Université Abou Bekr Belkaid
Tlemcen Algérie
جامعة أبي بكر بلقايد
تلمسان الجزائر

République Algérienne Démocratique et Populaire
Université Abou Bakr Belkaid– Tlemcen
Faculté des Sciences
Département d'Informatique

Mémoire de fin d'études

Pour l'obtention du diplôme de Master en Informatique

Option: *Géni Logiciel (G.L)*

Thème

Découverte hybride des services web à base Condorcet flou

Réalisé par :

- **Mme. BENDADDA Zeyneb Fatima Zahra**
- **Mr. MOUSSOUS Islam**

Présenté le 23 Juin 2015 devant le jury composé de MM.

- *Mr. MESSABIHI.M* *(Président)*
- *Mr. HADJILA .F* *(Encadreur)*
- *Mme. HALFAOUI.A* *(Examinateur)*
- *Mme. KHITRI.S* *(Examinateur)*

Année universitaire : 2014-2015

Remerciements

Avant tout nous formulons notre gratitude à Allah le tout puissant de nous avoir donné la force d'achever ce travail

Nous tenons aussi à remercier : Mr FETHALLAH HADJILA, notre encadreur, pour ses conseils, ses orientations, sa disponibilité et son encouragement qui nous ont permis de réaliser ce travail dans les meilleures conditions.

Ainsi Mr Messabihi.M « Président du jury », Mme. Halfaoui.A et Mme. Khitri.S pour avoir fait l'insigne honneur d'accepter de jurer notre mémoire.

Un très grand merci à tous les enseignants de notre département d'informatique qui ont assuré notre formation durant ces cinq ans d'études.

Nous ne saurons oublier de remercier nos parents pour leur contribution, leur soutien et leur patience, nos proches, nos amis et toutes les personnes qui nous ont aidées par leur soutien permanent de près ou de loin de nos études.

DÉDICACE

Je dédier ce modeste travail :

A

Mes très chers parents, en témoignage de leur amour, et dont le soutien de tous les instants m'a permis de mener à bien ce travail. J'espère que Dieu tout puissant me donne la force et le courage pour que je puisse rendre leurs sacrifices.

A
Mes chers frères et à mes chères sœurs
Pour leur encouragement, leur soutenir et leur aide pendant tout long année
A
tous mes amis sans exception
En témoignage de me sincère reconnaissance pour les efforts Qu'ils ont consentis pour me soutenir au cours de mes études.
Que dieu nous garde toujours unis
A
Toute personne qui nous aidés du près ou du loin à terminer ce modeste travail.

islam

DÉDICACE

Je dédie ce modeste travail à ceux qui me sont chers

Mon père

Qui m'a toujours poussé et motivé dans mes études, sans lui je ne serai pas devenue ce que je suis maintenant.

Ma chère mère

Parce qu'il est impossible de trouver les mots à la hauteur de l'amour et le soutient que vous m'avait toujours témoigné tous en long de ma vie et ma scolarité

A mon mari

Qui n'a jamais cessé de me soutenir et de m'apporter son aide, je ne le remercierai jamais assez pour tout ce qu'il a fait pour moi ;

A mes sœurs et mes frères

Ma véritable source de bonheur qui n'ont jamais cessé de croire en moi ;

A ma belle-famille

Pour tous l'aide et le soutient que m'ont apporté ;

A toute la famille Et A tous mes amis

Zeyneb

Table de matière :

INTRODUCTION GENERALE

Introduction générale

Contexte

Depuis quelques années, il est devenu naturel pour les organisations d'utiliser et de faire coexister des systèmes d'information, des processus différents ou des services. Cela permet à ces organisations de mener à bien des projets communs, ou de réaliser des coopérations afin de pouvoir continuer à progresser. Les échanges entre partenaires sont donc devenus de plus en plus nombreux et de plus en plus complexes.

Dans ce contexte, l'interopérabilité et l'intégration des applications sont devenues de plus en plus importantes afin de pouvoir satisfaire à la fois les besoins des citoyens et ceux des organisations, tout en rentabilisant les investissements consentis dans des systèmes généralement coûteux. Les organisations doivent à la fois connecter les nombreuses applications hétérogènes existantes, exploiter des données issues de systèmes d'information différents et définir de nouveaux processus tout en garantissant à terme la cohérence du système. C'est principalement pour répondre à cette problématique que sont nées les premières technologies d'applications réparties.

A la fin des années quatre-vingt-dix, plusieurs environnements sont utilisés pour mettre en pratique les applications réparties, il s'agit principalement de CORBA [OMG, 2008] (Common Object Request Broker Architecture), Java RMI [Downing, 1998] (Remote Method Invocation), DCOM (Distributed Component Object Model). [Horstmann et Kirtland, 1997]. Cependant l'interopérabilité qu'offraient ces derniers n'autorisait pas l'incompatibilité entre les langages de programmation.

Un inconvénient parmi plein d'autres qui ont suscité le développement des architectures orientées services (SOA) qui constituent une réponse à ces problématiques en termes de description, de réutilisabilité, d'interopérabilité et de réduction de couplage entre les différents systèmes (services). Ainsi, grâce à SOA, de nouveaux services peuvent être créés à partir d'une infrastructure informatique de systèmes déjà existante. Ces derniers peuvent être utilisés par des processus métier ou par des clients dans différentes applications.

La réalisation la plus importante d'une architecture SOA est les services web, ce sont des applications auto-descriptives, modulaires et faiblement couplées fournissant un modèle simple de programmation et de déploiement d'applications. Ils reposent

principalement sur des technologies basées sur SOAP pour la structure et le contenu de messages échangés entre services, WSDL pour la description des services, UDDI pour la découverte des services.

Problématique

En raison de l'augmentation rapide du nombre de services sur Internet. La découverte de services Web représente une thématique de recherche importante depuis l'émergence de la technologie des services Web. L'une des principales préoccupations, est de localiser, d'évaluer et de classer tous les services web qui soit adaptée (partiellement / ou globalement) aux besoins fonctionnels du demandeur (requête), à cause du nombre important de services qui offrent des fonctionnalités similaires. Plus précisément, nous avons donc besoin d'une approche de découverte de services efficiente et efficace assurant un bon rappel et une bonne précision des résultats en fonction de plusieurs critères de correspondance.

Contribution

En pratique, il n'y a pas de solution idéale qui donne toujours des résultats satisfaisants. En fait, nous distinguons différentes approches de découverte : nous avons des approches logiques, des approches non-logiques, et des approches hybrides qui combinent les catégories précédentes en vue d'améliorer la performance.

Dans notre travail, nous proposons en premier lieu une approche qui appartient à la classe hybride, elle consiste à choisir un mécanisme d'agrégation des scores partiels en score global. Pour calculer ce score global (qui gère tous les paramètres), nous adoptons une approche de vote majoritaire, et plus spécifiquement, nous calculons le classement selon le principe de Condorcet. En second lieu, nous proposons une relation floue pour comparer les services.

L'approche ainsi développée est dénommée 'Condorcet Flou'.

Organisation du manuscrit

Ce document est composé d'une introduction générale, de deux chapitres et d'une conclusion générale.

Le chapitre 1 : est consacré à l'étude des services Web et leur découverte pour mieux comprendre les concepts de base de cette technologie. Ce chapitre se divise en

deux parties : nous présentons dans la première partie L'architecture orientée services (SOA), ainsi que les services Web qui permettent de mettre en œuvre ce type d'architecture, le cycle de vie des services web et leurs architectures. Nous aborderons ensuite la technologie des services web traditionnels et sémantiques. Dans la deuxième partie nous analysons des approches de découverte des services Web, en vue d'identifier les mécanismes qui font leurs points forts et faibles qui par la suite pourront servir de guide pour déterminer les fondements d'une approche de découverte optimisée.

Le chapitre 2 : dans ce chapitre, nous présentons formellement nos besoins et nos objectifs, ensuite nous décrivons nos algorithmes de découverte. Ainsi, nous présentons le prototype destiné à la découverte de service web et la mise en œuvre de nos approches, nous montrons également des expérimentations de performances de ces algorithmes.

CHAPITRE I

WEB SERVICE

I. Introduction

Avec le développement rapide et l'augmentation de la complexité des systèmes, la communication entre machines devienne de plus en plus difficile. C'est principalement pour répondre à cette problématique que sont nées les premières technologies d'applications réparties. Mais inopportunément l'interopérabilité entre les langages de programmation incompatible qui n'est pas offerte par ces dernières, est une difficulté majeure qui a emmené à développer de nouveaux paradigmes d'interaction entre applications telles que la AOS L'architecture orientée services.

Plus spécifiquement, les services web qui sont une implémentation particulière de cette architecture, sont des composants logiciels interopérables permettant la communication et l'échange de données entre applications et systèmes hétérogènes dans des environnements distribués.

Dans ce chapitre, nous commençons par présenter l'architecture orientée services et sa principale réalisation, ensuite nous introduisons le concept de services web, son cycle de vie ainsi les principaux standards qui leur sont associés, et finalement nous présentons la découverte des services web et les différentes approches de cette dernière.

II. L'Architecture Orientée Service (AOS)

- Définition

L'AOS est un paradigme fondé sur la description et l'interaction de services, autrement dit L'AOS est une approche architecturale permettant la création des systèmes basés sur une collection de services développés dans différents langages de programmation, hébergés sur différentes plates-formes avec divers modèles de sécurité et processus métier [Barry, 2003]

Selon [Dodani, 2004], ‖ L'architecture orientée service permet l'intégration d'applications et de ressources de manière flexible en : (1) représentant chaque application ou ressource sous la forme d'un service exposant une interface standardisée, (2) permettant à un service d'échanger des informations structurées (messages, documents, ‖objets métier‖), (3) coordonnant et en organisant les services afin d'assurer qu'ils puissent être invoqués, utilisés et changés efficacement.‖

L'idée maîtresse de l'architecture orientée service est que tout élément du système d'information doit devenir un service identifiable, documenté, fiable, indépendant des autres services, accessible, et réalisant un ensemble de tâches parfaitement définies [David, 2004].

- Intérêt d'utiliser AOS

De nos jours l'AOS est devenu une nécessité, car c'est une architecture basée sur des standards, ainsi elle est caractérisée par son couplage faible entre les services et par l'indépendance par rapport aux aspects technologiques, en plus, elle sécurise l'investissement des applications existantes.

Cette architecture n'est pas simplement une mode, elle se place plutôt dans la continuité logique des multiples tentatives de distribution de traitements, de répartition de données, d'intégration d'applications, d'homogénéisation du système d'information, etc. L'adoption de la AOS a été grandement facilitée par l'émergence opportune de la technologie des services web et leurs standards bien définis. [Daniel et al.., 2004]

- Avantages de l'AOS

L'AOS encapsule plusieurs avantages bénéfiques pour le domaine de la technologie d'information et de communication. Elle Offre :

- ✓ la réutilisation et la composition : permettant le partage des modules entre les applications.
- ✓ la pérennité : impliquant le support des technologies existantes et à venir.
- ✓ l'évolutivité : permettant aux applications d'ajouter de nouveaux modules afin de répondre aux nouveaux besoins fonctionnels.
- ✓ réduction du coût : minimisant le coût de développement des grands projets.
- ✓ Une grande tolérance aux pannes avec une souplesse dans la maintenance.

III. Service Web

L'architecture orientée service est apparue pour pallier les limites des architectures distribuées. La technologie des services web représente la technologie la plus utilisée pour migrer vers AOS.

III-1Définition

Plusieurs définitions des services Web ont été mises en avant par différents auteurs.

Le consortium W3C définit un service Web comme suit [W3C', 2004] : « Un service web est un système logiciel destiné à supporter l'interaction ordinateur– ordinateur sur le réseau. Il a une interface décrite en un format traitable par l'ordinateur (e.g. WSDL). Autres systèmes réagissent réciproquement avec le service web d'une façon prescrite par sa description en utilisant des messages SOAP, typiquement transmis avec le protocole http et une sérialisation XML, en conjonction avec d'autres standards relatifs au web ».

Selon Justin et al. [Justin, 2002] : « *Un service web est une agrégation de fonctionnalités publiées pour être utilisées. Il utilise Internet comme conduit pour réaliser une tâche. Il est semblable à un processus métier virtuel qui définit des interactions au niveau application.* »

IBM [Colan, 2003] donne la définition suivante des services Web : « *Les services web sont la nouvelle vague des applications Web. Ce sont des applications modulaires, auto-contenues et auto-descriptives qui peuvent être publiées, localisées et invoquées depuis le web. Les services web effectuent des actions allant de simples requêtes à des processus complexes. Une fois qu'un service Web est déployé, d'autres applications peuvent le découvrir et l'invoquer* ».

Les services web sont des compléments aux programmes et applications existantes, développées dans différents langages de programmation, et servent de pont pour que ces programmes communiquent entre eux [Gilbert et al, 2003].En déduire que Toutes ces définitions partagent l'idée qu'un service Web est tout simplement une technologie permettant à des applications de dialoguer à distance via Internet, et ceci indépendamment des plates-formes et des langages sur lesquelles elles reposent. Pour ce faire, les services Web s'appuient sur un ensemble de protocoles Internet très répandus (XML, HTTP), afin de communiquer. Cette communication est basée sur le principe de demandes et réponses, effectuées avec des messages XML.

III-2Description du modèle fonctionnel des services web

Le fonctionnement des services web passe par plusieurs étapes : (voir la figure I.1)

- Le fournisseur de service se charge de l'enregistrement et de la publication de la description WSDL des services sur un ou plusieurs annuaires de

services UDDI, et cela se fait en envoyant un message (SOAP) à ce l'annuaire. Ce message contient la localisation du service, la méthode d'invocation (et les paramètres associés) ainsi que le format de réponse.

- Lorsqu'un utilisateur veut consulter un service, il interroge l'annuaire UDDI afin de savoir quels sont les services disponibles correspondant à ses besoins. Le serveur UDDI lui retourne la liste des services parmi lesquels il fait son choix, et il va récupérer l'URL identifiant ce service, donnant accès à une interface WSDL pour faciliter son utilisation et à partir de cette dernière il peut invoquer les méthodes et communiquer avec le fournisseur de ce service en utilisant les messages SOAP.

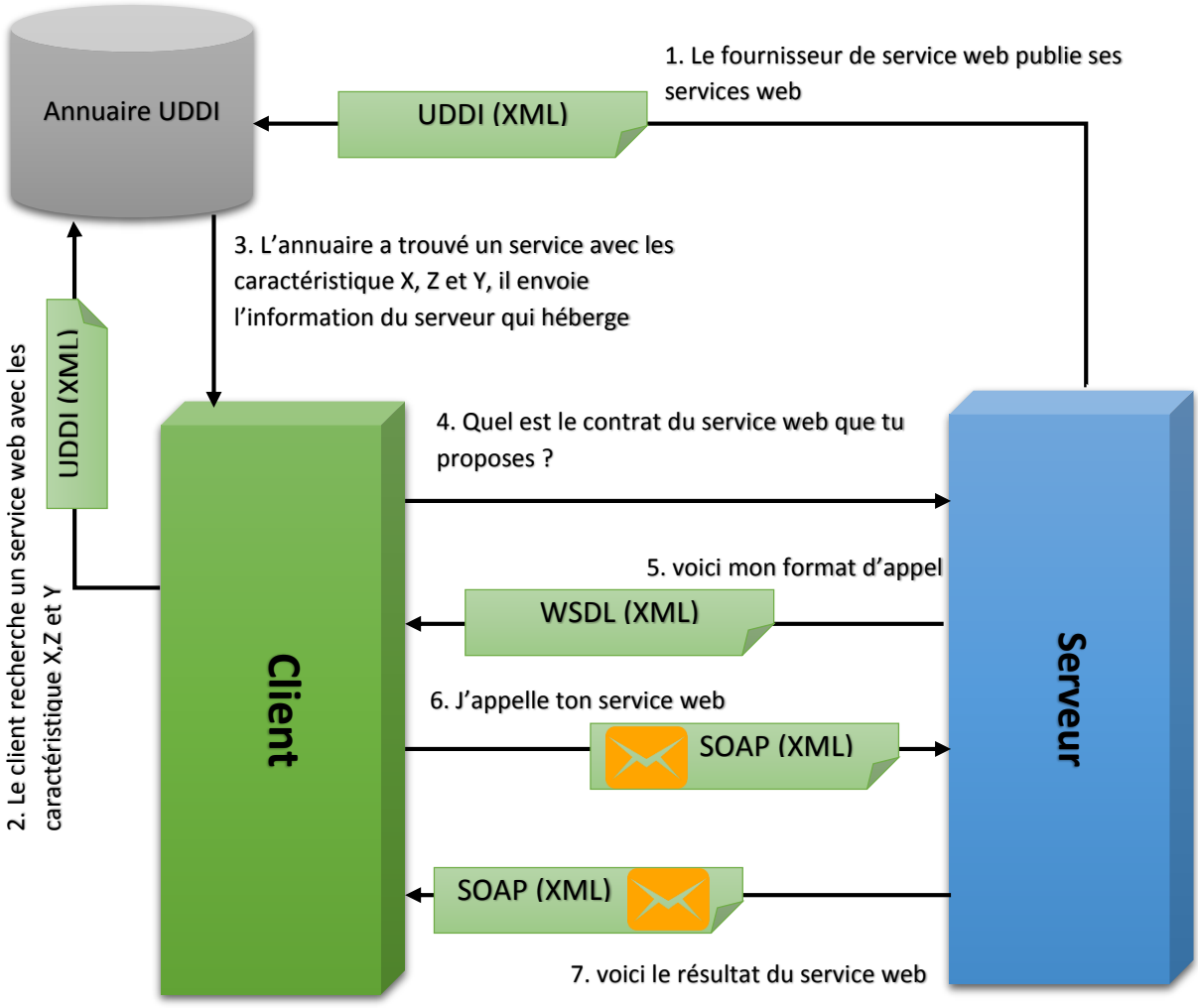

Figure I. 1 : Modèle fonctionnel des services web [VEZAIN, 2005].

III-3 Principales technologies de développement de Services Web

Une caractéristique qui a permis un grand succès de la technologie des services web est qu'elle est construite sur des technologies standards [Sanjiva, 2005].

Les technologies utilisées par les services Web sont HTTP, WSDL, XML, SOAP et UDDI. L'architecture standard d'un service Web est organisée en plusieurs couches. Chacune d'elles répond à des préoccupations fonctionnelles différentes telles que la découverte, la description, la communication et le transport, comme illustré sur la Figure I.2 [VEZAIN ,2005].

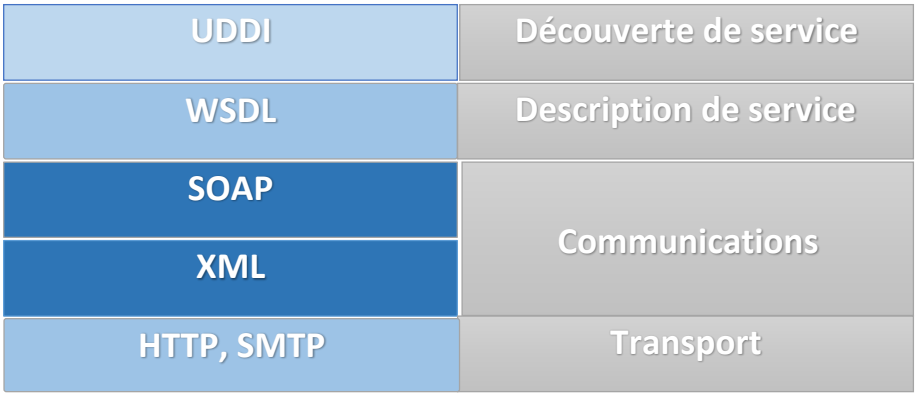

Figure I. 2 : Couches technologiques des Web Services [VEZAIN ,2005].

a. SOAP: Simple Object Access Protocol

SOAP [Gudin et al, 2003] est un protocole de communication indépendant des plateformes, c'est un produit de Microsoft et IBM. Sa première version a été acceptée par le W3C (Word Wide Web Consortium) en 2000.

SOAP est le protocole qui assure l'échange de messages dans les AOSs. Comme il est basé sur XML, il permet l'échange de données structurées indépendamment des plateformes, des langages de programmation ou des systèmes d'exploitation. Ainsi SOAP permet l'échange d'informations dans un environnement soit décentralisé soit distribué, indépendamment du contenu du message.

SOAP se base généralement sur les deux standards HTTP et XML :

- HTTP : est le protocole de transport des messages SOAP. Il constitue un bon moyen de transport pour sa réputation sur le web.

- XML pour structurer les requêtes et les réponses, indiquer les paramètres des méthodes, les valeurs de retours, et les éventuelles erreurs de traitements.

- Modèle d'échange de messages en SOAP

En dehors du protocole auquel SOAP est attaché, les messages sont routés sur un chemin appelé un 'message path'. Un message SOAP permet la transmission d'un message au format XML, d'un nœud (terminal virtuel) émetteur vers un nœud récepteur en passant éventuellement par un certain nombre de «nœuds» intermédiaires. Les nœuds intermédiaires, peuvent traiter plusieurs entêtes à la fois, faire des vérifications ou bien encore modifier, supprimer ou rediriger les messages. Figure I.3 montre les éléments qui participent au processus d'échange de messages SOAP

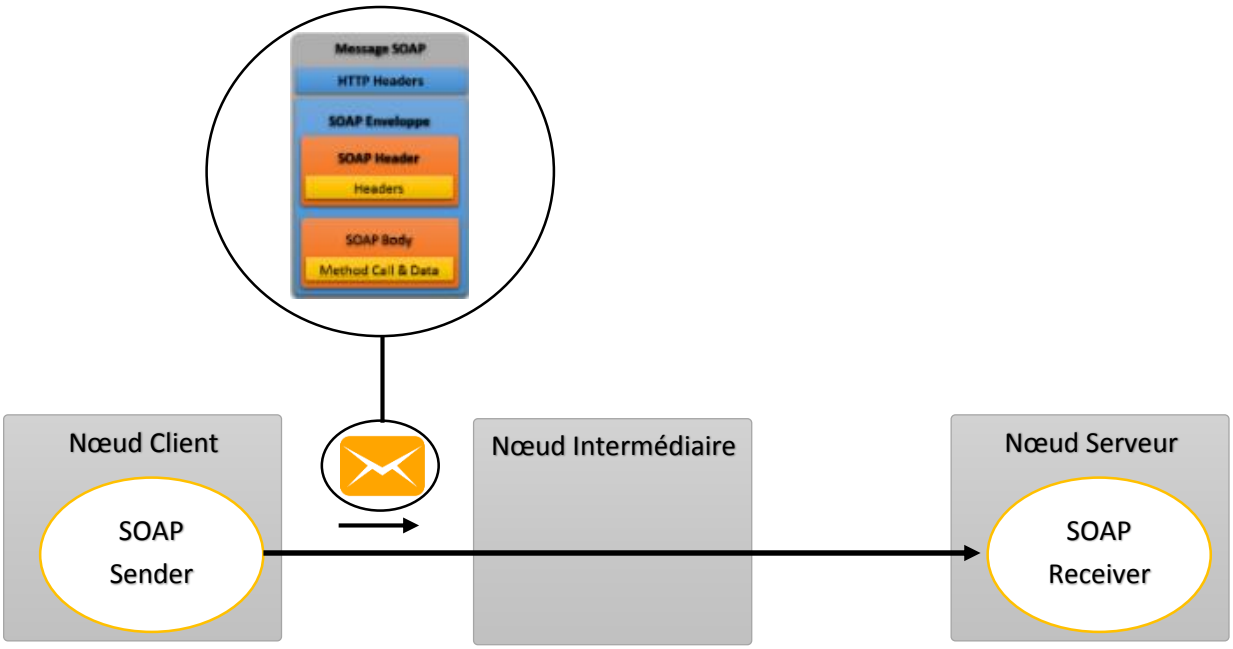

Figure I. 3 : Le chemin d'un message SOAP.

Quand un message SOAP arrive dans une entité SOAP, il suit le processus décrit ci-dessous [Martin, 2001] :

1. Identifier toutes les parties du message SOAP destiné à cette entité.
2. Vérifiez que toutes les parties obligatoires identifiées dans le pas 1 sont soutenues par l'entité et traitez-les en conséquence. Si ce n'est pas le cas rejeter le message.

3. Si l'entité n'est pas la destination suprême du message enlève alors toutes les parties identifiées dans le pas 1 avant l'expédition du message.

- Structure d'un message SOAP

Les messages transmis lors de l'utilisation du protocole SOAP sont basés sur le langage XML, un message SOAP est constitué de deux parties : l'en-tête de protocole de transport et l'enveloppe SOAP comme indiqué sur la Figure I.4.

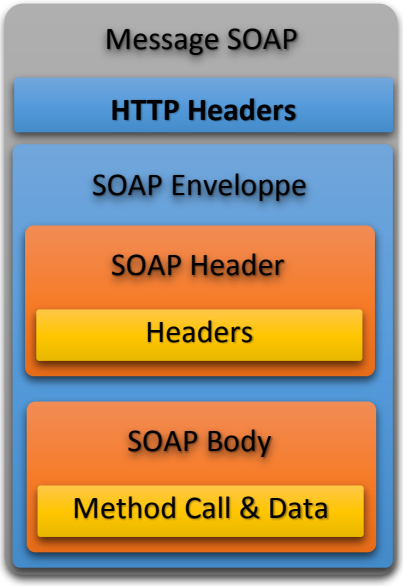

Figure I. 4 : Structure de message SOAP

1- **L'en-tête du protocole de transport :** qui dépend de protocole de transport utilisé, par exemple si le protocole HTTP est utilisé, l'en-tête comporte :
   - La version de protocole HTTP utilisée.
   - Le type d'encodage du contenu (généralement de type XML).

La figure I.5 montre la structure de L'en-tête du protocole de transport http du message

```
1  POST /globalweather.asmx HTTP/1.1
2  Host: www.webservicex.net
3  Content-type: text/xml; charset=utf-8
4  Content-Length: length
5  SOAPAction: "http://www.webserviceX.NET/GetWeather"
```

Figure I. 5 : L'en-tête du protocole de transport HTTP

2- **L'enveloppe SOAP :** est l'élément de base du message SOAP, elle se compose de deux partie la partie entête et la partie corps du message :

- ✓ **L'en-tête du message SOAP** : est un élément facultatif dans un message SOAP, cependant, si un en-tête est présent, il doit être le premier élément qui apparaît dans l'enveloppe SOAP. L'en-tête est utilisé principalement pour ajouter des fonctionnalités au message comme l'authentification et la gestion des transactions, il est à la disposition des clients et des services pour leur propre usage. L'en-tête du message SOAP commence avec la balise <soap:Header> et se termine avec la balise </soap:Header>. (Figure I.6)

```
<soap:Header>

<!-- Élément USER : À destination du nœud RightManager -->
      <m:User xmlns:m= "http://www.monsite.com/rights/"
      soap:actor="http://www.monsite.com/rights/RightsManager" >
                    Thunderseb

      </m:User>

 <!-- Élément Session : À destination du nœud final -->

      <m:Session xmlns:m="http://www.monsite.com/session/"
      soap:mustUnderstand ="1" >
                   12AE3C
      </m:Session>

 <!-- Élément Lang : À destination du prochain nœud -->

      <m:Lang xmlns:m = "http://www.monsite.com/lang/"
      soap:actor ="http://schemas.xmlsoap.org/soap/next"
      soap:mustUnderstand = "0" >
                        FR
       </m:Lang>
<soap:Header/>
```

Figure I. 6 : Entête de Message SOAP.

Trois attributs associés à l'en-tête SOAP peuvent être utilisés :

soap:mustUnderstand : indiquer si une entrée d'en-tête est obligatoire ou facultative, de ce fait, elle prend deux valeurs soit 1 ou 0, la valeur 1 signale que le récepteur doit reconnaître l'information présente dans l'entête et que son traitement est obligatoire.

soap:role(soap:actor(v1.1)) : sert à indiquer comment traiter l'entrée et par qui, autrement dit il sert à indiquer le destinataire SOAP auquel un bloc d'en-tête SOAP particulier est destiné, donc, la valeur de l'attribut d'acteur de SOAP est un URI.

soap:relay: est utilisé pour indiquer si un bloc d'en-tête SOAP ciblé sur un récepteur SOAP doit être relayé s'il n'est pas traité.

- ✓ **Le corps du message SOAP :** Le corps SOAP est un élément obligatoire dans le message SOAP, le style RPC impose un format précis aux messages SOAP, il contient les informations destinées au dernier Destinataire du message, ainsi il doit fournir le nom de la méthode invoquée par une requête ainsi que les paramètres associés à celle-ci, par contre le style DOC autorise n'importe quel document XML bien formé. Le corps du message SOAP commence avec la balise <soap:body> et se termine avec la balise </soap:body >.

La figure I.7 montre un exemple de message SOAP requête d'un service web qui donne le factoriel d'un nombre entier.

```
<?xml version="1.0" encoding="UTF-8"?>
<S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
    <SOAP-ENV:Header/>
    <S:Body>
        <ns2:Factoriel xmlns:ns2="http://leswebservice/">
            <number>6</number>
        </ns2:factoriel >
    </S:Body>
</S:Envelope>
```

Figure I. 7 : Message SOAP requête.

b. WSDL : Web Services Description Language

WSDL est un langage de description standard qui permet de définir la structure abstraite et concrète d'un service. En particulier, c'est l'interface présentée aux utilisateurs. Il indique comment utiliser le service Web et comment interagir avec lui. WSDL est un document XML qui permet de décrire de façon précise les détails concernant le service Web, tel que la signature des opérations offertes par le service (nom d'opérations, noms et types des paramètres d'entrées /sorties), il définit aussi des liaisons concrètes pour ces opérations telles que le protocole de transport, l'URI du service, le style du service, et les formats des messages d'entrée et de sortie.

Nous mentionnons que cette interface décrit juste la structure du service et non pas le comportement, c'est-à-dire qu'est-ce qu'un service Web est capable de faire, son emplacement et comment l'invoquer.

Le document WSDL comporte 05 sortes d'éléments XML : <types>, <message>, <opération>, <types de port >, <binding> et <service>. (Voir La figure **I.8)**

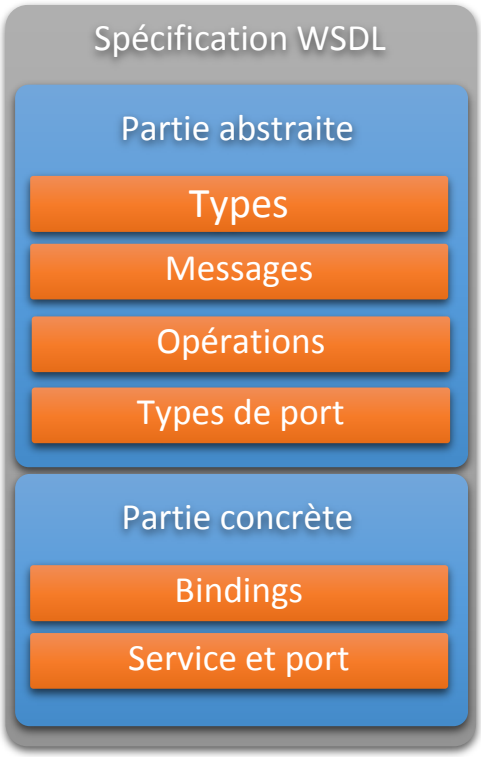

Figure I. 8 : Structure d'une description WSDL

Comme indiqué dans la figure I.8 WSDL se compose de deux parties de descriptions : une première partie de description abstraite contient le contrat du service, car elle est composée des éléments qui sont orientés vers la description des capacités du service Web, et une deuxième partie concrète qui est composée des éléments qui sont orientés vers le client pour le service physique, elle contient les liaisons (bindings) et la localisation du service.

Partie Abstraite :

- ✓ Types : définie les types de données utilisés par le service en utilisant généralement XML Schema.

- ✓ Message : est une unité de communication qui définit d'une manière abstraite des données transmises.
- ✓ Opération : qui décrit d'une manière abstraite les actions supportées par le service.
- ✓ Type de port (appelé Interface depuis WSDL2.0) : qui représente un ensemble d'opérations correspondant chacune à un message entrant ou sortant.

Partie concrète :

- ✓ Binding (Rattachement): qui est un protocole de communication et un format des données échangées pour un port– Port : qui est une adresse d'accès au service.
- ✓ Service : une collection de ports (port ou endpoints).
- ✓ Port : une adresse d'accès au service.

La figure I.9 correspond à la description des opérations offertes par un service qui calcule le factoriel d'un nombre entier dans le fichier WSDL.

```
<portType name="Factoriel">
<operation name="Factoriel">
<input wsam:Action="http://leswebservice/Factoriel/FactorielRequest"
message="tns:Factoriel"/>
<output wsam:Action="http://leswebservice/Factoriel/FactorielResponse"
message="tns:FactorielResponse"/>
</operation>
</portType>
```

Figure I. 9 : Exemple d'opération du service Factoriel dans un fichier WSDL

c. UDDI: Universal Description Discovery and Integration

L'annuaire des services UDDI est un standard fondé sur XML et plus particulièrement destiné aux services Web, il utilise SOAP pour la communication, son objectif principal la publication et la découverte des informations sur les services Web, La spécification UDDI est une collaboration d'ARIBA, Microsoft et IBM,

Selon « Peter » [Peter, 2012], UDDI est similaire à un annuaire téléphonique qui présente donc des structures similaires aux pages blanches, pages jaunes et pages vertes. (Voir la figure I.10)

Pages Blanches : Liste des entreprises, avec les informations associées par exemple : Nom entreprise, Description (texte), Contact (Noms, n ° tél., fax, Site Web, etc.), Identificateurs publics (N °série, etc.).

Pages jaunes : Représente la description textuelle de services et décrivent leurs informations de classification en catégories.

Pages vertes : Informations techniques précises sur les services fournis tels que la description de services et d'information de liaison et le processus métiers associés.

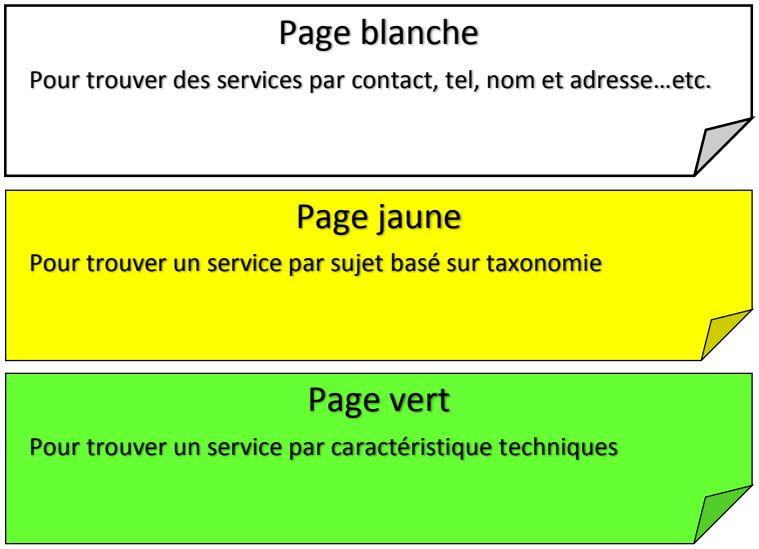

Figure I. 10 : Le contenu de l'annuaire UDDI

La norme UDDI offre aussi une API aux applications clientes, pour rechercher des services et/ou ses fournisseurs, ajouter ou modifier des services ou des entreprises. Nous distinguons des d'annuaires UDDI publiques et privés l'annuaire UDDI publique est implémenté sous forme d'un réseau de nœuds UDDI, ces nœuds sont synchronisés, chacun d'eux est possédé par une entreprise donné. La publication d'un service chez une entreprise propage automatiquement ses informations (pages blanches jaunes et vertes) aux différents nœuds UDDI. L'accès à l'ensemble des informations des registres peut se faire à n'importe quel nœud UDDI. Ce type d'annuaire est gratuit. L'annuaire UDDI privé n'est accessible que sur l'intranet d'une entreprise particulière (ou d'un groupe d'entreprises). [Hadjila, 2014]

UDDI est un annuaire orienté Business, et ses structures de données sont décrites comme suit (voir la figure I.11) :

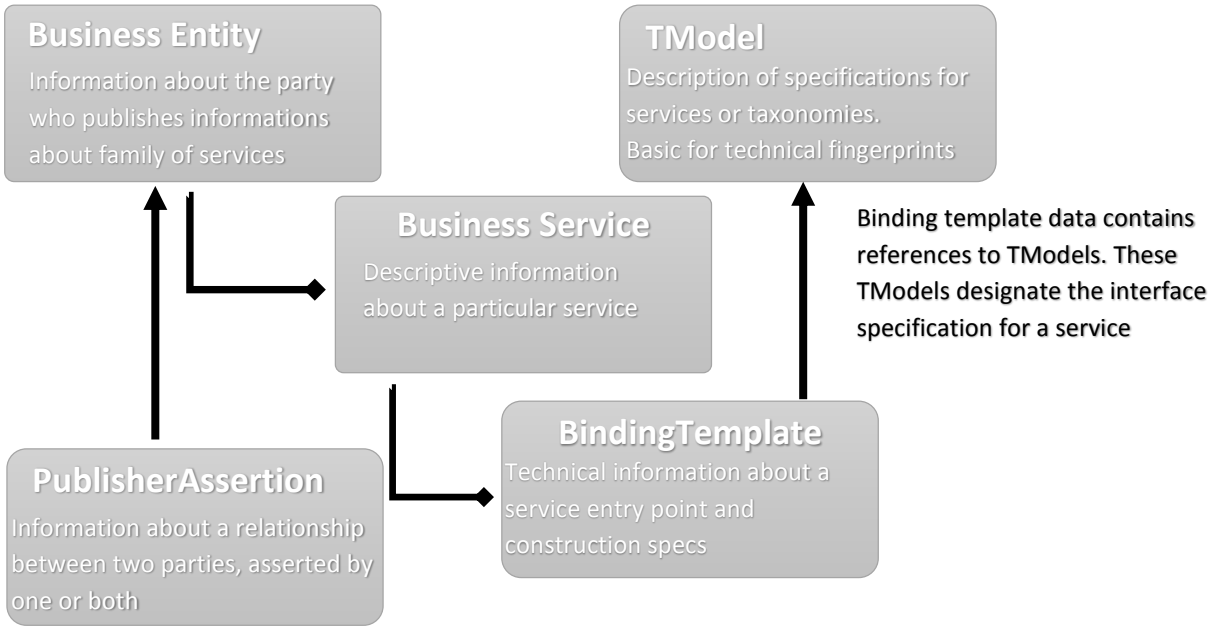

Figure I. 11 : La structure de l'annuaire UDDI

- **businessEntity :** elles décrivent de manière non technique la description d'une "entreprise', et les informations concernant les fournisseurs de services, On y trouve nom de l'entreprise, liste de contacts, les emails des fournisseurs, etc.
- **publisherAssertion :** donne les informations sur une relation entre deux parties, elle spécifie les liens d'affiliations entre deux entreprises (mère et fille). Lorsque deux éléments <businessEntity> référencent la même <publisherAssertion>, nous parlons de relation d'affiliation entre ces <businessEntity>. [Hadjila, 2014]

La partie « pages jaunes » est composée des éléments <businessService>.

- **businessService :** décrit la liste des services web offerts par l'organisation, On y trouve essentiellement le nom et la description textuelle des services, les informations de classification. Un <businessService> est un sous élément de <businessEntity>.

La partie « pages vertes » est composée des éléments <bindingTemplate> et <tModel>.

- **bindingTemplate :** décrit l'aspect technique du service offert, les coordonnées des services (URLs), les éventuels « tModels » associés. Un <bindingTemplate> est un sous élément de <businessService>.

- **tModel :** sont les descriptions techniques des services. Ce niveau que WSDL intervient comme le vocabulaire pour publier des descriptions techniques de services.

IV. Service web sémantique

Les technologies actuelles basées sur SOAP, WSDL et UDDI sont suffisantes pour mettre en place des composants interopérables et intégrables, mais elles ne permettent qu'une description syntaxique de l'interface des SW. Par conséquent, les technologies du web sémantique telles que les ontologies permettent de décrire sémantiquement les services Web toutes en facilitant leur utilisation automatique. La combinaison des deux dernières technologies, services web et ontologie ont donné naissance aux services Web sémantiques.

Le Web sémantique est donc une nouvelle approche pour l'organisation du contenu du Web instaurée dans le but d'améliorer l'interopérabilité, la découverte et la récupération de ressources.

Les approches les plus représentatives des Web services sémantiques sont WSDL-S [Akkiraju et al, 2005], SAWDL, OWLS [Martin et al, 2004], WSMO. Les deux premières sont des annotations sémantiques du standard WSDL, alors que les deux dernières sont des ontologies de services (elles ne sont pas directement liées à WSDL).

- **WSDL-S**

WSDL-S (Web Service Description Language-Semantic) [Akkiraju et al., 2005] est un langage WSDL augmenté de sémantique, cette extension de WSDL [Miller et al, 2004] permet l'ajout de trois éléments <category>, <precondition>, <effect> et deux attributs modelReference et schemaMapping.

- **Precondition**: un ensemble d'états sémantiques qui doivent être vérifié avant l'exécution de l'opération.
- **Effect:** un ensemble d'états sémantiques qui doivent être vrais après qu'une opération termine son exécution.

- **Category:** élément fournit un pointeur vers une certaine catégorie de la taxonomie, il permet de donner la fonctionnalité d'un service, aussi Il peut être utilisé sur une interface WSDL.
- **ModelReference :** un attribut pour lequel la valeur d'URL désigne un concept dans le modèle sémantique des services web, il permet l'association entre un composant WSDL et un concept d'une ontologie.
- **schemaMapping :** un attribut pour spécifier les correspondances entre la grammaire XML et les ontologies utilisées.

- **OWLS**

OWL-S (Web Ontology Language for Web services) [Martin et al, 2004] est une ontologie de haut niveau pour la description sémantique de services web .OWL-S se base sur le langage standard OWL, son objectif est de permettre l'automatisation de la recherche, de la découverte, de l'invocation et de l'interconnexion des Web services [Ankolekar et al, 2002].

Une description OWL-S se compose de trois concepts (voir figure I.12):

- **Le service profile :** exprime ce que le service web réalise.
- **Le process model :** définit le fonctionnement de service web.
- **Le service grounding :** définit comment accéder au service Web.

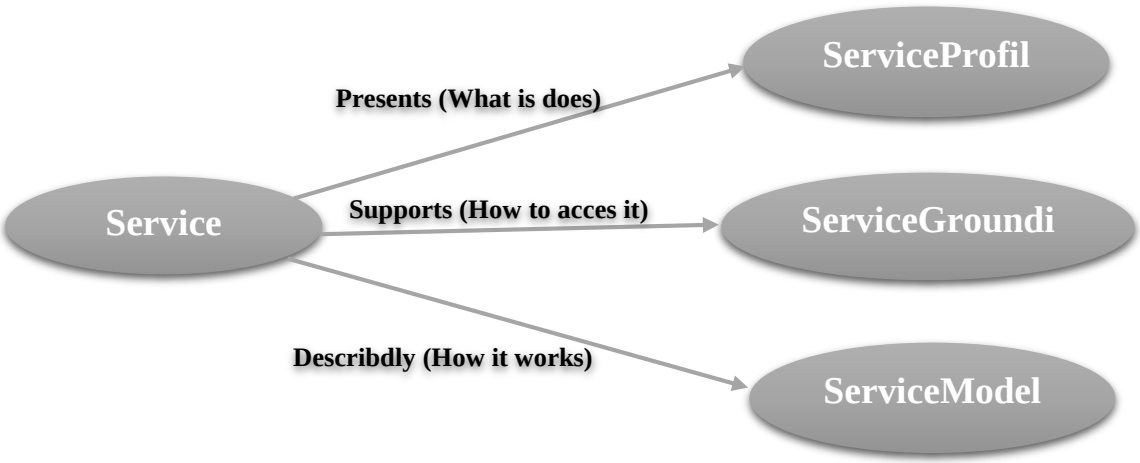

Figure I. 12 : Structure générale de l'ontologie OWL-S.

- **WSMO**

WSMO (Web Service Modeling Ontology) est une ontologie introduite par [Roman et al, 2005], elle définit les éléments de modélisation pour la description de différents aspects des services web sémantiques.

WSMO est organisée autour de quatre éléments principaux : les services web, les objectifs, les ontologies, et les médiateurs, chacun d'eux est décrit avec un formalisme basé sur le langage WSML5 [Bruijn et al, 2005].Ce modèle permet de réaliser un couplage faible entre les services web en utilisant un ensemble de médiateurs. Ces derniers assurent les tâches d'intégration d'ontologies, de découverte des services, de composition [Hadjila, 2014].

- **SAWSDL**

Semantic Annotations for WSDL and XML Schema (SA-WSDL) [Farrell et Lausen, 2007] est une recommandation W3C établie en aout 2007, elle ajoute des extensions au standard WSDL, elle permet d'utiliser tous types d'ontologies (OWL, WSML et UML). L'objectif de SAWSDL est de définir comment une annotation doit être réalisée.

SAWSDL présente deux sortes d'annotations sémantiques une pour identifier les concepts sémantique représentés par l'attribut modelReference et la deuxième pour faire le lien entre les concepts et le fichier WSDL référencés par les attributs liftingSchemaMapping et loweringSchemaMapping.

V. La découverte des services web

Le succès des services Web a impliqué son adoption par divers fournisseurs de services à travers le Web, ce qui a conduit à la croissance du nombre des services web et leurs diversités, par conséquent leur découverte devient de plus en plus une tâche difficile.

V-1 Définition

Le W3C définit la découverte de services comme suit: "*Web service discovery is the act of locating a machine-processable description of a Web service that may have been*

previously unknown and that meets certain functional criteria. It involves matching a set of criteria with a set of Web service descriptions. The goal is to find an appropriate Web service." [W3C, 2004 b].

[Booth et al, 2004] décrivent le processus de découverte, comme étant la localisation d'une description compréhensible par la machine d'un service éventuellement inconnu au préalable et correspondant à certains critères fonctionnels.

[Toma et al, 2005] Définissent la découverte comme le processus qui prend en entrée une requête utilisateur et retourne une liste de ressources ou services, pouvant combler éventuellement le besoin décrit.

Toutes ces définitions partageant la même idée qui dit que la découverte est un point clé nécessaire à la réutilisation des services, elle se base sur un besoin présenté sous forme de requête.

Le principe de la découverte est simple : un fournisseur de services publie les descriptions des fonctionnalités de ses services dans un annuaire de services. De l'autre côté, un demandeur de services utilise cet annuaire pour localiser un service et par la suite l'invoquer.

V-2 Approches de Découverte

Dans la littérature, et comme mentionné dans [Schumacher, 2008], il existe trois catégories d'approches de découverte de services Web : les approches logiques, les approches non-logiques et les approches hybrides qui visent à combler les limites des deux premières approches.

Les approches de découverte qualifiées de non-logiques sont basées sur le calcul du degré de similarité textuelle à partir de graphes structurés construits à cet effet ou encore le calcul de distance (le chemin) entre concepts. Tandis que les approches de découverte logique sont basées essentiellement sur des approches déductives, et les approches qui combinent les mécanismes logiques et non-logiques sont qualifiés d'hybrides.

a. Approches Logiques

[Paolucci et al, 2002] présentent l'une des premières approches sémantiques pour la découverte de services, pour cela, ils comparent les sorties de la requête avec les sorties du service offert.

Il existe principalement trois types d'appariement pour calculer le degré de pertinence :

- **IO-matching** : Ce type d'appariement est déterminé à partir des données sémantiques des paramètres de service : les entrées : inputs (I) et les sorties : outputs (O). [Srinivasan et al., 2004a] [Fan et al., 2005] [Paolucci et al., 2002].
- **PE-matching** : Ce type d'appariement est déterminé à partir d'appariements sur des pré-conditions (P) et des effets (E) des services et des requêtes. [Schumacher et al., 2008].
- **IOPE-matching** : Ce type d'appariement est déterminé à partir d'appariements sur les données sémantiques des inputs (I), des outputs (O), des préconditions (P) et des effets (E) des services et des requêtes. [Jaeger et al., 2005] [Keller et al., 2005] [Stollberg et al., 2007] [Küster et KonigRies, 2008] .

On présente par la suite, comme exemples de systèmes basés sur cette catégorie de découverte L'appariement sémantique de « WSC ».

❖ **WSC** :

L'appariement sémantique de WSC (Web Services Capabilities) est basé sur l'utilisation des ontologies DAML (prédécesseur de OWL [W3C, 2004 a]) [Paolucci et al., 2002], c'est-à-dire Les services publiés et les requêtes font référence à des concepts DAML qui les décrivent sémantiquement. L'appariement adopté dans WSC est de type IO-matching.

Un service publié est considéré comme un service qui répond bien à une requête lorsque tous les outputs de la requête correspondent à des outputs du service publié, et tous les inputs du service publié correspondent à des inputs de la requête. Quatre degrés d'appariement sont proposés : EXACT, PLUGIN, SUBSUMES et FAIL [Paolucci et al., 2002].

La figure I.13 présente l'algorithme des règles d'affectation des degrés d'appariement des outputs. Considérons « *S* » un service publié, « *Q* » une requête de service, et « *outS* » et « *outQ* » output de service respectivement de la requête. Selon l'algorithme le degré EXACT est défini lorsque les deux outputs de S et Q sont équivalents ou lorsque outputs de « Q » est une sous classe de output de « S ». Le degré PLUGIN est identifié si un output de « S » est plus générique qu'un output de « Q ». Le degré SUBSUMES correspond au cas où l'output de « Q » est plus générique qu'un output de « S ». Le degré

FAIL est définit lorsque aucune relation d'équivalence ou d subsomption n'existe entre eux.

```
DegreeOfMatch (outQ, outS):
if outS = outQ then return exact
if outQ subClassOf outS then return exact
if outS subsumes outQ then return plugIn
if outQ subsumes outS then return subsumes
otherwise fail
```

Figure I. 13 : Règles d'affectation des degrés d'appariement d'outputs dans WSC [Paolucci et al, 2002].

Le critère principal de tri dans WSC se base sur la sélection du service possédant le meilleur score au niveau des outputs. La comparaison des scores d'appariement des inputs obtenus pour chaque service, n'est utilisée qu'en deuxième lieu où les scores d'appariement des outputs de ces services sont égaux. (Voir La Figure I.14)

```
sortRule (score1,score2) {
if (score1.output > score2.output) then score1 > score2
if ((score1.output = score2.output)
     & (score1.input > score2.input)) then score1 > score2
If ((score1.output = score2.output)
     & (score1.input = score2.input))  then score1 = score2
```

Figure I. 14 : Algorithme de tri des appariements des services dans WSC. [Paolucci et al., 2002].

b. Approches Non Logiques

Selon [Hadjila, 2014] « *Les approches logiques possèdent plusieurs inconvénients, la complexité élevée du raisonnement affaiblit les chances de scalabilité, en outre les raisonneurs logiques ont un faible rappel (beaucoup de faux négatifs), puisque la subsomption ne couvre pas tous liens sémantiques, et en dernier nous constatons que la majorité des services web actuels ne sont pas annotés avec des langages logiques formels*. ».

Pour cette raison la fréquence des termes au niveau des services, peut être efficace dans la découverte. Cette approche est fondée sur le calcul du degré de similarité textuelle, ou encore sur le calcul de distance entre les concepts, elle utilise des mécanismes syntaxiques, structurels et numériques tels que le concept de distance numérique, l'appariement de graphes structurés, la similitude syntaxique, etc.

On présente par la suite, comme exemples de systèmes basés sur cette catégorie de découverte « **iMatcher1 ».**

❖ **iMatcher1 :**

iMatcher1 [Schumacher et al., 2008] est un système de découverte non logique de services Web qui utilise un matchmaker (apparieur) syntaxique de profils de services. Il prend en entrée un ensemble de profils de services web décrit en OWL-S. Ces services sont stockés sous forme de graphes RDF (Resource Description Framework) sérialisés dans une base de données RDF, avec une extension de RDQL (RDF Data Query Language), appelée iRDQL [Bernstein et Kiefer, 2005].

Le degré d'appariement entre la requête et un service est calculé à partir de quatre métriques de calcul de similarité syntaxique : TFIDF (Term Frequency-Inverse Document Frequency) [Jones, 1972] [Salton, 1983], la mesure du vecteur Cosinus [Garcia, 2006] [Zhu et al., 2010] et la mesure de divergence de Jensen-Shannon [Fuglede et Topsoe, 2004]. Les résultats sont classés en fonction des scores numériques de ces mesures de similarités syntaxiques et d'un seuil défini par l'utilisateur.

c. Approches Hybrides (Logiques et Non Logiques)

Les approches hybrides utilisent une combinaison entre les deux approches précédentes logiques et non logiques, pour remédier aux limites de chacune de ces deux approches.

❖ **OWLS-iMatcher2 :**

OWLS-iMatcher2 [Kiefer et al., 2007] [Kiefer, 2009] [Kiefer et Bernstein, 2008] est le successeur hybride de iMatcher1,c'est une approche hybride de découverte dont la composante logique se base sur un appariement entre les concepts d'inputs/outputs et la composante non logique utilise un appariement basé sur la similarité textuelle des noms et des signatures des services [Bernstein et al., 2005] [Bernstein et Kiefer., 2005].

L'évaluation d'appariement utilise une valeur binaire de pertinence sémantique ainsi que les différentes valeurs de métriques de similarité telles que : Bi-Gram [Kiefer, 2009] et les mesures de Levenshtein [Levenshtein, 1996] [Gusfield, 1997], Monge-Elkan [Kiefer, 2009] et Jaro [Winkler et Thibaudeau, 1991] [Kiefer, 2009].

L'algorithme d'appariement utilisé calcule les valeurs de similarité syntaxique entre une requête donnée et tous les services disponibles, puis utilise un modèle mathématique de régression pour prédire la valeur d'agrégation d'appariements pour chaque service. Les services découverts sont renvoyés à l'utilisateur dans l'ordre décroissant des valeurs d'appariement obtenues. Une évaluation statistique des résultats retournés ainsi leur présentation graphique lui permettent de les interpréter et les évaluer facilement [Chabeb, 2011].

- **WSMO-MX :**

L'approche WSMO-MX [Kaufer et Klusch, 2006] est une approche hybride d'appariement de services décrits en WSML-MX [Klusch et al., 2008]. Elle utilise un appariement sémantique [Klusch et al., 2006] et syntaxique, aussi elle procède par des techniques d'appariement de graphes orienté objet héritées du DSD-Matchmaker [Klein et König-ries, 2004 a] et d'appariement intentionnel de services [Keller et al., 2005].

Le degré global d'appariement est calculé par l'agrégation de quatre éléments, l'appariement des types ontologiques, l'appariement logique des contraintes spécifiées en F-logic (Frame Logic), l'appariement des noms de relations et la mesure de similarité syntaxique.

VI. Conclusion

Dans ce chapitre nous avons présenté les principales notions et concepts des services web, donc nous avons décrit, plus particulièrement les primordiaux standards tel que le protocole SOAP, l'interface WSDL, et l'annuaire UDDI, en suite nous avons présenté les travaux existants sur le problème de découverte avec l'étude des avantages et des inconvénients de quelque uns.

A travers les différentes sections que nous avons montrées, nous concluons que l'AOS et plus particulièrement les services web sont à l'heure actuelle de plus en plus incontournable, ils sont maintenant capables d'échanger des données de façon quasi autonome à travers internet.

CHAPITRE II

CONCEPTION ET RÉALISATION

I. Introduction

La découverte des services web est une étape primordiale, dans le travail réalisé dans le cadre de notre projet vise à implémenter plusieurs approches de découverte des services web, nous décrivons en premier lieu les outils utilisés ainsi que toutes les étapes de conception de notre application commençant par les algorithmes jusqu'à leurs implémentations.

II. Corpus de Découverte

Les données utilisées dans nos expérimentations sont issues d'un extrait du corpus open source OWLS-TC version 2.2.1, Ce dernier est développé par le centre allemand pour la recherche en intelligence artificielle (http://www.dfki.de/scallops). Cette base de données est constituée de 1007 descriptions de service définis sous forme OWL-S. Elles sont classées en sept domaines différents : le domaine d'éducation, le domaine médical, le domaine de nourriture, le domaine militaire, le domaine de voyages (tourisme), le domaine de communication, et le domaine d'économie.

La plupart de ces services sont extraits de l'annuaire UDDI d'IBM, et sont traduits du format WSDL en format OWLS de façon semi-automatique. La base contient aussi un ensemble de requêtes reparties sur les 07 classes, ces requêtes sont décrites en OWLS. Chaque document OWLS possède un « profile: hasinput » et « profile: hasoutput », ces derniers sont utilisé dans la partie de découverte des services web. Chaque service web (i.e. document OWLS) est classé manuellement par des experts humains comme étant pertinent ou non par rapport à une requête donnée. Ceci permet le calcul des rappels et des précisions des approches proposées. La base offre aussi un ensemble d'ontologies pour décrire les services et les requêtes. Dans ce qui suit, nous donnons un exemple d'une description de service web et d'une description de requête.

- **Requête**

Cette requête cherche le prix d'un livre.

Concepts d'entrée : Book.

Concepts de sortie : Price.

- **Service**

Ce service donne le prix d'un petit déjeuné, en spécifique un café et un sandwich.

Nom de service : coffeesandwich_price_service.

Concepts d'entrée : -Coffee.

-Sandwich.

Concepts de sortie : Price.

III. Conception

Nous allons présenter nos trois approches d'agrégation : « Borda », « Condorcet » et « Condorcet Flou », chaque une de ces approches combine différemment cinq (5) approches de similarité qui sont : Cosinus (Cos), Extended Jaccard (Ej), Loss Information (Li), jenson shanon (Js) et logique (Log) en utilisant une fonction d'agrégation.

Nous adoptons dans ce qui suit, les principales notations suivantes :

$Sim_x(S, R)$: retourne une valeur de similarité selon la méthode 'x' entre un service 'S' et une requête 'R'.

$W_{i,R}$: désigne le poids du i-ème index dans la requête 'R'.

$W_{i,S}$: désigne le poids du i-ème index dans le service 'S'.

- La mesure de similarité Cosinus :

$$Sim_{Cos}(S,R) = \frac{\sum_{i=1}^{M} w_{i,R} \times w_{i,S}}{\sqrt{\sum_{i=1}^{M} w_{i,R}^2} \times \sqrt{\sum_{i=1}^{M} w_{i,S}^2}}$$

- La mesure de similarité extended Jacquard :

$$Sim_{EJ}(S,R) = \frac{\sum_{i=1}^{M} w_{i,R} \times w_{i,S}}{\sum_{i=1}^{M} w_{i,R}^2 + \sum_{i=1}^{M} w_{i,S}^2 - \sum_{i=1}^{M} w_{i,R} \times w_{i,S}}$$

- La mesure de similarité loss of information :

$$Sim_{LOI}(S,R) = \frac{|PC_{R,x} \cup PC_{S,x}| - |PC_{R,x} \cap PC_{S,x}|}{|PC_{R,x}| + |PC_{S,x}|}$$

Avec $X \in \{In, Out\}$, $PC_{R,x}$ et $PC_{S,x}$ ensemble de composants logique Input/output de la requête 'R' et le service 'S'

- La mesure de similarité Jensen-Shannon :

$$Sim_{JS}(S,R) = \frac{1}{2\log 2}\sum_{i=1}^{n} h(p_{i,R}) + h(p_{i,S}) - h(p_{i,R} + p_{i,S})$$

Avec une probabilité de fréquence du terme par exemple : $p_{i,R}$ désigne la probabilité du i-ème index dans la requête R, et h (x) = -xlogx.

- Logique :

Est un algorithme qui utilise le test de subsomption pour établir la parenté entre une requête et un service. Le test de subsomption donne cinq scores : Exact, Plugin, Subsumer, SubsumedBy et Fail.

III.1 Approche Borda

La méthode Borda est un système de vote, et se déroule comme suite :

1. Avec X choix, on affecte X points au candidat préféré, X-1 au deuxième candidat préféré et ainsi de suite.
2. Le candidat qui obtient le plus grand nombre de point remporte le vote.

La figure suivante montre l'algorithme proposé pour cette approche.

Algorithme Borda

Entré : liste_serv, methode[] ;
Sortie : liste_serv
Début
  Pour i ← 0 à taille(liste_serv) **faire**
    score ← 0 ;
    Pour j ← 0 à taille(methode) **faire**
      Rang = pos(liste_serv[i], methode[j])
      score = score + taille(liste_serv) – Rang
    FinPour
    ajouter_score (liste_serv[i], score) ;
  FinPour
Retourner trié_borda(liste_serv) ;
Fin

Figure II. 1 : Algorithme de Borda

✓ Les fonctions utilisées :

La fonction « Pos » : prend en paramètre (le service ; et la liste des résultats de la méthode choisie pour laquelle on fait le traitement). Et elle retourne la position de ce service dans cette liste.

La fonction « Ajouter_score » : prend en paramètre le service et le score, et elle va les associer.

✓ explication de l'algorithme :

Ligne 1 : nous avons mis pour entrées, un tableau qui contient les résultats des méthodes de similarité, et pour notre cas nous avons utilisé cinq listes : Cos, Ej, Li, Js et Log ces listes contiennent des services ordonnées selon un score.

Ligne 2 : notre algorithme va retourner comme sortie, une liste des services trié selon l'approche de Borda.

Ligne 4..11 : algorithme pour calculer le score de chaque service avec toutes les méthodes de similarité.

Ligne 10 : nous avons fait appel à la fonction « trié Borda» pour faire le tri de la liste des services selon le score calculé.

✓ Exemple

Notre exemple est constitué de 4 services triés selon les 5 méthodes de similarité cos, Ej, Li, Js et Log, les résultats de classement est donnée dans le tableau méthode [].

Nous appliquons cet exemple pour les 3 approches de découverte (Borda, Condorcet, Condorcet flou).

liste_serv : contient la liste de tous les services

S1	S2	S3	S4

Tableau II. 1 : la liste de tous les services

Methode[] : est un tableau qui contient les service classé selon chaque approche, le service est caractérisé par son nom, input et output.

COS			EJ			LI			JS			LOG		
Nom	**In**	**Out**	**Nom**	**In**	**Out**	**Nom**	**In**	**Out**	**Nom**	**In**	**Out**	**Nom**	**In**	**Out**
S1	1	0.8	**S2**	1	0.8	**S4**	1	0.8	**S3**	1	0.8	**S2**	1	0.8
S2	0.9	0.8	**S1**	0.9	0.8	**S1**	0.9	0.8	**S4**	0.9	0.8	**S3**	0.9	0.8
S3	0.8	0.8	**S3**	0.8	0.8	**S3**	0.8	0.8	**S2**	0.8	0.8	**S4**	0.8	0.8
S4	0.4	1	**S4**	0.4	1	**S2**	0.4	1	**S1**	0.4	1	**S1**	0.4	1

Tableau II. 2 : les services classés selon chaque approche,

La position de s1 dans Cos est 0 ↔ Pos(s1,cos) = 0 → score = 0 + (4-0) = 4.

Le tableau suivant montre les résultats de calcul de l'approche borda

Nom	Calcul de score	Classement final
S1	4 +3+3+1+1 = 12	**3**
S2	3+4+1+2+4 = 14	**1 (vainqueur)**
S3	2+2+2+4+3 = 13	**2**
S4	1+1+4+3+2 = 11	**4**

Tableau II. 3 : Résultats de calcul selon l'approche de Borda

Le classement final de ces services est donné dans le graphe suivant :

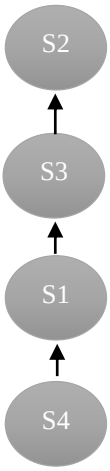

Figure II. 2 : Graphe de classement selon Borda

III.2 Approche Condorcet

La méthode Condorcet (vote Condorcet) est un système de vote dans lequel l'unique vainqueur est celui, qui comparé tour à tour à tous les autres candidats, s'avèrerait à chaque fois être le candidat préféré. Chaque électeur classe les candidats par ordre de préférence. Pour chaque paire de candidat, on détermine le nombre d'électeur ayant voté pour l'un ou l'autre. Ainsi pour chaque pair, il y a un candidat vainqueur. Dans la plupart des cas, il y a un unique candidat qui remporte tous ses duels : il s'agit du vainqueur du vote.

La figure suivante montre l'algorithme proposé pour cette approche.

Algorithme Condorcet

```
Entré : liste_serv, methode[] ;
Sortie : liste_serv
Début
  Pour i ← 0 à taille(liste_serv) faire
     Pour j ← i+1 à taille(liste_serv) faire
        S1 ← 0 ;     S2 ← 0 ;
        Pour k ← 0 à taille(methode) faire

              Si (Pos(liste_serv[i], methode[k]) >Pos(liste_serv[j], methode[k])) alors
                    S1 ← S1+1 ;
              Sinon
                    S2 ← S2+1 ;
              FinSi

        FinPour

        Si (S1>S2) alors
              Ordonne(liste_serv[i], liste_serv[j])
        Sinon
              Ordonne(liste_serv[j],liste_serv[i])
```

```
        FinSi
      FinPour
    FinPour
Retourner liste_serv;
Fin.
```

Figure II. 3: Algorithme de Condorcet

✓ Les fonctions utilisées **:**

La fonction «ordonne» : prend en paramètre deux services (service a, service b), et elle va classer le service 'a' avant le service 'b'.

✓ Explication de l'algorithme

Ligne 1 : nous avons mis pour entrées, un tableau qui contient les résultats des méthodes de similarité, et pour notre cas nous avons utilisé cinq listes citées précédemment qui contiennent des services ordonnées selon un score.

Ligne 2 : notre algorithme va retourner comme sortie, une liste des services trié selon l'approche de « Condorcet ».

Ligne 3..7 : Nous comparons chaque service avec ses successeurs c'est-à-dire avec tous les services qui le suivent jusqu'à la fin de la liste des services.

Ligne 7..12 : Pour chaque méthode, on teste si la position de service courant est supérieur à celle du service suivant alors on incrémente son score, sinon on incrémente le score du servie suivant.

Ligne 13..18 : Si le score de service i est supérieur au score du service j alors la fonction ordonne va classer le service i avant le service j, sinon l'inverse.

✓ Exemple

En utilisant les mêmes entrés de l'exemple cité précédemment le résultat avec l'approche Condorcet seront comme suite :

S1 est classé meilleur que S2 dans Cos et Li (score S1 = 2), par contre S2 est classé meilleur que S1 dans Ej, Js et Log (score S2 = 3).

Le tableau suivant montre les résultats de calcule entre les services

services	*Calcule score*	*services*	*Calcule score*	*décision*
S1 > S2	1 + 1 = 2	**S2 > S1**	1+1+1 = 3	**S2 > S1**
S1 > S3	1+1+1 = 3	**S3 > S1**	1+1 = 2	**S1 > S3**
S1 > S4	1+1 = 2	**S4 > S1**	1+1+1 = 3	**S4 > S1**
S2 > S3	1+1+1 = 3	**S3 > S2**	1+1 = 2	**S2 > S3**
S2 > S4	1+1+1 = 3	**S4 > S2**	1+1 = 2	**S2 > S4**
S3 > S4	1+1+1+1 = 4	**S4 > S3**	1	**S3 > S4**

Tableau II. 4 : Résultats de calcule selon l'approche de Condorcet

Le graphe suivant présente le résultat final de trié selon l'approche de Condorcet :

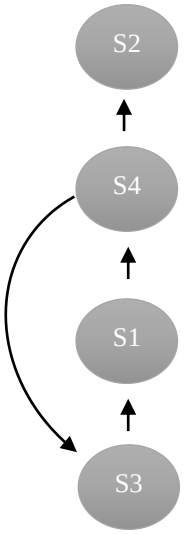

Figure II. 4 : Graphe de classement selon Condorcet

L'inconvénient majeur dans l'approche de Condorcet est la présence de cycles et parmi les approches qui ont minimisé ce problème de cycles c'est le Condorcet flou.

III.3 Approche Condorcet flou

Condorcet flou est la combinaison de l'approche de Condorcet vu précédemment, et la relation de dominance « flou ».

- **La relation floue :**

  - **La formule**

$$\begin{cases} si\ (score1 - score2) < \mathcal{E}\ alors\ 0 \\ sinon\ (score1 - score2) \end{cases}$$

✓ Exemple

Ɛ = 0.01

s1 = 0.07

s2 = 0.05

s1 - s2 = 0.02 > Ɛ Donc *DF* (s1, s2) =0.02

s2 - s1 = -0.02 < Ɛ Donc *DF* (s2, s1) =0

Algorithme Condorcet flou

Entré : liste_serv, methode[] ;

Sortie : liste_serv

Début

```
Pour i ← 0 à taille(liste_serv) faire
    Pour j ← i+1 à taille(liste_serv) faire
        S1 ← CalculScore(liste_serv[i], liste_serv[j], methode)
        S2 ← CalculScore(liste_serv[j], liste_serv[i], methode)
        Si (S1>S2) alors
            Ordonne(liste_serv[i], liste_serv[j])
        Sinon
            Ordonne(liste_serv[j],liste_serv[i])
        Finsi
    FinPour
FinPour
Retourner liste_serv;
Fin.
```

Figure II. 5: Algorithme de Condorcet Flou

Fonction CalculScore

Entrée : **service a, service b,** methode[]

Sortie : score

Initialisation : eps ← 0.01, scoreFinal← 0

```
Pour L← 0 à taille(methode) faire
    score ← 0
    Pour k ← 0 à taille(methode) faire
            scoreInput ← 0 ; scoreOutput ← 0
```

```
différence ← (Input( a, methode[L]) - Input(b, methode[k]))
Si (différence > eps)  alors
        scoreInput ←  différence;
Sinon
        scoreInput ← 0;
FinSi
différence ← (Output(a,  methode[L]) - Output(b, methode[k]))
Si (différence > eps)  alors
        scoreOutput ← différence;
Sinon
        scoreOutput ←0 ;
Finsi
score ← score + (scoreInput + scoreOutput) /2
FinPour
scoreFinal ← scoreFinal+ score/ taille(Methode) ;
FinPour
retourner scoreFinal/ taille(Methode)
```

Figure II. 6 : Fonction qui fait le calcul du score du Condorcet Flou

- ✓ Les fonctions utilisées :

La fonction « Input » : retourne le score d'Input dans l'approche sélectionné, elle a deux paramètre : le service et la liste de similarité sélectionné.

La fonction « Output » : retourne le score d'Output dans l'approche sélectionné avec les mêmes paramètres de la fonction Input.

La fonction « CalculScore » : cette fonction fait le calcul de score selon la relation de dominace « floue », elle prend comme paramètre le service 'a' et le service 'b' et ainsi un tableau qui contient les résultats des cinq méthodes de similarité.

Ligne 3 : initialisation du eps (epsilon) à 0.01 et le scoreFinal à 0.

Ligne 5 : le calcul de la différence entre l'Input du service 'a' et l'Input du service 'b' (a-b).

Ligne 6..10 : Si la différence est supérieur à 'eps' alors **'scoreInput'** du service 'a' prend la différence, sinon '**scoreInput'** reçoit 0.

Ligne 11 : le calcul la différence entre l'Output du service 'a' et l'Output du service 'b' (a-b)**.**

Ligne 12..17 : Si la différence supérieur à 'eps' alors **'scoreOutput'** du service 'a' prend la différence, sinon '**scoreOutput'** reçoit 0.

Ligne 18 : le calcul de score se fait en ajoutant la moyenne du **'scoreInput'** et **'scoreOutput'** à l'ancien score.

Ligne 20 : La fonction calcul retourne le résultat du score calculé précédemment divisé par le nombre des méthodes de similarité.

- ✓ Explication de l'algorithme

Ligne 1 : nous avons mis pour entrées, un tableau qui contient les résultats des méthodes de similarité, et pour notre cas nous avons utilisé cinq listes citées précédemment qui contiennent des services ordonnées selon un score.

Ligne 2 : notre algorithme va retourner comme sortie, une liste des services trié selon l'approche de « Condorcet Flou».

Ligne 4..15 : nous comparons chaque service avec leurs successeur c'est-à-dire avec tous les services qui le suivent jusqu'on termine tous la liste des services,

- ✓ Exemple

En utilisant le même exemple des deux dernières approches les résultats avec l'approche Condorcet Flou seront :

CalculScore(s1,s2,methode) = (0.09+0.06+0.06+0.06+0.08) / 5 = 0.07

services	***Calcule score***	***services***	***Calcule score***	***décision***
S1 > S2	0.074	S2 > S1	0.104	S2 > S1
S1 > S3	0.066	**S3 > S1**	0.096	**S3 > S1**
S1 > S4	0.098	**S4 > S1**	0.088	**S1 > S4**
S2 > S3	0.0539	**S3 > S2**	0.054	**S3 > S2**
S2 > S4	0.11	**S4 > S2**	0.07	**S2 > S4**
S3 > S4	0.1	**S4 > S3**	0.06	**S3 > S4**

Tableau II. 5 : résultats de calcule selon l'approche de Condorcet Flou

S3 a gagné trois fois par rapport aux autres services donc, il est sélectionné comme un vainqueur de Condorcet, maintenant il reste à classer les 3 services restants S1, S2 et S4. S2 a gagné deux fois par rapport à S1 et S4 donc, il sera classé le deuxième, à la fin il reste deux service S1 et S4. S1 a gagné une fois par contre S4 n'a jamais gagné donc, S1> S4.

Le classement final de ces services est donné dans le graphe suivant :

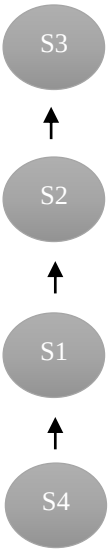

Figure II. 7 : Graphe de classement selon Condorcet flou

IV. Présentation de prototype

IV.1 Outils et environnement de développement

Avant de commencer l'implémentation de notre application, nous allons tout d'abord spécifier les outils utilisés qui nous ont semblé être un bon choix vu les avantages qu'ils offrent, Notant que Java est un langage de programmation orienté objet, libre, simple et portable, nous avons utilisé le langage de programmation « Java » (JDK 1.7.0), avec l'IDE « NetBeans 7.0 ».

Les API's utiliser :

- JDOM : est une API open source Java son but est de représenter et manipuler un document XML, elle utilise des classes plutôt que des interfaces. Ainsi pour créer un nouvel élément, il faut simplement instancier une classe, nous avons utilisé cette API aussi pour manipuler les fichiers de description des web services et des requêtes.
- JFreeChart : est une API open source Java sous licence LGPL, sa documentation est payante. elle permet la création des graphiques et des diagrammes de très bonne qualité.

IV.2 Présentation de l'interface d'application

➢ **Fenêtre d'accueil :**

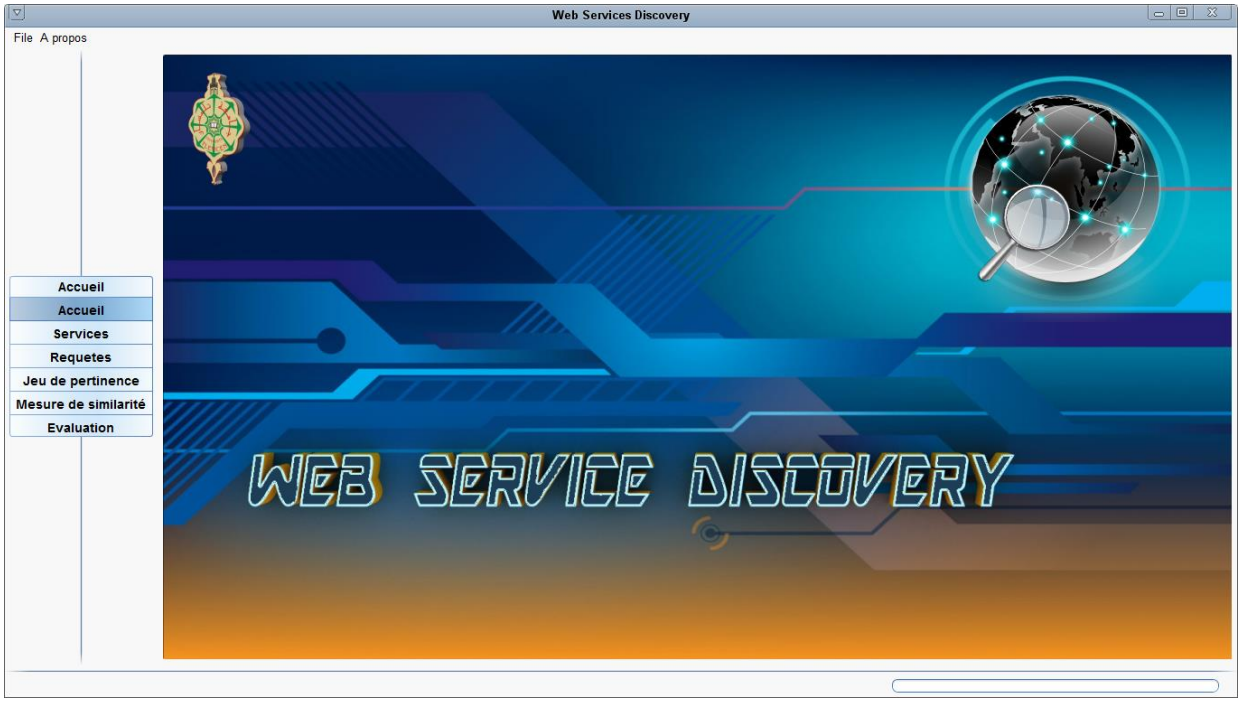

Figure1 : Fenêtre d'accueil.

➢ **Fenetre des services web :**

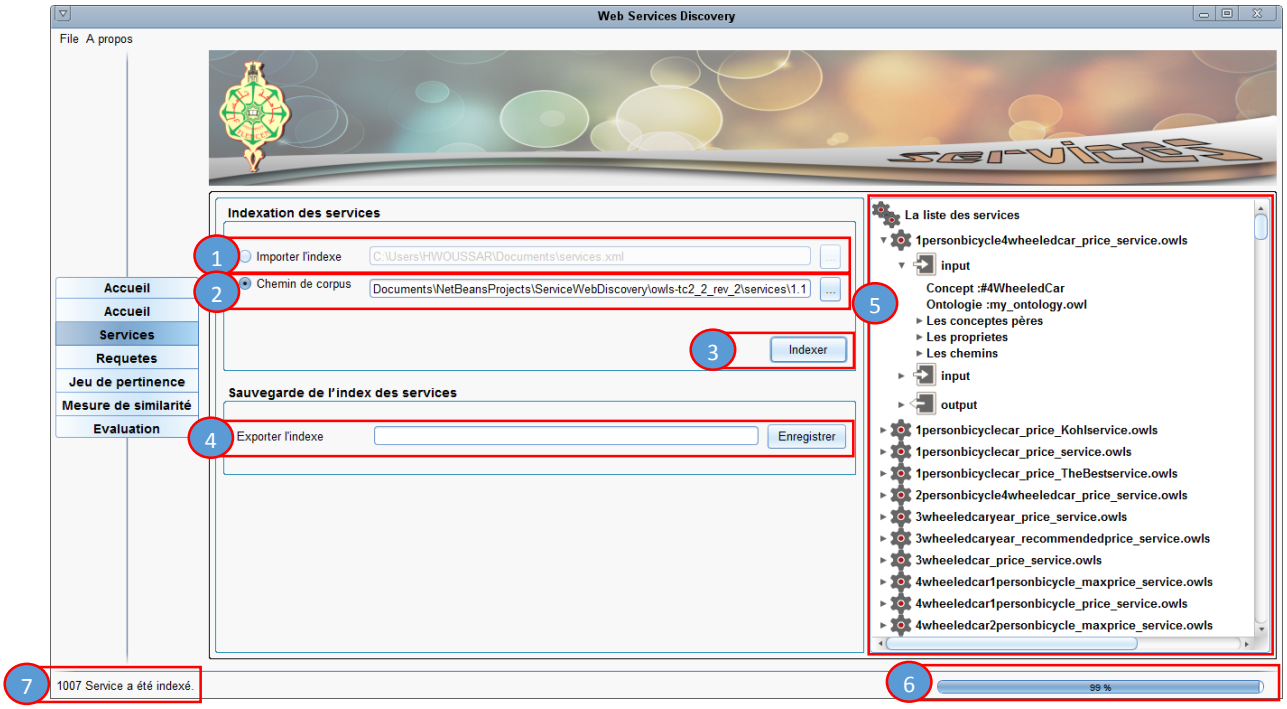

Figure2 : Fenêtre de services web.

Commentaire :

1- Choisir l'indexation des services web à partir d'un fichier d'index qui a été sauvegardé précédemment dans l'étape 4.

2- Choisir l'indexation à partir d'un corpus qui contient les fichiers de description des services web.

3- Bouton pour lancer l'indexation.

4- Sauvegarder l'index dans un fichier XML pour pouvoir l'ouvrir dans l'étape 1.

5- L'affichage de la liste des services web indexé.

6- La barre de progression pour indiquer l'état d'avancement de l'opération qui est en cours d'exécution.

7- Zone de texte pour afficher l'opération qui est en cours d'exécution.

➢ **Fenêtre des requêtes :**

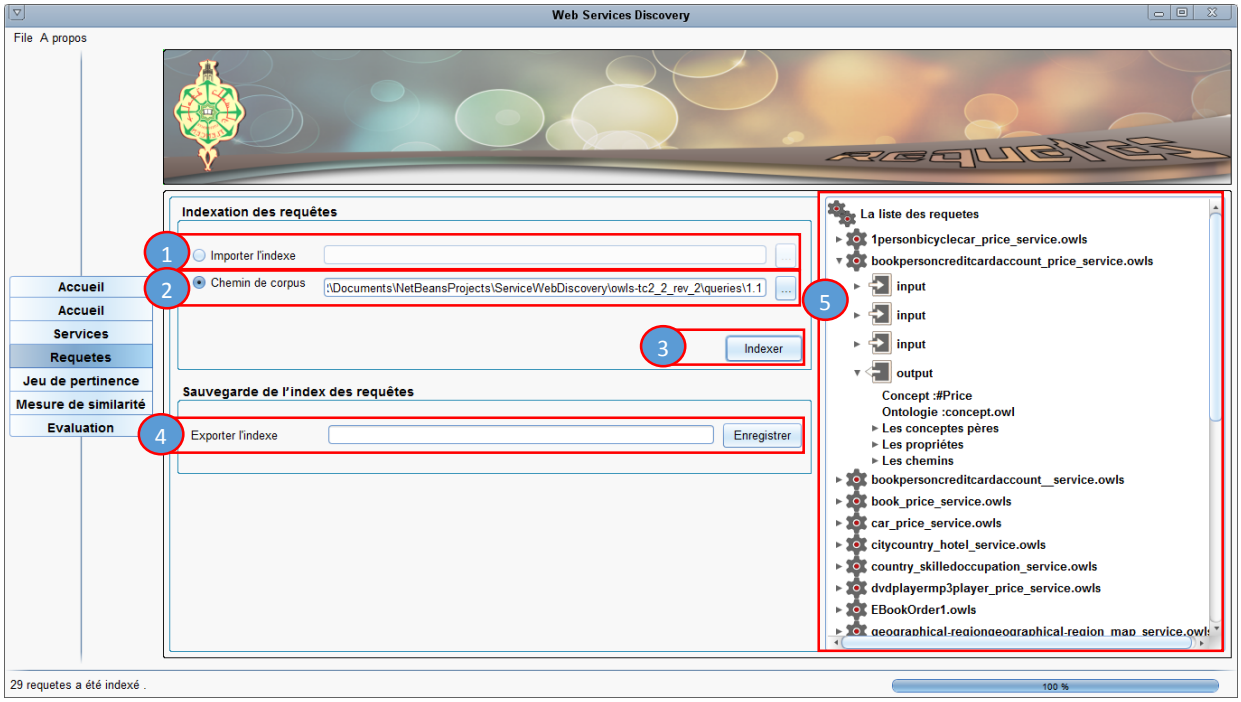

Figure3 : Fenêtre des requêtes.

Commentaire :

1- Choisir l'indexation des requêtes à partir d'un fichier d'index qui a été sauvegardé précédemment dans l'étape 4.

2- Choisir l'indexation des requêtes à partir d'un corpus qui contient les fichiers de description des requêtes.

3- Bouton pour lancer l'indexation.

4- Sauvegarder l'index dans un fichier XML pour pouvoir l'ouvrir dans l'étape 1.

5- L'affichage de la liste des requêtes indexé.

➢ **Fenêtre de jeu de pertinence :**

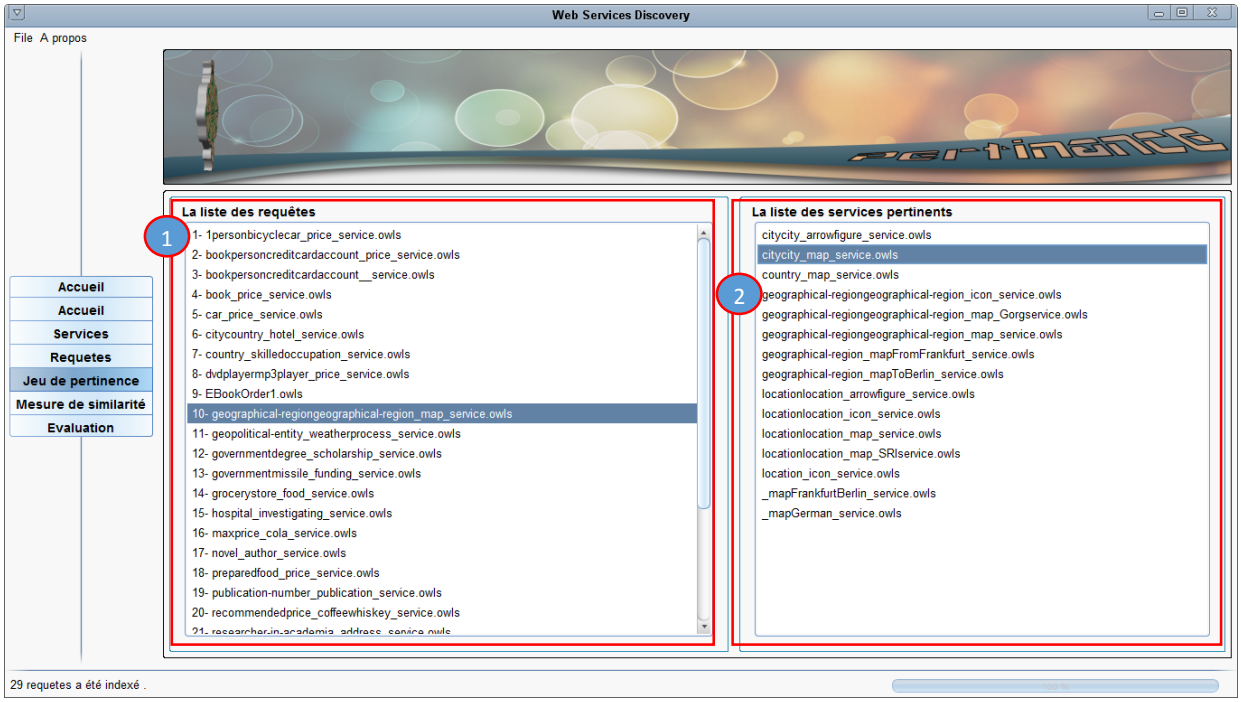

Figure 4: Fenêtre de jeu de pertinence.

Commentaire :

1- La liste des requêtes indexées précédemment.

2- La liste des services pertinents pour la requête sélectionnée.

➢ **Fenêtre de mesure de similarité :**

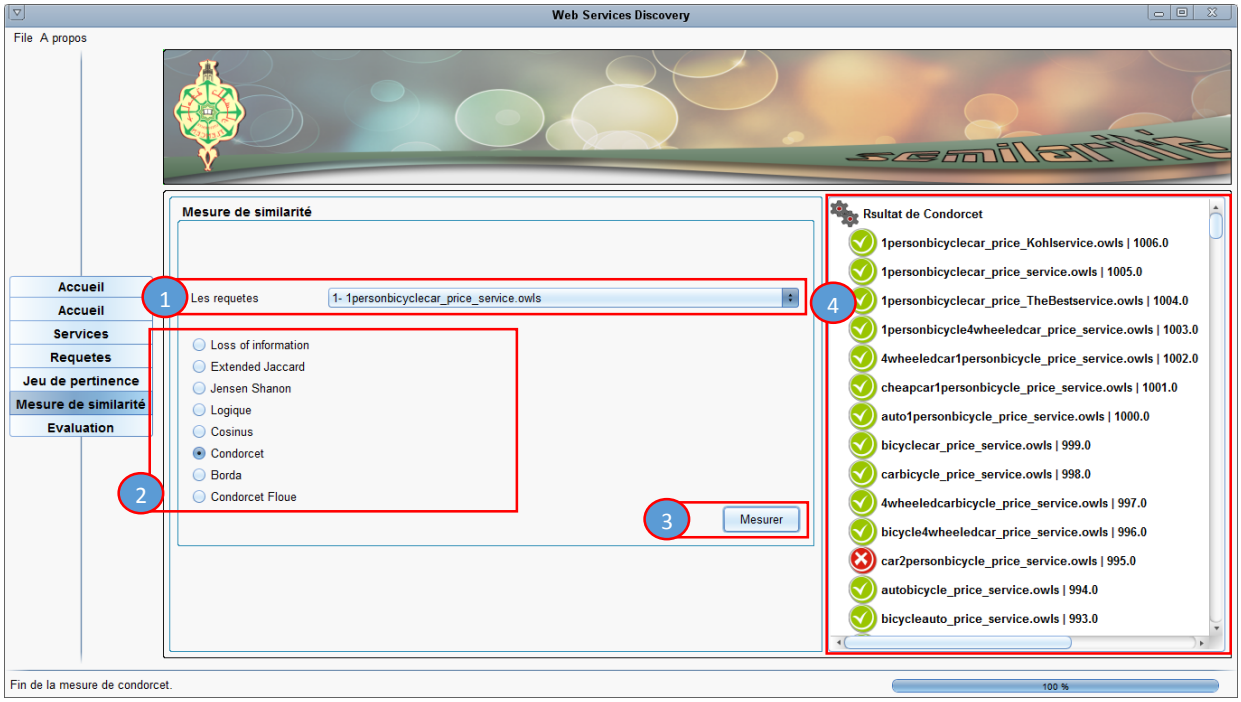

Figure 5: fenêtre de mesure de similarité.

Commentaire :

1- La liste des requêtes indexées précédemment.

2- La liste des approches de mesure de similarité.

3- Botton pour lancer le calcul de mesure de similarité de la requête sélectionné avec l'approche sélectionnée.

4- Le résultat de mesure de similarité (la liste des services classés)

L'icône vert indique que le service est pertinent et l'icône rouge indique que le service n'est pas pertinent.

5- Fenêtre d'évaluation :

Figure 6: Fenêtre d'évaluation.

Commentaire :

1- La liste des requêtes indexées précédemment.

2- La liste des critères qu'on peut évaluer.

3- Botton qui permet d'afficher le graphe dans une nouvelle fenêtre.

4- La liste des approches.

5- Le graphe de résultat.

V. Expérimentation

D'une façon générale, toutes les approches de similarité ont deux objectifs principaux : retrouver tous les documents pertinents, et rejeter tous les documents non pertinents. Ces objectifs sont évalués par les mesures de rappel et précision.

Nous procédons à une série d'expériences pour évaluer l'efficacité des approches, nous avons utilisé 29 requête pour évaluer la qualité des approches (précision, rappel et F-mesure). Tous les algorithmes ont été implémentés en Java, et les expériences ont été menées sur une machine Intel(R) Core(TM) i5-2410M CPU@ 2.30 GHz avec 4 Go de RAM, en cours d'exécution sous Windows 8.1.

Dans cette étude, nous avons mis en place cinq fonctions de correspondance et trois algorithmes de fusion, qui sont respectivement : Extended Jaccard (EJ), Loss Of Information (LI), Jenson shanon (JS), mesure cosinus (cosinus), logique (Logic), Borda, Condorcet et Condorcet Flou. Et nous avons pris (eps) $\varepsilon = 0.01$.

- ✓ Précision

Elle mesure la proportion de services pertinents relativement à l'ensemble des services rendus par le système. Elle est exprimée par

Précision = vp/(vp+fp). Avec :

vp : les vrais positifs. (Un résultat correct, et considéré comme juste par le système).
fp : les faux positifs. (Un résultat erroné, mais considéré comme juste par le système).

Le tableau suivant montre la moyenne de précision pour les 29 requêtes.

Approche	Top 10	Top 20	Top 30	Top 40	Top 50	Top 60	Top 70
EJ	0.817	0.666	0.575	0.476	0.412	0.367	0.325
LI	0.817	0.666	0.576	0.478	0.419	0.372	0.33
JS	0.817	0.678	0.571	0.478	0.413	0.374	0.327
Logique	0.766	0.586	0.506	0.416	0.354	0.306	0.269
Cos	0.828	0.671	0.572	0.473	0.406	0.37	0.327

Borda	0.834	0.684	0.563	0.467	0.401	0.35	0.309
Condorcet	0.828	0.68	0.585	0.484	0.418	0.373	0.33
Condorcet Flou	0.865	0.729	0.605	0.507	0.435	0.389	0.338

Tableau II. 6 : moyenne de précision pour les 29 requêtes

- ✓ Rappel

Il mesure la proportion de services pertinents restitués par le système relativement à l'ensemble des services pertinents contenus dans l'annuaire. Il est exprimé par :

Rappel = vp/(vp+fn). Avec :

vp : les vrais positifs. (Un résultat correct, et considéré comme juste par le système).
fn : les faux négatifs. (Un résultat correct, et considéré comme faux par le système).

Le tableau suivant montre la moyenne du rappel pour les 29 requêtes.

Approche	**Top 10**	**Top 20**	**Top 30**	**Top 40**	**Top 50**	**Top 60**	**Top 70**
EJ	0.394	0.612	0.759	0.810	0.851	0.891	0.911
LI	0.394	0.61	0.758	0.814	0.864	0.904	0.921
JS	0.393	0.614	0.741	0.813	0.856	0.905	0.915
Logique	0.369	0.518	0.633	0.684	0.71	0.726	0.735
Cos	0.395	0.607	0.742	0.806	0.846	0.903	0.917
Borda	0.393	0.602	0.716	0.767	0.808	0.834	0.849
Condorcet	0.397	0.619	0.764	0.821	0.862	0.905	0.923
Condorcet Flou	0.411	0.656	0.785	0.852	0.89	0.938	0.943

Tableau II. 7 : moyenne du Rappel pour les 29 requêtes

- ✓ F-mesure

La F-mesure est la combinaison du rappel et de la précision :

F-mesure= (2*rappel*précision) /(rappel+précision)

Le tableau suivant montre la moyenne du F-mesure pour les 29 requêtes.

Approche	Top 10	Top 20	Top 30	Top 40	Top 50	Top 60	Top 70
EJ	0.497	0.602	0.62	0.569	0.528	0.495	0.457
LI	0.497	0.601	0.62	0.572	0.537	0.503	0.463
JS	0.496	0.607	0.610	0.572	0.531	0.504	0.46
Logique	0.465	0.515	0.529	0.488	0.446	0.407	0.373
Cos	0.5	0.601	0.612	0.567	0.522	0.501	0.46
Borda	0.501	0.604	0.596	0.55	0.508	0.468	0.430
Condorcet	0.503	0.611	0.628	0.579	0.536	0.504	0.464
Condorcet Flou	0.523	0.653	0.647	0.603	0.555	0.524	0.475

Tableau II. 8 : moyenne du F-mesure pour les 29 requêtes

VI. Discussion

La précision dans toutes les approches se diminue, si on compare les 5 méthodes de similarité on trouve que la méthode ‘cos’ prend le meilleur score dans Top10 et Top70 cependant la méthode ‘Li’ prend le meilleur score dans Top30, Top40 et Top50, la méthode ‘Js’ prend le meilleur score dans Top20, Top40, Top60 et Top70, ainsi la méthode ‘logique’ donne des score faible. Donc on conclut que ‘Js’ a des meilleurs résultats par rapport aux autres.

Ensuite on compare la meilleure méthode ‘Js’ avec les approches de fusion. En comparant avec borda, cette dernière prend un meilleur score seulement dans Top10 et Top 20, donc ‘Js’ reste meilleur que borda malgré la fusion, par contre Condorcet a donné des meilleur résultats, enfin ‘Condorcet flou’ donne les meilleur résultats dans tous les Top-k.

Le rappel augmente dans toutes les approches mais on remarque une disparité des valeurs entre les méthodes. En comparant les 5 méthodes de similarité, la méthode ‘Ej’ atteint le meilleur score dans Top10 par contre ‘Li’ reste toujours la meilleure méthode, car elle atteint le meilleur score dans Top40, Top50 et Top70.

Si on compare ‘Li’ avec ‘Borda ‘ on remarque bien que ‘Li’ est meilleur que ‘Borda’, par contre ‘Condorcet’ a donné de meilleur résultats, et enfin le ‘Condorcet Flou’ reste la méthode qui atteint le meilleur score dans tous les Top-k.

Dans F-mesure on remarque que le score augmente jusqu’à Top30 il atteint son maximum et après se diminue, donc nous allons juger l’efficacité de l’approche en se basant sur le meilleur critère qui est Top30. Parmi les 5 méthodes ‘Li’ et ‘Ej’ prend le meilleur score, et si on prend aussi en considération les autres Top-k restants, on trouve que ‘Li’ est meilleur à ‘Ej’ et même meilleur que ‘Borda’, par contre ‘Condorcet’ est meilleur, et Enfin ‘Condorcet Flou’ atteint le meilleur score dans tous les Top-k.

VII. Conclusion

Ce chapitre a été consacré à la mise en œuvre et la validation de notre approche de découverte qui a pour but de combiner la méthode ‘Condorcet’ et la relation ‘flou’. Nous avons présenté les détails des travaux expérimentaux de notre approche qui a permis de comparer cette dernière avec différentes approches de mesure de similarité : logique, non logique et hybride, ce qui nous a confirmé qu’elle est plus efficace en termes de rappel et de précision que d’autres.

CONCLUSION ET PERSPECTIVE

Conclusion :

Les services Web sont la plus récente technologie adoptée pour l'intégration et l'interopérabilité des systèmes répartis. Ils sont caractérisés par leurs indépendances aux plateformes et aux systèmes d'exploitation, ce qui a impliqué leur adoption par les différentes entreprises et organisations commerciales et industrielles, ces organisations offrant leurs services à distance via le Web, ce qui augmente le nombre de services offerts (publies). Et par conséquence la tache de découverte de services web est devenue très difficile.

La découverte de services Web constitue un axe de recherches émergeant. Diverses approches ont été proposées, des approches logiques, non-logiques et hybrides qui combinent les deux.

Dans notre travail on a proposé un algorithme d'agrégation hybride « Condorcet flou » qui se base sur les principes de l'approche de Condorcet pour la découverte et la sélection des services web, notre proposition prend quelques paramètres au cours de processus de la découverte comme tous les approches de similarité, notre contribution utilise cinq approches de similarité : quatre méthodes de mesure de similarité non-logique et une méthode logique. Les résultats d'expérimentation montre que l'algorithme proposé est meilleur et plus efficace que les cinq approche utilisé (les approches logique et les approches non-logique), et même meilleur que des approches hybrides tel que l'approche de borda et l'approche de Condorcet.

Perspective :

- Appliquer le principe du jugement majoritaire au lieu de vote majoritaire.
- Combiner l'approche de 'possibiliste' avec l'approche de 'Condorcet flou'
- Ajouter des coefficients dans la combinaison des approches.

Référence :

[Akkiraju et al, 2005]	R. Akkiraju, J. Farrell, J. Miller, M. Nagarajan, M.T. Schmidt, A. Sheth, and K. Verma. Web service semantics - WSDL-S, November 2005.
[Ankolekar et al, 2002]	A. Ankolekar, M. Burstein, J. R. Hobbs, O. Lassila, D. Mcdermott, D. Martin, S. A. Mcilraith, M. Paolucci, T. R.Payne, and K. Sycara. Daml-s: Web service description for the semantic web. Daml s Coalition. Pages: 348-363, 2002.
[Arnaud, 2005]	Arnaud VEZAIN, Les service web - présentation générale, rapport technique, Association HERMES, Février 2005.
[Barry, 2003]	Barry Douglas K. *The Savvy Manager's Guide to Web Services and ServiceOriented Architectures*. Morgan Kaufmann Publishers Inc., San Francisco, CA,USA, 2003.
[Bernstein et al., 2005]	A. Bernstein, E. Kaufmann, C. Kiefer, and C. Bürki. SimPack: A Generic Java Library for Similiarity Measures in Ontologies. Technical report, Department of Informatics, University of Zurich, 2005.
[Bernstein et Kiefer., 2005]	A. Bernstein and C. Kiefer. iRDQL - Imprecise RDQL Queries Using Similarity Joins. In 3rd International Conference on Knowledge Capture (K-CAP), October 2005.
[Booth et al, 2004]	D. Booth, H. Haas, F. McCabe, E. Newcomer, M. Champion, C.Ferris, et D. Orchard. Web services architecture, W3C Working Group Note 11. In http://www.w3.org/TR/ws-arch/ , 2004.
[Bruijn et al, 2005]	D.Bruijn, H. Lausen, A. Polleres, D. Fensel: WSML - a Language Framework for Semantic Web Service. W3C Workshop on Rule Languages for Interoperability, Washington USA, 27–28

	April 2005. http://dip.semanticweb.org/WSMLaLanguageFrameworkforSemanticWebServices.htm.
[Chabeb, 2011]	Y. Chabeb. Contributions à la Description et la Découverte de Services Web Sémantiques. Thèse de doctorat de Télécom SudParis dans le cadre de l'école doctorale S&I en co-accréditation avec l'Université d'Evry-Val d'Essonne, Paris, France, Novembre 2011.
[Colan, 2003]	IBM.BPEL4WS (version1.1), http://www.ibm.com/developerworks/library/ws-bpel , 2003.
[Daniel et al..,2004]	Daniel Austin, Abbie Barbir, Christopher Ferris, and Sharad Garg. *Web Services Architecture Requirements*. W3C, http://www.w3.org/TR/wsa-reqs , February 2004.
[David, 2004]	David Booth, Hugo Haas, Francis Mccabe, Eric Newcomer, Michael Champion, Chris Ferris, and David Orchard. Web services architecture. Technical report, W3C, Web Services Architecture Working Group, February 2004.
[Dodani, 2004]	M-H. Dodani. From objects to services. A journey in search of component reuse nirvana. Journal of Object Technology, 2004.
[Fan et al., 2005]	J. Fan, B. Ren, and L.-R. Xiong. An Approach to Web Service Discovery Based on the Semantics. In FSKD (2), pages 1103–1106, 2005.
[Fuglede et Topsoe, 2004]	B. Fuglede and F. Topsoe. Jensen-shannon divergence and hilbert space embedding. In Information Theory, 2004. ISIT 2004. Proceedings. International Symposium on, page 31, june 2004.

[Garcia, 2006]	E. Garcia. Cosine similarity and term weight tutorial, 2006.
[Gilbert et al, 2003]	Gilbert Babin and Michel Leblanc. Les web services et leur impact sur le commerce b2b. CIRANO Burgundy Reports 2003rb-07, CIRANO, 2003.
[Gudin et al, 2003]	M. Gudgin, M. Hadley, N. Mendelsohn, J.J. Moreau, and H.F. Nielsen. Simple object access protocol (soap) 1.2. World Wide Web Consortium, http://www.w3.org/TR/soap , 2003.
[Gusfield, 1997]	D. Gusfield. Algorithms on strings, trees, and sequences: computer science and computational biology. Cambridge University Press, New York, NY, USA, 1997.
[Hadjila, 2014]	Hadjila FethAllah, Composition et interopération des services web sémantiques, thèse de doctorat, UNIVERSITE DE TLEMCEN, 2014.
[Jaeger, 2011]	Michael C. Jaeger. OWL-S Matcher.
[Jones, 1972]	K. S. Jones. A statistical interpretation of term specificity and its application in retrieval. Journal of Documentation, 28 :11 –21, 1972.
[Justin, 2002]	O'Sullivan Justin, Edmond David, and Ter Hofstede Arthur. What's in a service? *Distrib. Parallel Databases*, 12(2-3) :117–133, 2002.
[Kaufer et Klusch, 2006]	F. Kaufer and M. Klusch. WSMO-MX: A Logic Programming Based Hybrid Service Matchmaker. In European Conference on Web Services, pages 161–170, Los Alamitos, CA, USA, 2006. IEEE Computer Society.
[Keller et al., 2005]	U. Keller, R. Lara, H. Lausen, A. Polleres, and D. Fensel. Automatic location of services. In Proc. of the 2nd European Semantic Web Conference

	(ESWC), pages 1–16, Heraklion, Crete, 2005. LNCS 3532, Springer.
[Kiefer et al., 2007]	C. Kiefer, A. Bernstein, H. J. Lee, M. Klein, and M. Stocker. Semantic process retrieval with iSPARQL. In ESWC, pages 609–623, 2007.
[Kiefer et Bernstein, 2008]	C. Kiefer and A. Bernstein. The Creation and Evaluation of iSPARQL Strategies for Matchmaking. In Proceedings of the 5th European Semantic Web Conference (ESWC), volume 5021 of Lecture Notes in Computer Science, pages 463– 477. Springer-Verlag Berlin Heidelberg, 2008.
[Kiefer, 2009]	C. Kiefer. Non-Deductive Reasoning for the Semantic Web and Software Analysis. PhD thesis, University of Zurich, Department of Informatics, Zürich, Switzerland, January 2009.
[Klein et König-ries, 2004 a]	M. Klein and B. Konig-ries. Coupled Signature and Specification Matching for Automatic Service Binding. In Proc. of the European Conference on Web Services (ECOWS 2004, pages 183–197. Springer, 2004.
[Klusch et al., 2008]	M. Klusch, P. Kapahnke, and F. Kaufer. Evaluation of WSML Service Retrieval with WSMOMX. In IEEE International Conference on Web Services, 2008. ICWS ’08, pages 401 –408, Sept. 2008.
[Klusch et al., 2006]	M. Klusch, B. Fries, and K. Sycara. Automated semantic web service discovery with OWLSMX. In Proc. of the fifth international joint conference on Autonomous agents and multiagent systems, pages 915–922, New York, NY, USA, 2006. ACM Press.

[Küster et Konig-Ries, 2008]	U. Küster and B. Konig-Ries. Evaluating semantic web service matchmaking effectiveness based on graded relevance. In Proc. of the 2nd International Workshop SMR on Service Matchmaking and Resource Retrieval in the Semantic Web at the 7th International Semantic Web Conference (ISWC08), Karlsruhe, Germany, October 2008.
[Levenshtein, 1996]	V. Levenshtein. Binary Codes Capable of Correcting Deletions, Insertions and Reversals. Soviet Physics Doklady, 10: 707, 1966. 71 - A. E. Monge and C. P. Elkan. The field matching problem: Algorithms and applications. In Proceedings of the Second International Conference on Knowledge Discovery and Data Mining (KDD-96), pages 267–270, 1996.
[Martin, 2001]	Martin Gudgin, Marc Hadley, Jean-Jacques Moreau, and Henrik Frystyk Nielsen. *Simple Object Access Protocol (SOAP) Version 1.2*. W3C, http://www.w3.org/TR/2001/WD-soap12-20010709 , July 2001.
[Martin et al, 2004]	D. Martin, M. Paolucci, S. Mcilraith, M. Burstein, D. Mcdermott, D. Mcguinness, B. Parsia, T. Payne, M. Sabou, M.Solanki, N. Srinivasan, and K. Sycara. Bringing semantics to web services: The owl-s approach. Pages 26-42. Springer, 2004.
[Paolucci et al., 2002]	M. Paolucci, T. Kawamura, T. R. Payne, and K. Sycara. Semantic Matching of Web Services Capabilities. In International Semantic Web Conference (ISWC), Sardinia, Italy, 2002.
[Roman et al, 2005]	D.Roman, U.Keller, H.Lausen, J.Bruijn, R.Lara, Michael Stollberg, A.Polleres, C.Feier,

	C.Bussler, et D.Fensel. Web service modeling ontology. Appl. Ontol., 1(1) :77-106, 2005.
[Salton, 1983]	G. Salton. Introduction to Modern Information Retrieval. McGraw-Hill, New York, 1983.
[Schumacher et al., 2008]	M. Schumacher, H. Helin, and H. Schuldt. Semantic Web Service Coordination. Chapter 4, CASCOM: Intelligent Service Coordination in the Semantic Web, Birkhäuser Basel, 2008.
[Sanjiva, 2005]	Sanjiva Weerawarana, Francisco Curbera, Frank Leymann, Tony Storey, and Donald F. Ferguson. *Web Services Platform Architecture: SOAP, WSDL, WS-Policy,WS-Addressing, WS-BPEL, WS-Reliable Messaging and More*. Prentice Hall PTR,Upper Saddle River, NJ, USA, 2005.
[Srinivasan et al., 2004a]	Srinivasan N., Paolucci M., Sycara K.; An Efficient Algorithm for OWL-S Based Semantic Search in UDDI, Semantic Web Services and Web Process Composition, Springer-Verlag Berlin Heidelbery, 2004, pp96-100.
[Stollberg et al., 2007]	M. Stollberg, U. Keller, H. Lausen, and S. Heymans. Two-Phase Web Service Discovery Based on Rich Functional Descriptions. In ESWC '07: Proc. of the 4th European conference on The Semantic Web, pages 99–113, Berlin, Heidelberg, 2007. Springer-Verlag.
[Toma et al, 2005]	I. Toma, D. Fensel, M. Moran, K. Iqbal, T. Strang, and D. Roman. An Evaluation of Discovery approaches in Grid and Web services Environments. In The 2nd International Conference on Grid Service Engineering and Management, Erfurt, Germany, September 2005.
[VEZAIN, 2005]	A. VEZAIN, "*LES SERVICES WEB – Présentation générale*", Version 0.0.1,

	Association HE.R.M.E.S, Château du Montet, 2005.
[VEZAIN, 2005]	Arnaud VEZAIN. Les services web - présentation générale. Technical report, As sociation HERMES, February 2005.
[W3C, 2004 a]	W3C. OWL Ontology Web Language, 2004. http://www.w3.org/TR/owl-ref/, http://www.w3.org/TR/owlfeatures/.
[W3C, 2004 b]	W3C. Web Services Architecture. W3C Working Group Note 2004; disponible à: http://www.w3.org/TR/ws-arch/.
[Winkler et Thibaudeau, 1991]	W. E. Winkler and Y. Thibaudeau. An application of the fellegi-sunter model of record linkage to the 1990 u.s. decennial census. Technical Report Statistical Research Report Series RR91/09, U.S. Bureau of the Census, Washington, D.C., 1991.
[Zhu et al., 2010]	S. Zhu, J. Wu, and G. Xia. Top-k cosine similarity interesting pairs search. In Fuzzy Systems and Knowledge discovery (FSKD), 2010 Seventh International Conference on, volume 3, pages 1479 –1483, auguste 2010.

Table des figures

La liste des tables