N° d'ordre :

REPUBLIQUE ALGERIENNE DEMOCRATIQUE & POPULAIRE

MINISTERE DE L'ENSEIGNEMENT SUPERIEUR & DE LA RECHERCHE SCIENTIFIQUE

UNIVERSITE DJILLALI LIABES
FACULTE DES SCIENCES EXACTES
SIDI BEL ABBÈS

THESE
DE DOCTORAT EN SCIENCES

Présentée par
KAOUAN Moussa

Spécialité : Informatique
Option : Interopérabilité et intégration des systèmes D'informations dans le Web

Intitulée

Un référentiel partagé pour la découverte sémantique des services Web

Soutenue le 31 Mars 2022.
Devant le jury composé de :

Président :	*M. ADJOUDJ Reda*	*Professeur*	*UDL-Sidi Bel Abbès*
Examinateurs :	*M. KHATER Maamar*	*MCA*	*Univrsité de Saida*
	M. MALKI Abdelhamid	*MCA*	*ESI SBA*
	M. BERRABAH Djamel	*MCA*	*UDL-Sidi Bel Abbès*
Directeur de thèse :	*M. BOUCHIHA Djelloul*	*Professeur*	*CU-Naama*
Co-Directeur de thèse :	*M.BOUKLI HACENE Sofiane*	*Professeur*	*UDL-SBA*

Année universitaire 2021/2022

الحمد لله وحده الذي وفقنا لإتمام هذا العمل

أهدي هذا العمل إلى الوالديـــن الكريمـين

أنجزت هذا العمل بفضل عنايتها الدائمة و صبرها علي و دعائهمـــا

إلـى أمي العزيزة

إلى أبي رحمه الله الذي كانت نصائحه سببا في إرشادي.

إلى أخواتي و إلى زوجتي و أبنائي

إلى كل من ساهم من بعيد أو قريب في إنجاح هذا العمل

Je dédie ce travail

À mes parents

À ma femme

À toutes mes sœurs

À toute ma famille

À tous ceux qui m'ont encouragé et soutenu

Remerciements

Il me sera très difficile de remercier tout le monde car c'est grâce à l'aide de nombreuses personnes que j'ai pu mener cette thèse à son terme.

Je voudrais tout d'abord remercier grandement mon directeur de thèse, Prof. Bouchiha Djeloul, pour toute son aide. Je suis ravi d'avoir travaillé en sa compagnie car outre son appui scientifique, il a toujours été là pour me soutenir et me conseiller au cours de l'élaboration de cette thèse. Je voudrais également adresser mes sincères remerciements à Prof. Boukli Hacene Sofiane, pour son aide, ses remarques, et conseils de fond tout au long de ce travail.

Je remercie également Prof. Benslimane Sidi Mohammed pour son aide et son soutien.

J'adresse tous mes remerciements à Monsieur Adjoudj reda, Professeur à l'Université de Sidi Bel Abbès, Monsieur Khater Maamar, Maitre de conférences à l'Université de Saida, Monsieur Malki Abdelhamid, Maitre de conférences à l'Ecole Supèrieure en Informatique de Sidi Bel Abbès, ainsi qu'à Monsieur Berrabah Djamel, Maitre de conférences à l'Université de Sidi Bel, de l'honneur qu'ils m'ont fait en acceptant d'être examinateurs de cette thèse.

Ma reconnaissance à mon défunt père.

Mes sincères remerciements à ma mère, et toute ma famille, en particulier à mon épouse et mes sœurs.

Enfin, je tiens à remercier toutes les personnes qui ont participé de près ou de loin à la réalisation de ce travail.

Résumé

Les ontologies ont marqué leur présence dans plusieurs domaines de recherche afin de résoudre les problèmes de modélisation des connaissances, de l'intelligence Artificielle, de la classification, du regroupement et plus précisément de l'ingénierie des connaissances. Dans le domaine de services Web, les ontologies interviennent surtout pour décrire les interfaces de services Web en raison d'automatiser les processus fonctionnels de cette technologie (OWL-S, WSMO, WSDL-S, etc.). Malheureusement, cette intervention reste à ce jour insuffisante. Soutenir cette description par l'ajout d'une ontologie partagée (base de connaissance) peut améliorer les performances de la technologie services Web (Kaouan et al., 2017). La construction manuelle d'une ontologie est une tâche fastidieuse, coûteuse et qui exige la présence d'un expert de domaine. Une méthodologie de construction ontologique automatique sera une solution intéressante et moins coûteuse en termes de temps et de prix. Dans cette thèse, nous proposons d'introduire une couche connaissance (SO) à la technologie de service Web ; ainsi, nous proposons d'utiliser les algorithmes de regroupement pour créer un réseau ontologique (construire l'ontologie SO) à partir d'un corpus de services Web sémantique. Le référentiel créé peut être utilisé pour le stockage et la découverte des services Web dans un environnement distribué et intelligent.

Mots clés : Service Web, UDDI, Référentiel partagé, Ontologie, Découverte des services, Stockage des services, Algorithme de classification.

Abstract

The ontology has marked its presence in several research fields in order to address issues in knowledge modeling, Artificial Intelligence, classification, clustering and more specifically in knowledge engineering. In the Web service field, ontologies are mainly used to describe Web service interfaces due to automating the functional processes of this technology. This is well known as semantic Web services, such as OWL-S, WSMO and WSDL-S. Unfortunately, this intervention remains insufficient to date. So, supporting this description by adding a shared ontology (knowledge base) can improve the performance of the Web Service technology (Kaouan et al., 2017). Manual construction of ontology is a tedious and expensive task which requires domain expert intervention. An automatic ontological construction process will be interesting and less costly solution in terms of time and money. In this thesis, we propose to introduce a knowledge layer to the Web service technology ; thus, we propose to use clustering algorithms to create an ontological network (to build the SO) from a semantic Web service corpora. The created repository can be used for storage and discovery of Web Services in a distributed and intelligent environment.

Keywords: Web service, UDDI, Shared repository, Ontology, Service discovery, Service storage, Classification algorithm.

ملخـــص

تميزت الأنطولوجيا بوجودها في العديد من مجالات البحث من أجل معالجة القضايا في تصميم المعرفة، الذكاء الاصطناعي، التصنيف، التجميع، وبشكل أكثر تحديدًا في هندسة المعرفة. في مجال خدمة الويب، تُستخدم الأنطولوجيا بشكل أساسي لوصف واجهات خدمة الويب لأجل تنفيذ العمليات الوظيفية لهذه التكنولوجيا بشكل آلي. يُعرف هذا باسم خدمات الويب الدلالية SWS، مثل: OWL-S ، WSMO و WSDL-S . للأسف، لا يزال العمل في هذا المجال غير كافٍ حتى الآن؛ لذلك، فإن دعم هذا الوصف عن طريق إضافة أنطولوجيا مشتركة (قاعدة معرفة) يمكن أن يحسن أداء تقنية خدمة الويب (Kaouan et al. 2007). يعد البناء اليدوي للأنطولوجيا مهمة شاقة ومكلفة وتتطلب تدخل خبير في المجال ؛ سيكون البناء الأنطولوجي بشكل آلي حلاً مثيرًا للاهتمام وأقل تكلفة من حيث الوقت والمال. في هذه الأطروحة، نقترح تقديم طبقة معرفية لتكنولوجيا خدمة الويب ؛ وبالتالي، فإننا نقترح استخدام خوارزميات التجميع لإنشاء شبكة أنطولوجية (SO) من خلال مخزن به مجموعة كبيرة من خدمات الويب الدلالية. يمكن استخدام الأنطولوجيا التي تم إنشاؤها لتخزين واكتشاف خدمات الويب في بيئة موزعة وذكية.

الكلمات المفتاحية: خدمة الويب، UDDI، نظام مرجعي مشترك، أنطولوجيا، اكتشاف الخدمات، تخزين الخدمات، خوارزمية التجميع.

Table des matières

Table des figures

Liste des tableaux

Liste des algorithmes

Liste des Acronymes et des Abréviations

B2B Business to Business

B2C Business to Customer

OWL-S Web Ontology Language for Web Services

SO Service Ontology

SOA Service Oriented Architecture

SOC Service-Oriented Computing

SWS Semantic Web Services

UDDI Universal Description Discovery and Integration

WSDL Web Services Description Language

WSMO Web Service Modeling Ontology

XML eXtensible Markup Language

Chapitre 1

Introduction Générale

Sommaire

1.1 Contexte

Depuis quelques années, l'intégration inter-applications est devenue l'un des principaux intérêts de l'industrie des applications logiciels. Cela a entraîné la croissance émergente du paradigme des architectures orientées services (SOA), qui sont devenus les références principales en termes d'architectures logicielles distribuées. L'orientation Service est un paradigme de conception destiné à la création des services autonomes et indépendants pour réaliser des tâches bien définies. Ces services peuvent être réutilisés et combinés dynamiquement pour répondre aux nouveaux besoins, et donner des solutions aux nouveaux problèmes qui ne peuvent pas être résolus par l'utilisation d'un seul service.

Les services Web sont une technologie émergente implémentant l'architecture orientée service. Ce sont des composants softwares qui fournissent des fonctionnalités accessibles via des protocoles standardisés du Web. Les services Web ont marqué les méthodes actuelles d'ingénierie Web, et sont universellement pris en charge par les fournisseurs informatiques et les utilisateurs. En bref, ils sont des composants logiciels interopérables pouvant être utilisés dans l'intégration d'applications et de développement d'applications à base de composants.

Les services Web sémantique sont des services Web dotés de descriptions sémantiques. Cette sémantique est apportée grâce aux ontologies, l'une des technologies importantes du Web sémantique.

La découverte des services Web est un domaine fondamental de la recherche en informatique. De nombreux chercheurs ont mis l'accent sur la découverte de services Web via un système centralisé (Paolucci et al., 2002a,b). Bien que les registres centralisés puissent fournir des méthodes plus au moins efficaces pour la découverte de services Web, ils souffrent d'un nombre de problèmes classiques connus dans les systèmes centralisés comme le seul point centralisé (point d'échec), le coût élevé de la maintenance et la dégradation des fonctionnalités. Actuellement, et en conséquence de la demande croissante d'utilisation des technologies services Web, une série de questions se posent concernant les procédures de découverte, les méthodes, et les architectures utilisées pour la technologies de services Web.

1.2 Problématique

Comme les services Web ont commencé à se développer à travers l'Internet, les utilisateurs doivent être en mesure d'accéder et de partager efficacement les services Web. La production et l'interopérabilité d'un plus grand nombre de services Web ont conduit à l'apparition de nouveaux besoins sur la façon dont les services peuvent être publiés, découverts, ou utilisés. Par conséquent, des mécanismes sont nécessaires pour une sélection efficace de l'instance du service Web appropriée en termes de facteurs de qualité et de performance.

L'UDDI (Universal Description, Discovery and Integration) joue un rôle très important dans le concept de service Web. Il peut être utilisé pour publier et rechercher des services. La plupart des modèles actuels de l'UDDI sont centralisés de sorte que leur rendement diminue s'il y a trop de services inscrits ou interrogés. La structure centralisée est moins robuste et exprime une mauvaise interopérabilité. L'UDDI actuel est un service annuaire passif, ce qui rend la détection des changements de service très difficile. Le futur UDDI tentera de réduire les inconvénients de l'approche centralisée. Dans ce contexte, plusieurs solutions ont été proposées. Une solution consiste à répliquer toutes les informations pour les mettre sur différents registres. Bien que cette solution améliore temporairement les performances de l'UDDI, elle implique un coût élevé en matière de déploiement et de maintenance. La réplication des données de l'UDDI est une approche non évolutive et inconsistante. L'utilisation des technologies distribuées (systèmes P2P) semble une solution plus performante. Cependant, dans l'absence d'un modèle commun entre les nœuds, l'interopérabilité devient un objectif difficile.

1.3 Motivation

La technologie des services Web utilise un ensemble de normes, notamment SOAP, WSDL et UDDI. WSDL permet une description syntaxique des services Web. Des projets de recherche, tels que OWL-S, WSMO et SAWSDL, ont montré que les descriptions sémantiques peuvent permettre une découverte plus précise que la représentation syntaxique, mais jusqu'à présent, aucune solution complète n'existe pour offrir un accès facile et évolutif aux services Web (Della Valle et al., 2008). Ces solutions ne décrivent que les interfaces des services Web, soit en augmentant le WSDL avec une ontologie de domaine (WSDL-S), soit en créant une structure plus riche (WSMO, OWL-S) utilisant également les ontologies. L'ontologie, qui est une représentation partagée et explicite des connaissances, est plus puissante que la description syntaxique. Le premier à avoir introduit le concept d'ontologie dans l'In-

telligence Artificielle (IA) est McCarthy (1980) qui a affirmé que les concepteurs de systèmes intelligents devaient énumérer tout ce qui existe avec les ontologies (McCarthy, 1980). Ainsi, dans cette thèse, nous introduisons une couche de connaissances (SO) dans la technologie des services Web. Cette ontologie est construite en appliquant l'algorithme de clustering k-means. Ainsi, chaque cluster correspondra à un concept de l'ontologie (TBox), et les services Web représenteront les instances de l'ontologie (ABox assertions).

1.4 Objectif

La distribution des informations sur différents sites nécessite l'existence d'un modèle partagé qui peut être utilisé comme un schéma pour répartir/reconstruire les informations, l'astuce absent dans les systèmes centralisés. L'ancienne architecture de découverte des services ne peut être étendue dans les systèmes distribués. La réussite de la distribution des annuaires des services est garantie par la construction d'une nouvelle architecture ayant un modèle référentiel partageable.

Dans le cadre de notre travail, nous allons nous intéresser aux problèmes de la découverte des services Web. En particulier, nous allons nous focaliser sur la recherche d'une solution pour surmonter les questions et les problèmes des architectures centralisées.

Nous proposons, en résultat, un modèle référentiel qui peut être partagé entre les différents UDDIs. Ce modèle référentiel (ontologie) a été conçu dans le but d'assister l'UDDI à stocker et à découvrir les services Web. L'utilisation d'un référentiel partagé peut unifier l'esprit de la découverte des services Web entre les UDDIs et peut satisfaire un environnement collaboratif et interopérable.

1.5 Démarche de la recherche et principales contributions

Pour remplir l'objectif fixée, il fallait tracer une feuille de route, ce qui nous a permis d'apporter les contributions suivantes :

- L'élaboration d'un état de l'art détaillé survolant les différentes approches proposées pour la découverte des service Web.
- L'élaboration d'un état de l'art sur les annuaires distribués.
- L'élaboration d'un état de l'art sur les travaux qui ont entamé le problème de regroupement des services Web.

- Proposer une nouvelle approche en couches pour la découverte des services Web sémantiques basée sur un référentiel partagé qui représente la couche connaissance pour les service Web (Ontologies).
- Proposer et implémenter un algorithme pour la découverte (Kaouan et al., 2017, 2015).
- Proposer et implémenter une méthode semi-automatique pour construire cette ontologie. Nous proposons particulièrement les algorithmes de regroupement pour construire les concepts ontologique (Kaouan et al., 2020).

1.6 Organisation de la thèse

Le reste de la thèse est divisé en trois grandes parties. La première concerne le Background qui décrit les principaux concepts relatifs au contexte général de notre travail. La deuxième partie est un état de l'art sur les principaux travaux qui abordent les solutions distribuées de la découverte des services. Finalement, la dernière partie concerne notre contribution ainsi que son implémentation. Ces parties sont détaillées en plusieurs chapitres :

- **Partie I :**

  Le Chapitre 2 présente l'approche orientée services, commençant par une définition de l'architecture SOA, ainsi que son élément constructif de base, qui est le service, et en montrant la pile architecturale qui énonce les éléments qui pourraient être observés dans cette approche.

  Ensuite, **Le Chapitre 3** présente la technologie des services Web, qui est la technologie la plus utilisée pour implémenter l'architecture SOA. Ainsi, ce chapitre présente ses standards de base, tels que WSDL, SOAP et UDDI.

  Dans **le Chapitre 4**, nous présentons succinctement un état de l'art sur les principes et l'architecture du Web sémantique. Ainsi, nous présentons l'utilisation de la sémantique pour les Services Web montrant la contribution du Web sémantique à la recherche de solutions pour les services Web sémantiques. La dernière partie de ce chapitre est consacrée à la présentation de solutions pour les services Web sémantiques.
- **Partie II :**

  Le Chapitre 5 fournit une vision particulière sur la découverte des services Web. La première partie du chapitre, explique le processus de la découverte et expose les facteurs qui influent sur ce processus. Par la suite, le chapitre détaillera la solution de la découverte « centralisée».

  Le Chapitre 6 est un état de l'art sur la découverte distribuée des services

Web. Le chapitre commence par la présentation du problème, en montrant les caractéristiques importantes d'un UDDI distribué, et il se terminera par l'exposition des principaux travaux et approches distribuées présentés dans la littérature.

– **Partie III :**

Le Chapitre 7 explique notre approche pour résoudre le problème de la distribution de la découverte de services Web basée sur la création d'une ontologie. Cette dernière va être employée comme un modèle partagé utilisé pour regrouper, stocker et rechercher les services Web.

Le Chapitre 8 décrit les prototypes implémentés pour construire l'ontologie et le Matching proposé. Aussi, le chapitre expose deux parties d'évaluations chacune pour une partie d'implémentation. La dernière partie de ce chapitre expose une étude comparative entre l'approche proposée et les principaux travaux cités dans l'état de l'art.

Finalement, le manuscrit se termine par une conclusion générale qui conclut nos travaux et énumère quelques perspectives de recherche.

Première partie

BACKGROUND

Chapitre 2

L'architecture Orientée Services (SOA)

Sommaire

2.1 Introduction

Au cours de ces dernières années, les entreprises expriment des intérêts progressifs pour s'intégrer dans l'inter-applications. Cela a entraîné la croissance émergente du paradigme de l'architecture orientée services (SOA), et de la technologie des services Web, qui sont devenus les principaux références en termes d'architectures logicielles distribuées. En outre, L'architecture orientée services (SOA) est un style architectural qui favorise le partage et la réutilisation des composants logiciels sous la forme de services publiés.

Ces services peuvent être réutilisés et combinés dynamiquement pour répondre aux nouveaux besoins, et donner des solutions aux nouveaux problèmes qui ne peuvent pas être résolus par l'utilisation d'un seul service.

Ce chapitre présente l'Architecture Orientée Services (SOA). Il explique les modèles et les spécifications de cette architecture comme modèle de base abstrait permettant de fonder les moyens pour souscrire vers les systèmes interopérables. Le chapitre expose aussi les définitions les plus célèbres qui expliquent les notions de service et d'(SOA). Ainsi, il explique la notion de service avec leurs composants et les interactions fonctionnelles entre les acteurs de l'architecture comprenant comment cette architecture fonctionne et comment elle pourrait être mise en œuvre et utilisée.

2.2 L'architecture Orientée Services

L'architecture orientée services (SOA) est une approche de la construction de systèmes qui se concentre sur la construction d'applications à partir d'une combinaison de services commerciaux. Cette approche offre la possibilité de créer des entreprises dynamiques qui disposent d'une flexibilité maximale et d'un temps de réponse minimal. A l'origine, l'architecture Orientée Services (SOA) a été inventée en 1996 par Gartner Group. Elle décrit un style multi niveau qui aide les organisations à partager la logique et les données entre plusieurs applications(Schulte and Natis, 1996). Depuis ce temps « 1996 », plusieurs publications ont redéfini ce terme. La plupart de ces définitions se concentrent sur l'aspect fonctionnel de la SOA, d'autres combinent les aspects commerciaux et techniques (Papazoglou, 2008; Erl, 2005), alors que d'autres travaux ont mis l'accent sur l'aspect métier (Sharpe and Margolis, 2007). Dans la section suivante, nous exposons les principales définitions de SOA.

2.2.1 Définitions

En réalité, il existe plusieurs définitions proposées pour définir SOA ; tout dépend de l'aspect étudié. Chaque définition émet son point de vue définissant SOA, mais toutes, elles conviennent que SOA propose une architecture pour garantir la réutilisation et l'interopérabilité en se basant, bien sûr, sur la notion de service. Dans ce qui suit, nous exposons certaines définitions proposées pour l'explication de SOA.

Définition 1 : ***"SOA est un ensemble de composants qui peut être invoqué, et dont la description peut être publiée et découverte"***(Booth et al., 2003). Cette brève définition présente, de façon très simple, les trois fonctionnalités réalisées dans l'architecture SOA, à savoir l'invocation, la publication et la découverte, et ce sans employer les termes techniques : services ou architecture.

Définition 2 : ***"SOA est un paradigme pour l'organisation et l'utilisation des capacités distribuées, qui peuvent êtres sous le contrôle de divers propriétaires. Elle fournit un moyen cohérent qui permet d'offrir, de découvrir et d'utiliser des ressources pour produire les résultats voulus"*** (MacKenzie et al., 2006). Cette deuxième définition pointe sur la vue distribuée de l'architecture où les organisations partagent les ressources logicielles (services) pour réaliser l'objectif voulu. A partir delà, on peut dire que SOA est un paradigme pour concevoir, implémenter et déployer des systèmes d'information,

pour qu'un système soit créé à partir de composantes séparées, implémentant des fonctions métiers discrètes. Ces composantes peuvent être réparties géographiquement à travers plusieurs entreprises, et peuvent être reconfigurées et réutilisées dans de nouveaux processus métiers selon les besoins (Rosen et al., 2012).

Définition 3 : ***"L'architecture SOA permet l'intégration d'applications et de ressources de manière flexible, en représentant chaque application ou ressource sous la forme d'un service exposant une interface standardisée, permettant à un service d'échanger des informations structurées (messages, documents, objets métiers), en coordonnant et en organisant les services, afin d'assurer qu'ils puissent être invoqués, utilisés et changés efficacement"***(Hack and Lindemann, 2007). Dans cette définition, l'auteur expose l'intégration d'applications comme objective de l'architecture SOA, en échangeant les informations à travers les interfaces standards de chaque service. Par cela, la vue application est masquée définitivement, le fait que les applications soit intégré dans une seule et unique vue appelée service. Ces services peuvent interopérer automatiquement entre eux pour réaliser d'autres besoins. En effet, l'approche SOA pousse l'interopérabilité entre les systèmes. Elle devient alors un support typique pour réaliser l'interopérabilité (Jardim-Goncalves et al., 2006).

Définition 4 : ***"Une architecture SOA est une architecture d'applications au sein de laquelle toutes les fonctions sont définies en tant que services indépendants, comprenant des interfaces bien définies et qui peuvent être sollicitées dans des séquences définies pour créer des processus métier"***(Rosen et al., 2012). Par cette définition, les principaux éléments de l'architecture SOA avec leurs relations sont présentés :

- toutes les fonctions sont définies par une seule représentation appelée service.
- Ces services sont totalement indépendants et autonomes.
- L'interaction avec les services se fait uniquement à travers une interface bien définie.
- Les interfaces peuvent être appelées indépendamment de leur emplacement.
- Les services peuvent se composer et s'intégrer pour concevoir d'autres services dites composites.

Définition 5 : ***"SOA est un style architectural dont l'objectif est d'obtenir un couplage faible entre les agents logiciels en interaction."***(He, 2003)

2.3 La notion de service

L'une des abstractions de base pour la mise en place de la SOA est le concept de service. Dans la littérature, on trouve plusieurs définitions de ce concept. Parmi ces définitions, on peut citer :

Définition 1 : ***"Un service est une fonction qui est bien définie, autonome, et ne dépend pas du contexte ou de l'état des autres services"***(Thuraisingham, 2008).

Définition 2 : ***"Un service est un processus par lequel le fournisseur remplit une mission pour son client de manière à créer une valeur pour chacune des deux parties intervenantes"***(Lau et al., 2011).

Définition 3 : ***"Un service est une unité de travail effectuée par un fournisseur de services pour obtenir les résultats finaux souhaités pour un consommateur de services"***(He, 2003).

Dans le cas général, un service peut être défini comme une implémentation d'un logiciel autonome, développé, déployé, géré et maintenu indépendamment de son environnement, et qui implémente des fonctions spécifiques, intégrables et nécessaires au fonctionnement de l'entreprise (MacKenzie et al., 2006). La fonctionnalité d'un service est définie par une interface basée sur des standards. Cette interface peut être soutenue par de multiples implémentations. Cela implique que le service n'est pas seulement une structure de programmation ou un ensemble d'API, mais est plutôt une unité architecturale (unité de conception, de mise en œuvre et de maintenance) utilisée pour l'implémentation des solutions d'entreprise, et il est accessible via une interface bien définie.

2.4 Composantes d'un service

Un service est composé de deux parties principales (voir Figure 2.1). La partie supérieure représente l'interface du service, alors que la partie inférieure représente son l'implémentation. Dans une architecture SOA, l'interface du service est clairement séparée par rapport à son implémentation.

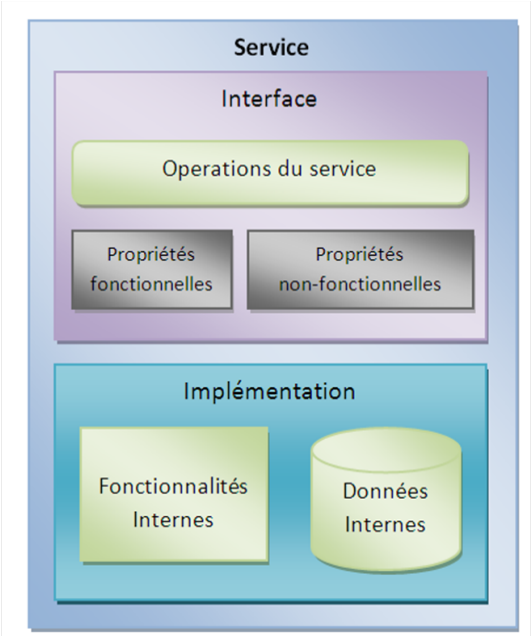

Figure 2.1 – Les composantes d'un service

2.4.1 L'interface

L'interface représente la partie abstraite du service. Elle indique **que fait le service ?** en représentant les opérations qui lui implémente, les paramètres qui lui sont passées, les résultats obtenus en retour, et les protocoles utilisés lors de la communication avec les services. Cette notion d'interfaces est essentielle dans l'architecture SOA. Elle permet l'indépendance vis-à-vis les technologies, ce qui résulte un environnement interopérable permettant la réutilisation et l'intégration des services. Typiquement, un service contient plusieurs différentes opérations reliées entre eux. Pratiquement, ces opérations décrivent le service en définissant la fonctionnalité métier réalisée à travers ce service. Chaque opération possède (Rosen et al., 2012) :

- **Propriétés fonctionnelles :** définissent les opérations décrites dans l'interface et leurs paramètres. Chaque opération accepte en entrée un ensemble de paramètres (appelé entrées), et retourne en sortie, après exécution, un autre ensemble de paramètres (appelé sorties). Les propriétés fonctionnelles décrivent l'activité accomplie par une opération, et les différents types de données de ses paramètres d'entrée et de sortie.
- **Propriétés non fonctionnelles :** décrivent les paramètres de configuration du service. Dans la littérature, les propriétés non fonctionnelles sont souvent considérées comme des qualités d'une application. D'autres qualificatifs utilisés souvent pour les propriétés non fonctionnelles comme : les contraintes, les

attributs de qualité, les objectifs de qualité et les exigences de qualité du service. Elles couvrent le niveau de sécurité, le contrôle d'accès, le cryptage des données, et les performances des opération que ce soit en termes de durée où du coût d'exécution.

2.4.2 L'implémentation

L'implémentation représente la partie concrète et opérationnelle du service. Elle est basée sur des applications existantes ou sur un code écrit spécifiquement pour le service ou bien par composition d'autres services. Ce qui est important ici, est que le consommateur doit voir seulement ce que fait le service, et non jamais son implémentation. Le fournisseur du service est libre de modifier l'implémentation, tant que cette modification n'affecte pas l'interface et le comportement du service.

2.5 Les caractéristiques d'un Service

En complément de la structure discutée dans la section précédente, les bons services doivent justifier certaines caractéristiques (Belfedhal and Malki, 2011). Nous citons ci-après les principales propriétés :

- Modularité : En SOA, les services sont identifiés à partir d'un processus métiers décomposé en petits modules. Alors, les services doivent être modulaires et autonomes. Aussi, ils peuvent être composés pour créer d'autres services dites composites ;
- L'encapsulation : Cacher les détails techniques des services est une propriété essentielle pour les services. Cette caractéristique, en particulier, permet la séparation entre l'implémentation et l'interface ;
- Le couplage faible : Cette caractéristique exprime la minimisation de dépendances entre les services. Les services faiblement couplés peuvent être indépendants en interagissant avec d'autres ;
- La réutilisabilité : Elle permet de rendre les services partageables et réutilisables comme blocs de construction pour répondre à une demande, ou pour réaliser d'autres services composites ;
- Découverte et liaison dynamiques : Les services sont découverts dynamiquement grâce à l'utilisation d'un annuaire de service. Après la découverte appropriée d'un service, le client sera lié dynamiquement à son fournisseur. La liaison dynamique augmente le couplage faible et permet des fonctionnalités supplémentaires, telles que la médiation ;

- L'auto description : Le contrat du service contient une description complète de l'interface du service, de ses activités, et de ses paramètres d'entrée et de sortie. Le contrat peut également avoir des pré- et post- conditions et des contraintes sur les opérations ;
- La composabilité : Les services peuvent être composés à partir d'autres services atomiques et, à leurs tours, les services composites peuvent êtres combinés avec d'autres services pour composer de nouveaux services ou des processus métiers ;
- Indépendants de l'emplacement, des langages, et des protocoles : Les services sont conçus d'une façon qui rend leur emplacement transparent, et les rend indépendant des protocoles et des plateformes. En d'autres termes, ils sont accessibles à tout utilisateur autorisé, sur n'importe quelle plateforme, depuis n'importe quel endroit.

2.6 Les principaux composants de l'Architecture SOA

2.6.1 Service

un service est destiné à une fonction encapsulée dans un composant que l'on peut interroger à l'aide d'une requête composée d'un ou plusieurs paramètres. En conséquence, le service doit fournir une réponse après son exécution. Idéalement, chaque service doit être indépendant des autres afin de garantir la réutilisabilité et l'interopérabilité.

2.6.2 Client

Le client (où demandeur de service) est une entité constituée d'un ou plusieurs individus qui peuvent être structurés ou non (Lau et al., 2011). Le client est lié dynamiquement à un fournisseur. La liaison dynamique augmente le couplage faible et permet d'assurer d'autres fonctionnalités supplémentaires, telles que la médiation.

2.6.3 Fournisseur

Le fournisseur est un individu ou groupe d'individus structuré comme une entité responsable qui possède une certaine compétence, assure un ensemble de connaissances, et il possède une structure capable de mener à bien certaines tâches (Lau et al., 2011). La structure peut être une personne, une entreprise, un réseau social,

une agence gouvernementale, une organisation à but non lucratif ou toute combinaison convenue des éléments précités.

2.6.4 Registre de services

Il est appelé aussi annuaire. Les annuaires, tels qu'ils sont déjà déployés dans d'autres domaines, représentent une base de données permettant de diffuser certaines informations. Il est accédé surtout en lecture qu'en écriture pour chercher et retrouver facilement des personnes ou des ressources. Dans SOA, le registre joue un rôle d'intermédiaire entre les fournisseurs des services et les clients, où il stocke l'ensemble des descriptions de services déclarées par les fournisseurs. Il permet aussi aux clients de rechercher et de sélectionner le service qui leur sera utile.

2.6.5 Description du service

Elle consiste à décrire les paramètres d'entrée du service avec leurs format et le type des données retournées. Cette description est importante lors des processus SOA, spécialement pour la découverte.

2.6.6 La publication et la découverte

La publication et la découverte représentent les deux opérations les plus élémentaires de la technologies SOA. La publication consiste à enregistrer dans le registre (annuaire de service) les services disponibles pour les utilisateurs, tandis que la notion de découverte recouvre la possibilité d'accéder et de rechercher des services utiles parmi ceux qui ont été déjà enregistrés et publiés.

2.6.7 L'invocation

Après avoir trouvé les services les plus précieux, les utilisateurs doivent l'invoquer auprès des fournisseurs. Donc, l'invocation consiste à la connexion et à l'interaction du client avec le service à l'aide d'une infrastructure réseau distribuée.

2.7 L'architecture fonctionnelle SOA

En résumé, SOA s'appuie sur une architecture fonctionnelle triangulaire (Voir la Figure 2.2), d'où on retrouve trois acteurs et trois opérations. Les acteurs fournisseurs de services, demandeurs de services et registre de services agissent entre eux les opérations de publication, de découverte et d'invocation. Cette collaboration

assure la communication des entreprises avec leurs clients ou partenaires. En plus, cette technologie offre les mécanismes qui permettent de traiter les problèmes de l'interopérabilité.

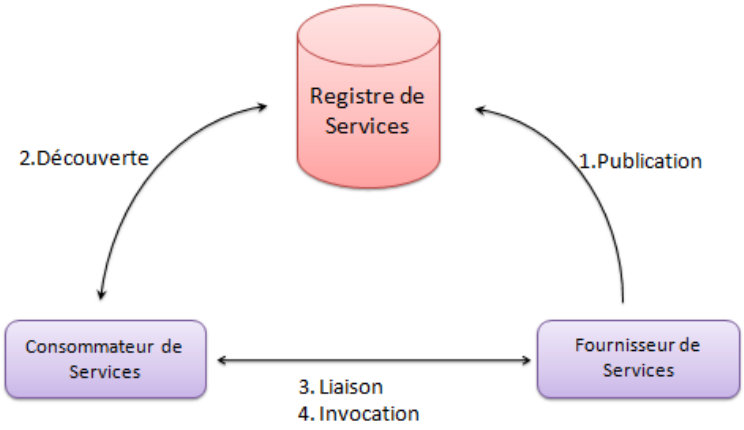

Figure 2.2 – Les éléments de base de l'architecture SOA

2.8 Pile architecturale de SOA

La pile architecturale (Endrei et al., 2004) énonce les éléments qui pourraient être observés dans une architecture orientée services. Ces éléments peuvent être classés en deux catégories (Voir la Figure 2.3). La partie gauche de la pile adresse la première catégorie, dite fonctionnelle, qui montre les différente couches employés dans SOA. La deuxième catégorie, illustrée dans la partie droite, montre les qualités soutenant certains aspects du service. Dans ce qui suit, nous détaillons les deux catégories.

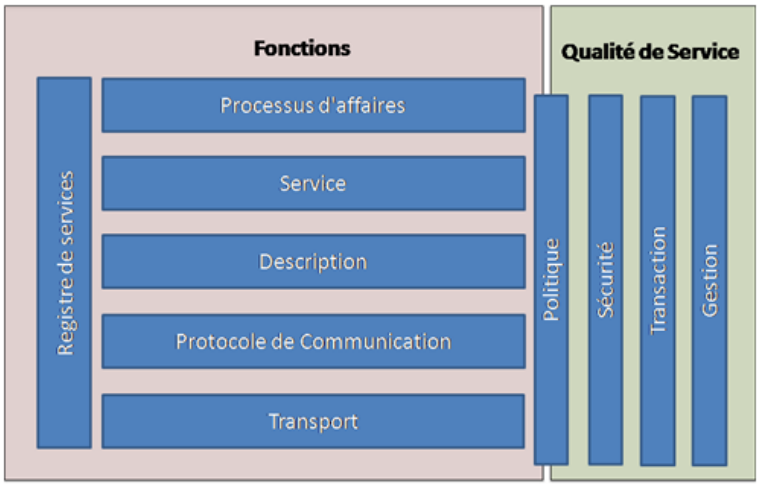

Figure 2.3 – La pile de l'Architecture Orientée Services

2.8.1 Les aspects fonctionnels

Cette catégorie décrit les différentes couches proposées durant l'utilisation des services depuis sa description ou sa composition jusqu'à l'acheminement vers les demandeurs. Ces aspects fonctionnels incluent les couches suivantes :

- **La couche processus d'affaires :** le terme processus d'affaires est une traduction française de Business Process. Alors, cette couche comprend les techniques utilisées pour composer de nouveaux services à partir des services existants avec un ensemble particulier de règles pour répondre à une exigence métier, cela implique : définir un ordre de collaboration entre une collection de services dont le but est d'atteindre l'objectif souhaité, puis de l'exécuter.
- **La couche Service :** cette couche décrit un service réel et atomique qui est mis à la disposition pour être utilisé.
- **La couche description :** à travers cette couche, une structure ou un modèle doit être adopté pour décrire ce que représente le service, comment il doit être invoqué et quelles données sont nécessaires pour l'invoquer avec succès.
- **La couche protocole de communication :** au niveau de cette couche, le mécanisme de liaison et de communication entre le fournisseur de services et le consommateur doit être exposé pour assurer ce contact en toute simplicité.
- **La couche Transport :** cette couche représente les mécanismes employés pour le déplacement applicatif des demandes de services au fournisseur, et les réponses au consommateur de services.

Les aspects de qualité, ou ce que nous l'appelons souvent les aspects non-fonctionnels, comprennent :

- **La politique :** c'est un ensemble de conditions ou de règles selon lesquelles un fournisseur de services met le service à la disposition des consommateurs. Il y a des aspects de la politique qui sont fonctionnels, et des aspects qui se rapportent à la qualité du service ; nous avons donc la fonction politique dans les domaines fonctionnels et de la qualité du service.
- **La sécurité :** c'est l'ensemble de règles qui peuvent être appliquées à l'identification, à l'autorisation et au contrôle d'accès des consommateurs appelant des services.
- **La transaction :** c'est l'ensemble des attributs qui peuvent être appliqués à un groupe de services pour fournir un résultat cohérent. Par exemple, si un groupe de trois services doit être utilisé pour exécuter une fonction commerciale, tous doivent être exécutés ou aucun ne doit être terminé.
- **La gestion :** c'est l'ensemble des attributs qui peuvent être appliqués à la gestion des services fournis ou consommés.

2.9 Conclusion

Dans ce chapitre, nous avons présenté l'architecture SOA de manière abstraite : sa définition, ses différents acteurs, ainsi que ses composants. Également, nous avons figuré la pile architecturale de SOA, et nous avons énuméré ses couches. Dans le chapitre qui suit, nous présenterons l'aspect service Web, dans lequel nous exposons sa définition, ses standards recommandés et son architecture.

Chapitre 3

Service Web

Sommaire

3.1 Introduction

Dans le chapitre précédant, nous avons exposé l'Architecture Orientées Services (SOA). SOA a proposé la séparation entre l'implémentation et l'exécution des programmes. Pour cela, l'architecture propose un paradigme de conception destiné à la création des services autonomes en se basant sur le concept service. Un service peut être réutilisé et combiné dynamiquement pour répondre aux nouveaux besoins. En réalité, SOA n'est qu'une architecture abstraite qui recommande des solutions générales loin de toutes solutions technologiques. La solution la plus utilisée pour implémenter SOA est la technologie du service Web. Les services Web reprennent les recommandations de SOA et se reposent sur les technologies standards de l'infrastructure d'Internet qui sont totalement indépendantes du matériel, ou du langage de programmation utilisé, ce qui permet de les mettre en œuvre facilement.

Dans ce chapitre, nous présentons la technologie du service WWeb, nous citons ses définitions les plus employées, et nous présentons ses standards et sa pile de programmation.

3.2 Définitions

En réalité, SOA n'est qu'une architecture abstraite. Les services Web sont une nouvelle technologie pour la création des applications accessibles à travers l'infrastructure Web. Plusieurs définitions existent dans la littérature pour définir cette technologie. Parmi ces définitions, nous avons choisi les trois définitions suivantes :

Définition 1 : Selon le W3C, un service Web est : ***"Un système logiciel conçu pour permettre les interactions machine à machine d'une façon interopérable à travers un réseau. Il a une interface décrite dans un format traitable par les machines (en particulier WSDL). D'autres systèmes peuvent interagir avec le service Web de la manière prescrite par sa description, en utilisant des messages SOAP, typiquement transmises via le protocole HTTP avec une sérialisation XML en conjonction avec d'autres normes liées au Web"***(Lévy, 2003).

Définition 2 : ***"Un service Web est une application, ou un composant logiciel qui vérifie les propriétés suivantes :***

- ***il est identifié par un URI ;***
- ***ses interfaces et ses liens peuvent être décrits en XML ;***
- ***sa définition peut être découverte par d'autres services Web ;***
- ***il peut interagir directement avec d'autres services Web à travers le langage XML en utilisant des protocoles Internet standards."***

(Cerami, 2002).

Définition 3 : ***"Un service Web est une interface qui décrit un ensemble d'opérations accessibles par le réseau via une messagerie XML normalisée. Un service Web exécute une tâche spécifique ou un ensemble de tâches.Un service Web est décrit à l'aide d'une notation XML formelle standard, appelée description de service, qui fournit tous les détails nécessaires pour interagir avec le service, y compris les formats de message, les protocoles de transport et l'emplacement. Les descriptions des services Web sont exprimées en WSDL"*** (Gottschalk et al., 2002a).

Donc, un service Web est une application logiciel exécutée telle qu'elle est créée la première fois sur sa plateforme, mais elle peut être invoquée dans le Web à travers sa description. Cette migration vers le Web a été réalisée grâce à certains langages d'encapsulation recommandés, en particulier WSDL, SOAP et UDDI. Le dénominateur commun le plus important entre ces langages est XML, ce qui facilitera l'échange de

données, la collaboration entre les applications distribuées, et les interactions (B2C) et (B2B). Ainsi, cette séparation transparente entre l'exécution et l'invocation dans le Web, fournit un modèle de programmation indépendant du langage et de l'environnement, qui accélère l'intégration des applications à l'intérieur et à l'extérieur de l'entreprise, parce que les services Web sont facilement créés en tant que technologie d'encapsulation autour des langages qui sont existants et recommandés. Cette intégration vers les services Web produit certainement des systèmes d'entreprise flexibles, interopérables et faiblement couplés.

3.3 Architecture

Les services Web ont été proposés pour déployer, dans un format standard, des applications sur les réseaux Internet ou Intranet. Ces services respectent les principes de l'approche orientée services précédemment présentée ; ils sont donc décrits, publiés et découverts. Ainsi, cela permet à une application d'utiliser à distance les services fournis. L'architecture des services Web repose sur les fondements de l'approche orientée services précédemment présentée ; comme le montre la Figure 3.1, l'architecture des services Web est une architecture qui comprend trois composantes et trois opérations. Le fournisseur de service Web enregistre, auprès d'un annuaire UDDI, son service en décrivant ses fonctionnalités et ses aspects non-fonctionnels dans un fichier WSDL (ou sous un autre format, comme nous le verrons par la suite dans la section 5.2, page 49). Le client peut consulter l'annuaire UDDI pour l'interroger en vue de trouver un ou plusieurs services qui répondent à ses besoins. De point de vue consommateur, le service Web n'est qu'une boîte noire qui reçoit des entrées et émet des résultats, les détails techniques sur l'implantation ou la plateforme sont masquées. Ce qui est important pour un consommateur de services, ce sont les fonctionnalités et quelques propriétés, telles que la localisation et les moyens pour consommer ces services. Les communications se font par le biais du protocole SOAP.

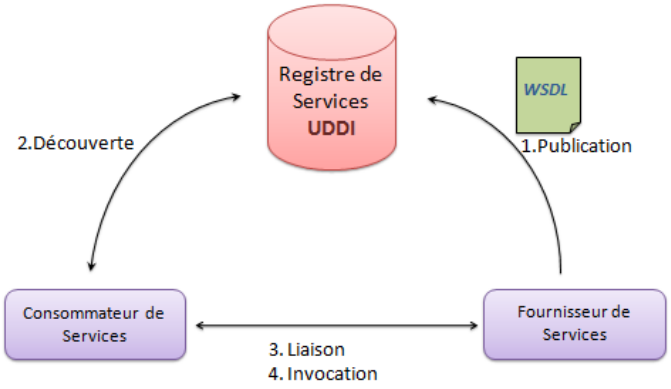

Figure 3.1 – Architecture des services Web

3.4 Standards des services Web et SOA

À l'heure actuelle, les normes de services Web sont la meilleure solution pour développer des produits à base SOA. L'utilisation de standards recommandés est l'un des aspects qui permettent à la technologie du service Web de déployer des services techniquement compatibles. Dans cette section, nous exposons les principaux standards recommandés pour cette technologie.

3.4.1 XML

XML (eXtensible Markup Language) (XML, 2003), standardisé par le W3C (World Wide Web Consortium) en 1998, est aujourd'hui largement reconnu, accepté et utilisé par de nombreuses entreprises comme format universel d'échange de données. Il représente un format de texte très souple dérivé du SGML (ISO 8879), conçu à l'origine pour répondre aux défis de la publication électronique à grande échelle. Il joue également un rôle de plus en plus important dans l'échange d'une grande variété de données sur le Web et sur d'autres supports. Aussi, il repose sur un système de balises au sein d'un fichier texte. XML peut être employé pour exprimer n'importe quel type d'information. On le retrouve par exemple aussi bien comme élément de sauvegarde de documents au sein de fichiers ou de bases de données, ou encore comme format d'échange de données.

Dans HTML, les balises sont prédéfinies et représentent l'affichage que doit revêtir la page dans le navigateur Web. Dans XML, les balises sont extensibles et permettent d'associer toutes sortes d'informations au fil du texte.

XML est aujourd'hui un standard qui permet la représentation des données structurées où semi-structurées transportables sur des protocoles communs d'Internet.

Ainsi, XML a une importance prouvée pour éviter les barrières techniques pour la communication entre les plateformes hétérogènes. Pour cette raison, il constitue la technologie de base pour les services Web. En effet, il apporte aux services Web l'extensibilité et la transparence vis à vis les plateformes et les langages de développement. De plus, grâce à l'extensibilité de sa structuration, il permet la distinction entre les données de différentes applications, et il facilite aussi l'encapsulation des données de différents langages et protocoles utilisés. Aussi, il permet de régler les problèmes d'hétérogénéité entre les données et les messages échangés en adoptant la notion des Namespaces et la spécification d'XML Schema pour exprimer les structures des données habituellement complexes figurant dans les messages échangés. Ces deux notions assurent l'échange compréhensible des message, ainsi que l'interopérabilité entre les systèmes.

3.4.2 WSDL : le langage de description des services Web

Le succès des services Web est fondé sur le faible couplage qui existe entre les consommateurs et le fournisseur de service. La description des différentes fonctionnalités du service dans un langage standard permet de séparer entre la consommation du service d'un côté, et les langages de programmation et des plateformes utilisés de l'autre côte. Seules les fonctionnalités du service sont présentées dans le fichier de description ; ainsi, les détails techniques propres au fournisseur de service sont masqués pour les consommateurs.

La description d'un service Web est faite à l'aide du langage WSDL (Web Services Description Language). WSDL a été proposé par sa version 1.1 en 2001. Ultérieurement, en 27 juin 2007, la version WSDL2.0 (WSDL, 2007) a été approuvée et recommandée officiellement par W3C. Un fichier WSDL est un fichier exprimé en XML. Il indique les informations nécessaires pour consommer les services. Il indique, ainsi, la description des fonctionnalités d'un service. En particulier, il décrit :

- Le protocole de communication utilisé (SOAP RPC ou SOAP orienté message)
- Le format des messages requis pour consommer le service.
- Les méthodes offertes par le service.
- La localisation du service.

En pratique, WSDL 2.0 décrit un service Web en deux parties fondamentales : une partie abstraite et une partie concrète (Voir la Figure 3.2). À l'intérieur de chaque partie, la description utilise un certain nombre de constructeurs pour promouvoir la réutilisation de la description et pour séparer les soucis de conception de manière indépendante. La partie abstraite est définie par les constructeurs types, messages et port types, qui décrivent respectivement, les types de données utili-

sées, les messages échangés et les opérations possibles pour un service Web. La partie concrète est spécifique à l'implémentation. Elle comprend les balises binding et service. L'élément binding indique le protocole utilisé pour invoquer le service. L'élément service permet d'associer au service l'adresse réseau qui localise le service Web.

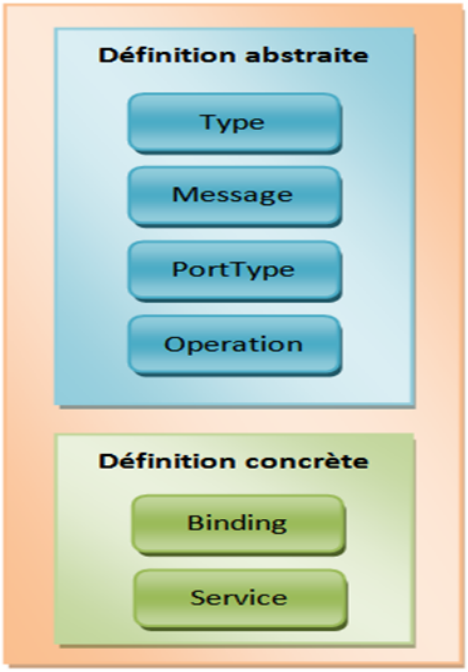

Figure 3.2 – La structure d'un fichier WSDL

Le fichier WSDL décrit les fonctionnalités d'un service Web. Cependant, il ne tient pas en compte les aspects non-fonctionnels qui peuvent lui être associés, tels que la sécurité et les transactions. Pour traiter ces aspects, de nouvelles extensions d'WSDL sont proposées, comme les spécifications WS-Security (Lawrence et al., 2006) pour la sécurité, WS-Transaction (WSA, 2007) pour les transactions, etc. D'autres travaux discutent la description sémantique des services Web d'une manière à assister les processus de découverte et de composition des services. Dans le chapitre 4 (page 32) nous discuterons en détails ces descriptions.

3.4.3 UDDI : l'annuaire de services Web

L'un des principaux avantages de l'orientation des entreprises vers les technologies de services Web est le partage de services sur les réseaux (Intranet ou Internet). Le partage de services Web permet de minimiser le coût de développement d'applications, en termes de temps et d'argent. L'élément-clé pour ce partage des services est l'annuaire UDDI, qui permet de référencer et de classifier leurs fonctionnalités.

L'UDDI est une spécification d'annuaire qui propose d'enregistrer et de rechercher des fichiers de description de services Web correspondant aux attentes d'un client (UDDI, 2007). L'UDDI a été initialement conçu par des industriels, en ayant pour but d'avoir un standard indépendant des plateformes d'implémentation, afin de connaître les entreprises qui fournissent des services Web et de découvrir les services Web disponibles qui répondent aux attentes du client (Belfedhal and Malki, 2011).

Pour simplifier les recherches de services, le standard UDDI propose trois types d'annuaires qui ont des critères particuliers :

1. **les pages blanches** permettent de connaître les informations concernant les entreprises ;
2. **les pages jaunes** présentent les services selon leurs fonctionnalités en utilisant une taxonomie industrielle standard ;
3. **les pages vertes**, quant à elles, informent sur les services fournis par les entreprises. Elles référencent la localisation des fichiers de description WSDL des services Web.

WSDL et UDDI sont deux standards pour les services Web. Le premier permet de décrire les fonctionnalités et les moyens d'utilisation du service d'un fournisseur. Ce dernier peut aussi enregistrer son service dans un annuaire UDDI pour qu'il soit référencé et utilisé par des consommateurs de service. Il existe un autre point important dans l'architecture des services Web, à savoir la communication entre les différentes parties qui se fait avec le protocole SOAP présenté dans la section suivante.

3.4.4 SOAP : communications entre les services Web

SOAP (Box et al., 2000), à l'origine acronyme de Simple Object Access Protocol, est un protocole dont la syntaxe est basée sur XML. Son but est de permettre des échanges standardisés de données dans des environnements distribués. Il permet d'accéder à des services distants indépendamment de la plateforme d'exécution. Initialement proposé par Microsoft et IBM, la spécification SOAP est aujourd'hui une recommandation W3C.

Un message SOAP contient une enveloppe obligatoire, un en-tête facultatif et un corps obligatoire qui se présente sous la forme d'un document XML (Voir Figure 3.3 et Figure 3.4) :

- **Enveloppe** (envelope) : contient la spécification des espaces de désignation (namespace), du codage de données (version du schéma XML-SOAP supporté), etc.

– **En-tête** (header) : contient des éléments-fils, appelés entrées, permettant d'ajouter des extensions à un message. Ces extensions permettent, par exemple, de prendre en charge les transactions et la sécurité. SOAP se voulant un protocole léger et simple a volontairement ignoré la sécurité.
– **Corps** (body) : contient la méthode à invoquer ainsi qu'éventuellement ses paramètres d'appel.
– **Erreurs** (fault) : réactions en cas d'erreur (ayant des causes différentes).

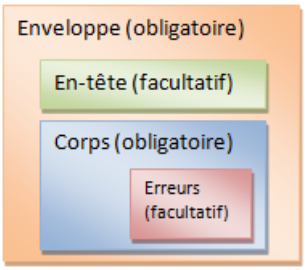

Figure 3.3 – Éléments de SOAP

```
<?xml version='1.0' ?>
<env:Envelope xmlns:env="http://www.w3.org/2002/06/soap-envelope">
 <env:Header>
  <m:reservation xmlns:m="http://travelcompany.example.org/reservation"
      env:role="http://www.w3.org/2002/06/soap-envelope/role/next"
        env:mustUnderstand="true">
   <m:reference>uuid:093a2da1-q345-739r-ba5d-pqff98fe8j7d</reference>
   <m:dateAndTime>2001-11-29T13:20:00.000-05:00</m:dateAndTime>
  </m:reservation>
  <n:passenger xmlns:n="http://mycompany.example.com/employees"
      env:role="http://www.w3.org/2002/06/soap-envelope/role/next"
        env:mustUnderstand="true">
   <n:name>John Q. Public</n:name>
  </n:passenger>
  <z:travelPolicy
    xmlns:z="http://mycompany.example.com/policies"
      env:mustUnderstand="true">
   <z:class>economy</z:class>
   <z:fareBasis>non-refundable<z:fareBasis>
   <z:exceptions>none</z:exceptions>
  </z:travelPolicy>
 </env:Header>
 <env:Body>
  <p:itinerary
    xmlns:p="http://travelcompany.example.org/reservation/travel">
   <p:departure>
     <p:departing>New York</p:departing>
     <p:arriving>Los Angeles</p:arriving>
     <p:departureDate>2001-12-14</p:departureDate>
     <p:departureTime>late afternoon</p:departureTime>
```

Figure 3.4 – Exemple de message SOAP

Le protocole SOAP s'appuie sur des standards de communication, comme le protocole HTTP, mais il peut aussi utiliser d'autre protocoles, comme SMTP (Belfedhal and Malki, 2011). L'avantage d'utiliser SOAP avec le protocole HTTP est que la communication est facilitée, en particulier, les proxys et les pare-feu peuvent être franchis sans problème. Il est ainsi facilement adaptable à toutes les technologies antérieures, tout en restant simple et extensible. Le protocole SOAP a pour avantage d'être indépendant de la plateforme d'exécution et du langage de programmation.

3.5 La pile de programmation des services Web

Identiquement à SOA, la technologie du service Web offre une pile de programmation (Voir la Figure 2.3) inspirée de la pile architecturale SOA détaillée dans la section 2.8 (page 17), sauf que la pile de programmation des services Web propose des solutions technologiques concrètes qui décorent cette pile. Maintenant, nous donnons une brève explication sur cette pile (Voir la Figure 3.5), qui présente un ensemble de protocoles et d'interfaces de programmation d'application (API) standardisés permettant aux individus et aux applications de localiser et d'utiliser les services Web.

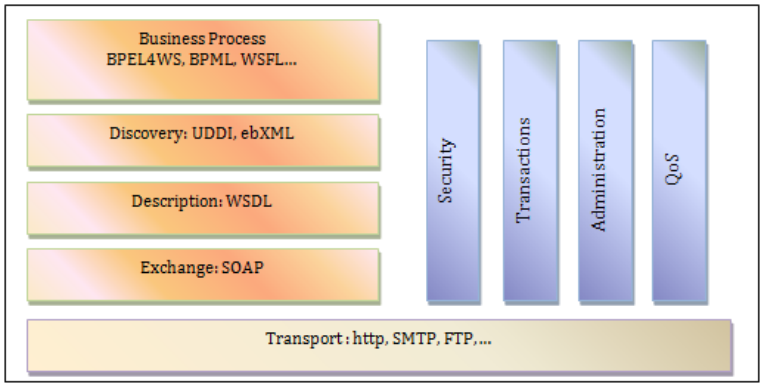

Figure 3.5 – Architecture en Pile des services Web

L'organisation de la pile de programmation des services Web en couche justifie la place prépondérante du déploiement omniprésent des architectures des services Web. Nous présentons dans ce qui suit les différentes couches qui apparaissent dans cette pile.

La couche réseau est la couche inférieure de la pile de laquelle tous les services Web doivent être expédiés à travers les réseaux. Cette expédition est assurée par un protocole http ou d'autres protocoles réseau, tels que le protocole Internet IIOP utilisé par CORBA ou le protocole MQSeries d'IBM. La couche réseau est suivi par la couche messagerie, qui est basée sur XML, ce qui facilite les communications entre les services Web et leurs clients. La couche messagerie recommande le protocole SOAP. SOAP, décrit précédemment, est un protocole écrit en XML qui facilite les opérations de publication, de recherche, de liaison et d'invocation. WSDL, adopté dans la couche de description, est une spécification qui décrit les services Web disponibles pour les clients. Présenté aussi en XML, la description employée dans WSDL considèrent l'interface de programmation et l'emplacement des services Web.

Alors, ces trois couches (réseau, messagerie et description) sont nécessaires pour obtenir des services Web interopérables en créant également une entrée peu coû-

teuse pour l'exploitation des services Web permettant le déploiement des services sur Internet. Les couches restantes de la pile, qui seront décrites par la suite, sont facultatives, et elles sont utilisées en fonction des besoins de l'entreprise (Gottschalk et al., 2002b). La publication consiste également à annoncer le WSDL dans un registre UDDI, afin que de nombreux développeurs ou de nombreux utilisateurs des services puissent le trouver. De même, la découverte d'un service est toute action qui donne au demandeur de service l'accès au WSDL pour un service. L'action peut être aussi simple que d'accéder à un fichier ou à une URL contenant le WSDL, ou aussi complexe que d'interroger un registre UDDI et d'utiliser le ou les fichiers WSDL pour sélectionner l'un des nombreux services potentiels.

La couche flux de processus métier de la pile facilite la composition des services Web en flux de travail, et représente cette agrégation de services Web en tant que service Web de niveau supérieur ou ce que nous l'appelons service composite.

Pour qu'une application des services Web répond aux exigences strictes des entreprises, il faut fournir une infrastructure de classe entreprise, y compris la sécurité et la gestion de la qualité de service. Ces éléments d'infrastructure, représentés par les tours verticales sur le côté droit de la Figure 3.5, doivent être traités à chaque couche de la pile. Les solutions de chaque couche peuvent être indépendantes les unes des autres.

3.6 Conclusion

En conclusion, il est nécessaire de faire le point sur la technologie des services Web. Les services Web décrivent un ensemble de protocoles standards utilisés pour établir un domaine d'intégration des applications. L'un des facteurs ayant contribué au succès des services Web est sans doute l'utilisation des standards Internet, tels que XML et HTTP. En conséquence, tout système capable d'analyser du texte et de communiquer via un protocole de transport Internet standard peut communiquer avec un service Web. XML a engendré l'apparition de nouveaux protocoles, tels que SOAP pour l'échange de messages, WSDL pour la description des services, et UDDI pour la publication et la découverte des services. Ces protocoles reposent sur une architecture orientée services (SOA), et correspondent à des composants logiciels qui peuvent être combinés, grâce à un langage de composition, pour former de nouveaux services plus élaborés.

Chapitre 4

Service Web Sémantique (SWS)

Sommaire

4.1 Introduction

La description syntaxique en WSDL ne permettrait pas d'automatiser complètement les différentes opérations liées aux services Web, en particulier la publication, l'invocation, la découverte, la composition, etc. Pour résoudre ce problème, il est nécessaire d'enrichir les descriptions des services Web avec d'autres informations supplémentaires, réutilisables, partageables et compréhensibles par la machine. Ces informations sont des métadonnées, qui permettent de présenter les particularités des services Web. En d'autres termes, il est nécessaire d'ajouter une couche de description sémantique et des mécanismes de raisonnement sur cette description à l'infrastructure des services Web, afin de pouvoir automatiser les tâches mentionnées précédemment.

Dans ce chapitre, nous présentons les services Web sémantiques comme une technologie émergente qui permet l'exploitation des technologies des services Web et du Web sémantique. Pour commencer, nous allons tout d'abord présenter un État de l'Art qui définit les principaux travaux dans le cadre du Web sémantique et les services Web sémantiques, en montrant ses principes et ses objectifs. Par la suite, nous présentons les principaux outils et approches de sa mise en œuvre. Dans la deuxième partie de ce chapitre, nous allons exposer la notion de service Web sémantique, les standards, les normes et les langages utilisés pour les implémentés.

4.2 État de l'Art

4.2.1 Concept de la sémantique

Les technologies telles, que WSDL, UDDI, SOAP ou WS-BPEL, sont principalement motivées par l'interopérabilité syntaxique de composants logiciels sur le Web (Benatallah et al., 2005; Yu et al., 2008). Elles s'appuient sur la description syntaxique des fonctionnalités des services Web et les propriétés en XML. Ainsi, ces standards ne supportent pas l'interprétation automatique d'information, par la machine, concernant les aspects fonctionnels et non fonctionnels.

Intuitivement, nous pourrions supposer qu'un service Web peut être déduit de la description de ses entrées et sorties. Toutefois, ce n'est pas nécessairement le cas, parce que, par exemple, la norme WSDL n'impose pas de conventions relatives à la désignation et la dénomination des éléments. WSDL est basé sur XML ; comme les documents XML ne peuvent pas être nécessairement interprétés sans intervention humaine, c'est alors aussi le cas pour WSDL. Généralement, et contrairement à un être humain, un ordinateur ne comprend pas la signification sous-jacente sémantique des termes. Et similairement pour les services Web, un ordinateur ne comprend pas la signification sémantique des interfaces du service, ce qui conduit au point que les services Web, qui offrent des fonctionnalités similaires, pourraient posséder des interfaces et des opérations sensiblement différentes, et l'inverse, les services qui possèdent les mêmes interfaces pourraient offrir complètement différentes fonctionnalités (Gomadam et al., 2006; Steinmetz et al., 2008). Il s'agit d'une lacune majeure qui affecte en particulier la découverte des services.

(Paolucci et al., 2002a,b) affirment que ce problème doit être résolu par l'augmentation de la représentation des interfaces de services et l'identification des similitudes sémantiques entre elles, c'est-à-dire, la description de service ne doit pas être seulement basée sur la syntaxe, mais devrait également être augmentée par des concepts sémantiques décrivant les composants à usage unique. Ici, les idées et les technologies du Web sémantique entrent en jeu. Au début, Tim Berners-Lee a proposé l'idée du Web sémantique (Berners-Lee et al., 2001). Il a fait valoir que des avantages énormes pourraient naître de l'information et des fonctionnalités disponibles sur l'Internet, si le Web devient interprétables par la machine. En conséquence, Berners-Lee a proposé d'étendre le Web avec la sémantique dans le but de donner aux informations un sens bien défini, et de fournir une couche de données pour la machine (Berners-Lee et al., 2001; Sure et al., 2002; Shadbolt et al., 2006). En particulier, Berners-Lee a considéré des agents logiciels qui pourraient travailler ensemble en vue de collecter et de traiter la sémantique des données. Il a également affirmé que dans un "pro-

cessus, appelé service de découverte", il serait nécessaire de décrire sémantiquement les fonctionnalités d'un agent par « un langage commun » pour décrire un service d'une manière qui permet aux d'autres agents de « comprendre » à la fois la fonction offerte et « comment en profiter » (Berners-Lee et al., 2001).

À ce jour, plusieurs technologies, comme Resource Description Framework (RDF) ou le langage d'ontologie Web (OWL), ont été exploitées respectivement standardisées, pour fournir les fonctionnalités de base du Web sémantique. La figure 4.1 montre les différents aspects du Web sémantique (Berners-Lee, 2006). Le Web sémantique est au centre de nombreux projets de recherche. Les technologies sémantiques sont appliquées à divers domaines, tels que la gestion des connaissances, business intelligence, Web 2.0, SOC (Service-Oriented Computing), et enfin les services Web (Benjamins et al., 2008). Ce dernier est assez évident, car les normes de service Web sont basées sur XML, qui est également l'un des éléments constitutifs du Web sémantique. Cela permet de combiner facilement les services Web avec les technologies du Web sémantique pour créer des services Web sémantiques définis comme des services Web dont "les propriétés, la capabilité, les interfaces et les effets sont encodés de façon non ambiguë et interprétable par la machine". Dans l'un des articles fondamentaux sur SWS, McIlraith et al. (2001) proposent l'utilisation des technologies du Web sémantique, afin de rendre les services interprétable par la machine, et par conséquent de faciliter la découverte automatique, l'exécution, la composition, et l'interopérabilité entre les services Web(McIlraith et al., 2001).

En général, en ce qui concerne la recherche dans le domaine des services Web, l'intégration des technologies sémantiques est souvent décrite comme l'un des grands défis qui doivent être remplis, afin de faciliter le succès des services Web sur une large échelle (par exemple, (Bichier and Lin, 2006; Krummenacher et al., 2005; Martin et al., 2007b; Papazoglou et al., 2006, 2007)). Entre autres, l'ajout de la sémantique Web aux services est considéré comme un facteur de réussite majeur en vue de faciliter les fondations, la gestion et l'ingénierie des services (Papazoglou and Van Den Heuvel, 2007; Kashyap and Sheth, 1994).

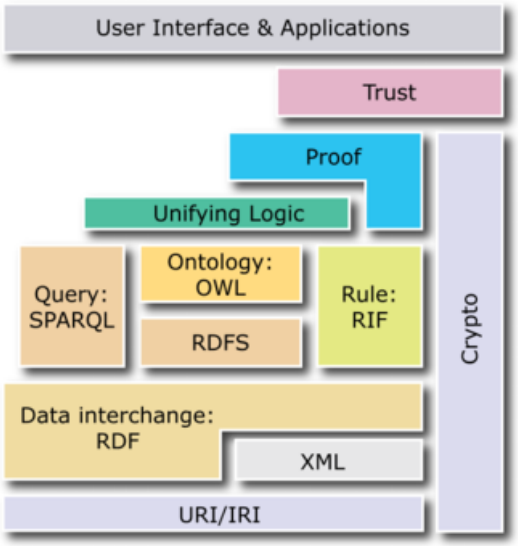

Figure 4.1 – L'architecture de Référence du Web Sémantique (Berners-Lee and Hendler, 2001)

4.2.2 Utilisation de la sémantique pour les Services Web

(Kashyap and Sheth, 1994) définit la sémantique de l'information comme la signification et l'utilisation de l'information. En revanche, la syntaxe définit la structure des données. Par exemple, la syntaxe WSDL définit la façon dont un document WSDL doit être structuré. En règle générale, le contenu du Web sémantique peut être divisé en données et métadonnées. Bien que les données représentent le contenu réel, les métadonnées peuvent être définies comme des données ou des informations sur les données (Kashyap et al., 2008).

En ce qui concerne les services Web, la sémantique peut être utilisée pour donner un sens à certains composants du service, par exemple, pour fournir des informations détaillées pour une opération, une interface ou un message dans une description WSDL. Outre que les composants déjà existants dans la description des services Web, l'intégration de la sémantique dans les normes de service Web a également abouti à la définition de nouveaux éléments, qui n'ont pas été considérés comme des données non sémantiques du service Web : les pré-conditions et les effets (aussi dénommé post-conditions) sont des exemples bien connus pour ces nouvelles constructions. Contrairement aux entrées et aux sorties, les pré-conditions et les effets accordent des informations sur un état avant, respectivement après, l'invocation d'un service. Les entrées et les sorties, les pré-conditions et les effets forment le profil de service (Schumacher et al., 2008). Les pré-conditions et les effets ont été présentés dans WSDL-S (Miller et al., 2004) et OWL-S (Martin et al., 2007a).

(Patil et al., 2004; Gomadam et al., 2006) Distinguent quatre différents types de la sémantique utilisés pour les services Web :

1. La sémantique de données : définir formellement les données en entrée et en sortie.
2. La sémantique fonctionnel : définir formellement les capacités des services Web, c'est à dire en définissant les post-conditions et les effets et annoter sémantiquement les interfaces et les opérations.
3. La sémantique non-fonctionnelle : se référence à la qualité de service et les exigences de politique générale et les contraintes.
4. La sémantique d'exécution : décrire l'exécution des services et les opérations qui permettront d'effectuer des tests de validation du service.

Afin de définir le sens des composants de service, soit par des annotations sémantiques ou par une description ontologique, il est nécessaire d'avoir un modèle qui peut être utilisé comme une base de connaissances. Le formalisme le plus connu pour exprimer une base de connaissances est l'ontologie. Bien qu'il existe d'autre formats, les ontologies fournissent une grande expressivité en fournissant des relations sémantiques plus riches entre les termes et les attributs (Cardoso, 2007a).

4.3 Les Ontologies

Historiquement, l'ontologie est un concept philosophique. Elle désigne la science de l'être, et elle décrit une théorie à propos de la nature de l'existence. Actuellement, les ontologies ont marqué leur présence dans plusieurs domaines de recherche, afin de résoudre les problèmes de modélisation des connaissances, de l'intelligence Artificielle, de la classification, du regroupement et plus précisément de l'ingénierie des connaissances. Elle est définie par Gruber (1993) comme étant une spécification formelle et explicite d'une conceptualisation partagée. La conceptualisation est une vue abstraite et simplifiée du monde que l'on veut représenter ; de ce fait, dans le chapitre 7, nous ciblons la modélisation de la vue abstraite des services Web en proposant la construction d'une ontologie de services Web.

Dans la section suivante, nous présentons quelques approches utilisées ou recommandées dans le cadre de SWS.

4.4 Approches proposées pour les Services Web Sémantiques

Comme le Web actuel s'appuie sur un ensemble de standards (HTML, HTTP, etc.), et comme le service Web syntaxique repose lui aussi sur un ensemble de standars (XML, WSDL, SOAP, etc.), le service Web sémantique lui aussi recommande et propose un ensemble de standards et architectures universels. Pour cela, plusieurs solutions et langages ont été proposés. Dans cette section, nous allons présenter les fameux solutions proposées après avoir été classées. Alors, la réalisation des services Web sémantiques suit deux approches :

1. La première approche, nommée annotation, consiste à relier (augmenter) les langages syntaxiques avec de l'information sémantique en référent leurs interfaces à des concepts ontologiques en vue d'enlever toute ambiguïté.
2. La deuxième approche consiste à produire un langage complet et bien organisé pour décrire les interfaces syntaxiques.

En se basant sur cette classification, nous présentons les solutions proposées pour la première classe d'approches d'annotation. Ensuite, nous exposons la seconde qui est communément appelée langages de description sémantique ou ontologies de description sémantique.

4.4.1 Annotation des langages Syntaxiques

L'annotation sémantique consiste à augmenter la description syntaxique des services. Elle établit des liens entre les éléments des interfaces syntaxiques et les concepts d'un ensemble d'ontologies de référence. L'ontologie de référence permet d'enlever les ambiguïtés syntaxiques, et de fournir un modèle avec des définitions conventionnelles qui sollicite l'automatisation et l'interprétation machine.

Trois modèles ont été proposés pour l'annotation sémantique, à savoir WSDL-S (Miller et al., 2004) et SAWSDL (Farrell and Lausen, 2007), permettant d'annoter manuellement une description WSDL, alors que MWSAF (Patil et al., 2004) propose la correspondance et l'annotation des services Web de façon semi-automatique utilisant des ontologies de domaine pour catégoriser les services Web en domaines.

WSDL-S

La version originale du WSDL-S (Miller et al., 2004) propose une approche légère pour ajouter de la sémantique aux services Web. Ce projet est issu du projet METEOR-S, du laboratoire LSDIS de l'Université de Géorgie, qui vise à ajouter de

la sémantique au cycle de vie complet des processus Web. Cette approche présente un méta-modèle étendu pour WSDL 2.0 (Chinnici et al., 2007) en lui ajoutant deux nouvelles balises : la balise « **Action** » qui permet de décrire l'action effectuée par l'opération, et la balise « **Contrainte** » pour décrire les prés et post conditions d'une opération. L'approche a été mise à niveau par (Akkiraju et al., 2005) de façon évolutive. Particulièrement, il s'agit d'une révision pour augmenter l'expressivité de WSDL avec la sémantique en employant des concepts analogues à ceux de OWL-S tout en étant agnostique au langage de représentation sémantique.

SAWSDL

Annotations sémantiques pour WSDL et XML Schema (SAWSDL) (Farrell and Lausen, 2007) est une approche recommandée par W3C en avril 2007. Elle explique comment l'annotation sémantique est accomplie en utilisant des références à des modèles sémantiques (Ontologies). Ainsi, l'approche ne spécifie pas de langage pour représenter les modèles sémantiques. Au lieu de cela, elle fournit des mécanismes par lesquels les concepts des modèles sémantiques peuvent être référencés à partir des composants WSDL et XML Schema à l'aide d'annotations. Généralement, ces annotations sont définies en dehors du document WSDL.

Les attributs d'extension proposés dans l'approche s'inscrivent dans les cadres d'extensibilité WSDL 1.1(Christensen et al., 2001), WSDL 2.0 (Chinnici et al., 2007) et XML Schema (Beech et al., 2001). Aussi, SAWSDL propose l'annotation des interfaces et des opérations WSDL avec des informations de catégorisation qui peuvent être utilisées lors de la publication dans un registre. Identiquement, des annotations sur les types de schéma peuvent être utilisées pendant la découverte et la composition, De plus, l'approche SAWSDL définit des annotations pour spécifier le mappage de données entre XML schéma vers une ontologie ; ces mappages pourraient être utilisés lors de l'invocation. Les principales extensions permettant d'annoter un document WSDL 2.0 (Chinnici et al., 2007) sont :

1. Un attribut d'extension, nommé **modelReference**, pour spécifier l'association entre un concept dans un modèle sémantique et un composant d'XML schéma ou WSDL. Cela est utilisé pour annoter les définitions, les déclarations d'éléments et les déclarations d'attributs dans XML schéma, ainsi que les interfaces, les opérations et les erreurs dans WSDL.
2. Deux attributs d'extension, nommés **liftingSchemaMapping** et **loweringSchemaMapping**, qui sont ajoutés aux déclarations d'éléments et aux définitions de type d'XML Schema pour spécifier les mappages entre les données sémantiques et XML.

METEOR-S WSAF

Bien que les deux approches ci-dessus proposent l'annotation manuelle de WSDL, METEOR-S Web Service Annotation Framework (MWSAF) (Patil et al., 2004) propose une méthode pour annoter de manière semi-automatique les descriptions WSDL avec les ontologies pertinentes. MWSAF fait partie d'un projet METEOR-S au laboratoire LSDIS de l'Université de Géorgie. Il implémente un certain nombre d'algorithmes pour faire correspondre les concepts des fichiers WSDL aux ontologies. Alors, il présente un cadre pour le balisage semi-automatique des descriptions des services Web avec des ontologies. Ainsi, MWSDI fournit une infrastructure en quatre catégories pour la sémantique :

- Sémantique des données (entrées et sorties).
- Sémantique fonctionnelle (que fait un service).
- Sémantique d'exécution (vérification d'exécution)
- Sémantique QoS (paramètres de performance et de coûts associés au service).

Le processus d'annotation est réalisé en quatre étapes. La première dite **SchemaGraphs** s'occupe de convertir les deux modèles, XML Schéma représenté dans WSDL et les ontologies en entrée, en un format de représentation commun, appelé **SchemaGraph**, pour faciliter une meilleure correspondance. L'avantage d'utiliser **SchemaGraph** est que l'ontologie peut être dans n'importe quel langage. Un **SchemaGraph** est un ensemble de nœuds connectés par des arêtes. La deuxième étape compare chaque concept **SchemaGraph** d'WSDL avec tous les concepts **SchemaGraph** de l'ontologie ; un degré de similitude **MS** est calculé basé sur la moyenne pondérée de la similitude linguistique et la similitude structurelle. Comme chaque concept WSDL est comparé à tous les concepts des ontologies, il est nécessaire de trouver le meilleur concept correspondant. Alors, la troisième étape cherche ce concept en se basant sur la meilleure valeur **MS**. La dernière consiste à écrire le mappage trouvé donnant la main à l'utilisateur d'annuler ou de modifier les règles déjà calculées.

4.4.2 Langages de description sémantique

Les solutions proposées pour l'annotation des services Web utilisent et supportent les langages syntaxiques tels que WSDL. Contrairement à cela, la seconde classe propose la production d'une nouvelle structure symantique qu'est totalement indépendante aux langages syntaxiques. Nous présentons OWL-S et WSMO, les principales solutions proposées dans cette classe.

OWL-S

OWL-S est l'acronyme de "Ontology Web Langage - Services". Le projet de cette ontologie a été officiellement publié au cours de l'année 2000 intitulé "DAML-S : Web Service Description for the Semantic Web" (Ankolekar et al., 2002). L'ontologie portait alors le nom de DAML-S, car elle était exprimée dans le langage DAML+OIL (Darpa Agent Markup Language + Ontology Inference Layer). Le but d'OWL-S (Martin et al., 2004) est de rendre les services Web interprétables par la machine et permettre l'automatisation de la découverte, de l'invocation, de l'interopérabilité, de la composition, de l'exécution, et de la vérification des propriétés des services Web en décrivant les propriétés et les capacités des services Web.

L'ontologie OWL-S fournit trois essentiels types de connaissances sur un service (illustrés dans la Figure 4.2), chacun étant caractérisé par une question à laquelle il répond :

1. **ServiceProfile :** « que fait un Service ? » cette classe est utilisée pour annoncer et présenter le service. Dans ce qui suit, nous décrivons les principales parties de la classe ServiceProfile ; elles sont classées en trois sous classes : la première décrit les propriétés de liaison qui lient la classe ServiceProfile avec la superclasse Service et avec ServiceModel, la deuxième décrit les informations de contact et de description du profile, et la troisième représente la description fonctionnelle du service Web (les entrées, les sorties, les prés-conditions, et les poste-conditions (effets)) :
   - **Propriétés de liaison :** Il existe une relation bidirectionnelle entre la superclasse service et la classe ServiceProfile, de sorte qu'un service peut être lié à un ServiceProfile, et un ServiceProfile à un service. Ces relations sont exprimées par les propriétés *presents* et *presentedBy*.
   - **Propriétés de contact :** Certaines propriétés du profile fournissent des informations lisibles par l'homme qui sont peu susceptibles d'être traitées automatiquement. Ces propriétés incluent *serviceName*, *textDescription* et *contactInformation*.
   - **Propriétés fonctionnelle :** Un élément essentiel du profil est la spécification de la fonctionnalité fournie par le service et la spécification des conditions qui doivent être remplies pour un résultat réussi. En outre, le profile spécifie les conditions résultant du service, y compris les résultats attendus et inattendus de l'activité de service. Le profile OWL-S représente deux aspects de la fonctionnalité du service : la transformation de l'information (représentée par les entrées et les sorties), et le changement d'état produit par l'exécution du service représenté par les pré-conditions et les

effets. Généralement, ils sont abrégés par (IOPR).

2. **ServiceModel :** « comment il fonctionne ? » OWL-S 1.2 définit la classe Process comme une sous-classe de ServiceModel. Ainsi, la classe Process est une sous-classe de ServiceModel. La notation OWL-S d'un processus s'appuie sur des travaux bien établis dans divers domaines, notamment des travaux en IA sur la normalisation des langages de planification (Ghallab et al., 1998). La classe Process décrit le fonctionnement du service Web. Pour cela, elle spécifie ses entrées, ses sorties, ses états, et les transitions entre ses états. Les transitions d'un état à un autre sont décrites par les pré-conditions et les effets de chaque processus.
3. **ServiceGrounding :** « comment y accéder ? »ServiceGrounding d'un service spécifie les détails de la manière d'accéder au service, les détails ayant principalement un lien avec les formats de protocole et de message, la sérialisation, le transport et l'adressage. Bien que les deux premières classes ServiceProfile et ServiceModel s'attachent à la représentation abstraite d'un service Web, la classe ServiceGrounding représente la forme concrète d'une représentation abstraite d'un service Web. Ainsi, ServiceGrounding commente les détails concrets d'un service Web, tels l'accès au service Web, les protocoles, les URIs, les messages envoyés, etc.

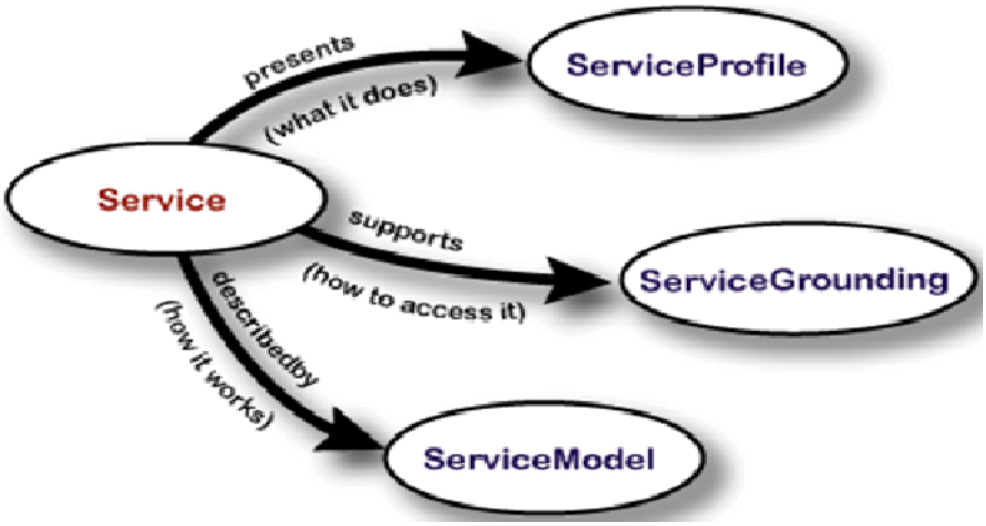

Figure 4.2 – Structure générale de l'ontologie OWL-S

WSMO (Web Service Modeling Ontology) :

WSMO (Roman et al., 2005) est un langage formel pour décrire sémantiquement tous les aspects pertinents des services. Tous comme OWL-S, WSMO (Roman et al.,

2005) vise à automatiser la découverte, la composition et l'invocation des services Web en se basant sur WSMF (Fensel and Bussler, 2002) (Web Service Modeling Framework). WSMO est un méta-modèle pour les aspects liés aux services Web sémantiques. La spécification (MOF, 2004) (Meta-Object Facility) est utilisée pour spécifier ce modèle. MOF définit un langage et un cadre abstrait pour la spécification, la construction et la gestion de méta-modèles technologiquement neutres. Ainsi, il définit une architecture de métadonnées composée de quatre couches, à savoir : la couche données, la couche modèle, la couche méta-modèle et la couche méta-méta-modèle. En termes de quatre couches MOF pour WSMO, les services Web représente la couche données, la description de WSMO des services Web correspond à la couche modèle, le langage définissant WSMO symbolise la couche méta-modèle, et le méta-méta-modèle sera donc le méta-modèle du langage définissant le langage WSMO, par exemple, si le langage définissant WSMO est exprimé en UML(comme indiqué (UML/WSMO, 2004)), le méta-méta-modèle sera alors le méta-modèle d'UML. WSMO développe les ontologies, les services Web, les buts et les médiateurs comme quatre éléments de niveau supérieur en tant que concepts principaux (illustrés dans la Figure 4.3), et qui doivent être décrits afin de décrire les services Web sémantiques :

1. **Ontologies :** Les ontologies fournissent la terminologie utilisée par d'autres éléments du WSMO pour décrire les aspects sémantique pertinents des services Web.
2. **Services Web :** Les descriptions de service Web WSMO comprennent les aspects fonctionnels, non fonctionnels et comportementaux d'un service Web.
3. **Buts :** Les objectifs représentent les désirs des utilisateurs, pour lesquels la réalisation pourrait être recherchée en exécutant un service Web. Aussi, ils modélisent la vue utilisateur dans le processus d'utilisation du service Web, et constituent donc une entité de niveau supérieur distincte et particulière dans WSMO.
4. **Médiateurs :** Les médiateurs décrivent les éléments qui permettent de surmonter les problèmes d'interopérabilité entre les différents éléments du WSMO. Les médiateurs sont le concept de base pour résoudre les problèmes d'incompatibilités au niveau des données, des processus et des protocoles, c'est-à-dire pour résoudre les discordances entre les différentes terminologies utilisées (niveau données), dans la manière de communiquer entre les services Web (niveau protocoles), et pour la coordination des services Web et des buts (niveau processus).

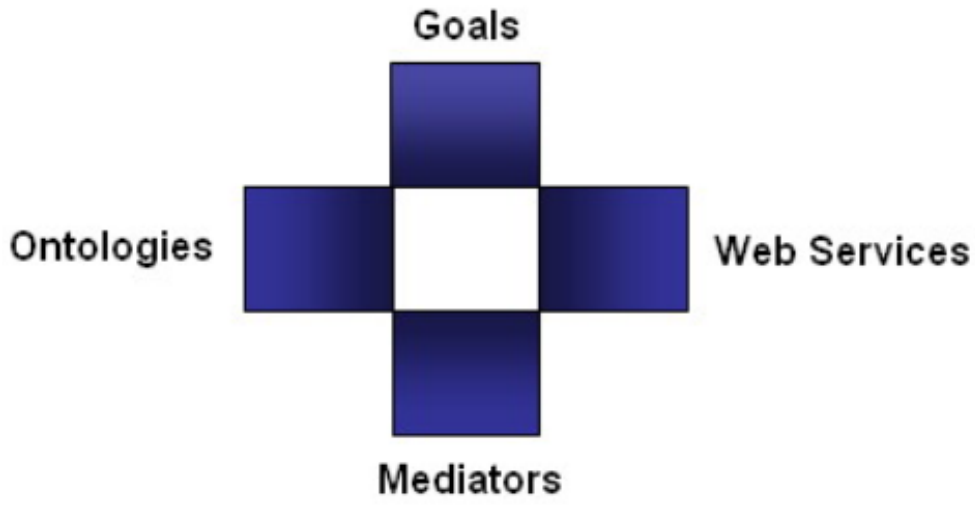

Figure 4.3 – Éléments de haut niveau de l'ontologie WSMO.

4.5 Conclusion

Les services Web sémantiques représentent un domaine de recherche intéressant. Plusieurs solutions ont été prposées pour formuler la sémantique pour les services Web. Soit par annotation ou par la construction d'un nouveau langage, les techniques du Web sémantique sont souvent utilisées pour décrire les interfaces des services Web.

Les solutions proposées par l'annotation sont simples et consistent à établir des liens entre les éléments des interfaces syntaxiques et les concepts d'un ensemble d'ontologies de référence. L'ontologie de référence permet d'enlever les ambiguïtés syntaxiques, et de fournir un modèle avec des définitions conventionnelles qui sollicite l'automatisation et l'interprétation machine. Les solutions qui proposent la construction d'un nouveau langage sont plus riches, mais elles sont compliquées le fait qu'elles demandent la maitrise de plus de langages, et nécessitent plus de mécanisme pour faire la liaison entre les descriptions syntaxiques et ces nouveaux modèles. Pratiquement, ce qui est intéressant à travers ces efforts, c'est l'automatisation, autant que possible, des différents aspects relatifs aux services Web, en particulier la découverte des services Web sémantiques, qui fera l'objet du prochain chapitre.

Deuxième partie

État de l'art

Chapitre 5

La découverte des services Web

Sommaire

5.1 Introduction

La découverte des services est abordée dans plusieurs projets et travaux de recherche (Schulte, 2010). Elle est définie par "la localisation automatique des services répondant à une requête utilisateur." (Keller et al., 2004). Ainsi, (Booth et al., 2004) décrivent le processus de découverte comme étant "la localisation d'une description compréhensible par la machine d'un service inconnu au préalable et correspondant à certains critères fonctionnels". Trois aspects majeurs de la découverte ressortent de ces définitions. Premièrement, l'aspect localisation des services qui consiste à trouver l'adresse où est fournie la description d'un service. Deuxièmement, l'aspect traitement des données qui indique le degré d'automatisation de la découverte. Troisièmement, l'aspect mise en correspondance ou "matching" qui compare la requête utilisateur aux services répertoriés. Dans ce chapitre, nous présentons premièrement un état de l'art qui va mettre l'accent sur deux classes du processus de la découverte, l'un dans un environnement centralisé et l'autre dans un environnement distribué. Nous citons, également, les facteurs qui influent sur le processus de découverte et les types de recherche employés. Un exemple sera aussi présenté pour illustrer le processus de découverte.

5.2 Processus de découverte de service

La découverte de service est le processus de trouver des services similaires à partir d'un ensemble de services, qui pourraient être fournis dans un référentiel de service central ou distribué (Cardoso, 2007b). Après avoir trouvé une série de services similaires qui ont été découverts, il est possible de sélectionner le service le plus adéquat et l'invoquer (Verma et al., 2005). En ce qui concerne l'invocation automatique et l'exécution des services Web ou même la composition automatique ou semi-automatique des services Web, il est crucial d'aborder la découverte des services. Ce n'est que les services appropriés qui peuvent être identifiés. Il est possible d'aborder d'autres problèmes, tels que la qualité de service (Berbner et al., 2006) (Eckert, 2009), et sélectionner un service et l'inclure dans un flux de travail.

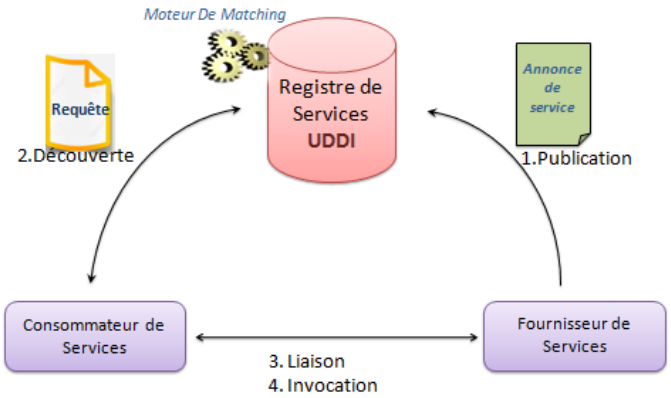

Figure 5.1 – Composantes principales de la découverte

En se basant sur le schéma déjà connu à partir des Figures 2.2 et 3.1, la Figure 5.1 montre les principales composantes du processus de découverte de service. Par rapport à la Figures 3.1, trois éléments ont été ajoutés ou changés. D'abord, le consommateur du service doit formuler une demande de service afin de trouver des services appropriés dans le registre de service. Cette demande de service peut avoir des formes différentes, par exemple, être basée sur des mots clés. Deuxièmement, le WSDL est remplacé par "l'annonce de service", un terme plus générique lié à la publication des descriptions des services. Les annonces de services peuvent prendre différentes formes, entre autres, le WSDL cité autrefois. Troisièmement, un moteur a été ajouté à la base de registre de service. Moteur de Matching tient compte de la demande de service, et il la correspond avec les annonces des services disponibles. Selon (Tsetsos, 2007), les facteurs suivants influent sur le processus de découverte :

1. La capacité des fournisseurs de services à décrire leurs services.
2. La capacité des demandeurs de services à décrire leurs besoins.
3. L'efficacité de l'algorithme matchmaking de service.

Les deux premiers éléments de la liste dépendent directement de l'expressivité du formalisme utilisé dans (SWS), c'est à dire à quel degré la sémantique des services peut être formalisée. Á cette notification, les concepts suivants sont essentiels pour la découverte de service :

- **Les annonces des services :** Il faut décrire une offre de services en fonction de différents aspects, c'est-à-dire les capacités fonctionnelles, les aspects non fonctionnels, etc. Ainsi, selon le type de registre adopté et l'expressivité de la norme de service Web utilisée, la description du service peut être plus ou moins détaillée et formelle.
- **Demandes de service :** Il faut déclarer les exigences d'un demandeur de service concernant les capacités fonctionnelles, non fonctionnelles et les techniques du service. Les demandes de service sont formulées sous forme de requêtes concrètes, qui peuvent prendre différentes formes.
- **Moteurs de Matching (ou matchmakers) :** Pour découvrir ou chercher un service, il faut réaliser ou exécuter une comparaison entre les demandes des clients et les annonces des services fournies dans le registre, puis calculer un degré de correspondance DOM (Degree of Match). Bien sûr, les services retenus seront les services ayant les DOMs les plus élevés.

5.3 Moteurs de Matching (Matchmaker)

Le Moteur de Matching est un algorithme qui vise à exécuter l'opération de correspondance ou de comparaison entre les annonces de service exprimées dans les registres UDDI et les demandes envoyées par les clients en calculant un degré de correspondance DOM «Degree of Match» entre eux. La mise en correspondance est une comparaison réelle (deux à deux) entre une annonce de service et une demande de service (Studer et al., 2007). Selon le type d'information fournie dans une demande de service, et par conséquent disponibles pour la mise en correspondance, il existe plusieurs approches pour réaliser la correspondance entres les services. Ces approches peuvent être classées en différentes classes. Ci-après, sont brièvement présentées :

Recherche par mots-clés

Il s'agit du format de requête le plus classique qui a été utilisé pour la recherche sur Internet, où le demandeur spécifie des termes qui sont comparés, par le moteur de recherche, avec la description d'un article ou d'une page Web à l'aide d'une correspondance de chaînes de caractères. Dans le cas des services Web, les termes seront comparés à la description d'un service, et il serait nécessaire de définir les composants des services (les messages, les entrées et les sorties) par un ensemble de concepts ou de mots-clés. Cette méthode est utilisée surtout par les registres des services classiques, tel que UDDI et ebXML. Comme nous l'avons déjà vu dans la section 4.2.1 page (34), les approches syntaxiques ne sont pas suffisantes pour décrire les capacités de service, et par conséquent les requêtes de services. De plus, les descriptions textuelles incluent des informations sur la façon d'appeler un service et non pas sur son comportement. De ce fait, les services peuvent exiger et fournir le même ensemble de paramètres tout en offrant un comportement différent.

5.3.1 Recherche basée sur la sémantique

Elle permet de définir des requêtes à base de concepts sémantiques, plutôt que sur des mots-clés. De cette manière, il est possible de définir une demande de service de manière beaucoup plus précise, et de dériver des relations de subsomption, ou autres, entre les concepts sémantiques définis dans une demande de service avec d'autres concepts sémantique d'une offre de service. Ces approches prennent en compte la sémantique des éléments de service, de sorte que les similarités peuvent être détectées, même s'il existe des différences syntaxiques. Pour cela, ces approches utilisent, par exemple, les relations de subsomption, qui peuvent être déduites par un raisonneur à partir des hiérarchies de subsomption de concepts définis dans une ontologie sous-jacente.

5.3.2 Recherche basée sur l'état

Cette méthode améliore la précédente (basée sur la sémantique) en permettant de définir des pré- et post-conditions pour un service. Ainsi, il est possible de modéliser le comportement souhaité d'un service, à la fois dans une offre de service et dans une demande de service. Contrairement à la méthode basée sur la sémantique, qui est principalement limitée à la signature de service, la recherche basée sur l'état s'applique au profil de service, et elle inclut la spécification des pré-conditions et des post-conditions, qui doivent être remplies avant et après l'exécution d'un service (Voir section 4.2.2, page 36).

Finalement, il faut noter qu'un Moteur de Matching peut être syntaxique, sémantique ou hybride.

5.3.3 Étapes de Matching

Nous supposons qu'un certain nombre d'offres de services ont été enregistrées dans un registre central. Pour une demande, les services correspondants nécessitent d'être identifiés. Pour cela, les offres doivent être classées en fonction de leur similarité relativement à la demande. Dans ce contexte, le processus de Matching peut être divisé en trois étapes bien distinctes :

a. **Identification des éléments de données**
   Dans un premier temps, il est nécessaire de définir les composants et les informations sur les descriptions de services Web disponibles avant d'exécuter le Matching. Durant cette phase, le processus lit les composants et collecte les informations concernées. La variété des informations ciblées dépend essentiellement de la norme de service Web considérée, et des concepts syntaxiques ou sémantiques référencés. Généralement, cette identification est réalisée dans une phase de pré-traitement.

b. **Mesurer les similarités**
   Après avoir identifié les objets de données à mettre en correspondance, il est nécessaire d'exécuter la comparaison entre les données identifiées. Cela revient à quantifier la similarité ou ce que nous l'appelons mesurer la similarité. Là encore, le choix d'une mesure de similarité est crucial, et il dépend aussi au format des données choisi et aux algorithmes exécutés. Puisqu'il existe un nombre important des mesures de similarité, le choix est justifier, parfois, durant l'évaluation.

c. **Résultat** l'Affectation des résultats est une autre étape qui consiste à trouver les résultats les mieux adaptés, c'est-à-dire associer à chaque demande l'offre du service le mieux adapté. Celui-ci est généralement réalisé en considérant un seuil de similarité fixé à l'avance. Dans un cas plus compliqué, la demande de service (composite) peut avoir plusieurs offres composant le résultat. Dans ce cas là, il faut affecter à chaque demande les composants les mieux adaptés de l'offre du service en optimisant la somme des seuils. Ceci, est généralement considéré comme un problème d'affectation (Burkard et al., 2009).

5.4 Degré de correspondance (Degree of Match)

Le DOM peut être formellement défini comme une valeur tiré d'un ensemble ordonné de valeurs qui expriment la similarité de deux entités par rapport à une métrique de similarité (Tsetsos, 2007). Cela peut être une valeur de similarité numérique dans un intervalle strictement défini, ou une valeur retirée d'un ensemble prédéfini. Dans le Matching des services basé sur les informations sémantiques, les relations de subsomption basées sur la logique entre les concepts sémantiques d'une ontologie sont souvent utilisées comme DOMs. Comme il a été mentionné dans la section 4.2.1, une description syntaxique des services est insuffisante pour la découverte des services, car elle pourrait entraîner une mauvaise évaluation de l'algorithme de matching. Cela peut être le cas, parce que les services peuvent avoir une description syntaxiquement similaire, mais sémantiquement différente, et vice versa. Dans la section 5.6, nous citons les deux classes de registres « centrales ou distribués ».

5.5 Exemple

En ce qui concerne les services Web, le matchmaking est le processus consistant à trouver des offres de services adaptées pour répondre à une demande de service donné. En conséquence, un moteur de Matching est un outil qui exécute un tel processus. Supposons, par exemple, qu'un consommateur de services ait besoin d'un service qui fournit une distance relative entre deux régions géographiques. Dans un registre, de nombreuses offres de services ont été enregistrées. elles fournissent plusieurs descriptions de services qui offrent la réponse à la demande en question. Ces différentes offres se distinguent par un certain nombre de dimensions qui sont liées aux quatre différents types de sémantique définis par Kashyap and Sheth (1994) (Voir la section 4.2.2). Par exemple, une offre de service peut nécessiter juste le nom (adresse) des deux régions, tandis qu'une autre nécessite leurs emplacements géographiques. Une autre offre de service pourrait permettre de préciser les deux informations en commun. De même, le nombre de paramètres en sortie peut différer les offres de service. Par exemple, un service peut renvoyer tous les chemins possibles avec leurs distances, alors qu'un autre peut renvoyer exactement le plus court.

Un exemple est illustré à la Figure 5.2. La tâche consiste à trouver un service optimal qui fournit une distance relative entre deux régions géographiques données. La demande de service est représentée en haut de la Figure 5.2. Dans cet exemple, la demande est formalisée sous forme de service ; là, c'est un autre problème : il s'agit de la formulation des requêtes pour l'adapter avant d'être passer au moteur

de Matching (Schulte, 2010).

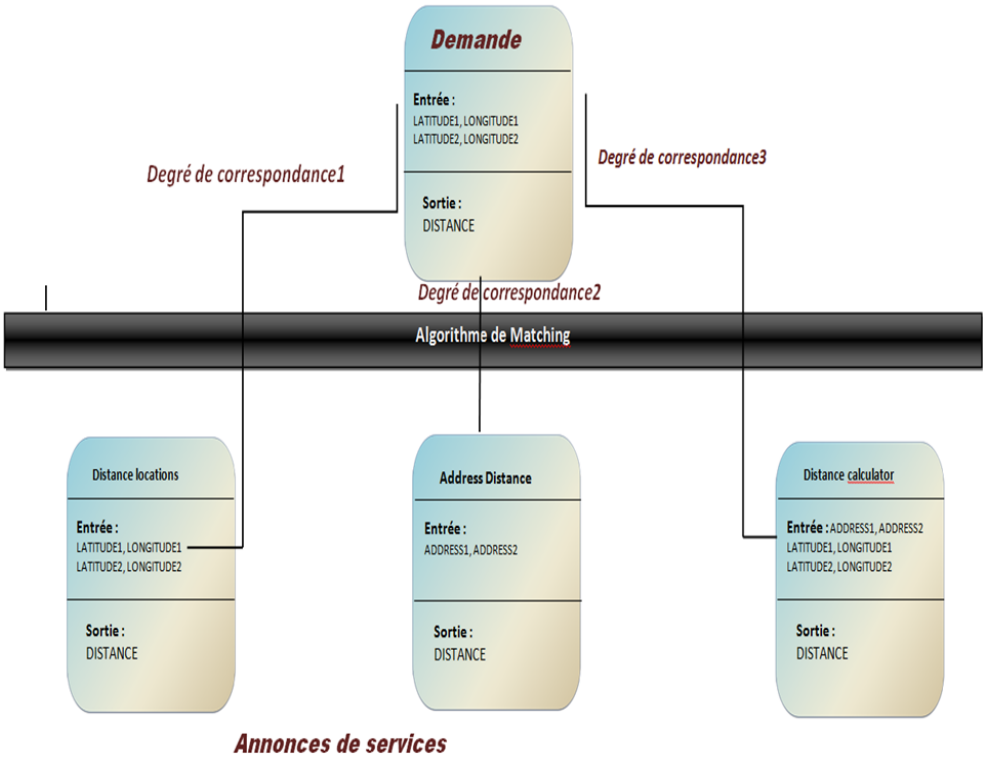

Figure 5.2 – Exemple de la découverte

5.6 Approches de Découverte

Initialement, le processus de découverte des services Web a été implémenté de façon manuelle, où les descriptions de services Web sont découvertes en fonction des besoins de l'utilisateur. Ces descriptions de services sont scannées et comparées manuellement avec les besoins des utilisateurs, où les services répondant aux besoins sont choisis. Dans un environnement d'intégration de systèmes distribués, une telle intervention manuelle est peu pratique, lourde et pénible.

Après avoir instauré des solutions automatiques pour la découverte, les approches peuvent être classées comme : centralisées et décentralisées.

5.6.1 Découverte centralisée des services Web

L'UDDI classique s'inscrit dans une approche entièrement centralisée qui favorise la réplication lorsque plusieurs registres centraux sont utilisés pour stocker les descriptions des services Web. Le maintien de la cohérence dans les registres répliqués conduit à la consommation des ressources, ce qui cause aussi la dégradation des

performances. Ainsi, ces méthodes souffrent d'un nombre de problèmes classiques connus dans les systèmes centralisés, comme le seul point centralisé (Maillon faible) et le coût élevé de la maintenance. De plus, ces méthodes ne garantissent pas l'évolutivité (scalabilité) requise pour soutenir un environnement flexible et dynamique comme le Web (Mandreoli et al., 2007).

Les méthodes centralisées représentent la première génération des travaux proposés pour le processus de découverte. La première initiative industrielle et académique issue pour la découverte des services Web est la recommandation UDDI. Ce dernier a été utilisé de deux façons différentes. La première considère un UDDI comme un annuaire universel (public ou privé), tandis que la deuxième propose d'utiliser une solution indépendante de l'architecture abstraite SOA définie dans la sous-section 2.7, page 16. Cette proposition vise à mettre en place un moteur de recherche pour chercher et découvrir les services Web. Nous présentons, ci-dessous, les solutions centralisées proposées, ainsi que les inconvénients et les problèmes rencontrés au niveau de chaque solution.

Annuaire UDDI public ou privé

Une entreprise peut déployer un ou plusieurs registres UDDI privés et/ou publics. Le registre privé autorise l'accès uniquement aux utilisateurs autorisés, alors qu'il est public lorsqu'il ne restreint pas l'accès au registre. Une entreprise peut choisir de déployer plusieurs registres afin de séparer les informations de service internes et externes. Un registre interne prend en charge les applications intranet, tandis qu'un registre externe prend en charge les applications extranet. Les groupes industriels peuvent déployer un registre UDDI pour prendre en charge les échanges publics ou privés (OASIS, 2004).

La première proposition d'UDDI est publiée en septembre 2000 par Ariba, IBM et Microsoft (Chauvet, 2002) visant à centraliser les points d'entrée pour accéder aux services Web, c'est à dire créer un annuaire universel et public pour les services Web fournis par les différents propriétaires (entreprises ou individus). Cette proposition pose de véritables problèmes de sécurité et de fiabilité. Ainsi, il va y avoir un manque de modération dans les enregistrements UDDI existants. Á cause de ces problèmes, cette solution a été supprimée et remplacée par d'autres, en particulier les annuaires privés. Cette deuxième version d'UDDI a pour but de corriger quelques problèmes apparaissant dans la première version public. En particulier, cette nouvelle version est moins lourde à mettre en œuvre, et elle est en plus adaptée aux échanges B2B privés (B2B : Business to Business sur un Extranet), au commerce électronique (dans un contexte B2C : Business to customers) et même à l'intégration

des applications à l'intérieur de l'entreprise. L'UDDI reste la spécification la plus utilisée, et le standard le plus cité dans les travaux de recherche. Cependant, même dans sa deuxième version, l'UDDI est incapable de supporter les environnements dynamiques et flexibles (Della Valle et al., 2008).

Moteurs de recherche

Les Moteurs de recherche (Search engine) représentent une nouvelle proposition de recherche et de découverte (Liu and Wong, 2009) (différente à l'architecture SOA) basée sur la production des moteurs de recherche similaires à ceux utilisés pour la recherche des pages Web. La recherche des points d'accès aux services Web n'est plus associée aux registres de services, car les moteurs de recherche Web sont devenus une nouvelle source majeure de services Web, de sorte que les fournisseurs des services puissent simplement publier leurs services avec une URL valide, et laisser le moteur de recherche de services faire la collecte, la création de correspondance, la découverte et la composition des services Web.

5.6.2 Découverte décentralisée des services Web

Récemment, plusieurs approches décentralisées ont été proposées. La distribution peut être réalisée en utilisant des technologies distribuées et P2P. En outre, jusqu'à ce jour, il n'y a pas d'architecture distribuée standard recommandée pour un UDDI distribué. Il existe deux techniques de base qui sont utilisées pour implémenter la décentralisation : les Systèmes Multi Agents et les réseaux P2P.

5.7 Conclusion

Dans ce chapitre, nous avons mis le point sur la découverte des services, l'un des processus de la technologie des services Web, et nous avons distingué entre deux solutions : la découverte classique centralisée et distribuée. La distribution de la découverte est apparue récemment comme solution de problèmes produits par l'augmentation continue de l'utilisation de la technologie des services Web. Dans ce qui suit, nous abordons en particulier la deuxième génération des architectures de découverte, c'est-à-dire les architectures distribuées.

Chapitre 6

Annuaires distribués

Sommaire

6.1 Introduction

La technologie de services Web utilise un ensemble de standards, à savoir SOAP, WSDL et UDDI. Particulièrement, l'UDDI est une plateforme destinée à stocker et rechercher les services Web. Ainsi, le problème de découverte des services Web est souvent rattaché à l'UDDI. L'UDDI permet aux entreprises de publier leurs services Web, et par la suite, de les rechercher en s'appuyant sur les préférences des demandeurs. L'UDDI centralisé est une structure moins robuste et qui ne supporte pas un grand nombre de services Web. Si l'environnement distribué est introduit dans UDDI, il y aura une augmentation rapide des services avec minimisation de la consommation des ressources. Dans ce qui suit, nous mettrons en évidence les architectures, et les caractéristiques essentielles d'un futur UDDI distribué. Par la suite, nous présentons les principaux travaux de recherche pour la mise en œuvre d'annuaires distribués, ainsi que les travaux qui abordent le problème du clustering des services Web.

6.2 Structure générale d'un registre UDDI classique

La tâche principale d'un registre UDDI est l'enregistrement des services offerts par les différentes entreprises. Alors, il s'agit d'un fichier XML qui est utilisé pour décrire une entité commerciale avec ses services Web proposés. Les informations fournies dans un enregistrement commercial UDDI sont trois composantes [UDDI](Voir la section 3.4.3, page 26) : les "**Pages blanches**" comprennent les renseignements administratifs, comme les spécifications d'adresse, contact concernant l'accord de licence et de distribution des formulaires ... ; les "**Pages jaunes**" comprennent la classification du service ou de l'entreprise, basée sur des taxonomies standard ; et les "**Pages vertes**" contiennent les descriptions techniques, telles que la façon d'accéder à un service Web (l'emplacement d'un service Web et de son interface). Malheureusement, la spécification de l'UDDI ne fournit aucune information conceptuelle d'un service Web. Par conséquent, les «Pages bleues» doivent être mises en place afin de fournir des informations sémantiques d'un service Web, qui met l'accent sur les termes, les concepts et les relations entre eux.

6.3 Caractéristiques essentielles d'un future UDDI distribué

La mise en œuvre des registres dans un environnement distribué pose de nombreux problèmes (Rajmohan et al., 2011). Pour une meilleure performance, certaines caractéristiques sont essentielles :

- **Efficacité :** l'UDDI doit supporter le stockage et la recherche d'un nombre important de services sans consommer énormément de ressources.
- **Évolutivité :** l'efficacité de la recherche dans l'UDDI ne devrait pas se dégrader lors de l'augmentation de la taille du réseau.
- **Flexibilité :** les descriptions sémantiques des services devront être fournies pour la récupération exacte des informations demandées par les utilisateurs.
- **Exhaustivité de la recherche :** l'UDDI devrait découvrir le maximum d'objets demandés avec une grande précision et un grand rappel.
- **Résistance aux pannes :** l'UDDI devrait maintenir la disponibilité des données et réduire les frais généraux de la défaillance d'un nœud.
- **Équilibrage de charge :** l'UDDI devrait répartir les charges de routage, de stockage et de traitement, selon les capacités des nœuds participants.

6.4 Différentes architectures proposées

En se basant sur le partage du contenu, l'architecture distribuée pour un UDDI peut être :

- **Décentralisée :** Dans une structure décentralisée, les informations de service Web sont réparties dans différents registres UDDI. Le registre UDDI est une collection d'un ou de plusieurs nœuds. Les nœuds sont des serveurs qui prennent en charge la spécification UDDI et qu'ils appartiennent à un registre UDDI. Ci-après, certains travaux décentralisés sont décrits. Un nouveau modèle interopérable (Yu et al., 2008) et un modèle fédéré UDDI (Liang and Chung, 2008) sont utilisés pour décentraliser les registres UDDI. Le METEOR-S Web Service Discovery Infrastructure (MWSDI) (Sivashanmugam et al., 2004) est proposé pour découvrir les services Web dans un environnement fédéré de registres. Une architecture distribuée active appelée Ad-UDDI (Du et al., 2005) est proposée pour organiser le secteur privé ou semi-privé ; elle est basée sur les classifications industrielles. Une architecture qui est déployée pour l'enregistrement flexible et hybride (Sivashanmugam et al., 2004) est proposée pour la gestion des registres UDDI distribués. Ces architectures décentralisées surmontent les problèmes de la défaillance d'un nœud centrale et augmentent les performances des registres.
- **Agent ou middleware :** Un agent est une autorité de confiance autonome qui agit au nom d'un organisme ou d'entités de métier. Ces agents agissent en tant que mandataire d'une communauté de registres UDDI. l'agent reçoit les demandes de service, cherche les services tout en répondant aux besoins des utilisateurs, puis, il retourne la liste des services trouvés. Certains travaux de recherche utilisent la technologie des agents : middleware Architecture (Stanescu et al., 2005) est proposé pour l'organisation des services Web dans la communauté en fonction des domaines spécifiques de métier. Un agent proxy autonome (Maximilien and Singh, 2003) est proposé pour collecter les entrées utilisateur et exécuter la découverte des services. Une architecture basée sur un agent optimal (Rajendran and Balasubramanie, 2010) est proposée pour la découverte dynamique des services Web basée sur un Matching QoS. Toutes ces architectures supportent une infrastructure distribuée des registres UDDI. L'utilisation de la technologie des agents permet :
  - Stockage distribué des descriptions des services.
  - Recherche parallèle.
  - Réduction de la surcharge sur les nœuds et l'augmentation des performances des registres UDDI.

– **P2P :** Dans l'architecture P2P, les fonctionnalités de l'UDDI sont réparties entre tous les nœuds dans un réseau. Le nœud partage ses ressources, telles que la puissance de traitement, la puissance de stockage ou même la bande passante sur le réseau, avec d'autres participants du réseau sans la nécessité d'une coordination d'un nœud centrale. Ces caractéristiques place le P2P comme le plus souple pour la mise en œuvre d'un environnement distribué pour l'UDDI. Avec P2P, nous pouvons accomplir le dynamisme des nœuds, la flexibilité dans l'expressivité des requêtes et la résistance aux pannes. Plusieurs cadres, comme PDUS (Yulin et al., 2010), DWSDM (Lin et al., 2006), Chord (Stoica et al., 2001; Lv and Yu, 2007; Yun et al., 2010), la DHT (Wang et al., 2009), CAN (Sun and Hao, 2010; Channa et al., 2005), et les nœuds sémantiques (Zhong and Ying, 2005), sont proposés pour mettre en œuvre l'architecture distribuée pour l'UDDI.

6.5 Différents Mécanismes de stockage proposés

Dans les architectures distribuées, les informations de répertoire sont stockées dans différents endroits du réseau. Les systèmes distribués peuvent être classés comme répliqués, distribués ou Hybrides :

- **Répliqués :** Dans le cas répliqué, les informations de répertoire sont stockées au niveau des différents nœuds UDDI. Le maintien de la cohérence dans les registres répliqués conduit à la consommation de la bande passante et les ressources. Il résulte aussi la dégradation des performances.
- **Distribués :** Dans le cas distribué, les informations de répertoire sont partitionnées, et ces partitions sont stockées dans différents nœuds de l'UDDI. Lorsque les informations sont distribuées dans les différents nœuds, l'échec de l'un d'eux conduit à l'indisponibilité d'une partie de l'information.
- **Hybrides :** Dans le cas hybride, les nœuds stockent des copies multiples de l'information du répertoire partagé dans différents nœuds de l'UDDI. Les architectures hybrides offrent le meilleur compromis entre consommation de bande passante, évolutivité et tolérance aux pannes.

6.6 État de l'art sur les annuaires distribués

Tous les travaux qui ont résolu le problème de la découverte sont limités par le fait que les services Web offrant les mêmes fonctionnalités peuvent avoir des interfaces et des opérations différentes. Inversement, les services qui ont les mêmes

interfaces peuvent fournir des fonctionnalités complètement différentes (Gomadam et al., 2006; Steinmetz et al., 2008). Il s'agit d'un problème majeur qui affecte de manière articulaire la découverte des services Web. Ainsi, les interfaces et les opérations peuvent être des sources ambiguës pour la découverte. Pour cela, nous devons annoter les interfaces ou utiliser des descriptions sémantiques de services, comme OWL-S et WSMO, qui fournissent un langage ou une ontologie pour décrire les services sans fournir un modèle pouvant être partagé entre plusieurs sites. De plus, si nous utilisons une ontologie (OWL-S ou WSMO) pour décrire un seul service dans deux sites différents, ces deux définitions peuvent ne pas être cohérentes, en raison du niveau de détail de conceptualisation et de perception utilisé par les deux sites. La stratégie "UDDI distribuée" est adoptée pour pallier la dégradation causée par la surcharge des ressources (réseau et registre). Elle surmonte également le problème de l'échec d'un nœud UDDI centralisé. La mise en œuvre de registres dans un environnement distribué pose de nombreux problèmes, notamment la concurrence, le traitement des requêtes, la réplication, la fiabilité, l'évolutivité, etc. (Rajmohan et al., 2011). Même s'il n'y a pas d'architecture distribuée standard pour un UDDI distribué, les approches existantes peuvent être globalement affectées à l'une des trois classes, chacune construite autour d'une idée clé :

1. Approches basées sur la construction d'une ontologie de services.
2. Regroupement de services autour d'une ontologie de domaine.
3. Technologies distribuées.

Approches basées sur la construction d'une ontologie de services

L'infrastructure de découverte des services Web METEOR-S (MWSDI) fournit un accès transparent aux registres publics et privés des services Web (Sivashanmugam et al., 2004). Sivashanmugam et al. (2004) proposent une ontologie de registres étendue (XTRO) qui supporte une classification complexe avec des registres de fédérations. Chaque fois qu'un nouveau registre est ajouté à (MWSDI), le XTRO est mis à jour avec les détails pertinents de ce nouveau registre, ce qui conduit à certains problèmes, tels que la flexibilité, nécessitant d'établir une médiation pour résoudre ce problème. Bianchini et al. (2008) proposent un mécanisme de découverte basé sur une architecture P2P et sur l'interopérabilité entre les nœuds. L'interopérabilité entre les nœuds est réalisée à l'aide d'une ontologie construite par le mappage entre les nœuds. Les résultats seront pertinents puisque tous les registres sont interrogés ; l'inconvénient de cette approche est la difficulté de définir tous les liens sémantiques

entre tous les registres. Khouja and Juiz (2015) proposent de partager un modèle de contexte pour décrire le vocabulaire des services en utilisant une ontologie. Ce modèle est partagé entre les utilisateurs pour faciliter la description de leurs pétitions. Le profil des dispositifs pour les services Web (DPWS) a été intégré dans l'architecture comme un cadre pour l'envoi, la description et la découverte de services Web. Malheureusement, la solution adoptée permet de résoudre uniquement un problème spécifique (environnement du campus). Boukhadra et al. (2013) proposent une approche évolutive et distribuée dans un réseau P2P pour la découverte des services Web sémantiques. Ils proposent d'utiliser le matching d'ontologies OWL-S, et la collaboration entre les différentes nœuds dans le réseau P2P. Cela permet de créer un espace collaboratif, où chaque nœud peut utiliser les expériences des autres, afin de réduire l'espace de recherche. Dans (Boukhadra et al., 2014), la découverte distribuée est basée sur la correspondance d'ontologies OWL-S dans différentes machines. DA5DCSWS, un outil qui implémente la découverte et la composition automatique, contient quatre modules : interface utilisateur, description sémantique des services Web, découverte automatique des services Web, et la composition automatique des services Web.

Regroupement de services autour d'une ontologie de domaine

Canturk and Senkul (2011b, 2012) décrivent un nouveau mécanisme de découverte de services basé sur l'architecture P2P pour réduire le temps de recherche, fournir des mises à jour sur le statut des services et équilibrer la charge sur les registres. Les travaux définissent un mécanisme d'interopérabilité entre les registres qui s'exécute en cinq étapes, ce qui implique un coût de construction élevé. L'architecture adoptée par Sellami et al. (2009) consiste à structurer les registres des services Web en groupes ayant la même ontologie dans un réseau P2P. Bien que cette approche permet de réduire le temps du processus de découverte puisqu'elle scanne une partie des registres, les résultats seront moins pertinents. Folino et al. (2014) proposent un modèle efficace pour la découverte des services Web du data mining. Ce modèle propose une extension de Chord DHT qui prend en compte la taxonomie des services du data mining (Folino et al., 2014). Benghida and Boufaida (2014) proposent un processus de découverte des services Web mobiles, basé sur trois concepts : les réseaux P2P, le service Web mobile et le clustering. L'approche proposée est bien adaptée aux appareils mobiles et au réseau, et elle est caractérisée par la communication multicast, mais la description utilisée des services Web est syntaxique (WSDL). Zhang (2014) propose un modèle de découverte des services Web qui combine les deux technologies : service Web sémantique et P2P. L'approche

consiste à créer une infrastructure de registre pour la publication et la découverte des services Web sémantiques (OWL-S). L'idée consiste à utiliser une ontologie de domaine spécifique pour chaque registre.

Approches basées sur les technologies distribuées

Il faut noter que la majorité des études ont utilisé les technologies distribuées (P2P) comme plateforme. En particulier, Zhang et al. (2013) proposent Chord4S, une approche décentralisée pour la découverte des services basée sur le P2P. Le célèbre protocole P2P Chord est amélioré pour augmenter la disponibilité des données. Chord4S prend en charge la qualité du service pendant la découverte. En outre, le protocole de routage Chord est étendu pour prendre en charge la découverte efficace de plusieurs services avec une seule requête.

Enfin, Sapkota et al. (2006) proposent d'utiliser une nouvelle notion d'espace partagé pour les services Web, où les services Web peuvent être interrogés en fournissant une description en RDF (Graham and Brian, 2004). Cette architecture est fiable, flexible, évolutive, et supporte les communications synchrones et asynchrones. Cependant, elle nécessite des efforts pour protéger l'espace partagé, car elle est ouverte à la mise à jour par les utilisateurs. Dans notre approche (détaillée dans le chapitre 7, page 75), nous introduisons une architecture d'enregistrement/découverte distribuée et évolutive pour surmonter le problème de centralisation de l'UDDI. Notre travail est basé sur la classification des services Web selon une ontologie de services (SO). Une fois mise en œuvre, notre approche permettra un accord entre les utilisateurs et les registres afin de récupérer des résultats pertinents. Le tableau 6.1 résume les approches qui ont abordé le problème de la découverte des services Web. Aussi le tableau 6.2 expose une synthèse de ces travaux :

Tableau 6.1: Approches de la découverte distribuée des services Web

Approche	***Objectif***	***Principe***	***Outil***	***Architecture***	***Méthode Stockage***	***Avantages***	***Inconvénients***
(Sivashanmugam et al., 2004)	Interopérabilité sémantique entre les nœuds	Ontologie étendue + fédération	OWL	P2P + fédération	Distribuée	Interface unique	Problèmes de flexibilité, nécessitant l'établissement des médiations
(Sapkota et al., 2006)	• Découverte fiable, flexible et évolutive • communications synchrones et asynchrones	Espace partagé	RDF	Partagée	Distribuée	Nouvelle stratégie pour la découverte et la composition	Problèmes de sécurité
(Bianchini et al., 2008)	Interopérabilité sémantique entre les nœuds	Liens sémantiques entre les nœuds (liens intra-peer et inter-peer)	P2P-SDSD VS Gnutella	P2P	Hybride	Résultats pertinents puisqu'il interroge tous les registres	Difficulté de définir les liens sémantiques
(Sellami et al., 2009)	• Réduire le temps de recherche, • Equilibrer la charge sur les registres.	Ontologie de domaine + système de recommandation	Plateforme JXTA YASA4WSD	P2P	Hybride	Temps de réponse raisonnable	Résultats moins pertinents

Approche	*Objectif*	*Principe*	*Outil*	*Architecture*	*Méthode Stockage*	*Avantages*	*Inconvénients*
(Canturk and Senkul, 2011b, 2012)	• Réduire le temps de recherche. • Qualité de résultat. • Equilibrer la charge sur les registres.	Ontologie de domaine	/	P2P	Hybride	• Interface unique. • Mise à jour des QOS.	Coût de construction élevé
(Boukhadra et al., 2013, 2014)	Composition et découverte distribuée	Alignement d'ontologie	OWL-S	Décentralisée	Distribuée	• Travail collaboratif entre les nœuds. • Architecture purement distribuée et hétérogène.	QoS
(Zhang et al., 2013)	• Une grande disponibilité. • Equilibrer la charge sur les registres.	Amélioration de Chord	Chord	P2P	Distribuée	• Amélioration de Chord pour supporter QoS. • Une grande disponibilité. • La découverte de plusieurs services similaires.	/

Approche	***Objectif***	***Principe***	***Outil***	***Architecture***	***Méthode Stockage***	***Avantages***	***Inconvénients***
(Folino et al., 2014)	Découverte efficace des services	Catégorisation des Services Web	Chord DHT	P2P	Distribuée	Découverte efficace dans les réseaux à grand échelle.	Une taxonomie spécifique " modèle de services produit par extraction de données "
(Zhang, 2014)	Découverte distribuée	Ontologie de domaine + P2P	Chord4S	P2P	Distribuée	Méthode évolutive d'accès aux registres.	/
(Benghida and Boufaida, 2014)	La découverte des SW dans un environnement mobile	Clustering + P2P + service Web mobile	JXTA/JXME	P2P	Distribuée	Environnement mobiles	La description des SW est représentée par WSDL.
(Khouja and Juiz, 2015)	Modèle de contexte basé sur une ontologie pour décrire le vocabulaire des services	Contexte partagé (ontologie)	DPWS (Web Services for Devices)	Partagée	Distribuée	Réseau mobile + Améliore le taux de découverte (nombre moyen de services découverts par demande)	L'ontologie utilisée est spécifique (domaine campus).

Tableau 6.2: Synthèse des travaux exposés

Approche	*Clustring*	*Ontologie de service*	*Système de recommandation*	*P2P*	*Espace partagé*
(Sivashanmugam et al., 2004)	+	+	−	+	-
(Sapkota et al., 2006)	-	-	-	-	+
(Bianchini et al., 2008)	+	+	-	+	-
(Sellami et al., 2009)	+	-	+	+	-
(Canturk and Senkul, 2011b, 2012)	+	-	-	+	-
(Boukhadra et al., 2013, 2014)	-	+	-	+	-
(Zhang et al., 2013)	+	-	-	+	-
(Folino et al., 2014)	+	-	-	+	-
(Zhang, 2014)	+	-	-	+	-
(Benghida and Boufaida, 2014)	+	-	-	+	-
(Khouja and Juiz, 2015)	+	+	-	-	-

6.7 État de l'art sur le regroupement des services Web

Les services Web sont le fruit de la programmation à base de composants qui s'appuie profondément sur le concept d'objet distribué (Vinoski, 1993) ; alors, dès le début des années 2000, les chercheurs ont commencé à introduire sérieusement le terme "Service Web" (Glass, 2000; Roy and Ramanujan, 2001) en essayant de répondre aux défis d'interopérabilité, de découverte, de sécurité, d'évolutivité, etc. Après la mise en place de l'architecture, les défis ont été rapidement rénovés vers l'automatisation des différents processus de la technologie (découverte, composition, etc.). Les solutions adoptés affirment que ce problème doit être résolu essentiellement par l'augmentation de la représentation des interfaces des services et l'identification des similarités sémantiques (Paolucci et al., 2002a,b). Récemment, et en raison de l'augmentation continu du nombre des services Web, les recherches sont principalement axées sur le clustering des services Web adoptant la Machine Learning. De même, dans le domaine du Web sémantique, plusieurs travaux ont proposé une approche automatique ou semi-automatique pour construire une ontologie de domaine (Singh and Aswal, 2019). Cependant, dans le contexte des services Web, les algorithmes d'apprentissage automatique sont principalement utilisés pour classer (Cao et al., 2019; Sánchez-Sánchez and Sheremetov, 2018) et regrouper les services Web. Dans notre thèse, nous proposons d'introduire et de construire une ontologie de ser-

vices de façon semi-automatique par regroupement (Voir la section 7.4.1, page 81). D'abord, nous présentons, dans la suite, les travaux de recherche les plus célèbres qui abordent fondamentalement les problèmes de regroupement des services Web et/ou de construction d'ontologies de services (Voir le Tableau 6.3).

Richi and Lee (2007) proposent une méthode dénommée SWSC pour regrouper les services Web afin d'améliorer la recherche des services Web dans les registres. La méthode comporte en trois étapes : la première, dite Pre-Processing, a pour objectif de préparer et d'analyser les fichiers WSDL ; la deuxième est une étape de transformation dans laquelle les fichiers WSDL sont transformés en fichiers OWL-S en leurs ajoutant de la sémantique, l'outil WSDL2OWL-S (Paolucci et al., 2003) est utilisé pour cette transformation ; la troisième étape est une étape de clustering qui consiste à regrouper les services Web en utilisant l'algorithme hiérarchique ascendant (Karypis et al., 1999). (Funk and Bontcheva, 2010) vise à créer des catégories (ontologie des catégories) pour cibler la découverte. Cette catégorisation a été créée manuellement, et elle est basée essentiellement sur les annotations des utilisateurs. (Liang et al., 2009) décrit une approche basée sur le clustering des services Web afin de former une hiérarchie de taxonomie des services (comme UNSPSC). Ce schéma de regroupement prend en considération non seulement les facteurs individuels, tels que l'entrée et la sortie du service, mais également les interrelations latentes entre les facteurs individuels. Lorsqu'un nouveau service est publié, la description de ce service est comparée avec chaque cluster de l'hiérarchie de la taxonomie ; sur la base de ce calcul, le service sera affecté au cluster le plus approprié. Pour construire l'hiérarchie des services, les auteurs ont implémenté une méthode de clustering en trois étapes : l'étape de prétraitement qui s'occupe de convertir chaque document WSDL en une arborescence après avoir appliqué stemming algorithm (Liang et al., 2008) et supprimer les mots vides ; la deuxième étape (nommée clustering approximatif) marque chaque service Web avec une balise de classe en fonction des similitudes des descriptions textuelles fournies en utilisant l'outil UNSPSC et en appliquant l'algorithme classique k-means (MacQueen et al., 1967) ; la troisième étape (nommée fine clustering) regroupe l'ensemble des services Web selon un certain nombre de clusters en tenant compte les informations d'opérations des services, plus les entrées et les sorties des opérations et leurs relations. Pour l'implémentation de ce clustering, les auteurs ont utilisé l'algorithme of bisecting k-means (Steinbach et al., 2000). El Bouhissi and Malki (2014) proposent une approche pour produire une ontologie WSMO-lite à partir des fichiers WSDL fournis en utilisant une démarche d'ingénierie logicielle (Reverse Engineering). Durant le processus d'ingénierie, les auteurs ont implémenté une phase de classification qui consiste à associer un service à son

domaine correspondant. Cette étape est entièrement automatique permettant de calculer la similitude entre les entités du fichier WSDL avec toutes les entités des ontologies disponibles. Sur la base de ces similitudes, le système choisit une ontologie de domaine parmi plusieurs fournies. L'ontologie de domaine sélectionnée détermine la catégorie du document WSDL. Ensuite, elle sera utilisée pour annoter les fichiers WSDL dans la phase d'annotation. (Liu and Wong, 2009) décrit une nouvelle méthode (différente à l'architecture SOA) de recherche et de découverte des services Web basée sur la production des moteurs de recherche pour les services Web, de sorte que les fournisseurs de services puissent simplement publier leurs services avec une URL valide, et laisser le moteur de recherche des services faire le clustering, la classification, la création de correspondance, et la composition. Le regroupement des services Web est implémenté dans le sens de produire un arbre de parcours des services Web basé sur l'algorithme de colonie de fourmi implémentant ant-traversing ant (Wong et al., 2007). (Shen and Liu, 2019) décrit une approche de découverte des services Web basée sur deux étapes : le clustering des services Web et la sélection des services Web. Pour le processus de clustering, les auteurs ont considéré les textes de description (balise dans OWL-S) pour représenter les services Web comme des vecteurs à l'aide de quatre méthodes non-supervisées de représentation de phrases ; k-means (MacQueen et al., 1967) est utilisé pour regrouper ces vecteurs. Pour la sélection, LDA (Blei et al., 2003) est utilisé pour extraire les informations sémantiques dans un cluster spécifique face à une demande de service Web. Dans (Bravo et al., 2019), un algorithme hybride pour le clustering de services Web a été proposé ; il est basé sur l'algorithme d'optimisation "Artificial Bee Colony" (ABC) combiné avec K-means et la méthode Consensus. Cette hybridation a été proposée pour rendre l'algorithme ABC auto-ajustable pendant chaque itération, pour décider le nombre de clusters. Le processus de clustering commence par l'extraction et le traitement des documents WSDL, puis le calcul des similarités sémantiques basée sur WordNet entre les paires d'opérations des services Web. Le tableau 6.3 récapitule les approches qui ont abordé le problème de regroupement des services Web :

Tableau 6.3 – Approches pour le regroupement et la classification des services Web

Reference	*Entrée*	*Sortie*	*Algorithme*	*Expériences*
Richi and Lee (2007)	WSDL transformé en OWLS	Catégorisation des services Web	Algorithme de clustering Hiérarchique ascendant	Qualité du résultat
Funk and Bontcheva (2010)	Fichier WSDL annoté manuellement par les utilisateurs	Catégorisation des services Web	Catégorisation manuelle par mots-clés	Découverte efficace
Liang et al. (2009)	Fichiers WSDL	Taxonomie des services	K-means & Bisecting K-means	
El Bouhissi and Malki (2014)	Fichiers WSDL	Ontologie WSMO-lite	Reverse engineering	
Liu and Wong (2009)	Fichiers WSDL	Catégorisation des services Web (tree-traversing)	Text mining & colonie de fourmi	Moteur de recherche pour services
Shen and Liu (2019)	OWL-S (balise Text-description)	Catégorisation des services Web	K-means	Algorithmes d'apprentissage non supervisé
Bravo et al. (2019)	Fichiers WSDL	Catégorisation des services Web	Artificial Bee Colony (ABC) combiné avec K-means et la méthode Consensus	Regroupement de très grandes collections de Services Web
Wang (2021)	OWL-S (balise Text-description)	Catégorisation des services Web	AGNES (Agglomerative Nesting)	

6.8 Conclusion

Dans ce chapitre, nous avons vu les principaux travaux qui abordent la distribution de la découverte des services Web. Chaque approche énoncée a proposé une stratégie pour constituer la distribution des services Web. De ce fait, nous pouvons les classées en plusieurs axes (Voir le tableau 6.2) : le regroupement des services selon les domaines ontologique (Canturk and Senkul, 2011b; Sellami et al., 2009; Bianchini et al., 2008; Sivashanmugam et al., 2004) ; la création d'une ontologie de services pour répartir les services (Bianchini et al., 2008; Sivashanmugam et al., 2004) ; l'utilisation des P2P comme plateforme de collaboration entre les registres UDDIs (Canturk and Senkul, 2011b; Sellami et al., 2009; Bianchini et al., 2008; Sivashanmugam et al., 2004). Certains travaux ont utilisé un système de recommandation pour diminuer le temps de réponse (Sellami et al., 2009). Finalement, Sapkota et al. (2006) proposent un nouveau concept pour la découverte basé sur les espaces partagés. Ainsi, nous avons cité, dans un deuxième état de l'art, les travaux de recherche les plus célèbres qui abordent fondamentalement les problèmes de regroupement des services Web et/ou de construction d'ontologies de services (Voir le Tableau 6.3).

Dans le chapitre qui suit, nous proposons une approche fondée sur l'intersection de deux axes : le regroupement des services et l'utilisation des ontologies. Aussi, nous proposons la création d'une ontologie de services utilisée comme un modèle pour répartir et regrouper les services dans les unités de stockage.

Troisième partie

Approche Proposée

Chapitre 7

Approche Proposée

Sommaire

7.1 Introduction

La stratégie "UDDI distribué" est adoptée pour surmonter la dégradation des fonctionnalités due à la surcharge des ressources (registre et réseau). Elle permet aussi de pallier le problème de défaillance d'un nœud UDDI centralisé. La mise en œuvre des registres dans un environnement distribué a fait surgir de nombreux problèmes, notamment l'accès simultané, le traitement des requêtes, la réplication, la fiabilité, l'évolutivité (scalabilité), etc. (Rajmohan et al., 2011). Ces problèmes peuvent être résolus en utilisant des technologies distribuées et P2P. Dans le chapitre 6 (page 58), nous avons discuté les principaux travaux pour les annuaires distribués ; chacun a utilisé une technique pour collaborer ou partager les registres. Parmi ces techniques, il y a l'utilisation d'une ontologie de services ou le clustring des services. Notre approche, détaillée dans ce qui suit, se base principalement sur la création d'une ontologie employée pour bien regrouper les services dans les registres. Ce regroupement va garantir la diminution du temps de la découverte, l'évolutivité du système, ainsi que l'interopérabilité entre les registres.

7.2 Le cadre général de l'approche proposée

Le problème de la découverte des services est très similaire à celui de la recherche d'information. Il comporte deux sous-questions :

- Identification des besoins : ce qu'on désire trouver, est ce qu'il est bien exprimé ? est ce qu'il est bien interprété ?
- Identification de l'information : ce qui est disponible, est-il bien présenté ? est-il bien classée pour être trouver ?

Le problème classique de recherche d'information est traité comme un problème de classification (index, bloc). Ainsi, dans notre contexte, nous supposons que : si nous classons mieux les services, nous les distinguons et nous pouvons mieux les découvrir. La classification est l'une des solutions adoptées pour résoudre les problèmes d'intelligence artificielle. Le problème de la découverte est vu comme un problème d'intelligence artificielle, où il est possible d'utiliser la classification comme solution.

En outre, l'ontologie est souvent utilisée dans la RI, en particulier dans l'indexation sémantique des documents, l'organisation des résultats de recherche et le raffinement automatique des requêtes (Sendhilkumar et al., 2009). L'ontologie est également utilisée dans les services Web sémantiques, tels que les services d'indexation (Mashayekhi et al., 2008) et l'annotation sémantique (Canturk and Senkul, 2011a).

De plus, Canturk and Senkul (2011b) définissent une solution dans un environnement distribué en introduisant une classification par domaine. le domaine est utilisé comme une information commune entre la demande et les services Web publiés. Avec cette façon, l'espace des services Web sollicités devient plus fin, c.-à-d. au lieu de chercher dans tous les services publiés, la recherche se fait juste dans les services ayant le même domaine.

Dans notre approche, nous avons construit un référentiel partagé, modélisé par une ontologie de services qui a été déployée sur une architecture distribuée pour mieux stocker les services Web, et ainsi réduire le temps de découverte des services. Pour cela, nous proposons d'étendre la classification de (Canturk and Senkul, 2011b) en construisant une architecture distribuée à base d'une ontologie de services par l'ajout d'autres informations, telles que les sous-domaines et les services abstraits. L'ontologie est employée comme un modèle partagé utilisé dans les deux processus : découverte et publication, déployée sur une architecture distribuée.

7.2.1 Modèle de représentation de service

L'utilisation des ontologies pour les services est énoncée, auparavant, dans OWL-S et WSMO pour décrire les services Web. Comme contribution, nous proposons l'utilisation d'ontologie pour développer une base de connaissance utilisée dans les différents processus de la technologie des services Web. Pour cela, un modèle pyramidal à 4 niveaux est proposé pour la représentation des services Web (Voir la Figure 7.1). La relation entre les niveaux de représentation des services Web est décrite à travers le modelé de classe de la Figure 7.1.

- Niveau Entreprise : où chaque Entreprise peut exprimer plusieurs domaines.
- Niveau Domaine : où chaque domaine est identifié par une clé D et renferme plusieurs services abstraits.
- Niveau Service Web Abstrait : où chaque service Web abstrait est identifié par une clé C et contient plusieurs services Web publiés.
- Niveau service Web : où chaque service Web est identifié par le couple $< D, C >$.

Les couches domaine et service Web abstrait constituent le référentiel partagé (l'ontologie de services) pour la représentation des connaissances sur les services Web. Ce même référentiel va être utilisé pour la publication, le stockage et la découverte.

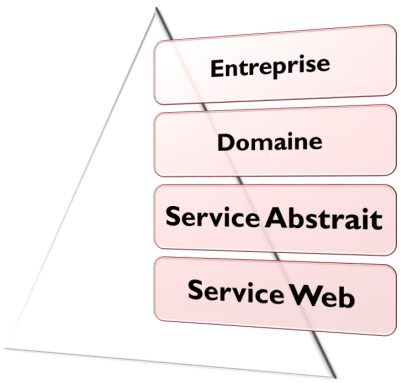

Figure 7.1 – Modèle pyramidal pour la représentation des services Web

7.2.2 L'architecture fonctionnelle

L'architecture fonctionnelle de notre système de publication (stockage) et de découverte de services Web repose sur trois couches (Voir la Figure 7.2). La couche service, la couche connaissance et la couche stockage/découverte. Cette répartition en couche va nous permettre de faciliter la conception du système, de réduire la

complexité de son développement et d'assurer l'évolution de chaque couche indépendamment des autres :

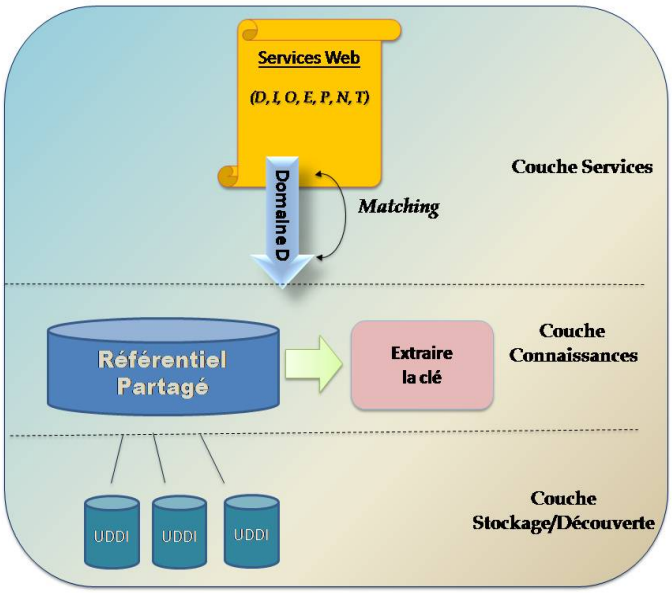

Figure 7.2 – Architecture fonctionnelle du système proposé

1. Couche service : l'objectif de cette couche est de préparer les services à stocker (ou à découvrir) pour qu'ils soient affectés à un domaine précis.
2. Couche connaissance : en se basant sur l'ontologie des services (SO) proposée, cette couche fournit une clé $< D, C >$ à travers le matching entre le service Web (SW) et l'ontologie SO.
3. Couche Stockage/Découverte : indépendamment des deux couches supérieures, cette couche sélectionne les unités de stockage participant à une requête de publication ou de découverte à partir de la clé $< D, C >$ fournie par la couche connaissance.

7.3 Matching

Comme il est présenté dans la figure 7.2, le Matching est exécuté sur le référentiel partagé (SO). Ce référentiel expose une vue simplifiée des services stockés. Ceci réduira l'espace de recherche, et par conséquent le temps de découverte diminuera, ce qui est un facteur positif et essentiel pour un tel système distribué.

7.3.1 L'algorithme de Matching

Le Moteur de Matching est un algorithme qui vise à calculer la correspondance entre les annonces de service exprimées dans les registres UDDIs et les demandes envoyées par les clients (Voir la section 5.3, page 50). L'approche proposée réalise la découverte en comparant la demande avec SO (Voir la Figure 7.2). Préalablement, la requête est formalisée sous forme d'un service Web, où il faut l'introduire sous forme d'un ensemble contenant : Domaine D_R ; les entrées I_R ; les sorties O_R ; avec les pré et les post-conditions (P_R, E_R). L'algorithme (Voir l'Algorithme 1) va parcourir l'ontologie SO (référentiel partagé) tout en comparant la demande du client avec les concepts de SO. Pour la première comparaison (fonction Comparer1), nous comparons le domaine de la requête D_R avec celui du service abstrait N. Alors, les services abstraits retenus (ayant une similarité supérieure au seuil fixé) seront insérés dans la liste F. Ainsi, pour la deuxième comparaisons (fonction Comparer2), nous comparons les services abstraits retenus dans F avec la demande. Puisque la requête et le service abstrait avaient les même caractéristiques, la comparaison sera exécutée deux à deux entre leurs entités (concepts). Les services abstraits retenus, leurs Clés C_A seront retenues comme résultat.

Algorithme 1 Matching

Entrée(s):
Requêtte $R = (D_R, I_R, O_R, P_R, E_R)$; ▷ /* La Demande */
Ontologie $SO(D, SD, SA)$ /
$D(N, T)$ ▷ /* Domaine */
$SD(N, T)$ ▷ /* Sous domaine */
$SA(D_A, I_A, O_A, P_A, E_A, C_A)$ ▷ /* Service abstrait */
Sortie(s): Un ensemble de Clé(s) dans C.

1: **Initialization :**
$F \leftarrow \varnothing$, $C \leftarrow \varnothing$: *deux listes*;
2: **Parcourir** SO :
3: **pour** *Chaque* SD **faire**
4: **si** $Comparer1(D_R, N) \geq Seuil$ **alors**
5: $F \leftarrow F + SD$;
6: **fin si**
7: **fin pour**
8: **Fin de Parcours**

9: **pour** *Chaque* $SA \in F$ **faire**
10: **si** $Comparer2(SA, R) \geq Seuil$ **alors**
11: $C \leftarrow C + C_A$;
12: **fin si**
13: **fin pour**

Pour réaliser la comparaison entre les concepts (entités), nous avons opté pour les mesure de similarité basées sur WordNet (Pedersen et al., 2004). La mesure de similarité va quantifier à quel point deux entités sont similaires. Dans ce cadre, le seuil est une valeur comprise entre 0 et 1 ; la valeur 1 indique qu'il y a une équivalence sémantique totale entre les deux entités.

7.4 Le référentiel partagé SO

Le référentiel partagé est la partie clé dans l'approche, là où il représente la connaissance (knowledge) employée pour les deux processus : découverte et stockage. Aussi, il est important puisqu'il se trouve au centre de l'architecture proposée ; il réalise l'accord entre les deux couche "services" et "découverte/stockage". Bien que durant la première implémentation, nous avons tester l'approche avec SO construite purement de façon manuelle, dans ce qui suit, nous détaillons une proposition d'une construction semi-automatique. Cette dernière, est réalisée en regroupant les services les plus similaires. Malgré que, les autre travaux ont considéré un regroupement basé sur le domaine ontologique, notre idée clé est de créer une deuxième catégorisation dans chaque domaine où les clusters seront plus homogène.

7.4.1 Construction semi-automatique de SO

Notre travail vise à construire l'ontologie SO de façon semi-automatique à partir d'un énorme nombre de services Web. Pour cela, la méthodologie employée se repose essentiellement sur la sélection d'un corpus de services Web sémantique, le choix d'une métrique, et le choix d'un algorithme de regroupement. Alors pour le premier, nous avons pu choisir un extrait du corpus open source OWLS-TC 4.0 (OWLS-TC-V4) cette version décrit un ensemble de services Web sémantiques sous forme de documents OWL-S [1]. Le corpus comportait 1083 services segmentés en différentes classes selon 09 domaines ontologique : éducation, soins médicaux, nourriture, voyage, communication, économie, armes, géographie et simulation. Notre objectif était la création des clusters au sein de chaque domaine (deuxième catégorisation), où les clusters deviennent plus homogènes et plus fines. le travail réalisé est limité à 05 domaines comme point de départ : alimentation, voyage, communication, arme et géographie. Pour le deuxième choix nous avons opté pour l'algorithme k-means (MacQueen et al., 1967) qui a été ajusté pour notre implémentation. La Figure 7.3 montre notre perception initiale du squelette de l'ontologie des services.

1. La majorité de ces services sont extraits de l'annuaire UDDI d'IBM, et sont traduits du format WSDL au format OWL-S de façon semi-automatique.

Une fois SO construite, elle sera utilisée pour le stockage d'un service entrant selon sa classe (cluster) correspondante, et elle sera également utilisée pour découvrir un service demandé déjà existant dans cette ontologie, sans ignorer qu'une meilleure classification conduit à un meilleur stockage, qui à son tour conduit à une découverte efficace des services. Enfin, et en ce qui concerne la métrique utilisée, nous avons choisi la métrique de Jaccard (Jaccard, 1902). Cette dernière, mesure le degré d'équivalence de deux ensembles considérés. Le choix de Jaccard (Jaccard, 1902) sera justifié dans la sous-section qui se suit. Donc, le processus d'implémentation comprend trois étapes bien distinctes : le prétraitement, le regroupement, et la génération des concepts comme le montre la Figure 7.4.

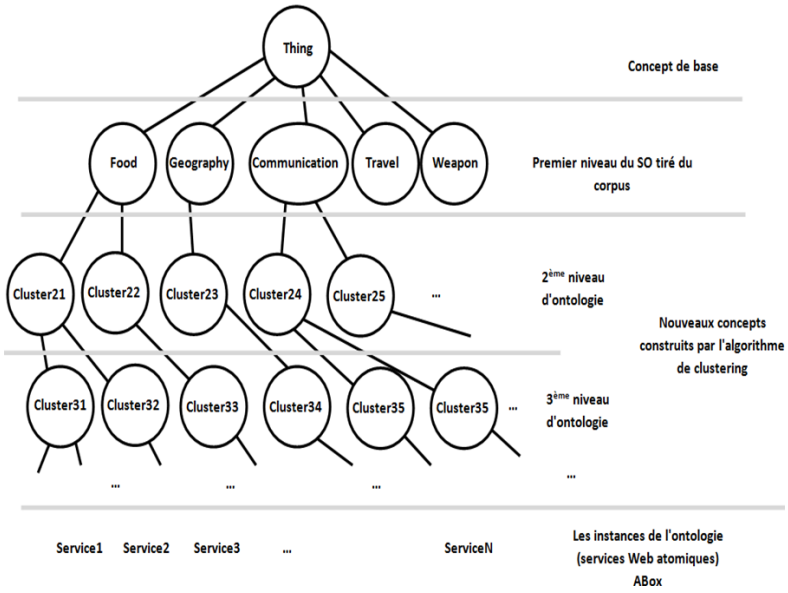

Figure 7.3 – Présentation SO

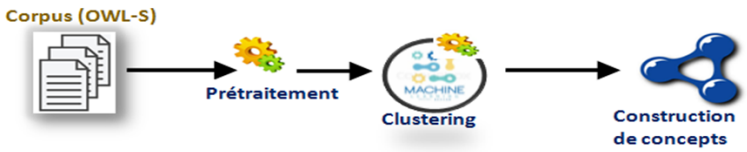

Figure 7.4 – Principales étapes du processus de génération du SO

Prétraitement

La phase de prétraitement analyse les fichiers OWL-S sélectionnés dans le corpus et retient les services atomiques, puis extrait la partie abstraite de chaque service.

La partie abstraite dans un fichier OWL-S est décrite dans la balise profile, où on trouve les entrées, les sorties, etc. Puis, elle va construire un arbre qui sera exploité comme entrée dans la phase qui suit (clustering).

Clustering

En se référant à l'algorithme 2, le processus de catégorisation employé consiste à implémenter l'algorithme classique k-means (MacQueen et al., 1967) qui se divise en deux étapes : l'initialisation puis le processus itératif de regroupement. L'initialisation est une affectation d'une valeur initiale, où il faut estimer le nombre de clusters et repérer les points de départ. Nous décrivons ci-après l'initialisation et la partie itérative :

Algorithme 2 Clustering K-means (MacQueen et al., 1967).

Entrée(s):
S_i : ensemble de services
K : Nombre du groupes
K points de départ.
Sortie(s): *S_i partitionnés en K groupes*

1: **Initialisation :** Choisir K services qui forment K clusters appliquant l'algorithme 3.
2: **Iteration :**
3: **pour** chaque $s \in S_i$ **faire**
4: (Ré)affecter s au groupe C_k ayant le centre M_k tel que $Jaccard(s, M_k)$ soit maximale.
5: **fin pour**
6: Pour chaque K groupe (re)calculer le centre M_k.
7: Aller à 2 jusqu'à stabilisation.

1. **L'initialisation :** L'initialisation dans l'algorithme k-means (MacQueen et al., 1967) est cruciale ; le résultat dépend toujours de la valeur initiale et des points de départ. Évidemment, un meilleur résultat nécessite une meilleure initialisation. Sachant que k-means est sensible à cette initialisation, il était nécessaire de trouver une méthode pour prédire la valeur initiale et les points de départ. Dans la littérature, il existe plusieurs procédures pour estimer la configuration initiale (Steinley and Brusco, 2007; Steinley, 2003). La plupart des travaux estiment cette initialisation en vérifiant et en exécutant l'algorithme plusieurs fois (Celebi et al., 2013). D'autres utilisent des méthodes hiérarchiques pour faire cette estimation (Su and Dy, 2004). Le choix dépend donc des problèmes

abordés et des domaines étudiés. Dans notre contexte, nous proposons d'utiliser la méthode (Astrahan, 1970). Comme le montre l'Algorithme 3, cette méthode est basée sur le choix d'un rayon de distance $R1$. Pour chaque point de données, nous calculons le nombre de points de données qui se trouvent dans $R1$, ce que nous appelons la densité de points. Le premier grain de données choisi pour l'initialisation est celui qui possède la densité la plus élevée. Les autres grains sont choisis par densité décroissante, à condition qu'ils soient supérieurs à un minimum pré-spécifié $R2$ et dont les densités soient supérieures à un (1). Selon (Astrahan, 1970), il est acceptable de définir $R2$ égal à $R1$. Dans notre implémentation, l'indice de Jaccard est utilisé pour calculer le rayon. Cette similarité ($1 - distance$) sera détaillée formellement dans la prochaine sous-section. L'indice de similarité de Jaccard compare les membres de deux ensembles pour voir quels sont les membres partagés et ceux qui sont distincts. Il s'agit d'une mesure de similarité de deux ensembles de données, avec une plage comprise entre 0 et 1. 1 indique qu'il existe une équivalence complète entre les deux ensembles.

Algorithme 3 Initialisation (Astrahan, 1970)

Entrée(s): Définir $R1$ et $R2$ deux rayons de similarité.
Sortie(s): Indicateurs d'initialisation.

1: **pour** Chaque point x_i **faire**
2: Calculer la densité D (le nombre de points qui se trouvent dans le cercle de rayon $R1$ centré sur x_i)
3: **fin pour**
4: Le premier point de données ayant la densité la plus élevée est choisi comme première grain de cluster.
5: Les $k - 1$ grains restants sont choisis en diminuant la densité, tant qu'ils soient supérieurs à un minimum pré-spécifiée $R2$, de tous les grains déjà choisis.

2. **Partie itérative :** Comme indiqué dans l'Algorithme 2, pour chaque itération, l'algorithme calcule la similarité entre les services et les centres. Ensuite, il recalcule les nouveaux centres. Par conséquent, il était nécessaire de quantifier la similarité, et de trouver un moyen pour calculer le centre d'un ensemble de services Web. Donc, si nous considérons les services Web comme un ensemble de concepts d'une ontologie de domaine, nous pourrons utiliser l'indice de similarité de *Jaccard*, puisqu'il permet de calculer la similarité entre les

ensembles. Notons donc S_i un service Web, tel que :

$$\begin{cases} S_i = \{In_i, Out_i\} \\ In_i = \{C_{in1}, ..., C_{inN}\} \\ Out_i = \{C_{out1}, ..., C_{outM}\} \end{cases} \tag{7.1}$$

où C_{inN} et C_{outM} sont les concepts d'entrée et de sortie de S_i. Ainsi, la similarité de $Jaccard$ entre deux services S_1 et S_2 definit comme suit :

$$Jaccard(S_1, S_2) = \frac{(\frac{\|In_1 \cap_t In_2\|}{\|In_1 \cup_t In_2\|} + \frac{\|Out_1 \cap_t Out_2\|}{\|Out_1 \cup Out_2\|})}{2} \tag{7.2}$$

avec $\cup_t$ est l'opérateur d'union avec un seuil t ($\cup$ est considéré sans double, alors deux concepts sont considérés identiques s'ils auront une similarité supérieure ou égale à t) en utilisant les mesures de similarité à base de WordNet (Pedersen et al., 2004) (de même pour $\cap_t$). En ce qui concerne le calcul du centre, nous considérons que le centre d'un ensemble de services est un service synthétique. Ainsi, si nous avons $S_1, .., S_i$ un ensemble de services du même cluster, le centre C_j de ces services est un service synthétique calculé en appliquant l'union des concepts d'entrées et de sorties de tous ces services, tel que :

$$C_j = \{In_j, Out_j\} = \left\{ \bigcup_{j=0,}^{N} In_j, \bigcup_{j=0,}^{M} Out_j \right\} \tag{7.3}$$

7.4.2 Génération des concepts

La construction d'ontologie de manière semi-automatique a fait l'objet de plusieurs travaux dans plusieurs domaines (Mazari et al., 2012; Jung et al., 2010). Dans notre approche, nous proposons l'utilisation des algorithmes de clustering pour cette construction. Dans cette section, nous avons expliqué l'application de l'algorithme k-means dans le processus de clustering. L'étape suivante consiste à produire les concepts ontologiques pour notre ontologie. Pour cela, nous proposons de considérer les centres. Alors, chaque centre est extrait pour construire un concept.

7.5 Conclusion

Dans ce chapitre, nous avons présenté notre approche pour la distribution de la découverte des services Web à base d'une ontologie, qui est utilisée non pas pour décrire les interfaces des services Web comme OWL-S (Martin et al., 2004) et WSMO (Roman et al., 2005), mais comme un modèle partagé pour répartir les services Web entre les registres. Ainsi, nous avons présenté la structure de l'ontologie SO. Ensuite, nous avons décrit le principe de fonctionnement de notre processus de découverte basé sur trois couches : service, connaissance et stockage. Chaque couche est responsable d'une tâche bien définie, tout en collaborant avec la couche voisine pour satisfaire le processus de découverte. Nous avons décrit aussi le scénario du Matching tout en montrant l'avantage de notre Matching. Ainsi, nous avons expliqué brièvement les éventuelles architectures pour implémenter la structure des registres. Finalement, nous avons détaillé le processus réalisant la construction de l'ontologie SO. Dans le chapitre qui suit, nous allons présenter l'implémentation ainsi que l'évaluation de notre approche tout en montrant les étapes et les outils utilisés.

Chapitre 8

Implémentation et évaluation

Sommaire

8.1 Introduction

Nous avons détaillé, dans le chapitre précédent, l'architecture proposée pour la découverte des services Web. L'architecture fonctionnelle proposée pour la publication (stockage) et pour la découverte repose sur trois couches (voir la Figure 7.2) : service, connaissance et stockage/découverte. Ainsi, comme il a été présenté dans la même Figure 7.2, Le Matching est exécuté entre la demande de l'utilisateur et le référentiel partagé (OS).

Dans ce chapitre, nous décrivons les prototypes implémentant l'approche proposée. La mise en œuvre s'effectue en deux phases : la construction de l'ontologie des services OS, et la mise en œuvre de l'algorithme de Matching. Dans ce qui suit nous présentons, en détail, les parties implémentation et évaluation de chaque phase.

8.2 Construction de l'ontologie

8.2.1 L'outil implémenté

Pour les expériences de la construction de SO, nous avons implémenté en utilisant l'environnement éclipse Indigo un outil prototype appelé "Générateur SO". Nous avons utilisé l'API JENA pour lire les fichiers OWL-S, et nous avons employé WordNet API (Pedersen et al., 2004) pour calculer la similarité entre les concepts. Les étapes prétraitement et clustering sont également implémentées en langage Java. La Figure 8.1 représente l'interface principale de l'outil "Générateur SO". Elle fournit cinq fonctionnalités : (1) Sélection du répertoire contenant les fichiers OWL-S, (2) Prétraitement permettant de charger les informations nécessaires dans la RAM (sous forme d'un arbre) ; (3) Lancement du processus d'initialisation en exécutant l'algorithme 3 et en introduisant un rayon $R1$; (4) Lancement du processus du regroupement ; et (5) Calcul les paramètres d'inertie Interclasse et Intraclasse. Les résultats sont affichés dans une deuxième fenêtre (Voir la Figure 8.2). Pour la génération des concepts de SO issus des centres de regroupement, nous avons utilisé l'outil Protégé3.4.4 pour créer les concepts (SA) de façon purement manuelle. Le listing présenté dans la Figure. 8.3 représente un extrait du (SO) généré à partir du corpus OWLS-TC 4.0.

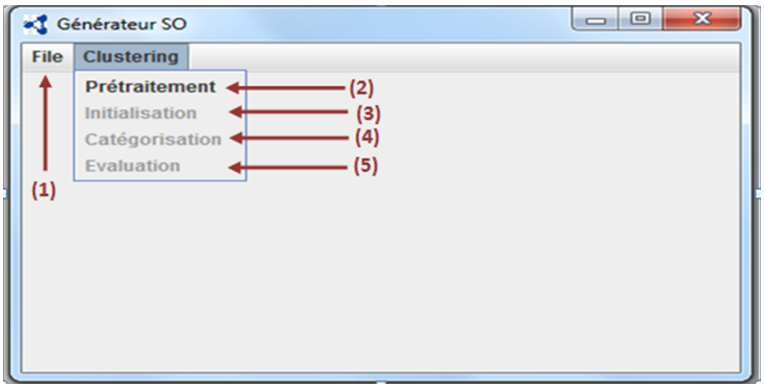

Figure 8.1 – L'interface principale de l'outil Générateur SO

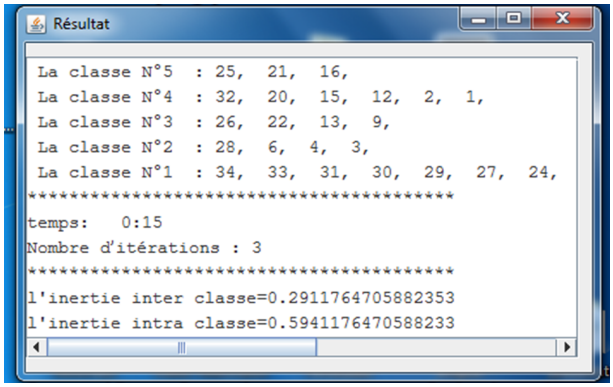

FIGURE 8.2 – Fenêtre des Résultats

```
  <rdfs:domain rdf:resource="#Domaine"/>
  <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:ID="Name">
  <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
  <rdfs:domain rdf:resource="#Domaine"/>
</owl:DatatypeProperty>
<SA rdf:ID="Geography_5"/>
<Geography rdf:ID="searchFormattedAddress">
  <hasInput xml:lang="en">LICENSEKEY</hasInput>
  <hasInput xml:lang="en">ADDRESS</hasInput>
  <Key xml:lang="en">0013</Key>
  <hasOutput xml:lang="en">LOCATION</hasOutput>
  <hasOutput xml:lang="en">LATITUDE</hasOutput>
  <hasOutput xml:lang="en">LONGITUDE</hasOutput>
  <Name xml:lang="en">Search Formatted Address service</Name>
  <textDescription rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  ></textDescription>
</Geography>
<Geography rdf:ID="getPopulationDensityOfLocation">
  <hasOutput xml:lang="en">DISTANCE</hasOutput>
  <hasInput xml:lang="en">USER ID</hasInput>
  <hasInput xml:lang="en">LATITUDE</hasInput>
  <hasInput xml:lang="en">LONGITUDE</hasInput>
  <Key xml:lang="en">00003</Key>
  <textDescription xml:lang="en">This service computes the distance in miles between two given coordinates.
  <Name xml:lang="en">Distance calculator between two locations</Name>
</Geography>
<Geography rdf:ID="addressDistanceCalculator">
  <hasOutput xml:lang="en">DISTANCE</hasOutput>
```

FIGURE 8.3 – Extrait de l'ontologie de service (OWL)

8.2.2 Expérimentations

Pour évaluer l'efficacité de l'algorithme de clustering sur le corpus choisi, nous avons testé l'outil prototype "Générateur SO". Les résultats du test sont présentés dans le tableau 8.1.

Le tableau 8.1 indique, dans sa première partie (partie gauche), les résultats obtenus en appliquant l'Algorithme 3 sur différentes valeurs du rayon de similarité (0,25, 0,50, 0,75, 1,00). Donc, la colonne SN du tableau indique le nombre de services

Tableau 8.1 – Résultats des expérimentations de clustering.

Domaine	*SN*	**R**	**D**	*NC*	*Itr*	*Temps*
Food	34	0.25	28-2	2	3	00 :10
		0.50	28-2	2	3	00 :10
		0.75	14-2	3	4	00 :13
		1.00	6-2	6	5	00 :25
Geography	39	0.25	33-2	2	4	02 :18
		0.50	16-2	8	6	07 :32
		0.75	7-2	12	4	05 :16
		1.00	4-2	18	6	10 :54
Weapon	40	0.25	40	1	-	-
		0.50	32	1	-	-
		0.75	22-3	4	3	00 :18
		1.00	8-2	8	4	00 :22
Communication	58	0.25	58	1	-	-
		0.50	58	1	-	-
		0.75	32-2	4	3	00 :58
		1.00	25-3	5	3	01 :19
Travel	165	0.25	94-31	2	3	01 :46
		0.50	57-11	7	7	06 :59
		0.75	18-3	15	12	11 :29
		1.00	7-2	32	8	09 :08

dans chaque domaine. La colonne R indique les différentes valeurs du rayon de similarité fixées pour le test. La colonne D indique la densité (max-min) calculée pour chaque classe retenue : la valeur max représente la densité de la première classe retenue, et la valeur min indique la densité de la dernière classe retenue. Enfin, la colonne NC indique le nombre de clusters retenus pour chaque test. Dans la deuxième partie (partie droite), le tableau indique les résultats obtenus en appliquant l'algorithme k-means 2 en tenant compte de l'initialisation trouvée par l'Algorithme 3, et exprimée dans la première partie. Les colonnes Itr et $Temps$ indiquent le nombre d'itérations et le temps d'exécution (minutes : secondes) consommé pendant que k-means converge vers la solution. Nous avons remarqué que le choix du rayon de similarité a une influence directe sur le nombre de clusters. Par conséquent, la densité diminue inversement avec l'augmentation de la similarité du rayon, et par conséquent le nombre de clusters augmentera. Nous avons également constaté que le domaine géographique est plus gourmand en temps ; ceci est justifié par la qualité des données des services géographiques qui ont plusieurs concepts en entrée et en sortie, où le nombre de concepts peut atteindre quatorze (14) concepts (exemple du service objectsMappingService.owls), inversement aux autres domaines qui ne dépassent pas cinq (5) concepts. Ainsi, nous constatons que le pourcentage de la densité dans les domaines est élevé, notamment dans les domaines d'alimentation, d'armes et de communication. Ceci est évident, car les services sont issus du même domaine (déjà regroupés en domaine ontologique). Notre objectif était de créer une deuxième couche de catégorisation au sein de chaque domaine.

8.2.3 Évaluation et discussion des résultats

Le défi majeur des problèmes de clustering est de trouver une solution optimale où les données d'un même groupe soient au maximum similaires, et se différent, au maximum, aux données des autres groupes. Pour cela, la méthode classique calcule deux paramètres (inertie interclasse et intraclasse) pour juger la meilleure solution. Ainsi, la section d'évaluation montre l'efficacité de l'algorithme de clustering en calculant ces deux paramètres, inertie Interclasse et Intraclasse, pour chaque proposition issue de différentes valeurs du rayon de similarité. Les graphes ci-dessous (Voir Figure 8.4) montrent les valeurs d'inertie Interclasse et Intraclasse obtenues avec différentes valeurs du rayon de similarité (0.25, 0.50, 0.75 et 1.00). Nous constatons que tous les domaines ont donné les meilleurs résultats avec un rayon de similarité égal à 1.00, à l'exception du domaine Communication qui a donné les meilleures valeurs d'inertie Interclasse et Intraclasse avec un rayon égal à 0.75.

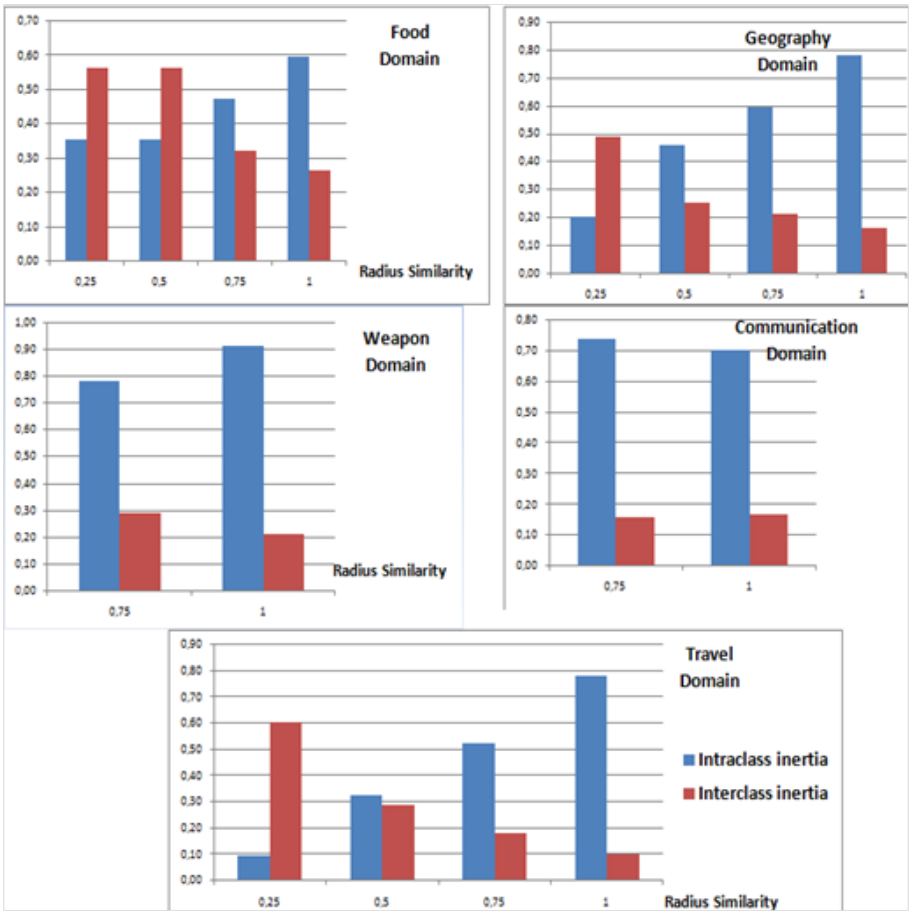

Figure 8.4 – Graphes représentant l'inertie interclasse et intraclasse de différents domaines avec différentes valeurs du rayon de similarité.

8.3 L'algorithme de Matching

8.3.1 L'outil implémenté

L'algorithme de Matching proposé (Algorithme 1) est un algorithme qui exécute la comparaison entre les annonces des services exprimées dans les registres UDDIs et les demandes envoyées par les clients. Pour sa mise en œuvre, nous avons implémenté en utilisant le même environnement éclipse Indigo un outil prototype appelé "Matching SO". La Figure 8.5 représente l'interface principale de l'outil. Elle permet d'exécuter les fonctionnalités :

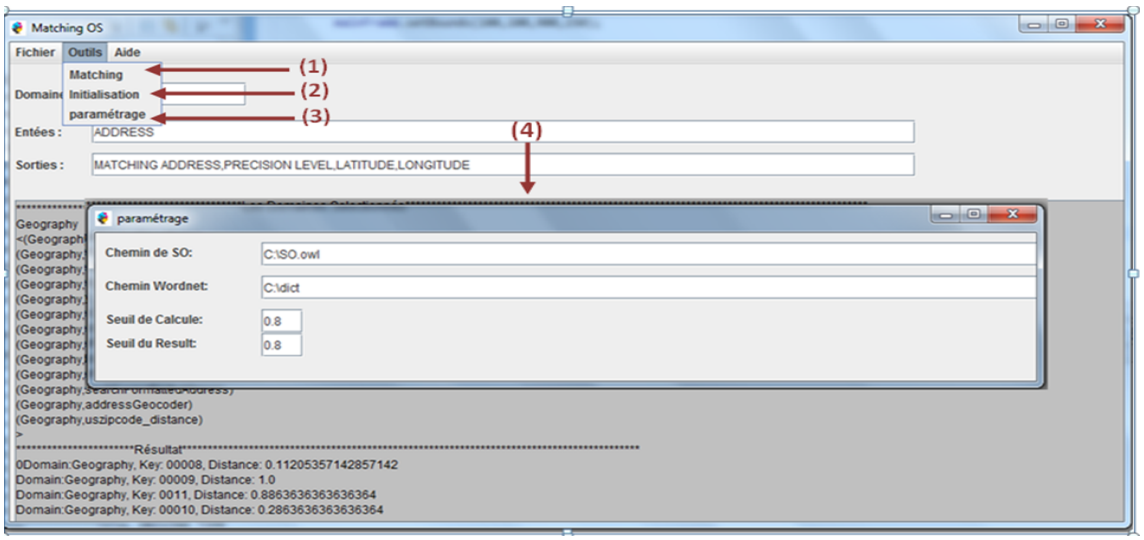

Figure 8.5 – L'interface principale de l'outil Matching SO

(1) Matching : exécuter l'opération de Matching, (2) Initialisation : permet de vider tous les champs, (3) Paramétrage : pour visualiser la fenêtre de paramétrage, et (4) Paramétrage : représente notre fenêtre de paramètres, où il faut renseigner le chemin de l'ontologie, le chemin du dictionnaire Wornet, et les deux seuils pour les fonctions Compare1() et Compare2().

8.3.2 Évaluation

La section d'évaluation de l'algorithme 1 vise à choisir la similarité la plus efficace à utiliser dans l'algorithme de correspondance (Compare1() et Compare2()). Pour cela, nous avons étudié cinq mesures de similarité WordNet : PATH, PIRRO_ SECO, RESNIKMETRIC, JIANGMETRIC et LIN (Giuseppe et Jérôme, 2010). Pour réaliser ce test, nous avons considéré le même corpus de services Web sémantiques vu dans la section 7.4.1, page 81. Notre intention initiale était de tester l'algorithme sur

l'ensemble du corpus. Puisqu'il n'y a pas de pré-classification dans le corpus, nous avons limité notre expérimentation à un seul sous-ensemble (domaine géographique). Ensuite, nous avons classé manuellement un ensemble de services Web sémantiques OWL-S (voir Tableaux 8.2 et 8.3).

Tableau 8.2: Classification manuelle

Service Web sémantique	***Domaine***	***Classe dans SO <Domaine ; clés>***
calculateDistanceInMiles.owls	Géographique	<Géographique ; 1>
getDistanceBetweenPlaces.owls	Géographique	<Géographique ; 1>
getLocationOfAddress.owls	Géographique	< Géographique ; 11 ; 12 ; 6>
getLocationOfAddressYahooMaps.owls	Géographique	< Géographique ; 11, 9>
getPlaceOfAddress.owls	Géographique	< Géographique ; 12>
getDistanceBetweenLocations.owls	Géographique	< Géographique ; 1>
renderMapService.owls	Géographique	< Géographique ; 8>

Tableau 8.3: Signification des classes

Domaine	***Numéro de classe « Clé »***	***Nom du Service Abstrait***
Géographique	<1>	addressDistanceCalculator
Géographique	<6>	getCoordinatesOfAddress
Géographique	<8>	getDrivingDirections
Géographique	<9>	getLocationOfAddress
Géographique	<11>	getMapOfAddres
Géographique	<12>	getPlaceOfAddress

Les graphes ci-dessous (voir la Figure 8.6) montrent les valeurs de précision, de rappel et de F-mesure obtenues en exécutant notre algorithme de mise en correspondance sur l'ensemble des services Web précédemment classés (voir Tableau 2), pour différentes valeurs du seuil. Nous attendions à ce que la précision de la mesure PATH diminue proportionnellement avec le seuil. Cependant, ce n'était pas le cas dans le graphique. Cela peut s'expliquer par le fait que les deux services Web sémantiques getDistanceBetweenPlaces.owls et enderMapService.owls ont été reconnus avec un seuil inférieur à 0,5. Par conséquent, la mesure PATH exprime les mauvais résultats de notre algorithme, puisqu'elle donne des valeurs minimales de précision, de rappel et de F-mesure. Pour la mesure RESNIK_ METRIC, les bons résultats sont apparus avec un seuil compris entre 0,8 et 0.7. Les autres mesures de similarité ont exprimé des valeurs de rappel presque identiques. C'est donc la valeur de précision qui fera la différence entre ces méthodes. Finalement, nous pouvons dire que pour avoir les meilleurs résultats, nous choisissons l'une des deux mesures de similarité PIRRO_

SECO_ METRIC ou Lin, avec un seuil de 0.8. Nous pouvons également utiliser la mesure RESNIK_ METRIC avec un seuil de 0,7.

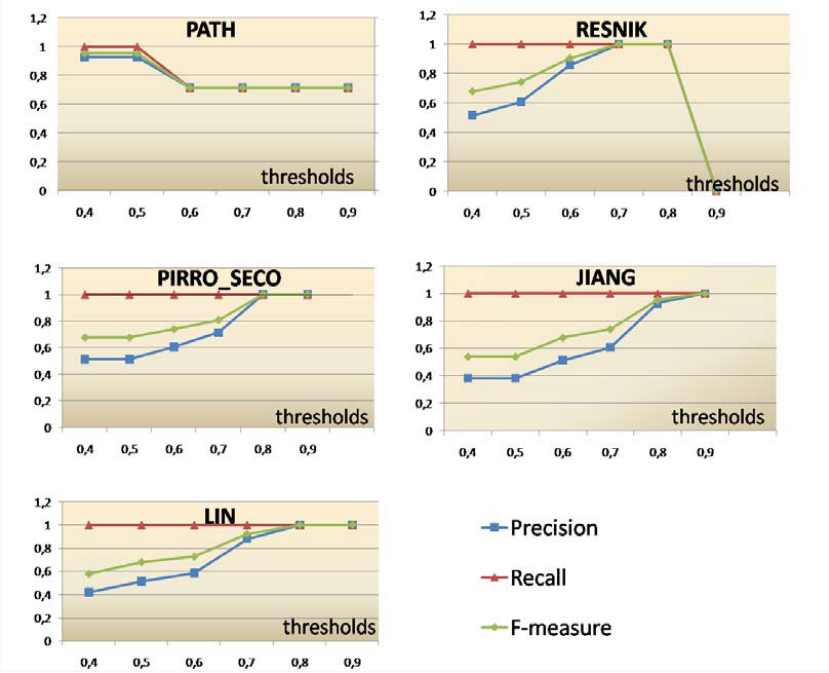

Figure 8.6 – Courbes de précision, de rappel et de F-mesure de l'algorithme de matching avec différentes mesures de similarité en utilisant différentes valeurs du seuil

8.4 Etude comparative

Dans cette section, nous comparons notre travail avec d'autres travaux traitant le même problème. Pour cela, nous avons choisi quatre critères (Voir le tableau 8.4) :

- **Méthode :** indique la méthode de classification utilisée.
- **Source :** décrit l'entrée du système.
- **Matching :** indique les éléments mis en correspondance.
- **Résultat de la classification :** donne des informations sur la sortie du système.

La section ci-après discute en détail le contenu du tableau 8.4.

Tableau 8.4: Comparaison des approches d'enregistrement et de découverte des services Web

	Notre approche	***(Canturk and Senkul, 2011b, 2012; Sellami et al., 2009)***	***(Sivashanmugam et al., 2004)***	***(Bianchini et al., 2008)***
Méthode	Utiliser l'ontologie des services pour classer les services Web.	Classer les services Web à l'aide d'une ontologie de domaine.	Créer une ontologie de service complexe (XTRO).	Créer une ontologie en utilisant le mappage entre les différentes ontologies des UDDI.
Source	SWS	SWS	SWS	SWS
Matching	La correspondance est effectuée entre la demande et l'ontologie de service (SO).	La correspondance est effectuée entre la requête et les services dans tous les registres concernés (ayant le même domaine avec la requête).	La correspondance est effectuée entre la requête et les services dans tous les registres sélectionnés par XTRO.	La correspondance est effectuée entre la requête et les services Web dans un seul registre ; le résultat est utilisé pour extraire d'autres résultats dans les autres registres en utilisant le mappage ontologique.
Résultat	Regroupement plus fine avec une classe spécifique (sous-classe de domaine spécifique) en utilisant SA.	Classes globales (plus générales).	Classement complexe.	Pas de regroupement, mais il existe des liens calculés (intra et inter registres).

8.4.1 Discussion

Les ontologies ont marqué leur présence dans le domaine de l'intelligence artificielle, pour résoudre les problèmes de modélisation des connaissances, de classification (Bloehdorn et al., 2006), de recherche sémantique (Sendhilkumar et al., 2009), et plus spécifiquement d'ingénierie des connaissances. La découverte de services Web distribués a fait l'objet de plusieurs études. Les travaux explorés dans ce contexte

peuvent être classés en deux catégories principales :

- Ceux qui ont regroupé les services en utilisant des ontologies de domaine (Boukhadra et al., 2013, 2014; Canturk and Senkul, 2011b, 2012; Sellami et al., 2009).
- Ceux qui ont créé des ontologies de services pour guider la découverte (Sivashanmugam et al., 2004; Bianchini et al., 2008).

Dans la première catégorie, la classification est basée sur le regroupement des services Web. Cette catégorisation est établie en utilisant les ontologies de domaine où les services appartenant à la même ontologie du domaine seront regrouper dans une même classe (Canturk and Senkul, 2011b, 2012; Sellami et al., 2009).

Dans la deuxième catégorie, une ontologie spécialement dédiée pour les services sera crée pour assister la découverte. Dans (Sivashanmugam et al., 2004), l'ontologie créée, nommée XTRO, est une ontologie complexe dans la mesure où elle contient des informations sur les domaines, les registres, les ontologies de domaine et la fédération. L'ontologie créée dans (Bianchini et al., 2008) est une ontologie développée par la mise en correspondance entre les différentes ontologies des différentes UDDI. L'hybridation entre les deux techniques pour augmenter les performances est employée dans plusieurs travaux (Dass et al., 2015).

Dans notre approche, nous avons hybridé les deux techniques :

- Nous avons utilisé une ontologie de services simple (non complexe) en ce référant à Sivashanmugam et al. (2004), le fait que nous séparons entre la partie stockage et la partie connaissance.
- Nous avons étendu la classification utilisée dans (Canturk and Senkul, 2011b, 2012) et dans (Sellami et al., 2009) en ajoutant la notion du SA (Service Abstrait) dans la couche connaissance.

Notre approche garantit l'évolutivité, dans la mesure où elle est basée sur un système en couche. Les classes dérivées de l'ontologie SO sont plus spécifiques et plus fines que les autres (Canturk and Senkul, 2011b, 2012; Sellami et al., 2009). Ces classes ne comprennent pas tous les services du même domaine, mais un sous-ensemble, ceux qui ont la même partie abstraite. Par conséquent, la distance intra-classe est minimisée par rapport aux autres, ce qui garantit des résultats plus pertinents.

8.5 Conclusion

L'implémentation et l'évaluation sont utiles et essentielles pour montrer les fonctionnalités et l'efficacité des programmes proposés. Dans le chapitre 7, nous avons décrit l'architecture fonctionnelle du notre approche, nous avons aussi présenté, dans le même chapitre, les étapes et les démarches suivis, aussi les algorithmes proposés.

Dans ce chapitre, nous avons discuté l'implémentation de notre approche. Effectivement, nous avons présenté les outils utilisés et l'interface conçue. Nous avons aussi exposé l'évaluation des deux implémentations : construction de l'ontologie et l'algorithme de matching. Les résultats des deux algorithmes 2 et 3 (Voir tableau 8.1) montre que l'algorithme 3 a contribué pour l'exécution rapide (en terme de nombre d'itération) et efficace de l'algorithme 2. Ainsi, les résultats obtenus figurant dans les courbes 8.4 et 8.6 prouvent l'efficacité du système implémenté.

Chapitre 9

Conclusion Générale et Perspectives

9.1 Conclusion générale

Avec le développement rapide des technologies de l'information, l'interopérabilité des systèmes distribués est de nos jours une problématique centrale. Ce domaine de recherche est favorisé par l'adoption de l'architecture orientée services comme modèle de développement, et particulièrement, par les services Web qui combinent les avantages de ce modèle avec ceux des langages et des technologies développés pour Internet. Les services Web sont admis pour l'intégration et l'interopérabilité des systèmes répartis. Ils sont caractérisés par leur indépendance des plateformes et des systèmes d'exploitation, ce qui a provoqué leur popularité et leur large utilisation par les différentes organisations commerciales et industrielles publiant leurs services à travers le Web, ce qui a causé l'augmentation rapide des services échangés. Au début, les travaux traitants les services Web se mettaient face au problème d'automatisation de la découverte. En effet, l'utilisation des technologies du Web sémantique afin de rendre les services interprétables par la machine (McIlraith et al., 2001; Shadbolt et al., 2006; Bichier and Lin, 2006; Krummenacher et al., 2005) est considérée comme un facteur de réussite majeur en vue de faciliter la recherche dans le domaine de services Web. Actuellement, et en conséquence de la demande croissante de l'utilisation de la technologie des services Web, les travaux se concentrent plus sur les solutions distribuées pour la découverte des services Web. Dans ce dernier contexte et à travers cette thèse, nous avons proposé une architecture de stockage/découverte conçue pour être distribuée, évolutive, et qui adresse le problème de centralisation d'un annuaire UDDI. Notre travail est basé sur la classification des services Web selon une ontologie de services. Notre ontologie est utilisée non pas pour décrire les interfaces des services Web, comme le fait OWL-S (Martin et al., 2004) et WSMO (Roman et al., 2005), mais plutôt un modèle partagé pour répartir les services Web entre les registres.

Pour la mise en œuvre de notre approche, nous avons proposé la génération semi-automatique d'une ontologie de services représentée en OWL. Nous avons aussi implémenté un algorithme de matching. Ce dernier est exécuté non pas sur tous les services Web stockés mais plutôt sur l'ontologie créée, ce qui réduit l'espace de recherche, et par conséquent diminue le temps de découverte qui est un facteur essentiel dans un tel système distribué. Les résultats encourageants (Voir chapitre 8) ont prouvé l'efficacité des algorithmes de matching et de regroupement proposés.

9.2 Perspectives

La réalisation de notre approche nous a permis de dégager deux perspectives pour un futur travail :

- La génération de l'ontologies SO peut être réalisée et comparée avec l'implémentation d'autres algorithmes, tels que l'hiérarchique ascendant ou descendant.

- Le système doit être étendu pour qu'il supporte d'autres fonctionnalités, telles que la composition. La composition des services peut être réalisée en exploitant l'ontologie SO prenant en considération les services Web composites.

Bibliographie

Akkiraju, R., Farrell, J., Miller, J., Nagarajan, M., Schmidt, M.-T., Sheth, A., and Verma, K. (2005). Web service semantics (wsdl-s). w3c member submission, november 2005. *Retrieved March*, 12 :2009.

Ankolekar, A., Burstein, M., Hobbs, J. R., Lassila, O., Martin, D., McDermott, D., McIlraith, S. A., Narayanan, S., Paolucci, M., Payne, T., et al. (2002). Daml-s : Web service description for the semantic web. In *International semantic web conference*, pages 348–363. Springer.

Astrahan, M. (1970). Speech analysis by clustering, or the hyperphoneme method. Technical report, STANFORD UNIV CA DEPT OF COMPUTER SCIENCE.

Beech, D., Lawrence, S., Maloney, M., Mendelsohn, N., and Thompson, H. S. (2001). Xml schema part 1 : Structures. *World Wide Web Consortium, Boston, USA*, page W3C.

Belfedhal, A. E. and Malki, M. (2011). *LA SECURITE DES SYSTEMES D'INFORMATION A BASE D'ONTOLOGIES*. PhD thesis, Université de Sidi Bel Abbés.

Benatallah, B., Hacid, M.-S., Leger, A., Rey, C., and Toumani, F. (2005). On automating web services discovery. *The VLDB Journal*, 14(1) :84–96.

Benghida, A. and Boufaida, M. (2014). Web services discovery in p2p networks based on clustering. In *2014 1st International Conference on Information and Communication Technologies for Disaster Management (ICT-DM)*, pages 1–7. IEEE.

Benjamins, V. R., Davies, J., Baeza-Yates, R., Mika, P., Zaragoza, H., Greaves, M., Gomez-Perez, J. M., Contreras, J., Domingue, J., and Fensel, D. (2008). Near-term prospects for semantic technologies. *IEEE Intelligent Systems*, 23(1) :76–88.

Berbner, R., Spahn, M., Repp, N., Heckmann, O., and Steinmetz, R. (2006). Heuristics for qos-aware web service composition. In *2006 IEEE International Conference on Web Services (ICWS'06)*, pages 72–82. IEEE.

Berners-Lee, T. (2006). Artificial intelligence and the semantic web : Aaai2006 keynote. *W3C Web site*.

Berners-Lee, T. and Hendler, J. (2001). Publishing on the semantic web. *Nature*, 410(6832) :1023–1024.

Berners-Lee, T., Hendler, J., and Lassila, O. (2001). The semantic web. *Scientific american*, 284(5) :34–43.

Bianchini, D., De Antonellis, V., Melchiori, M., and Salvi, D. (2008). A semantic overlay for service discovery across web information systems. In *International Conference on Web Information Systems Engineering*, pages 292–306. Springer.

Bichier, M. and Lin, K.-J. (2006). Service-oriented computing. *Computer*, 39(3) :99–101.

Blei, D. M., Ng, A. Y., and Jordan, M. I. (2003). Latent dirichlet allocation. *the Journal of machine Learning research*, 3 :993–1022.

Booth, D., Champion, M., Ferris, C., McCabe, F., Newcomer, E., and Orchard, D. (2003). Web services architecture-w3c working draft 14 may 2003.

Booth, D., Haas, H., McCabe, F., Newcomer, E., Champion, M., Ferris, C., and Orchard, D. (2004). Web services architecture, w3c working group note. *vol*, 2005 :W3C.

Boukhadra, A., Benatchba, K., and Balla, A. (2013). Da5dcsws : a distributed architecture for semantic web services discovery and composition. In *8th International Conference for Internet Technology and Secured Transactions (ICITST-2013)*, pages 182–187. IEEE.

Boukhadra, A., Benatchba, K., and Balla, A. (2014). Ranked matching of owl-s process model for distributed discovery of sws in p2p systems. In *2014 17th International Conference on Network-Based Information Systems*, pages 106–113. IEEE.

Box, D., Ehnebuske, D., Kakivaya, G., Layman, A., Mendelsohn, N., Nielsen, H. F., Thatte, S., and Winer, D. (2000). Simple object access protocol (soap) 1.1. https://www.w3.org/TR/2000/NOTE-SOAP-20000508/, (accessed February 9, 2020).

Bravo, M., Mora-Gutiérrez, R. A., and Hoyos-Reyes, L. F. (2019). Bio-inspired hybrid algorithm for web services clustering. In *Advanced Analytics and Artificial Intelligence Applications*. IntechOpen.

Burkard, R., Dell'Amico, M., and Martello, S. (2009). Assignment problems society for industrial and applied mathematics. *Philadelphia, PA, USA*.

Canturk, D. and Senkul, P. (2011a). Semantic annotation of web services with lexicon-based alignment. In *2011 IEEE World Congress on Services*, pages 355–362. IEEE.

Canturk, D. and Senkul, P. (2011b). Using semantic information for distributed web service discovery. *International Journal of Web Science*, 1(1-2) :21–35.

Canturk, D. and Senkul, P. (2012). Ontology-based routing of web services in distributed service discovery system containing domain specific nodes. In *2012 IEEE Symposium on Computers and Communications (ISCC)*, pages 000283–000288. IEEE.

Cao, Y., Liu, J., Cao, B., Shi, M., Wen, Y., and Peng, Z. (2019). Web services classification with topical attention based bi-lstm. In *International Conference on Collaborative Computing : Networking, Applications and Worksharing*, pages 394–407. Springer.

Cardoso, J. (2007a). *Semantic web services : Theory, tools and applications*. information science reference Chocolate Avenue Hershey.

Cardoso, J. (2007b). *Semantic web services : Theory, tools and applications*. information science reference Chocolate Avenue Hershey.

Celebi, M. E., Kingravi, H. A., and Vela, P. A. (2013). A comparative study of efficient initialization methods for the k-means clustering algorithm. *Expert systems with applications*, 40(1) :200–210.

Cerami, E. (2002). *Web services essentials : distributed applications with XML-RPC, SOAP, UDDI & WSDL*. " O'Reilly Media, Inc.".

Channa, N., Li, S., Shi, W., and Peng, G. (2005). A can-based p2p infrastructure for semantic web services publishing and discovery. In *2005 1st IEEE and IFIP International Conference in Central Asia on Internet*, pages 5–pp. IEEE.

Chauvet, J.-M. (2002). Services web avec soap, wsdl, uddi, ebxml... Technical report, Eyrolles.

Chinnici, R., Moreau, J.-J., Ryman, A., and Weerawarana, S. (2007). Web services description language (wsdl) version 2.0 part 1 : Core language. *W3C recommendation*, 26(1) :19.

Christensen, E., Curbera, F., Meredith, G., Weerawarana, S., et al. (2001). Web services description language (wsdl) 1.1.

Della Valle, E., Cerizza, D., Celino, I., Turati, A., Lausen, H., Steinmetz, N., Erdmann, M., and Funk, A. (2008). Realizing service-finder : Web service discovery at web scale. In *European Semantic Technology Conference (ESTC), Vienna*.

Du, Z., Huai, J., and Liu, Y. (2005). Ad-uddi : An active and distributed service registry. In *International workshop on technologies for e-services*, pages 58–71. Springer.

Eckert, J. (2009). *Cross-organizational Service-based Workflows-Solution Strategies for Quality of Service Optimization*. PhD thesis, Technische Universität.

El Bouhissi, H. and Malki, M. (2014). Semantic web service ontology construction : A reverse engineering approach. In *2014 IEEE/ACS 11th International Conference on Computer Systems and Applications (AICCSA)*, pages 161–168. IEEE.

Endrei, M., Ang, J., Arsanjani, A., Chua, S., Comte, P., Krogdahl, P., Luo, M., and Newling, T. (2004). *Patterns : service-oriented architecture and web services*. IBM Corporation, International Technical Support Organization New York, NY.

Erl, T. (2005). *Service-Oriented Architecture : Concepts, Technology, and Design*. Prentice Hall InternationalISBN : 978-0131858589.

Farrell, J. and Lausen, H. (2007). Semantic annotations for wsdl and xml schema. *W3C recommendation*, 28.

Fensel, D. and Bussler, C. (2002). The web service modeling framework wsmf. *Electronic Commerce Research and Applications*, 1(2) :113–137.

Folino, G., Pisani, F., and Trunfio, P. (2014). Efficient discovery of data mining services over dht-based overlays. In *2014 International Conference on High Performance Computing & Simulation (HPCS)*, pages 484–490. IEEE.

Funk, A. and Bontcheva, K. (2010). Ontology-based categorization of web services with machine learning. In *LREC*.

Ghallab, M., Howe, A., Knoblock, C., McDermott, D., Ram, A., Veloso, M., Weld, D., and Wilkins, D. (1998). Pddl-the planning domain definition language.

Glass, G. (2000). The web services (r) evolution applying web services to applications. *IBM Corp.*

Gomadam, K., Verma, K., Sheth, A., and Li, K. (2006). Keywords, port types and semantics : a journey in the land of web service discovery. In *Semantic Web Services, Processes and Applications*, pages 89–105. Springer.

Gottschalk, K., Graham, S., Kreger, H., and Snell, J. (2002a). Introduction to web services architecture. *IBM systems Journal*, 41(2) :170–177.

Gottschalk, K., Graham, S., Kreger, H., and Snell, J. (2002b). Introduction to web services architecture. *IBM systems Journal*, 41(2) :170–177.

Graham, K. and Brian, M. (2004). Resource Description Framework (RDF) :Concepts and Abstract Syntax. https://www.w3.org/TR/rdf-concepts/.

Gruber, T. R. (1993). A translation approach to portable ontology specifications. *Knowledge acquisition*, 5(2) :199–220.

Hack, S. and Lindemann, M. (2007). Enterprise soa einführen.

He, H. (2003). What is service-oriented architecture ? september 2003. https://www.xml.com/pub/a/ws/2003/09/30/soa.html , (accessed February 9, 2020).

Jaccard, P. (1902). Distribution comparée de la flore alpine dans quelques régions des alpes occidentales et orientales. *Bulletin de la Murithienne*, (31) :81–92.

Jardim-Goncalves, R., Grilo, A., and Steiger-Garcao, A. (2006). Challenging the interoperability between computers in industry with mda and soa. *Computers in industry*, 57(8-9) :679–689.

Jung, Y., Ryu, J., Kim, K.-m., and Myaeng, S.-H. (2010). Automatic construction of a large-scale situation ontology by mining how-to instructions from the web. *Web Semantics : Science, Services and Agents on the World Wide Web*, 8(2-3) :110–124.

Kaouan, M., Bouchiha, D., and Benslimane, S. M. (2015). Shared-repository based approach for storing and discovering web services. *Procedia Computer Science*, 73 :56–65.

Kaouan, M., Bouchiha, D., Benslimane, S. M., and Boukli, Hacene, S. (2017). Ontology-based web services classification for registration and discovery of web services. *International Journal of Artificial Intelligence and Soft Computing*, 6(2) :129–147.

Kaouan, M., Bouchiha, D., Benslimane, S. M., and Boukli, Hacene, S. (2020). Towards service ontology for web services storage and discovery. In *2020 4th International Symposium on Informatics and its Applications (ISIA)*, pages 1–6. IEEE.

Karypis, G., Han, E.-H., and Kumar, V. (1999). Chameleon : Hierarchical clustering using dynamic modeling. *Computer*, 32(8) :68–75.

Kashyap, V., Christoph, B., and Matthew, M. (2008). The semantic web-semantics for data and services on the web. *Berlin and Heidelberg*.

Kashyap, V. and Sheth, A. (1994). Semantics-based information brokering. In *Proceedings of the third international conference on Information and knowledge management*, pages 363–370.

Keller, U., Lara, R., Polleres, A., Toma, I., Kifer, M., and Fensel, D. (2004). Wsmo web service discovery. *WSML Working Draft D*, 5.

Khouja, M. and Juiz, C. (2015). Enhanced service discovery via shared context in a distributed architecture. In *2015 IEEE International Conference on Web Services*, pages 273–280. IEEE.

Krummenacher, R., Hepp, M., Polleres, A., Bussler, C., and Fensel, D. (2005). Www or what is wrong with web services. In *Third European Conference on Web Services (ECOWS'05)*, pages 235–243. IEEE.

Lau, T., Wang, H.-C., and Chuang, C.-C. (2011). A definition of service as base for developing service science. In *2011 International Joint Conference on Service Sciences*, pages 49–53. IEEE.

Lawrence, K., Kaler, C., Nadalin, A., Monzillo, R., and Hallam-Baker, P. (2006). Web services security : Soap message security 1.1 (ws-security 2004). *OASIS, OASIS Standard, Feb*.

Lévy, P. (2003). Le futur web exprimera l'intelligence collective de l'humanité. *Le Journal du Net*.

Liang, Q. and Chung, J.-Y. (2008). A federated uddi system for concurrent access to service data. In *2008 IEEE International Conference on e-Business Engineering*, pages 71–78. IEEE.

Liang, Q., Li, P., Hung, P. C., and Wu, X. (2009). Clustering web services for automatic categorization. In *2009 IEEE International Conference on Services Computing*, pages 380–387. IEEE.

Liang, Q. A., Lam, H., Narupiyakul, L., and Hung, P. C. (2008). A rule-based approach for availability of web service. In *2008 IEEE International Conference on Web Services*, pages 153–160. IEEE.

Lin, Q., Rao, R., and Li, M. (2006). Dwsdm : a web services discovery mechanism based on a distributed hash table. In *2006 Fifth International Conference on Grid and Cooperative Computing Workshops*, pages 176–180. IEEE.

Liu, W. and Wong, W. (2009). Web service clustering using text mining techniques. *International Journal of Agent-Oriented Software Engineering*, 3(1) :6–26.

Lv, W. and Yu, J. (2007). pservice : Peer-to-peer based web services discovery and matching. In *2007 Second International Conference on Systems and Networks Communications (ICSNC 2007)*, pages 54–54. IEEE.

MacKenzie, C. M., Laskey, K., McCabe, F., Brown, P. F., Metz, R., and Hamilton, B. A. (2006). Reference model for service oriented architecture 1.0. *OASIS standard http ://docs.oasis-open.org/soa-rm/v1.0/soa-rm.html*, 12(S 18).

MacQueen, J. et al. (1967). Some methods for classification and analysis of multivariate observations. In *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, volume 1, pages 281–297. Oakland, CA, USA.

Mandreoli, F., Perdichizzi, A. M., and Penzo, W. (2007). A p2p-based architecture for semanticweb service automatic composition. In *18th International Workshop on Database and Expert Systems Applications (DEXA 2007)*, pages 429–433. IEEE.

Martin, D., Burstein, M., Hobbs, J., Lassila, O., McDermott, D., McIlraith, S., Narayanan, S., Paolucci, M., Parsia, B., Payne, T., et al. (2004). Owl-s : Semantic markup for web services. *W3C member submission*, 22(4).

Martin, D., Burstein, M., Mcdermott, D., Mcilraith, S., Paolucci, M., Sycara, K., Mcguinness, D. L., Sirin, E., and Srinivasan, N. (2007a). Bringing semantics to web services with owl-s. *World Wide Web*, 10(3) :243–277.

Martin, D., Domingue, J., Brodie, M. L., and Leymann, F. (2007b). Semantic web services, part 1. *IEEE Intelligent Systems*, 22(5) :12–17.

Maximilien, E. M. and Singh, M. P. (2003). Agent-based architecture for autonomic web service selection. In *Workshop on Web Services and Agent-based Engineering at Autonomous Agents and Multi-Agent Systems*. Citeseer.

Mazari, A. C., Aliane, H., and Alimazighi, Z. (2012). Automatic construction of ontology from arabic texts. In *ICWIT*, pages 193–202. Citeseer.

McCarthy, J. (1980). Circumscription a form of non-monotonic reasoning. *Artificial intelligence*, 13(1-2) :27–39.

McIlraith, S. A., Son, T. C., and Zeng, H. (2001). Semantic web services. *IEEE intelligent systems*, 16(2) :46–53.

Miller, J., Verma, K., Rajasekaran, P., Sheth, A., Aggarwal, R., and Sivashanmugam, K. (2004). Wsdl-s : Adding semantics to wsdl-white paper. *LSDIS Lab, University of Georgia, Georgia, USA*.

MOF (2004). The Object Management Group : Meta-Object Facility, version 1.4, 2002.Available at. http://www.omg.org/technology/documents/formal/mof.htm.

OASIS (2004). OASIS UDDI Specification TC. https://www.oasis-open.org/committees/uddi-spec/faq.php#top.

OWLS-TC-V4. Owls- tc v4. http://projects.semwebcentral.org/frs/?group_id=89&release_id=380. Accessed : April 2017.

Paolucci, M., Kawamura, T., Payne, T. R., and Sycara, K. (2002a). Importing the semantic web in uddi. In *International Workshop on web services, e-business, and the semantic web*, pages 225–236. Springer.

Paolucci, M., Kawamura, T., Payne, T. R., and Sycara, K. (2002b). Semantic matching of web services capabilities. In *International semantic web conference*, pages 333–347. Springer.

Paolucci, M., Srinivasan, N., Sycara, K. P., and Nishimura, T. (2003). Towards a semantic choreography of web services : from wsdl to daml-s. In *ICWS*, pages 22–26.

Papazoglou, M. (2008). *Web services : principles and technology*. Pearson Education.

Papazoglou, M. P., Traverso, P., Dustdar, S., and Leymann, F. (2007). Service-oriented computing : State of the art and research challenges. *Computer*, 40(11) :38–45.

Papazoglou, M. P., Traverso, P., Dustdar, S., Leymann, F., and Krämer, B. (2006). Service-oriented computing roadmap. In *Dagstuhl Seminar Proc. 05462, Service-Orienrted Computing (SOC)*, pages 1–29.

Papazoglou, M. P. and Van Den Heuvel, W.-J. (2007). Service oriented architectures : approaches, technologies and research issues. *The VLDB journal*, 16(3) :389–415.

Patil, A. A., Oundhakar, S. A., Sheth, A. P., and Verma, K. (2004). Meteor-s web service annotation framework. In *Proceedings of the 13th international conference on World Wide Web*, pages 553–562.

Pedersen, T., Patwardhan, S., Michelizzi, J., et al. (2004). Wordnet : : Similarity-measuring the relatedness of concepts. In *AAAI*, volume 4, pages 25–29.

Rajendran, T. and Balasubramanie, P. (2010). An optimal agent-based architecture for dynamic web service discovery with qos. In *2010 Second International conference on Computing, Communication and Networking Technologies*, pages 1–7. IEEE.

Rajmohan, R., Padmapriya, N., and Jayakumar, S. (2011). A survey on problems in distributed uddi. *International Journal of Computer Applications*, 36(3) :0975–8887.

Richi, N. and Lee, B. (2007). Web service discovery with additional semantics and clustering. In *IEEE/WIC/ACM International Conference on Web Intelligence (WI'07)*, pages 555–558. IEEE.

Roman, D., Keller, U., Lausen, H., De Bruijn, J., Lara, R., Stollberg, M., Polleres, A., Feier, C., Bussler, C., and Fensel, D. (2005). Web service modeling ontology. *Applied ontology*, 1(1) :77–106.

Rosen, M., Lublinsky, B., Smith, K. T., and Balcer, M. J. (2012). *Applied SOA : service-oriented architecture and design strategies*. John Wiley & Sons.

Roy, J. and Ramanujan, A. (2001). Understanding web services. *IT professional*, 3(6) :69–73.

Sánchez-Sánchez, C. and Sheremetov, L. B. (2018). N-gram representation for web service description classification. In *International Conference on Machine Learning, Optimization, and Data Science*, pages 447–459. Springer.

Sapkota, B., Roman, D., Kruk, S. R., and Fensel, D. (2006). Distributed web service discovery architecture. In *Advanced Int'l Conference on Telecommunications and Int'l Conference on Internet and Web Applications and Services (AICT-ICIW'06)*, pages 136–136. IEEE.

Schulte, R. W. and Natis, Y. V. (1996). Service oriented architectures part 1. *Gartner, SSA Research Note SPA-401-068*.

Schulte, S. (2010). *Web Service Discovery Based on Semantic Information-Query Formulation and Adaptive Matchmaking*. PhD thesis, Technische Universität.

Schumacher, M., Helin, H., and Schuldt, H. (2008). Cascom : Intelligent service coordination in the semantic web. chapter 3, pages 31–57. Springer Science & Business Media.

Sellami, M., Tata, S., Maamar, Z., and Defude, B. (2009). A recommender system for web services discovery in a distributed registry environment. In *2009 Fourth International Conference on Internet and Web Applications and Services*, pages 418–423. IEEE.

Sendhilkumar, S., Mahalakshmi, G., and Rajasekar, S. (2009). Ontology-based automatic query refinement. *International Journal of Artificial Intelligence and Soft Computing*, 1(2-4) :316–337.

Shadbolt, N., Hall, W., and Berners-Lee, T. (2006). The semantic web revisited. *intelligent Systems, IEEE*, 21(3) :96–101.

Sharpe, J. and Margolis, B. (2007). Soa for the business developer concepts.

Shen, Y. and Liu, F. (2019). An approach for semantic web discovery using unsupervised learning algorithms. In *Cyberspace Data and Intelligence, and Cyber-Living, Syndrome, and Health*, pages 56–72. Springer.

Singh, S. and Aswal, M. S. (2019). Ontology learning procedures based on web mining techniques. In *International Conference on Advances in Engineering Science Management & Technology (ICAESMT)-2019, Uttaranchal University, Dehradun, India*.

Sivashanmugam, K., Verma, K., and Sheth, A. (2004). Discovery of web services in a federated registry environment. In *Proceedings. IEEE International Conference on Web Services, 2004.*, pages 270–278. IEEE.

Stanescu, E., Stanescu, I., and Popa, V. (2005). Distributed infrastructure for semantic web services. *National Institute for Research and Development in Informatics*.

Steinbach, M., Karypis, G., and Kumar, V. (2000). A comparison of document clustering techniques.

Steinley, D. (2003). Local optima in k-means clustering : what you don't know may hurt you. *Psychological methods*, 8(3) :294.

Steinley, D. and Brusco, M. J. (2007). Initializing k-means batch clustering : A critical evaluation of several techniques. *Journal of Classification*, 24(1) :99–121.

Steinmetz, N., Kerrigan, M., Lausen, H., Tanler, M., and Sirbu, A. (2008). Simplifying the web service discovery process. In *SeMMA*, pages 31–45.

Stoica, I., Morris, R., Karger, D., Kaashoek, M. F., and Balakrishnan, H. (2001). Chord : A scalable peer-to-peer lookup service for internet applications. *ACM SIGCOMM Computer Communication Review*, 31(4) :149–160.

Studer, R., Grimm, S., and Abecker, A. (2007). Semantic web services. chapter 8, pages 211–244. Springer.

Su, T. and Dy, J. (2004). A deterministic method for initializing k-means clustering. In *16th IEEE International Conference on Tools with Artificial Intelligence*, pages 784–786. IEEE.

Sun, F.-z. and Hao, S.-g. (2010). A discovery framework for semantic web services in p2p environment. In *2010 International Conference on Electrical and Control Engineering*, pages 44–47. IEEE.

Sure, Y., Erdmann, M., Angele, J., Staab, S., Studer, R., and Wenke, D. (2002). Ontoedit : Collaborative ontology development for the semantic web. In *International Semantic Web Conference*, pages 221–235. Springer.

Thuraisingham, B. (2008). Secure semantic web services. In *Handbook of Database Security*, pages 231–245. Springer.

Tsetsos, V. (2007). Semantic web service discovery : Methods, algorithms, and tools. In *Semantic web services : theory, tools and applications*, chapter 6, pages 240–280. IGI Global.

UDDI (2007). UDDI. http://www.uddi.org/pubs/uddi_v3.htm, (accessed February 9, 2020).

UML/WSMO (2004). Appendix B. UML Class Diagrams for WSMO Elements. https://www.w3.org/Submission/WSMO/#appendixB.

Verma, K., Sivashanmugam, K., Sheth, A., Patil, A., Oundhakar, S., and Miller, J. (2005). Meteor-s wsdi : A scalable p2p infrastructure of registries for semantic publication and discovery of web services. *Information Technology and Management*, 6(1) :17–39.

Vinoski, S. (1993). Distributed object computing with corba. *C++ Report*, 5(6) :32–38.

Wang, B. (2021). Service clustering based on gsdmm topic model. In *International Conference on Applications and Techniques in Cyber Security and Intelligence*, pages 120–127. Springer.

Wang, X., Liu, C., and Yang, Z. (2009). An efficient semantic web service discovery algorithm in dht-based p2p network. In *2009 First International Conference on Future Information Networks*, pages 188–193. IEEE.

Wong, W., Liu, W., and Bennamoun, M. (2007). Tree-traversing ant algorithm for term clustering based on featureless similarities. *Data Mining and Knowledge Discovery*, 15(3) :349–381.

WSA (2007). WSA. https://www.w3.org/TR/ws-arch/, (accessed February 9, 2020).

WSDL (2007). WSDL. https://www.w3.org/TR/wsdl/, (accessed February 9, 2020).

XML (2003). XML. https://www.w3.org/TR/2003/PER-xml-20031030/, (accessed February 9, 2020).

Yu, Q., Liu, X., Bouguettaya, A., and Medjahed, B. (2008). Deploying and managing web services : issues, solutions, and directions. *The VLDB Journal*, 17(3) :537–572.

Yulin, N., Huayou, S., Weiping, L., and Zhong, C. (2010). Pdus : P2p-based distributed uddi service discovery approach. In *2010 International Conference on Service Sciences*, pages 3–8. IEEE.

Yun, Z., Huayou, S., Hengnian, Q., and Yulin, N. (2010). An approach to discover semantic web services in distributed environment based on chord. In *2010 IEEE International Conference on Intelligent Systems and Knowledge Engineering*, pages 401–405. IEEE.

Zhang, L. (2014). Owl-s based web service discovery in distributed system. In *2014 IEEE Workshop on Electronics, Computer and Applications*, pages 882–885. IEEE.

Zhang, Y., He, H., and Teng, J. (2013). Chord-based semantic service discovery with qos. In *2013 Fifth International Conference on Measuring Technology and Mechatronics Automation*, pages 365–367. IEEE.

Zhong, J. and Ying, H. (2005). A semantic web based peer-to-peer service registry network. In *Semantics, Knowledge and Grid, International Conference on*, pages 122–122. IEEE Computer Society.