الجمهورية الجزائرية الديمقراطية الشعبية

REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE

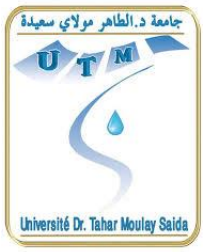
وزارة التعليم العالي و البحث العلمي

MINISTERE DE L'ENSEIGNEMENT SUPERIEUR ET DE LA RECHERCHE SCIENTIFIQUE

Université Dr. Tahar Moulay SAIDA **جامعة د. الطاهر مولاي سعيدة**

Faculté : Technologie **كلية : التكنولوجيا**

Département : Informatique **قسم : الإعلام الآلي**



MEMOIRE DE MASTER

Option : RISR

THEME

Recherche et sélection des services Web

Présenté par :

Awad Samir

Deghab Alaa Eddine

Encadré par :

Dr. Yahlali Mebarka

2015 - 2016

Remerciements

Nous remercions tout d'abord Dieu pour l'accomplissement de ce mémoire.

Ensuite, nous adressons nos sincères remerciements à Dr. Yahlali Mebarka, pour son encadrement, sa disponibilité, ses orientations et sa compréhension.

Nous remercions bien évidemment nos familles, nos parents et nos amis de nous avoir toujours soutenu et à qui nous devons tout.

Enfin nous tenons à remercier tous nos professeurs du département d'informatique pour leurs enseignements.

Dédicaces

Nous dédions ce modeste travail à

nos très chers parents,

que dieu les récompense et les garde.

Tous nos amis.

Nos collègues de promotion.

Et a tous ceux qui nous connaissent de près ou de loin.

Merci d'être toujours là pour nous.

ملخص

يزداد عدد خدمات الويب بسرعة كبيرة، وبالتالي خلق تحديات كبيرة في كيفية استخراج الخدمات المستهدفة من طرف المستخدمين بطريقة دقيقة، فعالة وآلية. تقترح هذه الأطروحة بنية واستراتيجية لاستخراج واختيار خدمات الويب.

تسمح عملية البحث المقترحة باستخراج كل الخدمات التي من المحتمل أن تلبي احتياجات المستخدم (الجانب الوظيفي) مع مساعدته على الاختيار بالاعتماد على الخصائص الغير وظيفية.

الكلمات المفتاحية : خدمة الويب، استخراج واختيار خدمات الويب، الويب الدلالي، أنتولوجيا، جودة الخدمة.

Résumé

Le nombre des services Web croit à une vitesse explosive, créant ainsi, de grands défis dans l'extraction exacte, efficace et automatique de services cibles pour les utilisateurs. Ce mémoire propose une architecture et une stratégie pour la recherche et la sélection des services Web.

La procédure de recherche proposée permet d'obtenir tous les services potentiels correspondant aux besoins d'un utilisateur (aspect fonctionnel) avec une sélection sur la base de propriétés non fonctionnelles.

Mots clés : Service Web, recherche et sélection des services Web, Web sémantique, ontologie, qualité de service.

Abstract

The number of web services is growing at an explosive speed which brings great challenges to the accurate, efficient and automatic retrieval services for users. This thesis proposes an architecture and a strategy for research and selection of Web services.

The proposed procedure of research provides all potential services that meet a user's needs (functional aspect), with a selection on the basis of non functional properties.

Keywords : Web service, discovery and selection of web services, semantic web, ontology, quality of service.

Table des matières

Introduction générale

Une architecture orientée service (Service Oriented Architecture, ou SOA) est une architecture logicielle visant à mettre en place un système d'information constitué de services applicatifs indépendants et interconnectés. L'architecture SOA permet la réutilisation des applications et des services existants, ainsi de nouveaux services peuvent être créés à partir d'une infrastructure informatique des systèmes déjà existante. En d'autres termes, SOA permet aux entreprises de tirer partie des investissements existants en leur permettant de réutiliser des applications existantes, en leur offrant une interopérabilité entre applications et technologies hétérogènes. L'objectif d'une architecture SOA est de décomposer une fonctionnalité en un ensemble de services et de décrire leurs interactions.

Dans une architecture SOA, les fournisseurs de services publient (ou enregistrent) leurs services dans un annuaire de services (qui peut être publique ou local à un réseau d'entreprise par exemple). Cet annuaire est utilisé par les utilisateurs pour trouver des services vérifiant certains critères ou correspondant à une certaine description. Si l'annuaire contient de tels services, il envoie au client les descriptions de ces services avec un contrat d'utilisation. Le client fait alors son choix, s'adresse au fournisseur et invoque le service [1].

Les services Web sont largement utilisés dans le cadre des architectures SOA, étant donné qu'ils peuvent être combinés pour réaliser de nouveaux processus (ou services) plus rapidement qu'un développement traditionnel. Les services Web sont la réalisation (ou l'instance) la plus importante d'une architecture SOA. Ils peuvent faire appel à différents services et peuvent être eux mêmes déployés en tant que nouveaux services.

Problématique

Un des problèmes majeurs qui peuvent faire face aux utilisateurs est le problème de recherche et sélection des services Web. Ainsi, trouver le "bon" service à invoquer, répondant à une requête d'un utilisateur, suppose avoir découvert un ensemble de ser-

vices potentiels. Puis, à partir de cet ensemble, il s'agit de sélectionner le service le "plus pertinent" pour le proposer à l'utilisateur. Plusieurs facteurs peuvent venir compliquer cette recherche, comme par exemple :
- Le choix du format de description du besoin : c'est le premier problème qui se pose lors d'une automatisation de la recherche et de la sélection.
- Le grand nombre possible de services renvoyés, suite à une requête, complique la tâche de sélection pour un utilisateur. Cette tâche est rendue encore plus complexe par l'utilisation de deux catégories de critères de sélection d'un service : ses aspects fonctionnels (ce que fait le service) et non fonctionnels (comment il le fait).

Objectif et contribution

L'objectif de ce travail est de :

1. Développer un système de recherche et sélection des services Web.
2. Proposer une extension au niveau des descriptions des services Web en terme de paramètres de qualité de service.

Organisation du mémoire

Ce manuscrit est structuré comme suit :

1. Le premier chapitre est consacré aux notions fondamentales des services Web où nous avons défini les standards de description, de publication et d'invocation des services Web. Par la suite, nous introduisons le Web sémantique pour définir les services Web sémantiques.
2. Le deuxième chapitre est consacré à un état de l'art sur les approches et méthodes proposées dans la littérature pour la résolution du problème de découverte des services Web.
3. Le troisième chapitre est consacré à un état de l'art sur les approches et méthodes proposées dans la littérature pour la résolution du problème de sélection des services Web.
4. Le quatrième chapitre présente la mise en œuvre de notre système. Nous montrons également des expérimentations pour évaluer la performance des algorithmes implémentés.
5. Enfin, nous terminerons notre mémoire par une conclusion générale et nous énoncerons quelques perspectives de recherche.

Chapitre 1

Les services Web sémantiques

1.1 Introduction

Les services Web sémantiques se situent à la convergence de deux domaines de recherche importants : le Web sémantique et les services Web. Le Web sémantique s'intéresse principalement aux informations disponibles sur le Web et les moyens de les décrire de manière intelligible pour les machines. Les services Web, quant à eux, ont pour préoccupation première l'interopérabilité entre applications [2].

1.2 Définition d'un service Web

On retrouve plusieurs définitions des services Web. Nous en retenons celle du W3C (World Wide Web Consortium) [3] :

« Un service Web est un composant logiciel identifié par un URI, dont les interfaces publiques sont définies et appelées en XML. Sa définition peut être découverte par d'autres systèmes logiciels. Les services Web peuvent interagir entre eux d'une manière prescrite par leurs définitions, en utilisant des messages XML portés par les protocoles Internet. »

1.3 Les caractéristiques d'un service Web

Les Web services possèdent les caractéristiques suivantes qui leur permettent une meilleure intégration dans les environnements hétérogènes [4] :

- Reconnue par un URI : Un service Web est une application logicielle qui est reconnue par un URI.

- Basé sur XML : Les données dans les protocoles et les technologies des Web services sont représentées en utilisant XML, ces technologies peuvent être interopérables. Comme un transport de données, XML élimine toute dépendance de gestion de réseau, du système d'exploitation, ou de la plateforme liée à un protocole.
- Faiblement couplés : Les Web services sont autonomes et peuvent fonctionner indépendamment les uns des autres. Il n'est pas nécessaire de connaître la machine, le langage ou le système d'exploitation.
- Auto-descriptif : Les Web services ont la capacité de se décrire d'une manière qui peut être facilement reconnu. Ainsi, l'interface, les informations de localisation et l'accès au Web service est identifié par n'importe quelle application externe.
- Modulaire : Les Web services fonctionnent de manière modulaire. Cela signifie qu'au lieu d'intégrer dans une seule application globale toutes les fonctionnalités, on crée plusieurs applications spécifiques et on les fait inter-opérer entre elles, et qui définissent chacune, une de ses fonctionnalités.
- Réutilisable : Une fonctionnalité, développée sous forme de Web service, peut être réutilisée et combinée à d'autres fonctionnalités afin de composer de nouveaux services.

1.4 Les technologies des services Web

Le concept des services Web s'articule autour les trois standards :
SOAP (Simple Object Access Protocol), WSDL (Web Services Description Language) et UDDI (Universal Description, Discovery and Integration).

1.4.1 SOAP

SOAP est un protocole léger destiné à l'échange d'informations structurées dans un environnement distribué. Il utilise des technologies XML pour définir une structure d'échange de messages fournissant une construction de messages pouvant être échangés sur divers protocoles sous-jacents. La structure a été conçue pour être indépendante de tout modèle de programmation.

Plus en détail, la spécification du SOAP ne définit rien d'autre qu'une enveloppe contenant les informations à transmettre, et un ensemble de règles pour la transformation des types des données spécifiques aux applications et aux plates-formes en représentation XML [5].

1.4.1.1 Structure d'un message SOAP

Un message SOAP est composé de deux parties obligatoires : l'enveloppe SOAP et le corps SOAP et une partie optionnelle : l'en-tête SOAP.

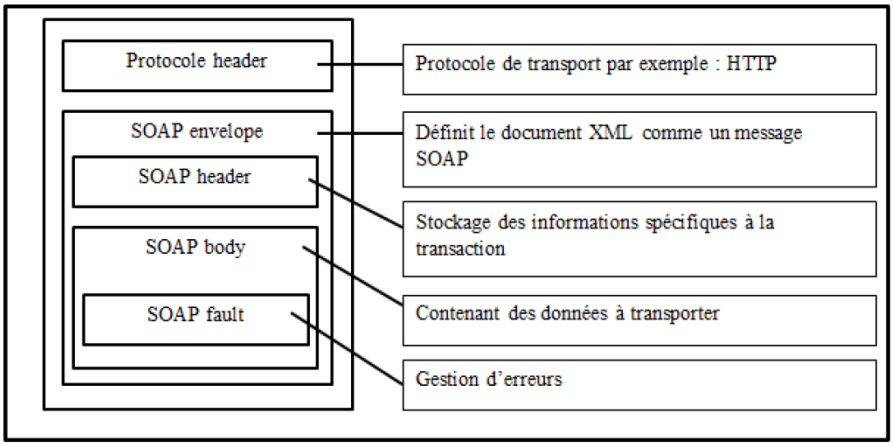


FIGURE 1.1 – Structure d'un message SOAP [6]

1.4.2 WSDL

WSDL est un langage de la famille XML permettant de décrire les types de données supportées et les fonctions offertes par un service Web. L'objectif est de fournir la description, en XML, des services indépendamment de la plateforme et du langage utilisés et sous une forme que des personnes ou des programmes peuvent interpréter.

WSDL a été créé dans le but de fournir une description unifiée des services Web. Il décrit précisément quelles méthodes du service sont accessibles et quels sont les types de leurs paramètres [7].

1.4.2.1 Structure d'un document WSDL

Le document WSDL peut être divisé en deux groupes de sections. Le groupe du haut est constitué des définitions abstraites, tandis que le groupe du bas contient les descriptions concrètes :

- Définition abstraite : Composée des éléments qui sont orientés vers la description des capacités du service Web. Ses éléments abstraits définissent les messages SOAP de façon indépendante de la plateforme et de la langue. Les quatre éléments qui peuvent être définis dans un WSDL sont :

a) Types : Décrit les types des données utilisées par le service.
b) Message : Décrit les noms et les types d'un ensemble de champs à transmettre. Il peut être comparé aux paramètres d'un appel de procédure.
c) portType (type de port) : Décrit le service Web, les actions qu'il peut exécuter et les messages qui lui sont associés. Il peut être comparé aux prototypes des fonctions dans un langage de programmation traditionnel.
d) Opération : Décrit les opérations invoquées de manière distante sur le service Web, chaque opération est décrite à l'aide de deux ou trois messages :
- Le message reçu par le service Web lorsqu'il est sollicité par une requête.
- Le message émis en réponse par le service Web.
- L'éventuel message retourné par le service Web en cas de problèmes.

- Descriptions concrètes : La description concrète est composée des éléments orientés vers le client pour le service physique. Les trois éléments concrets XML présents dans un WSDL sont :
a) binding (liaison) : Définit le format des messages et le protocole utilisé par chaque type de port.
b) Service : Il s'agit de l'ensemble des ports exposés pour permettre l'accès aux services correspondants.
c) Port : Représente un point de terminaison identifié de manière unique par la combinaison d'une adresse Internet et d'une liaison.

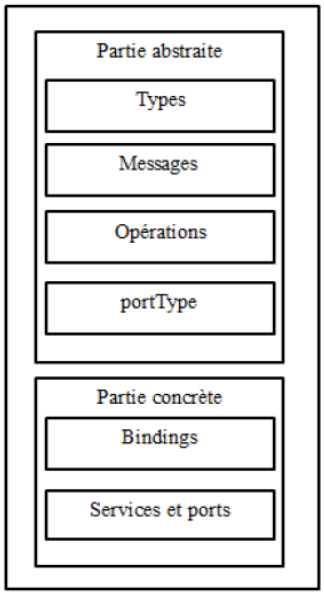


FIGURE 1.2 – Structure d'un document WSDL [7]

1.4.3 UDDI

Un service Web doit être référencé afin de pouvoir être retrouvé et utilisé par une autre organisation. Pour cela, il existe des annuaires pouvant être soit internes à l'organisation, soit universels.

L'annuaire UDDI définit les mécanismes permettant de répertorier des services Web. Ce standard gère donc, l'information relative à la publication, la découverte et l'utilisation d'un service Web. Les organisations publient les informations décrivant leurs services Web dans l'annuaire, et l'application client ayant besoin d'un certain service, consulte cet annuaire pour la recherche des informations concernant le service Web désiré, pour une éventuelle interaction, ainsi l'UDDI a été créé pour faciliter la découverte de services Web en plus de leurs publications. L'UDDI peut être vu comme un annuaire contenant [8] :

a) Les pages blanches : Noms, adresses, identifiants et contacts des entreprises enregistrées. Cette description inclut des informations de catégorisation permettant de faire des recherches spécifiques dépendant du métier de l'entreprise.

b) Les pages jaunes : Donnent les détails sur le métier des entreprises et les services qu'elles proposent. Ces informations sont décrites dans des entités de type "BusinessService".

c) Les pages vertes : Contiennent des informations techniques du service offert.

La structure de données du registre UDDI contient quatre types de données :

a) BusinessEntity : Les informations concernant l'entreprise qui publie ses services Web dans l'annuaire et le type de services qu'elle offre sont contenues dans la structure "BusinessEntity" et correspondent notamment aux pages blanches concernant l'entreprise.

b) BusinessService : L'élément "BusinessService" contient des informations sur les services métiers. Il s'agit des informations de description des services Web référencés : nom du service, sa description...etc. Une entreprise peut enregistrer plusieurs services. Le type d'informations contenu dans l'élément "BusinessService" correspond aux informations pages jaunes d'une entreprise.

c) BindingTemplate : Les informations de liaison contiennent des informations techniques sur un service Web. Elles servent également de pages vertes de l'annuaire UDDI. Il s'agit des informations indiquant les références aux "tModels" désignant les spécifications de l'interface pour un service Web, ainsi le point de terminaison (adresse Internet) de ce dernier. Ces informations aident le client à se connecter puis à invoquer le service désiré.

d) tModel : Les "tModels" sont les descriptions techniques des services. UDDI n'impose aucun format pour ces descriptions qui peuvent être publiées sous n'importe quelle forme et notamment sous forme de documents textuels (XHTML, par exemple). C'est à ce niveau que WSDL intervient comme le vocabulaire de choix pour publier des descriptions techniques de services.

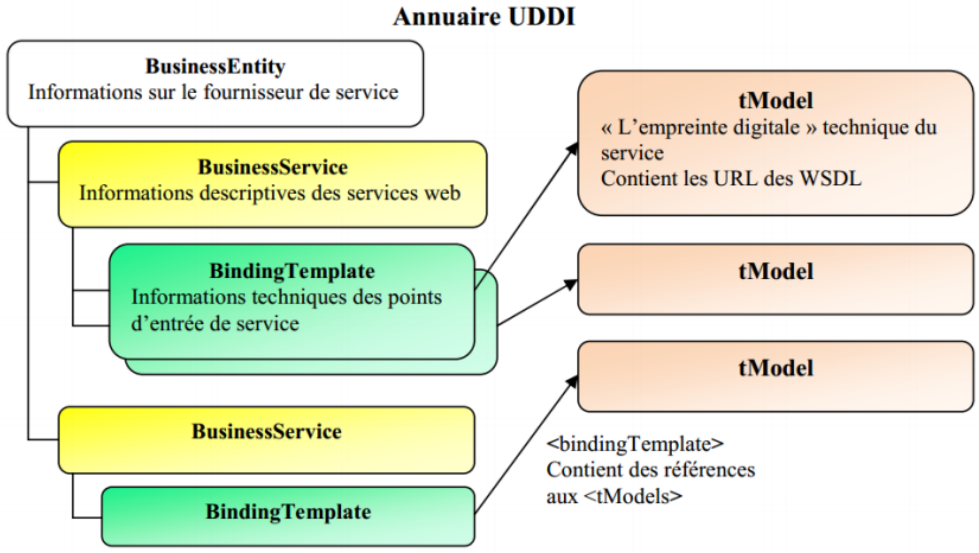


FIGURE 1.3 – Structure de données de l'annuaire UDDI [9]

1.5 L'architecture d'un service Web

1.5.1 Architecture de référence

L'architecture de référence se base sur les trois concepts suivants [10] :

- Le serveur (ou le fournisseur de service) : C'est le propriétaire du service.

- Le client (ou le consommateur de service) : C'est un demandeur de service. D'un point de vue technique, il est constitué par l'application qui va rechercher et invoquer un service.

- L'annuaire des services : C'est un registre de descriptions de services offrant des facilités de publication de services à l'intention des fournisseurs ainsi que des facilités de recherche de services à l'intention des clients.

Les interactions de base qui existent entre ces trois éléments sont les opérations de

publication, de recherche, d'invocation et de lien (bind) :
- Le fournisseur de service crée le service Web, puis publie son interface ainsi que les informations d'accès au service dans un annuaire de services Web.
- L'annuaire de service rend disponible l'interface du service ainsi que ses informations d'accès pour n'importe quel demandeur potentiel de service.
- Le consommateur de service accède à l'annuaire de service pour effectuer une recherche afin de trouver les services désirés. Ensuite, il se lie au fournisseur pour invoquer le service.

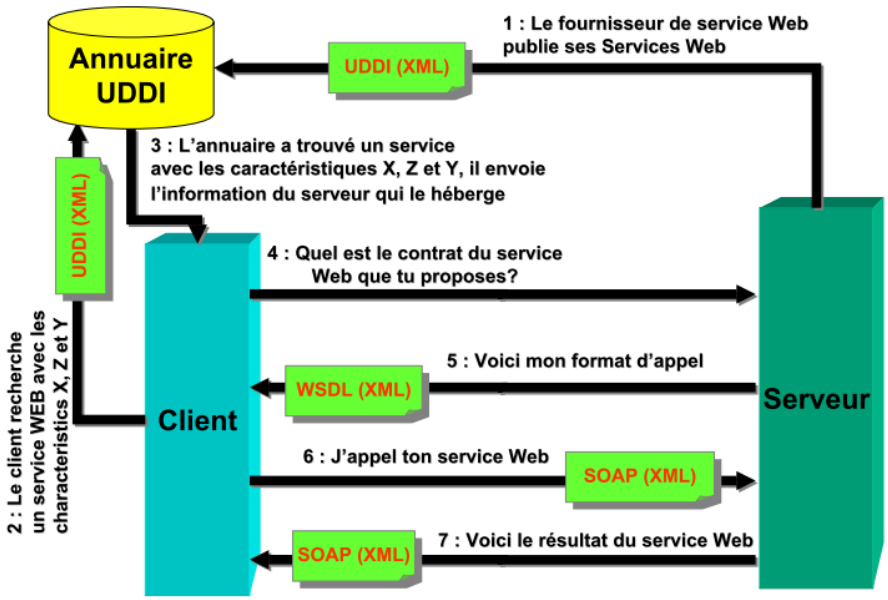


FIGURE 1.4 – Architecture de référence des services Web [11]

1.5.2 Architecture étendue

Cette architecture étendue est aussi appelée pile des Web services, du fait qu'elle est constituée de plusieurs couches se superposant les unes aux autres. Elle utilise les couches standards de la première architecture en ajoutant au-dessus d'autres couches spécifiques [4].

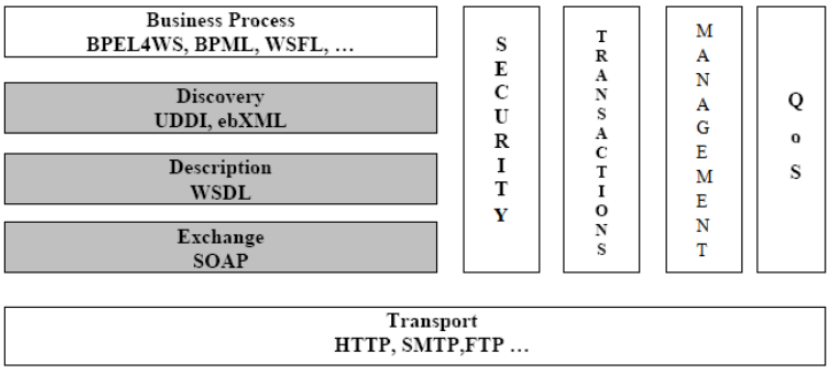


FIGURE 1.5 – Architecture en pile (étendue) [4]

La pile est constituée de plusieurs couches, chaque couche s'appuie sur un standard particulier. On retrouve, au dessus de la couche de transport, les trois couches formant l'infrastructure de base décrite précédemment.

Cette architecture peut être décomposée en trois types de couches :

- L'infrastructure de base : Elle est constituée de trois couches, ces couches s'appuient sur les standards des Web services (SOAP, WSDL, UDDI). Elle définit le fondement technique de l'architecture de référence.

- Les couches transversales (sécurité, administration, . . .) : Ce sont les couches qui rendent viable l'utilisation effective des Web services dans le monde industriel et dont on trouve toute les notions associées aux problématiques de sécurité et celles en rapport avec la QoS (Quality of Service). La partie administration est un peu particulière, car il s'agit de mettre en place des enchainements de services et par conséquent de créer des Web services composites.

- Business Process : Cette couche permet l'intégration de Web services, elle établit la représentation d'un Business Process comme un ensemble de Web services. De plus, la description de l'utilisation de différents services composants ce service est disponible par l'intermédiaire de cette couche.

1.6 Web sémantique

Le Web d'aujourd'hui est essentiellement syntaxique, dans le sens où la structure des documents, ou ressources au sens large, est bien définie mais que leur contenu sémantique reste quasi inaccessible aux traitements machines.

L'objectif du Web sémantique est de lever les difficultés rencontrées sur le Web

d'aujourd'hui (recherche d'information, services,...), en rendant la grande masse d'information disponible, accessible et interprétable par les machines, en plus de l'automatisation de certaines fonctionnalités grâce à la représentation sémantique du contenu des documents, des services, des ressources sur le Web au sens large.

Une représentation explicite de la sémantique des données, programmes, pages html, services Web et autres ressources du Web permettra d'avoir un Web qui fournira à son tour un niveau de service qualitativement meilleur.

Pour réaliser cette représentation, les ontologies ont été introduites. Les ontologies représentent la technologie clé pour la réalisation du Web sémantique [12].

1.6.1 Les ontologies

Plusieurs définitions ont été proposées dans la littérature. La définition la plus adoptée est celle donnée par Gruber [13] : « Une ontologie est une spécification explicite d'une conceptualisation. »

En 1997, Borst [14] modifia légèrement la définition de Gruber en énonçant que : « Une ontologie est définie comme étant une spécification formelle d'une conceptualisation partagée. »

Ces deux définitions ont été expliquées par Studer [15] et ses collègues comme suit :
- Conceptualisation : Réfère à un modèle abstrait d'un phénomène dans le monde, en ayant identifié les concepts appropriés à ce phénomène.
- Explicite : Signifie que le type de concepts utilisés et les contraintes liées à leur usage sont définis explicitement.
- Formelle : Réfère au fait que l'ontologie doit être traduite en langage interprétable par une machine.
- Partagée : Réfère au fait qu'une ontologie capture la connaissance consensuelle, c'est-à-dire non réservée à quelques individus, mais partagée par un groupe ou une communauté.

Une ontologie définit les termes basiques et les relations entre ces derniers. Elle regroupe le vocabulaire d'un domaine ainsi que les règles qui combinent ces termes et ces relations afin de définir une extension de ce vocabulaire.

Dans le cas du Web sémantique, une ontologie permet à l'utilisateur, lors d'une recherche sur le Web, d'accéder non seulement aux documents liés aux mots clés de la requête, mais aussi à ceux qui sont liés ontologiquement (sémantiquement) à ces derniers, ce qui rend la recherche encore plus pertinente. Une ontologie est souvent représentée en terme de classes, relations, propriétés, attribut et valeurs [12].

1.6.2 Structure du Web sémantique

La figure suivante montre l'organisation en couches du Web sémantique, proposé par le W3C [16] :

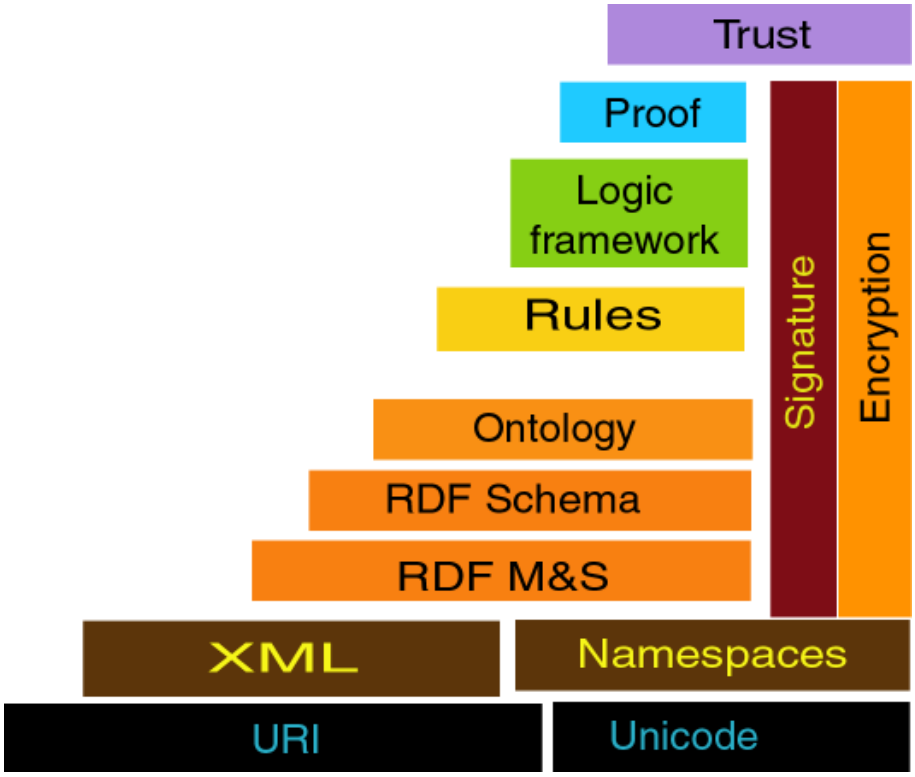


FIGURE 1.6 – Structure du Web sémantique [16]

Le World Wide Web repose sur un concept important qu'est l'URI. Tout ce qui est disponible sur Internet doit être identifié par un URI. Un URI identifie de manière unique et non ambiguë chaque ressource du Web, comme une page, une adresse email, ou une image.

Le premier des langages sémantiques est RDF(Ressource Description Framework) auquel s'est ajouté rapidement RDF Schema. Les objectifs initiaux de RDF étaient la représentation et une meilleure exploitation des méta-données. De manière générale, RDF permet de voir le Web comme un ensemble de ressources reliées par les liens étiquetés sémantiquement. RDFS est une extension de RDF permettant de décrire les ressources dans le contexte du Web sémantique.

La couche suivante évoquée par le W3C propose un langage standard pour la représentation et l'utilisation d'ontologies, appelé OWL (Web Ontology Language). OWL a été recommandé par le W3C afin d'enrichir RDFS en définissant un vocabulaire plus

complet pour la description d'ontologies complexes.

L'architecture du Web sémantique inclut aussi des couches supérieures (rules, proof, trust, signature, encryption). La possibilité de faire des déductions plus complètes pourra s'appuyer sur la standardisation d'un langage de règles, comme RuleML (Rule Markup Langauge). Le problème de confiance doit reposer aussi sur des méthodes de qualification de l'origine de l'information, par exemple par des métadonnées et des annotations, éventuellement certifiées par des signatures électroniques.

1.7 Les services Web sémantiques

Les services Web sémantiques sont la convergence de la technologie de services Web et du Web sémantique, ce sont des services Web à descriptions dotées de sémantique. Actuellement les services Web sont décrits par le langage WSDL qui permet de définir les opérations et les paramètres autorisés par le service Web. Cela est fait en attribuant des noms aux opérations et aux paramètres, puis associer ces paramètres à des types de données (exemple : string, char,...).

Cependant, le problème avec ce type de description est que lorsqu'on veut automatiser les divers aspects liés aux services Web (découverte, composition,...), l'agent manipulant les paramètres du service Web ne peut pas avoir la signification de ces données.

A ce stade, les services Web ne sont décrits qu'au niveau syntaxique. Un développeur voulant programmer une application cliente interagissant avec un service Web doit tout d'abord avoir connaissance de la syntaxe de sa description, l'interpréter, puis écrire le code client conforme aux paramètres de la description du service Web. Cependant, un agent logiciel ne peut lire la description d'un service Web comme un humain, il peut avoir connaissance de la structure syntaxique de la description mais pas sa sémantique.

Les services Web sémantiques sont des services Web décrits de telle sorte qu'un agent logiciel puisse interpréter les fonctionnalités offertes par le service Web. Un agent logiciel doit être 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 façon qu'un agent logiciel puisse avoir connaissance de leur signification [12].

1.7.1 Description sémantique des services Web

Deux méthodes possibles pour la description sémantique des services Web :

La première approche consiste à annoter les langages existants (WSDL) avec l'information sémantique. Le principal avantage de ce genre de solutions est la facilité pour les fournisseurs de services d'adapter leurs descriptions existantes aux annotations proposées. Exemple : SAWSDL (Semantic Annotations for WSDL and XML Schema).

La deuxième approche réécrit entièrement la description syntaxique des services Web en utilisant les technologies du Web sémantique. Exemple : OWL-S (Ontology Web Language for Services).

1.7.1.1 SAWSDL

SAWSDL 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. D'une part SAWSDL, fournit un mécanisme permettant d'annoter sémantiquement les types de données, les opérations, les entrées et les sorties de WSDL et d'autre part, il ajoute des éléments pour spécifier les pré-conditions, les effets et les catégories des services Web. Les aspects relatifs à la qualité des services ne sont pas traités dans SAWSDL [17].

1.7.1.2 OWL-S

OWL-S est un sous ensemble du langage OWL dédié à la description sémantique de Web services. Il permet de décrire d'une façon non ambiguë les Web services de telle sorte qu'un agent logiciel puisse exploiter automatiquement ces informations. Une description OWL-S se compose de trois éléments : le service profile, le service model, et le service grounding, qui décrivent respectivement que fait le service, comment le service fonctionne et comment accéder au service [18].

1.8 Conclusion

Les services Web sémantiques 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 utilisées pour palier à ce problème en enrichissant les services Web de sémantique, ce qui permet l'automatisation des divers aspects relatifs aux services Web.

Chapitre 2

La découverte des services Web

2.1 Introduction

L'un des problèmes les plus importants dans l'architecture des services Web est le problème de découverte. La découverte des services Web est le fait de mettre en correspondance deux entités tel que l'une offre ce que l'autre requiert. La correspondance permet ainsi de sélectionner le service approprié pour répondre à la requête de l'utilisateur [19]. La découverte des services Web peut être syntaxique ou sémantique selon leurs descriptions impliquées. Les premières techniques de découverte de services Web se basaient sur les descriptions syntaxiques présentes dans les descriptions WSDL. Ce type de solution s'est avéré trop imprécis, et a engendré l'apparition d'autres algorithmes de découverte basés sur des descriptions sémantiques. Toutefois ces techniques peuvent varier selon l'architecture adoptée (centralisée ou distribuée).

2.2 Critères de découverte

Plusieurs critères peuvent être utilisés pour catégoriser les approches de découverte [20], nous avons :

2.2.1 Critère d'architecture

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 tandis que les systèmes décentralisés stockent les descriptions de services d'une manière distribuée.

2.2.2 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 tandis que la découverte automatique n'implique pas l'intervention de l'utilisateur pendant la recherche.

2.2.3 Critère de Matchmaking

Ce point de vue concerne les données à comparer ainsi que l'algorithme d'appariement de la requête avec la description du service.

2.3 Approches de découverte

2.3.1 Approches basées sur la représentation syntaxique

Le principe général de ces approches est la comparaison syntaxique entre la requête de l'utilisateur et les descriptions syntaxiques (WSDL) des services Web.

2.3.1.1 Approche UDDI

UDDI est un registre de descriptions de services Web. Dans le cas d'une architecture centralisée UDDI est utilisé comme registre central pour la publication et la découverte, basée mots clés, des services Web.

A l'étape de recherche, l'utilisateur ou le programme de recherche envoie une requête constituée de mots clés, cette requête est ensuite comparée avec les mots clés du registre UDDI. Un ensemble de descriptions des services Web est ensuite donné comme résultat de recherche, l'utilisateur sélectionne le service Web qui répond au mieux à ses exigences.

2.3.1.2 UDDI distribué

Dans le cas d'une architecture distribuée où les descriptions des services ne sont pas centralisées dans un même registre UDDI, une approche est proposée dans [21]. Cette approche consiste à connecter un nombre arbitraire de nœuds du réseau pour former de façon virtuelle le registre UDDI, tel que chaque nœud contient une partie des descriptions des services Web du réseau. Cet ensemble de nœuds est appelé nuage ou fédération UDDI. Lorsqu'un utilisateur accède à l'un des nœuds pour la recherche d'un service Web, le nœud en question transmet la requête de recherche aux autres nœuds avec lesquels il est connecté, et ainsi de suite pour les nœuds recevant cette requête.

Pour éviter qu'un nœud renvoie la même requête, un identificateur unique (ID) est associé à la requête. De plus, le nœud source de propagation de la requête associe à cette requête un nombre de sauts, tel que chaque fois que cette requête est propagée par un nœud, le nombre de saut est décrémenté de un. Quand le nombre de sauts atteint zéro, la requête n'est plus propagée à travers les nœuds du nuage UDDI. Les résultats de chaque nœud ayant reçu la requête sont ensuite expédiés au nœud source.

2.3.2 Approches Web sémantique

Plusieurs travaux se sont focalisés sur la description sémantique des services Web. Ce développement est de plus en plus significatif puisqu'il semble pouvoir aborder certaines insuffisances des approches basées sur les mots clés.

2.3.2.1 Approche de Paolucci

OWL-S permet d'étendre UDDI avec la description sémantique des services Web. OWL-S décrit un service Web à l'aide de trois classes. De ces classes, ServiceProfil est la classe qui fournit les paramètres fonctionnels nécessaires pour la découverte de services Web.

L'algorithme de Matchmaking [22] [23] permet de rechercher les descriptions des services Web qui ont une correspondance sémantique entre les paramètres fonctionnels définis dans les descriptions des services et ceux introduits dans la requête de recherche.

```
match(request) {
   recordMatch= empty list
   forall adv in advertisements do {
      if match(request, adv) then
          recordMatch.append(request, adv) }
   return sort(recordMatch);}
```

FIGURE 2.1 – Fonction pricipale de l'algorithme de Matchmaking [22]

Il y a correspondance sémantique entre les paramètres de sortie mentionnés dans la requête et ceux d'un service Web si et seulement si pour chaque paramètre de sortie de la requête il existe un paramètre de sortie qui peut lui correspondre dans la description du service Web. Si un des paramètres de sortie de la requête n'a pas de correspondance

sémantique avec un des paramètres de sortie du service alors le service n'est pas sélectionné. La correspondance sémantique entre les paramètres d'entrée est aussi faite de la même façon que pour les paramètres de sortie, sauf que l'ordre est inversé, c'est-à-dire la recherche d'une correspondance sémantique entre les paramètres d'entrée d'un service Web contre les paramètres d'entrée cités dans la requête. Ce critère garantit que le service Web satisfait les besoins du demandeur et que le demandeur fournit au service Web tous les paramètres d'entrée dont il a besoin pour fonctionner correctement.

```
outputMatch(outputsRequest, outputsAdvertisement) {
   globalDegreeMatch= Exact
   forall 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(recordMatch);}
```

FIGURE 2.2 – Fonction de Matching des concepts de sortie [22]

Le degré de correspondance sémantique entre deux paramètres de sortie ou deux paramètres d'entrée dépend de la correspondance sémantique entre les concepts associés à ces paramètres d'entrée et de sortie. La correspondance sémantique entre deux concepts est basée sur la relation entre ces deux concepts dans leurs ontologies OWL. Par exemple, considérant un service Web de vente de véhicules (Selling vehicle Web Service) dont le paramètre de sortie est spécifié comme "Vehicle", et une requête de recherche dont le paramètre de sortie est spécifiée comme "Car". Bien qu'il n'y ait pas de correspondance sémantique exacte entre le paramètre de sortie de la requête et celui du service Web, étant donné le fragment de l'ontologie de véhicule (voir figure 2.3), l'algorithme a pu déterminer un autre niveau de correspondance puisque le concept "Vehicle" englobe (subsume) le concept "Car".

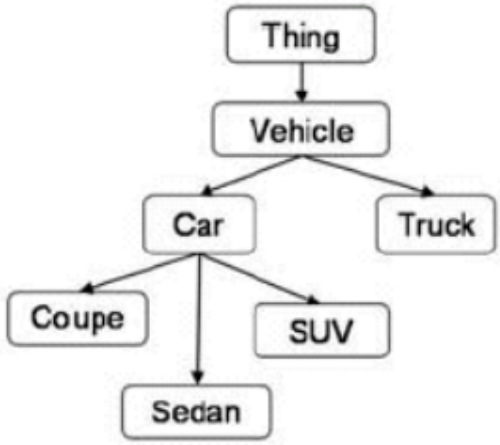


FIGURE 2.3 – Ontologie de véhicule [23]

L'algorithme permet de déterminer quatre niveaux de correspondance sémantique entre deux concepts. Soit OutR représente le paramètre de sortie cité dans la requête de recherche, et OutA le paramètre de sortie d'un service Web. Le degré de correspondance entre OutR et OutA peut être :
- Exact : Si OutR et OutA sont les mêmes ou si OutR est une sous classe immédiate de OutA dans l'ontologie du domaine de recherche.
- Plug in : Si OutA englobe (subsume) OutR.
- Subsume : Si OutR englobe (subsume) OutA, dans ce cas le service Web peut ne pas satisfaire la requête complètement.
- Fail : S'il n'y a aucune relation de subsomption entre OutA et OutR.

Le niveau de correspondance sémantique entre les paramètres d'entrée est assigné de la même manière que pour les paramètres de sortie, sauf que les arguments sont inversés, c'est à dire : degreeOFMatch(inA,inR) dans l'algorithme.

```
degreeOfMatch(outR,outA):
   if outA=outR then return exact
   if outR subclassOf outA then return exact
   if outA subsumes outR then return plugIn
   if outR subsumes outA then return subsumes
   otherwise fail
```

FIGURE 2.4 – Régles d'assignation du niveau de correspondance [22]

La dernière partie de l'algorithme est la partie consacrée à la classification des descriptions des services sélectionnés. Les services sont classés selon le niveau de correspondance sémantique entre leurs paramètres de sortie et ceux cités dans la requête. Si deux services sont du même niveau de correspondance sémantique avec la requête de recherche par rapport à leurs paramètres de sortie, une comparaison sur le niveau de correspondance sémantique par rapport aux paramètres d'entrée est alors effectuée.

Cet algorithme permet de rechercher et de sélectionner un ensemble de services Web sémantiques candidats satisfaisant la requête de recherche en terme de paramètres d'entrée et de sortie.

```
sortRule(match1,match2) {
   if match1.output > match2.output then match1 > match2
   if match1.output = match2.output
      & match1.input > match2.input then match1 > match2
   if match1.output = match2.output
      & match1.input = match2.input then match1 = match2
```

FIGURE 2.5 – Procédure de classification du résultat de recherche [22]

2.3.2.2 Approche de Bellur

Cette approche est basée sur le Matching des graphes bipartis et sur l'approche de Paolucci et al. [22][23]. Premièrement Bellur et al. [24][25] ont proposé une procédure alternative pour calculer le degré d'appariement (voir figure 2.6). Cette procédure inverse les concepts Plugin et Subsumes.

Algorithm 1 PROCEDURE match(outA, outQ)

```
1: if outA = outQ then
2:     return Exact
3: else if outQ superclass of outA then
4:     return Plugin
5: else if outQ subsumes outA then
6:     return Plugin
7: else if outA subsumes outQ then
8:     return Subsumes
9: else
10:    return Fail
11: end if
```

FIGURE 2.6 – Modification de la procédure match [24]

Graphe biparti : Un graphe $G = (V, E)$ est dit biparti s'il existe une partition de son ensemble de sommets en deux sous-ensembles telle que $V = V_0 \cup V_1$ et chaque arête $e \in E$ ait une extrémité dans V_0 et l'autre dans V_1. La figure 2.7 montre un graphe biparti pondéré G.

Matching : Un couplage ou appariement (en anglais matching) d'un graphe $G = (V, E)$ est un sous-graphe $G' = (V, E'), E' \subseteq E$ telle que E' est un ensemble d'arêtes qui n'ont pas de sommets en commun. La figure 2.7 montre un Matching G' pour le graphe G. Un Matching est complet si et seulement si, tous les sommets de V_0 sont liés. Le Matching G' de la figure 2.7 n'est pas complet car le sommet x n'est pas lié.

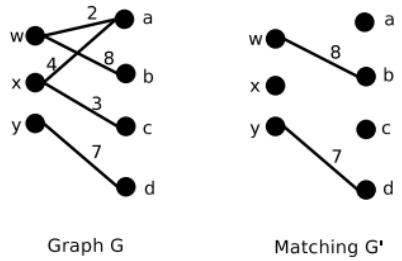


FIGURE 2.7 – Un graphe biparti et son Matching [24]

La modélisation du Matchmaking des services Web comme le Matching d'un graphe biparti

Considérons une requête Q et un service Web A. On modélise le problème de Matching de leurs sorties comme un problème de Matching d'un graphe biparti. Cela implique deux étapes :

- Construction d'un graphe biparti : Supposons que Q_{out} et A_{out} sont les ensembles des concepts de sortie en Q et A respectivement. On construit un graphe $G = (V_0 + V_1, E)$ telle que $V_0 = Q_{out}$ et $V_1 = A_{out}$.
Considérons deux concepts $a \in V_0$ et $b \in V_1$ et R leur degré de Matching (Exact, Plugin, Subsume, Fail). Si $R \neq Fail$, on définit une arête (a, b) étiquetée R.

- Définition d'un critère de Matching : On calcule un Matching complet de ce graphe biparti. Un Matching complet assure que chaque concept dans Q_{out} est lié à un concept dans A_{out}. Nous considérons les cas suivants :

- *Un Matching complet n'existe pas :* On conclue que la Matching échoue.
- *Plusieurs Matchings complets existent :* On doit choisir le Matching optimal.

Comment choisir le Matching optimal ?

Nous attribuons un poids numérique pour chaque arête dans le graphe biparti. Le poids d'une arête $e = (a, b)$ est attribué en fonction du degré de correspondance entre les concepts a et b.

Degré de Matching	Poids de l'arête
Exact	w_1
Plugin	w_2
Subsumes	w_3

Avec $w_1 < w_2 < w_3$

TABLE 2.1 – Poids des arêtes en fonction du degré de Matching

La figure 2.8 illustre un graphe biparti G construit à partir de Q_{out} et A_{out}. G' est un Matching complet de G.

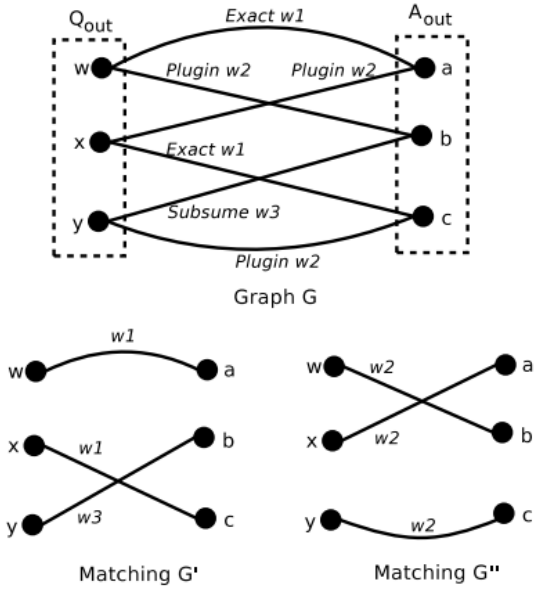


FIGURE 2.8 – Graphe biparti des concepts de sortie [24]

Supposons que $max(w_i)$ dénote le poids maximal dans G'. Le poids maximal représente le mauvais degré de Matching entre les deux ensembles de G'. Nous disons donc que $max(w_i)$ dénote le degré global de Matching pour G'.

Considérons le cas où plusieurs correspondances existent pour le graphe biparti. Un Matching optimal est un Matching complet dans lequel $max(w_i)$ est minimisé. Par exemple, dans la figure 2.8, G' et G'' sont deux Matchings complets de G. Nous pouvons maintenant déduire ce qui suit :

Matching	**max(w_i)**	**Matching global**
G'	w_3	Subsume
G''	w_2	Plugin

TABLE 2.2 – Le Matching global

L'algorithme de Bellur choisit la Matching dans lequel $max(w_i)$ est minimisé. Puisque $w_2 < w_3$, G'' (Plugin) est choisi.

Le Matching de concepts d'entrée est effectué de la même façon. En cas des sorties, chaque concept dans Q_{out} doit être lié. Alors que dans le cas des entrées, chaque concept dans A_{in} doit être lié. C'est-à-dire $V_0 = A_{in}$ et $V_1 = Q_{in}$.

Le calcul du Matching optimal

Pour calculer le Matching optimal l'algorithme hongrois [26] est utilisé et les poids des arêtes sont calculés comme suit :

Degré de Matching	Poids de l'arête
Exact	$w_1 = 1$
Plugin	$w_2 = (w_1 * \|V_0\|) + 1$
Subsumes	$w_3 = (w_2 * \|V_0\|) + 1$

$|V_0|$ = Cardinalité de l'ensemble V_0

TABLE 2.3 – Calcul du poids des arêtes

Algorithme de Bellur

L'algorithme est constitué de deux procédures *search()* et *match()*. La procédure *search()* (voir figure 2.9) accepte une requête en entrée et tente de la faire correspondre avec chaque service dans le référentiel. L'appariement est calculé pour les sorties et les entrées. Si l'appariement réussi (match $\neq$ Fail) alors on ajoute le service au résultat. Enfin, le résultat trié est envoyé au client.

Algorithm 2 search($Query$)

```
1:  Result = Empty List
2:
3:  for each Advt in Repository do
4:     outMatch = match(Query_out, Advt_out)
5:     if (outMatch = Fail) then
6:        Skip Advt. Take next Advt.
7:     end if
8:
9:     inMatch = match(Advt_in, Query_in)
10:    if (inMatch = Fail) then
11:       Skip Advt. Take next Advt.
12:    end if
13:
14:    Result.append(Advt, outMatch, inMatch)
15: end for
16:
17: return sort(Result)
```

FIGURE 2.9 – La procédure search [24]

La procédure *match()* (voir figure 2.10) accepte deux listes comme entrées et construit un graphe biparti. Elle appelle ensuite l'algorithme hongrois pour calculer un Matching complet.

Algorithm 3 match($List_1$, $List_2$)

```
1:  Graph G = Empty Graph (V_0 + V_1, E)
2:  V_0 ← List_1
3:  V_1 ← List_2
4:  (w_1, w_2, w_3) ← computeWeights(|V_0|)
5:
6:  for each concept a in V_0 do
7:     for each concept b in V_1 do
8:        degree = degreeOfMatch(a, b)
9:        if degree ≠ Fail then
10:          Add edge (a, b) to G
11:          if (degree = Exact) then w(a, b) = w_1
12:          if (degree = Plugin) then w(a, b) = w_2
13:          if (degree = Subsume) then w(a, b) = w_3
14:       end if
15:    end for
16: end for
17:
18: Graph M = hungarianMatch(G)
19: if (M = null) then
20:    No complete matching exists
21:    return Fail
22: end if
23:
24: Let (a, b) denote Max-Weight Edge in G
25: degree ← degreeOfMatch(a, b)
26: return degree
```

FIGURE 2.10 – La procédure match [24]

2.3.2.3 Système Speed-R

Développé à l'université de Georgie, le système Speed-R [27] vise à connecter l'ensemble des registres UDDI privés (chaque fournisseur de services a son propre registre UDDI) via le réseau P2P (Peer-to-peer). Afin d'apporter une sémantique aux descriptions des services, pour chaque registre du réseau P2P des ontologies spécifiques au domaine du registre sont implémentées à son niveau. La sémantique est apportée aux descriptions des services en faisant un mapping entre les spécifications du service et les concepts des ontologies. Quand un utilisateur veut chercher un service Web, il doit choisir un registre du domaine de recherche, le choix du registre est fait en se basant

sur l'ontologie de domaine des registres. L'utilisateur peut envoyer une requête simple ou peut utiliser l'agent de recherche sémantique pour exécuter la recherche sémantique de services Web dans le registre. La recherche sémantique des services est basée sur les paramètres fonctionnels du service. De plus, l'interaction entre les utilisateurs et les registres du réseau est réalisée par l'intermédiaire d'agents logiciels.

2.4 Conclusion

La découverte de services Web présente un axe de recherche émergeant. Diverses approches ont été proposées dans la littérature. L'objectif commun de ces approches était de découvrir des descriptions de services Web décrivant certains critères fonctionnels.

Avec le développement des services Web sémantiques de nouvelles approches de découverte ont été proposées. La plupart des approches sémantiques proposées étaient basées sur le calcul du niveau de correspondance sémantique entre les critères fonctionnels définis dans la description sémantique des services et ceux définis dans la requête de recherche.

Les approches de découverte sémantique donnent de bons résultats comparées aux approches syntaxiques.

Chapitre 3

La sélection des services Web

3.1 Introduction

La sélection des services Web consiste à choisir parmi un ensemble de services Web découverts ceux qui répondent aux préférences de l'utilisateur sur la base des besoins fonctionnels et non fonctionnels. Les besoins non fonctionnels sont généralement exprimés à l'aide des critères de QoS. Plusieurs approches et méthodes ont été présentées dans la littérature pour la sélection de services selon les propriétés non fonctionnelles. Dans la suite de ce chapitre, nous passerons en revue les principaux travaux proposés dans ce domaine.

3.2 Propriétés fonctionnelles et non fonctionnelles

Les propriétés fonctionnelles et non fonctionnelles de services définissent un service Web. En effet, les services se distinguent par les fonctionnalités qu'ils peuvent fournir. Ainsi deux services similaires, c'est à dire qui offrent sémantiquement tous les deux les mêmes fonctionnalités (exemple un service "Température" et un service "Heat" qui offrent des fonctions de mesure de température), peuvent se différencier par leurs propriétés non fonctionnelles. Par exemple le service "Température" a un temps d'exécution inférieur au service "Heat" alors que ce dernier dispose de propriétés de sécurités tel que la confidentialité [28].

3.2.1 Les propriétés fonctionnelles

Les propriétés fonctionnelles d'un service désignent les opérations qu'il peut fournir. Les propriétés fonctionnelles sont décrites dans la description de service en termes

d'entrées, sorties, pré-conditions et effets.

3.2.2 Les propriétés non fonctionnelles

Les propriétés non fonctionnelles de service (appelées aussi qualités de service) définissent les capacités de service à fonctionner dans de bonnes conditions en termes de disponibilité, performance, etc.

Ran [29] propose une classification des attributs de QoS, il propose alors quatre classes : la première classe regroupe les attributs reliés à l'exécution et qui peuvent évoluer dynamiquement tel que le temps d'exécution, le débit et la latence, la deuxième classe comporte les attributs de QoS reliés au support de la transaction et qui inclut les caractéristiques des transactions ACID (Atomicity, Consistency, Isolation, and Durability). La troisième classe inclut les attributs reliés à la gestion de coût. La dernière classe s'intéresse à la sécurité (l'authentification, la confidentialité, le cryptage, etc.).

Le groupe W3C propose un guide [30] pour définir les QoS et leurs métriques. Le guide proposé regroupe les QoS en 4 catégories : les attributs de performance, les attributs qui assurent la sureté de fonctionnement (Dependability), les attributs de sécurité et les attributs spécifiques au domaine d'application. Le tableau suivant illustre cette catégorisation :

Catégorie de QoS	**Attributs de QoS**
Performance	Temps d'exécution, débit, temps de réponse, ...
Sureté de fonctionnement	Disponibilité, accessibilité, fiabilité, ...
Sécurité	Authentification, non-répudiation, confidentialité, intégrité, ...
Attributs spécifiques aux domaines d'application	Prix, mode de payement, taux de pénalité, ...

TABLE 3.1 – Catégorisation des attributs de QoS

3.3 Sélection basée sur les besoins fonctionnels

Dans ce cas, la sélection de service se base principalement sur les propriétés fonctionnelles d'un service. Un fournisseur de service publie une description de son service avec laquelle un utilisateur peut chercher le service qui répond aux fonctionnalités requises. Les moyens de sélection de services actuels reposent sur l'adéquation syntaxique exacte

entre l'interface de service offerte par les fournisseurs et l'interface de service requise par l'utilisateur.

Cette technique de recherche syntaxique s'est avérée faible. Les fonctionnalités offertes sont décrites de manière hétérogène et le résultat peut ne pas correspondre aux attentes des clients, car c'est une recherche basée sur les mots clés. De nombreux travaux prennent par conséquent l'hypothèse que la sélection s'effectue sur une base sémantique ce qui rend la sélection encore plus efficace.

3.3.1 Insuffisance des approches basées sur les besoins fonctionnels

Cette approche basée sur les besoins fonctionnels présente un manque. En effet, dans l'environnement des services Web, plusieurs services peuvent offrir les mêmes fonctionnalités. Ce qui force l'utilisateur à choisir manuellement le service ou d'effectuer un choix aléatoire qui ne garantit pas une qualité de service. Par conséquent, l'aspect fonctionnel est certes important, mais insuffisant.

3.4 Sélection basée sur les besoins non fonctionnels

Pour pouvoir réaliser la sélection parmi un ensemble de services dont les fonctionnalités sont équivalentes : Seules les propriétés non fonctionnelles du service font l'objet de raffinement pour sélectionner le meilleur service. Les besoins non fonctionnels des services Web sont généralement exprimés à l'aide des critères de QoS.

3.5 Présentation du problème de sélection selon les propriétés de QoS

Un service Web composite est constitué d'un ensemble de tâches à exécuter, ses tâches composantes peuvent être exécutées à l'aide des services Web. En particulier, pour une même tâche, on peut découvrir plusieurs services Web aptes à l'exécuter, on serait alors face à un problème de sélection des services Web afin de choisir la combinaison de services qui offre la meilleure qualité de service. Cette sélection devra aussi prendre en considération le respect des contraintes de l'utilisateur.

3.6 Les stratégies de sélection

On distingue principalement deux stratégies de sélection différentes [31] :
- Sélection locale.
- Sélection globale.

3.6.1 La sélection locale

Elle a pour objectif de choisir le meilleur service Web pour chaque tâche individuelle à part entière en considérant des contraintes de QoS relatives à chaque tâche plutôt qu'en considérant des contraintes de QoS globales exprimées pour l'ensemble des tâches. Il s'agit de sélectionner pour chaque tâche un service Web apte à l'exécuter en tenant en compte les contraintes de l'utilisateur.

En d'autres termes, cela revient à choisir pour chaque ensemble de candidats S_i un service Web s_{ij} qui vérifie au mieux les contraintes de QoS définies par l'utilisateur. La figure 3.1 décrit la stratégie de sélection locale.

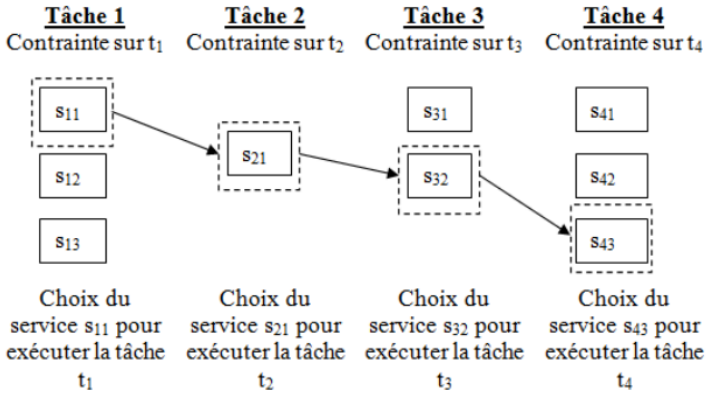


FIGURE 3.1 – La sélection locale [4]

3.6.2 La sélection globale

Contrairement à la méthode précédente, on choisit la combinaison des services Web qui garantit la meilleure qualité globale en tenant compte des contraintes de QoS et des préférences globales assignées pour l'ensemble des tâches.

Afin de pouvoir appliquer une stratégie de sélection globale, on calcule la valeur des critères de QoS relative au service composite. Le calcul de ces critères s'appuie sur l'application de fonctions d'agrégation (qui peuvent être une somme, un produit,

un minimum, . . .). Ensuite, on choisit la combinaison qui offre les meilleures valeurs de QoS parmi toutes les combinaisons possibles par rapport aux contraintes et aux préférences de l'utilisateur. La figure 3.2 décrit la stratégie de sélection globale.

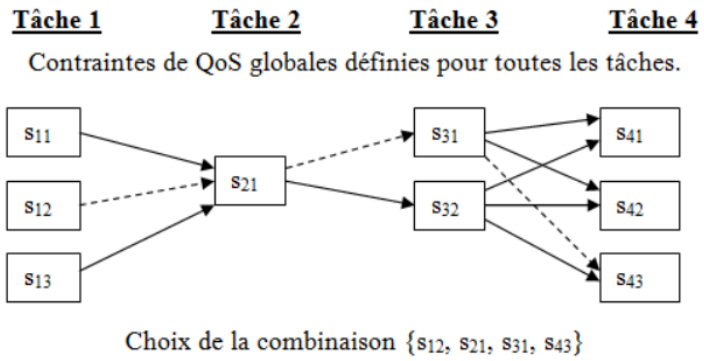


FIGURE 3.2 – La sélection globale [4]

3.7 Les méthodes de sélection

3.7.1 Les méthodes de sélection locale

3.7.1.1 La somme pondérée

L'une des méthodes d'agrégation qui a été utilisée dans le domaine de sélection des services Web est la méthode SAW (Simple Additive Weighting). Elle se caractérise par une agrégation additive des critères par une somme pondérée.

Dans [32] et [33] les auteurs proposent d'appliquer cette technique pour le problème de sélection de Web services. Le principe de cette méthode consiste à affecter un vecteur de QoS pour chaque Web service candidat. Ce vecteur contient les valeurs des critères de qualités qui caractérisent chaque Web service. Les différents paramètres sont exprimés dans des unités, échelles, et des dimensions différentes, ce qui conduit aux problèmes de compensation et de changement d'échelle si on applique directement la méthode SAW. Pour remédier à ce genre de problème, les auteurs effectuent dans une première étape une normalisation des paramètres de qualité de chaque vecteur.

Après l'étape de normalisation des paramètres, les auteurs effectuent une affectation des poids aux paramètres de qualité selon les préférences de l'utilisateur puis ils calculent le score de chaque Web service qui représente une somme pondérée des valeurs des paramètres de QoS normalisées. Ils sélectionnent ensuite le service qui offre le meilleur

score pour exécuter la tâche.

3.7.1.2 Distance euclidienne

Dans [34], les auteurs proposent une méthode basée sur la mesure d'une distance entre les préférences de l'utilisateur en termes de QoS et les annonces des fournisseurs de services. Ils utilisent la distance euclidienne pour leur algorithme qui est un moyen pour évaluer la similitude entre deux vecteurs. L'algorithme détermine quel Web service ws_i d'un ensemble de services $WS = ws_1, ws_2, ..., ws_n$ qui sera choisi pour exécuter la demande du client selon ses spécifications de qualité de services $QPc = qpc_1, qpc_2, ..., qpc_k$. Ils construisent dans un premier temps une matrice QoS_{nk}, où n représente le nombre total de services qui ont la même propriété fonctionnelle, et k représente le nombre total de propriétés de QoS. La matrice obtenue est la suivante :

$$QoS = \begin{pmatrix} qp_{11} & qp_{12} & \cdots & qp_{1k} \\ qp_{21} & qp_{22} & \cdots & qp_{2k} \\ \vdots & \ddots & \vdots & \\ qp_{n1} & qp_{n2} & \cdots & qp_{nk} \end{pmatrix}$$

Chaque ligne dans la matrice correspond à un Web service candidat, et chaque colonne représente une propriété de QoS. Suivant les indications de la matrice QoS, chaque service est modélisé en tant que vecteur de k-dimensions, chaque Web service est représenté comme un point dans les k-dimensions. Les auteurs emploient la distance euclidienne pour calculer la distance entre le vecteur de QoS spécifié par l'utilisateur et les propriétés existantes de QoS pour chaque vecteur dans la matrice suivant la formule suivante :

$$dist(QPc, QP_i) = \sqrt{\sum_{j=1}^{k} (qpc_j - qp_{ij})^2}$$

Ensuite les auteurs choisissent le Web service qui a la plus petite distance euclidienne. Mais avant de calculer la distance euclidienne une normalisation des attributs de QoS est employée pour ramener tous les attibuts de QoS dans l'intervalle $[0, 1]$.

3.7.1.3 Sémantique

Plusieurs travaux [35], [36] et [37] définissent les paramètres de QoS des fournisseurs de services et les paramètres de QoS de la requête de l'utilisateur avec des concepts d'une ontologie de domaine de QoS pour pouvoir effectuer la sélection.

Dans [35], les auteurs proposent un algorithme pour la sélection de services basé sur le matching sémantique entre la requête de l'utilisateur et l'annonce des différents fournisseurs de services en terme de QoS. Pour ajouter les informations de QoS dans la définition du service les auteurs utilisent le "serviceParameter" de la spécification de OWL-S pour ajouter ces informations. Les auteurs distinguent deux types de paramètres, les paramètres à maximiser et les paramètres à minimiser. L'algorithme commence par parcourir chaque paramètre de qualité défini dans la requête de l'utilisateur et fait un matching avec les qualités annoncées par les fournisseurs de service. Les auteurs associent à chaque service un degré de matching en choisissant le degré de matching le plus petit trouvé entre les paramètres de la requête et les paramètres annoncés par le service. Le service qui offre le meilleur degré de matching est choisi pour exécuter la demande de l'utilisateur.

Dans [36], les auteurs proposent une ontologie de qualité de service qui définit les différents paramètres de qualité pour résoudre le problème de sélection. Les auteurs distinguent deux types d'attributs de qualité, les attributs à maximiser et les attributs à minimiser. Ces différents attributs sont définis dans des dimensions et échelles différentes c'est pourquoi une phase de normalisation est exécutée en premier. Ensuite, les auteurs attribuent à chaque paramètre de qualité, un poids qui représente la préférence de l'utilisateur et ils calculent un score en utilisant une somme pondérée pour chaque service. Le service qui offre le meilleur score est alors choisi pour exécuter la requête de l'utilisateur.

Dans [37], les auteurs incluent le degré du matching sémantique dans le calcul du meilleur service. Ils calculent, dans une première étape, le score de chaque service en utilisant une somme pondérée, ensuite, ils utilisent ce résultat avec le degré de matching sémantique des propriétés fonctionnelles pour calculer un nouveau score en utilisant la somme pondérée. Le service qui offre le meilleur score est choisi pour exécuter la demande de l'utilisateur.

3.7.2 Les méthodes de sélection globale

3.7.2.1 Programmation mathématique

Le problème de sélection peut être considéré comme un problème d'optimisation multicritère, car plusieurs paramètres entrent en jeux pour sélectionner les services adéquats, en d'autres termes, on cherche à optimiser simultanément plusieurs critères à la fois. La tâche de trouver la meilleure combinaison de services avec la meilleure qualité est un genre de problème d'optimisation combinatoire. La plupart des problèmes

d'optimisation possèdent un ensemble fini de solutions (ou dénombrable). Il est donc possible, en principe, d'énumérer toutes ces solutions, et ensuite de prendre celle qui nous satisfait. L'inconvénient de cette solution est le nombre important de ces solutions.

Dans [38], les auteurs modélisent le problème sous forme d'un problème de programmation mathématique qui est défini par un ensemble de variables, une fonction objective et un ensemble de contraintes comme suit :

$$Fonction\ objective\ f(x_1, x_2, ..., x_n) = \sum_{i=1}^{n} c_i x_i$$

$$Contraintes \begin{cases} a_{11}x_1 + ... + a_{1n}x_n \leqslant b_1 \\ \vdots \qquad\qquad\qquad\qquad où\ x_i\ sont\ les\ variables \\ a_{m1}x_1 + ... + a_{mn}x_n \leqslant b_m \end{cases}$$

Pour résoudre le problème de sélection, les auteurs supposent que la composition est constituée de m tâches. Pour chaque tâche il y a n services candidats pour exécuter celle-ci. Les données du problème sont les suivantes :

- Les variables du problème : Les variables du problème sont les s_{ij} qui sont définis comme suit :

$$S_{ij} = \begin{cases} 1 \text{ Si le service i est sélectionné pour exécuter la tâche j} \\ 0 \text{ sinon} \end{cases}$$

- Les contraintes : Dans une composition de service, il y a exactement un service qui est choisi pour exécuter une tâche. Par conséquent, les variables précédemment définies ont la contrainte suivante :

$$\sum_{i=1}^{n} S_{ij} = 1\ avec\ 1 \leqslant j \leqslant m$$

- La fonction objective : Dans [38], les auteurs prennent en considération quatre critères différents (Temps de réponse, Fiabilité, Disponibilité et le Prix). La fonction objective est ainsi définie :

$$f(s_{11}, ..., s_{nm}) = Q_{temps\_reponse} + Q_{fiabilite} + Q_{disponibilite} + Q_{prix}$$

Pour résoudre ce problème, les auteurs appliquent la méthode Branch-and-Bound [39] (procédure par évaluation et séparation progressive) qui consiste dans l'énumération

des solutions possibles d'une façon intelligente. Cette technique arrive à éliminer des solutions partielles qui ne mènent pas à la solution recherchée. L'algorithme proposé est divisé en deux phases :

- Phase d'élagage : qui consiste à filtrer les candidats qui ne sont pas assez bons pour chaque tâche.

- Phase Branch-and-Bound : Parcourt les combinaisons possibles de composition des candidats restants de la première phase, afin de trouver la meilleure composition de services.

3.7.2.2 Le chemin optimal

Dans [40], les auteurs modélisent la composition des Web services sous forme d'un graphe. Ainsi, la résolution du problème de sélection revient à trouver le chemin le plus optimal qui satisfait les contraintes de QoS de l'utilisateur.

Les auteurs proposent une méthode basée sur la somme pondérée pour trouver le chemin le plus optimal dans la composition. Le problème est décrit comme suit, soit un ensemble de tâche $T = (t_1, t_2, ..., t_n)$ qui doit être exécuté, un ensemble de services $S = (s_1, s_2, ..., s_m)$ disponibles pour chaque tâche, et un ensemble de QoS $Q = (q_1, q_2, ..., q_z)$ pour chaque tâche. Les auteurs ajoutent deux états C et C' qui représentent respectivement l'état initial et l'état final. Pour ordonner les différentes tâches, un graphe de précédence est utilisé en partant de l'état initial C vers l'état final C' comme suit :

$$C \rightarrow t_1 \rightarrow t_2 \rightarrow ... \rightarrow t_n \rightarrow C'$$

Pour résoudre le problème de sélection, les auteurs modélisent le problème sous forme d'un arbre qui est construit comme suit, on commence par la tâche t_1, on crée un nœud pour chaque service disponible pour cette tâche, de même que pour toutes les autres tâches, ensuite, on relie les différents nœuds de t_1 avec tous les nœuds de t_2 avec des arcs allant de t_1 vers t_2, on refait l'opération jusqu'à ce qu'on arrive à la tâche t_n. On relie ensuite l'état initial C avec les nœuds de t_1 et chaque nœud de la tâche t_n est considéré comme un état final C'. Ainsi, le problème de sélection revient à choisir le chemin optimal dans l'arbre qui va de la racine vers l'une des feuilles.

Pour pouvoir calculer le coût d'un chemin dans l'arbre, les auteurs associent pour chaque nœud un score qui est calculé par une somme pondérée de ses différents paramètres de QoS. Chaque paramètre est sujet à une phase de normalisation avant de calculer le score du nœud, car ils sont définis dans des échelles et des dimensions différentes, ensuite les auteurs affectent à chaque paramètre un poids qui représente la

préférence de l'utilisateur d'un paramètre sur un autre. Le calcul du score d'un chemin est réalisé en faisant une addition des scores des différents nœuds qui composent ce chemin. Le chemin qui a le score le plus élevé est choisi pour exécuter la composition.

3.8 Conclusion

La sélection de Web services devient de plus en plus incontournable, vu le nombre de Web services qui peuvent répondre à un même besoin fonctionnel. Dans la littérature, différents efforts ont été dépensés pour résoudre ce problème en se basant sur les propriétés non fonctionnelles des Web services. Le problème de sélection est considéré comme étant une tâche difficile vue sa complexité exponentielle.

Chapitre 4

Implémentation et expérimentation

4.1 Introduction

Dans ce chapitre nous décrivons un prototype implémentant l'environnement de recherche et de sélection des services Web. Nous décrivons en premier lieu, l'architecture générale du système proposé (i.e. les différents composants ainsi que leurs rôles), ensuite nous présentons les algorithmes et les technologies utilisés pour la réalisation de notre prototype, après nous montrons les différentes expérimentations menées et finalement nous discutons les résultats obtenus.

4.2 Architecture générale de l'application

L'architecture globale que nous proposons pour la recherche et la sélection des services Web est illustrée par la figure 4.1 :

Un utilisateur envoie une requête composée de deux parties : une partie fonctionnelle (PF) constituée des concepts d'entrée et de sortie et une partie non fonctionnelle (PNF) constituée des valeurs des attributs de QoS.

Le module de découverte permet la comparaison d'une requête utilisateur (la partie fonctionnelle) avec un ensemble de descriptions OWLS de services. Les services retenus (SR) sont les services qui peuvent satisfiare les besoins fonctionnels de l'utilisateur selon le module de découverte.

Le module de sélection évalue la qualité des services retenus en tenant compte les contraintes de l'utilisateur (la partie non fonctionnelle de la requête).

Les deux modules précédents sollicitent le module d'analyse des descriptions des services Web (module de Parsing) qui permet l'extraction des concepts d'entrée et

de sortie associés aux descriptions OWLS. Ce module récupère aussi les valeurs des attributs de QoS associées à chaque service.

L'utilisateur aura comme réponse les services retenus (qui peuvent satisfiare les besoins fonctionnels) plus des informations concernant la qualité de ces services.

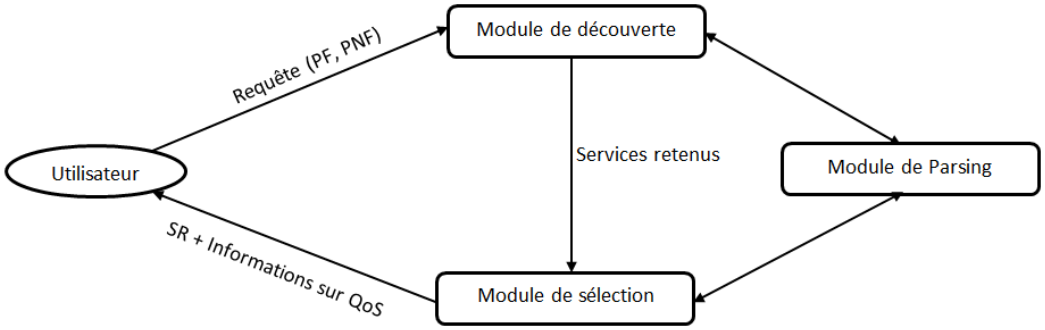


FIGURE 4.1 – Architecture globale

4.3 Les algorithmes de recherche

Le module de découverte propose trois algorithmes : algorithme de Paolucci sans suppression des concepts, algorithme de Paolucci avec suppression des concepts et l'algorithme de Bellur.

Remarque : L'algorithme de Paolucci parcourt la liste des concepts de sortie de la requête et il essaie de trouver le concept de sortie d'un service Web qui offre un matching maximal. Au début, chaque concept de sortie est un candidat pour un tel matching. Nous appelons cet ensemble de concepts de sortie du service Web la liste des candidats. L'algorithme original ne précise pas si un concept de la liste des candidats est supprimé une fois qu'il a été sélectionné par le processus de Matching, pour cela nous avons traité les deux cas : avec et sans suppression des concepts.

Algorithm 1 La fonction outputMatch sans suppression des concepts

```
1: function OUTPUTMATCH(queryOutputs, advertisementOutputs)
2:
3:     globalDegreeMatch = EXACT;
4:
5:     for i ← 1, |queryOutputs| do
6:         degreeMatch = FAIL;
7:         for j ← 1, |advertisementOutputs| do
8:             temp = DEGREEOFMATCH(queryOutputs[i], advertisementOutputs[j]);
9:             if temp > degreeMatch then
10:                degreeMatch = temp;
11:            end if
12:        end for
13:
14:        if degreeMatch = FAIL then
15:            return FAIL;
16:        end if
17:
18:        if degreeMatch < globalDegreeMatch then
19:            globalDegreeMatch = degreeMatch;
20:        end if
21:    end for
22:
23:    return globalDegreeMatch;
24:
25: end function
```

Algorithm 2 La fonction outputMatch avec suppression des concepts

```
1: function OUTPUTMATCH(queryOutputs, advertisementOutputs)
2:
3:     advertisementOutputsTemp = advertisementOutputs;
4:     globalDegreeMatch = EXACT;
5:
6:     for i ← 1, |queryOutputs| do
7:         degreeMatch = FAIL;
8:         for j ← 1, |advertisementOutputsTemp| do
9:             temp = DEGREEOFMATCH(queryOutputs[i], advertisementOutputsTemp[j]);
10:            if temp > degreeMatch then
11:                degreeMatch = temp;
12:                maxDegreeMatchIndex = j;
13:            end if
14:        end for
15:
16:        if degreeMatch = FAIL then
17:            return FAIL;
18:        end if
19:
20:        if degreeMatch < globalDegreeMatch then
21:            globalDegreeMatch = degreeMatch;
22:        end if
23:
24:        advertisementOutputsTemp.removeElementAt(maxDegreeMatchIndex);
25:    end for
26:
27:    return globalDegreeMatch;
28:
29: end function
```

4.4 Les algorithmes de sélection

Pour la sélection on a trois cas :

1. Si l'utilisateur veut donner des coefficients (des poids) aux attributs de QoS on utilise l'algorithme de la moyenne pondérée.

2. Si l'utilisateur veut donner des valeurs aux attributs de QoS on utilise l'algorithme de la distance euclidienne.
3. Si l'utilisateur veut classer les attributs de QoS on utilise l'algorithme MAGIQ (Multi-Attribute Global Inference of Quality).

4.4.1 La moyenne pondérée

Algorithm 3 La moyenne pondérée

procedure COMPUTEQOSSCORE($advertisements$, $query$)

  for all adv **in** $advertisements$ **do**

    $score = \frac{\sum_{i=1}^{|query|} query.att_i * adv.att_i}{\sum_{i=1}^{|query|} query.att_i}$;

    $adv.QoSScore = score * 100$;

  end for

end procedure

4.4.2 La distance euclidienne

Algorithm 4 La distance euclidienne

procedure COMPUTEQOSSCORE($advertisements$, $query$)

  NORMALIZE($query$);

  for all adv **in** $advertisements$ **do**

    $distance = 0$;

    for $i \leftarrow 1, |query|$ **do**

      if $adv.att_i < query.att_i$ **then**

        $distance = distance + (query.att_i - adv.att_i)^2$;

      end if

    end for

    $distance = \sqrt{distance}$;

    $score = 1 - \frac{distance}{maxDistance}$;

    $adv.QoSScore = score * 100$;

  end for

end procedure

4.4.3 MAGIQ

La technique MAGIQ [41] utilise le concept ROC (Rank Order Centroids). Les centroides de classement constituent un moyen de convertir des rangs (premier, deuxième, troisième) en notes ou pondérations qui sont des valeurs numériques. Si n est le nombre des attributs, la pondération de l'attribut k est :

$$W(A_k) = \frac{\sum_{i=k}^{n} \frac{1}{i}}{n}$$

Algorithm 5 MAGIQ

```
1: procedure computeQosScore(advertisements, query)
2:     weights = computeWeights(|query|);
3:     for all adv in advertisements do
4:         score = Σ_{i=1}^{|query|} weights[query.att_i] * adv.att_i;
5:         adv.QoSScore = score * 100;
6:     end for
7: end procedure
8:
9: function computeWeights(|attributes|)
10:    for k ← 1, |attributes| do
11:
12:        w[k] = Σ_{i=k}^{|attributes|} 1/i;
13:
14:        w[k] = w[k] / |attributes|;
15:
16:    end for
17:
18:    return w;
19: end function
```

4.5 Les technologies d'implémentation

4.5.1 Java

Comme langage de programmation, nous avons opté pour Java pour les raisons suivantes :

- C'est un langage orienté objet, ce qui facilite énormément notre travail.
- C'est un langage simple et efficace.
- Sa richesse en bibliothèques rend les tâches de création des interfaces graphiques et gestion des structures de données... beaucoup plus aisées. Ceci représente pour nous un atout intéressant.

4.5.2 NetBeans

Nous avons développé notre application sous l'EDI (Environnement de Développement Intégré) NetBeans. En tirant avantage de cette trousse à outils gratuite, basée sur des standards, les développeurs peuvent concevoir des applications complexes plus rapidement, avec une plus grande assurance de robustesse et de concevoir des applications qui résisteront à l'épreuve du temps. Pour cela le choix de Netbeans était fondamental.

4.5.3 L'API OWL

On a choisi l'API OWL pour manipuler les ontologies car elle offre beaucoup d'avantages :
- Elle permet de charger un document d'ontologie d'un URL donné ou d'un fichier local.
- Elle est capable de détecter et charger les ontologies qui sont incluses dans d'autres ontologies.
- Elle permet de naviguer récursivement sur la hiérarchie des classes de l'ontologie.

4.5.4 L'API JFreeChart

Pour la visualisation des résultats obtenus sous forme graphique nous avons utilisé la librairie JFreeChart qui est une librairie gratuite, écrite en java, qui simplifie la génération et l'affichage de graphe de qualité.

4.5.5 OWLS-TC (Test Collection)

OWLS-TC version 3.0 décrit un ensemble de services Web à travers des documents OWLS, elle dispose de 1007 services Web segmentés en 07 classes : le domaine d'éducation, le domaine médical, le domaine de nourriture, le domaine militaire, le domaine de voyage (tourisme), le domaine de communication et le domaine d'économie.

La base propose aussi un ensemble de requêtes réparti sur les 07 classes, ces requêtes sont modélisées sous forme de documents OWLS. Chaque service Web (i.e. document

OWLS) est étiqueté manuellement par des experts humains comme étant relevant (pertinent) ou non par rapport à une requête donnée. Ceci permet le calcul des rappels et des précisions des approches de découverte proposées.

4.5.5.1 Extension de la base OWLS-TC

OWLS-TC est une bonne base pour tester les approches de découverte mais elle ne prend pas en considération les attributs de QoS. Pour pouvoir tester les approches de sélection, nous avons modifié un extrait de la base (350 fichiers) en ajoutant 4 attributs de QoS (le temps d'exécution, le prix, la sécurité et la réputation). Les valeurs de ces attributs sont normalisées en utilisant les deux formules suivantes [42] :
Si att_i est un attribut à minimiser (le temps d'exécution ou le prix), on utilise la formule suivante :

$$norm(ValAtt_i) = \frac{ValAtt_i^{max} - ValAtt_i}{ValAtt_i^{max} - ValAtt_i^{min}}$$

Si att_i est un attribut à maximiser (la sécurité ou la réputation), on utilise la formule suivante :

$$norm(ValAtt_i) = \frac{ValAtt_i - ValAtt_i^{min}}{ValAtt_i^{max} - ValAtt_i^{min}}$$

Tel que $ValAtt_i \in [ValAtt_i^{min}, ValAtt_i^{max}]$

```
<profile:Profile rdf:ID="CAR_PRICE_PROFILE">
        <service:isPresentedBy rdf:resource="#CAR_PRICE_SERVICE"/>
        <profile:serviceName xml:lang="en">car price service</profile:serviceName>
        <profile:textDescription xml:lang="en">This service returns price of a car.</profile:textDescription>
        <profile:hasInput rdf:resource="#_CAR"/>
        <profile:hasOutput rdf:resource="#_PRICE"/>
        <profile:has_process rdf:resource="CAR_PRICE_PROCESS"/>
        <QoS>
                <executionTime>0.33</executionTime>
                <price>0.49</price>
                <security>0.33</security>
                <reputation>0.97</reputation>
        </QoS>
</profile:Profile>
```

FIGURE 4.2 – Fragment d'un fichier OWLS après l'ajout des attributs de QoS

4.6 L'interface utilisateur

La fenêtre principale de l'application est représentée par la figure 4.3. Elle permet l'accès à d'autres fenêtres pour simplifier et optimiser les tâches. La fenêtre contient trois boutons :

1. Charger les services : Ce bouton permet de lancer le module de Parsing qui analyse les descriptions des services Web (fichiers OWLS) et extrait les informations suivantes :
- Le nom du service.
- Le lien vers le fichier OWLS.
- Les entrées et les sorties (les concepts). Pour chaque concept, le module de Parsing extrait les informations suivantes :
- Le nom du concept.
- Le lien vers l'ontologie qui définit ce concept. Ensuite le module de Parsing va analyser l'ontologie pour extraire les informations suivantes :
- Les pères directs.
- Les pères directs et indirects (les ancêtres).
- Les fils directs.
- Les fils directs et indirects (les descendants).

2. Rechercher : Ce bouton permet de lancer une recherche basée sur les propriétés fonctionnelles et non fonctionnelles.

3. Tests et évaluation : Ce bouton permet d'afficher une fenêtre pour tester et évaluer les approches de découverte.

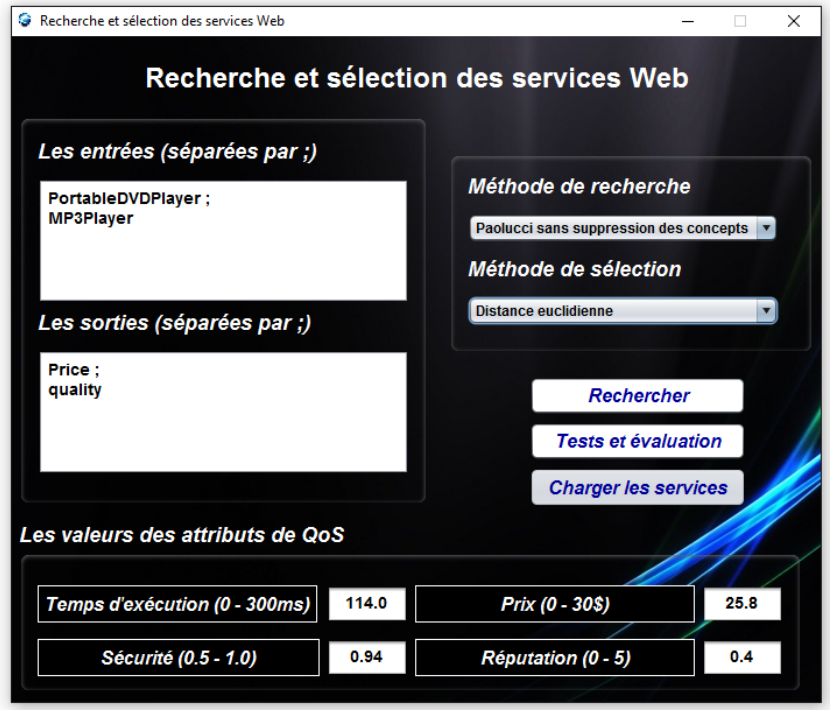


FIGURE 4.3 – Fenêtre principale

4.7 Expérimentation

Dans cette section, nous décrivons les expériences permettant l'analyse des performances des approches implémentées. Ces expérimentations sont menées sur une plateforme ayant un processeur *Intel Core i3 avec 04 Go de RAM* et sous le système d'exploitation *Windows 10*. Le système est développé avec *java* (*jdk 1.8.0*) et sous l'environnement *NetBeans 8.0.2*.

4.7.1 Performances des approches de découverte

La table 4.1 montre les requêtes utilisées pour évaluer les approches et les tables 4.2, 4.3 et 4.4 représentent le résultat d'évaluation des approches de découverte en terme de rappel, précision et f-mesure.

Selon [43] les critères de rappel et de précision sont définis comme suit :

- La précision : c'est le rapport entre les réponses correctes fournies par le système et le nombre total des réponses, et plus formellement :

$$Précision = \frac{VP}{VP + FP}$$

Avec :

- Les vrais positifs (VP) : sont les réponses correctes qui sont retournées par le système.
- Les faux positifs (FP) : sont les réponses incorrectes qui sont retournées par le système.

- Le rappel : c'est le rapport entre les réponses correctes fournies par le système et le nombre réel des réponses correctes, et plus formellement :

$$Rappel = \frac{VP}{VP + FN}$$

Avec :

- Les faux négatifs (FN) : sont des réponses correctes qui ne sont pas retournées par le système.
- Les vrais négatifs (VN) : sont des réponses incorrectes qui ne sont pas retournées par le système.

- F-mesure : c'est une mesure qui combine la précision et le rappel :

$$F - mesure = \frac{2 * Rappel * Précision}{Rappel + Précision}$$

N° de la requête	Les entrées	Les sorties
1	Title ; User	Book
2	Person ; Book ; CreditCardAccount	Price
3	Book	Price
4	Book	TaxedPrice ; Price
5	Car	TaxedPrice ; Price
6	Car	Price ; TaxedPrice
7	Price	IrishCoffee ; MixeryCola
8	MaxPrice	Cola
9	PortableDVDPlayer ; MP3Player	Price ; quality
10	Science-Fiction-Novel ; User	Price

TABLE 4.1 – Les requêtes de test

N° de la requête	VP	VN	FP	FN	Rappel	Précision	F-mesure
1	3	347	0	0	1.0	1.0	1.0
2	5	344	0	1	0.83	1.0	0.91
3	13	336	0	1	0.93	1.0	0.96
4	1	344	5	0	1.0	0.17	0.29
5	2	331	17	0	1.0	0.11	0.2
6	2	331	17	0	1.0	0.11	0.2
7	11	338	1	0	1.0	0.92	0.96
8	9	338	2	1	0.9	0.82	0.86
9	1	349	0	0	1.0	1.0	1.0
10	11	337	0	2	0.85	1.0	0.92

TABLE 4.2 – Résultat d'évaluation de l'approche de Paolucci sans suppression des concepts

N° de la requête	VP	VN	FP	FN	Rappel	Précision	F-mesure
1	3	347	0	0	1.0	1.0	1.0
2	5	344	0	1	0.83	1.0	0.91
3	13	336	0	1	0.93	1.0	0.96
4	1	349	0	0	1.0	1.0	1.0
5	2	348	0	0	1.0	1.0	1.0
6	1	348	0	1	0.5	1.0	0.67
7	7	339	0	4	0.64	1.0	0.78
8	9	338	2	1	0.9	0.82	0.86
9	0	349	0	1	0.0	0.0	0.0
10	11	337	0	2	0.85	1.0	0.92

TABLE 4.3 – Résultat d'évaluation de l'approche de Paolucci avec suppression des concepts

N° de la requête	VP	VN	FP	FN	Rappel	Précision	F-mesure
1	3	347	0	0	1.0	1.0	1.0
2	5	344	0	1	0.83	1.0	0.91
3	13	336	0	1	0.93	1.0	0.96
4	1	349	0	0	1.0	1.0	1.0
5	2	348	0	0	1.0	1.0	1.0
6	2	348	0	0	1.0	1.0	1.0
7	11	339	0	0	1.0	1.0	1.0
8	9	338	2	1	0.9	0.82	0.86
9	1	349	0	0	1.0	1.0	1.0
10	11	337	0	2	0.85	1.0	0.92

TABLE 4.4 – Résultat d'évaluation de l'approche de Bellur

Discussion

On remarque que les trois approches ont des performances similaires si les concepts d'entrée/sortie n'ont pas de relation de subsomption entre eux (requêtes 1, 2, 3, 8 et 10). Par contre si on a une relation de subsomption entre les entrées (ou les sorties) on remarque une augmentation dans le nombre des FP pour l'approche de Paolucci sans suppression des concepts et une augmentation dans le nombre des FN pour l'approche

de Paolucci avec suppression des concepts et l'ordre des concepts influe les résultats de cette dernière (requêtes 5 et 6) de l'autre côté on peut dire que l'approche de Bellur a montré son efficacité dans ce genre de requêtes en donnant les meilleurs résultats par rapport aux deux approches de Paolucci (avec et sans suppression des concepts).

4.7.2 Performances des approches de sélection

Le but des approches de sélection est d'aider l'utilisateur à choisir le meilleur service parmi plusieurs services qui offrent le même degré de matching fonctionnel. Par exemple si un utilisateur a envoyé une requête R_1 (voir la table 4.5) et a utilisé pour la recherche la méthode de Paolucci sans suppression des concepts et pour la sélection la distance euclidienne. Le résultat de cette requête est représenté par la table 4.6.

Partie fonctionnelle	**Les entrées :** Science-Fiction-Novel ; User
	Les sorties : Price
Partie non fonctionnelle	**Temps d'exécution :** 198.0
	Prix : 11.40
	Sécurité : 0.615
	Réputation : 2.15

TABLE 4.5 – La requête R_1

N° du service	**Matching des sorties**	**Matching des entrées**	**Score de QoS**
1	EXACT	EXACT	100.0 %
2	EXACT	EXACT	90.38 %
3	EXACT	EXACT	83.88 %
4	EXACT	EXACT	70.84 %
5	EXACT	SUBSUME	96.0 %
6	EXACT	SUBSUME	95.5 %
7	EXACT	SUBSUME	88.49 %
8	EXACT	SUBSUME	82.0 %
9	EXACT	SUBSUME	70.5 %
10	SUBSUME	EXACT	90.14 %
11	SUBSUME	EXACT	88.5 %

TABLE 4.6 – Résultat de la requête R_1

Discussion

On remarque que 4 services ont le même degré de matching fonctionnel (Matching des entrées = Matching des sorties = EXACT). Dans ce cas seul le score de QoS fait la différence car il y a un seul service qui a un score de 100%.

4.8 Conclusion

Nous avons présenté dans ce chapitre l'architecture générale de notre application avant de détailler la structure et le fonctionnement des différents modules ensuite nous avons essayé à travers les séries de tests effectués à suivre le comportement des algorithmes de recherche que nous avons programmés et on a montré l'utilité des algorithmes de sélection pour aider l'utilisateur à choisir un service Web parmi plusieurs qui offrent le même besoin fonctionnel.

Conclusion générale

Au terme de ce mémoire, nous procédons dans les lignes qui suivent à un récapitulatif du travail effectué. Rappelons que notre travail consistait à développer une application qui permet la recherche et la sélection des services Web.

La première partie de ce mémoire est réservée aux travaux d'état de l'art. Nous nous sommes attelés, dans un premier temps, à définir les différents concepts liés aux services Web et nous avons présenté quelques techniques et algorithmes pour la recherche et la sélection des services Web. Nous avons vu qu'il en existe une grande variété.

La deuxième partie était consacrée à l'implémentation. A ce niveau, nous avons présenté l'architecture générale de notre application avant de détailler la structure et le fonctionnement des différents modules.

Comme perspectives, nous envisageons un ensemble de travaux futurs qui concernent les aspects suivants :

- Etendre l'architecture pour permettre la composition des services Web.
- La prise en compte des propriétés de QoS de type qualitatif.

Bibliographie

[1] Mounir Lallali. *Modélisation et Test Fonctionnel de l'Orchestration de Services Web*. PhD thesis, L'école doctorale S&I en co-accréditation avec l'Université d'Evry-Val d'Essonne, France, 2009.

[2] Patrick Kellert and Farouk Toumani. Les Web services sémantiques. *I3 Journal*, 2004.

[3] Daniel Austin, Abbie Barbir, Christopher Ferris, and Sharad Garg. Web Services Architecture Requirements. W3C Working Group Note 11 February 2004. `https://www.w3.org/TR/wsa-reqs/`. Consulté le 10 Mars 2016.

[4] Hakim Boudjelaba. Sélection des Web Services Sémantiques. Master's thesis, Université A/Mira de Béjaïa, 2012.

[5] Oscar Mora. Conception et développement d'un service web pour la recherche d'objets immobiliers. Technical report, Université de Fribourg, Suisse, 2007.

[6] Refes Chabane. *Les services Web*, 2011.

[7] Zeyneb Kameche. Sélection des Web Services à Base Des Essaimes Particulaires. Master's thesis, Université Abou Bakr Belkaid de Tlemcen, 2011.

[8] Amina Bekkouche. Composition des Services Web Sémantiques À base d'Algorithmes Génétiques. Master's thesis, Université Abou Bakr Belkaid de Tlemcen, 2012.

[9] Abdelhamid Djelmoudi and Mohammed Belaredj. La sélection des services web à base des systèmes immunitaires artificiels (CLONALG). Master's thesis, Université Abou Bakr Belkaid de Tlemcen, 2012.

[10] Yahia Mohammed Dali. Sélection de services Web à base de QoWS. Master's thesis, Université Abou Bakr Belkaid de Tlemcen, 2011.

[11] Slimane Hammoudi and Denivaldo Lopes. *Introduction aux Services Web*.

[12] M. Chelbabi. *Découverte de Services Web Sémantiques : une Approche basée sur le Contexte*, 2006.

[13] Thomas Gruber. A translation approach to portable ontology specifications. Technical report, Knowledge Systems Laboratory, Stanford University, USA, 1993.

[14] Willem Nico Borst. *Construction of engineering ontologies for knowledge sharing and reuse*. PhD thesis, Université de Twente, Enschede, Pays-Bas, 1997.

[15] Rudi Studer, V. Richard Benjamins, and Dieter Fensel. Knowledge engineering : principles and methods. *Data & Knowledge Engineering*, 1998.

[16] Tim Berners-Lee. WWW Past & Future. `https://www.w3.org/2003/Talks/0922-rsoc-tbl/`. Consulté le 12 Mars 2016.

[17] Yassin Chabeb. *Contributions à la Description et la Découverte de Services Web Sémantiques*. PhD thesis, L'école doctorale S&I en co-accréditation avec l'Université d'Evry-Val d'Essonne, France, 2011.

[18] Imed Chouchani. *Utilisation d'un algorithme génétique pour la composition de services Web*. PhD thesis, Université du Québec à Montréal, Canada, 2010.

[19] Soheyb Ayad. Une approche basée agents pour la découverte sémantique des services Web. Master's thesis, Université Mohamed Khider de Biskra, 2009.

[20] FethAllah Hadjila. *Composition et interopération des services Web sémantiques*. PhD thesis, Université Abou Bakr Belkaid de Tlemcen, 2014.

[21] Pornpong Rompothong and Twittie Senivongse. A query federation of UDDI registries. In *ISICT '03 : Proceedings of the 1st international symposium on Information and communication technologies, Dublin, Ireland*, 2003.

[22] Massimo Paolucci, Takahiro Kawamura, Terry R. Payne, and Katia P. Sycara. Semantic Matching of Web Services Capabilities. In *ISWC '02 : Proceedings of the First International Semantic Web Conference on The Semantic Web, London, UK*, 2002.

[23] Naveen Srinivasan, Massimo Paolucci, and Katia P. Sycara. An efficient algorithm for OWL-S based semantic search in UDDI. In *SWSWPC : First International Workshop on Semantic Web Services and Web Process Composition, California, USA*, 2004.

[24] Umesh Bellur and Roshan Kulkarni. Improved Matchmaking Algorithm for Semantic Web Services Based on Bipartite Graph Matching. In *IEEE International Conference on Web Services (ICWS), Utah, USA*, 2007.

[25] Umesh Bellur, Harin Vadodaria, and Amit Gupta. Semantic Matchmaking Algorithms. Technical report, Department of Computer Science and Engineering, Indian Institute of Technology, Bombay, 2008.

[26] Harold Kuhn. The Hungarian Method for the Assignment Problem. *Naval Research Logistic Quarterly*, 1955.

[27] Kaarthik Sivashanmugam, Kunal Verma, Ranjit Mulye, and Zhenyu Zhong. *Speed-R : Semantic p2p environment for diverse Web services registries.*

[28] Ghazi Gharbi. Algorithme de sélection dans les applications a services : une approche basée sur la méthode d'analyse de concepts formels. Technical report, Université Joseph-Fourier, Grenoble, France, 2010.

[29] Shuping Ran. A model for Web Services Discovery with QoS. *SIGecom Exchanges*, 2003.

[30] Kangchan Lee, Jonghong Jeon, Wonseok Lee, Seong-Ho Jeong, and Sang-Won Park. QoS for web services : requirements and possible approaches. Technical report, W3C, Web Services Architecture Working Group, 2003.

[31] Liangzhao Zeng, Boualem Benatallah, Anne H.H. Ngu, Marlon Dumas, Jayant Kalagnanam, and Henry Chang. QoS-Aware Middleware for Web Services Composition. *IEEE Transactions on Software Engineering*, 2004.

[32] C. Jaeger Michael and Gero Mühl. Soft Real-Time Aspects for Service-Oriented Architectures. In *Eighth IEEE International Conference on E-Commerce Technology (CEC 2006) / Third IEEE International Conference on Enterprise Computing, E-Commerce and E-Services (EEE 2006) and Workshops, California, USA*, 2006.

[33] Yutu Liu, Anne H. Ngu, and Liang Z. Zeng. QoS Computation and Policing in Dynamic Web Service Selection. In *Proceedings of the 13th International World Wide Web Conference on Alternate Track Papers & Posters, New York, USA*, 2004.

[34] L. Taher, H. El Khatib, and R. Basha. A Framework and QoS Matchmaking Algorithm for Dynamic Web Services Selection. In *Proceedings of the Second International Conference on Innovations in Information Technology (IIT '05), Dubai, UAE*, 2005.

[35] Sathyavathy Poddy, Sneha Chandrababu, Uma Balasubramanian, S. Swamynathan, and T.V Geetha. Semantic and QoS based Web Service Selection using a Multi Agent System. In *Proceedings of the 2nd Indian International Conference on Artificial Intelligence, Pune, India*, 2005.

[36] Guangjun Guo, Fei Yu, Zhigang Chen, and Dong Xie. A Method for Semantic Web Service Selection Based on QoS Ontology. *Journal of Computers*, 2011.

[37] Kaijun Ren, Jinjun Chen, Tao Chen, Junqiang Song, and Nong Xiao. Grid-based Semantic Web Service Discovery Model with QoS Constraints. In *Proceedings of the Third International Conference on Semantics, Knowledge and Grid, Washington, USA*, 2007.

[38] Angus F. M. Huang, Ci-Wei Lan, and Stephen J. H. Yang. An Optimal QoS-based Web Service Selection Scheme. *Information Sciences*, 2009.

[39] Tapan Sen, Farhad M. E. Raiszadeh, and Parthasarati Dileepan. A Branch-and-Bound Approach to the Bicriterion Scheduling Problem Involving Total Flowtime and Range of Lateness. *Management Science*, 1988.

[40] Mohd Farhan Md Fudzee and Jemal H. Abawajy. QoS-based Adaptation Service Selection Broker. *Future Generation Computer Systems*, 2011.

[41] James D. McCaffrey. Using the Multi-Attribute Global Inference of Quality (MAGIQ) Technique for Software Testing. In *Proceedings of the 2009 Sixth International Conference on Information Technology : New Generations, Washington, USA*, 2009.

[42] Qi Yu and Athman Bouguettaya. *Foundations for Efficient Web Service Selection.* Springer Science + Business Media, 2009.

[43] C. J. Van Rijsbergen. *Information Retrieval.* Butterworths, London, 2nd edition, 1979.

Liste des figures

Liste des tableaux