gestion de projet (planification, contrôle, assurance qualité), et d'autre part, des activités de développement (spécification, conceptualisation, formalisation) ; s'y ajoutent des activités transversales de support telles que l'évaluation, la documentation, la gestion de la configuration.
Un cycle de vie inspiré du génie logiciel. Nous l'avons adapté à nos besoins et proposons notre vision du cycle de vie d'une ontologie. Il comprend une étape initiale de détection et de spécification des besoins qui permet notamment de circonscrire précisément le domaine de connaissances, une étape de conception qui se subdivise en trois phases, une étape de déploiement et de diffusion, une étape d'utilisation, une étape incontournable, d'évaluation, et enfin, une sixième étape consacrée à l'évolution et à la maintenance du modèle. Après chaque utilisation significative, l'ontologie et les besoins doivent être réévalués et l'ontologie peut être étendue et, si nécessaire, en partie reconstruite. La validation du modèle de connaissances est au centre du processus et se fait de manière itérative.
M. Fernandez insiste sur le fait que les activités de documentation et d'évaluation sont nécessaires à chaque étape du processus de construction, l'évaluation précoce permettant de limiter la propagation d'erreurs. Le processus de construction peut et doit être intégré au cycle de vie d'une ontologie comme indiqué en Figure 3.4. [7]

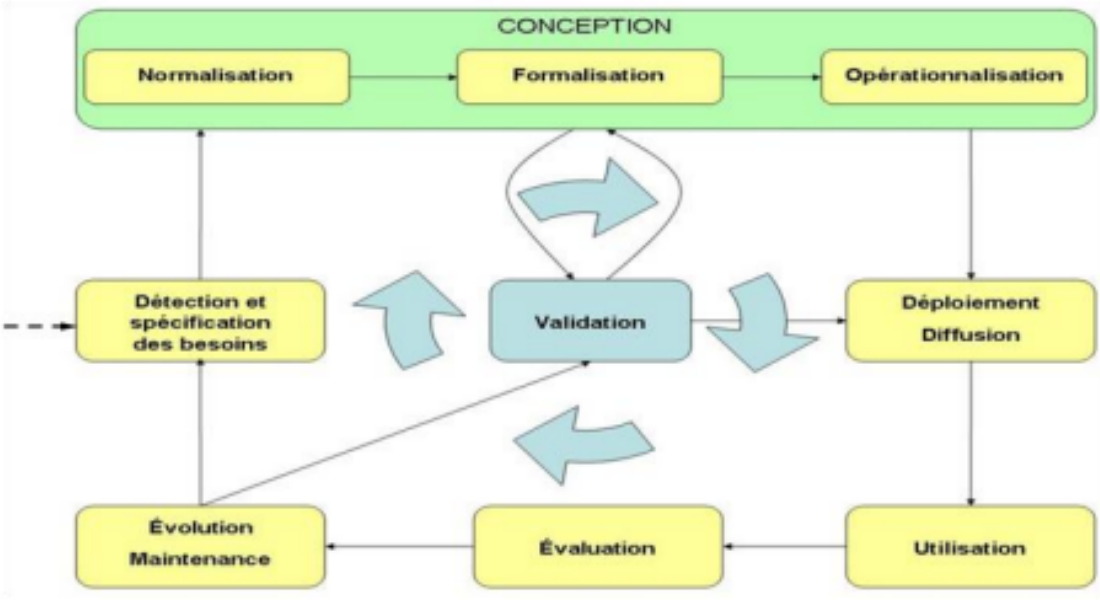


Figure 3.4 cycle de vie d'une ontologie

3.5.3 types d'ontologie

On distingue six types d'ontologie :

Ontologie de représentation de connaissances

Modélise les représentations primitives utilisées pour la formalisation des connaissances sous un paradigme donné. Par exemple, une ontologie sur le formalisme des Topic Maps comportera les concepts : Topic, Type de Topic, Association, Occurrence, Type Occurrence.

Ontologie de haut niveau / supérieure (Top-level / Upper-model)

Elle exprime des conceptualisations valables dans différents domaines. Elle décrit des concepts très généraux comme l'espace, le temps, la matière, les objets, les évènements, les actions, etc. ces concepts ne dépendent pas d'un problème ou d'un domaine particulier, et doivent être, du moins en théorie, consensuels à de grandes communautés d'utilisateurs. Ce type d'ontologies est fondé sur la théorie de la dépendance. Son sujet est l'étude des catégories des choses qui existent dans le monde. Comme les concepts de haute abstraction tels que les entités, les évènements, les états, les processus, les actions, le temps, l'espace, les relations, les propriétés ...

Ontologie Générique (Generic ontology)

Elle est appelée également noyau ontologique, modélise des connaissances moins abstraites que celles véhiculées par l'ontologie de haut niveau mais assez générales néanmoins pour être réutilisées à travers différents domaines. Cette ontologie inclut un vocabulaire relatif aux choses, évènements, temps, espace, causalité, comportement, fonction, etc.

Ontologie du domaine (Domain ontology)

Cette ontologie exprime des conceptualisations spécifiques à un domaine, elle est pour plusieurs applications de ce domaine. Elle fournit les concepts et les relations permettant de couvrir les vocabulaires, activités et théories de ces domaines. Selon, l'ontologie du domaine caractérise la connaissance du domaine ou la tache est réalisée. Par exemple, dans le contexte du e-Learning, le domaine peut être celui de formation. C'est ce type qui nous sera utile pour définir les objectifs ciblés de notre projet.

Ontologie de Taches (Task ontology)

L'ontologie de taches fournit un vocabulaire systématisé des termes employés pour résoudre des problèmes liés aux taches qui peuvent être ou non du même domaine. Elle fournit un ensemble de termes au moyen desquelles nous pouvons décrire généralement comment résoudre un type de problèmes. Elle inclut des noms, des verbes et des adjectifs génériques dans les descriptions de taches.

Ontologie d'application (Application ontology)

C'est l'ontologie la plus spécifique, elle contient des concepts dépendants d'un domaine et d'une tache particuliers, elle est spécifique et non réutilisable. Ces concepts correspondent souvent aux rôles joués par les entités du domaine lors de l'exécution d'une certaine activité. Il s'agit donc ici de mettre en relation les concepts liés à une tache particulière de manière à en décrire l'exécution.

3.6 Composantes d'une ontologie

Comme nous l'avons abordé, les ontologies fournissent un vocabulaire commun d'un domaine et définissent la signification des termes et des relations entre elles. La connaissance dans les ontologies est principalement formalisée en utilisant les cinq types de composants à savoir : **concepts** (ou classes), **relations** (ou propriétés), **fonctions**, **axiomes** (ou règles) et **instances** (ou individus).

— Les concepts, aussi appelés termes ou classe de l'ontologie, correspondent aux abstractions pertinentes d'un segment de la réalité (le domaine du problème) retenus en fonction des

objectifs qu'on se donne et de l'application envisagée pour l'ontologie .

- Les relations traduisent les associations (pertinentes) existant entre les concepts présents dans le segment analysé de la réalité. Ces relations incluent les associations suivantes :
  - Sous classes de (généralisation-spécialisation)
  - Partie de (agrégation ou composition)
  - Associe à
  - Instance de, etc

  Ces relations nous permettent d'apercevoir la structuration et l'interrelation des concepts, les uns par rapport aux autres.
- Les fonctions constituent des cas particuliers de relations, dans laquelle un élément de la relation, (le nième) est défini en fonction des N-1 éléments précédents
- Les axiomes constituent des assertions, acceptées comme vraies, à propos des abstractions du domaine traduites par l'ontologie.
- Les instances constituant la définition extensionnelle de l'ontologie ; ces objets véhiculent les connaissances (statiques, factuelles) à propos du domaine du problème.

Les méthodes d'ingénierie ontologique Il existe une multitude de méthodes d'ingénierie ontologique mais l'absence de directives structurées et communes ralentisse le développement d'ontologie à l'intérieur et entre les équipes, l'extension de n'importe quelle ontologie, la possibilité de réutilisation de l'ontologie. On entend par méthodologie, les procédures de travail, les étapes, qui décrivent le pourquoi et le comment de la conceptualisation puis de l'artefact construit. Dans la suite nous allons présenter que deux méthodes de l'état d'ontologies :

- **La méthode de Bachimont :** [10] Cette méthode propose de contraindre l'utilisateur à un engagement sémantique en introduisant une normalisation sémantique des termes manipulés dans l'ontologie. La méthode de normalisation suit trois étapes :
  - **Normalisation sémantique :** l'utilisateur doit choisir les termes du domaine et les normaliser en explicitant leurs propriétés et en exprimant les identités et les différences dans leur voisinage proche. La place d'une notion dans l'ontologie doit être justifiée par rapport à la communauté et la différence avec le père et la fratrie.
  - **Formalisation des connaissances** : Cette étape consiste à désambiguïser les notions de l'ontologie référentielle obtenue par l'étape précédente et choisir leurs sens pour un domaine spécifique. Cela peut nécessiter la création de nouveaux concepts, l'ajout de propriétés et d'axiomes.
  - **Opérationnalisation des connaissances** Le système utilise un langage opérationnel de représentation de connaissances qui possède les caractéristiques nécessaires pour répondre aux besoins exprimés lors de la spécification du système.
- **La méthode METHONTOLOGY** C'est une méthodologie mise au point par l'équipe du

laboratoire de l'intelligence artificielle de l'Université polytechnique de Madrid. Cette méthode inclut : [10]

- L'identification du processus de développement de l'ontologie.
- Le cycle de vie basé sur l'évolution de prototypes.
- Les techniques de gestion de projet (planification, assurance qualité), de développement (spécification, conceptualisation, formalisation, implémentation, maintenance) et des activités de support (intégration, évaluation, documentation).

Les outils de construction d'ontologie

On distingue deux familles d'outils : les outils de construction d'ontologie dépendants de formalisme de représentation et les outils de construction d'ontologie indépendants de formalisme de représentation.

- **Les outils dépendants de formalisme de représentation**
  - **Ontolingua** : Ontolingua est un serveur d'édition d'ontologies. Il utilise des classes, des relations, des fonctions, des instances et des axiomes pour décrire une ontologie. Une relation peut contenir des propriétés nécessaires (contraintes) ou nécessaires et suffisantes qui définissent la relation. En plus le serveur Ontolingua offre la possibilité d'intégrer les ontologies Ontolingua, ce qui permet la construction modulaire des ontologies.[10]
  - **OntoSaurus** : OntoSaurus de l'Information Science Institute de l'Université de Southern California est composé de deux modules : un serveur utilisant LOOM comme langage de représentation des connaissances, et un serveur de navigation créant dynamiquement des pages HTML qui affichent la hiérarchie de l'ontologie ; le serveur utilise des formulaires HTML pour permettre à l'usager d'éditer l'ontologie.[10]
  - **WebOnto** : WebOnto du Knowledge Media Institute de l'Open University, est une application Web pour naviguer et développer collaborativement les ontologies. Il supporte la navigation collaborative, la création et l'édition d'ontologies sur le Web. Les ontologies WebOnto sont implémentées dans le langage OCML. WebOnto distingue quatre types d'ontologies : ontologie de domaine, ontologie de tâche, ontologie de méthode, et ontologie d'application. [10]
  - **OilEd** : OilEd (Oïl Editor) est un éditeur d'ontologies utilisant le formalisme OIL. Il est essentiellement dédié à la construction de petites ontologies dont on peut ensuite tester la cohérence à l'aide de FACT, un moteur d'inférences bâti sur OIL. [10]
- **Les outils indépendants de formalisme de représentation**
  - **Protégé 2000** : est une interface modulaire permettant l'édition, la visualisation, le contrôle d'ontologie, l'extraction d'ontologies à partir de sources textuelles, et la fusion semi-automatique d'ontologies. Le modèle de connaissances sous-jacent à protégé 2000 est issu du modèle des frames et contient des classes, des slots (propriétés) et des

facets (valeurs des propriétés et contraintes), ainsi que des instances des classes et des propriétés. Il autorise la définition de méta-classes, dont les instances sont des classes, ce qui permet de créer son propre modèle de connaissances avant de bâtir une ontologie.

- **ODE et Web Ode** : L'outil ODE (Ontology Design Environment) permet de construire des ontologies au niveau connaissance, comme le préconise la méthodologie METHONTOLOGY. L'utilisateur construit son ontologie dans un modèle de type frame, en spécifiant les concepts du domaine, les termes associés, les attributs et leurs valeurs, les relations de subsomption.
- **OntoEdit** : OntoEdit (Ontology Editor) est également un environnement de construction d'ontologies indépendant de tout formalisme. Il permet l'édition des hiérarchies de concepts et de relations et l'expression d'axiomes algébriques portant sur les relations, et de propriétés telles que la généricité d'un concept. Des outils graphiques dédiés à la visualisation d'ontologies sont inclus dans l'environnement. OntoEdit intègre un serveur destiné à l'édition d'une ontologie par plusieurs utilisateurs. Un contrôle de la cohérence de l'ontologie est assuré à travers la gestion des ordres d'édition.

3.7 Domaines d'applications des ontologies

Système d'information L'intégration d'une ontologie dans un système d'information vise à réduire, voire éliminer, la confusion conceptuelle et termilogique à des points clefs du système, et à tendre vers une compréhension partagée pour améliorer la communication, le partage, l'interopérabilité et le degré de réutilisation possible, ce qui permet de déclarer formellement un certain nombre de connaissances utilisées pour caractériser les informations gérées par le système, et de se baser sur ces caractérisations et la formalisation de leur signification pour automatiser des taches de traitement de l'information. L'ontologie retrouve maintenant dans une large famille de systèmes d'information. Elle est utilisée pour :

- Décrire et traiter des ressources multimédia.
- Assurer l'interopérabilité d'applications en réseaux.
- Piloter des traitements automatiques de la langue naturelle.
- Construire des solutions multilingues et interculturelles.
- Permettre l'intégration des ressources hétérogènes d'information.
- Vérifier la cohérence de modèles.
- Permettre les raisonnements temporel et spatial.
- Faire des approximations logiques ; etc.

Ces utilisations des ontologies se retrouvent dans de nombreux domaines d'applications tel que :

- Intégration d'information géographique.
- Gestion de ressource humaine.

— Aide à l'analyse en biologie, suivi médicale informatisé.
— Commerce électronique.
— Enseignement assisté par ordinateur.
— Bibliothèque numériques.
— Recherche d'informations.

3.8 Conclusion

La manipulation des ressources du web par des machines requiert l'expression ou la description de ces ressources. Plusieurs langages ont été définis a cet effet, ils permettent d'exprimer données et métadonnées, de décrire les services web et leur fonctionnement, et de disposer d'un modèle abstrait de ce qui est décrit grâce a l'expression d'ontologies (RDFS, OWL), ces langages sont la base pour standardiser le tout, mais comme l'écrit , TIM BERNERS-LEE, le web sémantique est ce que nous obtiendrons si nous réalisons le même processus de globalisation sur la représentation des connaissances que celui que le web fit initialement sur l'hypertexte.

CHAPITRE 4

LES SERVICES WEB SÉMANTIQUE

LES SECTIONS

4.1 Introduction 46
4.2 Définition 46
4.3 Approches proposées pour réalisation web sémantique 46
4.3.1 approche basées sur langages sémantique 46
L'approche OWL-S 46
L'approche WSMO 49
4.3.2 A base d'annotation sémantique 50
L'approche USDL 51
L'approche SAWSDL 52
4.4 Conclusion 56

4.1 Introduction

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

4.2 Définition

Les services web sémantique sont des services décrits de telle sorte qu'un agent logiciel puisse interpréter les fonctionnalités offertes par le service web. Un agent logiciel doit entre capable de lire la description d'un service web pour déterminer si le service web fournit les fonctionnalités désirées, et s'il est lui-même capable d'utiliser ce service. Pour permettre cela, la description du service web doit être supplémentée en information sémantique interprétable par machine. Les paramètres du service web doivent être décrits de définition de vocabulaires organisés en ontologies.
La combinaison des technologies du web sémantique a celles des services web permettra de :

- Automatiser la découverte de services web, c'est-à-dire localisation automatique des services web qui fournissent une fonctionnalité particulière et qui répondent aux propriétés demandées par l'utilisation. Pour pouvoir effectuer une découverte automatique, le procédé de découverte devrait etre basé sur similitude sémantique entre la description déclarative, faite par l'utilisateur, du service demandé et celle du service offert.
- La composition automatique des services web.
- Automatiser l'invocation d'un service web, cela implique l'automatisation l'exécution du service web par le programme d'utilisateur ou par un agent.
- Automatiser l'interopérabilité des services web.

4.3 Approches proposées pour réalisation web sémantique

4.3.1 approche basées sur langages sémantique

L'approche OWL-S

OWL-S est un langage d'ontologie pour services web.il est basé sur le langage. Cette ontologie a pour objectif de d'écrire de façon non ambiguë les services web de telle sorte qu'un agent logiciel puisse exploiter automatiquement ces informations. OWL-S permet : la découverte automatique, la composition et l'interopérabilité de services web ainsi que la surveillance automatique de leur

exécution. OWL-S décrit un service a l'aide des trois classes suivantes :

- serviceProfile : définit le service web.
- serviceModel : définit le fonctionnement du web service.
- serviceGrounding : définit comment accéder au service web.

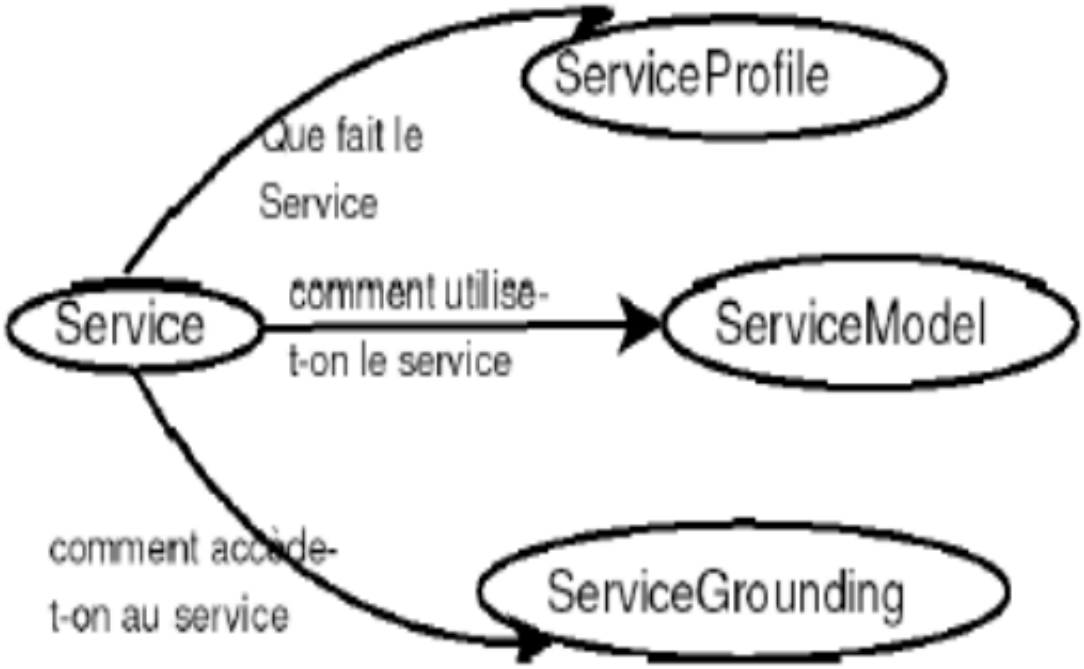


Figure 4.1 type d'information [8]

- ServiceProfile : Le ServiceProfile fournit une superclasse de chaque type de description de haut niveau du service. ServiceProfile ne prescrit aucune représentation des services, mais elle prescrit les informations de base pour lier toute instance de profil avec une instance de service. Il existe une relation bidirectionnelle entre un service et un profil, de telle sorte qu'un service peut être lié à un profil et un profil à un service. Ces relations sont exprimées par les propriétés des présents et des presentedBy.
  Presents Décrit une relation entre une instance de service et une instance de profil, on dit essentiellement que le service est décrite par le profil.
  PresentedBy Est l'inverse de presents ; il précise qu'un profil donné décrit un service.
- ServiceModel : La classe serviceModel décrit le fonctionnement du service web. Ceci est en exprimant les transformations faites par le service web sur les données (input a output), et transformations d'état (préconditions et effets). Les services web peuvent être modélisés

avec OWL-S en tant que processus grâce à la classe process. La classe ainsi définie est une sous-classe de serviceModel. Pour décrire un processus, on spécifie ces entrées, sorties et ses états. Les transitions d'un état à un autre sont décrites par les préconditions et les effets de chaque processus. Il existe trois types de processus :

— 1. les processus atomique (atomicprocess) : sont directement invocable (en les faisant passer les messages appropriés) et d'exécuter en une seule étape, dans la mesure où le demandeur de services est concerné. Autrement dit, ils prennent un message d'entrée, faire quelque chose, puis retournent leur message de sortie. Pour chaque processus atomique, il faut prévoir une mise à la terre qui permet à un demandeur de services pour construire des messages au processus de ses entrées et de déconstruire les réponses [9]

```
<owl:Class rdf:ID="Atomicprocess">
<owl:subClassOf rdf:resource="#Process"/>
</Owl:Class>
```

— 2. les processus composites (compositeprocess) : processus composites sont décomposable dans les processus d'autres (non composites ou mixtes) ; leur décomposition peut être spécifié en utilisant les structures de contrôle telles que la séquence et If-Then -Else, qui sont discutés ci-dessous. Parce que la plupart des constructions de contrôle ont des noms qui rappellent des structures de contrôle des langages de programmation, il est facile de perdre de vue une différence fondamentale : un processus composite est pas un comportement un service fera, mais un comportement (ou un ensemble de comportements) le client peut effectuer en envoyant et recevant une série de messages. Si le processus composite a un effet global, le client doit effectuer l'ensemble du processus afin de parvenir à cet effet. On n'a pas encore donné une description précise de ce que cela signifie pour exécuter un processus, mais tout ce que nous voulons dire est que, par exemple, si un composite est une séquence, le client envoie une série de messages qui font appel à toutes les étapes dans l'ordre. [10]

— 3. les processus simple (simpleprocess) : les processus simple ne sont pas invocable et ne sont pas associés à grounding, mais, comme les processus atomiques, ils sont conçus comme ayant des exécutions à une seule étape. Des procédés simples sont utilisés comme éléments de l'abstraction ; un processus simple peut être utilisé soit pour fournir une vue de (une façon spécialisée d'utiliser) un processus atomique, ou une représentation simplifiée de certains processus composite (à des fins de planification et de raisonnement). Dans le premier cas, le processus est simple realizedBy le processus atomique ; dans ce dernier cas, le processus simple expandsTo le processus composite [11]

— ServiceGrounding : Les classes serviceGrounding précise les détails de la façon d'accéder au service - de détails ayant principalement à voir avec le protocole et les formats de message , la sérialisation , le transport, et addressing.A Grounding peut être considéré comme une application à partir d'un résumé à une spécification concrète de ces éléments de description de services qui sont nécessaires pour interagir avec le service - en particulier, pour nos besoins , les entrées et sorties des processus atomiques . Notez que dans OWL- S, à la fois le ServiceProfile et ServiceModel sont considérés comme des représentations abstraites ; seuls les ServiceGrounding traitent du niveau concret de la spécification
Jusqu'à présent, nous avons seulement montré comment les définitions WSDL peuvent se référer aux déclarations OWL -S correspondant. Il reste à mettre en place un mécanisme par lequel les constructions WSDL pertinentes peuvent être référencées dans OWL- S. La classe WsdlGrounding OWL- S, une sous-classe de mise à la terre, sert à cette fin. Chaque instance WsdlGrounding, à son tour, contient une liste d'instances de WsdlAtomicProcessGrounding.

L'approche WSMO

WSMO est une ontologie qui décrit les différents aspects relatifs à la composition dynamique des services Web, y compris la découverte dynamique, la sélection, la médiation et l'invocation. Elle est basée sur WSMF (Web Service Modelling Framework) qui spécifie les éléments principaux pour décrire les services Web sémantiques. De tels éléments incluent ontologies, objectifs, services Web et médiateurs. Les ontologies définissent la terminologie, utilisée par les autres éléments, en termes de concepts, relations, fonctions, instances et axiomes. Les buts indiquent ce que l'utilisateur attend du service. La description du service Web définit les fonctionnalités offertes par le service. Les médiateurs lient les différents éléments afin de permettre l'interopérabilité entre les composants hétérogènes. Le langage WSML est utilisé pour décrire formellement tous les éléments de WSMO et l'environnement d'exécution WSMX [16] permet la découverte, la sélection, la médiation, l'invocation et l'interopérabilité des services Web sémantiques. Je présente dans la suite les motivations de WSMO.

— **Motivations de WSMO** WSMO partage avec OWL-S les mêmes motivations à savoir la découverte, l'invocation la composition automatique des services Web. Cependant WSMO ajoute à celles-ci l'objectif suivant : Un découplage fort entre les composants et un rôle central pour la médiation. L'un des principes fondamentaux de WSMO consiste en la séparation totale entre les différents éléments impliqués dans la composition des services Web. A cet effet, WSMO veut distinguer comment le client formule sa demande et comment le fournisseur expose son service. Sachant que la demande et l'offre sont décrites de façons différentes, un travail important doit être effectué afind'établir la correspondance entre la

demande et l'offre. Ce travail est le rôle des médiateurs.

— **Facettes de WSMO** Les facettes de WSMO sont les ontologies, les médiateurs, les services Web et les objectifs.

Les ontologies fournissent la sémantique compréhensible par une machine pour les informations utilisées par tous les acteurs d'un service Web. WSMO permet d'importer une ontologie dans une autre ontologie soit directement, quand il n'y pas de conflits entre les concepts, soit indirectement, par le biais d'un médiateur qui va résoudre les conflits possibles. Les blocs de base d'une ontologie sont concepts, relations, fonctions, instances et axiomes. Un ensemble de propriétés non fonctionnelles est donné généralement au début de chaque définition d'une ontologie.

Les médiateurs permettent de lier différentes ressources hétérogènes et résoudre les incompatibilités à plusieurs niveaux. Il peut y avoir une incompatibilité entre les données ou entre les processus.

Dans le premier cas, la médiation sert à établir la correspondance entre les différentes terminologies. Le deuxième type d'incompatibilité apparaît au moment de la communication entre différents services Web hétérogènes et dans ce cas, le médiateur fournit la fonctionnalité pour une analyse d'exécution de deux services Web donnés, et compense les éventuelles disparités.

Les services Web WSMO sont composés d'un ensemble optionnel de propriétés non fonctionnelles, un mécanisme permettant d'importer des ontologies soit directement en utilisant imports Ontology, soit indirectement via un médiateur, une capacité décrivant la fonctionnalité du service en termes de pré et post conditions et effets et une interface décrivant le comportement du service vis-à-vis de ses partenaires.

4.3.2 A base d'annotation sémantique

L'annotation sémantique consiste à assigner aux entités d'un document électronique, des liens à leurs descriptions sémantiques. Ce type de métadonnées fournit, à propos des entités du document annoté, des informations de classes et d'instances liées à une ontologie décrivant le contenu de la ressource annotée

les annotations sémantiques peuvent donc être définies comme : un ensemble d'instanciations attachées à un document (HTML ou XML) , qui peuvent être des instances de classes (avec des URIs uniques), des instances d'attributs d'une classe (propriétés Instanciées d'une instance de classe vers une instance de type de données) ou une instance de relation (propriété instanciées d'une instance de classe vers une instance d'une autre classe).

Par exemple en associant une notice comprenant des champs : Auteur, Date de création, Date de modification, Mots-clés, à une page Web, ceci permet de considérer ces informations non plus

seulement comme comprenant du texte qui ne pourra qu'être traité statistiquement par un robot indexeur (« wrapper »), mais également des informations structurées dotées de Sémantique connue et utilisable comme telle par un agent logiciel.
L'annotation se distingue de l'indexation automatique par l'utilisation d'une ou plusieurs ontologies qui définissent un domaine global de référence permettant de cadrer et de normaliser les annotations effectuées, par ailleurs une ressource annotée doit l'être non pas par une liste de mots clefs, mais par une ou plusieurs ontologies.

L'approche USDL

Développé au sein du laboratoire de recherche ALPS (Applied Logic Programming-Languages and Systems) de l'Université du Texas, l'approche USDL [12] (Universal Service-Semantics Description Language) propose une description sémantique et formelle pour les services Web. Elle se base sur l'utilisation de l'ontologie OWL WordNet [13].

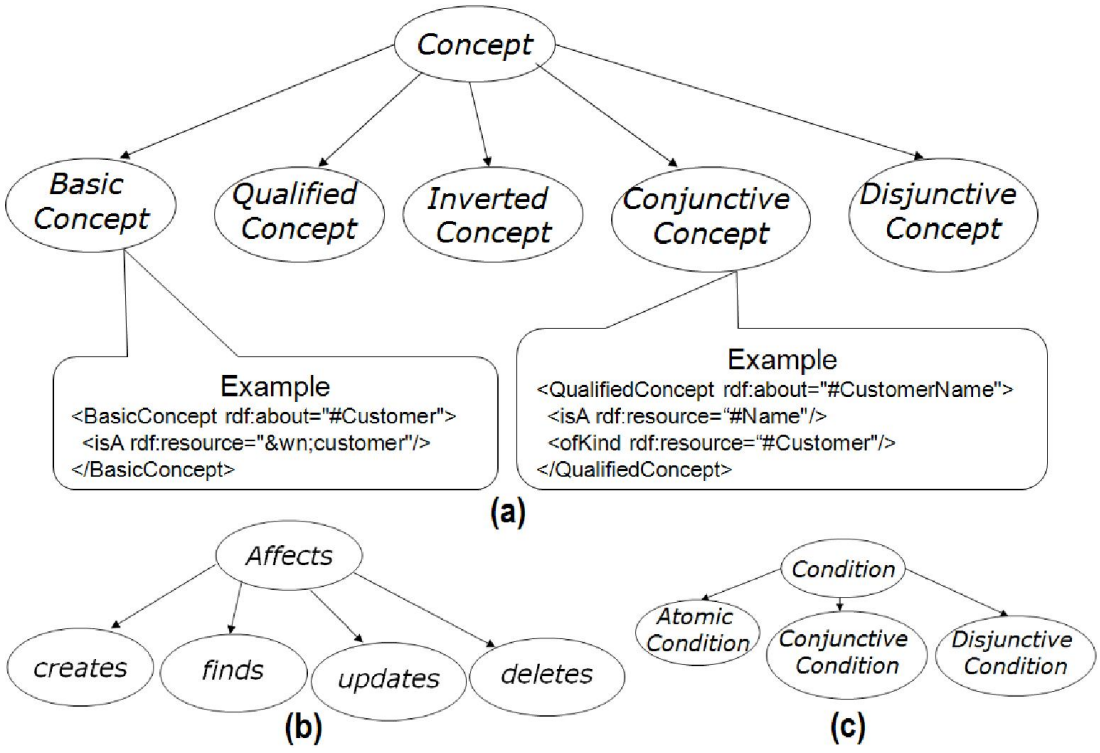


Figure 4.2 Principaux éléments de description dans USDL [14]

— **Éléments de base de l'approche USDL** Cette approche offre un langage qui décrit formellement et sémantiquement les services Web [15]. Pour ce faire, il considère que l'ontologie OWL WordNet est un outil universel grâce auquel tout le monde retrouve une représentation commune des concepts du monde réel. USDL propose une description basée sur des

concepts en utilisant la logique de proposition. L'approche justifie cette motivation pour l'aspect formel et l'utilisation de WordNet par l'abondance de relations sémantiques entre les éventuels termes figurant dans cette ontologie et dont on peut profiter pour décrire les services Web. Elle propose de puiser dans les relations fournies par l'ontologie OWL WordNet, telles que l'hyponymie, l'antonymie, la synonymie, etc. USDL propose une construction de description sémantique au-dessus de la description syntaxique WSDL 1.1. Cette construction engendre des modifications dans la syntaxe et la structure [16]. La Figure 4.2 illustre les trois principaux éléments pour décrire un service Web :

- La classe Concept : est une classe générique pour modéliser des concepts du monde réel,
- La propriété Affects : est une classe générique pour décrire des effets du service sur le monde réel,
- La classe Conditions : est une classe générique pour décrire des contraintes pour le service.

— **Construction d'une description USDL** Comme WSDL 1.1, USDL décrit un service Web en termes des balises XML "Messages" et "PortType". L'utilisation de l'ontologie WordNet intervient à ces niveaux. Pour décrire les opérations du service (au niveau des PortType) et leurs paramètres (au niveau des Messages), USDL Propose de les associer à des concepts basiques, des concepts qualifiés, des concepts inversés, des concepts conjonctifs et des concepts disjonctifs provenant de WordNet. La sémantique spécifie comment l'environnement autour du service Web est affecté, autrement dit, l'effet de l'exécution du service Web sur le monde. USDL limite sa considération aux propriétés sémantiques des effets secondaires suite à une opération de création, de mise à jour, de suppression ou de recherche. Par ailleurs, si aucun de ces effets n'est appliqué, alors un effet générique est utilisé. Toute application qui aurait besoin de faire appel à un service USDL doit impérativement avoir la capacité de raisonner avec le lexique et les termes spécifiques De WordNet appelés atoms (tels que les lemmas, senses, lexical semantic relations, etc.)

En appliquant l'approche USDL sur un service Web écrit en WSDL, il sera formellement défini comme une fonction associée à un ou des effets, en spécifiant le tout en concepts provenant de WordNet.

L'approche SAWSDL

Semantic annotation for WSDL (SAWSDL) [17] est un langage sémantique de description de service Web. Il est évolutif et compatible avec les standards des services Web existants, et plus spécifiquement avec WSDL [18]. SAWSDL augmente l'expressivité du langage WSDL avec la Sémantique en utilisant des concepts analogues à ceux utilisés dans OWL-S [19]. D'une part SAWSDL, fournit un mécanisme permettant d'annoter sémantiquement les types de données, les opérations, les en-

trées et les sorties de WSDL et d'autre part, il ajoute des éléments pour spécifier les préconditions, les effets et les catégories des services Web. Les aspects relatifs à la qualité et l'orchestration des services ne sont pas traités dans SAWSDL.

- **Motivations de SAWSDL** : En plus de la découverte et l'invocation automatiques des services Web, déjà citées dans l'introduction, SAWSDL vise la réalisation des objectifs suivants :
  - Un langage au dessus des standards des services Web existants : les standards de services Web sont devenus rapidement une technologie préférée pour l'intégration des applications. Les entreprises investissent dans des projets d'intégration basés sur des services Web. Par conséquent, les concepteurs de SAWSDL considèrent que n'importe quelle approche pour la description sémantique des services Web doit être compatible avec l'existant. A cet effet SAWSDL, une extension sémantique de WSDL a été conçue,
  - Concevoir un langage incrémental : il est relativement facile de modifier les outils existants autour de WSDL afin qu'ils s'adaptent à SAWSDL. Ce qui fait de SAWSDL une approche incrémentale,
  - Le mécanisme d'annotation doit être indépendant du langage de représentation de la sémantique : Il y a un certain nombre de langages potentiels pour représenter la sémantique comme Web Service Modelling Language (WSML) [20], OWL et Unified Modelling Language (UML) [21]. Chaque langage offre différents degrés d'expressivité. La position des concepteurs de SAWSDL est qu'il n'est pas nécessaire d'attacher les standards des services Web à un langage sémantique particulier. Cette approche est la vision décrite dans [22] et donne plus de flexibilité aux développeurs.
- **Annotation sémantique :** L'annotation sémantique des documents WSDL est possible grâce à l'extensibilité devWSDL 2.0. En effet, conceptuellement WSDL 2.0 est doté des constructions suivantes : interface, Opération, message, binding, service et endpoint. Les trois premiers à savoir interface, opération et message concernent la définition abstraite du service tandis que les trois autres sont relatifs à l'implémentation du service.

  SAWSDL fournit des mécanismes pour référencer des concepts de modèles définis à l'extérieur du document WSDL. Cela se fait grâce à l'attribut "sawsdl". Il existe trois extensions de cet attribut. La première est modelReference et permet d'associer un composant WSDL ou XML Schéma à un concept d'une ontologie. Les deux autres sont liftingSchemaMapping et lowering- SchemaMapping et permettent de spécifier la correspondance (dit Mapping) entre les données Sémantiques et les éléments XML. Les SchemaMapping sont utilisés pour établir la correspondance Entre les structures des entrées et des sorties, et est utile lorsque les structures XML demandées par le client et celles fournies par le service sont différentes. L'annotation des interfaces, opérations, entrées/sorties et les types XML simples s'effectue

en leur associant un concept dans une ontologie par le biais de l'attribut modelReference. Cependant, L'annotation des types de données XML complexes peut nécessiter en plus un Schéma Mapping. En effet, deux services Web peuvent manipuler le même type complexe mais avec deux structures différentes.

L'annotation des "complexType" et "simpleTypes" s'effectue en leur associant un concept dans une ontologie par le biais de l'attribut modelReference.
Le fichier xml au dessous présente un extrait de la description SAWSDL de service Web "CityHotelService". L'opération "get-HOTEL" contient deux composant :
L'input qui contient l'attribut Message : "get-HOTELRequest" et l'output qui contient l'attribut Message : "get-HOTELResponse"
dans les input ou les output l'attribut "Message" renvoi à la balise "Message" qui bien déterminer les noms des inputs et outputs de cette service dans la balise "part" au l'attribut "name"
chacun entre les deux concept "HOTEL" et "CITY" et annoté dans un "complexType" ou "simpleTypes" par l'attribut modelReference .
précisent que l'opération prend en entrée un nom d'une ville et renvoie en sortie les noms des hôtel dans cette ville.

Extrait de la description SAWSDL du CityHotelService

```
<?xml version="1.0" encoding="UTF-8"?>
<wsdl:definitions name="CityHotel" targetNamespace="http:
  //127.0.0.1/services/sawsdl_wsdl11/CityHotel" xmlns="http:
  //127.0.0.1/services/sawsdl_wsdl11/CityHotel" xmlns:apachesoap
  ="http://xml.apache.org/xml-soap" xmlns:impl="http:
  //127.0.0.1/services/sawsdl_wsdl11/CityHotel-impl" xmlns:wsdl=
  "http://schemas.xmlsoap.org/wsdl/" xmlns:tns="http:
  //127.0.0.1/services/sawsdl_wsdl11/CityHotel" xmlns:sawsdl="
  http://www.w3.org/ns/sawsdl" xmlns:xsd="http://www.w3.org
  /2001/XMLSchema" xmlns:wsdlsoap="http://schemas.xmlsoap.org/
  wsdl/soap/" xmlns:intf="http://127.0.0.1/services/
  sawsdl_wsdl11/CityHotel" xmlns:SOAP-ENC="http://schemas.
  xmlsoap.org/soap/encoding/">
  <wsdl:types>
    <xsd:schema version="OWLS2WSDL_Wed_Sep_22_14:33:46_CEST_2010"
        xmlns:xsd="http://www.w3.org/2001/XMLSchema">
      <xsd:annotation>
```

```
    <xsd:documentation source="Translation_(OWL2XSD-SimpleType)
      _of_http://127.0.0.1/ontology/portal.owl#City"/>
  <xsd:documentation source="Translation_(OWL2XSD-SimpleType)_
    of_http://127.0.0.1/ontology/travel.owl#Hotel"/>
</xsd:annotation>
  <xsd:element name="Hotel" type="HotelType"/>
  <xsd:element name="City" sawsdl:liftingSchemaMapping="http:
    //127.0.0.1/services/liftingSchemaMappings/
    city_hotel_service_City_liftingSchemaMapping.xslt" type="
    CityType"/>
  <xsd:simpleType name="HotelType" sawsdl:modelReference="http:
    //127.0.0.1/ontology/travel.owl#Hotel">
    <xsd:restriction base="xsd:string"/>
</xsd:simpleType>
  <xsd:simpleType name="CityType" sawsdl:modelReference="http:
    //127.0.0.1/ontology/portal.owl#City">
    <xsd:restriction base="xsd:string"/>
</xsd:simpleType>
</xsd:schema>
</wsdl:types>
<wsdl:message name="get_HOTELResponse">
  <wsdl:part name="_HOTEL" type="HotelType">
  </wsdl:part>
</wsdl:message>
<wsdl:message name="get_HOTELRequest">
  <wsdl:part name="_CITY" type="CityType">
  </wsdl:part>
</wsdl:message>
<wsdl:portType name="CityHotelSoap">
  <wsdl:operation name="get_HOTEL">
    <wsdl:input message="get_HOTELRequest">
  </wsdl:input>
    <wsdl:output message="get_HOTELResponse">
  </wsdl:output>
  </wsdl:operation>
</wsdl:portType>
```

```
  <wsdl:binding name="CityHotelSoapBinding" type="CityHotelSoap">
    <wsdlsoap:binding style="rpc" transport="http://schemas.
       xmlsoap.org/soap/http"/>
    <wsdl:operation name="get_HOTEL">
      <wsdlsoap:operation soapAction=""/>
      <wsdl:input>
        <wsdlsoap:body use="encoded" encodingStyle="http://
           schemas.xmlsoap.org/soap/encoding/" namespace="http:
           //127.0.0.1/services/sawsdl_wsdl11/CityHotel"/>
      </wsdl:input>
      <wsdl:output>
        <wsdlsoap:body use="encoded" encodingStyle="http://
           schemas.xmlsoap.org/soap/encoding/" namespace="http:
           //127.0.0.1/services/sawsdl_wsdl11/CityHotel"/>
      </wsdl:output>
    </wsdl:operation>
  </wsdl:binding>
  <wsdl:service name="CityHotelService">
    <wsdl:port name="CityHotelSoap" binding="CityHotelSoapBinding
       ">
      <wsdlsoap:address location="http://127.0.0.1/services/
         sawsdl_wsdl11/CityHotel"/>
    </wsdl:port>
  </wsdl:service>
</wsdl:definitions>
```

4.4 Conclusion

Les services web sémantique sont un domaine de recherche émergent. A l'état actuel il existe des technologies pour le développement des services web. Cependant, ces technologies exigent que l'utilisateur humain interagisse avec le système tout au long du processus de découverte de services web.les technologies du web sémantique ont été utilisés pour palier a ce problème en enrichissement les services web de sémantique, ce qui permet l'automatisation des divers aspects relatifs aux services web. Les avantages de l'utilisation du web sémantique pour la description des services web sont nombreux. En plus, de rendre l'interface du service web accessible automatiquement par des machines, ils permettent également, la description de propriétés non fonctionnelles

telles que la qualité de services, les contraintes de sécurité, et l'intégration effective des services web dans des applications industrielles, d'une manière uniforme compréhensible par tous.
Concrètement, ce qui nous intéresse est l'automatisation, autant que possible, des divers aspects relatifs aux web, en particulier découverte des services web sémantique qui fera l'intérêt du prochaine chapitre.

CHAPITRE 5

LA DÉCOUVERTE DES SERVICES WEB

LES SECTIONS

5.1 Introduction 60
5.2 Définition 60
5.3 Critères de découverte 60
5.3.1 Critère d'architecture 60
5.3.2 Critère d'automatisation 61
5.3.3 Critère de matchmaking 61
5.4 Les approches de découverte 62
5.4.1 Approches fonctionnelles 62
Approches Syntaxiques 62
Approches Sémantiques 64
Approches Comportementales 68
5.4.2 Approches Non Fonctionnelles 68
Approches à Base de Réputation et Confiance 69
5.5 Conclusion 69

5.1 Introduction

Le succès des services web a impliqué l'adoption de cette technologie par divers fournisseurs de services a traverses ale web, ce qui a induit l'augmentation du nombre des services web, rendant par suit leur découverte une tache fastidieuse. La découverte de services web présente un axe de recherche important. Divers mécanisme de découvert ont été proposés dans la littérature. Les auteurs ont défini les mécanismes de découvertes comme étant "l'act de localisation d'une description, traitable par machine, d'un service web non connu auparavant décrivant certains critères fonctionnels".
Actuellement la description des services sont publiées dans des registres spécialement conçus a cet effet (ex : uddi).le but de ces registres est de faciliter la recherche des services publiés par les différents organismes commerciaux. Ce pendant, vu le nombre et la diversité des services web, leur découverte reste une tache ardue.

5.2 Définition

La découverte des SWSs est le processus qui permet de localiser tous les services web qui matchent les besoins fonctionnels du demandeur (requête), à cause du nombre important de services qui offrent des fonctionnalités similaires, cette phase peut retourner un nombre important de services comme résultat. En fait, même si un SWS matche les fonctionnalités demandées par l'utilisateur, il peut encore être inacceptable par un autre utilisateur en considérant d'autres paramètres (par exemple, le coût est trop élevé ou la localisation n'est pas adéquate).

5.3 Critères de découverte

Plusieurs critères peuvent être utilisés pour catégoriser les approches de découverte, nous avons :

- Le critère de centralisation/distribution des annuaires.
- Le principe de l'algorithme de matching (syntaxique, sémantique (logique, non logique), hybride, comportemental, non fonctionnel…)
- Le critère d'automatisation

5.3.1 Critère d'architecture

(centralisation/distribution) Ce critère concerne la manière de stockage et de localisation des descriptions de services dans le réseau. Selon ce point de vue, les systèmes peuvent être centralisés ou décentralisés.
Les systèmes centralisés reposent sur un seul annuaire qui peut être géré par un matchmaker. Les systèmes décentralisés stockent les descriptions de services de manière distribuées (tels que

réseaux pair-a-pair (p2p))[23]

5.3.2 Critère d'automatisation

Ce critère concerne le degré d'intervention de l'utilisateur dans la découverte, nous distinguons les approches manuelles et automatiques. La découverte manuelle est initiée et assistée par un utilisateur humain. La découverte automatique n'implique pas l'intervention de l'utilisateur, pendant la recherche. [23]

5.3.3 Critère de matchmaking

Ce point de vue concerne les données a comparer ainsi que l'algorithme d'appariement de la requête avec description du service. On distingue les approches fonctionnelles et les approches non fonctionnelles :
Les approches non fonctionnelles comparant les valeurs de QOS des services web et les classent selon le besoin de l'utilisateur. Les algorithmes de comparaison s'inspirent des méthodes de décision multicritères des systèmes différentiels flous ...
Les approches fonctionnelles comparent les descriptions d'interfaces ou de comportement des services web, elles peuvent être syntaxique (WSDL), sémantique (OWLS, SAWSDL, WSMO) ou comportementales. La figure suivant montre une classification des approches de découverte, en utilisant le critère de matching. [23]

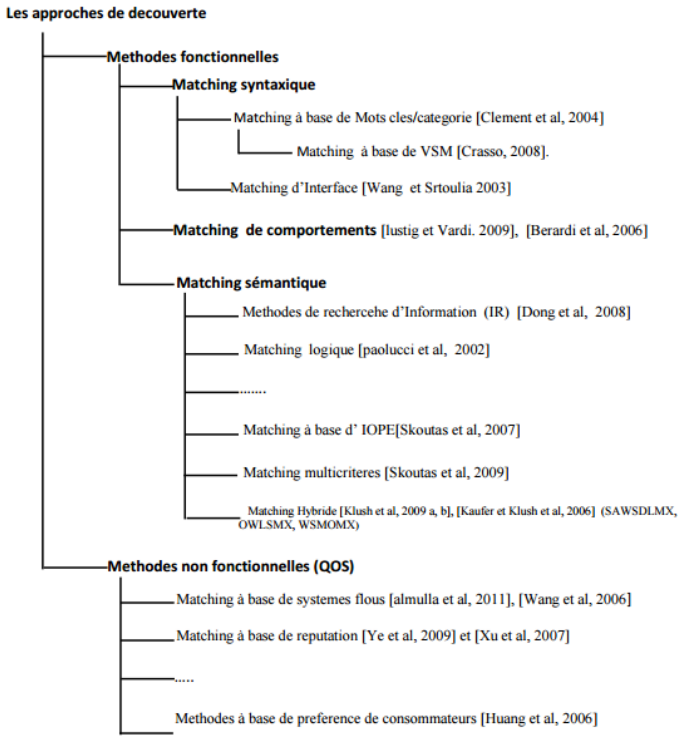


Figure 5.1 quelques approches de découverte [4]

5.4 Les approches de découverte

5.4.1 Approches fonctionnelles

Les capacités fonctionnelles d'un service peuvent inclure, des informations telles que les entrées, les sorties, la catégorie du service, le comportement, les annotations informelles et éventuellement les préférences. Plusieurs approches ont été proposées pour la découverte fonctionnelle de services, on distingue les approches sémantiques, les approches syntaxiques (non sémantiques), et les approches hybrides.

Les approches syntaxiques, emploient les techniques de recherche d'informations pour comparer des interfaces syntaxiques telles que le document WSDL. Les approches sémantiques emploient la logique, le datamining, et même les mesures de similarité pour comparer des interfaces sémantiques.

Approches Syntaxiques

(ou Basées sur les Interfaces Syntaxiques)

Ces approches utilisent généralement l'interface WSDL, comme description de services, et adoptent les techniques de recherche d'informations et éventuellement le clustering pour le matching.

La recherche dans l'annuaire UDDI est l'exemple typique d'une découverte syntaxique, les mots

clés de la requête sont comparés avec les attributs enregistrés, nous pouvons faire une recherche par nom de services ou nom d'entreprise, ou sa catégorie. D'autres attributs tels que la localisation ou le binding template ou les tmodels peuvent être utilisés.
'Liang et al, 2004' emploie le thesaurus wordnet 'Fellbaum, 1998' pour enrichir les mots clés de la requête avec des synonymes. Ceci permet l'augmentation du rappel et l'extension des résultats. Dans 'Li et al, 2004' les auteurs adoptent la recherche par mots clés sur un réseau P2P.
Query by Example (QBE) permet à un utilisateur de rechercher des services en se basant sur un exemple (des mots-clés) [Crasso et al, 2008]. La requête comprend des mots-clés décrivant la fonctionnalité, le nom d'opération et les noms de paramètre. Cette méthodeadopte le modèle de représentation vectoriel ou Vector Space Model (VSM) et la mesure cosinus pour le processus de matching. Les inconvénients de cette méthode sont :

- La mauvaise affectation de catégorie de services affaiblit le rappel.
- L'utilisation des caractères jockers, affaiblit la précision, en plus la non prise en charge de la synonymie dans les documents wsdl réduit le taux de rappel.

[Ernst et al, 2006] traitent les problèmes de substituabilité et de compossibilité de services. Les auteurs supposent que les informations concernant des exécutions passées des services sont disponibles, et proposent un algorithme comportant deux phases. La première phase calcule la similarité des entrées d'un service et des sorties d'un autre, Ces valeurs sont extraites d'un journal d'exécutions passées ensuite le système décide si les deux services sont composables, Sinon, il les marque en tant qu'éventuellement substituables.
Si les services sont considérés comme équivalents, les opérations similaires sont marquées comme interchangeables après découverte de leurs paramètres d'entrées /sorties et génération des éventuelles correspondances. Ces traitements sont effectués, durant la deuxième phase. Les expérimentations conduites par les auteurs confirment l'absence de faux positifs i.e. que la précision =100%. Les auteurs suggèrent le renforcement des tests en utilisant des services concrets (les paramètres d'entrées-sorties ont des valeurs réels). De plus, ils envisagent d'employer une technique d'apprentissage automatique pour améliorer l'appariement des paramètres d'entrées/ sorties.
L'approche [1] utilise deux algorithmes : le matching basé sur les niveaux et le matching des schémas. Le premier compare syntaxiquement les éléments concrets de WSDL, et le deuxième compare les éléments abstraits du document WSDL. Les limites de ce type d'approches (i.e. syntaxiques) sont nombreuses :

- L'ambigüité présente dans le langage naturel est l'inconvénient principal de ces méthodes, en effet les méthodes syntaxiques ont un faible taux de rappel et de précision.
- La nécessite de l'intervention de l'utilisateur : l'utilisateur doit intervenir pour corriger les

décisions et gérer les éventuels conflits et ambigüités rencontrées lors du matching.

Approches Sémantiques

Plusieurs interfaces sémantiques ont été créées pour assurer la découverte et la composition de services WSMO, OWL-S et SAWSDL toutes les approches de cette catégorie adoptent les ontologies pour le matching de la requête avec les services. Nous notons que ces approches, sont plus complexes et plus fiables que les techniques syntaxiques. Nous distinguons 03 classes d'approches sémantiques, les approches logiques, non logiques et hybrides. Les approches logiques exploitent les inférences pour vérifier la compatibilité entre la requête et le l'annotation de service (subsomption, test de consistance…), alors que les approches non logiques exploitent la sémantique implicite ou informelle des services et la traite avec d'autres techniques, telles que le datamining, le matching de graphes, la recherche d'informations, les mesures de similarité. La troisième classe mélange les deux premiers types.

— **Les Approches Logiques** Paolucci présentent l'une des premières approches sémantiques pour la découverte de services, pour cela, ils comparent les sorties de la requête avec les sorties du service offert, selon l'algorithme suivant :
Matching (R.O, S. O)
// Avec S.O dénote la sortie du service S et R.O dénote la sortie de la requête.
Retourner
Exact : si les deux concepts ontologiques R.O et S.O sont équivalents.
Plug-in : si S.O subsume R.O
Subsume : si R.O subsume S.O
Fail : aucune relation d'équivalence ou d subsomption n'existe entre eux

L'algorithme de matching de cette approche est un algorithme glouton-greedyalgorithmequi propose de trouve un max-match entre chaque concept de la requête (entrées/sorties) et les concepts du service publié (entrées/sorties). Il définit quatre classes sémantiques pour représenter le degré de correspondance entre les concepts en se basant sur la relation subsomption de l'ontologie de domaine. La mesure de similarité entre concepts est une fonction discrète qui retourne quatre valeurs (Exact, plugin, subsumes, Fail) selon la véracité de la relation de subsomption.
Notons que le matching des concepts outputs est différent de celui des concepts inputs, les algorithmes 1 et 2 présentent les procédures utilisées dans les deux cas :
DegreeOFMATCH (out R, out A) :
IF out A=out R Then return exact

IF out R subclassof out A THEN return exact
IF out A subsumes out R THEN return plugin
IF out R subsumes out A THEN return subsumes
Otherwise FAIL

Algorithme 1 : règles d'affectation de degré de correspondance des concepts de sorties
Degreeofmatch (inA, inR) :
If inR=inA Then return exact
If inA subclassof inR Then return exact
If inR subsumes inA Then return plugin
If inA subsumes inR Then return subsumes
Otherwise FAIL
Algotihme2 : règles d'affectation de degré de correspondance des concepts d'entrées
Avec :
OutR est le concept output de la requête
OutA est le concept output du service publié
InR est le concept input de la requête
InA est le concept input du service publié
Les classes sémantiques de matching :

- Exact : dans le cas ou les deux concepts sont équivalents ou un concept est un père directe de l'autre, dans la figure 5.2, outR =caret outA=vehicle, alors dom (car, vehicle)=exact Dans le cas des concepts inputs, inA=car et inR=vehicle, alors dom (car, vehicle)=exact.
- Plugin : dans le cas ou outA englobe outR dans la procédure de correspondance des concepts d'output, et inR englobe inA dans la procédure de correspondance des concepts d'inputs, outA=vehicle et outR=sedan, alors dom (sedan, vehicle)=plugin.
- Subsumes : dans les cas ou le concept du service ne répond pas complètement au concept de la requête. le concept output de la requête englobe le concept output du service, outR=vehicle et outA=sedan, alors dom (vehicle, sedan)=subsumes.
- Fail : dans le cas ou n'existe aucune relation de subsumption entre les concepts de la requête et du service.

Dans la littérature, généralement les auteurs affectent des valeurs numériques aux classes de degré de correspondance entre les concepts.
Procédure de matching Le matching entre services et la requête se fait sur deux étapes :

- Matching des concepts output : consiste à faire le matching entre tous les concepts out-

put de la requête et les concepts output du service, le résultat est représenté par la variable match.output (nommée aussi Gdom-out) qui contient le plus petit degré trouvé.

— Matching des concepts input : consiste à faire le matching entre tous les concepts output de la requête et les concepts input du service, le résultat est représenté par la variable match. Input (nommée aussi Gdom-in) qui contient le plus petit degré trouvé.

L'algorithme 3 présente la procédure de matching des outputs :

outputMatch (outputsRequest, outputsAdvertisement)
GlobaldegreeMatch=Exact
Forral outR in outputsRequest do
Find outA in outputsAdvertisement such That
DegreeMatch=maxdegreeMatch (outR, outA)
If (degreeMatch==Fail) return Fail
If (DegreeMatch< globaldegreeMatch)
GlobaldegreeMatch=degreeMatch
Return sort (record Match) ;

Algorithme 3 : procédure de matching des concepts de sorties **Rinking des résultats :** L'algorithme trie les résultats du matching stockés dans la structure match[i] ou chaque résultat possède deux attributs (match.Output ou Gdom-out et match.Input ou Gdom-in).

SortRule (match1, match2)
If match.output > match2.output then match1 > match2
If match1.output=match2.output et match1.input >match2.input then match1 > match2 If match1.output=match2.output et match1.input=match2.input then match1=match

Algorithme 4 : règles de tri des résultats de matching

Le meilleur score de matching est le résultat qui possède l'attribut match.output plus élevé, et dans le cas d'égalité, on fait le recourt au deuxième attribut match.Input comme un critère secondaire pour différencier les résultats de matching.

La procédure du matchmaker : La fonction principale du matchmaker de Paolucci est présentée par l'algorithme 5, et qui a comme argument la requête de l'utilisateur. Cette fonction consiste a balayer la base des services en enregistrement que les services qui peuvent répondre a cette requête (services avec score de matching différent de Fail).

Match (request)
RecordMatch = empty List
Forral adv in advertisement do
If match (request, adv) then

RecordMatch.Append (request, adv) ;
Algorithme 5 : la fonction principale de l'algorithme du matchmaker les approches logiques possèdent plusieurs inconvénients, la complexité élevée du raisonnement affaiblit les chances de scalabilité, en outre les raisonneurs logiques ont un faible rappel (beaucoup de faux négatifs), puisque la subsomption ne couvre pas tous liens sémantique, et en dernier nous constatons que la majorité des services web actuels ne sont pas annotés avec des langages logiques formels.

— **Les approches non logiques** La présence des descriptions informelles ou la sémantique implicite (la fréquence des termes) au niveau des services, peut être efficace dans la découverte, en plus la complexité du matching est moins élevée que celles des approches logiques. En général, ces techniques utilisent le matching de graphes, la linguistique, le data mining, les techniques de recherche d'information, et les mesures de similarité.
Dong proposent un système nommé woogle, il permet de trouver des opérations de services similaires à un prototype donné. Pour cela il utilise plusieurs sources d'évidences. Il propose aussi les services composables avec un service donné.
Le système calcule les similarités entre les descriptions textuelles des opérations de services et entre les identifiants des paramètres d'entrées - sorties. Les auteurs utilisent un algorithme de clustering pour grouper les noms de paramètres, dans des classes sémantiques, qui seront par la suite, utilisées pour calculer la similarité inter paramètres (Inputs/Outputs).
Différents types de similarités sont combinés en utilisant une fonction d'agrégation linéaire, les poids de cette fonction sont assignés manuellement, (après l'analyse des résultats de différentes expérimentations). Les auteurs proposent l'apprentissage des poids en fonction du feed-back des utilisateurs dans les travaux futurs. Cette approche utilise des similarités sémantiques et syntaxiques.
L'algorithme de clustering adopte le principe suivant : « les paramètres expriment souvent le même concept s'ils apparaissent souvent ensemble". Des règles d'association sont définies entre les identifiants des paramètres. La règle d'association entre deux identifiants t1 et t2 est t2 (a, b) ou le support a est la probabilité que le paramètre t1 apparait dans désignée par t1 une entrée ou une sortie, et la confidence b est la probabilité que le paramètre t2 apparait dans une entrée ou une sortie sachant qu'elle contient t1.
Il est utile de noter que ces règles ne sont pas symétriques : t1-t2 et t2-t1 peuvent avoir des valeurs de supports et de confidences différentes. Les auteurs emploient la fusion (merging) et la scission (splitting) pour améliorer le clustering et garantir une meilleure cohésion. Les auteurs utilisent un autre principe, pour éliminer les bruits des concepts. Un terme associé

à un concept est considéré comme bruit, si dans la moitié de ses occurrences il n'est pas associé à d'autres termes du même concept.
Les expérimentations ont confirmé que, la cohésion rigide empêche les grands clusters étroitement associés de se fusionner. En plus la fusion précoce peut être inadéquate car elle empêchera une éventuelle fusion ultérieure.

— **Approches Hybrides** Ces dernières implémentent plusieurs filtres, certains d'entre eux sont purement logiques alors que d'autres se basent sur les techniques de recherche d'information, ou le datamining…, Nous notons que certains filtres compensent la défaillance des autres, et de ce fait un service web sera accepté s'il satisfait au moins un filtre. Nous notons aussi que les scores des filtres logiques sont plus élevés que ceux des filtres à base de techniques de recherche d'information (en termes de pertinence).
- LOG4SWS.KOM est un matchmaker sémantique et hybride, il associe des valeurs numériques aux scores discrets exact, plug-in, subsumes, subsumed-by, intersection et Fail. Pour cela, il utilise la profondeur des relations de subsomption, ce système utilise aussi les similarités à base de wordnet [Fellbaum, 1998] pour comparer les noms des paramètres et des opérations.

Approches Comportementales

Les services atomiques sont généralement composés pour satisfaire les besoins des clients ou des entreprises. Pour rechercher ces compositions (workflows). Les méthodes de découverte doivent prendre en charge la notion de comportement des processus, (ou les cheminements d'exécution), ce comportement doit être modélisé avec des moyens formels tels que, les réseaux de pétri, les automates d'états finis, les processus d'algèbre.
Plusieurs modèles à base d'automates COLOMBO [5] et Romain [6] ont été proposés pour la modélisation du comportement, d'autres tentatives [6] proposent des processus d'algèbre nommés calcul de systèmes communicants (CCS9) [7]. D'autres auteurs appliquent la retro ingénierie pour transformer les interfaces de services (exprimés avec BPEL ou WSDL) en spécifications à base de processus d'algèbre. Ces spécifications seront utilisées, par la suite, pour vérifier des propriétés exprimées en logique temporelle.
Les approches à base d'automates sont présentées avec plus de détail dans la section composition à base de workflow dynamique.

5.4.2 Approches Non Fonctionnelles

(Prenant en Compte la Qualité de Service (QOS))

Approches à Base de Réputation et Confiance

Se basent sur la réputation comme moyen de découverte de services. De façon générale, les auteurs adoptent le feed-back des communautés d'utilisateurs pour sélectionner les services.
Ces approches ont plusieurs inconvénients :

- Pas de modèle précis pour les propriétés de sélection de services
- Ils supposent que les valeurs de propriétés de service, sont facilement calculables, et même ces moyens ne sont pas évalués.
- Les fonctions l'agrégation ne sont pas détaillées.

Les Extensions d'UDDI Certaines approches ajoutent des améliorations au standard UDDI, telles que l'ajout d'un langage de requête structuré (une variante du SQL) , [Dong et al, 2004] proposent l'ajout d'un facilitateur de qualité (broker) qui sépare l'UDDI et l'utilisateur, ce facilitateur gère 03 critères de qualité de service : la fiabilité, la performance, et le coût, et offre 03 scores de matching : gold, silver et bronze.
Le modèle de données de l'uddi est étendu pour prendre en compte la QOS. L'appariement requête-service est fait à travers un calcul d'une corrélation floue.

5.5 Conclusion

Nous avons présenté dans ce chapitre les critères ainsi que les approches existants.

CHAPITRE 6

L’APPROCHE PRPOSÉE

LES SECTIONS

6.1 Introduction 72
6.2 La démarche de notre approche 72
6.3 L'approche SAWSDL Long path 72
6.4 Les étapes de l'algorithme de matching 73
6.5 L'algorithme de matchmaking 73
6.6 La procédure de matching 74
6.7 Le passage au graphe 75
6.8 Le tri des résultats 76
6.9 Discussion des Résultats 77
6.9.1 Le temps de Calcule 77
6.9.2 Les Tableaux des resultats 79
6.10 Outils de travail 80
6.10.1 Le langage java 80
6.10.2 Eclipse 80
6.10.3 Xampp 81
6.10.4 Raisonneur Pellet (version) 81
6.10.5 JavaFX 82
6.11 Expérimentation 83
6.12 Conclusion 87

6.1 Introduction

Dans cette section on présente les motivations de contribution qui porte sur la découverte des services web style SAWSDL.

6.2 La démarche de notre approche

L'idée de la découverte par rapport aux objectifs permet d'afficher une liste de tous les services web appropriés. Chaque service web trouvé est accompagné par sa valeur de similarité par rapport à la requête de l'utilisateur.
Nous supposons qu'il existe un ensemble de services web relatifs à plusieurs différents domaines décrit avec le langage de SAWSDL et publiée dans la base de test SAWSDL-TC.
Du point de vue du demandeur du service, notre système offre une interface graphique conviviale et simple pour l'introduction des besoins des utilisateurs afin de faciliter le processus de découverte.

6.3 L'approche SAWSDL Long path

Notre proposition consiste à améliorer les performances du matchmaking des services atomiques SAWSDL. L'algorithme proposé s'inscrit dans la catégorie des approches logique qui exploitent les propriétés input/output du service SAWSDL.
On propose de représenter le processus de matching par une matrice carré Mn, m et de retenir les même règles d'attribution de degré de correspondance (exact, plugin, subsume, Fail) et de les faire associer a des poids, qui sont des valeurs numérique définis de la façon suivante :

	Poids	
Degré de correspondance	Pondération 1	Pondération 2
Exact	W4=(w3*|M|)+1	W4=(w3+|M|)
Plugin	W3=(w2*|M|)+1	W3=(w2*|M|)+1
Subsume	W2=(w1*|M|)+1	W2=(w1*|M|)+1
Fail	W1=1	W1=1

Tableau 6.1 poids des degrés de correspondance SAWSDL

Ou (M) est la dimension de la matrice M. le matching est représenté par une matrice Mn, m, avec : mij= dom (Ci, Cj) tels que Ci et Cj sont des concepts de l'ontologie de domaine.

6.4 Les étapes de l'algorithme de matching

1 - Calculer la matrice de matching, dans le cas d'une matrice non carrée n, m alors n représente les lignes et m représente les colonnes.

2 - Transformer la matrice de matching Mn, m a un graphe G en respectant les règles suivantes :

- Les sommets sont les mij de la matrice M, les arcs sont organisés par colonne (arc (ci, ci+1)), et il n'est pas autorisé de créer un arc entre la première et la 3eme colonne
- il n'est pas permis de connecter deux sommets de la même ligne ou de la même colonne.
- chaque sommet (élément) dans une colonne est connecté a tous les autres sommets (éléments) de la colonne suivante, si elle existe.
- le sommet source est connecté à tous les sommets de la première colonne
- tous les sommets de la dernière colonne sont connectés au sommet fin.
- les arcs sortants du sommet source sont pondérés par zéro 0
- chaque arc Ai qui relie deux sommets ni et nj est pondéré par la valeur ni ou ni est le nœud source de l'arc Ai, Pij est le poids de la transition entre ni et nj, alors Pij= poids (Ai (ni, nj))= ni.
- on cherche le plus Leng chemin dans G.
- la solution optimale de la matrice de matching M est donnée par les sommets du plus Lang chemin, noté Pi.
- le plus Long chemin est valide lorsque tous leurs sommets ne partagent pas entre eux ni la même ligne ni la même colonne dans Mn, m, il est constitué de nœuds indépendants. π

La valeur retournée est le Gdom_in et Gdom_out.

6.5 L'algorithme de matchmaking

Dans cette section, on présente la fonction de base de l'algorithme de matchmaking de notre approche ainsi que ses fonctions et ses procédures.
Input : Query Q
Output : set of services ranked in descending order, called result
Result=empty
For each service Ai in repository do
If card(Qout)>card(Aout) Then Gdom_out=0//pour garantir que le service satisfait le besoin
Else Gdom_out=Matching_output_concepts (Qout Aout)
If Gdom_out $\neq$ 0 Then if card(Ain)>(Qin) Then Gdom_in=0//pour que je peux garantir la satisfaction du service

Else Gdom_in=Matching_input_concepts (Ain Qin)
If (Gdom_out $\neq$ 0 and Gdom_in $\neq$ 0) and not _exist_fail_in(Gdom_out) Then
Append (A, result)
End-for
Trier la structure result en ordre décroissant, selon l'algorithme de trie proposée par Paolucci.
Return result.

6.6 La procédure de matching

Le matching entre les services et la requête se fait sur deux étapes :
Matching des concepts outputs : consiste à faire le matching entre tous les concepts output de la requête et les concepts outputs du service.
Matching des concepts inputs : consiste à faire le matching entre tous les concepts input de la requête et les concepts inputs du service.

Inputs	Book	Ontologie : http ://127.0.0.1/ontology/books.owl
	User	Ontologie : http ://127.0.0.1/ontology/books.owl
Outputs	Price	Ontologie : http ://127.0.0.1/ontology/concept.owl

Tableau 6.2 les outputs et les inputs du service requête

Inputs	User	Ontologie : http ://127.0.0.1/ontology/books.owl
	Romantic Novel	Ontologie : http ://127.0.0.1/ontology/books.owl
Outputs	Price	Ontologie : http ://127.0.0.1/ontology/concept.owl

Tableau 6.3 les outputs et les inputs du service web

Raisonnement de pellet :
(Book, User)= Fail
(Book, Romantic Novel)= PlugIn
(User, Romantic Novel)= Fail
(User, User)= Exact
La pondération des degrés de matching :

	Poids	
Degré de correspondance	Pondération 1	Pondération 2
Exact	W4=(7*2)+1=15	W4=(7+3)=10
Plugin	W3=(3*2)+1=7	W3=(3*2)+1=7
Subsume	W2=(1*2)+1=3	W2=(1*2)+1=3
Fail	W1=1	W1=1

La matrice est donc :

Pondération 1 :

$$\begin{pmatrix} 7 & 1 \\ 1 & 15 \end{pmatrix}$$

Pondération 2 :

$$\begin{pmatrix} 7 & 1 \\ 1 & 10 \end{pmatrix}$$

6.7 Le passage au graphe

Les valeurs obtenues seront rangés dans une matrice qui sera à son tour transformé à un graphe en respectant plusieurs critères :

- Calculer la matrice de matching dans le cas d'une matrice non carrée n>m alors n représente les lignes et m représente les colonnes.
- Transformer la matrice de matching Mn, m a un graphe G en respectant les réglés suivantes :
  - Les sommets sont les mij de la matrice M, les arcs sont organisés par colonne (arc (ci, ci+1)), et il n'est pas autorisé de créer un arc entre la première et la 3eme colonne.
  - Il n'est pas permis de connecter deux sommets de la même ligne ou de la même colonne.
  - Chaque sommet (élément) dans une colonne est connecté a tous les autres sommets (élément) de la colonne suivante, si elle existe.
  - Le sommet « source » est connecté à tous les sommets de la première colonne.
  - Tous les sommets de la dernière colonne sont connectés au sommet « fin».
  - Les arcs sortants du sommet « source » sont pondérés par zéro.
  - Chaque arc Ai qui relié deux sommets ni et nj est pondéré par la valeur ni ou ni est le nœud source de l'arc Ai. Pij est le poids de la transition entre ni et nj, alors Pij= poids (Ai (ni, nj))=ni.

Le graphe de l'exemple précédant devient donc :

pondération 1 :

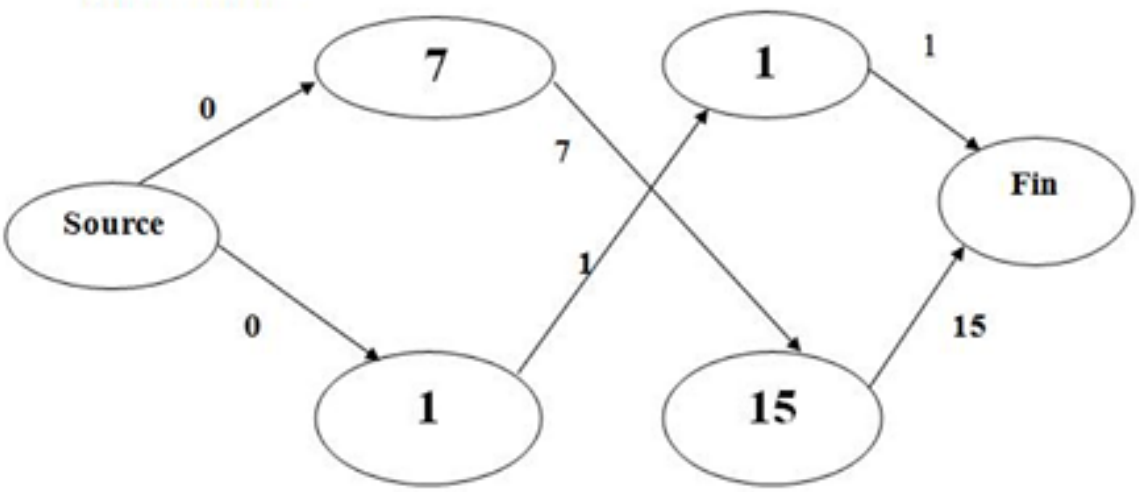


Figure 6.1 le graphe de matching(Pondération1)

pondération 2 :

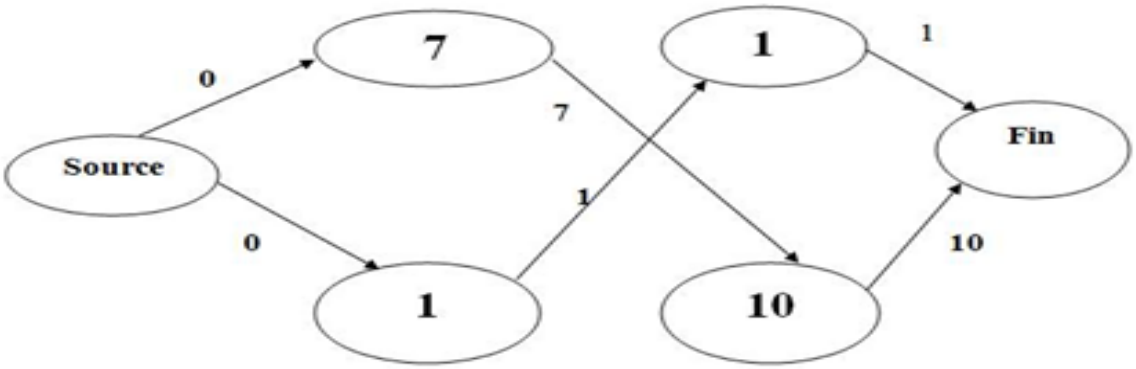


Figure 6.2 le graphe de matching(Pondération2)

6.8 Le tri des résultats

La valeur retournée de score est le résultat final de la découverte et selon cette valeur les services sont donc rangés dans un ordre décroissant. Algorithme de trie de Paolucci :
SortRule (match1, match2)
If match.output > match2.output Then
match1 > match2
If match1.output=match2.output
and match1.input >match2.input Then
match1 > match2
If match1.output=match2.output

and match1.input=match2.input Then
match1=match

Algorithme : règles de tri des résultats de matching

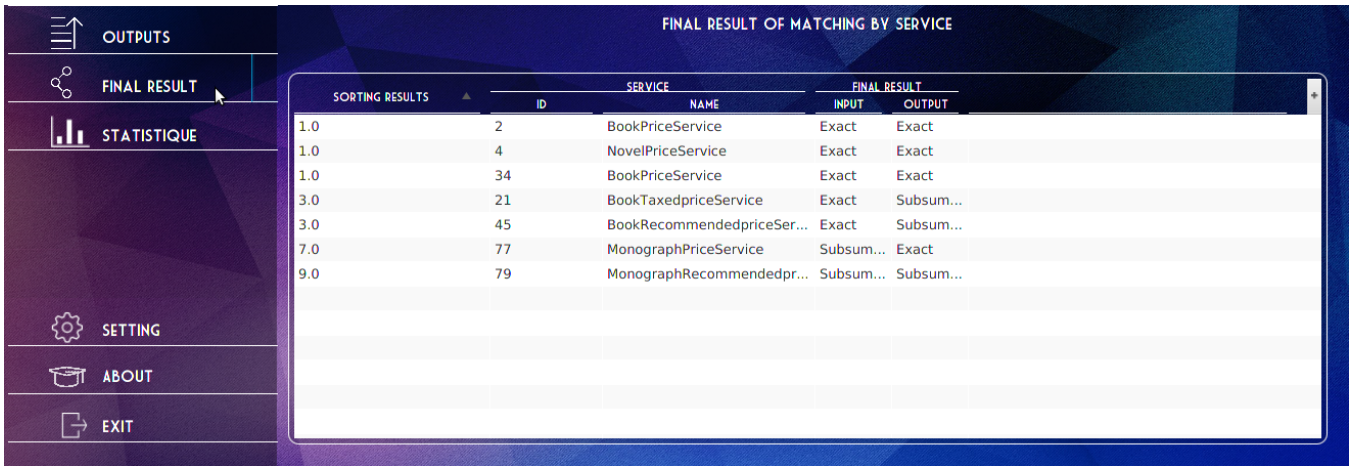


Figure 6.3 Résultat pour book_price_service trier

6.9 Discussion des Résultats

6.9.1 Le temps de Calcule

Caractéristique De Machine

- OS : Linux Ubuntu 14.04 LTS 32 bit
- Processeur Intel Core i3-4010U CPU 1.70GHz x 4
- RAM 1.9 GiB

Les Données de Test

- Data set 95 Service a peut pris de 227 concept et Ontologies
- Le service requèt contient 2 input et 1 output (UserBook-Price-service)

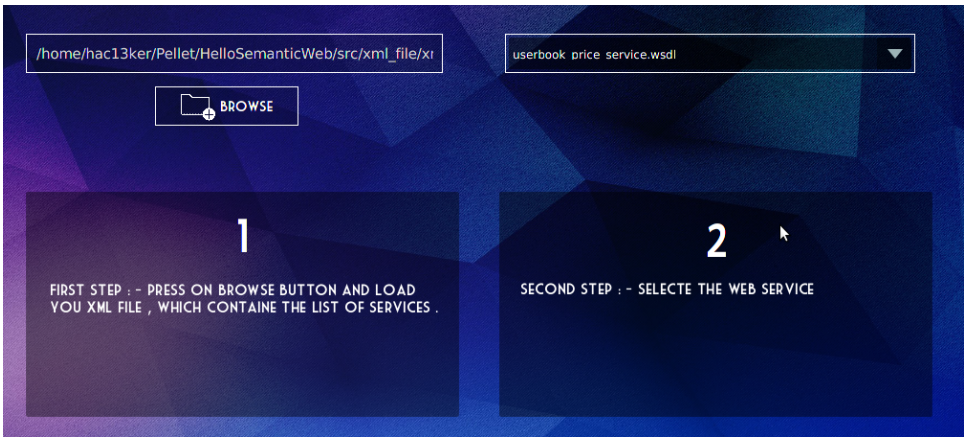


Figure 6.4 la selection de fichier XML contient 95 service et le service requète

Les Temps

— Le temps de calcule avec la pondération 1 (Exact W4=(W3*|M|)+1) = **24 second**

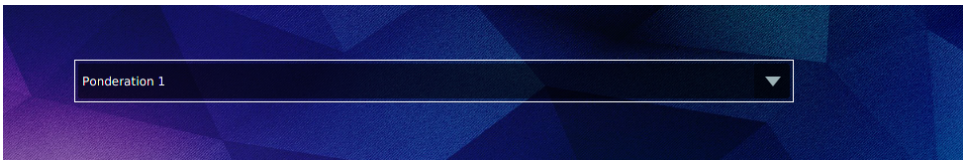


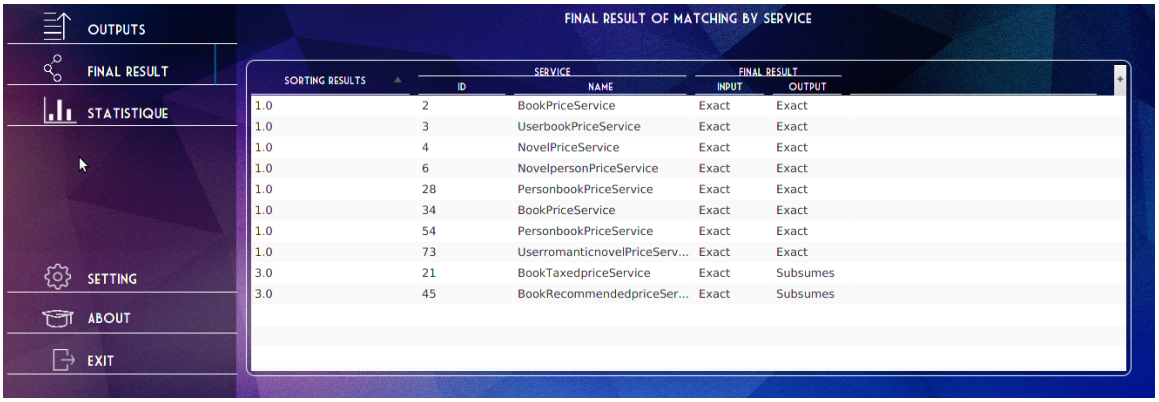
Figure 6.5 La résultat de 95 services / UserBookService / pondération 1 = 24 seconds

— Le temps de calcule avec la pondération 2 (Exact W4=(W3+|M|) = **23 seconds**

Figure 6.6 La résultat de 95 services / UserBookService / pondération 2 = 23 seconds

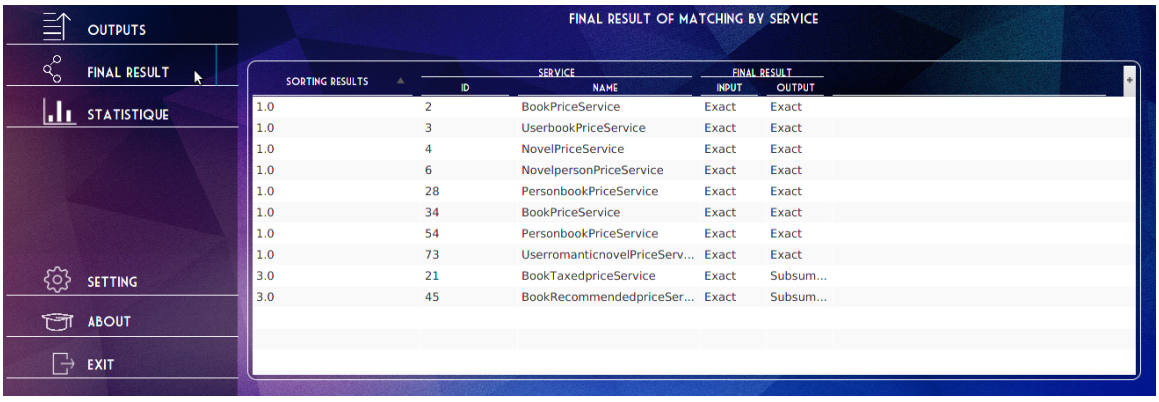
6.9.2 Les Tableaux des resultats

Dans les deux cas on a retenu les mème resultats et le mème Nombre des Resultat



SORTING RESULTS	SERVICE ID	SERVICE NAME	FINAL RESULT INPUT	FINAL RESULT OUTPUT
1.0	2	BookPriceService	Exact	Exact
1.0	3	UserbookPriceService	Exact	Exact
1.0	4	NovelPriceService	Exact	Exact
1.0	6	NovelpersonPriceService	Exact	Exact
1.0	28	PersonbookPriceService	Exact	Exact
1.0	34	BookPriceService	Exact	Exact
1.0	54	PersonbookPriceService	Exact	Exact
1.0	73	UserromanticnovelPriceServ...	Exact	Exact
3.0	21	BookTaxedpriceService	Exact	Subsumes
3.0	45	BookRecommendedpriceSer...	Exact	Subsumes

Figure 6.7 La list des services résultat de 95 services / UserBookService / pondération 1 = 10 services



SORTING RESULTS	SERVICE ID	SERVICE NAME	FINAL RESULT INPUT	FINAL RESULT OUTPUT
1.0	2	BookPriceService	Exact	Exact
1.0	3	UserbookPriceService	Exact	Exact
1.0	4	NovelPriceService	Exact	Exact
1.0	6	NovelpersonPriceService	Exact	Exact
1.0	28	PersonbookPriceService	Exact	Exact
1.0	34	BookPriceService	Exact	Exact
1.0	54	PersonbookPriceService	Exact	Exact
1.0	73	UserromanticnovelPriceServ...	Exact	Exact
3.0	21	BookTaxedpriceService	Exact	Subsum...
3.0	45	BookRecommendedpriceSer...	Exact	Subsum...

Figure 6.8 La list des services résultat de 95 services / UserBookService / pondération 2 = 10 services

Remarque : Remarque La différence entre les deux pondération étre bien clair dans une grande Data set

6.10 Outils de travail

6.10.1 Le langage java

Le langage Java est un langage de programmation informatique orienté objet créé par James Gosling et Patrick Naughton, employés de Sun Microsystems, avec le soutien de Bill Joy (cofondateur de Sun Microsystems en 1982), présenté officiellement le 23 mai 1995 au SunWorld.
La société Sun a été ensuite rachetée en 2009 par la société Oracle qui détient et maintient désormais Java.
La particularité et l'objectif central de Java est que les logiciels écrits dans ce langage doivent être très facilement portables sur plusieurs systèmes d'exploitation tels que UNIX, Windows, Mac OS ou GNU/Linux, avec peu ou pas de modifications. Pour cela, divers plateformes et frameworks associés visent à guider, sinon garantir, cette portabilité des applications développées en Java.

6.10.2 Eclipse

Eclipse est un IDE, Integrated Development Environment (EDI environnement de développement intégré en français), c'est-à-dire un logiciel qui simplifie la programmation en proposant un certain nombre de raccourcis et d'aide à la programmation. Il est développé par IBM, est gratuit et disponible pour la plupart des systèmes d'exploitation.
Au fur et à mesure que vous programmer, eclipse compile automatiquement le code que vous écrivez, en soulignant en rouge ou jaune les problème qu'il décèle. Il souligne en rouge les parties du programme qui ne compilent pas, et en jaune les parties qui compilent mais peuvent éventuellement poser problème (on dit qu'eclipse lève un avertissement, ou warning en anglais). Pendant l'écriture du code, cela peut sembler un peu déroutant au début, puisque tant que la ligne de code n'est pas terminée (en gros jusqu'au point-virgule), eclipse indique une erreur dans le code.
Il est déconseillé de continuer d'écrire le programme quand il contient des erreurs, car eclipse est dans ce cas moins performant pour vous aider à écrire le programme.

6.10.3 Xampp

Est un ensemble de logiciels permettant de mettre en place facilement un serveur Web confidentiel, un serveur FTP et un serveur de messagerie électronique. Il s'agit d'une distribution de logiciels libres (X (cross) Apache MariaDB Perl PHP) offrant une bonne souplesse d'utilisation, réputée pour son installation simple et rapide. Ainsi, il est à la portée d'un grand nombre de personnes puisqu'il ne requiert pas de connaissances particulières et fonctionne, de plus, sur les systèmes d'exploitation les plus répandus. Il est distribué avec différentes bibliothèques logicielles qui élargissent la palette des services de façon notable : OpenSSL, Expat (parseur XML), PNG, SQLite, zlib… ainsi que différents modules Perl et Tomcat. Nombre de ces extensions étant inutiles aux débutants, une version allégée — version lite — est en conséquence aussi proposée.

6.10.4 Raisonneur Pellet (version)

Pellet est un raisonneur open source pour OWL 2 DL en Java. Elle fournit des services de raisonnement standard et de pointe pour les ontologies OWL.

— open source (AGPL) or commercial license

— pure Java
— developed and commercially supported by Complexible Inc.

6.10.5 JavaFX

JavaFX est un outil édité par la société SUN. JavaFX permet de créer de RIA (rich internet application). En effet, JavaFX évite le recours au java dans la mesure où il simplifie la création de contenu graphique. JavaFX possède plusieurs avantages : open source, basé sur java, plateforme universelle...

6.11 Expérimentation

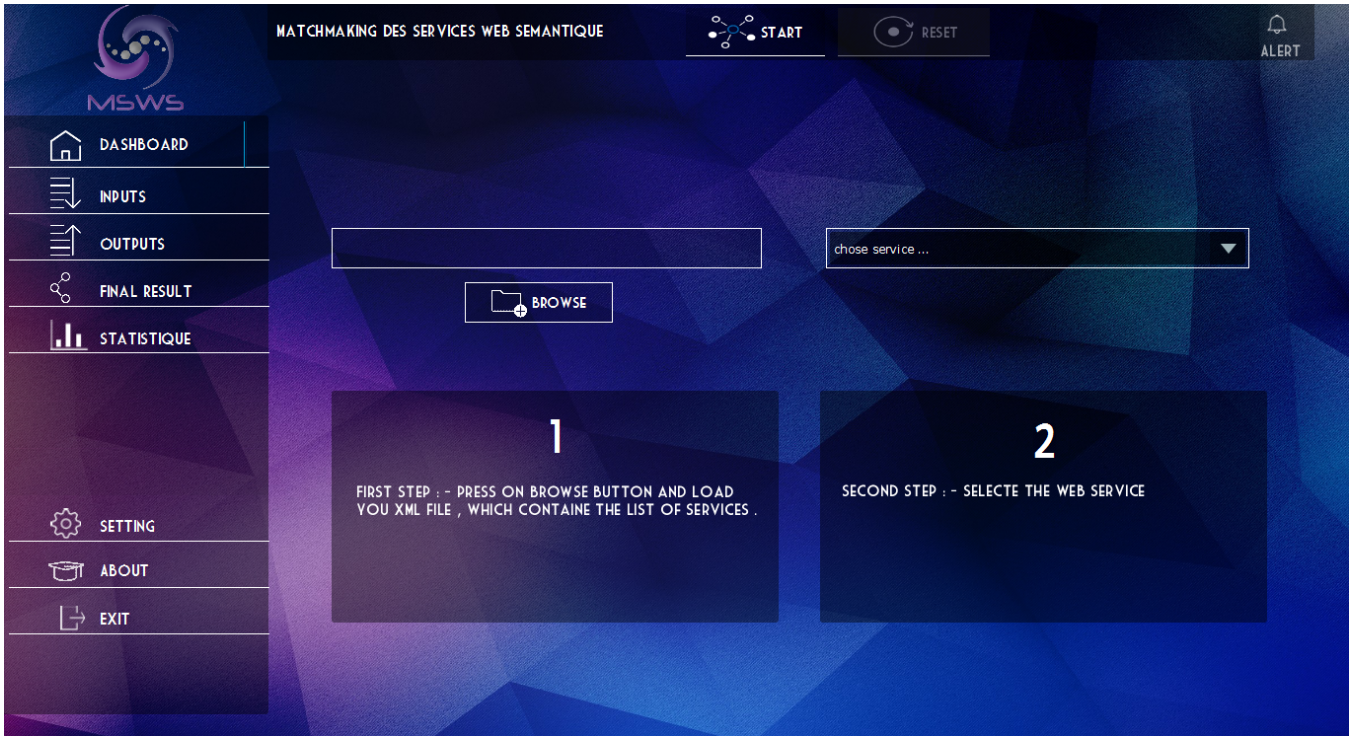


Figure 6.9 L'interface principale de notre application MSWS

Figure 6.10 L'entête d'application pour le lancement , le re-lancement et l'affichage d'etat de progression

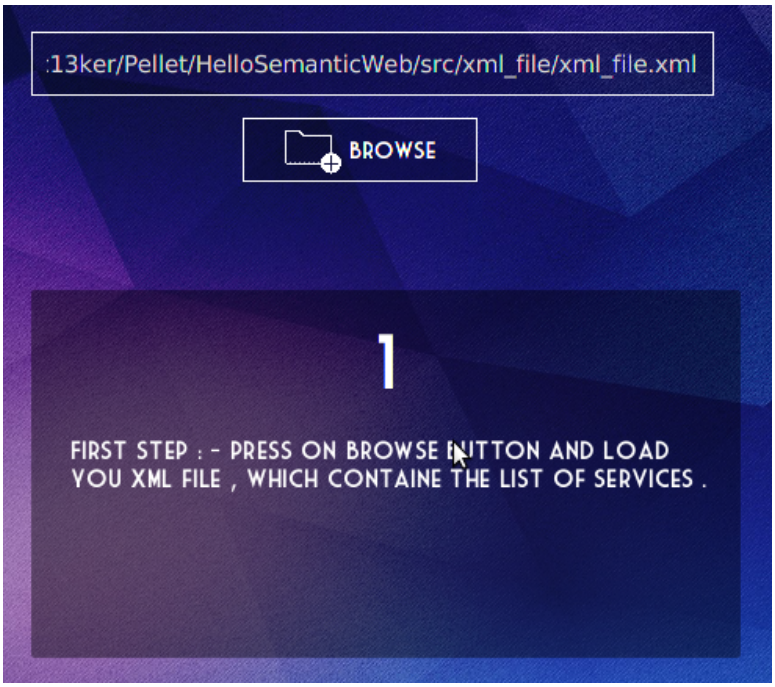


Figure 6.11 Ici tu peux charger le fichier XML qui contient la list des service

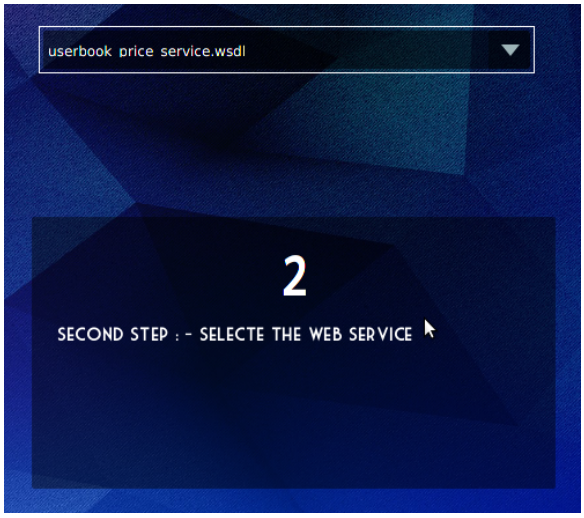


Figure 6.12 Ici tu peux sélectionner la service de requête

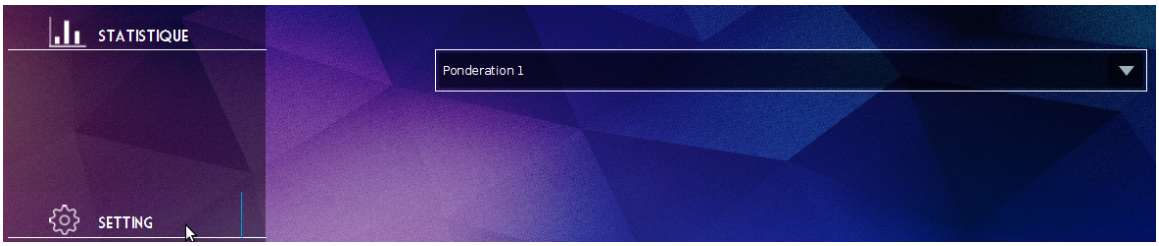
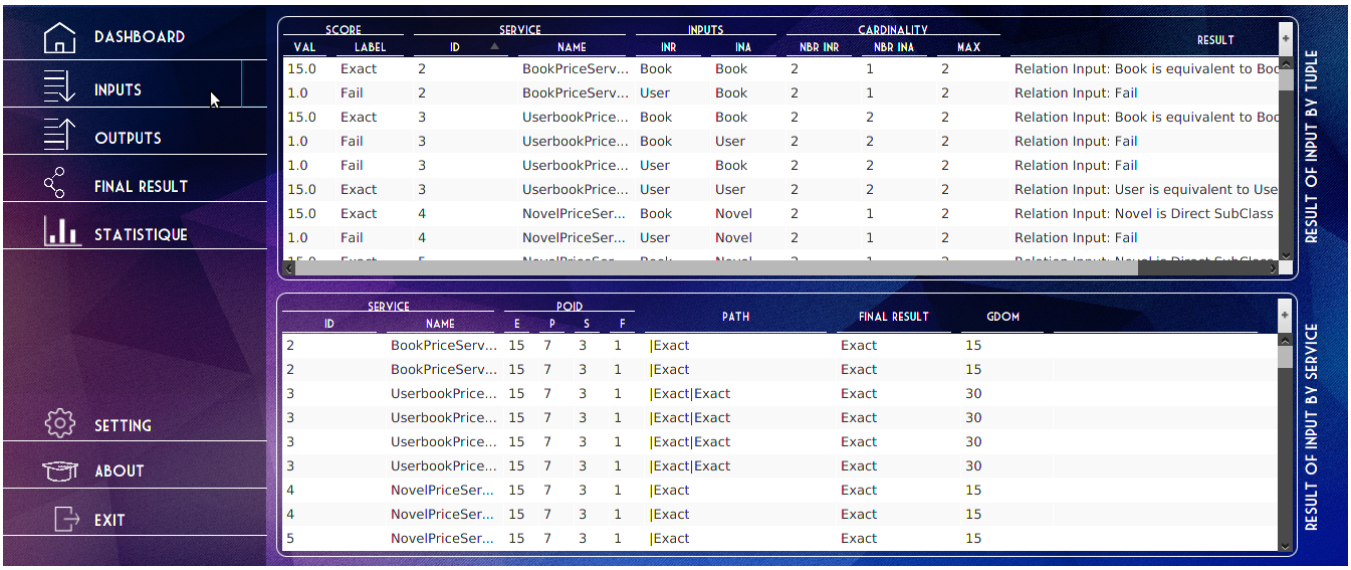


Figure 6.13 Les option d'application , selection de pondération



SCORE		SERVICE		INPUTS		CARDINALITY			RESULT
VAL	LABEL	ID	NAME	INR	INA	NBR INR	NBR INA	MAX	
15.0	Exact	2	BookPriceServ...	Book	Book	2	1	2	Relation Input: Book is equivalent to Boo
1.0	Fail	2	BookPriceServ...	User	Book	2	1	2	Relation Input: Fail
15.0	Exact	3	UserbookPrice...	Book	Book	2	2	2	Relation Input: Book is equivalent to Boo
1.0	Fail	3	UserbookPrice...	Book	User	2	2	2	Relation Input: Fail
1.0	Fail	3	UserbookPrice...	User	Book	2	2	2	Relation Input: Fail
15.0	Exact	3	UserbookPrice...	User	User	2	2	2	Relation Input: User is equivalent to Use
15.0	Exact	4	NovelPriceSer...	Book	Novel	2	1	2	Relation Input: Novel is Direct SubClass
1.0	Fail	4	NovelPriceSer...	User	Novel	2	1	2	Relation Input: Fail

SERVICE		POID				PATH	FINAL RESULT	GDOM
ID	NAME	E	P	S	F			
2	BookPriceServ...	15	7	3	1	|Exact	Exact	15
2	BookPriceServ...	15	7	3	1	|Exact	Exact	15
3	UserbookPrice...	15	7	3	1	|Exact|Exact	Exact	30
3	UserbookPrice...	15	7	3	1	|Exact|Exact	Exact	30
3	UserbookPrice...	15	7	3	1	|Exact|Exact	Exact	30
3	UserbookPrice...	15	7	3	1	|Exact|Exact	Exact	30
4	NovelPriceSer...	15	7	3	1	|Exact	Exact	15
4	NovelPriceSer...	15	7	3	1	|Exact	Exact	15
5	NovelPriceSer...	15	7	3	1	|Exact	Exact	15

Figure 6.14 Résultat de Matching entre les Inputs

Figure 6.15 Résultat de Matching entre les outputs

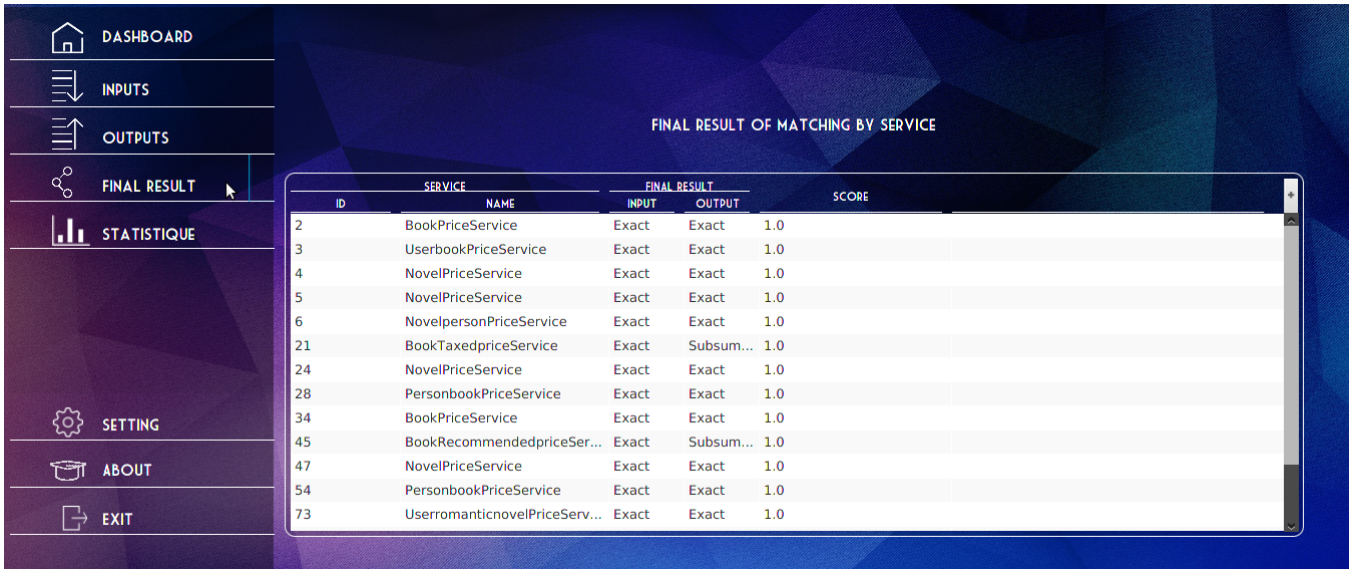


Figure 6.16 Resultat Final des service compatible avec le service requète

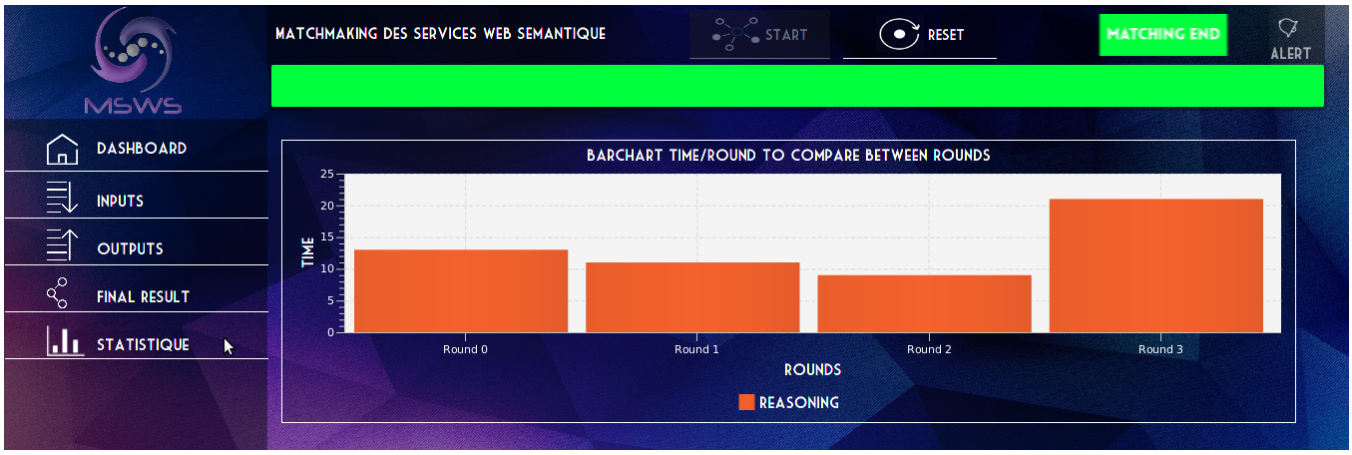


Figure 6.17 charts Bar pour la comparaison entre les Rounds

Figure 6.18 Message de confirmation pour quitter l'application

6.12 Conclusion

Nous avons présenté dans ce chapitre l'implémentation de notre approche de découverte de service web basé sur a partir d'une requête fondée par l'utilisation et comparer cette requête avec les services web disponible en vue de retourner un ou plusieurs services appropriés et répondant aux besoins des utilisateurs.

CHAPITRE 7

CONCLUSION GÉNÉRALE

7.1 Conclusion générale et perspectives

Plusieurs types de paradigmes de développement d'applications distribuées ont été proposés, au cours de ces dernières années. Le paradigme SOA est l'un des modèles les plus convenables au développement d'applications distribuées sur le web. Avec l'immense prolifération des services web sur le réseau internet, les utilisateurs auront toujours besoin de techniques efficaces pour satisfaire les exigences de découverte, de composition et de sélection de services. La première partie de cette thèse est réservée aux travaux d'état de l'art, nous avons mis en évidence les principaux standards liés à la technologie des services web sémantiques, ainsi que leurs avantages et leurs vulnérabilités.la découverte de services web constitue un axe de recherche émergeant. Diverses approches ont été proposées. Ces approches sont passées d'une recherche basée mots clés (correspondance syntaxique de la requête avec les descriptions des services web) aux méthodes basées sémantique (degré de correspondance sémantique de la requête avec la sémantique des descriptions des services web) les travaux de recherche menés sur la description de services web utilisent de plus en plus les ontologies pour fournir une représentation de l'information sémantique. Pour le stockage des services, les architectures d'annuaires de services web n'offrent pas assez de composants dédiés la prise en charge complète de la sémantique. En maître de découverte de services, les approches sémantiques actuelles présentent des lacunes. Les nombreux services disponibles aujourd'hui sur des annuaires purement syntaxiques ou supportant en partie la sémantique continuent d'exposer des problèmes d'imprécision et d'ambiguté par rapport l'appariement sémantique, ce qui inue négativement sur la qualité d'une découverte. La découverte de services web traditionnels consiste apparier syntaxiquement les termes d'une requête une description offerte. L'étude de l'état de l'art nous a permis de mieux comprendre la problématique d'un point de vue système d'annotation, stockage et appariement et la découverte de services web, il y a des manques combler d'une part au niveau de la spécialisation de la nature de l'annotation sémantique dans certaines approches, et d'autre part, dans la prise en charge de la sémantique fonctionnelle dans les mécanismes de l'appariement et/ou dans le calcul des degré d'appariements. Ces manques engendrent ambiguté et imprécision lors d l'appariement et de la découverte. Une des perspectives de notre travail consiste à développer la vérication et l'application de règles et de concepts additionnels dans l'ontologie technique de services web pour permettre une composition efficace de services.

RÉFÉRENCES

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

[2] Contributions à la description et la découverte de, services web Sémantiques Yassin Chabeb, https ://tel.archives-ouvertes.fr/tel-00843597 Submitted on 11 Jul 2013

[3] livre : RESTful Web Services Published by O'Reilly Media, Inc., 1005 Gravenstein Highway North, Sebastopol, CA 95472. May 2007 : First Edition [4] https ://openclassrooms.com/courses/les-services-web. Visité le 2/2016

[5] : gomez,j.m.,rico,m.,garcia-sanchez,f.,bejar,r.m.,bussler,c.,2004.godo :goal driven orchestration for semantic web services. In : 1st workshop on web services modeling ontology implémentations (WIW2004).vol.113.ceur workshop proceedings Gruber t., a translation approache to portable ontology specification,Knowledg acquisition 5(2),pages 199-220,1993.

[6] PDF : Les ontologies chapitre1 page 1-3

[7] cabral,l., domingue,j.,motta,E.,payne,T. ,and hakimpour f.,approaches to semantic web services : An overview and compraisons. In (bussler et al., 2004), pages 225-239,2004

[8] OWL-S Coalition. Bringing Semantics to web services : The OWL-S Approache. In Semantic Web Services and Web Process Composition Workshop, volume 3387 of Lecture Notes in Computer Science, pages 26–42, San Diego, CA, USA, July 2004.

[9] D. B. Claro, P. Albers, and J.-K. Hao. Approaches of web services composition – comparison between BPEL4WS and OWL-S. In International Conference on Enterprise Information Systems (ICEIS 4), pages 208–213, 2005.

[10] R. Lara, D. Roman, A. Polleres, and D. Fensel. A conceptual comparison of WSMO and OWL-S. In Proceedings of the European Conference on Web Services (ECOWS 2004), 11 2004.

[11] M. K. Smith, C. Welty, D. L. McGuinness. OWL Web Ontology Language Guide. http : //www.w3.org/TR/2004/RECowlguide20040210/, Février 2004.

[12] A. Bansal, S. Kona, L. Simon, and T. D. Hite. A Universal service-Semantics description Language. Web Services, European Conference on, 0 :214–225,2005.

[13] Université de Princeton. WordNet. http ://wordnet.princeton.edu/ (Avril 2011).

[14] S. Kona. Universal service-Semantics description Language. Jonson School Forum (JSF), 2006.

[15] L. Simon, A. Bansal, A. Mallya, S. Kona, G. Gupta, and T. D. Hite. Towards a Universal service description Language. Next Generation Web Services Practices, International Conference, pages 175–180, 2005.

[16] S. Kona. Universal service-Semantics description Language. Jonson School Forum (JSF) ,2006.

[17] Large Scale Distributed Information Systems. SAWSDL : Semantic Annotations for WSDL.

http ://lsdis.cs.uga.edu/projects/meteor-s/SAWSDL/ (Avril 2011).[60] H. Lausen and N. Steinmetz. Survey of current means to discover

[18] E. Christensen, F. Curbera, G. Meredith, and S. Weerawarana. Web services description language (WSDL) Version 1.1. Technical report, W3C Note, Mars 2001.

[19] OWL-S Coalition. Bringing semantics to web services : The OWL-S Approach. In Semantic Web Services and Web Process Composition Workshop, volume 3387 of Lecture Notes in Computer Science, pages 26–42, San Diego, CA, USA, July 2004.

[20] De Bruijn J. et al. The web service modeling language WSML. WSML Final Draft, DERI, Octobre 2005.

[21] OMG. Unified Modeling Language (UML) Version 1.5. http ://www.omg.org/technology/documents/formal/uml.htm.

[22] K. Sivashanmugam, K. Verma, A. Sheth, and J. Miller. Adding semantics to web services standards. In International Conference on Web Services (ICWS'03), pages 395–401, Las Vegas, Nevada, June 2003.

[23] : meng,l., huang r. and gu,j.,2013. A review of semantic similarity measures in wornet. International journal of hybrid information technology. Vol. 6, no. 1.