N°: d'ordre 62/2023-C/IN

République Algérienne Démocratique et Populaire
Ministère de L'Enseignement Supérieur et de la Recherche Scientifique
Université des Sciences et de la Technologie Houari Boumediene
Faculté d'informatique

Thèse de doctorat

Présentée pour obtention du grade de Docteur En Informatique

Spécialité: Systèmes Informatique

Par ABDELLI Djallel Eddine

Thème

Découverte des services Web dans une architecture hybride avec optimisation multicritère

Soutenue publiquement le, 15/07/2023 devant le jury composé de:

M.	A.BELKHIR	Professeur	à l'USTHB	président
M.	F.M. BOUYAKOUB	Professeur	à l'USTHB	Directeur de thèse
M.	A. TARI	Professeur	à l'ESSTIN Bejaia	Examinateur
M	S.BITAM	Professeur	à l'U.Biskra	Examinateur
Mme	H.NACER	Professeur	à l'USTHB	Examinatrice
Mme	L.BERKANI	Maître Conference/A	à l'USTHB	Examinatrice

Acknowledgements

All praise to Allah for giving me the strength and the resilience to achieve this work.

All thanks to my parents for her unwavering support.
All thanks to my wife for living this journey with me.

I would like to express my sincere gratitude to my advisor Prof. BOUYAKOUB M'hamed Fayçal for having directed this PhD research projects. Thank you for your hard working, guidance and patience, for all the facilities, and for introducing me to such an interesting research area as Web services.

Besides my advisors, I would like to thank the rest of my thesis committee: Prof. A.BELKHIR, Prof. A.TARI, Prof. S.BITAM, Prof. H.NACER and Dr. L.BERKANI for having accepted to judge and comment this research work.

A special thanks for all teachers and workers in the faculty of computer science and the USTHB University.

I would like to thank Dr SAM Yacine and Dr Nizar and the administration form TOURS University for providing me the opportunity to join you as an exchange student in France, and for all research facilities.

Special thanks go for my colleagues Dr CHENIKI Nacer, Dr LABBACI Hamza, Dr KOMEIHA Fouad for their help and guideness.

Abstract

Service Oriented Architecture (SOA) is a modern model of programming and consuming applications, based on the concept of a service over a standardized platform.

This thesis is concerned in the web services discovery such is significant stage in the service consuming life-cycle. After a deep study of the domain, a hybrid architecture was proposed to enhance the service discovery by using the most adopted architecture for the implementation of services publishing and discovering mechanism. The proposed architecture relay on a multi-domain registry that divide the entirety of existing services into a group of domains, each domain handled by one or multiple registries.
The domain-based discovery solution is composed of two matching phases; functional matching that takes into account the functionalities of the services desired by the user and non-functional matching that allows filtering and ordering the services according to the context and quality of service information.
The distribution of services over several domains was enhanced by a deployment of a multi-agent system to take advantage of the system's task distribution instead of a unique heavy program in every registry.
For the non-functional discovery a fuzzy ranking approach was developed with multi-criteria classification to fulfil the users' requirements.

This thesis has been the subject of several contributions:

First, an evaluation tool for similarity measures was proposed. It is a simulation tool for contextual similarity measures to evaluate measures and determine their effectiveness in finding suitable services for users during the discovery process. Indeed, our objective was to give the possibility to compare and evaluate new similarities measures with existing measures.

Second, the improved multi-criteria ranking approach; This work consist on using the ranking based approach, used as a solution for the context multi-criteria matching problem, then we improved the accuracy of this approach by introducing fuzzy logic into the algorithm.

Last, we present our web services discovery hybrid architecture that allows services' providers to publish functional and non-functional information about their services and allows the users to search for services based on this information. The context utilization allows an improvement of the ranking approach used in the architecture.

Key-words: Web services, services discovery, architectures, context, multi-criteria, fuzzy logic.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

Introduction

Context and Problematic

The Web services technology aims to standardize the presentation of the services offered by an enterprise, to make access to them transparent for any type of platform and to allow interoperability between the different actors (service providers and service requesters) because of their architecture based on standard technologies.

The primary process that Web services (WS) relays on are the WS publication, WS discovery and the invocation. The goal of Web service discovery is to find an appropriate service, and this requires those services' descriptions to be published, in one or several registries, according to the adopted architecture (centralized, distributed or hybrid) [1].

The large number of service functionalities offered, as well as the variation of information related to the nature of services and users, called non-functional information, pose great challenges for both service providers and users, on tasks related to service providing and discovery in service-oriented environments.

The challenges are mainly due to everything related to (i) the invocation environment (user's surroundings, e.g. temperature, location, time, etc.), (ii) the technical characteristics of the media used to access the services (memory capacity, processor, screen size, etc.), and (iii) even the presence of user preferences and quality of service requirements.

Therefore, accessing Web services requires discovery methods that take into account the functionalities of the searched services but especially the non-functional information that influences the relevance of the discovery results to the users as they respond in a personalized way to their requirements and preferences.

(i) First the functional matching which aims at finding the desired functionalities of the searched service, then (ii) the non-functional matching to select a subset of the services resulting from the first phase in order to further refine the list of selected services, i.e., to keep only those that satisfy the user as closely as possible to his needs.

The works that has been done to enhance the discovery process is very rich in terms of the architecture implementation (centralized, distributed and hybrid) also in terms of service functionality and content (syntactic, semantic and context) search engine; than in term of services composition to fulfill the requirement that cannot be found in atomic services.

In this work, we mainly focus on providing a mechanism for service discovery in both phases, in functional discovery phase by implementing a hybrid architecture that dispatch the cloud of services into domains and sub domains depending their area of application this way will significantly decrease the search area and time. In addition, in non-functional information filtering by proposing a fuzzy multi-criteria ranking offering the best level of quality parameters between services with the same functionalities.

Objectives

This thesis interest focus on the improvement the service discovery by using the most adopted architecture for the implementation of services publication and the discovering mechanism.

To achieve this goal, set of objectives are defined as follow:

- Study closely the Web service field to put together a coherent vision of the existing solutions, their advantages and their limitations.

- Propose a hybrid architecture that take into account the limitations of the existing approaches.

- The proposed solution must provide a mechanism for modeling functional and non-functional service's information.
- The solution must provide users with a simplified approach to publish and discover services, while hiding the technical details related to the services and their use, which can complicate the user's task.
- A domain-based discovery solution consisting of two matching phases; functional matching that takes into account the functionalities of the services desired by the user, and non-functional matching that allows filtering and ordering the services according to the context and quality of service information.
- Deployment of a multi-agent system to take advantage of the system's task distribution instead of a single heavy program.

- propose a service filtering approach based on a multi-criteria method according to the user's request.

- Uses a multi-criteria ranking algorithm to rank the services based on the quantitative attributes in the context profile of the service and the user quantitative requirement.
- A fuzzy logic approach is used to overcome the problem of strict scaling of context values during classification.
- An algorithm proposed to automate the dynamic computation of average distances between contextual profiles of services used in the fuzzy approach.

- To validate our contributions several tests must be conducted, for this reason an evaluation tool is proposed to validate our contributions.

Thesis organization

This thesis is composed of a general introduction, six chapters and a general conclusion.

Chapter 1 introduces Web services. In this chapter, we start by introducing the SOA model: its principles and components. Then, we present the two traditional types of Web services (SOAP and REST). We define the latter and we present their characteristics and their standards.

Chapter 2 presents a state of the art on Web services discovery. This chapter presents the technologies that allow the realization of functional and non-functional discovery of Web services, also the approaches that implement these technologies.

Chapter 3 is a state of the art of service discovery architectures. We present in this chapter the different approaches to service discovery in a centralized, distributed and hybrid environment. We first introduce the different techniques used in the service-matching phase. A detailed study of the different service discovery approaches is presented in each environment followed by a critical discussion presenting the advantage also the limitations of some approaches and then finishing the chapter with a descriptive table with a synthesis and conclusion.

Chapter 4, 5 and 6 presents our contributions to Web services discovery. In these chapters, we first present the motivations of these contributions. We describe in detail the elements of each contribution.

First, the evaluation tool for the similarity measure: describe the system functions and modeling, the used dataset, interfaces and operations, testing and implementation and finally the conclusion and perspectives.

Second, the improved multi-criteria ranking approach where we describe the used algorithm and how it is adopted to our problem, the introduction of the fuzzy logic in the algorithm and its impact. The test and results to validate the approach, lastly, the conclusion and perspective of the approach.

Last, we present our Web services discovery hybrid architecture that allows service providers to publish functional and non-functional information about their services, and users to search for services based on these information. We start by introducing the background and the related works followed by a deep review and discussion. Then, we present the architecture by describing each component and every possible scenario. Furthermore, we describe the context utilization and the improvement of the ranking approach used in the architecture. Finally, we present the tests and evaluation of the ranking approach implemented in the hybrid architecture.

The last chapter, chapter 7 is a general conclusion and the perspective of this thesis.

Chapter 1: Introduction to Web services technology

1.1 Introduction
1.2 SOA (Service Oriented Architecture)
1.3 Web Service technology
1.4 Conclusion.

1.1 Introduction

With the emergence of the internet and the interconnection of machines, it becomes possible to interact with applications on remote machines.
The evolution of this phenomenon went through several stages, firstly the human-machine interaction then the machine-machine interaction.
This evolution has opened a new research area and risen a whole new generation of technologies based on services known as Service Oriented Architecture.

1.2. SOA (Service Oriented Architecture)

From the beginning of the 1980 to the 1990, it has occurred a new problem which is the interoperability in information systems as consequence of coupling different company's information systems. [1]
By the 1992 other functional requirement was imposed with the invention of meditation[1] architecture by Geo Wiederhold. [2]
In 1996 the arrival of the XML has given for researchers a new aspect to solve the arising Web technologies problems.
When second millennium (2000) come many companies were involved in developing Service Oriented Architecture technology such IBM, Microsoft, Sun, Cisco, Google and so on; providing an important number of solution in this field.

Service Oriented Architecture (SOA) is a modern model of programming and consuming applications, based on the concept of a service over a distributed platform.
Service based applications represent a repeatable (reusable) business activity provided over a network, it has a specified pattern (inputs, outputs…), it may be composed of other services and can be inquired without human interaction (Application to application interaction).

Other simple definitions are set for the SOA like "SOA is a loosely-coupled architecture designed to meet the business needs of the Web organizations".

We can distinguish two main architectures in SOA, the Remote procedure calling (RPC)[2] which is a distributed component architecture where the aspect of service is all most absent, and we find the Web services.
The difference between these main architectures is that the Web services are independent form the programming languages and the platforms unlike the RPC methods. Also in RPC, the clients must exchange manually the invocation details (the interfaces) otherwise the Web services invocation details are published publicly upon the discovery registries.

Different RPC platforms exists such JAVA RMI, Microsoft DCOM and .NET Remoting and COBRA object request broker.

The main actors in service-oriented architecture are the client and the provider of the service. The provider is the person who create the service and publish it; and the client is the service

[1] Meditation: refer to the dynamic interface function.
[2] RPC: Remote Procedure Calling (for the Oriented Object programming it become RMI: Remote Method Invocation);
We note that in RPCs exist a way enabling communication between software components that do not share one language called IDL (interface description language).

consumer who search the services using various registries and then binds with the service provider in order to invoke one of its services

The dependency advantage for the Web services lays on the XML technologies. XML has privileged the appearance of Web services over the component concept, but component concept stay always primary and Web services depend on these components.

1.3. The Web Services technology

In 1980, Tim Berners-Lee presented his Web vision: as simple human-to-machine interaction with graphics and hypertext data. During the Web community development, a new way of exploiting the Web have occurred: it is the machine-machine communication.
The need of a standard mechanism to establish a machine-to-machine interaction has given birth to a novel concept that is the Web service technologies.

In 1994, Tim founded the World Wide Web Consortium (W3C), an international community devoted to developing open Web standards. In 2002 the foundation lunch the Web services activity after that the Web Services Architecture, Description and Coordination Groups have been created.

In 2004, the W3C set a definition for Web services (WS) like "WS is a software system designed to support interoperable machine-to-machine interaction over a network. It has an interface described in a machine-process able format (specifically Web Service Description Language WSDL). Other systems interact with the Web service in a manner prescribed by its description using SOAP[3]-messages, typically conveyed using Hyper Text Transfer Protocol (HTTP) with an XML serialization in conjunction with other Web-related standards". [3] [4]

The primary process that Web services relays on are: the WS description, WS discovery and the invocation.

-WS description: describing the Web service interface (the in/out parameters, data types and format, location, communication protocols). The normalized format used by the W3C to describe Web services is WSDL description language that is an XML based notation.

-WS discovery: the Web service discovery is to search and locate a Web service satisfying the requester. Different approaches are used, like syntactic approach, semantic approach and finally contextual approach.

-WS invocation: After locating the Web service, the next step is to invoke it. Two methods exist: stub based remote calling approach and XML document based calling.

WS provider publishes a description of the service in the UDDI[4] registry, as well as details about the use of the service. Providers use WSDL, an XML-based language, to describe the WS, its location and its operations (or methods).

[3] SOAP: Simple Object Access Protocol (will be defined in the next sections)
[4] UDDI: Universal Description, Discovery and Integration

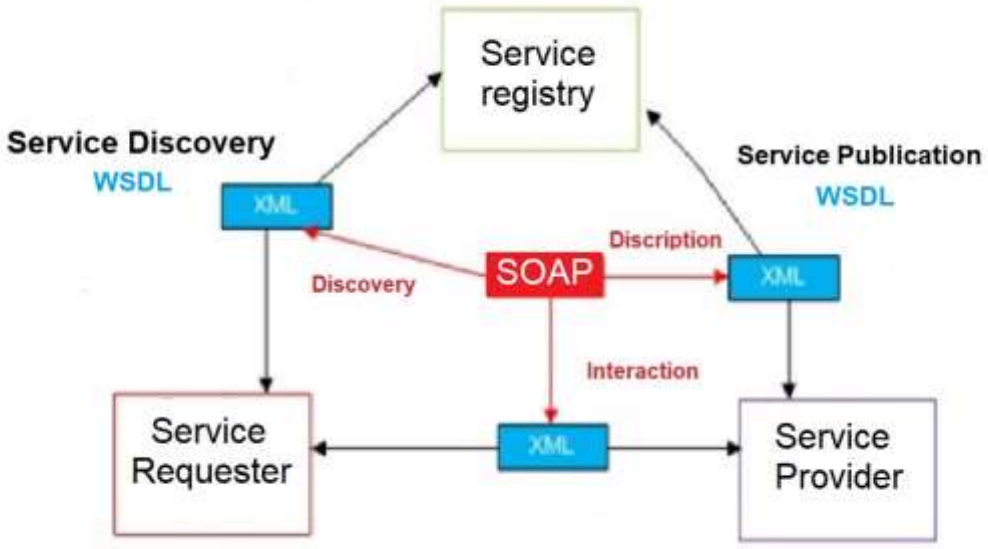

Figure 1 Web services (WS) architecture.

The WS consumer uses UDDI to discover appropriate services that meet his/her requirements, chooses one service and invokes it.
The presented WS model, allows the interaction between three actors: the service consumer (client), the service provider and the service discovery system. This interaction is made possible thanks to three functions: the publishing function, the discovery function and the binding function.
The publishing function allows providers to publish description of their services in order to make them available for client, in the UDDI registry.
The discovery system allows client to search, in the UDDI registry, for services according to their requirements.
The binding function: when the discovery system presents to the user the list of discovered services, he invokes one of these services. The client and the provider will use SOAP protocol to communicate.
The role of the discovery process is very important in the Web service model, it search and locate services by matching service descriptions with service requests.

The service description provides information about the service. This description can be advertised by a provider and used during the discovery process.
When a client search for a service, the discovery process retrieves services satisfying the functionality requested by the client.

1.3.1 Web Services Description Language (WSDL)

WSDL was developed by IBM and Microsoft and Ariba, and accepted and published as a notice on W3C site in March 2001. By June 2007 the second version 2.0 of WSDL was referenced in W3C.
As communications protocols and message formats are standardized in the Web community, it becomes increasingly possible and important to be able to describe the communications in some structured way. WSDL addresses this need by defining an XML grammar for describing network services as collections of communication endpoints capable of exchanging messages.

WSDL service definitions provide documentation for distributed systems and serve as a recipe for automating the details involved in applications communication.

A WSDL document defines **services** as collections of network endpoints, or **ports**. In WSDL, the abstract definition of endpoints and messages is separated from their concrete network deployment or data format bindings. This allows the reuse of abstract definitions: **messages**, which are abstract descriptions of the data being exchanged, and **port types** which are abstract collections of **operations**. The concrete protocol and data format specifications for a particular port type constitutes a reusable **binding**. A port is defined by associating a network address with a reusable binding, and a collection of ports define a service.

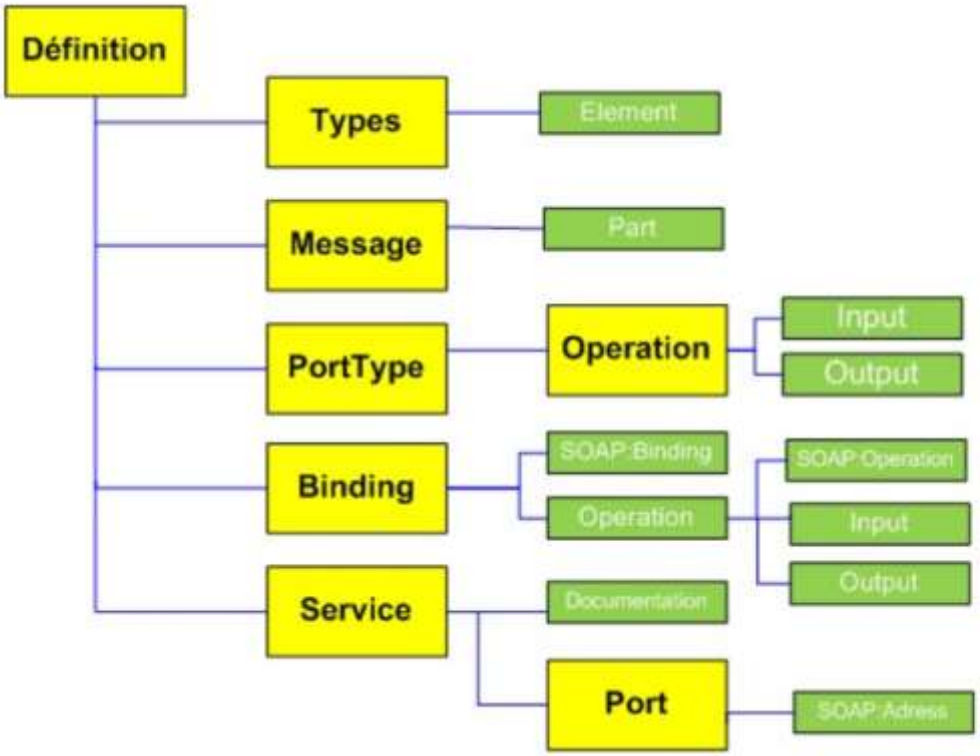

Figure 2 WSDL document structure

Hence, a WSDL document uses the following elements in the definition of network services:

- **Types**– a container for data type definitions using some type system (such as XSD).
- **Message**– an abstract, typed definition of the data being communicated.
- **Operation**– an abstract description of an action supported by the service.
- **Port Type**–an abstract set of operations supported by one or more endpoints.
- **Binding**– a concrete protocol and data format specification for a particular port type.
- **Port**– a single endpoint defined as a combination of a binding and a network address.
- **Service**– a collection of related endpoints.

It is important to observe that WSDL does not introduce a new type definition language. WSDL recognizes the need for rich type systems for describing message formats, and supports the XML Schemas specification (XSD) as its canonical type system.
However, since it is unreasonable to expect a single type system grammar to be used to describe all message formats, the present one and future format, WSDL allows using other type definition languages via extensibility.
In addition, WSDL defines a common **binding** mechanism. This is used to attach a specific protocol or data format or structure to an abstract message, operation, or endpoint. It allows the reuse of abstract definitions. [5]

The following example illustrates the type and message definitions for a Skateboots.com purchase order service that returns the total value of one or more purchase orders[5]. The XML schema data types used in the WSDL file are mapped to messages using the schema element names.

```
<types>
<schema targetNamespace="PurchaseOrderService-xsd" xmlns= "http://www.w3.org/2001/
XMLSchema" xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/">

<complexType name="PurchaseOrder">
<all>
<element name="Compan yName" type="xsd:string"/>
<element name="Items" type="xsd1:ArrayOfItem"/>
<element name="Address" type="xsd1:Address"/>
</all>
</complexType>

<complexType name="Item">
<all>
<element name="Price" type="xsd:float"/>
<element name="PartID" type="xsd:string"/>
<element name="Description" type="xsd:string"/>
<element name="Quantity" type="xsd:int"/>
</all>
</complexType>

<complexType name="Address">
<all>
<element name="State" type="xsd:string"/>
<element name="PostalCode" type="xsd:string"/>
<element name="City" type="xsd:string"/>
<element name="Line2" type="xsd:string"/>
<element name="Country" type="xsd:string"/>
<element name="Line1" type="xsd:string"/>
</all>
</complexType>
</schema>
</types>
```

The following example illustrates a request/response operation definition for the Skateboots.com service.

```
<portType name="PurchaseOrderPortType">
<operation name="postPurchaseOrder">
<input message="tns:postPurchaseOrderRequest" name="postPurchaseOrder"/>
<output message="tns:postPurchaseOrderResult" name="postPurchaseOrderResult"/>
</operation>
```

[5] This is an example from the book : Understanding Web Services: XML, WSDL, SOAP, and UDDI [6]

```
<operation name="postPurchaseOrders">
<input message="tns:postPurchaseOrdersRequest" name="postPurchaseOrders"/>
<output message="tns:postPurchaseOrdersResult" name="postPurchaseOrdersResult"/>
</operation>
</portType>
```

The following example illustrates the service binding for the Skateboots.com purchase order totaling service [6]:

```
<service name="PurchaseOrderService">
<port binding="tns:PurchaseOrderBinding" name="PurchaseOrderPort">
<soap:address location="http://localhost:9000/ xmlbus/contain er/PurchaseOrder/
PurchaseOrderService/ PurchaseOrderPort"/>
</port>
</service>
```

1.3.2 Universal Description, Discovery and Integration (UDDI) for registration and publication

The Universal Description, Discovery, and Integration (UDDI), defines a standard method for publishing and discovering the network-based software components in a service-oriented architecture (SOA).

1.3.2.1. UDDI in the Web Services

The specification defines a group of Web services and programmatic interfaces for publishing, retrieving, and managing information about services. UDDI builds upon several other established industry standards, including SOAP, HTTP, WSDL, XML and XML Schema (XSD).

UDDI has two main parts: registration and discovery. The registration part means that businesses can post information to UDDI that other businesses can search for and discover, which is the discovery part.
Businesses and individuals interact with UDDI by using SOAP APIs or one of the user interfaces provided by the operators or other Web services vendors. UDDI operators post WSDL descriptions of their Web services for registration and discovery. UDDI provides separate WSDL files for registration and discovery services, using its own XML document format.

The data structure of UDDI is expressed using complex types in XML schemas. These schemas allow for extensibility and great flexibility in the data stored for a particular business or entity.

1.3.2.2. Information structures in UDDIs

UDDI information is often described as being divided into three main categories of business information:
White Pages: Business name and address, contact information, Web site name, and Data Universal Numbering System (DUNS) or other identifying number.

Yellow Pages: Type of business, location, and products, including various categorization taxonomies for geo-graphical location, industry type, business ID, and so on.
Green Pages: Technical information about business services, such as how to interact with them, business process definitions, and so on. A pointer to the business's WSDL file, if any, would be placed here. Information in this category describes a service's features/functionality, including a unique ID for the service. This category is quite new and specific to the Internet. [6]

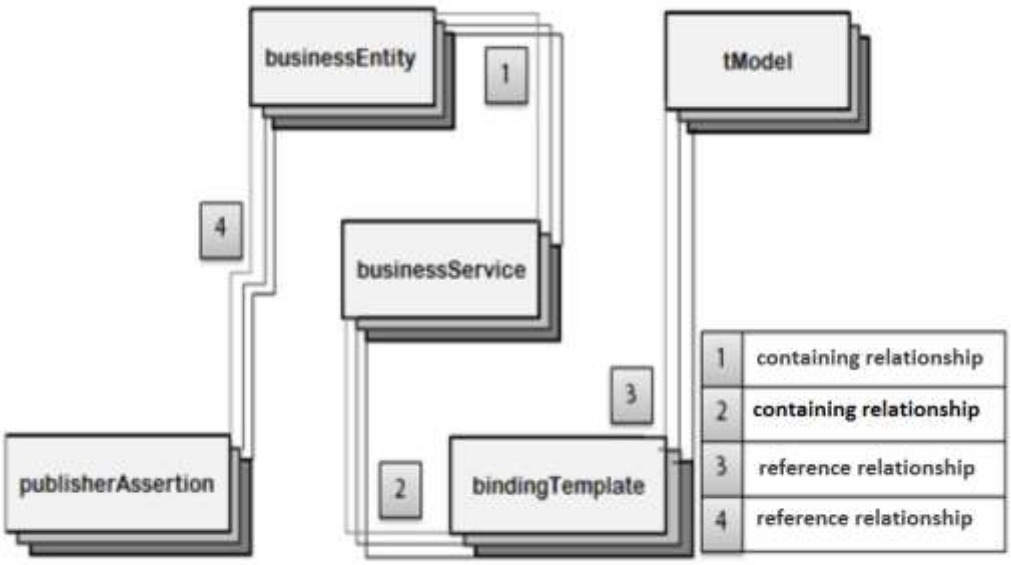

Figure 3 The relation between Data structure in the UDDI

1.3.2.3. The UDDI registries architectures

-Centralized registries: in this architecture, WS registry entries are contained within a single 'well known' central entity used by all the providers.

-Distributed registries: in this architecture, services' descriptions are localized in several distributed registries. This architecture support modern environments like peer-to-peer (P2P) networks

-The third class is hybrid registries: in this class, registry information is distributed among multiple registries in a P2P manner; however the access to the registry information is done via dedicated nodes. This hybrid architecture is transparent to users and this appears as a centralized architecture.

Accessing to the UDDI registries can be establish in two ways:

- Throw a Web navigator who communicate with a specific Web application conceived as a registry interface.
- Or by program using the APIs (Application Programming interface) which has two main functions:
  - Inquiry API: for navigation, search, and consulting information from the directory.
  - Publishing API: for publishing, creating, and modifying or deleting information from the directory.

1.3.2.4 Usage Scenario

In this section we will give a real example for the Web services deployment and consumption:

Imagine that the Skate-boots Company wants to register its contact information, service description, and online service access information with UDDI. The following steps are necessary:

1. Choose an operator with which to work. Each operator has different terms and conditions for authorizing access to its replica of the registry and may provide some value-added services on top of the basic UDDI services. Also, whenever the company need to update or to modify the data they have registered, they have to go back to the operator with which they entered the data.
2. Build or otherwise obtain a UDDI client, such as those provided by the operators.
3. Obtain an authentication token from the operator.
4. Register information about the business. Include as much information as might be helpful to those searching for matches.
5. Release the authentication token.
6. Use the inquiry APIs to test the retrieval of the information, including binding template information, to ensure that someone who obtains it can use it successfully to interact with your service.
7. Fill in the tModel information in case someone wants to search for a given service and find your business as one of the service providers.
8. Update the information as necessary to reflect changing business contact information and new service details, obtaining and releasing a new authentication token from the operator each time.

This example illustrates a SOAP message requesting to register a UDDI business entity for Skate-boots Company.

```
POST /save_business HTTP/1.1 Host: www.skateboots.com Content-Type: text/xml; charset="utf -8"
Content-Length: nnnn SOAPAction: "save_business" <?xml version="1.0" encoding="UTF-8" ?>
<Envelope xmlns="http://schemas/xmlsoap.org/soap/envelope/">
 <Body>
<save_business generic="2.0" x mlns="urn:uddi -org:api_v2">
<businessKey="">
</businessKey>
<name>
 Skateboots, Inc.
</name>
<description>
You know, like the ones in Batman and Robin . . .
</description>
<identifierBag>
...
</identifierBag>
...
</save_business>
</Body>
</Envelope>
```

The next step is a sample SOAP request to obtain business detail information about the Skate-boots Company.

```
POST /get_businessDetail HTTP/1.1 Host: www.skateboots.com Content-Type: text/xml; charset="utf -8"
Content-Length: nnnn
SOAPAction: "get_businessDetail"
<?xml version="1.0" encoding="UTF-8" ?>
<Envelope xmlns="http://schemas/xmlsoap.org/soap/envelope/">
 <Body>
<get_businessDetail generic="2.0" xmlns="urn:uddi org:api_v2">
<businessKey="C90D731D -777D-4130-9DE3-5303371170C2">
</businessKey>
</get_businessDetail>
</Body>
</Envelope>
```

This is a sample of message that might be returned from any of the UDDI operators on receipt of the find_businessrequest [6].

```
<businessList generic="2.0 " operator="uddi.sourceOperator" truncated ="false" xmlns:="urn:uddi-org:api_v2">
 <businessInfos>
<businessInfo businessKey="C90D731D-777D-4130-9DE3-5303371170C2">
<name>
Skateboots Inc.
</name>
<serviceInfos>
<serviceInfoserviceKey="D90D731D-777D-4130- 9DE3-...">
<name>
PreSeason Orders
</name>
</serviceInfo>
<serviceInfoserviceKey="E90D731D-777D-4130- 9DE3-...">
<name>
In-season Orders
</name>
 </serviceInfo>
</serviceInfos>
</businessInfo>
</businessInfo>
 ... [more Skateboot manufacturers]
</businessInfos>
</businessList>
```

1.3.3. Simple Object Access Protocol (SOAP) Web services

UserLand, DevelopMentor and Microsoft developed SOAP, it was published on the Web in late 1999. SOAP is an extension of the XML -RPC specification. Originally defined by Dave

Winer[6] of UserLand. Implementations of XML-RPC predate SOAP and continue to be used, but SOAP has gained a wider acceptance because of its additional features and functionality.

SOAP is a lightweight protocol for exchange of information in a decentralized and distributed environment. It is an XML based protocol composed of three parts: an envelope that defines a framework for describing what is in a message and how to process it, a set of encoding rules for expressing instances of application-defined datatypes, and a convention for representing remote procedure calls and responses. SOAP can potentially be used i with a variety of other protocols [7].
SOAP transports an XML document across the Web and, potentially, across other types of networks, to reach a Web service implementation. A SOAP message is, in fact, an XML document created in a specific format.
The SOAP specification defines a layered approach to the protocol so that it can be used in combination with other transports such as the OMG Internet Inter-Orb Protocol (IIOP), Java Messaging System (JMS), and the IETF Blocks Environment Extensible Protocol (BEEP). However, the HTTP is the common used. [6]

We note that the SOAP 1.2 XML-based messaging framework became a W3C recommendation in June 2003. [4]
A SOAP message is an XML document that consists of a mandatory SOAP envelope, an optional SOAP header, and a mandatory SOAP body. This XML document is referred to as a SOAP message for the rest of this document.

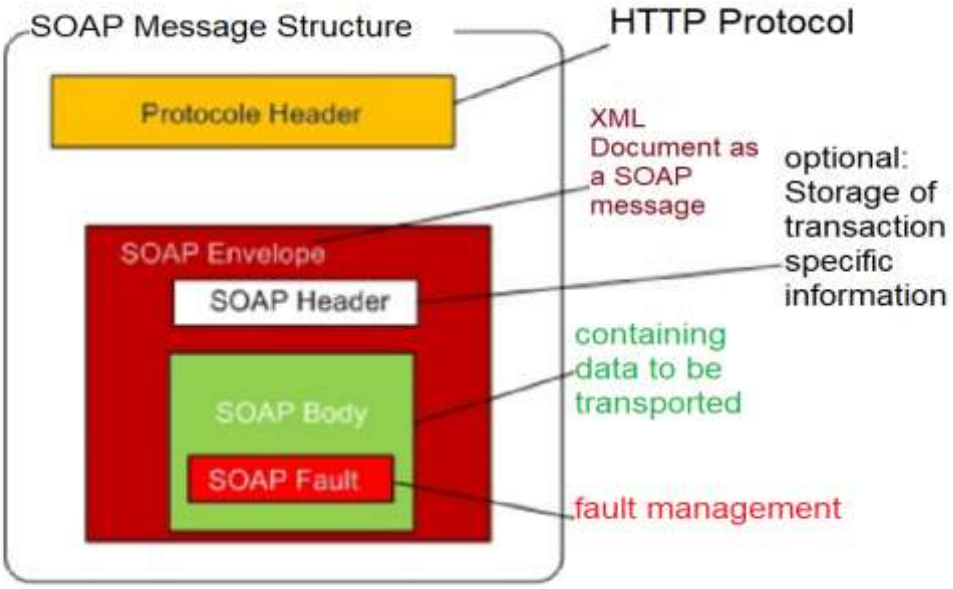

Figure 4 SOAP message structure

A SOAP message contains the following elements:
- The Envelope is the top element of the XML document representing the message.
- The Header is a generic mechanism for adding features to a SOAP message in a decentralized manner without prior agreement between the communicating parties. SOAP

[6] Dave winer: Dave Winer is an American software developer, entrepreneur, and writer that resides in New York City. Winer is noted for his contributions to outliners, scripting, content management, and web services, as well as blogging and podcasting.

defines a few attributes that can be used to indicate who should deal with a feature and whether it is optional or mandatory.

- The Body is a container for mandatory information intended for the ultimate recipient of the message. SOAP defines one element for the body, which is the Fault element used for reporting errors [7].

The following example shows a simple application of SOAP, a one-way broadcast message to a list of communication mechanisms. [6]

```
<env:Envelopexmlns:env="http://www.w3.org/2001/12/soap-envelope">
<env:Header >
<n:broadcastService xmlns:n="http://www.xmlbus.com/broadcastServices">
<n:list>
PDA, Cell, Email, VoiceMail, IM
</n:list>
</n:broadcastService>
</env:Header>
<env:Body>
<m:Function xmlns:m="http://www.xmlbus.com/broadcastServices/send">
<m:message>
Eric, you are late for the concall again!
</m:message>
</m:Function>
</env:Body>
</env:Envelope>
```

1.4. Conclusion

In this chapter, we have presented generalities about SOA and Web services that provides flexibility for interoperability and integration through the use of three standard protocols: SOAP, WSDL, and UDDI. Aiming to get a clear view on the concepts of this domain.
In the next chapter, we go further in describing the fundamental registries architectures and discovery approaches.

The Stat of the art

Chapter 2: Web service discovery

2.1 Introduction
2.2 Web Service Discovery
2.3 Web services descriptions
2.3.1 Functional descriptions of Web services
2.3.1.1 Syntactic description of Web services
2.3.1.2 Semantic descriptions of Web services
2.3.2 Non-functional descriptions discovery and multi-criteria optimization
2.4 Discussion

2.1. Introduction

We have seen previously the SOA architecture (especially WS) and its main functions and parts like UDDI [6], WS discovery and description.
The success of WS allowed the adoption of this technology by various providers, which induced an increase number of services, making their discovery a complicated task.
The descriptions of services are published in registries conceived specially for this function (UDDI registries). The aim of these registries is to facilitate the discovery of services.
The approaches based on UDDI was limited by the syntactic search, the weakness of key word based approach has given the chance for a new semantic solution which has a rich and more precise description and search result. However, this was not able to cover the nonfunctional aspect in the WS description, so more work was established in this area, the proposed solutions are contextual approach.
The discovery mechanism become an important research subject, in this chapter we present the discovery approaches in comparative and critical vision.

2.2. Web service discovery

A first description of discovery mechanisms for service providers appears as the matchmaking process of the client requirement and the existing services.
Decker, and al. [8] define It by "the process of finding an appropriate service provider for a service requester through a middle agent" . A more up-to-date approach defines the WS Discovery mechanism in a broader sense as "the act of locating a machine-exploitable description of a WS that may have been previously unknown and that meets certain functional criteria." It is a service responsible of performing discovery, a logical role, which could be performed by either the requester agent, the provider agent or some other agents [9].
Malaimalavathani [10], describes the Web service discovery by "a process of discovering services that are most suitable to user's request according to requester's requirement. Discovery is one of the major challenges of the Web service technology. An effective and automated search and selection of relevant services is necessary for both human users and programs"**.**

Generally, Web service discovery is based on terms such as request, keyword, matching or mapping (figure 5):
• **Request**: generally a request is addressed by a user to a search engine, during a session, with one or more keywords.
• **Keyword**: A keyword is a succession of characters not containing blanks and appearing in the field reserved for this purpose in the request. Operators (characters used to make complex expressions) are not considered as keywords.
• **Matching**: The matching is a mechanism which aims to find semantic similarities, and possibly structural similarities, between two information: the required information (necessary) and the provided information by the system (published) [11], using a matching algorithm. The matching algorithm compares all the announcements (descriptions of published services) with the request, and provides as a result, services, which are close to the request.
• **Mapping**: The mapping consists in finding semantic relations, which allow the transition from an entity to another. That is why we must look for correspondences to establish transformations between objects having comparable nature but not having the same form. Consequently, the mapping uses the matching results to carry out the possible transformations of the objects.

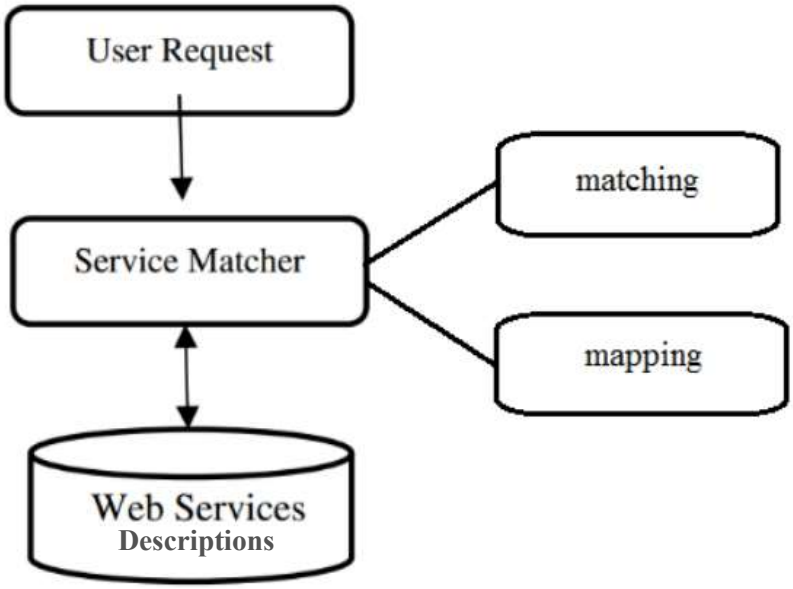

Figure 5 Web discovery diagram

In literature, many approaches for WS discovery were proposed. Searching for services in UDDI is based on keyword matching which is not efficient because huge number of WS may match a keyword and it is difficult to find the best one. Other approaches take advantage of semantic Web concept, where WS matching is done using ontologies.
When the user sends his/her request, the discovery system will search for WS, according to the client requirements. The performance of the discovery system depends on how mature WS matching process. In other words, how requirements of user are represented and how they are matched with available services [12].

2.3. Web services descriptions

Web services descriptions is a formal way to characterize a service information such it functionality, the service provider information (data flowing in and out), the service requirement, pattern, and other useful information.
A service description plays an important role to facilitate the discovery of the service and enabling access to it. Most description languages are designed for machine consumption and manage information about how to invoke the Web service and the expected effect of using it.
Web service description is divided in tow main classes: functional parameters class and non-functional parameters class[3]**.**

2.3.1. Functional descriptions of Web services

The first model for describing Web services functional information was the IR (Information Retrieval) based syntactic approach, then for some requirement, a new aspect of description model has occurred in semantic approach.

2.3.1.1 Syntactic description of Web services:

The principle of the syntactic approaches services descriptions is based on key word

comparison between the user query and the Web services descriptions that are almost XML based.
We have many approaches which has syntactic based discovery mechanism like **UDDI** registries and the **AASDU** system (Agent Approach for Service Discovery and Utilization) proposed in [13]**.** The proposed solution is a multi-agent approach for the discovery of Web services. AASDU system contains four components: a *GUI* (Graphical User Interface) to express the required parameters in a string via the interface, a *QAA* (Q*uery Analyzer Agent*) that analyzes the query sent by the GUI and a *Multi-Agent Referral System* (which is a repository system containing areas of expertise service agents and service module).
The query sent by the user is expressed as a string via a GUI. It is then sent to the QAA. This agent extracts the relevant keywords it uses subsequently, to select other experts agents from the repository system areas of expertise service agents. To do this, it performs a syntactic matching on a simple variant of the technical TFIDF (Term Frequency Inverse Document Frequency) [14]. Selected experts agents, transmit then the parameters for services in their area of expertise, to a composition agent. It invokes one-candidate services as selected by the user or made some of them to respond to his request.

DSD (Document Structure Description) matchmaker is also a syntactic approach [15], which uses graphs for matching service description graphs. This approach is using the object-oriented language (Diane Service Description) for service description [16], that specifies variables and sets of declarative objects without any logic-based semantics. The matching process determines from a graph the variables to satisfy and selects, based upon the services' status, the one that better responds to the query from the set of discovered services. Than returns a numerical value representing the corresponding matching degree. To perform the process, DSD matchmaker runs two parallel descriptions (the current offer description and the query description) as trees and recursively compares the nodes of the two graphs [17] [18].

In spite of its simplicity and its facility of implementation, this approach presents some limitations.

2.3.1.2 Semantic descriptions of Web services:

The large scale of Web services and their descriptions published on the network makes the discovery of relevant services an ambiguous task for the humans and for the machines also, since the services can be interoperated by other applications (machine).
In this discovery class (semantic approaches), researchers focused on the semantic aspect (Logical) of services. This development is increasingly significant since it seems to be able to fill certain insufficiencies of syntactic approaches.
In semantic Web discovery approaches, computers must have access to structured collections of information and sets of inference rules that they can use to conduct automated reasoning.
Two important technologies for developing the Semantic Web are already in place: eXtensible Markup Language (XML) and the Resource Description Framework (RDF).
XML allows users to add arbitrary structure to their documents but says nothing about what the structures mean. The meaning is expressed by RDF, which encodes it in sets of triples, each triple being rather like the subject, verb and object of an elementary sentence [19]**.**

The Resource Description Framework is an infrastructure that enables the encoding, exchange and reuse of structured metadata. RDF is an application of XML that imposes needed structural constraints to provide unambiguous methods of expressing semantics. RDF additionally provides a means for publishing both human-readable and machine-processable

vocabularies[20].

In this section, we present some semantic based approaches.

OWL-S [21] is an OWL (Ontology Web Language) based Web service ontology, which supplies Web service providers with a core set of markup language constructs for describing the properties and capabilities of their Web services in unambiguous, computer-interpretable form. OWL-S markup Web services to facilitate the automation of Web service tasks, including automated Web service discovery, execution, composition and interoperation. Following the layered approach to markup language development, the current version of OWL-S builds on the (OWL) recommendation proposed by the Web-Ontology Working Group at the World Wide Web Consortium**.**

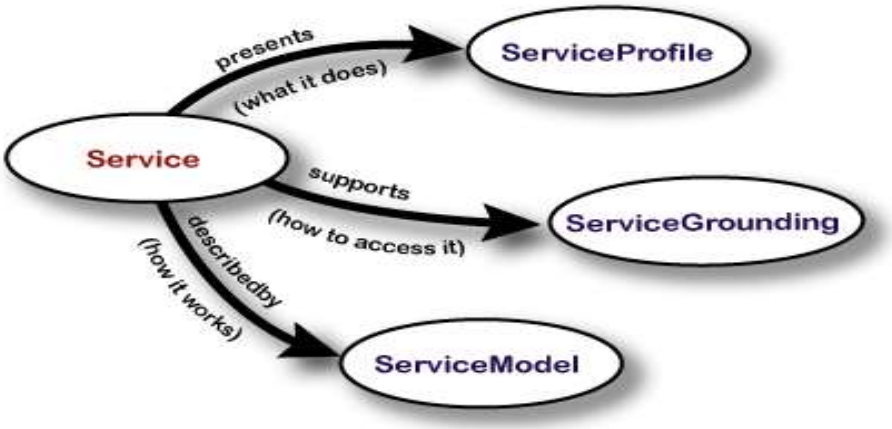

Figure 6 owl-s service model

Jaeger and al., present in [22] an OWL-S Matchmaker based on IOPE-matching (inputs, outputs, preconditions and effects) . The matchmaker researches for semantic mappings between the functional parameters defined in the WS OWL-S descriptions and the parameters introduced in the query. The matching process involves four tasks: inputs matching, outputs matching, service categories matching and a fourth task during which constraints and pre-defined features are applied by the user. The algorithm computes the matching degree in each of the first three tasks and performs the fourth one to finally aggregate the results and return back the rank of the matched WS whether it is selected as a service that meets the query. The ranking classes the service according to its matching degree among all discovered WS [18]**.**

Vu and al., have proposed in [23] a semantic WS discovery approach based on P2P (Pear to Pear) network called **PSWSD** (P2P-based Semantic Web Service Discovery). This approach uses a deductive architecture for WS discovery in P2P network. WS descriptions are provided according to the WSMO (Web Service Modeling Ontology) using specific techniques and are published in various registries distributed in the P2P network. A user seeking for a WS, with specific QoS (quality of service) constraints can address his query to any registry in the network. The one that received the query forwards it to the registries that can satisfy it. The functionality of the requested service are then extracted from the query and sent to the matchmaker module. The matchmaker selects services descriptions which match semantically with the user query and sends the results to the user to choose the service to invoke [18].

CASD [24] (Context Aware Service Discovery) is a further service discovery proposal taking the context into account. The discovery module uses domain ontology to determine the services' categories that have a semantic relationship with the user query.

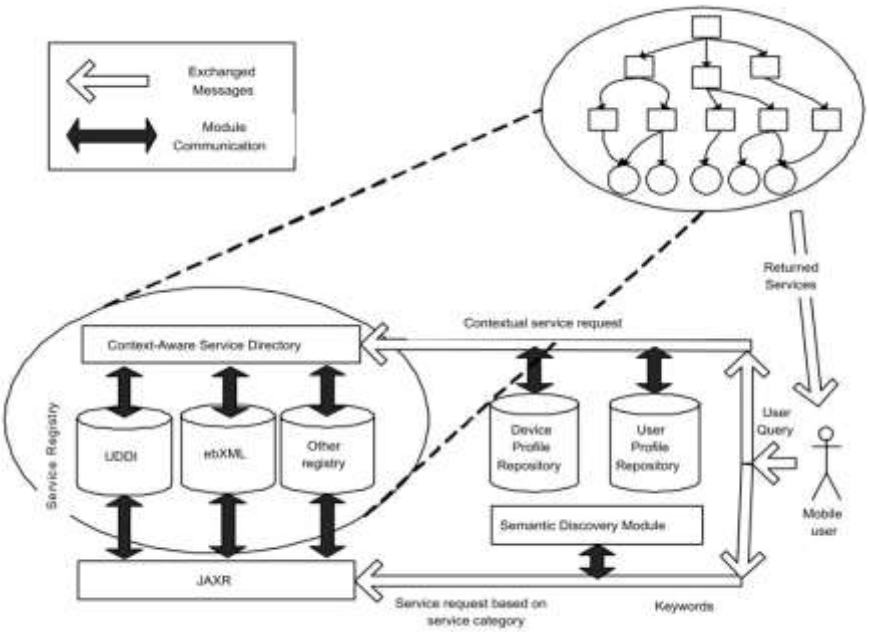

Figure 7 CASD architecture.

When first the user makes his query (Qusr) in terms of keywords, another richer query (Qctx) is generated and attached to it. Quick query allows finding WS that have a semantic relationship with the user research terms, while the Qctx query acts as a filter to select the services that match the user's context. Queries made by users are expressed in SPARQL (Simple Protocol and RDF Query Language) which is a query language compatible with OWL graphs. The distributed agent architecture for service discovery DCASDAA (Distributed Context Aware Service Discovery Agent architecture) is the distributed version of CASD approach it was proposed by Schulzrinne and Arabshian [25] [18]**.**

The semantic approaches bring the logical processing advantages for the service discovery field. However, still the service discovery needs other information process regarding the service environment and capacities.

2.3.2. Non-functional descriptions of Web services Approaches

When a person offers a service to another person it creates a kind of interaction, which can be developed throw gathering some information from each other. This ability cannot be used in human machine interaction with classic services. In this way, the context processing improves the human machine interaction and communication.

Dey[7], defines context as «all information being able to be used to characterize the situation of an entity, where an entity is a person, a place, or an object who can be relevant for the interaction between the user and the application, including the user and the application themselves» [26].

The user context is defined by a set of characteristics that are related either to the user or to the device used by the user, or to the environmental conditions in which the user and the device are connected. We can define these three general contexts as follows [27]:

- Context related to the device: a user can access a service through different types of devices: smartphone, computer, tablet. This type of context describes the specific characteristics of the access devices (hardware or software) for example storage capacity, screen size, type of networks, etc.
- Environmental context: describes the environmental conditions in which the user and the devices are connected. This information such as location, temperature, lighting, etc. can be provided by sensors or other devices. It can be provided by sensors or by other services.
- User context: a user context is associated with a session. A session is the act of connection by the user to the system. The dynamic characteristics of the user can evolve either from one session to another or during the same session. This dynamic characteristic can vary during the same session if the user has a mobile device as access terminal. An example of the dynamic context is the user's location.

The term QoS stands for Quality of Service. In the field of communication networks, QoS attributes are generally data transmission delay, throughput, jitter, packet loss rate, etc.

In the context of service-oriented computing, QoS is considered at the application level, even we also find some QoS attributes related to the network.

Therefore, we need to distinguish between application-related and network-related QoS attributes, such as the packet delivery delay parameter, etc. These parameters can be classified, according to their measurement, into three categories [28]:
1. Metrics announced by the provider: these are the metrics whose values have been announced by the service provider at the time of its publication. Example: the cost of a service.
2. Metrics evaluated by the consumer: these are the metrics whose values have been given by the users as an evaluation value of the services after usage experiences. Example: reputation.
3. Observable metrics: these are metrics whose values have been obtained through monitoring or testing. Example: response time or availability.

To describe context information, different languages and models can be used. We distinguish between those languages/models for modeling and implementation. For a given system, a language can be used to model representation of the context information (modeling phase),

[7] Anind K. Dey College of computing & GVU center, Georgia institute of technology, Atlanta, GA, USA.

while in the implementation of the system, concrete, instance context information might be described by the same or another representation. One example is to use UML to model the context information but the instance data is described in XML [29].

In order to provide application dependent context information through a context framework, a uniform way of representing and sharing context is required.
A context model provides an unambiguous definition of the context elements, their representations, semantics and usage. It takes into account the general characteristics of context information, such as its temporal nature, ambiguity, impreciseness, incompleteness and privacy. Furthermore, a context model must also address the special requirements of pervasive computing environments like distribution, mobility, heterogeneity of context sources and resource-constrained device [30].

The data structures used for representing and exchanging context information are: key-value pair, markup scheme, graphical, object oriented, logic based and ontology based models [31].

The PPDL (pervasive profile description language) is an XML-based language that can be used to describe preferences of peers within situations peers operate [32].

Some XML schemas are defined for describing contextual information of mobile networks, for example, activity, device status and reachability [33].

CC/PP (Composite Capability/Preference Profile) can be used to describe contextual information of service capabilities and user preferences. CC/PP is based on RDF. CC/PP, however, does not specify how contextual information can be stored [34].

```
<?xml version="1.0"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
         xmlns:ccpp="http://www.w3.org/2002/11/08-ccpp-schema#"
         xmlns:ex="http://www.example.com/schema#">
  <rdf:Description
     rdf:about="http://www.example.com/profile#MyProfile">
   <ccpp:component>
     <rdf:Description
        rdf:about="http://www.example.com/profile#TerminalHardware">
       <rdf:type
          rdf:resource="http://www.example.com/schema#HardwarePlatform" />
       <ex:displayWidth>320</ex:displayWidth>
       <ex:displayHeight>200</ex:displayHeight>
     </rdf:Description>
   </ccpp:component>

   <ccpp:component>
     <rdf:Description
        rdf:about="http://www.example.com/profile#TerminalSoftware">
       <rdf:type
          rdf:resource="http://www.example.com/schema#SoftwarePlatform" />
       <ex:name>EPOC</ex:name>
       <ex:version>2.0</ex:version>
       <ex:vendor>Symbian</ex:vendor>
     </rdf:Description>
   </ccpp:component>
   <ccpp:component>
     <rdf:Description
        rdf:about="http://www.example.com/profile#TerminalBrowser">
```

```
      <rdf:type
         rdf:resource="http://www.example.com/schema#BrowserUA" />
      <ex:name>Mozilla</ex:name>
      <ex:version>5.0</ex:version>
      <ex:vendor>Symbian</ex:vendor>
      <ex:htmlVersionsSupported>
        <rdf:Bag>
          <rdf:li>3.2</rdf:li>
          <rdf:li>4.0</rdf:li>
        </rdf:Bag>
      </ex:htmlVersionsSupported>
    </rdf:Description>
  </ccpp:component>
 </rdf:Description>
</rdf:RDF>
```

Figure 8 Complete CC/PP profile example in XML.

The Comprehensive Structured Context Profiles (CSCP) was developed based on RDF to represent context by means of session profiles. However, this approach does not deal with all our required context characteristics, like the temporal nature of context [30][35].
Here is an example of context representation in CSCP model:

```
<?xml version="1.0" encoding="UTF-8"?> <rdf:RDF
xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns:cscp="context-aware.org/CSCP/CSCPProfileSyntax#"
xmlns:dev="context-aware.org/CSCP/DeviceProfileSyntax#"
xmlns:net="context-aware.org/CSCP/NetworkProfileSyntax#"
xmlns="context-aware.org/CSCP/SessionProfileSyntax#"
<SessionProfile rdf:ID="Session">
<cscp:defaults rdf:resource= "http://localContext/CSCPProfile/previous#Session"/>
<device>
<dev:DeviceProfile>
<dev:hardware>
 <dev:memory>9216</dev:memory>
</dev:Hardware>
</dev:DeviceProfile>
 </device>
</SessionProfile>
</rdf:RDF>
```

Figure 9 example of context in CSCP.

CML (Context Modelling Language) [36] is based on Object-Role Modeling (ORM) [37], which was developed for conceptual modelling of databases. CML provides a graphical notation designed to support the software engineer in analyzing and formally specifying the context requirements of a context-aware application. It extends ORM with modelling constructs [30].

We will refer to some approaches that include the context in the discovery process

Chan and al. defined a context ontology based on OWL to support intelligent agents in their Context Broker Architecture (CoBrA). Their approach targets home area intelligent environments and applies sensor information detection and context awareness as a way of dealing with users' activities, intentions and movements between different home areas [30][38].

We have seen the **CASD** [24] (Context Aware Service Discovery) and it distributed version DCAASD [25], as semantic service discovery approaches taking the context into account. The

context is modeled through an OWL ontology. The contextual information of the basic services is stored in a graph, called a service graph Similar to an RDF graph.

CA-WIS architecture was proposed by B.Soukkarieh and al [39], based on the Web services standards (WSDL, SOAP, UDDI). This architecture is thus composed of three layers depending on each other; the third layer is the Context layer that is responsible of capturing and managing the user and the service context. The context is represented and stored in CSCP model.

In [40], author's present a generic framework that combines service-oriented and context-aware computing in order to provide users with more tailored services. This approach **Cb-sec** (Context-Based Service Composition) adds the context awareness to the existing standers using the CSCP (Comprehensive Structured Context Profiles) as a context representation model and XML for service description.
Multi-Agent System (MAS) is in place for gathering, processing and interpreting the user's contextual information, in which each agent is associated with one type of contextual information. The agents are responsible to gather the data provided by the software and hardware sensors and to match recommend appropriate services among available services.

Bouyakoub et al, proposed in [41] a Web service discovery solution, based on a distributed architecture, uses a multi-agent system (MAS) for the publication and the discovery of services. The user/service contexts and QoS parameters are taken into account during the discovery process. For this purpose, they use technical models[8] to store QoS information of services, and uses the CC/PP model for context representation and storage. The proposed discovery process is realized in two layers according to the type of the context/QoS parameters: in the first step, they select services satisfying all the qualitative attributes of the user context. This shortlist of preselected services passes by a selection based on the quantitative attributes of the two contexts (service and user) using a new quantitative similarity measure (QSim) to evaluate the correspondence degree between the user requirements and the service parameters.

QoS Web services selection has been proven NP hard problem. Many factors lead to this high complexity such as 1- the users set different profiles for selection and compositions, 2- QoS parameters of a Web service differ than what its provider publishes, 3- the number of available similar Web services and 4- within this huge search space, compositions may be built by many ways. This factors increase time complexity of the Web services selection and requires a multi-criteria optimization approach [42].

An optimization problem is defined as the search for the minimum or the maximum (the optimum) of a given function. In most practical optimization problems, several criteria are to be taken into account in order to obtain a satisfactory solution.
As its name indicates, multi-objective optimization aims at optimizing several objectives simultaneously. These objectives are in general in conflict: the improvement of one objective causes the deterioration of another.

Many methods of solving problems of different complexities have been proposed. This variety has made it possible to group the different methods of solving different problems into three main classes: the class of exact methods, the class of approximate methods and the hybrid class.

- Exact algorithms:

[8] A Technical Model (t-Model) is a data structure that represents an XML Web services type (a generic representation of a registered service). https://msdn.microsoft.com/en-us/library/aa303717.aspx

There are many exact algorithms including simplex algorithm, dynamic programming, separation and evaluation algorithms (Branch and Bound, Branch and Cut, Branch and Price and Branch and Cut and Price) The backtracking algorithms, also the algorithms specific to the problem treated like Johnson's algorithm for the solving of scheduling problems.

- ❖ approximate methods:

The metaheuristic solutions this class is devised two sub classes:
The Single-solution metaheuristics: proposed in the literature are metaheuristics based on a single solution and metaheuristics based on the population of solutions.
Many single solution methods have been proposed in the literature. These include: descent, simulated annealing, taboo search, Variable Neighborhood Search (VNS), ILS: Iterated Local Search, GLS-guided local search … etc.
The population-based metaheuristics: starting with evolutionary algorithms, going through genetic algorithms and arriving at algorithms based on swarm intelligence (the particle swarm optimization algorithm, Ant colonization algorithm, bee colony algorithm, coucou search, coucou optimization algorithm, etc.).

2.3.3. Optimization multi-criteria in Web service discovery

Some optimization solutions in the selection of Web services:

- ❖ **Electre Tri**

Extension of the conventional architecture of the Web services and the use of the multicriteria method of decision support.
This solution consists in extending the conventional architecture of the Web services by adding in the UDDI register a new MEC component which aims to evaluate the different possible compositions according to quality of service criteria. The component MEC has a set of Web services and their evaluations according to QoS criteria and generates a set of recommendations for the requestor of the service in order to help him in his choice of the service to be invoked [43].

- ❖ **The weighted sum**

The principle of this method is to assign a QoS vector for each Web service candidate. This vector contains the values of the quality criteria that characterize each Web service. The different parameters are expressed in different units, scales, and sizes, which leads to the problems of compensation and scaling when the SAW method is used directly. To remedy this problem, the authors in a first step, normalization of the quality parameters of each vector.
After the parameter normalization step, the authors make a weight assignment to the quality parameters according to the preferences of the user. Then, they calculate the score of each Web service that represents a weighted sum of the values of the standardized QoS parameters. They then select the service that offers the best score to perform the task [44].

- ❖ **Genetic algorithms**

Other works focused on the application of genetic algorithms to solve the problem of selecting Web services. The functioning of a genetic algorithm for selecting Web services is then based on the following phases [45]:
- Initialization: The Web service set is sorted;
- Evaluation: Each service is evaluated by the fitness function;
- Selection: Creation of a new Web service set by the use of a selection method;
- Reproduction: Possibility of crossing and mutation within the new ensemble.
- Return: Return to the selection phase as long as the stop condition is not satisfied.

2.4. Conclusion

We have seen the evolution of the WS description from a form to another; first, the structured XML based description and the syntactic search approaches, then for the Insufficiency of this approach, appears the semantic with several semantic descriptions and discovery approaches. The reason that makes semantic approach are more efficient is the possibility of searching information by meaning (logical meaning). For more accuracy, other information can be considered in the discovery process, in this case the context information are used to give suitable discovery result.

Chapter 3: Web Services Discovery Architectures

3.1 Introduction
3.2 Centralized architectures
3.3 Distributed approaches
3.4 Hybrid Web service discovery architecture
3.5 Synthesis
3.6 Conclusion

3.1. Introduction.

In this part of the chapter, we present how the discovery registries are implemented. The implementation of the registries influences many aspects: efficiency, client-service communication, scalability and complexity.
Three proposed architectures have been developed known as centralized, distributed and hybrid architectures.
For each type of architecture, we will give a description, examples and discuss their advantages and disadvantages.

3.2. Centralized architectures:

In a centralized registry, all Web services descriptions entries are contained within a one "well known" central entity used by all providers and clients, as a client-server system. Each of client and provider uses the central Web services registry to seek and publish service descriptions.
The major known centralized approach is the UDDI registry.

3.2.1. The UDDI approach

UDDI (Universal Description, Discovery and Integration) was a collaboration between Microsoft, IBM, and Ariba to promote the adoption and use of Web services standards. UDDI is a standard mechanism for registering and discovering Web services. It is a platform independent registry, based on extensible markup language XML and uses standard technologies (SOAP, WSDL, and HTTP). It allows businesses to give list of services and describe how they interact with each other.
UDDI has two main parts: registration and discovery. The registration part means that businesses can post information on UDDI that other businesses can search for and discover; this is the discovery part.
Search in UDDI is based on keyword matching (Syntactic) which is not efficient as huge number of Web services may match a keyword and it is difficult to find the best one [6].

3.2.2. The IRS II approach

IRS II (A framework and infrastructure for semantic Web services) [46] is a centralized approach composed of three main components the IRS II Server, IRS II Publisher and the IRS II Client. Each of these components communicate using the SOAP Protocol (see the next figure number 10).
The ISR II server uses the UPML (Unified Problem Solving Method Development Language) to storage services descriptions and PSM (Problem Solving Methods) as the domain ontology.

The IRS II Publisher links every Web service with it associate semantic description in the server, each PSM is linked to one Web service and the Web service can have different PSM descriptions. The Publisher also generates the code **LISP** or Java of the Web service so the service can be invoked in the same way as the WSDL description [46].

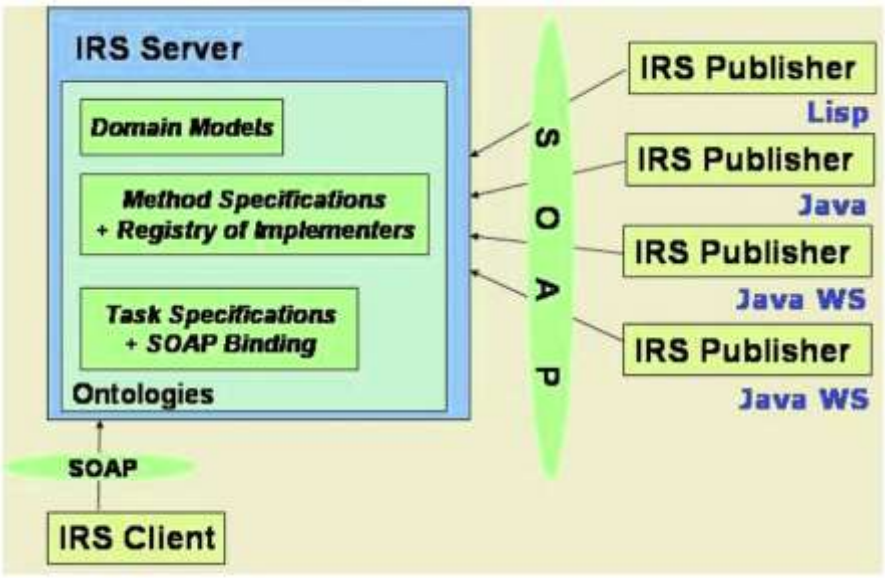

Figure 10 IRS II component architecture [43].

In order to search for a Web service, the IRS II client sends a request to the server looking for a service, the server chose a Web service PSM based on the client request. The client invokes the Web service using the chosen PSM.

3.2.3. The CA-WIS approach

CA-WIS architecture was proposed by B.Soukkarieh and al [39] [47], it is a centralized architecture based on the Web services standards (WSDL, SOAP, UDDI).
In this architecture, authors extend the WS architecture by adding an adaptation layer, which contains various components dedicated to the context acquisition and adaptation. The context is represented by a generic model regarding the user and the service.
The architecture is composed of three layers depending on each other (figure 3.2):

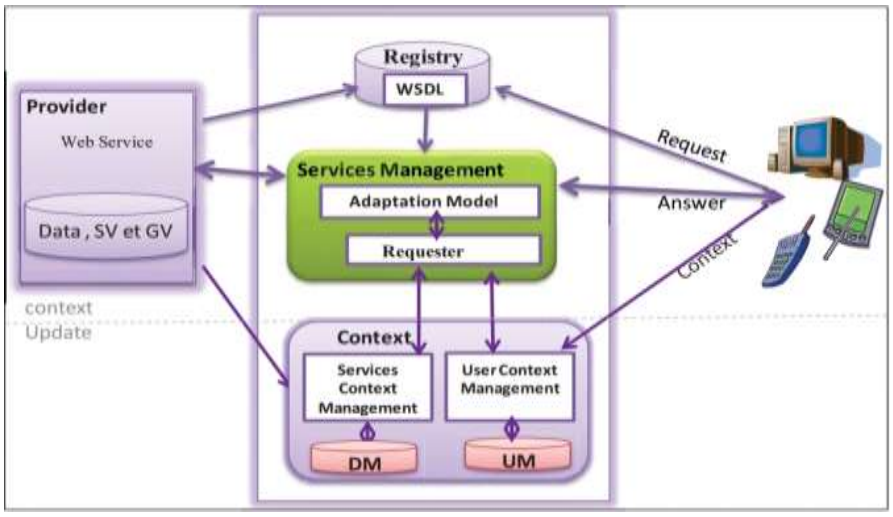

Figure 11 CAWIS approach layers.

1) The registry layer, corresponds to a registry for service descriptions, offers facilities to publish services by providers and to search for services by users. This layer does not play any role in the interaction phase; it is used exclusively during the research phase.
2) The Services Management layer is responsible for managing the adaptation process and

computing the final results to the user. This fundamental layer is composed of two elements:
i. The "Requester", responsible of verifying the user's context evolution by contacting the User Context Management component.
ii. The "Adaptation Model", responsible of the adaptation process.
3) The Context layer is responsible of capturing and managing the user and the service context. It is composed of two modules:
i. User Context Management, responsible of capturing the user context and storing it in a database (User Model (UM)). The user context is obtained in an explicit and implicit way. More precisely, when the user launches his request, he may express explicitly his static characteristics and his preferences but the remaining characteristics (localization, network, etc) are obtained implicitly.
ii. Services Context Management, responsible of the extraction of the Web Services contexts and the storage of these contexts in a database (Domain Model (DM)). The service context is expressed directly by the service provider [47].

3.2.4. WSDA approach

The Web Service Discovery Architecture was proposed in 2002 by Wolfgang Hoschek [48] to fill the interoperability requirement for the Web service on the internet, using industry standards such as XML, XML Schema, SOAP, WSDL and XQuery[9].
It works with four interfaces, The **Presenter** interface which allows clients to retrieve the current (most up-to-date) service description. The **Consumer** interface which offers to the content providers the mean to publish a tuple, set to a consumer (figure 12).

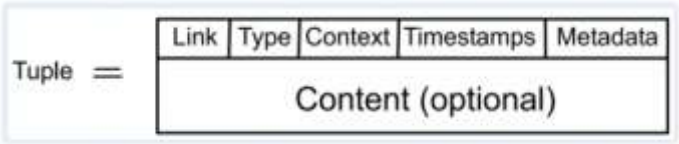

Figure 12 A service description tuple.

The **MinQuery** interface provides the simplest possible query support ("select all"-style). It returns tuples including or excluding cached content. As a minimum, clients can query by invoking minimalist query operations.
The **XQuery** interface provides powerful XQuery support, which is important for realistic service and resource discovery.
All these interfaces and agents are implemented in one central discovery register characterized by Interoperability, modularity, Ease-of-use, Openness and Flexibility.

Although the WSDA is based on a central architecture, it is noticed that it can be transformed into a Peer to peer network database nodes for service discovery. P2P nodes maintain a local database and implement the query interface; Service providers publish (their) service descriptions and/or other metadata in one or more P2P nodes. When a client wish to search for a service, he sends the query to an agent node. The node applies the query to its local database and returns matching results; it forwards the query to select neighbor nodes. These neighbors return their local query results; they also forward the query to select neighbors, and so on [48].

[9] XQuery: is the standard XML query language developed under the auspices of the W3C. It allows for powerful searching, which is critical for non-trivial applications [31].

3.2.5. AASDU approach

As we have seen in the syntactic based approach, AASDU [13] is a centralized multi-agent based approach using the main XML standers: WSDL, SOAP included in JAXR.
The repository system areas of expertise service agents allows to reference agents according to their expertise, so it is not necessary that each agent is aware of all services published in distributed registries. Each agent has just about services related to it area of expertise. In this system, each agent has a decisive profile of interests and expertise. The expertise of the agent is represented by a vector of keywords such that each keyword represents a given area. For each keyword, a score is assigned indicating the degree of expertise of the staff in this area. In addition, each agent has a list of nearby agents (Neighbor list) indicating the areas of expertise of its neighbors. When an agent joins the system, a set of neighbors is assigned randomly.

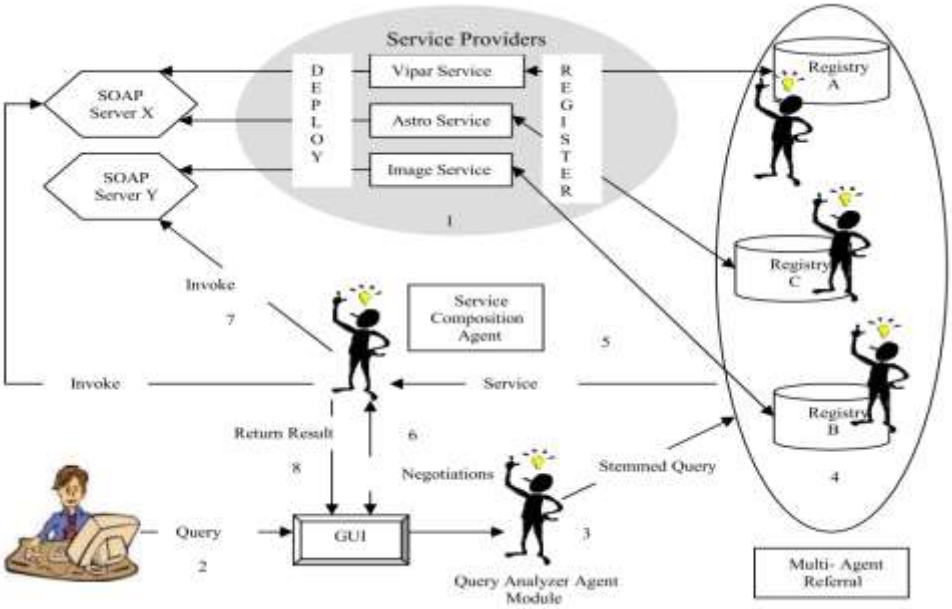

Figure 13 The AASDU architecture [13].

3.2.6. Discussion

Even if central architectures do not require many resources and are easy to implement, they suffer from many problems as having a single failure point and the lack of the scalability in such growing field.

3.3. Distributed approaches:

As a solution for the centralized architectures problems, researchers oriented there focus on the distributed architectures.
Contrary to centralized architecture, the localization of services in a decentralized registry is completely distributed and dispersed. This type of registry architecture has been applied to different types of environments [49].

3.3.1. Distributed Web Service Discovery Architecture approach

The distributed Web service discovery architecture [50] is designed to be robust, reliable and efficient. It is based on the concept of distributed shared memory[10] where applications can read and write information. Thus, it supports both synchronous and asynchronous communications, provides persistent publication of information and is scalable. It provides basic primitives required for publishing and discovering Web service descriptions.
The architecture uses RDF, which makes it adaptable to the need of a particular application domain. Also, the architecture can be adapted to any ontology based languages through the use of format adapters [50]**.**

3.3.2. P2P Based Web Services discovery architectures

At the opposite of the client-server architecture, peer-to-peer is based on the rule that each node is in the same time a client and server. Based on the way in which the nodes are organized and the connection protocols used in these architectures. The peer-to-peer architecture can be separated in two main class: unstructured and structured architectures.

3.3.2.1 Structured P2P architectures

This type of architectures uses a predefined structure to interconnect the peers, while in unstructured architectures each node is arbitrarily attached to a certain number of other nodes [51].

Structured P2P are built according to a well-defined geometric structure as well as deterministic links between nodes. Due to these properties, they can offer guarantees regarding the lookup time, scalability and can also decrease the maintenance cost.
Note that each peer, in structured P2P, maintains the indices of data items of others (but not the data items of its own!) and reliance among peers exists [51][52].

In the structured p2p architectures, we distinguish two categories: indexed table structured p2p approach and semi-structured p2p approaches.

3.3.2.1.A. Indexed Structured P2P Approach for Service Discovery

In this category, indexed structured P2P systems maintain a logical defined structure among the peer nodes using DHT (Distributed Hash Table) indexing tables, which offers an efficient routing for publishing and discovering.
In addition to well-known approaches of the structured P2P systems such as Chord [53], CAN

[10] Distributed shared memory: is a form of memory architecture where physically separated memories can be addressed as one logically shared address space.

[54], and Pastry [55] that was proposed, we will present a some other p2p structured discovery approaches.
Chord was the first structured P2P protocol that is a scalable lookup in a dynamic network with peers frequently oncoming and leaving. Both peers and resources are mapped through the same hash function to an m-bit key space [56]. A DHT stores key-value pairs by assigning keys to different peers. Peers are organized in one-dimensional circle according to their keys. The lookup process emulates the binary search, thus requires O(log N) steps and messages. The important advantages of Chord are its simplicity, provable correctness and provable performance, however Chord cannot support range and multi attribute queries, and due to lookup latency, it suffers from high response time.
Nevertheless, the maintenance of the indexes when the peers join and leave affects the performance of the system (in the case of Web services, the frequency of peer mobility is very low in compare with other P2P network like VANET or MANET, so the performance cannot be effected with this factor).
Moreover, the accuracy of service discovery is low because the search on DHT is based on numeric keys and do not consider semantic information which provides more accurate results.

The CAN (Content Addressable Network) is another DHT d-dimensional torus based P2P system that was introduced by Ratnasamy, Francis [54]. It uses greedy routing strategy where a message is routed to the neighbor of the next peer that is situated near to the required resource [57]. Each peer is connected to its next and previous peer in each dimension [54]. Resources are assigned identifiers by hashing their unique names and Peers are assigned random identifiers in the d-dimensional space.

Authors in [58] design an approach for distributed SWS (Semantic Web Services) discovery based on Chord technology and OWL ontology inference. The SWS Ontology is used to store and match semantic information of published and requested SWS. They adopt Chord Overlay Network (CON), which is composed of many SWS Registries with the same type, to organize a large number of SWS registries into a P2P network so that they can cooperate with each other.

In [59]**,** authors present **A P2P-Based Semantic Web services Composition Architecture**, the structured P2P architecture distributes the functions of UDDI to the local Web peers, and groups Web peers and public Web peers. The Web service functions are described by OWL-S including the QoS information. The SOAP protocol is used for the Web peer communication and requests.
It is based on CAN infrastructure and is used to arrange P2P infrastructure, and it provides distributed hash table to ensure that each service is registered in a specific peer of the public Web [60].

3.3.2.1.B. Semi-Structured P2P Approach to Service Discovery

Semi-structured P2P systems organize the peer nodes into clusters, which are more structured than unstructured P2P architecture, and less structured than DHT-based P2P architecture. Compared with unstructured P2P approach to service discovery, the semi-structured P2P approach has better scalability in message routing by avoiding the flooding within the service discovery. Compared with indexed (DHT) structured P2P approach, the semi-structured P2P approach has smaller overhead to maintain the routing table entries updated[61].

In this case, Xiang et al. [62] presented a P2P service discovery framework, in which, each peer maintains a group of friends peers that it has recently interacted with, allowing a fully distributed and flexible discovery process.
Sahota et al. [63] proposed a grouped P2P network that can be used for scalable grid information service discovery, by dynamically dividing the search space into small sections to enhance its scalability and resilience.

Padmanabhan et al. [64] presented a self-organized grouping framework that achieves efficient grid service discovery by forming and maintaining autonomous resource groups.

Di Modica et al. [65] presented a P2P-based infrastructure for semantic service discovery. Using semantic links, they create different organized groups of peers that share the same field. Peers are grouped together in the network space according to their characterization in the semantic space. A peer may also belong to several groups, as the services it publishes can span different semantic domains.
All main semantic languages are being used for semantic annotation of Web services: OWL-S, WSMO, SA-WSDL enabling adaptable framework for the semantic concept representation.
The proposed architecture use JXTA[11] as infrastructure and a communication protocol [61].

We add to this classification the next generation of P2P network, the self-organized online P2P network, which links on the peers and manages their interactions without centralized control to guarantee the service availability and the efficiency of service discovery and more resilient decentralized infrastructure.

Yuan, Bo et al. recently in 2016 [66] proposed a **self-organized architecture for efficient service discovery in future peer-to-peer online social networks.** It supports service discovery mechanisms for next-generation P2P social networks to enable dynamic interaction, context-aware collaboration and personalized service recommendation.
The main challenge focused is to define a churn-resilient P2P social network where the overlay connections are self-organized based on the social interactions, without centralized control to guarantee the service availability and the efficiency of service discovery. This solution is based on context-aware search algorithms, and interest-based community strategies. In addition, the key word search is enhanced with semantic contents and graphical structures.

Another work on the same orientation is presented by **Zhang, Wenyu et al**, [61] Author's present a **self-organized semi-structured P2P framework that supports scalable and efficient manufacturing services discovery**. A semi-structured P2P overlay network is self-organized by forming and maintaining autonomous enterprise peer groups.
The services are represented comprehensively and formally with a generalized ontology (i.e., manufacturing services ontology that is built using Web Ontology Languages for Services (OWL-S) and Semantic Web Service Framework (SWSF)) for more accurate service matchmaking. Each enterprise PG (Peer Group) dynamically clusters a set of enterprise peers offering semantically similar MSs, and elects the most reputed peer through multi-criteria trust evaluation as its core (i.e., super peer, SP). Therefore, a "SP"-based P2P overlay network [7] is self-organized, which realizes both advantages of centralized and decentralized search. During collaborative problem solving, the similarity values and reputation values of peers can be evaluated periodically, and so the overlay network connections among peers are rearranged autonomously as the network evolves. The proposed framework addresses the drawbacks of

[11] JXTA: is a Protocol of an open network-computing platform designed for peer-to-peer (P2P) computing.

both unstructured and structured P2P architectures, by minimizing the overhead incurred by either the random peer communication or maintenance of peer overlay structure. Therefore, a MS request can be first routed to the suitable super peer and further to its leaf peer in a systematic way, thus supporting efficient service discovery. The proposed architecture has been implemented on JXTA [61].

3.3.2.2. Unstructured P2P architectures

In the other hand, in unstructured P2P, the peers form loosely coupled objects without restrictions on the network structure and the data position and any prior knowledge of the topology. For the neighbor selection peers are flexible eventually.
In unstructured P2P, each peer maintains its own services and service descriptions, otherwise simple indices, such as the two dimensional table, can be employed [67], [68].

As examples of the unstructured p2p architectures, we have Gnutella and KaZaA, two of the most known unstructured systems that has no information about the location of the resource, thus the resource location method is "flooding". A peer looking for a file send a broadcast message in the network saying "looking for this resource". All matching responses are send to the originating peer through the reverse path [67], [68].

Zhang et al. [69] proposed an ontology in P2P-based data and service discovery approach using unstructured P2P infrastructure based on JXTA. To increase the scalability and discovery success rate, an ontology-based approach is used to describe data and services. As for service, the quality of service (QoS) information is added to OWL-S files to get results that are more accurate for users.
By taking the advantage of JXTA architecture, they organized network peers into groups and related providers, which publish similar resources form a Friend Group. Group leader in the system shows much resemblance to the hubs in the JXTA network; Hubs can be organized by geography, peer content similarity or application (figure 14).

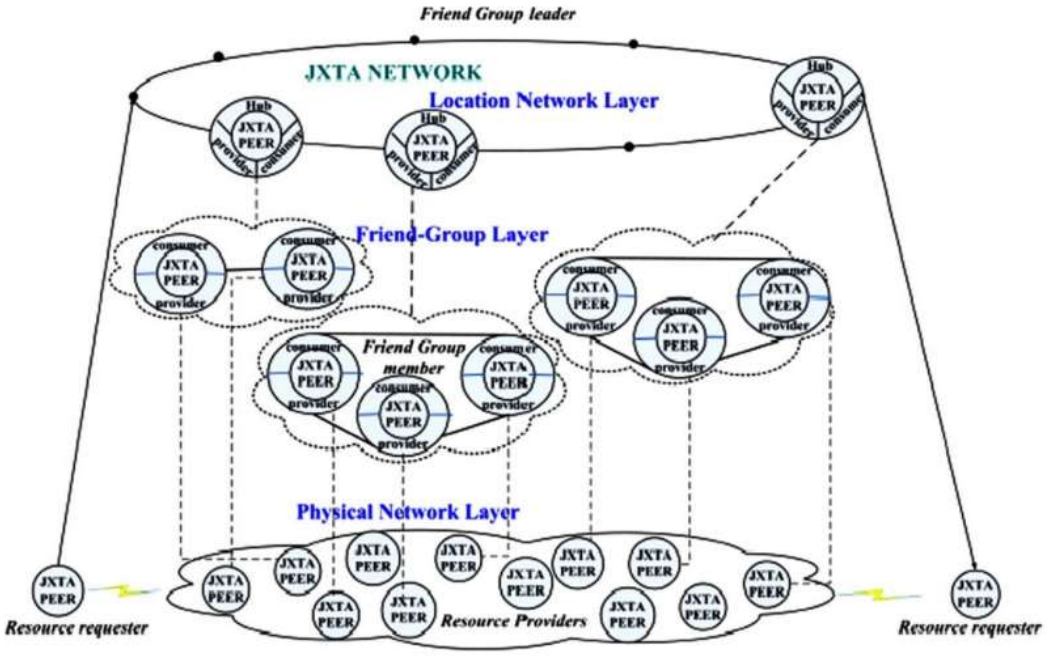

Figure 14 The proposed architecture in [66].

A self-scaling cooperative discovery approach of service compositions in unstructured P2P networks was proposed by F, Angelo and Z, Eugenio [70]. It is a distributed discovery approach for both atomic and composite services. Services functional and non-functional parameters are semantically annotated using owl-s in addition of WSDL descriptions. They used the services descriptions to generate an abstract that is translated to executable processed language (Ws-Bipel) for the service composition discovery. JXTA API's are used for the inquiring and publishing services, also the QoS parameters may be included [61] [71] [67].

3.3.3. Discussion

Compared with traditional centralized client-server architectures used for service discovery, the use of distributed approaches are mainly reflected to improve scalability, heterogeneity and robustness in dynamic and open environments.

Unstructured P2P approach present some problems like discovery loops: Uncontrollable number of hops for reaching the wanted resource, congestion in the network because of the flooding and the inconsistency in the information, so the merits of DHT structured P2P approach are mainly reflected on its high scalability in message routing.

Also a search through unstructured P2P services, basically has no information about the location of the resource (blind), and consequently generate significant traffic overhead when using the flooding method to locate the services and all matching responses are send to the originating peer through the reverse path.
Still the blind flooding is the simplest and the most frequently used method for resource discovery in P2P networks. Therefore, high routing scalability is hard to achieve when employing unstructured P2P approach for service discovery [72].

However, a well maintenance of a DHT or logical overlay among the peer nodes is not a trivial work in dynamic environments where peer nodes may join or leave P2P networks frequently, and consequently incur significant maintenance overhead. Therefore, high maintaining scalability of DHT is hard to achieve to employ DHT structured P2P approach for service discovery.
Also due to the hash characteristic, DHT-based systems can only support keyword searches.
DHT has become the dominant methodology for service discovery in a distributed environment. DHTs provide efficient service discovery in P2P networks, which only need a small number of messages to resolve a query. However, efficiency not only relies on the number of messages to resolve a query, but also the number of messages to maintain the networks. DHTs are not efficient in a dynamic environment, which require a large number of messages to maintain the network topology to keep consistent hash tables at each peer node, although queries can be resolved by a few messages.
Moreover, balancing load on the DHT is a well know problem, because some information may be accessed more frequently. In sum, any attempts of additional control could be difficult to achieve in the distributed P2P architecture due to the lack of a centralized server[72][66][61].

3.4. Hybrid Web service discovery architecture:

Hybrid architectures have been proposed in addition to centralized and decentralized architectures, in which registry information is distributed amongst multiple entities in a peer-to-peer mode, but access to the registry information is through dedicated "super peer" nodes. Hybrid architectures uses the advantages of centralized and decentralized systems, Such architecture appear to users as centralized, since the user is unaware of the distributed implementation and the use of peers is transparent to him [49].
Yang.b and Garcia.Molina proposed a Super-Peer network architecture [73]. It uses the efficiency of a centralized search with the autonomy, load balancing and robustness to attacks provided by distributed search. Super-peers act as centralized servers to their peers; they can handle queries more efficiently than each individual peer could. Moreover, since there are relatively many super-peers in a system, no single super-peer need to handle a very large load.

3.4.1. Super Peer Web Service Discovery Architecture (SPWSDA)

Using the idea from Yang and Garcia approach [73], Evren Ayorak et al has proposed the Super Peer Web Service Discovery Architecture (SPWSDA) [74]. The SPWSDA architecture proposes a hybrid peer-to- peer solution and consists of three types of JXTA peer, registry peer, index peer and client peer. Each peer has specific roles in the architecture.
Index-peer indexes the registry-peer information and registry-peer is the repository storage peers to hold the information. Client peer is a GUI application generated to insert the Web services to the SPWSDA network and search for a Web service in the network.

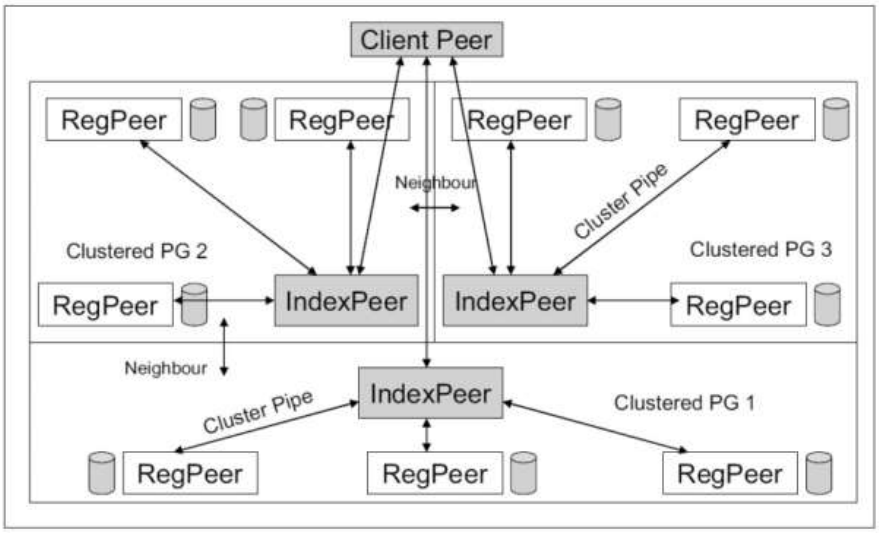

Figure 15 SPWSDA architecture.

When using the CAN data look-up protocol, the client peers are connected in a star topology will combine the efficiency of a centralized search and the load balancing and robustness features that are provided by distributed search. Super-peer and its client peers form a cluster on the network.

They choose the OWL-S Service Profile ontology to describe and classify the Web services in indexed groups called "Concept Cluster Group". Also, the registry peers stores Web services with WSDL description where in discovery process it also returns this description [74].

3.4.2. METEOR-S Web Services Discovery Infrastructure (MWSDI)

MWSDI [75], is a scalable infrastructure for semantic publication and discovery of Web services. It was proposed as a prototype system that allows different registries to register in a P2P network and categorize registries based on domains.
Semantic annotation of Web services and separating registries into domains is the purpose in using the semantic ontology in METEOR-S architecture.
MWSDI approach aim to resolve the overhead problem of finding service registries as a result of the increase of registries number, by creating a unified view of all the registries and providing simple and common means to access them (figure 3.7).

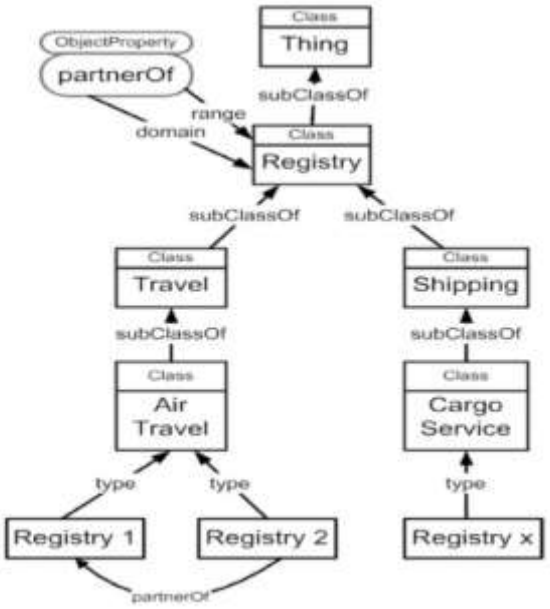

Figure 16 Registries ontology.

The semantic publication and discovery uses WSDL descriptions and the DAML+OIL [76] as the ontology mapping and the SAWS(Semantic annotation of Web services) for semantic matching algorithm. This way it maps inputs and outputs of Web services to ontological concepts. Subsequently, searching can be carried out using templates constructed using ontological concepts.
The implementation and architecture of the peer-to-peer network used by MWSDI gives us the scalability and flexibility required for creating an infrastructure for diverse Web service registries. Implementation is done with UDDI registry provided in JWSDP (Java Web service development pack). However, this concept is applicable to any UDDI registry implementation and other type of Web services registries.

3.4.3. Hybrid Peer-to-Peer Approach for Distributed Discovery in WSMX

In [77], Madani, H et al**.** presented a scalable and automatic discovery mechanism over distributed Web service execution environments. They have implemented their solution based on: WSMO as conceptual model (using WSMX as reference implementation) for services and requests (figure 17).
WSMX is a reference implementation for WSMO (Web Service Modeling Ontology). It uses Web technologies to discover, mediate, select and invoke Web services. WSMX can achieve a user goal specified in WSML by discovering and selecting a matching Web service based on

data and process mediation facilities [77].

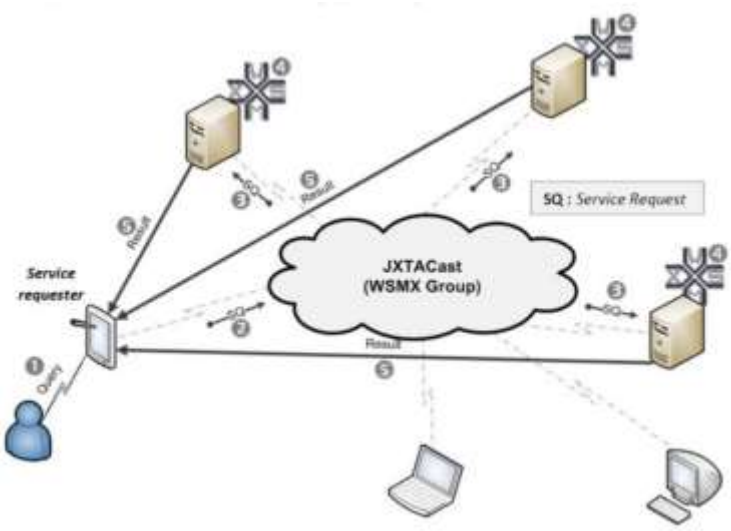

Figure 17 A Hybrid Peer-to-Peer Approach for Distributed Discovery in WSMX.

Built on the top of a semi-decentralized (hybrid) P2P infrastructure using the JXTA specification, the rationale behind choosing JXTA is that JXTA provides a generic infrastructure allowing quickly defining and implementing custom P2P services. Peers are organized in group peer and uses JXTACast as a service to forward a given user request to all super node and child node registries using diffusion in the same group to get a favorable discovery model on large scale, with satisfying and more correct discovery results. The entry and exit of peers is done without other configurations needed on the other peers.

3.4.4. HPS5DSWS Architecture

In [78], authors presented a Hybrid P2P strategy for discovering Web Services **HPS5DSWS**.In this architecture, Web services are distributed in all peers of the network. The network nodes are well structured. We can distinguish two types of peers in this approach: 'Child Nodes' and 'Super Nodes'. A Child Node communicates with only one Super Node. Super Nodes role is the same as that of Child Nodes beside that it act as directories of their kind.

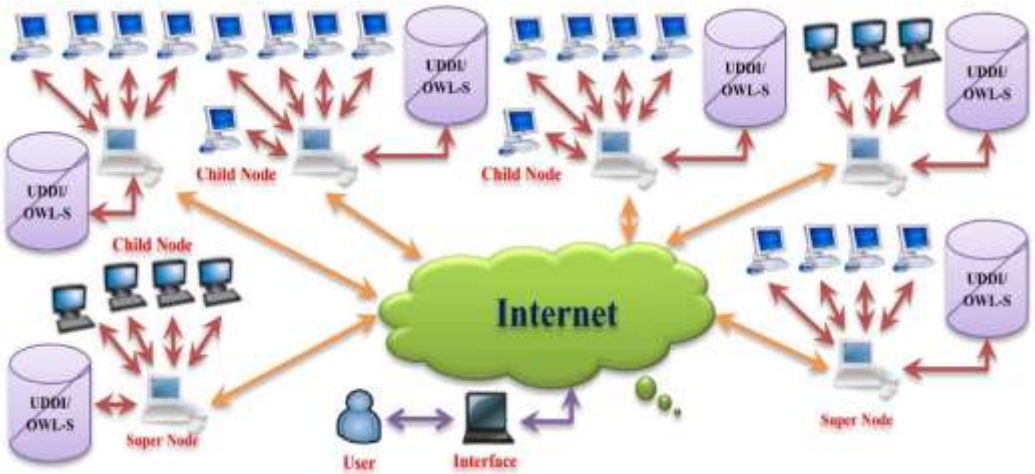

Figure 18 HPS5DSWS architecture [68].

For semantic automatic discovery, they use wsdl2owl-s[12], the WSDL descriptions and the domain ontology to generate the OWL-S descriptions. They implemented their architecture based on JXTA as a generic P2P network platform, JXTACast as a service for sending any query services to all peers network, which will allow designing a large-scale discovery system. This framework discover WS according to user request and preferences in terms of QoS.

3.4.5. Discussion

The decision to use Hybrid P2P networks in this class is justified by several reasons. Hybrid P2P approaches are scalable solutions compared with centralized approaches.
Hybrid P2P networks are designed to implement a search by content, generally using one or more keywords. While these features are not provided by the structured P2P systems, where to find WSs, it is necessary for the applicant to know in advance the object identifier.
The role of Super Nodes varies from an application to another. However, they are generally used to perform the functions relating to the location, routing or organizing Child Nodes to avoid the security and point failure problems like the centralized approaches so, replications and security protocols must be implemented.
This architecture reduces the number of messages received by a large number of Child Nodes in the case of pure p2p, increasing in the other hand the load of Super Nodes. Hybrid P2P systems are used by very large online communities. Thus, the Hybrid P2P systems are more suited to distributed discovery architecture [78]. It is a proven practical solution, but are more complex to implement.

[12] **wsdl2owl-s**: http://semwebcentral.org/projects/wsdl2owl-s/

Table 1 resume the study of some main approaches in the three architectures.

Architecture		Approach	Service description	Communi ca- tion protocol	Using Ontology /context(QoS)	Using replication	Using agents	Advantages	Disadvantages	Suggested Solutions
Centralized		UDDI [6]	WSDL	SOAP	No / No	No	No	-Easy to implement. -Don't require lot of resources. -Autonomy.	-Single failure point. -Not scalable for a large number of service.	- distributed architecture.
		IRS II [46]	PSM	SOAP	UPML/ No	No	No			
		CA-WIS [39]	WSDL	SOAP	No /CSCP	No	No			
Decentralized	Unstructured P2P	DWSDA [50]	RDF	SOAP	WSMO/ No	No	No	-Scalability. -Load balancing. -Robustness. -Heterogeneity.	-Complex implementation. -Congestion of the network because of flooding and DHT tables maintenance. -Security problems because of information propagation and the absence of the identity verification. - DHT do not include semantic information. - Low response time.	-Agent for verifying objects identity. -encrypted published descriptions.
		P2P unified [69]	OWL-S	JXTA	OWL-S/QoS	Yes, access Node replication	No			
		Self-scaling [70]	WSDL/Owl -s	JXTA	OWL-S/QoS	No	No			
		Multi-agent [41]	WSDL	SOAP	No/cc- pp(QoS)	Yes	Yes			
	Structured P2P	**Based on Chord** [58]	OWL-S	Overlay Weaver	No / No	Yes	No			
		Based on CAN [59]	OWL-S	SOAP	OWL-S/QoS	No	No			
		Semi structured [65]	**OWL-S, WSMO, SA-WSDL**	JXTA	No / No	No	No			
	Self-organised	**self-organized manif-service** [61]	MS ontology	JXTA	Yes/ No	No	No			

		self-organized social [66]	Yes	--	Yes/Yes	No	No			
Hybrid		**HPS5DSWS** [78]	WSDL	JXTA	OWL-S/ QoS	No	No	-Autonomy. -Scalability. -Load balancing. -Robustness. -Heterogeneity. -Security (registries and information access goes throw super peers).	-Complex implementation. -Congestion of the network because of flooding and DHT tables maintenance. -Single default failure (Gateway).	-for [64] create replicate for the gateway, which is the case in [69]. - using JXTA security Group for security issue.
		Hybrid based on WSMX [77]	WSMO	JXTA (Cast)	WSMO/ No	Yes	No			
		SPWSDA [74]	WSDL	JXTA (Cast)	OWL-S/ No	No	No			
		METEOR'S [75]	Annotated WSDL	JXTA (SOAP)	DAML+OIL/ QoS	No	No			

3.5. Synthesis

The previous table resume the study of the main approaches of which we mentioned in the three architectures. The study of the approaches was based on the chosen criteria, which are service description, communication protocol, used ontologies, used context, replication and using agent technology.

From a view of architecture we see that centralized approaches are not suitable solution for the increased amount of Web services number when taking in consideration the respond time, performance and resilience (like the case of UDDI). In the same time remarkable efforts was to enhance the discovery richness with the ontologies and context information (the UDDI+ uses context and CASD include the context and the semantics). As a remark, the replication is not considered in this solution what make it vulnerable for the failure problems. Centralized approaches [8], and [24] used agent technology in their approaches.

Then, researchers focused on applying the distributed systems in the discovery process in several deferent approaches. Unstructured P2P networks are used to reduce the load by devising tasks on each peer in the network like in [70] and [41] (in [41] author used unstructured distributed system that include agent technology and context aware environment), which initial problems like the inconsistency and network congestion because the flooding. So structured P2P are build according well defined geometric structure that can offer scalable system with stable lookup time and decrease the number of message circling in the network in the case of a static network (contrarily of the dynamic network, incur significant maintenance overhead which decrease the system scalability). Also the structured p2p using DHT tables can't use semantics in the services lookup because the DHT tables (like structured P2P based on CHORD [59]) cannot contain ontologies and the network congestion cannot be avoided because of the table's maintenance. Self-organized structure has occurred lastly as a solution based on the social interactions between the peers to create the overlay network connection, and [67] are using this technique with and without a centralized control (Super peers). This type can be implemented using ontologies and context aware systems in JXTA environment. From the distributed seen approaches, the replication appears only in [41], [59]. (In [70] the replicate is only for the access node). For the agent technology, we find in the distributed approaches only [41] that include it.

Inspired from the two previous architectures (centralized and distributed), the hybrid architecture is implemented to gain the advantages of both of centralized and decentralized architectures like scalability, search by content (semantic search), resilience and more.

For these reasons some approaches was developed using this concept. As example SPWSDA developed by E.AYORAK [75] based on syntactic and semantic descriptions (WSDL and OWL-S), same as METEOR by K.Verma [76] and HPS5DSWS by A.BOUKHADRA [79] with Qos parameters included for context discovery. These approaches are implemented on JXTA distributed platform. WSMX [78] also use JXTA as a distributed platform and WSMO as semantic annotation for the services description with a replication in the registries as a strength point.

3.6. Conclusion

In this chapter, we presented the state of art of the Web services discovery. We have seen several Web service discovery approaches that was grouped in different architectures. First, the centralized architecture that was a simple and efficient using the syntactic matching than the semantic matching that brings more accuracy. However, with the growing number of Web services and client requests the centralized approach cannot fulfill the large number of requests. Here comes the role of the distributed architectures. Many solutions came from many directions, relays on gathering several servers within a single user interface to do the discovery tasks using generally the P2P technology. How the servers are interconnected and collaborate is distinguished from an approach to another. For example, structured P2P uses DHT tables to create geo-organized network, self-organized approach is based on social interaction between the peers.

The distributed architecture comes with important advantages like the scalability and load balancing, also robustness and heterogeneity. On the other hand, it has some disadvantages like network congestion because of the peers states are constantly updated also high complexity, low response time and some security problems.

Hybrid architecture take advantages of both of centralized and distributed architectures offering multiple keyword and semantic searching witch not the case in pure p2p approaches. Using the super nods reduce the load on the network due to the large number of exchanged messages. In addition, super nods help to avoid some security issues.

For these reasons, the hybrid architecture is an efficient solution for the Web service discovery, but the implementation is more complicated

Contributions

Chapter 4: Evaluation tool for contextual similarity measures in Web services discovery approaches

4.1 Introduction
4.2 Motivation
4.3 The evaluation tool for similarity measures in WS discovery approaches description
4.4 Implementation.
4.5 Experimentation and results.
4.6 Conclusion

4.1. Introduction

Web services are software components that can be accessed via standardized Web protocols to provide functionalities. Based on Extensible Markup Language (XML), Web services are platform and operating system independent, allowing their adoption by various commercial and industrial organizations offering their services on the Web.
In order to determine the Web services that best respond to client (user) requirement, contextual similarity measurements have been proposed to determine the degree of correspondence between the user context and the Web services context. But so far there is no means or a tool for testing and evaluating the proposed similarity measures

The context is the set of information external to the application that can influence its behavior. This information can be used to characterize the situation of an entity. An entity is a person, device, application or object that may be relevant to the interaction between the user and the application, including the user and the application (service) [80].
With this definition, it can be said that each entity (user and Web service) has its own context. The service context can group service location (geographic restriction), service cost, service category, QOS parameters, and so on. The user context can be constituted by its location, preferences, etc.
description of Context information

In order to optimize the discovery of Web services, and to return the services that best meet user requirements, similarity measures have been proposed to determine the correspondence between the user context and the Web service context.

4.2. Motivation of the contribution

Similarity defines the degree of resemblance or correspondence between two objects. To compare two characteristics that form a context, we can compare their values. However, when we have to compare several characteristics, this method is insufficient. Consequently, similarity measures were proposed to determine the degree of similarity between two contexts while taking into account all the characteristics that make up the context.

In general, a similarity measure is defined in a universe U which can be modelled using a quadruplet: (Ld, Ls, T, SF) [81].
• Ld the representation language used to describe the data;
• Ls the representation language of the similarities;
• T a whole of knowledge about the studied universe;
• SF the similarity binary function: SF: Ld × Ld→Ls.
A similarity measure is a function that satisfies the following properties:

$$\begin{gathered} \forall\, x, y \in Ld : SF\,(x, y) \geq 0 \\ \forall x, y \in Ld : SF(x, x) = SF(y, y) \geq SF(x, y) \\ \forall\, x, y \in Ld : SF\,(x, y) = SF(y, x) \end{gathered} \qquad \textbf{Formula/Eqt (1)}$$

In the same way, a dissimilarity measure is defined as a function, which checks the following properties:

$$\forall x, y \in Ld : DF(x, y) \geq 0$$
$$\forall x, y \in Ld : DF(x, y) = 0$$
$$\forall x, y \in Ld : DF(x, y) = DF(x, y)$$

Formula/Eqt (2)

It is also possible to transform a similarity measure SF to a dissimilarity measure DF by using the following relation:

$$\forall x, y \in Ld : DF(x, y) = 1 - SF(x, y)$$

Formula/Eqt (3)

Contextual similarity measures are used to determine the degree of correspondence between the user's context and the service context, and subsequently find the relevant services. In our work, we will compare and test new contextual similarity measures with existing measures.

Many similarity measures are proposed, so how to determine the more effective one and how can we test and validate these measures?

For example, Bouyakoub et al used in [41] a Web service discovery solution based on a new quantitative similarity measure (QSim) to calculate the correspondence degree between the user requirements and the service parameters. The lack of such tool or dataset to test their measure led them to use a group of four users on a 12 services to evaluate the approach, this way can give us an idea about the measure performance but it still not sufficient, for this matter we propose to create an evaluation tool for the contextual similarity measures.

4.5. The evaluation tool for similarity measures in WS discovery approaches description

We propose a tool to evaluate the context similarity measures using a generated dataset that contain about six hundred services. The tool permit to test, compare existing measures and thus to determine the effectiveness and validity of these measures. It also allows introducing new measures and to save and show the results using graphs.

4.5.1. System and functions description

The architecture of our system is illustrated in Figure 19, this architecture is composed of 4 layers: the WSDL description layer, the context file layer, the Web services search layer, and the graphical representation layer.

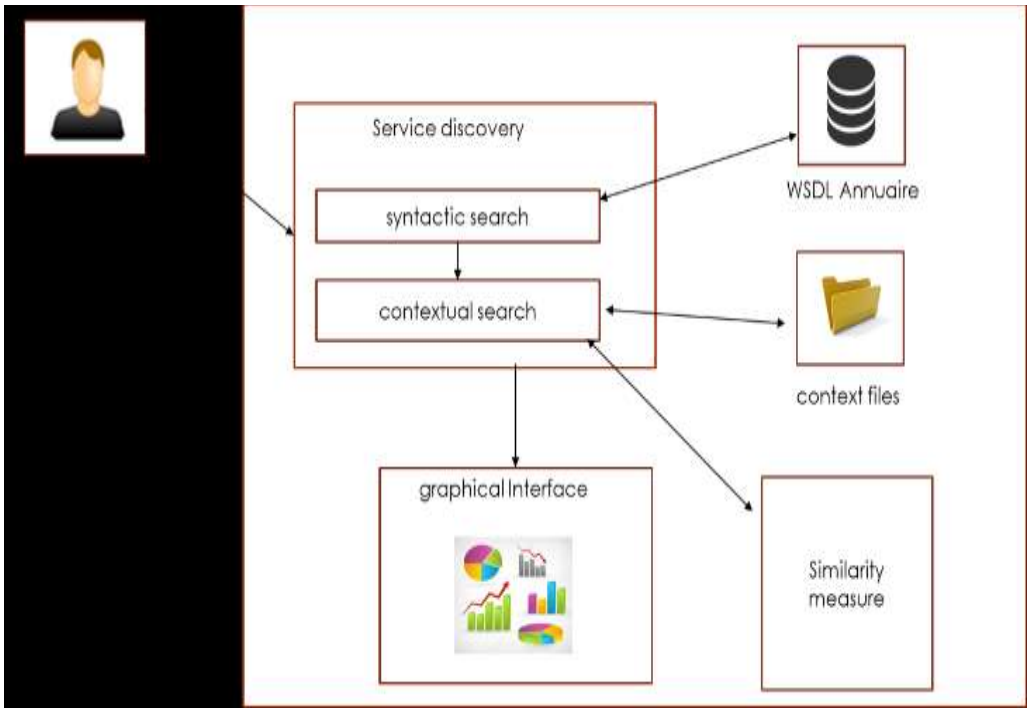

Figure 19 The system architecture.

- Layer 1: WSDL description Layer.

From a list of three thousand URL of Web services descriptions, we retrieve six hundred functional descriptions of Web services (WSDL) from the internet.

- Layer 2: Contextual information generation process:
  - Step 2.1 Randomly and using correlation we generate the contextual information (attributes) of Web services from a set of values, then the creation of the cc/pp files corresponding to the contexts of the services.
  - Step 2.2: Save the generated cc/pp files to a folder containing all services contexts.
- Layer 3: Web Services Search Process:
  - Step 3.1: The user introduces a request specifying his needs. This request contains keywords that allow to specify the domain (function) of the Web service.
  - Step 3.2: A syntactic search is performed on the descriptions of the Web services of the directory (WSDL), this search makes it possible to find the services that respond syntactically to the request of the user.
  - Step 3.3: Retrieve the context of the user and perform a matching between the context of the user and the context of each service found during the syntactic search using the similarity measures.
  - Step 3.4: Return to the user the list of services that fit his / her context.
- Layer 4: Graphical representation of results:
  - Step 4.1: Retrieve user search results.
  - Step 4.2: Represent the search results as a graph in order to be interpreted and compared.

4.5.2. Choosing Contextual model and attributes

In our work, we use the CC/PP model, an XML-based model to represent the contextual information of services and users. This model is easy to implement, and appropriate to our context structure that has two levels. Moreover, this model is more structured and organized. The CC/PP model is an extensible model that allows us to add new attributes when needed.

The context is composed of two types of attributes: quantitative attributes like the service performance, the availability, capacity… and qualitative attributes like the operating system, the type of terminal, the service's language, etc. [41].

The choice of contextual attributes is important in order to represent the information concerning the Web services such as location, language, quality of service characteristics. For this, we have chosen attributes, which are used in [82] to model the context.

4.5.2.1. Service Context Attributes

For the service context, we chose a composed model of two components:

- *Service components:* Contains Web service information and QoS parameters.

• Service information: contains information about the service such as name, description, provider...
• Qos: contains quality information related to the Web service (performance, security, etc.).
-Security: the trust and security mechanisms implemented.
-Performance: represents the execution speed of a query.
-Reliability: the ability of a service to perform its required functions under specified conditions.

These three chosen parameters of QoS have a correlation and a dependence between them, ie. in some cases, if the level of reliability and safety are high then the performance will be low.
• The language used by the service.
• The geographical location of the service.

- *Terminal component:* This context component contains information about the appropriate terminal for the Web service.

• Software settings: concern the operating system of the service.
• Hardware parameters: refers to the device adapted to the service.

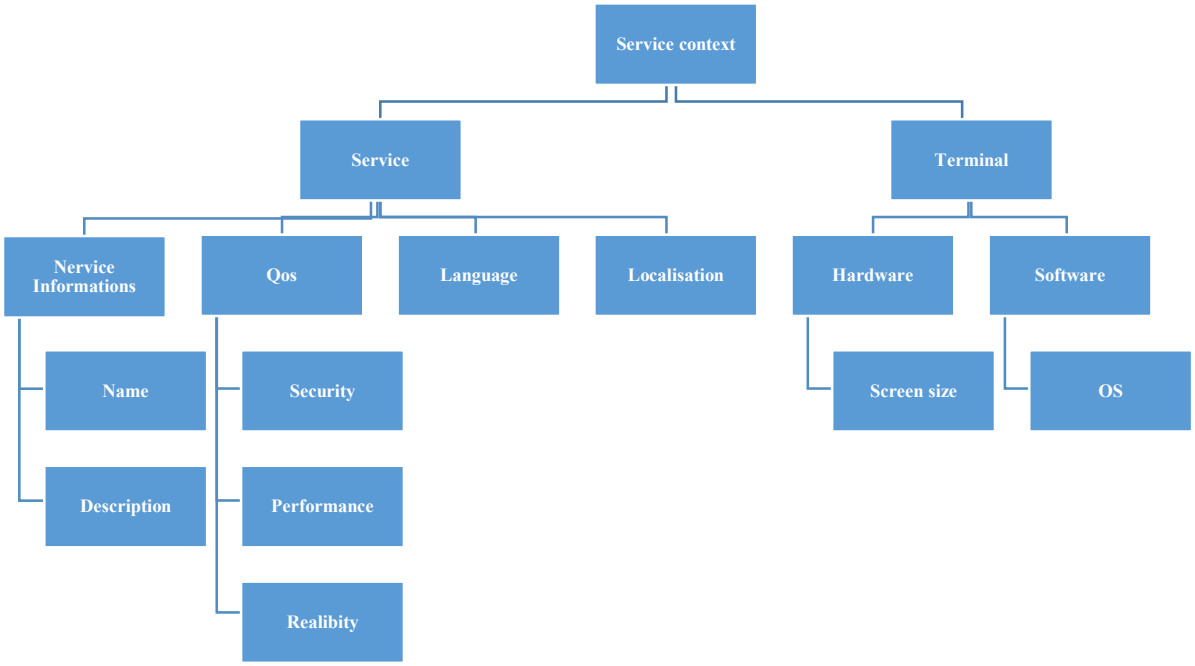

Figure 20 Service context model.

4.5.2.2. User context attributes

The user context is also composed of two components:

User component: contains the information and characteristics of QoS required (requested) by the user.

- User information: contains the user's personal information such as name, first name and e-mail.
- QoS (required): represents the quality of service information required by the user (performance, security and reliability).
- The language preferred by the user.
- The geographical location of the user.

Terminal component: This component contains information about the terminal used by the user.

The main problem is that there was no available dataset of Web services (a set of Web services descriptions) containing contextual information. In addition, there is no method or tool to automatically generating the context of existing Web services.

In our solution, we will create a module whose role is to generate the contextual information of the Web services for each attribute in an XML file. As an example: for the “Location” attribute values are (Algiers, Paris, ...), for the “OS” attribute: Windows, Linux, Mac OS, etc.

For user profiles (user context) containing personal information such as name, first name and his requirements including QoS, language, location etc. we give the user the possibility to enter his profile directly into our tool.

4.5.3. The Web services discovery steps

In this work, the discovery of services goes throw two steps the search for functional information and the search for non-functional information.

4.5.3.1. Functional search:

In returns a set of Web services that meet User's functional needs, using syntactic search. Syntactic search is a basic search, which consists of providing a query by the user containing keywords such as service domain ..., these keywords will then be compared to the different descriptions of available Web services, and return those corresponding to Functional requirements.

4.5.3.2. Non-functional information research

The search for non-functional information consists of searching for the contextual information of the Web services found during the syntactic search, and to keep only services having a context corresponding to the user context. In our work, we use two types of research: single-phase search and two-phase search.

a) Single Phase Search: this search is performed in a single part, consists in determining the degree of similarities between the user context and the context attributes of Web services returned during the syntactic search using similarity measures, and without differencing the quantitative attributes from the qualitative one.

b) Two-Phase search: This search performs a matching between the context of the user and that of each Web service found during the syntax search, using two types of matching: qualitative matching and quantitative matching.

   i) Qualitative Matching: consists in keeping only Web services having a qualitative context identical to that of the user. For example, if the user indicates in his location as "Algiers" only services with "Algiers" as a value for the "location" attribute will be kept.

   ii) Quantitative Matching: Quantitative matching involves retrieving all candidate Web services during qualitative matching and calculating the degree of resemblance between the context of the user and the context of each service using the similarity measures.

We define also X and Y are two vectors (service context and user context): $X = (x_1, x_2, \dots, x_n)$ and $Y = (y_1, y_2, \dots, y_n)$

We associate for some similarity measures a vector $a = (a_1, a_2, \dots, a_n)$ where a_i represents a weight of an attribute of the context which is between 0 and 1, where: $\sum_i a_i = 1$

4.5.4. Introduction of similarity measures

Various studies are carried out to improve the effectiveness of contextual similarity measures.
We have chosen to introduce the existing similarity measures directly into the source code when implementing our tool, by translating the various similarity measures in the form of an algorithm.
Among the measures of contextual similarities that are introduced in our tool:

4.5.4.1. Jaccard's measure

This measure is defined as follows

$$J = \frac{a}{a+b+c} \qquad \textbf{Formula/Eqt (4)}$$

where:

- a: number of features proposed by the service and requested by the user.
- b: number of features proposed by the service but not requested by the user.
- c: number of characteristics requested by the user but not offered by the service.

4.5.4.2. The quantitative similarity measure QSim

This measure [82] is used to compare the context of the user and the context of the service. The QSim measure is defined as follows:

$$\mathrm{QSim}(\mathrm{X},\mathrm{Y}) = \frac{\mathrm{a}\sum_{i=1}^{i=n} w_i \ * ASim(Xi, Yi)}{\mathrm{a}\sum_{i=1}^{\mathrm{i=a}} w_i + \ b \ \sum_{i=1}^{i=b} w_{a+i} \ + c\sum_{i=1}^{i=c} w_{a+b+i}} \qquad \textbf{Formula/Eqt (5)}$$

- a: is the set of common characteristics between X and Y.
- b: is the set of characteristics existing in X and not existing in Y.
- c: is the set of characteristics existing in Y and non-existent in X.

ASim: is the atomic similarity between each characteristic of X and Y and is defined as follows:

In the case of a single-phase search:

$$ASim(X_i, Y_i) = \begin{cases} 1 & if \quad (X_i = Y_i) \\ 0 \quad if \ ((X_i \neq Y_i) \wedge (type = Qualitative)) \\ \dfrac{\min(X_i, Y_i)}{\max(X_i, Y_i)} \ if \ \ ((X_i \neq Y_i) \wedge (type = Quantitative)) \end{cases} \qquad \textbf{Formula/Eqt (6)}$$

In the case of a two-phase search:

$$ASim(X_i, Y_i) = \begin{cases} 1 \ if \ (X_i = Y_i) \\ 0 \ if \ (X_i = 0 \ or \ Y_i = 0) \\ \dfrac{\min(X_i, Y_i)}{\max(X_i, Y_i)} \ else \end{cases} \qquad \textbf{Formula/Eqt (7)}$$

4.5.4.3. The cosine similarity

This measure allows to determine the resemblance degree between the service context and the context of the user [83].
The cosine measure is defined as follows:

$$Cosine(X,Y) = \frac{\sum_{i=1}^{n} X_i * Y_i}{\sqrt{\sum_{i=1}^{n} X_i^2} * \sqrt{\sum_{i=1}^{n} {Y_i}^2}} \quad \textbf{Formula/Eqt (8)}$$

Furthermore, we propose to the user the possibility to introduce new measures in our tool and to compare them with the existing measures in order to evaluate and validate their effectiveness.

The user enters in the form of a string representing the new measure of similarity using operators such as (sum, min, max, etc.) and variables (such as user and service context attribute values, number of common attributes between user and the service context, etc.). These variables are presented in the interface and should be used in the formula to calculate the degree of similarity. Once the user has entered the new measure, it is saved for later use.

4.5.5. Testing and Comparing Measures of Contextual Similarities

In this part, we are interested in testing similarity measures in order to study the results returned. To do this, we test each new measure on a set of services and determine the degree of correspondence between each service context and user context. We then compare the results of the new measure with the results of the existing measures using the same contextual information, and finally, in order to evaluate the new measure of similarity, we calculate the sum of the similarity degree and the dissimilarity degree. The best result is the one that has a sum approaching 1.

The measure of dissimilarity between two contexts is defined by the Manhattan distance [82]:

$$Distance(X,Y) = \sum_{i=1}^{i=n} w_i * Dist(a_i, b_i) \quad \textbf{Formula/Eqt (9)}$$

n: Number of attributes.
w_i : The weight of the i th attribute such that $\sum_{i=1}^{i=n} w_i = 1$

4.5.6. Storage and result representation

We offer to the user the possibility to visualize the results in the form of a graph (curves and histograms) to be able to compare between the different values obtained. We chose to save the results in an Excel spreadsheet, in this way the backup is done quickly and allows us to find the results and the graphs quickly and to consult them later.

4.6. Implementation

4.6.1. The development environment

To implement our tool, we chose to use the Java-based object-oriented programming language. We have chosen this language for various advantages:

- Free and open source.
- Allows a program to be run in different operating systems.
- Has rich and comprehensive libraries.
- Compatible with the tools we use.

4.6.2. Implementation of service directories

In order to enable the search, we have implemented the service directory as a database. To do this we chose the MySQL database management system for the following reasons:

- Simple and easy to handle.
- Very often used.
- Has a JDBC driver that supports compatibility with Java applications.
- Available in open source distribution.

4.6.3. Interfaces and operations

The window shown in the next Figure (21) shows the provider interface. This window allows the provider to view the published Web services, showing their id, service name, service description and WSDL file link. In addition, this window gives the possibility of adding a new service

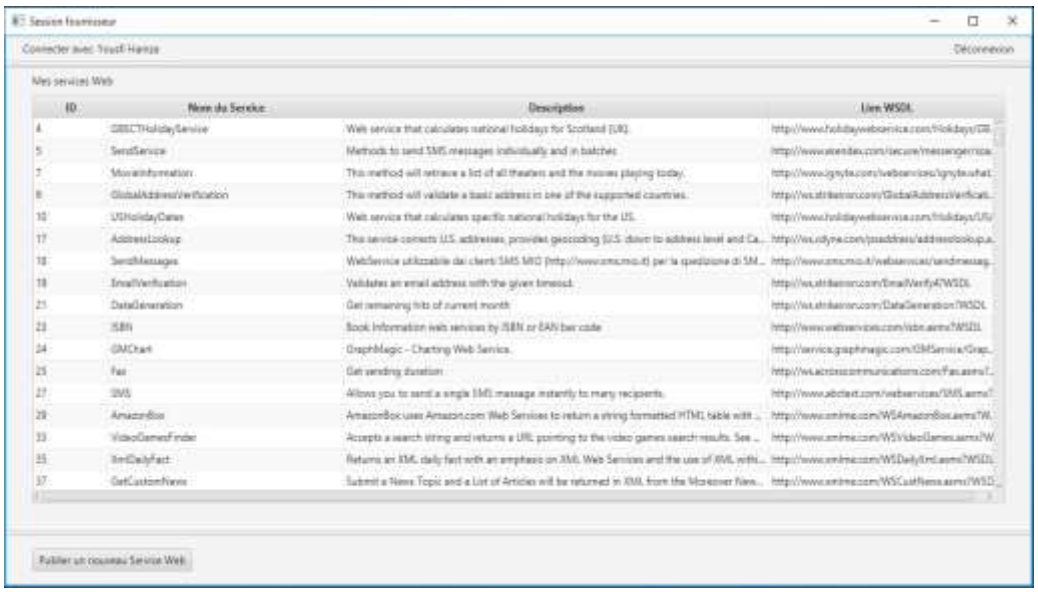

Figure 21 Provider window.

The window shown in Figure 22 shows the user's session. It allows starting a search of Web services by parameterizing its search thanks to field query 1, radio group 2 and button 3.

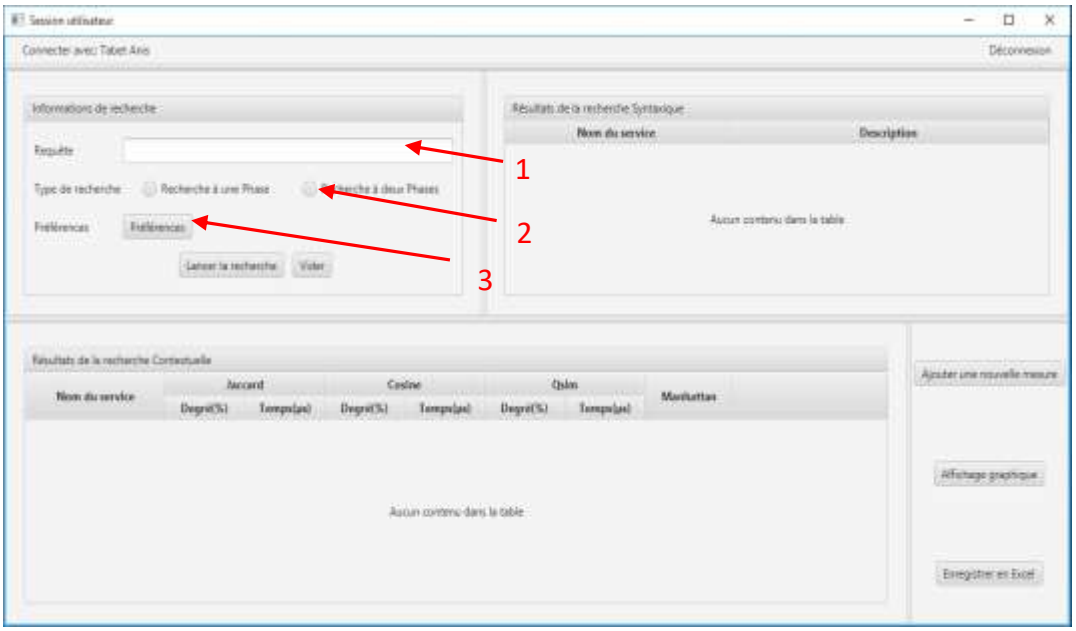

Figure 22 User discovery window.

Field 1 "Query" allows the user to enter the keywords that describe the service that interests him. These keywords will then be compared syntactically with the description of each service to return only those relevant to its query.

The radio group 2 "Search type" allows the user to choose the type of search he wants either one phase or two phases. Each search type has its own preferences window.

The button 3 "Preferences" allows the user to access the window shown Figure 23, to enter the preferences in order to filter only the relevant services in relation to its context. Preferences are divided into two groups: preferences that relate to qualitative and quantitative attributes. Using this form, the user can define the values and weights (in one phase mode the weights will shows up automatically in part 1) that each attribute will have at discovery, and the minimum acceptance threshold, the default threshold is set to 50%.

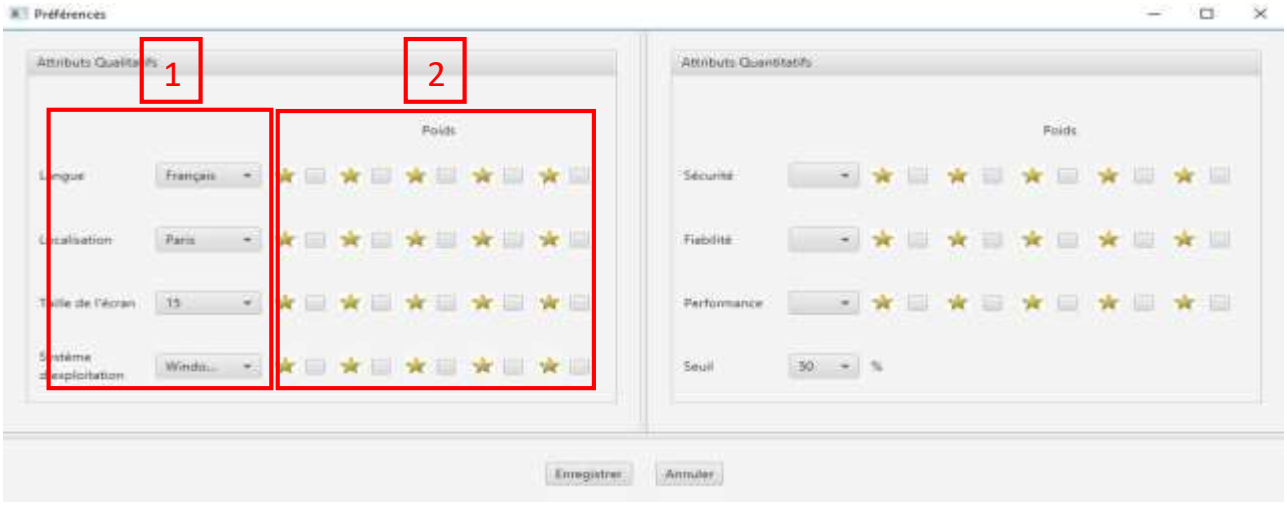

Figure 23 Context values and weight window.

4.6.4. Introduction of a new similarity measure

We have used the Jep Java library. Jep Java is an analyzer that evaluates mathematical expressions. This library allows users to enter a formula as a string and evaluate it instantly. Jep supports user-defined variables, constants, and functions. A number of common mathematical functions and constants are included.
The user has the possibility to introduce a new measure of similarity in order to compare it with the measures already existing. To do this, the user chooses the "Add New Measure" button in Figure 24 and must:

- Enter the name of the measure and the formula of the measure using Form 1 and 4.
- Check (specify) the parameters used by the formula using form 2.
- The user can use a similarity measure that he has already introduced using the list in 3.

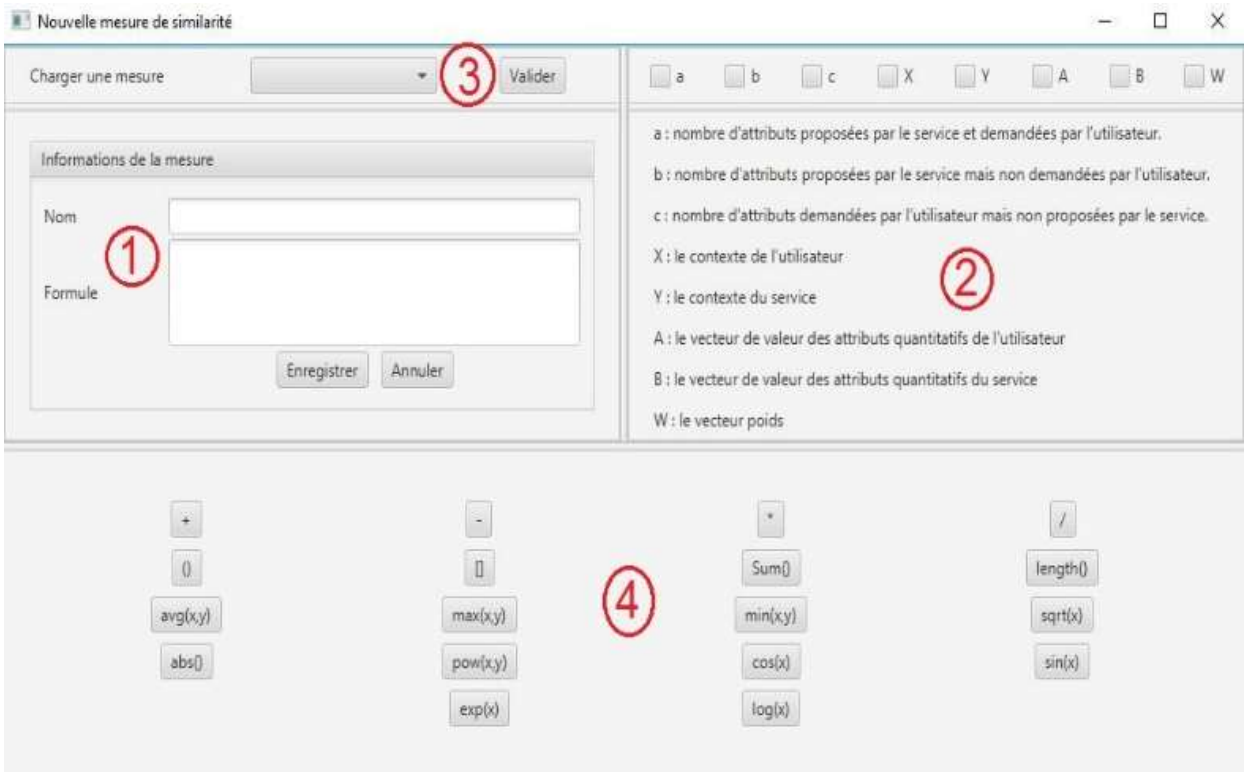

Figure 24 Introducing new similarity measures.

Once the user has completed the search parameters and introduced the new measure, he can start the search by clicking the "Start the Search" button (figure 25).

In Table 1, the window shows the list of services found during the syntax search.
In Table 2, this table shows the correspondence degrees between the contexts of the services found in syntactic search with the context of the user, as well as the execution time in (µs) of each similarity measure.
The user can display the results as a graph by selecting "Graphic Display" button to compare the results of the different measures of similarity and to save the results as a document Excel by choosing the "Save to Excel" button to store them.

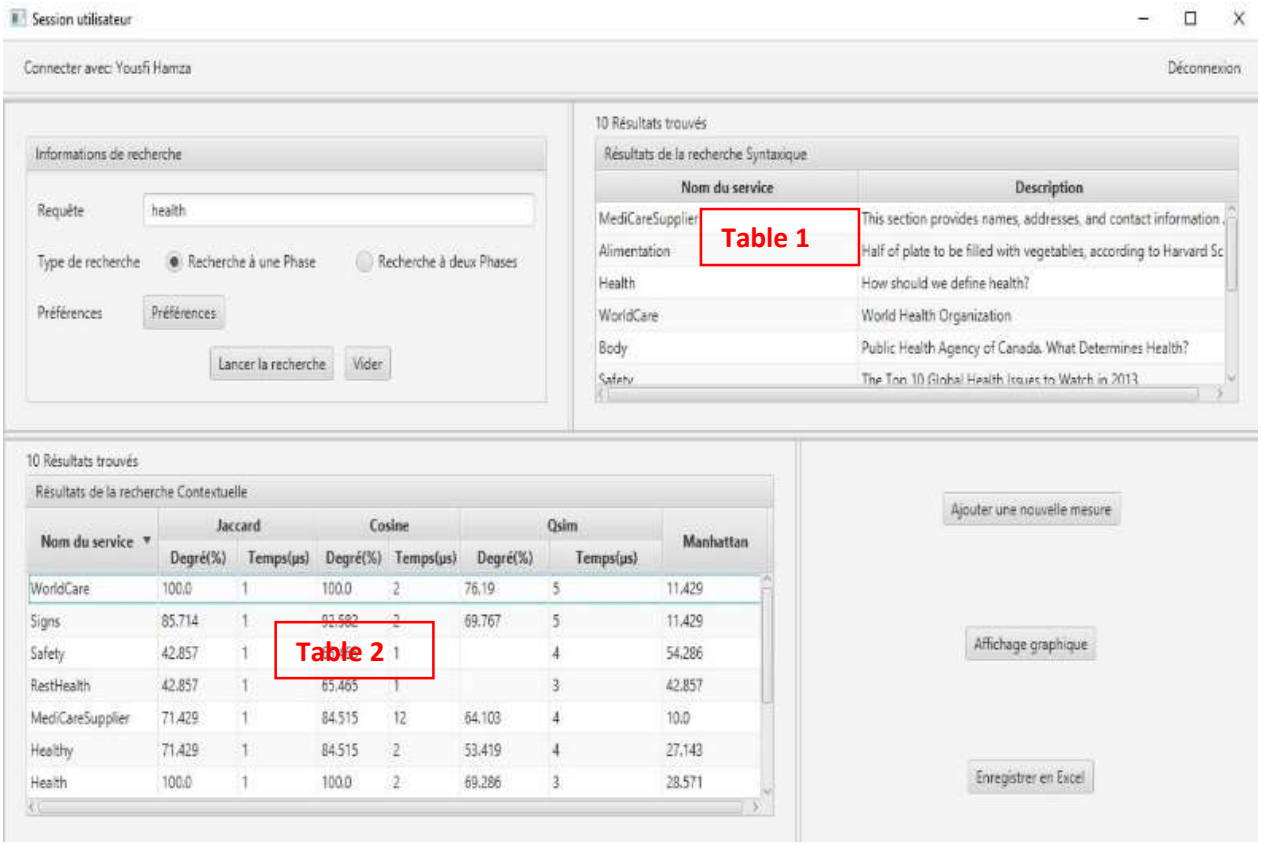

Figure 25 Search result window.

4.7. Experimentation and Results

For this section, we will talk about the experimentation result done on the realized tool, using three different user contexts with different qualitative and quantitative attributes, and across about 600 available Web services. For each user, we calculated the degree of similarity and the execution time between each context of the Web services returned during the syntactic search. In order to evaluate the effectiveness of similarity measures, we compared the similarity measures with the Manhattan distance.

The minimum results for contextual discovery is set at 50% to avoid displaying services with results below.

The following table shows the samples chosen for the user contexts:

Table 2 User's context.

User context	**Users values**		
	User 1	***User 2***	***User 3***
Language	Fr	En	Ar
Localization	Paris	London	Alg
Screen size	15	17	19
Operating system	Win	Linux	Mac
security	3	0	2

User context	Users values		
	User 1	***User 2***	***User 3***
Reliability	1	3	0
performance	2	1	0

The following tables represent some results of the discovery process for both, one-phase search and two-phase search.

Table 3 One phase discovery result for user 1.

Service name	Jaccard		Cosine		Qsim		Distance Manhattan (%)
	Degree (%)	Time (µs)	Degree (%)	Time (µs)	Degree (%)	Time (µs)	
MediCare	71,42	1	84,51	5	67,30	21	20,34
Health	100	0	100	1	84,82	6	14,48
WorldCare	100	0	100	1	100	2	0
Body	<50	0	65,46	1	<50	4	45,51
Safety	<50	1	65,46	1	<50	4	64,13
Signs	85,71	0	92,58	1	88,67	4	12,41
ChooseUr	<50	1	53,45	2	<50	5	47,58
RestHealth	<50	1	65,46	2	<50	5	60,69
Healthy	71,42	1	84,51	2	57,69	4	34,13

Table 4 Two phase discovery result for user1.

Service name	Jaccard		Cosine		Qsim		Distance Manhattan (%)
	Degree (%)	Time (µs)	Degree (%)	Time (µs)	Degree (%)	Time (µs)	
Health	100	1	100	2	51,11	13	38,88
WorldCare	100	1	100	2	100	2	0
Signs	66,66	1	81,65	2	<50	2	33,33

4.7.1. Discussion of results

From this experience, we can conclude the following points:

A) From results we can see that the number of services discovered with the Jaccard measure and the Cosine measure is higher compared to the Qsim measure, and the number of services discovered with the two-phase searches and lower than the one-phase search, due to the following reasons:

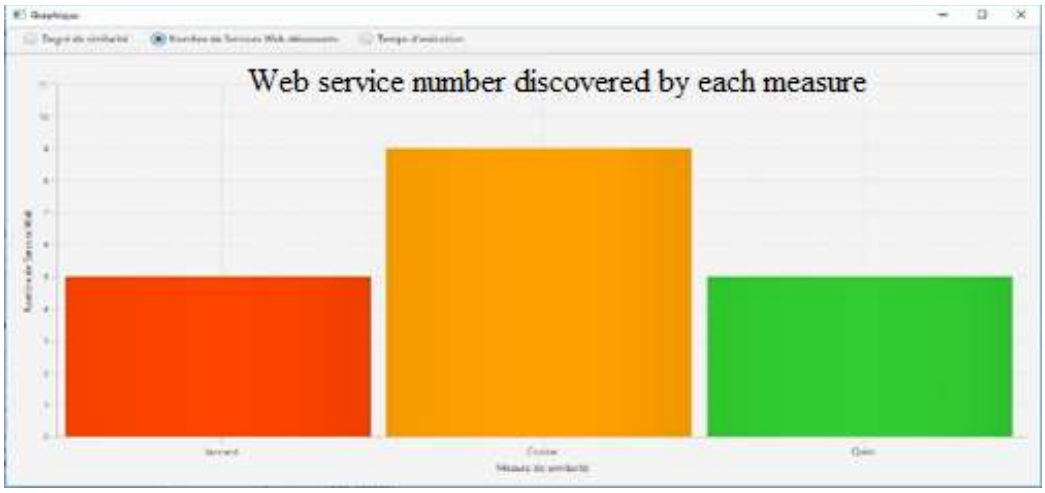

Figure 26 Number of discovered services for user 1 in one phase mode.

The Jaccard measure and the Cosine measure do not take into account the values of the attributes forming the context of the Web service; they only consider the existence of the attribute in the description.

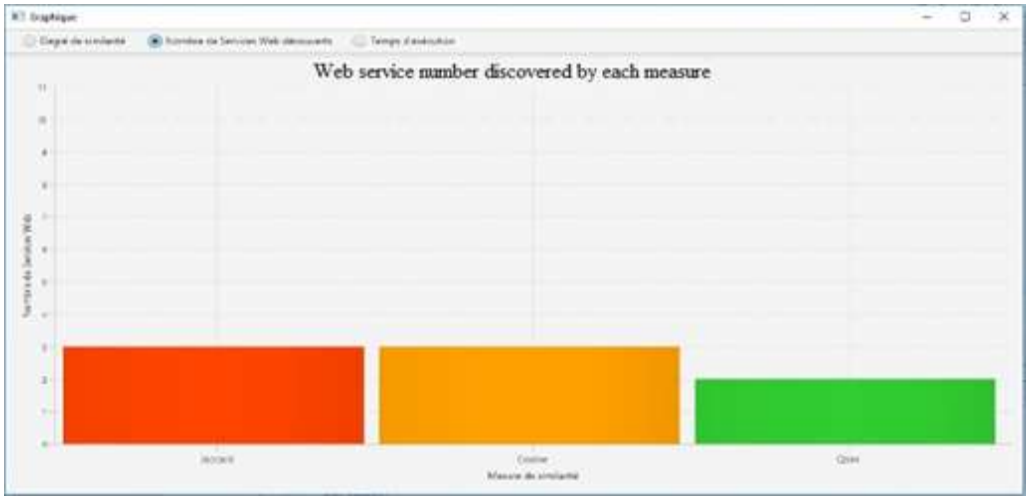

Figure 27 Number of discovered services for user 1 in two phase mode.

Unlike the Qsim measure, the Jaccard and Cosine measures do not distinguish between functional and non-functional attributes.
Two-phase search eliminates services that do not match the qualitative attributes of the user's context to the service context.

B) The results shows that the Qsim measure is more efficient and accurate compared to the Jaccard and Cosine measures, because from this experiment it can be seen that there is complementarity between the Qsim measure and the distance of Manhattan, contrary to the Jaccard and Cosine measures.

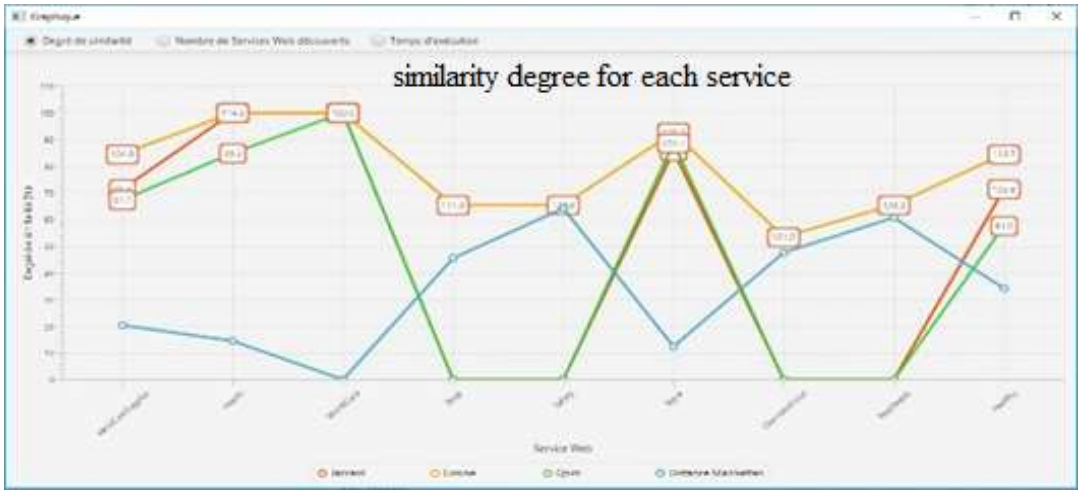

Figure 28 Similarity degrees of Web services in one-phase search for user 1.

For example:

- For user 1, the degree of similarity of the Health service is 84.82% and the Manhattan distance is 14.48%
- For user 2, the degree of similarity of the ChooseUr service is 96.16% and the Manhattan distance is 7.69%,
- For user 3, the degree of similarity of the Safety service is 93.47% and the Manhattan distance is 2.60%.

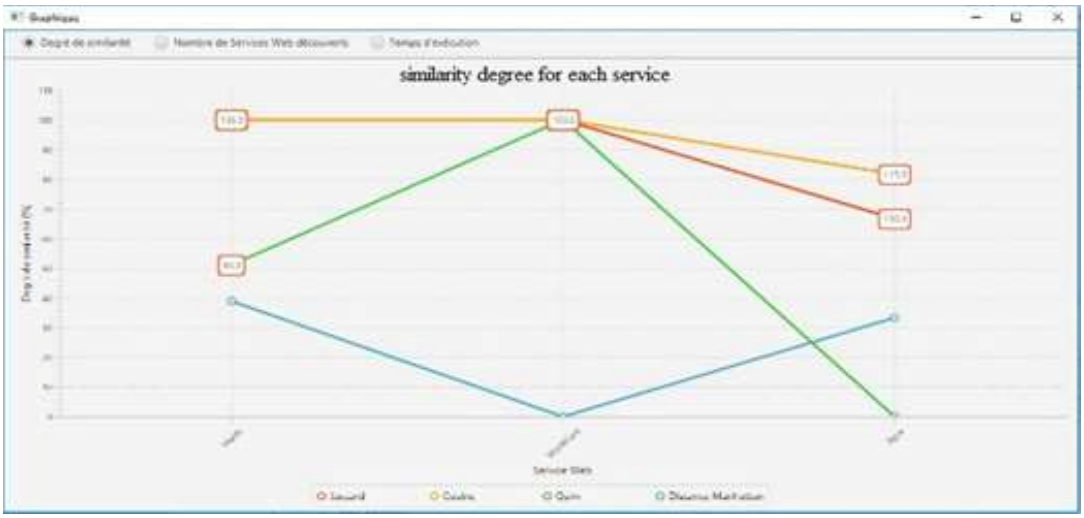

Figure 29 Similarity degrees of Web services in two-phase search for user 1.

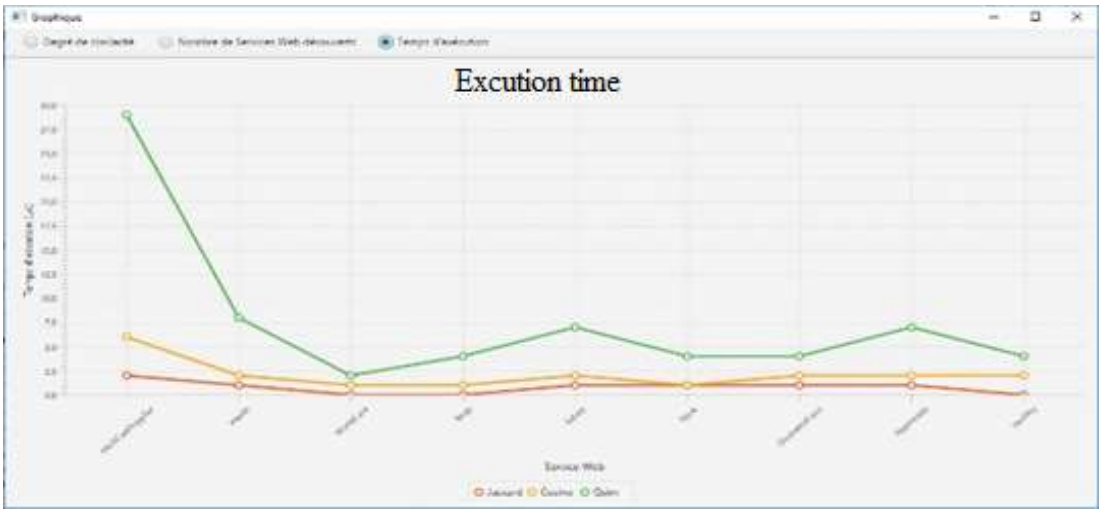

Figure 30 Search response time of user 1.

C) The response time of the Qsim measure is higher compared to the response times of the Jaccard and Qsim measures, and hence the more complex and efficient is the measurement response time increases (see figure 30).

4.8. Conclusion

In this chapter, we present the proposed simulation tool for contextual similarity measures to evaluate similarity measures and determine their effectiveness in choosing suitable services for users in the discovery process. Indeed, our objective was to gathering a dataset of real web services from the internet and realize a platform that enable to compare and evaluate similarity measures with various functionalities as possibility of adding and using new measures, comparing several measures at the same time, graphical representation and storing of the obtained results.

Chapter 5: Improved multi-criteria ranking based Web service discovery approach

5.1 Introduction
5.2 Motivation of the contribution
5.3 Multi-criteria: Definition and the Proposed Solution
5.4 Implementation and Results
5.5 Conclusion

5.1. Introduction

Web services (WS) are software free of any constraint of software or hardware compatibility. Web services have many advantages, they are usable remotely via any type of platform. WS can be used to develop distributed applications and are designed to be accessible from any type of devices. Services are published by providers, in directories, that host their descriptions. They are accessible via a network for customers who discover, select, invoke and use them.

The context is the set of information external to the application that can influence its behavior. This information can be used to characterize the situation of an entity. An entity is a person, device, application or object that may be relevant to the interaction between the user and the application, including the user and the application (service) [81]. With this definition, we can say that each entity (user and Web service) has its own context. The service context can group service location (geographic restriction), service cost, service category, QoS parameters, and so on. The user context can be constituted by its location, preferences, etc. [41].

To optimize the discovery of Web services according to the user requirements, similarity measures are proposed to determine the correspondence between the user context and the Web service context.

5.2. Motivation of the contribution

To look for a service that much the user's Qos requierement, we can uses similarity measures. Some approach uses arithmetic equation that has some lacks sush us low accurency in some situation. We decide to applie a multicriteria ranking approach improved with a fuzzy logic function, in order to choose the service offering the best quality of service between services offering the same functionalities.

Let us have a look on the similarity measures used in previous works,
We associate for some similarity measures a vector $w = (w_1, w, \dots, w_n)$ where w_i represents a weight of an attribute of the context which is between 0 and 1, where: $\sum_i w_i = 1$.

We define also X and Y are two vectors where X is the service context vector and Y is th user context vector and the x_i, y_i are the contxet attributes: $X = (x_1, x_2, \dots, x_n)$ and $Y = (y_1, y_2, \dots y_n)$.

The cosine similarity defined in [84] used to determine the resemblance between the service context and the context of the user.
The cosine measure is defined as follows:

$$Cosine(X,Y) = \frac{\sum_{i=1}^{n} X_i * Y_i}{\sqrt{\sum_{i=1}^{n} X_i^2} * \sqrt{\sum_{i=1}^{n} {Y_i}^2}} \qquad \textbf{Formula/Eqt(10)}$$

JACCARD measure was used in [85] in order to quantifying the correspondence degree between the features required by the client and the ones proposed by the service.
Jaccard is defined as follows:

$$JSM = \frac{a}{a+b+c}$$ **Formula/Eqt (11)**

where:
• a: number of features proposed by the service and requested by the user.
• b: number of features proposed by the service but not requested by the user.
• c: number of characteristics requested by the user but not offered by the service.

In [82], authors proposed a quantitative similarity measure Qsim, based on the previous measure (JACCARD) aiming to fill the weakness mentioned on [85]**.**

The quantitative similarity measure QSim is used in [41] to compare the context of the user and the context of the service. The QSim measure is defined as follows

$$\mathrm{QSim(X,Y)} = \frac{\mathrm{a}\sum_{i=1}^{i=n} w_i \ * ASim(Xi, Yi)}{\mathrm{a}\sum_{i=1}^{\mathrm{i=a}} w_i + \ b \ \sum_{i=1}^{i=b} w_{a+i} \ + c\sum_{i=1}^{i=c} w_{a+b+i}}$$ **Formula/Eqt (12)**

- With a, b, c: same as defined in jaccard.

ASim: is the atomic similarity between each characteristic of X and Y and is defined as follows:
In the case of a single-phase search:

$$ASim(X_i, Y_i) = \begin{cases} 1 & if \quad (X_i = Y_i) \\ 0 & if \ ((X_i \neq Y_i) \wedge (type = Qualitative)) \\ \dfrac{\min(X_i, Y_i)}{\max(X_i, Y_i)} & if \ \ ((X_i \neq Y_i) \wedge (type = Quantitative)) \end{cases}$$ **Formula/Eqt (13)**

In the case of a two-phase search:

$$ASim(X_i, Y_i) = \begin{cases} 1 \ if \ (X_i = Y_i) \\ 0 \ if \ (X_i = 0 \ or \ Y_i = 0) \\ \dfrac{\min(X_i, Y_i)}{\max(X_i, Y_i)} \ else \end{cases}$$ **Formula/Eqt (14)**

Similarity measures defines the degree of resemblance or correspondence between two objects. To compare two characteristics that form a context, it is sufficient to compare their values. But when we have to compare several characteristics, this method is insufficient. Consequently, similarity measures were proposed to determine the degree of similarity between two contexts while taking into account all the characteristics that make up the context [85].

For the mentioned measures, we notice that these measures: are based on mathematic metrics (weighted some, sets operations). In addition of the inconvenient that was mentioned in [82] we add that the context attributes aren't compared pair to pair. That means that the result of similarity measure are not based on comparing each of the user context attribute value to the same service context attribute value.
Example: we set a user and service contexts as (security, fiability, performance).

User1 (5, 2, 1) Servicei (3, 1, 5) with the weight of (2, 2, 2) respectively.
For this case, the previous measures give us an ambiguous result: like Qsim gives a 43% match when the service contexts can satisfy the user request. Because the user requested for 1st attribute the value 5 when the service offer the value 3 that is not far from the user value also for the second attribute and for third one the service value is more than satisfying for the user request.
Same case for this scenario User2 (1, 2, 2) Servicek (4, 4, 5), weight (3, 3, 3).
In this case, Qsim gives a 38% similarity result when the service is very satisfying for the user request.
For this reason, we decide to look for a solution to solve these problems. This type of problems are known as a multi-criteria or multi objective problems.

5.3. Multicreteria: definition and the proposed solution.

A. Rolland define in [86] as: "A classical problem in multicriteria decision making (MCDM) is about aggregating preference relations on each criterion to obtain a global preference relation on the set of the considered alternatives. Several axiomatic papers show the theoretical and practical difficulties in aggregating preference relations which are partially in conflict".

Two different ways exist to aggregate the different values of the criteria of two alternatives to compare:
- The "aggregate then compare" approach: For each alternative, we aggregate all the values of the criteria to obtain a unique "score". It is then sufficient to compare two alternatives by comparing their respective scores. The theory of methods based on the definition of a single synthetic criterion (a utility function, or Multi Attribute Utility Theory, MAUT) has been developed since the 1970s mainly in the United States [87]. This is the method used when averaging, possibly weighted, students' grades before comparing them to each other. This approach is relatively simple and intuitive in the context where the alternatives are in finite or countable numbers. As this approach is totally compensatory, it is necessary to know which performance on a criterion can compensate a performance on another criterion. This information is not always obvious to obtain, especially in the case where the criteria are evaluated on different scales [88].

- The "compare then aggregate" approach: On the contrary, it is a matter of comparing, criterion by criterion, the two alternatives in order to obtain as many relations of partial preference between the two alternatives as there are criteria, and then to try to aggregate these partial preferences into one. overall preference using an aggregation procedure similar to a vote. The ELECTRE method, proposed by [89], is the oldest method of multicriteria decision support using this principle. This method has seen many variations and practical developments in decision support [88].

A. Rolland has proposed an interesting method to compare a list of student according to their marks using reference point.
According to A. Rolland [88], the use of reference points is already present in the field of multi-criterion decision.
He propose to use reference points to help compare alternatives indirectly; indeed, the direct comparison of the two-to-two alternatives induces preferential relations that are not necessarily transitive: this is the Condorcet paradox.

Initially, A. Rolland approach is based on a set of rules that control the ranking. Every student in the list is compared to the entire list based on the criteria of preference previously fixed: the note of course, report of interview, complete file and opinion of head of establishment. For a student 'A' to be better ranked than the student 'B' it is necessary that 'A' defeat 'B' on 3 criteria Minimum and the number of criteria in favor of 'A' is higher than 'B'.
Then the students are ranked using the score of each student that is calculated with the number of students defeated minus the number of the students defeated by them.

Same way the service ranking compare each service context attribute with all the rest of services that result in the syntactic search. But, we consider a service 'A' better ranked then 'B' if only if 'A' defeat 'B' in just one criteria (we think that is sufficient to consider a service 'A' is better than 'B' if the other attributes are even) and the number of criteria in favor of 'A' is higher than 'B'. We take as reference point the user preferences.
We can encounter an ambiguous case, like when service 'A' is better in criterion C1 and C2, in the same time 'B' is better in C3 and C4. In this case, it is very handy to use the weight set by the user on each criterion.

The ranking rules:
We set some variables as follow:
The Web service context is a vector of values $\{X_1(k), X_2(k), \ldots, X_i(k)\}$ where $i \in lN$; and $X \in \{S, Rs, U\}$.

Si: is the i^{th} attribute value of the context of the selected services in the syntactic[13] search.
Rs_i : is the i^{th} attribute value of the ranked service context.
U_i : is the i^{th} attribute of the user context.
Rank: is the score obtained by the ranking algorothim for each service.

We define the following ranking rules:
1- If $Rs_i = Si = U_i$ Then no added value.
2- If $Si < Rs_i < U_i$ Then Rank ++;
3- If $Rs_i = U_i$ and $Si < Rs_i$ Then Rank ++;
4- If $Rs_i = U_i$ and $Rs_i < Si$ Then no added value.
5- If $Rs_i > U_i$ and $Rs_i > Si$ Then Rank++;
6- If $Rs_i < Si <= U_i$ Then Rank --;
7- If $Rs_i < U_i$ and $U_i < Si$ Then Rank--;

These are the rules for services ranking according to the approach proposed in [88]. However, there is another case which can generate unfair ranking results. We take as an example two services 'A' and 'B' that are compared on set of criterion C when:

C={C1,C2,C3,C4};
C(A)={4, 4, 4, 6} and C(B)={4, 4, 3.9, 4.8}.
C(U)= {4, 3, 4, 5}.

We remark that services ‘A’ and ‘B’ are equal in criterion C1 and C2 and satisfy U. Although, for C3 and C4 ‘A’ satisfy U and for ‘B’ it is not the case when following the rules (rule 7). While its values of C3 and C4 are very close to the one requested by the user. In this case, the algorithm eliminate this service when its attribute values are very close to the required user values.
Taking into consideration the difference between the attribute values in the ranking process is very interesting.
Therefore, we propose to use a fuzzy logic method to consider the difference between the reference point and the candidate service to solve the problem presented in the previous example.

Some performance differences are not necessarily significant for conferring a preference between two profiles Rs and Ss on a criterion i. This is why we introduce a preference model with two thresholds: pj and qj (pseudo criterion), where:

- qj corresponds to the maximum performance difference, compatible with a total indifference between Rs and Ss on the criterion j; it is called the indifference threshold. Where: $Rs_i = Ss_i$ if $| Rs_i - Ss_i | <= qj$;

- pj corresponds to the minimum performance difference totally incompatible with the indifference on criterion j, it is called the discrimination threshold.
  Where: $Rs_i > Ss_i + pj$

So we denote as next:
$g(X_i)$ a functoin that retrive the value of the atribute X_i from the context Profil $X\{X_1(k), X_2(k), \ldots, X_i(k)\}$ where $i \in lN$; and $X \in \{S, Rs, U\}$.

Rs is indifferent from Ss on criterion j : if $|g(Rs_i) – g(Ss_i)| <= qj_i$
Rs is different from Ss on criterion j : if $g(Rs_i) >= g(Ss_i) + pj_i$
Rs weakly indifferent from Ss on criterion j : if $qj_i < |g(Rs_i) – g(Ss_i)| < pj_i$

In [90], Henriet used a coalitions function to avoid the strict preference model. This makes it possible to obtain a gradual transition between the thresholds qj and pj and gives the criterion the possibility of having a gradual membership of the coalition. Thus, a very weak performance difference cannot radically change the membership of a criterion in the coalition.
We define the indifference function by:

$$C_j^{\sim}(x,y) = \begin{cases} 0 & \text{if } x_j - y_j \leq -p_j \\ \frac{1}{2} + \frac{1}{2}\sin\frac{\pi}{p_j - q_j}\left(x_j - y_j + \frac{p_j+q_j}{2}\right) & \text{if } -p_j \leq x_j - y_j \leq -q_j \\ 1 & \text{if } |x_j - y_j| \leq q_j \\ \frac{1}{2} + \frac{1}{2}\sin\frac{\pi}{p_j - q_j}\left(x_j - y_j - \frac{p_j+q_j}{2}\right) & \text{else } q_j \leq x_j - y_j \leq p_j \\ 0 & \end{cases}$$

Formula/Eqt (15)

Where x and y are the context attribute values that we intend to compare using the thresholds qj and pj.

We use the defined function to calculate the value added to ranking score of each service selected in the syntactic search instead of add a static value by taking in consideration the distance between the attribute of each pair of services.
So, in the case of Rank++ : we replaced it with:
Rank = Rank + $C_j(Rs_i, Ur_i)$; where C_j is the indifference.

And when we have Rank -- : we replace it with:
Rank = Rank - $(1-C_j(Rs_i, Ur_i))$;

With this, we calculate the rank of each service regarding the differences between all the context attributes.

5.4. Implementation and results

For the implementation and tests we use the tool developed previously to evaluate similarity measures [91], this tool has a dataset of about six hundred Web services. Each sevice has a WSDL description file and a CC/PP[14] context profil. We used the java development language in Netbeans IDE 8.1 environment to implement the proposed approach.

We used a labtop with 8GB RAM, Intel Core(TM) i5 4310U @ 2.00 GHz Vpro and 500GB HDD.

For the functional information, a syntactic search is used. Then we run the contextual selection approach on the result list of the syntactic search.

We applied our approach on the dataset and we obtaned results using two users context profiles:

Table 5 User's preference and weights.

Attribute/user	USER1(SMS)		USER2(SMS)	
	Value	Weight	Value	Weight
Language	FR	1	FR	1
Localization	Paris	1	London	1
Screen	15"	1	15"	1
SE	Windows	4	Windows	4
Security	4	4	2	1
Fiability	2	3	3	4
Performance	2	2	5	4

Table 5.2 shows the sorted result of syntactic search of 'SMS' services.
In table 5.3 we used the Qsim measure to compare and validate the proposed approach. We also compare the improved ranking and the simple one.

Table 6 Services selected in syntactic search.

	SERVICE CONTEXT	LANG	LOCATION	SCR	SE	S	F	P
1	SendSMSWorld	Anglais	Paris	17	Windo	4	2	2
2	AdvancedSendService	Arabe	Londres	17	Windows	5	2	4
3	SendSMS1	Anglais	Alger	15	Windows	5	5	4
4	SendService	Arabe	Londres	15	Windows	3	5	4
5	SmsGate2	Anglais	Paris	15	Windos	2	4	1
6	SendMessages	Anglais	Alger	17	Windows	3	3	4
7	Service1	Anglais	Londres	19	Windows	5	3	5
8	SendServiceNoHeader	Anglais	Alger	19	Windows	3	3	5
9	SMS	Arabe	Paris	19	Windows	0	2	3
10	ServiceSMS	Français	Paris	17	Windows	1	1	5
11	AccountServiceNoHeade	Anglais	Londres	15	Mac	4	5	2
12	eCall	Arabe	Alger	15	Mac	4	3	1
13	ScheduledSendServiceNo	Français	Paris	17	Mac	2	4	2
14	FileShopHandler	Arabe	Alger	15	Mac	3	2	5
15	AccountEventHandler	Arabe	Londres	15	Mac	4	4	5
16	Lodge	Français	Paris	15	Mac	1	4	2
17	insuranceauthority	Français	Londres	15	Linux	2	1	1
18	Service1	Arabe	Paris	17	Linux	1	2	3
19	TA_SMS_RB	Français	Paris	19	Linux	3	5	0
20	SendSMS2	Français	Paris	17	Linux	2	0	2
21	SendContentServiceNoH	Anglais	Alger	19	Linux	4	3	2
22	checkcompany	Arabe	Londres	19	Windows	1	3	0
23	MScienceSMSWebServi	Français	Alger	19	Mac	1	1	5
24	TextAnywhere_SMS	Français	Londres	19	Linux	5	1	0
25	ContactServiceNoHeader	Arabe	Alger	19	Linux	5	4	1
26	ContactService	Français	Alger	19	Linux	5	0	3
27	WebServices	Anglais	Alger	17	Linux	3	4	1
28	ScheduledSendService	Anglais	Alger	17	Mac	2	3	1
29	InboxServiceNoHeader	Arabe	Alger	17	Linux	2	4	5
30	SmSService	Anglais	Alger	15	Mac	0	5	5
31	InboxService	Anglais	Londres	19	Linux	4	0	3
32	SendWDPServiceNoHea	Anglais	Paris	15	Mac	0	0	5
33	SendUTF8	Arabe	Alger	15	Mac	0	1	0
34	ASPSMS_x0020_SOAP	Arabe	Paris	17	Mac	0	0	4
35	SendMessagesEx	Anglais	Londres	17	Mac	1	0	0

The next table shows the results of the contextual discovery using the Qsim mesure and the ranking approach with and without the indifference function Cj. The services are sorted in a ascending way (the best match is the result number one).

The table contain the ranking results for the User1 and the User2 profiles and the execution time for the Qsim and the ranking approaches, there is no significant difference between the Rank and the Rank Cj so we keeped one result for the two of them.

Table 7 Services ranking results.

	SERVICES	Qsim			Rank		RankCj		
		U1	U2	Time (µs)	U1	U2	U1	U2	Time (µs)
1	SendSMSWorld	1	11	3	5	16	1	15	8618
2	AdvancedSendService	2	2	5	3	8	7	3	6077
3	SendSMS1	3	3	5	1	3	6	6	10198
4	SendService	4	4	72	4	1	2	4	19024
5	SmsGate2	5	7	5	10	10	5	16	5129
6	SendMessages	6	8	5	7	6	4	5	15809
7	Service1	7	1	5	2	2	10	1	6169
8	SendServiceNoHeader	8	6	3	6	4	3	2	8966
9	SMS	9	19	5	17	18	16	10	13841
10	ServiceSMS	10	5	3	16	11	15	7	4027
11	AccountServiceNoHeade	11	14	3	8	12	8	19	5169
12	eCall	12	18	5	14	22	11	18	6596
13	ScheduledSendServiceN	13	17	5	15	15	18	25	8526
14	FileShopHandler	14	13	4	13	13	9	8	5963
15	AccountEventHandler	15	9	5	8	5	12	9	5629
16	Lodge	16	12	5	18	14	22	20	3790
17	insuranceauthority	17	16	3	31	27	19	22	7671
18	Service1	18	24	3	27	28	29	17	7570
19	TA_SMS_RB	19	28	3	19	20	14	27	6878
20	SendSMS2	20	31	4	29	32	21	34	11976
21	SendContentServiceNoH	21	22	3	11	19	13	21	5868
22	checkcompany	22	10	3	23	17	25	13	8467
23	MScienceSMSWebServi	23	15	5	30	21	30	14	5469
24	TextAnywhere_SMS	24	25	5	26	31	23	28	6233
25	ContactServiceNoHeader	25	27	3	12	23	24	29	6027
26	ContactService	26	30	3	20	29	27	31	7059
27	WebServices	27	26	5	24	24	17	30	5760
28	ScheduledSendService	28	23	5	28	25	26	24	9695
29	InboxServiceNoHeader	29	20	4	21	9	28	12	7419
30	SmSService	30	21	3	22	7	31	11	3859
31	InboxService	31	29	4	25	30	20	32	8653
32	SendWDPServiceNoHea	32	32	3	32	26	32	23	4787
33	SendUTF8	33	33	3	34	34	33	33	5582
34	ASPSMS_x0020_SOAP	34	35	3	33	33	34	26	6247
35	SendMessagesEx	35	34	5	35	35	35	35	8943

The results show that the RankCj approach give more accuracy and reliability depending on the case.

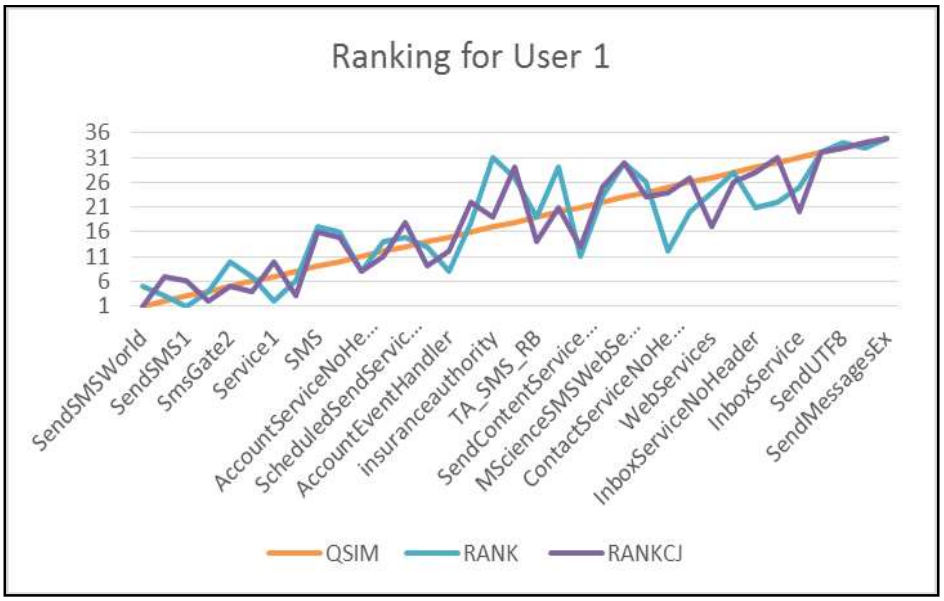

Figure 31 Ranking result graph for User1.

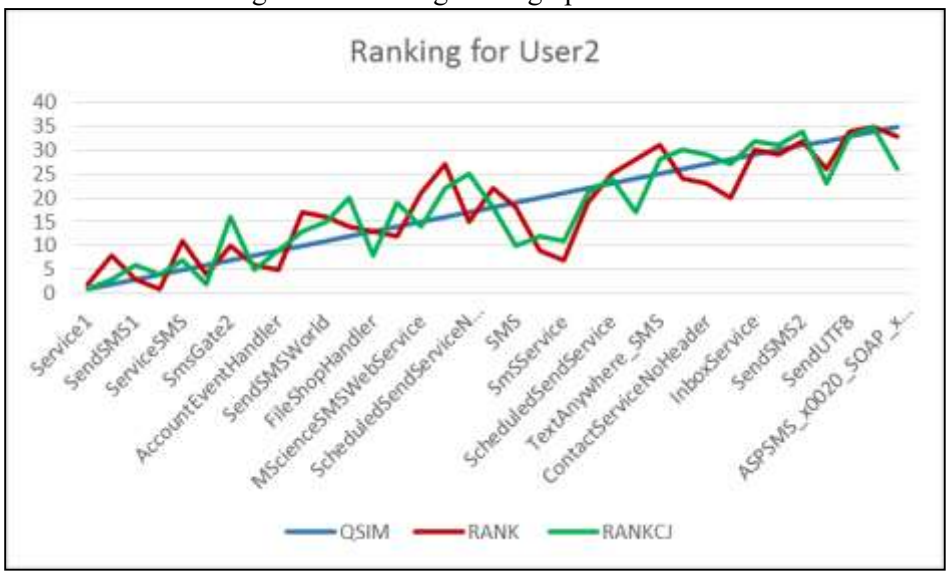

Figure 32 Ranking result graph for User2.

For the tables and graphs, the services that has smallest ranking value represent the best choice.

The results show that in most cases, when a service do not have the exact requested context, but it's values are nearby, the improved ranking approach give a better rating then the normal ranking (the case of service number 4, 6, 8, 14, 19).

For service number 4: SendService (Arabe, Londres, 15, Windows, 3, 5, 4) the **Qsim** and **Rank** gives a ranking of 4 when **RankCj** gives a ranking of 2. Same thing for service 6: SendMessages (Anglais, Alger, 17, Windows, 3, 3, 4) **Qsim:** 6 and **Rank:** 7 when **RankCj** gives 4 for User1 and a close result for User2.

In the other hand, our approach gives also a significant rating for services that are not even near to the user requested context (service number 18, 23, 26, 34, 35).

For the respond time, there is a significant difference between the arithmetic measure and the Ranking based measure, which is normal and still under the one second limit for each service.

5.5. Conclusion

Our work consist on using the ranking based approach used in [88] as a solution for the context selection multicriteria problem, then we improve the accuracy of this approach by using the coalition function to calculate the difference between the attributes values.

Our approach aim to fill in the weakness in the arithmetic approaches using the ranking approach in [88] of the multicreateria field and improving it with a fuzzy logic function used in [90] that considerate the difference between the attribute values.This work give us more accuracy in selecting Web services based on their context and help in some ambiguous situation.

Chapter 6: Contextual fuzzy ranking for Web services discovery in a hybrid architecture

6.1. Introduction

The conventional method of designing applications has significantly evolved thanks to recent developments in service-oriented computing. A wide range of storage, analytics, computing, and deployment Web services are used in modern application development.

Cloud services (such Platforms as Services PaaS, Databases as Services DBaaS, etc.) and APIs are examples of Web services (such as Maps API, Amazone cloud services, Facebook API, etc.).

Employing Web services for application development has several benefits, including the ability to (I) reuse previously developed capabilities, which is easier than trying to recreate them, (II) allow users to consume cloud resources (memory and processor) at scale, which enables dealing with unpredictable user demand during pick periods and (III) manage costs effectively because users only pay for what they use (also known as Pay as you go).

There is a growing need for context-aware applications, as the increasing number of deployed services come with comparable functionalities and quality of service (QoS). That makes it difficult to recommend the best services to developers. The context discovery offer much better user experience and adaptive features compared to traditional web application. However, the current techniques used for contextual discovery and QoS-filtering based on static information do not support dynamic changes and rate uncertainty. In addition, the required computational resources are expensive due to large number of services and high accuracy of requestors. The current study aimed at providing a solution that could overcome those problems.

6.2. Motivation and objectives of the contribution

The used discovery technologies vary from an aspect to another: from registries deployment (centralized or distributed) to functional discovery (using keywords or semantic search) and non-functional discovery (using the context and the Quality of Service information) to get more precise and accurate results. The service discovery process is a key element in the Service-oriented architecture (SOA), although the increasing number of services put the reliability of the used technology of the discovery process in question.

The main aim in this contribution is to propose a hybrid Web service discovery architecture with an objective of a platform that can replies to the growing SOA requirements by offering an effective service discovery in terms of efficiency, time response, and search quality.

This hybrid architecture is based on a multiagent system (MAS), including a discovery process based on a contextual fuzzy ranking approach. The proposed architecture aims to solve some of weaknesses in the discovery process like latency, overload failure and network congestion. It is based on a hybrid network using a single access point (Gateway), offers a solution for managing the increasing number of services using multi-level domain service distribution and an agent platform that manages the different tasks as publishing and discovery in a paralle and light way.

The discovery agent implement a contextual fuzzy ranking approach. This approach helps to choose the service offering the best level of quality parameters between services that have the same functionalities, which shows a promising result.

6.3. Background and related works review and discussion

In this section, we take an overview of different Web service discovery approaches focusing on registries architecture and the contextual discovery of the non-functional parameters. Three main architectures are proposed: centralized, distributed and hybrid.

Centralized architectures are simple, easy to implement and does not require a lot of resources. However, these solutions have a single failure point and problem of scalability. Compared with traditional centralized client-server architectures presented previously, distributed architectures improve scalability, heterogeneity and robustness in dynamic and open environments. However, unstructured P2P architectures present some problems like discovery loops: uncontrollable number of hops for reaching the wanted resource, network congestion because of the flooding and the inconsistency in the information, so the merits of DHT structured P2P architectures are mainly reflected on its high scalability in message routing. Moreover, a search in an unstructured P2P architecture, basically has no information about the location of the resource, and therefore generates significant traffic overhead when using the flooding method to locate services and all corresponding responses are sent to the originating peer through the reverse path.

Blind flooding is still the simplest and the most frequently used method for resource discovery in P2P networks. As a result, it is difficult to achieve high routing scalability when employing unstructured P2P approach for service discovery [73]. However, proper maintenance of a DHT or logical overlay among the peer nodes is not a trivial task in dynamic environments where peer nodes may frequently join or leave P2P networks, and thus incur significant maintenance costs. This makes it difficult to achieve high maintenance scalability of the DHT to use a structured P2P approach for service discovery. Also due to the hash characteristic, DHT-based systems can only support keyword searches.

DHT has become the dominant methodology for service discovery in a distributed environment. DHT enables efficient service discovery in P2P networks, requiring only a small number of messages to resolve a query. However, efficiency depends not only on the number of messages to resolve a query, but also the number of messages to maintain the network. DHTs are not efficient in a dynamic environment, because it requires a large number of messages to maintain the network topology in order to keep consistent hash tables at each peer node, although queries can be resolved by a few messages. In addition, load balancing on the DHT is a well know problem, as certain information can be accessed more frequently. In summary, any additional control attempts may be difficult to achieve in the distributed P2P architecture due to the lack of a centralized server [73], [67], [62].

Compared with indexed structured P2P approach, the semi-structured P2P approach has smaller overhead to maintain the routing table entries updated [62].

Compared with unstructured P2P architectures, semi-structured P2P architecture has better message routing by avoiding the flooding within the service discovery.

The main challenge in self-organized architectures is to define a P2P social network where the overlay connections are self-organized based on the social interactions, without centralized control to guarantee the service availability and the efficiency of service discovery.

Structured P2Ps are built according to a well-defined geometric structure and a deterministic links between nodes. Thanks to these properties, they can offer guarantees regarding the response time, scalability and can also decrease the maintenance cost.

The decision to use Hybrid P2P networks in this class is justified by several reasons. Hybrid P2P approaches are scalable solutions compared with centralized approaches.

Hybrid P2P networks are designed to implement a search by content, generally using one or more keywords. While these features are not provided by the structured P2P systems, where to find WSs, it is necessary for the applicant to know in advance the object identifier.

The role of Super Nodes varies from an application to another. However, they are generally used to perform the functions related to locating, routing or organizing Child Nodes. To avoid the security and point failure problems like we find in the centralized approaches so, replications and security protocols must be implemented.

This architecture reduces the number of messages received by a large number of Child Nodes like in the case of pure p2p, increasing in the other hand the load of Super Nodes. Hybrid P2P systems are used by very large online communities. Thus, the Hybrid P2P systems are more suited to distributed discovery architecture [71]. It is a proven practical solution, but are more complex to implement.

In order to improve the discovery process, the authors propose a pyramidal architecture of registries to organize services with a fuzzy contextual ranking. In the following section, the authors present the Web services platform based on a multi-agent system.

6.4. A pyramidal registries architecture based on a multi-agent system

The proposed registry architecture is a hybrid P2P architecture, where registries are classified by domain, and if necessary, by subdomains. A domain represents a class of Web services, for example "Commerce", "Transport", etc. These classes can also contain several subclasses, for example "Commerce-> Industry> Automotive, etc.", as shown in Figure 33.

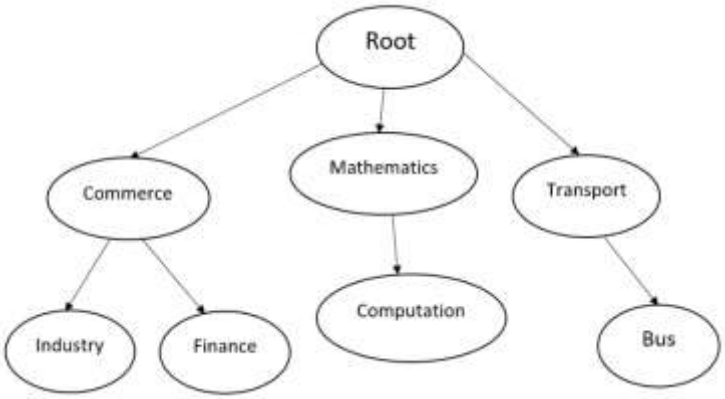

Figure 33 Registries classification.

Each node represents a service class, which allows to classify services that have a similar application domain in the same cluster. This approach will avoid network congestion when the user searches for a Web service, minimizing the number of messages to be routed in the network. Searching for services, consists of locating, in the first stage, the classes and subclasses that the user has chosen, and then locating the appropriate services in the class registry. This will allow to send the request to the correct registry, and thus to process a small number of services descriptions, in order to find those that match the client's needs.

However, as the number of classes, and thus the number of registers, increases, scalability becomes an important issue. For this, the paradigm of P2P networks, which have properties such as autonomy and scalability, perfectly meets the desired requirements. Since each peer is independent, it may have different roles in the network.

The proposed registry architecture is a super-peer architecture. It consists of three types of peers, namely gateway peer, index peer and registry peer, as shown in Figure 34.

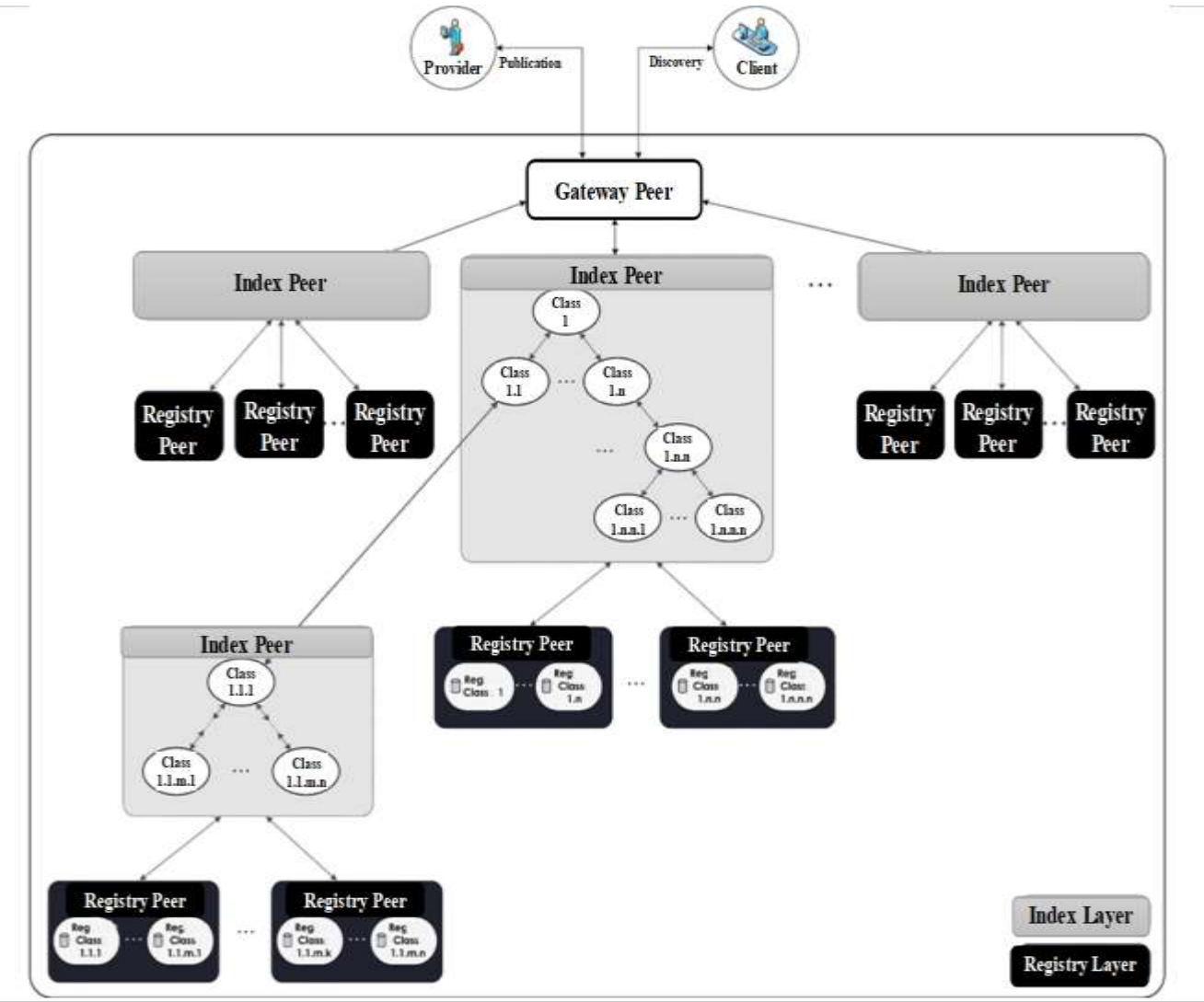

Figure 34 The proposed architecture.

The gateway peer is a super-peer compared to peer-level indexes, which are in turn super-peers relative to peer registries and possibly relative to other peer child indexes, and so on. Super-peers act as centralized servers to their peers, allowing them to handle queries more efficiently. Moreover, since there may be many indexes in this kind of architecture, no peer index will be forced to handle a large load.

Access to the system, to perform a service publication or a service discovery, is done each time by a super node (gateway peer). Thus, the system appears to the users as being centralized, since peer use is transparent, and the user is not aware of distributed implementation.

The proposed architecture is a hybrid solution for the service discovery in P2P systems, where some resources or processes are centralized in each super-peer (such as index information about child peers), while others are distributed (such as registries). As a result, the solution benefits from the efficiency of a centralized search (autonomy), as well as the distribution of loads and the

robustness provided by a distributed search. Figure 35 represents the meta-model of the proposed architecture.

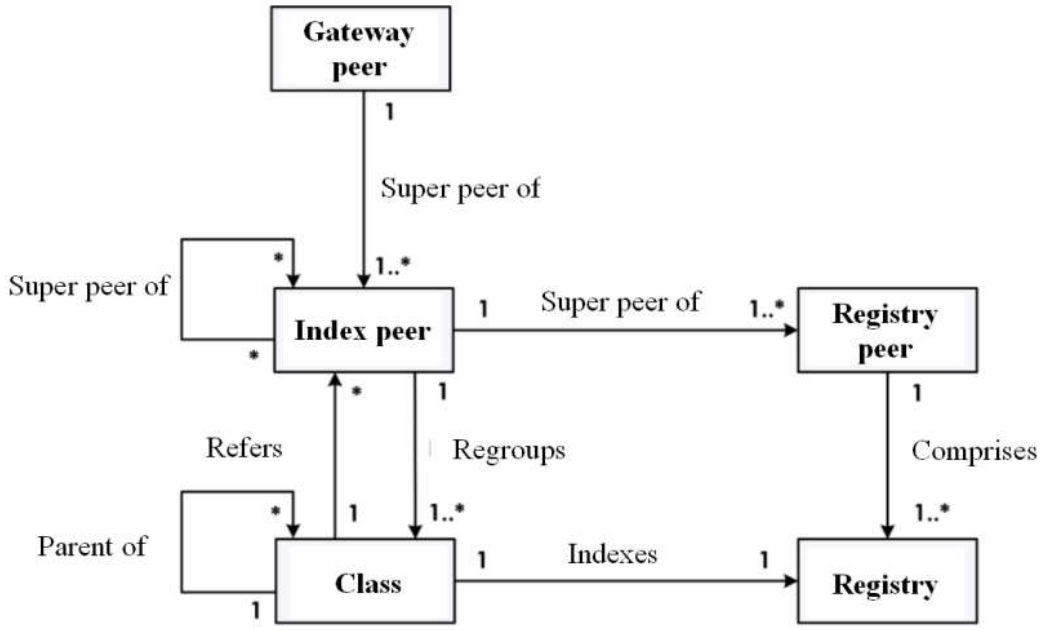

Figure 35 Meta-model of the proposed architecture.

The proposed architecture is supported by a multi-agent system (MAS). We have defined software agents in each peer, in order to support services publication and discovery needs. As a result, various roles have been defined, which gives the desired autonomy, since each of these peers will be able to ensure various well-defined treatments.

Furthermore, the proposed solution considers the user's context as well as the service's context during the discovery process. As a result, users can obtain services that best suit their contexts.

In the following, we present details about the role of each component of the proposed architecture, then a presentation of the implemented MAS.

6.4.1. Component's role

In the proposed architecture, all components are implemented as peers (Figure 35). Below the role of each element is explained.

1. **Gateway Peer:** The Gateway Peer allows interaction between users and the system to publish and discover services. It can be considered as the root node of the class tree, as shown in Figure 35 Thus, it is important to store the tree that describes all the existing classes and subclasses, as well as the description of each of these classes, in order to assist the user during the publication and discovery processes. This peer also stores index information about its child peers. This information will be necessary for routing discovery/publication requests to the correct registries. In addition, the Gateway Peer contains a registry, used to store data about services' providers, as well as information about clients (preferences and context). Tasks performed in the Gateway Peer are handled by different agents.

2. **Index Peers**: Index Peers are used to keep indexes organized in a tree form (Figure 35), so that they can organize and retrieve the registries according to the established classification. Index Peers represent the index layer of the system and are used to route the publication/discovery queries to the appropriate peer registries. Therefore, Index Peers can be considered as intermediate peers between the Gateway Peer and Registries Peer. The internal structure of the Index Peer uses different agents to perform different publication/discovery operations.
3. **Registries peers:** The Registries Peer layer is fundamental and it contains the different registries. Each registry is indexed by a class and maintains only descriptions of Web services that are related to that class. Each registry has an UDDI directory having the same role as the traditional Web services architecture (registration of services descriptions).
   In addition, and in order to return to the user a list of WS adapted to his context, the registry uses a database to store services contexts.

To fulfill its role, this layer employs a set of agents, with specific roles, which allows to have a better quality in terms of performance.

6.4.2. The Multi-agent system

The proposed registries architecture is supported by a multi-agent system that brings advantages among which the distribution of tasks on several agents instead of a single program that manages the whole. It allows the parallelism of different tasks as well as the creation of several instances dynamically of each agent according to the arrival of requests.

The multi-agent system (MAS) is composed of ten different agents, namely:

- The Gateway Admin Agent (GAA);
- The User Manager Agent (UMA);
- The Client Publication Agent (CPA);
- The Client Discovery Agent (CDA);
- The Admin Index Agent (AIA);
- The Unifier Publication Agent (UPA);
- The Unifier Discovery Agent (UDA);
- The Registry Admin Agent (RAA);
- The Discovery Agent (DA);
- and the Publication Agent (PA).

As explained before, each of these agents is part of one of the architecture components, as shown in Figure 36.

1. Client Publication Agent (CPA): This agent receives publications queries from providers and directs them to the most suitable index pair.
2. Client Discovery Agent (CDA): This agent receives discovery requests from clients and directs them to the most suitable index pair. At the end, the CDA returns to the client the list of discovered Web services.
3. User manager agent (UMA): this agent manages information about providers and clients looking for services. Uma registers the information about the request in a directory when

UMA receives a registration request. This information is used to verify users' ids when they login, and to find their contexts and preferences. Uma will create an instance of the ACP or ACD agents for each login request (respectively for a provider or a client).

4. Gateway Admin Agent (GAA): This agent can be considered as the platform's interface agent. It is always listening to registration and connection requests from providers and clients. This agent will create an instance of the UMA for each request.
5. Unifier Publication Agent (UPA): The UPA is located at the peer index, it receives publication requests and then verifies whether the class registry, designated by the provider, is indexed by the current peer in order to route the request to its registry, if not to forward it to the child peer index. Thus, it guarantees the routing of the publication request to the adapted registries.
6. Unifier Discovery Agent (UDA): Like UPA, the UDA is located at the peer index. It receives discovery requests and then checks whether the class registry designated by the client is indexed by the current peer or if it must transmit it to the child peer index. Thus, it guarantees the routing of the discovery request to the correct registries. This agent also performs a quantitative matching to returned list of services by the DA and then communicate the final list of selected services to the gateway peer. More details will be given about the matching process in service discovery explication.
7. Admin Index Agent (AIA): This agent listens for publication and discovery requests sent to the index peer. It creates an instance of an UPA or an UDA agent for respectively each publication and discovery received request.
8. Publication Agent (PA): The role of this agent is to receive publication requests and to save the descriptions and the contexts of the Web services in the class registry designated by the provider.
9. Discovery Agent (DA): The role of this agent is to receive discovery requests and to select services descriptions in the UDDI directory. Then, the DA retrieves the context of these services from the contexts database. Before sending the results, the DA performs a matching on context qualitative attributes, eliminating many services that do not match the client's context. More details on this process are presented in section three.
10. Registry Admin Agent: This agent listens to publication discovery requests, sent to the registry peer. It creates an instance of a PA or DA for each received request.

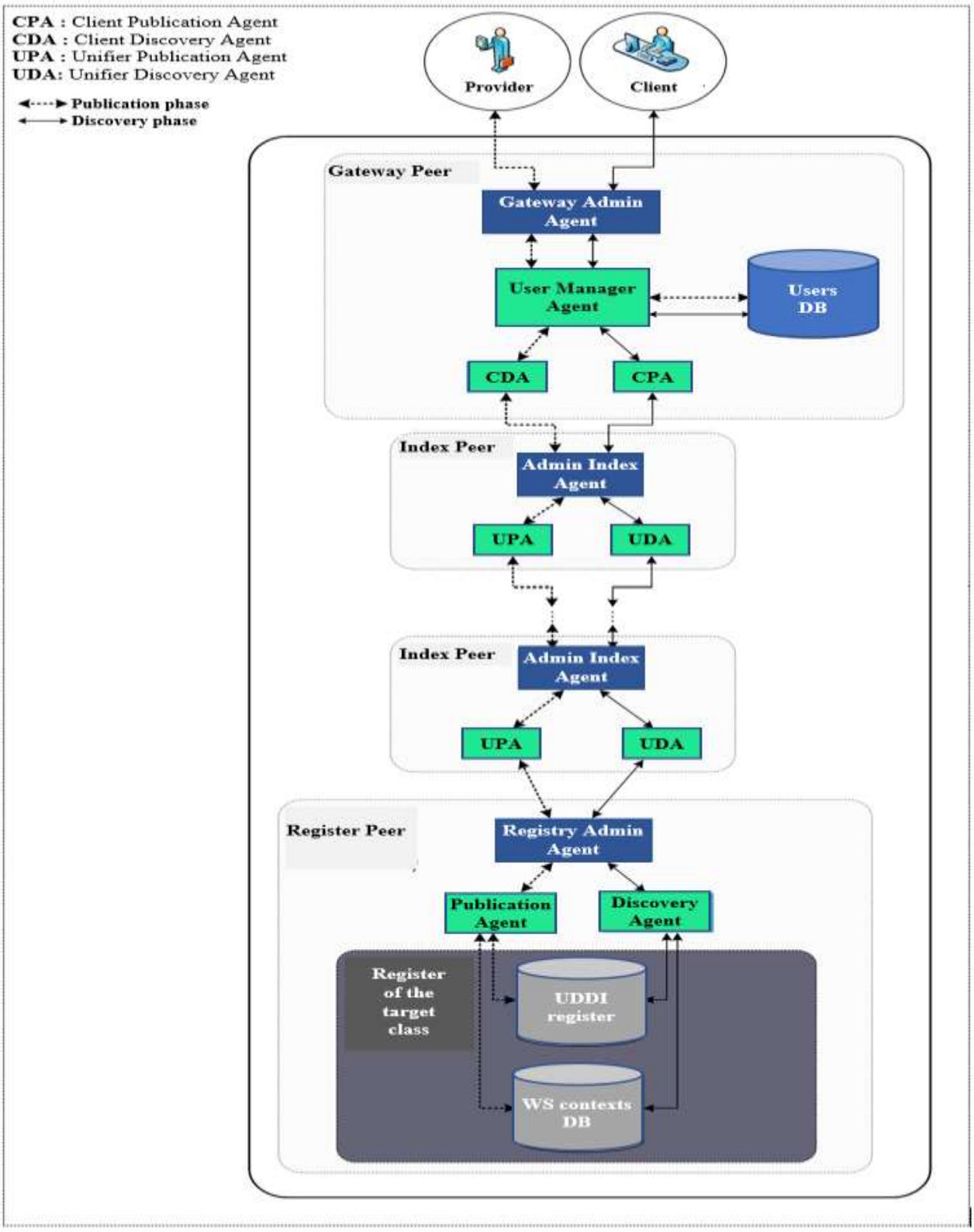

Figure 36 The MAS architecture.

6.4.3. Publication of services

The publication of services is based-on three main actions.

- First, the provider selects the class in which he wants to publish his service;
- He provides a description of his service to store it in the UDDI directory. The description will be used during the discovery process to find relevant services in relation with the client's request;
- Finally, the provider publishes the service's context. The provider describes the context containing information about the non-functional parameters of the service.

The service context is used during the discovery phase to find services having contexts corresponding to the context of the user. A publication request gives the functional description of the service, as well as the nonfunctional description. The publication phase involves interactions between different peers of the platform. Figure 37 shows these interactions.

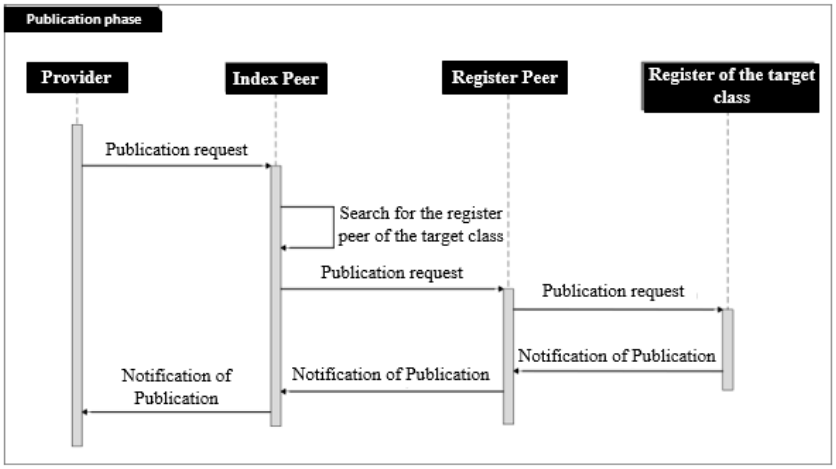

Figure 37 Service's publication diagram.

The following scenario illustrates the different steps of the publication process.
Figure 38 illustrates the different steps in the process:
Scenario: Consider a provider who wants to publish a service. The provider chooses to publish the service in the class "Class1.1.2".

Step 1: The provider publishes, via his account, the service. The *Gateway Admin Agent* will creates an instance of the *User Manager Agent*, to verify the provider's information and create an instance of the *Client Publication Agent* which is attached to the provider throughout his session.
Step 2: The provider chooses to publish his service in the "Class1.1.2" and initiates a publication request by communicating the description and the context of his service.
Step 3: The CPA receives the publication request, and searches in the routing table, the address of the peer index to reach the class "Class1.1.2". When the address is retrieved, the CPA sends the publication request to the index peer.
Step 4: The *Admin Index Agent* receives the incoming publication request, sent to the peer index, and creates an instance of the *Unifier Publication Agent* and sends it the publication request.

Step 5: The created UPA analyzes the publication request and scans the index tree to retrieve the address of the target class (Class1.1.2). In the considered scenario, another index peer that is referenced by the "Class1.1" indexes the target class registry. The UPA retrieves the address of this peer index and sends it the request for publication.
Step 6: The Admin Index Agent receives the incoming publication request, sent to the peer index, and creates an instance of the UPA and sends it the publication request.
Step 7: The created UPA analyzes the publication request and scans the index tree to retrieve the address of the target class (Class1.1.2). At this level, the index is located locally, so the UPA can retrieve the address of the peer registry that contains the registry of the target class and sends it the publication request.
Step 8: The *Registry Admin Agent* receives the publication request, sent to the registry peer, creates an instance of the *Publication Agent*, and sends it the publication request.
Step 9: The *Publication Agent* performs the publication of the service description in the UDDI directory.
Step 10: The PA saves the service context in the contexts database.

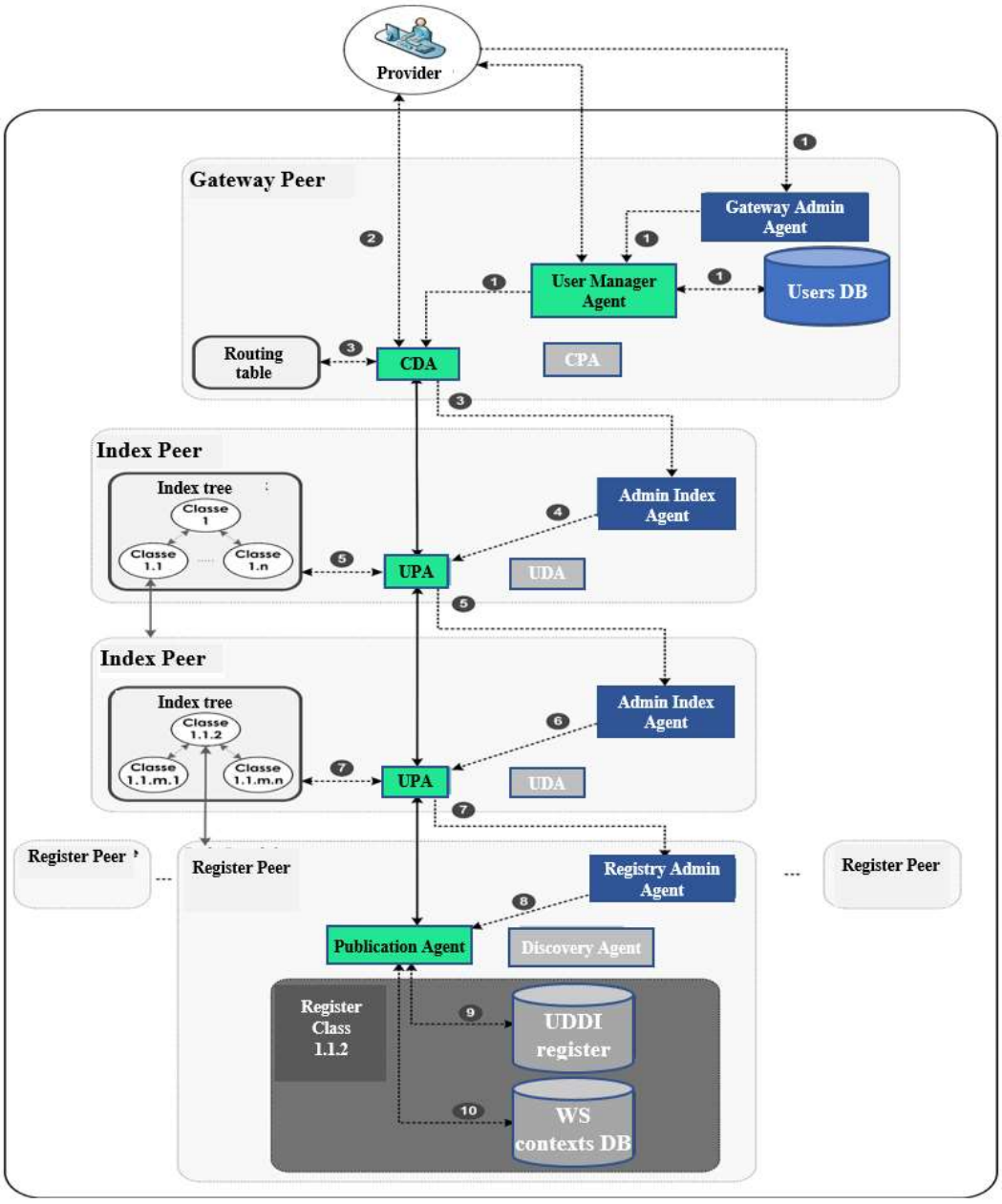

Figure 38 Publication process.

6.4.4. Discovery of services

During the discovery phase, the client searches for relevant services, according to his request and adapted to his context. Indeed, before sending the request, the client must explicitly define certain information and preferences about his context, this context is saved in a user directory. The search for services responding to the client's request and context is implemented according to the following two steps:

- The first step is to search for services, in the UDDI directory, using keywords describing the functional parameters of services (WSDL), following the traditional search method;

- The second step is to look for the services' contexts, discovered during the first step and to rank these services using a fuzzy ranking approach based on a reference point in order to keep only the best services.

In the proposed platform, two types of discovery are distinguished:

- In the first one, the client can search in a specific leaf class of the tree as well as in all its subclasses. In this scenario, the search will be performed in the registry peer of the selected leaf class and will be propagated to all its child classes. This scenario can happen when the client does not want to have to choose subclasses and wants to have a list of services based on: his preferences, services of the class and its subclasses. The client will choose one of the services from the final list.
- In the second one, the client decides to search only in a specific leaf class. The discovery will be done in the registry peer of the leaf class and will not be propagated to other classes.

As for the publication phase, the discovery process involves interactions between the different peers, and thus between the different agents implemented in these peers. Figure 39 is a diagram that represents these interactions.

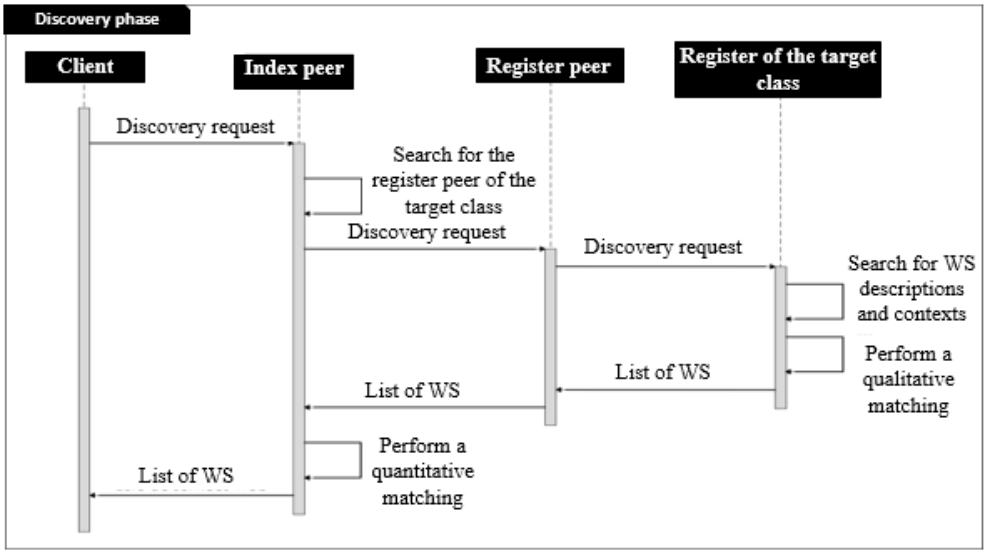

Figure 39 Service's discovery diagram.

The following scenario illustrate the different steps of the discovery process through the example of a tree structure that proposes classes and subclasses as shown in Figure 40.

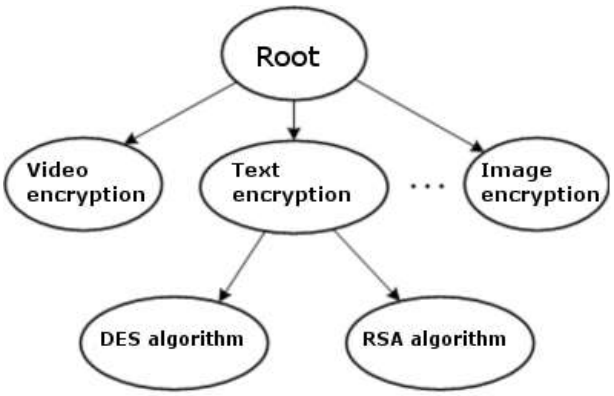

Figure 40 An example of a tree structure.

A client search for a Web service to encrypt his text. To do so, he decides to search in the "Text Encryption" class (without choosing any subclass of the *Text Encryption* class). The client may not be interested in the choice of the algorithm to encrypt his text, but may rather be interested in the QoS parameters to make his choice.

Step 1: The client connects with his account to search for services. The *Gateway Admin Agent* will create an instance of the *User Manager Agent* to verify the client's information, retrieve his context, create an instance of the *Client Discovery Agent*, and pass on the client's context.

Step 2: The client chooses to search for a service in the "Text encryption" class, indicating that it involves also searching in subclasses. Then, he initiates the discovery request by specifying his needs, as well as his context.

Step 3: The *Client Discovery Agent* receives the discovery request and looks in the routing table to find out the address of the Peer Index to reach the registry class of "Text Encryption". Once the address is retrieved, the CDA sends the query to the index peer.

Step 4: The *Admin Index Agent* receives the discovery request, sent to the Index Peer, creates an instance of the *Unifier Discovery Agent* and sends it the discovery request.

Step 5: The UDA analyzes the discovery request and checks its type (type1 or type2). In the considered scenario, the UDA finds that it is a request of type1, it then traverses the index tree to retrieve the registry address of the target class "Text encryption", as well as the addresses of the registries of its child classes (in this case "DES Algorithm" and "RSA Algorithm" classes). Once the addresses have been retrieved, the UDA sends them the discovery request.

Step 6: The *Registry Admin Agent* receives the discovery request, sent to the registry peer, and creates an instance of the *Discovery Agent* for each target registry, and passes them the discovery request.

Step 7: Each DA asks the UDDI directory to retrieve services that match the client's functional requirements.

Step 8: Each DA retrieves, from the Web services context database, the contexts of the services discovered in step 7.

Step 9: Once the contexts are recovered, the DA performs a first qualitative matching between the qualitative attributes of the client context and the qualitative attributes of services' contexts. This process will be detailed in the qualitative filtering presentation within the contextual discovery

section. Each registry peer will subsequently transmit to its parent peer index a clean list of services corresponding to the client's qualitative contexts, as well as their quantitative context. These contexts are used during the second filtering stage quantitative matching (contextual ranking).

Step 10: Once the UDA receives all the lists of Web services, sent by the different peer registries, the UDA performs the second quantitative matching between the quantitative attributes of the client context and the quantitative attributes of services contexts, in order to get the final list of Web services.

This list will contain the most appropriate services to the client context. The UDA will forward this list to the CDA in the gateway peer. This process will be detailed in next section.

Step 11: The CDA receives the final list and returns it to the client.

As mentioned previously, filtering is applied to return to the client a list of services relevant to his context. A first filtering, based on the qualitative attributes of the context (qualitative matching), is executed by the Discovery Agent, in the Peer Registry, in order to eliminate all services that do not meet the requirements of the user. The second filter is executed in the Super-Peer Index, in order to extract services that most closely match the client's requirements. In this step the services are selected using a fuzzy ranking approach based on a reference point. Figure 41 illustrates the different stages of the discovery process.

In order to effectively take advantage of the context and ensure that the client will obtain Web services adapted to his context, it is necessary to detect and model the contextual information. In the next section, we present the conceptual definition of the context, defining the attributes that constitute it, as well as the format used for its modeling.

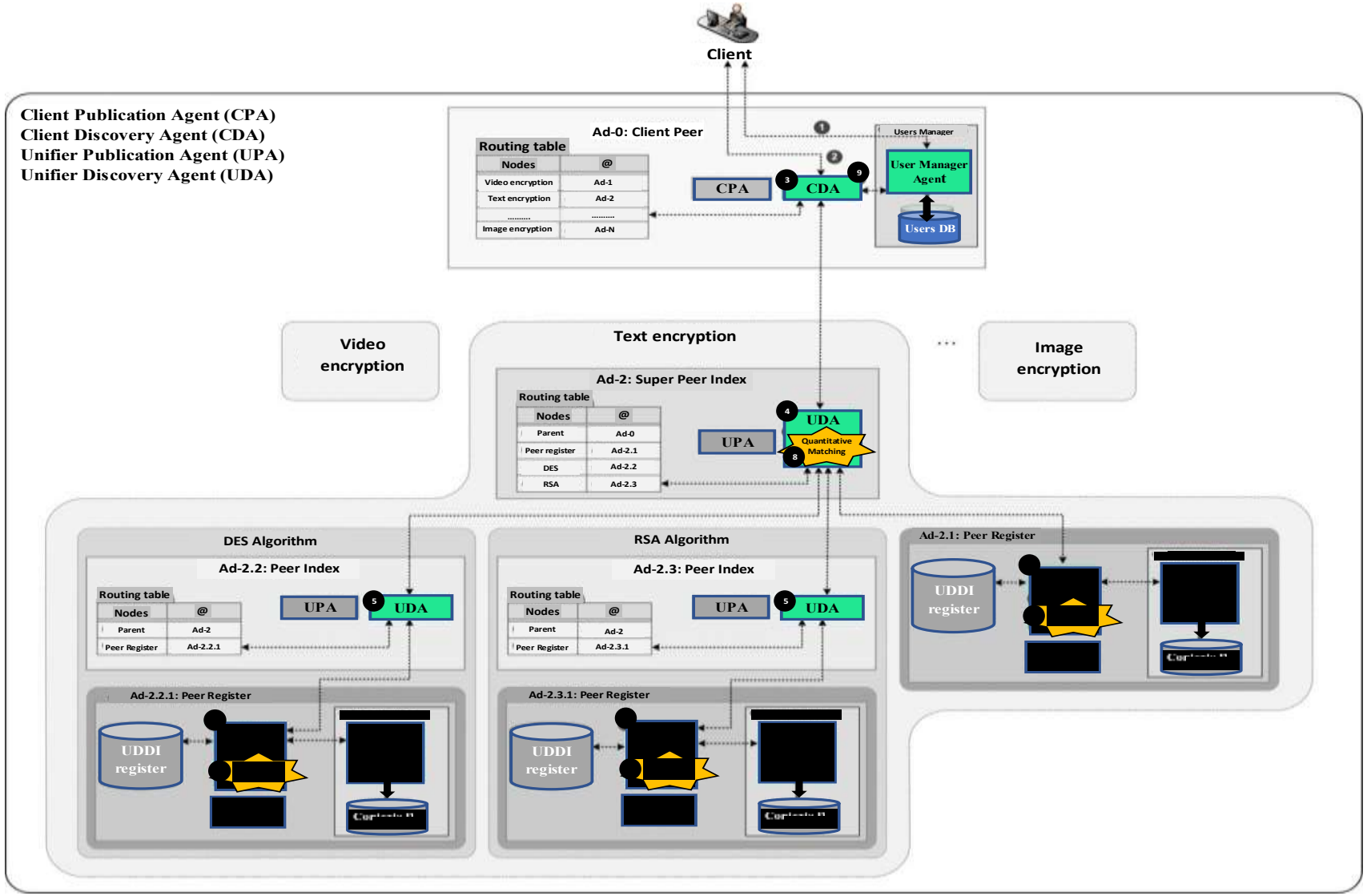

Figure 41 Discovery process.

6.5. Using Contexts in Services Discovery

In this section, we explain how the contextual discovery process goes throw the proposed architecture, showing first the model used to organize the context in general then the model of each of the client and the service separately. Finally, we present how the context is stored and the proposed approach used for discovering WS

6.5.1. Modeling the context structure

The general structure proposed to represent the context is presented in this section.

6.5.1.1. Generic context structure

The proposed structure is based on a generic model, while distinguishing between qualitative and quantitative attributes. Figure 42 is a class diagram presenting the meta-model of the proposed structure.

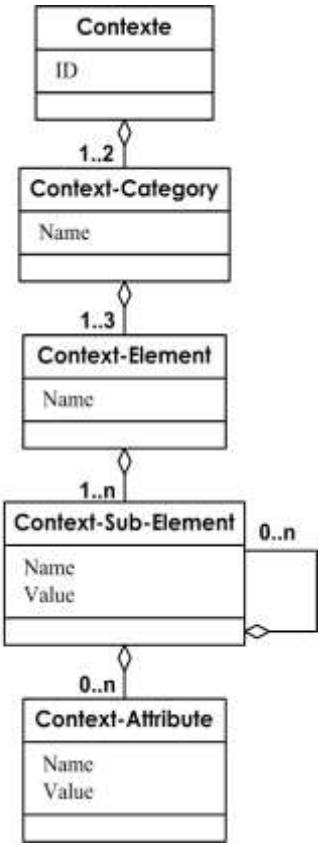

Figure 42 Meta-model of the proposed context structure.

Every user and service description are associated with a “Context”. The context is organized according to two main categories "Context-Category": *qualitative* or *quantitative*. Each category describes the elements of the context "Context-Element": the user profile or service profile, the terminal profile, the network profile.
The "Context-Element" can contain "Context-Sub-Elements" which presents the sub-elements of the context. For example, the user profile has the name, first name, etc., and the terminal profile has the software and hardware characteristics.

The «Context-Sub-Element" can contain other "Context-Sub-Elements" and a "Context-Attribute" element. For example, in the hardware sub-element, it has "Name" attribute and sub-elements like "device type", "screen size", and so on.
At a given moment, the contextual situation is defined by the values associated to "Context-Sub-Elements". For example, the "device type" is "mobile phone", "operating system" is "Android", "network type" is "Wi-Fi", and so on.
Based on the general structure of the context model proposed in Figure 42, the user and service contexts are presented with explaining the main difference between these two contexts.

6.5.1.2. The user context

The proposed user context is composed of three profiles: *user profile*, *terminal profile* and *network profile* (Figure 43). Each element is instantiated from the "Context-Element" of the generic model, presented in the previous section. Each of these elements describes a part of the user context via the "Context-Sub-Elements".

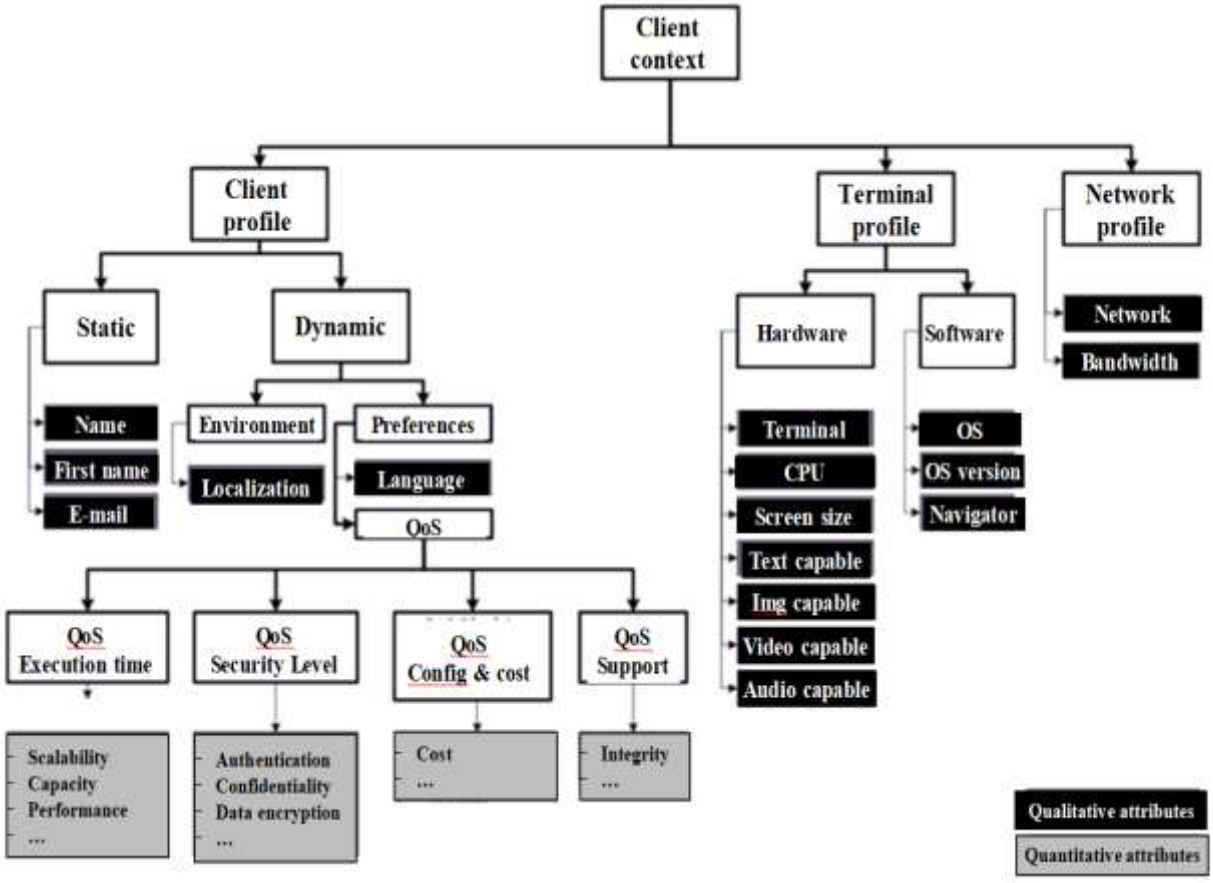

Figure 43 The user's context.

6.5.1.3. The service context

The proposed service context model is composed of three profiles: *service profile*, *device profile* and *network profile* (Figure 44).

The service provider and the client are not obliged to completely fill all the context attributes however, it can affect the results of the discovery process.

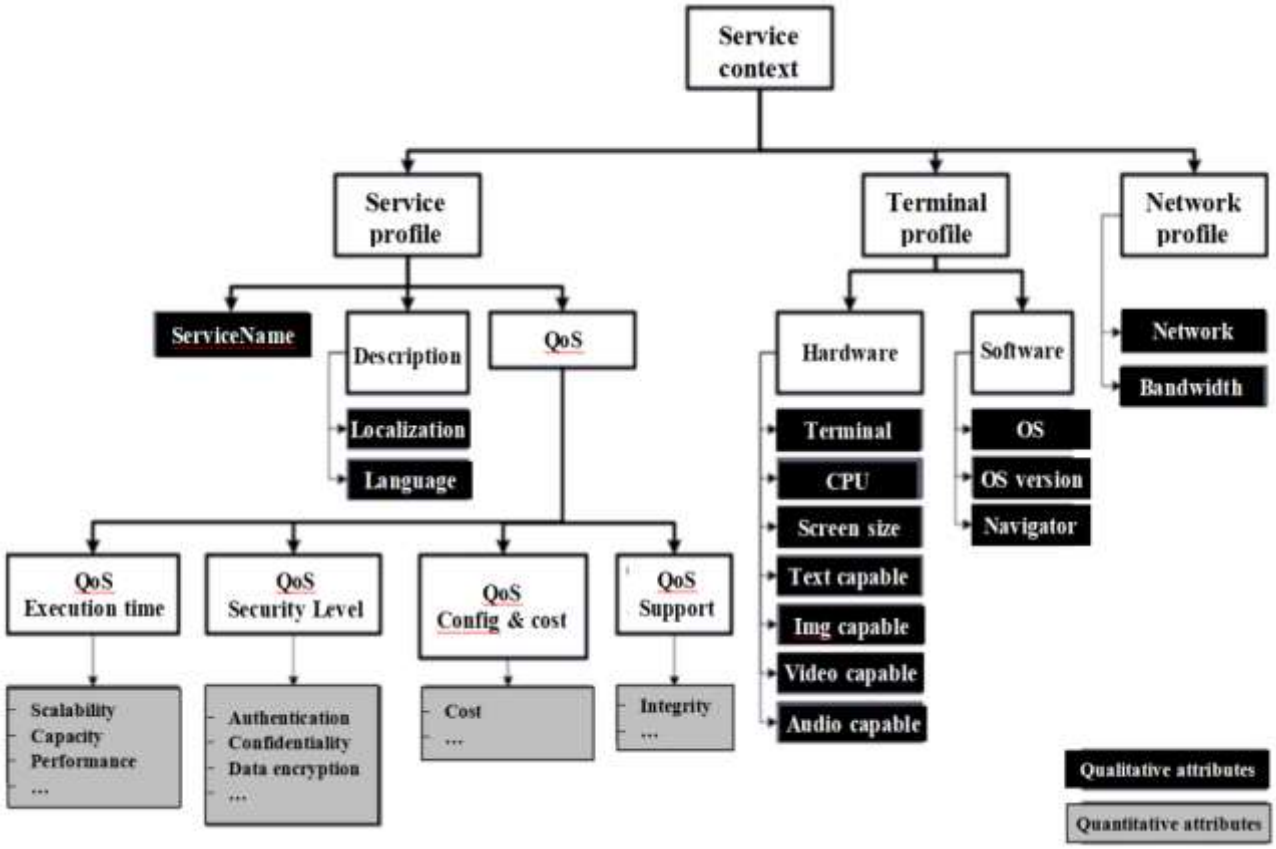

Figure 44 The Service's context

6.5.2. Context representation

The general structure of the context allows the characteristics of the user's environment (the device, the network, the location, etc.). Each characteristic can be integrated into the context as a "Context-Sub-Element", in order to ensure the generality of the proposed context model.
To represent and store contextual information, many researchers used the composite/preference profiles (CC/PP) format (Klyne, G. 2001). This format, based on XML, is considered as a standard for describing contexts. CC/PP describes the characteristics of a device and some preferences of the user. These profiles are structured in such a way to enable a client to describe his abilities using a predetermined vocabulary.
Although CC/PP is decomposable, uniform and extensible, it cannot be used to represent the proposed context because it lacks structuring capabilities. Indeed, its strict hierarchy in two levels is not appropriate for capturing complex profile structures.
Therefore, the CSCP (*Comprehensive Structured Context Profiles*) format is selected as a context presentation format, which overcomes CC/PP lacks, by providing a multi-level structure which allows representation of all types of contextual information. Each element of the CSCP document represents an element of the user's context. Furthermore, all services contexts are stored in CSCP documents.

6.5.2.1. Qualitative filtering

The objective of qualitative matching is to retain only those services with a qualitative context similar to the client's qualitative context. For example, if the client context specifies "Windows" for the "OS" attribute, then only services that have "Windows", in the "OS Type" attribute of their context, are retained. This process produces a short list of services, which will later move to the second stage of filtering. The qualitative filtering is ensured using the algorithm 1 presented below.

Algorithm 1 Web service first stage selection according to context qualitative attributes

```
Input:
   User.Att = {At1, At2,…,AtM}; // Client qualitative attributes
   WS = {WS1, WS2,…,WSN}; // set of Web Services

Output:
   WSL //list of preselected WS

Begin

 WSL = Ø;
 for i = 1 to N do
    j = 1;
    Exist = true;
   While (( j <= M) and (Exist == true)) do
         if (User.Attj ∉ WSi.AttListj)
           then Exist = false
        end;
   done
   if (Exist == true)
     then WSL = WSL ∪ {WSi}
   end;
 done
 return WSL
end
```

6.5.2.2. Quantitative filtering

In this stage, we take the short list obtained during the qualitative matching, in order to keep only services having a qualitative context close to that of the client. The quantitative context is considered as a combination of several numerical criteria that included the QoS attributes. The objective, is to choose services offering the best level of QoS among preselected services, using a MultiCriteria Optimization (MCO).

6.6. Improved Contextual Fuzzy Ranking Approach Based On A Reference Point

In a previous work, we proposed a fuzzy ranking approach for the contextual service discovery [92]. It consists on using a multi-criteria ranking approach instead of simple arithmetic approaches like Cosine and Jaccard [85].

To enhance the results of the previous approach, the fuzzy logic is used in the ranking process. The authors used a coalition function [90] to avoid the gradual transition between the context values.

Moreover, authors improve the ranking process by introducing some modification in the ranking algorithm and by proposing a dynamic method to calculate the coalition function parameters. Furthermore, for more accurate evaluation, a real Web services dataset is used to validate the approach.

Variables are set as follow:

- S: contains the values of the service contextual attributes, preselected for the ranking process;
- Rs: contains the values of the service contextual attributes, which is in the raking process.
- U: contains the values of the contextual attributes required by the user.
- Si: is the i^{th} attribute of a service from the preselected services list.
- Rsi : is the i^{th} attribute from the ranked service context.
- Ui : is the i^{th} attribute from the required context.
- Scorei : is the calculated score of the ranked service.

The following ranking rules were defined in [92]:

1- if ($Rs_i = Si$ and $Rs_i = U_i$) then no added value;

2- if ($Si < Rs_i$ and $Si < U_i$) then Score ++;

3- if ($Rs_i = U_i$ and $Si < Rs_i$) then Score ++;

4- if ($Rs_i = U_i$ and $Rs_i < Si$) then no added value to the Score;

5- if ($Rs_i > U_i$ and $Rs_i > Si$) then Score ++;

6- if ($Rs_i < Si <= U_i$) then Score --;

7- if ($Rs_i < U_i$ and $U_i < Si$) then Score --;

However, we noticed that some rules could be changed to reduce the number of rules as an improvement of the previous version.

The reduced and improved version of the rules are:

1- if (Rsi = Ui) then a value is added to the Score i;

2- if ((Si < Rsi) and (Rsi < Ui)) then a value is added to the Score i;

3- if (Rsi > Ui) then a value is added to the Score i;

4- if ((Rsi < Si) and (Si <= Ui)) then a value is subtracted from the Score i;

5- if ((Rsi <Ui) and (Ui < Si)) then a value is subtracted from the Score i.

The added or subtracted value from the score can be a constant (for example we can set value = 1) or considered as a variable. In this approach, the value will be defined by the weight of each attribute (variable) to promote the services that have the required attributes with higher weights.

These weights replace the importance factor used in [41], thus, this approach gives more accuracy because it considers all services attributes even when they are not important for the user while the proposed approach in [41] will consider only the important attributes.

Examples: In the next example, the previous rules are applied to calculate the score. First, using a constant value and then using a weight associated to each attribute.

Table 8 Example for weighted ranking.

Service/attributes	Att1	Att2	Att3	Att4	Score	Weighted Score
User context	4	4	3	5	-	-
Weights	1/5	1/5	1/5	2/5	-	-
Service 1	2	2	3	1	-6	-1,6
Service 2	3	4	4	2	4	0,8
Service 3	4	2	3	5	6	1,6

This example shows the difference in accuracy using weights in the proposed approach. Indeed, it is noticed that the weight of the fourth attribute influences the distance between the score of the second and the third services.

The following example presents another issue:

Table 9 Motivation example for fuzzy logic.

Service/att	Att1	Att2	Att3	Att4	Score	Weighted Score
User context	**4**	**3**	**4**	**5**	**-**	**-**
Weights	**1/5**	**1/5**	**1/5**	**1/5**	**-**	**-**
Service 1	**4**	**4**	**4**	**6**	**2**	**2/5**
Service 2	**4**	**4**	**3.9**	**4.8**	**-3**	**-1/5**

As a remark, the Service 1 and Service 2 are equal in Att1 and Att2 and satisfy user context. Also note that Att3 and Att4 in Service 1 satisfy the user context relatively to those of service 2 (according to the rule 5), although the values of these attributes are close to the values requested by the user. In this case, the algorithm penalizes this service despite the values are very close to the required attributes values.

From human perspective some differences in context values are not necessarily significant to favor one service over another service on a criterion i. So, considering the difference between the compared values in the ranking process is very promising.

Therefore, we propose to use a fuzzy logic method to consider the difference between the reference point and the candidate service to avoid the problem mentioned previously.

A non-gradual preference model was proposed in [90] and presented first time in [93]. This approach uses two thresholds: the indifference threshold qj and the discrimination threshold pj (pseudo criterion), where:

- The indifference threshold qj: corresponds to the maximum performance difference;

- The discrimination threshold pj: corresponds to the minimum performance difference.

So, we note as follows:

- g(X) is the function that returns the value of the attribute X;
- X_i is the i^{th} attribute of the context profile X;
- Rs is indifferent from S on attribute i if: ($|g(Rs_i) - g(S_i)| <= qj$) ;
- Rs is different from S on attribute i if: ($g(Rs_i) >= g(S_i) + pj$) ;
- Rs is weakly indifferent from S on attribute i if: ($qj < |g(Rs_i) - g(S_i)|$) and ($|g(Rs_i) - g(S_i)| < pj$))

To avoid the strict graduation problem, the concept of indifference is used which proposed by [93]. The goal is to distinguish, in the set of criteria, two fuzzy sub-sets separating the concordant coalition and the discordant coalition.

In this approach, we use the concordant function [94], defined as follow:

$$C_j^{\sim}(x,y) = \begin{cases} 0 & \text{if } x_j - y_j \leq -p_j \\ \frac{1}{2} + \frac{1}{2}\sin\frac{\pi}{p_j - q_j}\left(x_j - y_j + \frac{p_j + q_j}{2}\right) & \text{if } -p_j \leq x_j - y_j \leq -q_j \\ 1 & \text{if } |x_j - y_j| \leq q_j \\ \frac{1}{2} + \frac{1}{2}\sin\frac{\pi}{p_j - q_j}\left(x_j - y_j - \frac{p_j + q_j}{2}\right) & \text{else } q_j \leq x_j - y_j \leq p_j \\ 0 & \end{cases}$$

Formula/Eqt (16)

Where: x_i and y_j are the contexts' attributes that intend to compare using the thresholds qj and pj.

In our previous work [92], we set qj and pj as constant values. However, in this approach we calculate the two thresholds qj and pj dynamically thanks to the use of a services dataset [95] that contain real context values collected from the Web.

To achieve this, considering the list of the resulted services during the functional discovery. For each preselected service, the average of the Euclidian Distance (AED) is calculated with the required profile this function is presented in Algorithm 2.

In order to define the indifference threshold qj, the average is calculated for all Euclidian distances between the required profile and each preselected service that are smaller than AED.

The same process is applied to define the threshold pj but with the bigger distances.

$$\boldsymbol{d(x,y) = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2}}$$

Formula/Eqt (17)

Algorithm 2

{int Service_number = 0;

for (each $service_i$ preselected in the syntactic search) do
{
for (each $attribute_k$ of $service_i$) do
{
Euclidian $distances_k$ += (User_$profil_k$ - $attribute_k$)² ;
Service_number += 1;
}
}
for (k = 1 to n) do
{
Average Euclidian $distances_k$ = $\sqrt{\text{Euclidian distances}_k}$ /*Service_number*;
}
}

As an example, in table 10, the Average Euclidean Distances for these services.

Table 10 Example of calculating Euclidean Distance Average.

Service/att	Att1	Att2	Att3	Att4	Score	Weighted Score
User profile	4	4	3	5	-	-
Weights	1/5	1/5	1/5	2/5	-	-
Service 1	2	2	3	1	-6	-1,6
Service 2	3	4	4	2	4	0,8
Service 3	4	2	3	5	6	1,6
AED	2,23	2,82	1	5	-	-

The table 11 contains for each attribute the distance between the required profile (user profile) and the list of preselected services.

Table 11 Example of calculating AVG of bigger and smaller distances to calculate the qj and the pj.

Service/att	Att1	Att2	Att3	Att4	Score	Weighted Score
User profile	4	4	3	5	-	-
Service 1 distance	|2-4|=2	|2-4|=2	|3-3|=0	|1-5|=4	-6	-1,6
Service 2 distance	|3-4|=1	|4-4|=2	|4-3|=1	|2-5|=3	4	0,8
Service 3 distance	|4-4|=0	|2-4|=2	|3-3|=0	|5-5|=0	6	1,6
AVG Euclidian distance	2,23	2,82	1	5	-	-
AVG Bigger distance	0	0	1	0	-	-
AVG Smaller distance	1	2	0	2,33	-	-

The average of the bigger distances is taken as the pj threshold and the average of the smaller distances as the qj threshold if the bigger AVG value is bigger than the other average, else the threshold values are inversed.

The result of this process will give us more indication about the attributes values distribution and how to consider the differences between them.

The obtained thresholds qj and pj in the indifference function Cj (x,y) are used in the rules 2, 3 and 5 to resolve the strict graduation problem.

The indifference function Cj (x,y) returns a value between 0 and 1 which can be converted to a percentage indicating the indifference between two profiles X and Y.

In this case, instead of adding/subtracting the weight to the service score, a fraction of the weight is added using the indifference function results.

6.7. Evaluation of the ranking approach

For the evaluation of the proposed approach, we used the QWS dataset [95], a public database containing 2507 services. The dataset presents a set of 9 QoS attributes to describe each Web service. Table 6.5 describes the attributes used in the dataset.

Table 12 Description of the dataset attributes.

	Attribute	Description	Unit	The optimum
1	Response Time	Time to send a request and receive a response	ms	To Minimize
2	Availability	Successful invocations/total invocations	%	To Maximize
3	Throughput	Total Number of invocations/period of time	Invoc/s	To Maximize
4	Successability	Ratio: # response messages / # request messages	%	To Maximize
5	Reliability	Ratio: number of error messages/total messages	%	To Maximize
6	Compliance	Extent a WSDL follows a specification	%	To Maximize
7	Best Practices	Extent of following WS-I Basic Profile	%	To Maximize
8	Latency	Time to process a given request	ms	To Minimize
9	Documentation	Measure of documentation in WSDL	%	To Maximize

The authors performed a test on four lists of preselected services (resulted from the syntactic search of four different requests). For each preselected service, we apply the ranking approach and the QSim measure [41] on its context to get the similarity results.

The user profile is considered as the reference point in the ranking process, each service in the list will be compared with all other services using the reference point.

The algorithm is programmed with JAVA programming language in Netbeans IDE 8.1 and tested using personal laptop with Intel I5 vPro processor and 8 GB RAM.

Table 13 and 14 present the results obtained for a request for a mail services and data services.

Table 15 and 16 present the results obtained for a request for a geolocation services and SMS services.

Table 13 Example of 15 services resulted from the ranking of the data services.

	Att 1	Att 2	Att 3	Att 4	Att 5	Att 6	Att 7	Att 8	Att 9	Qsim	Ranking approach (score)
TNM_Elevation_Service	174,5	99	2,5	100	80	78	87	1	96	74,57	104,64
FullerData_x0020_High	115,4	96	18,7	99	80	78	87	1,4	93	70,77	101,30
DataEnhancement	63,8	99	18,1	100	73	78	84	1,8	95	73,47	97,73
USDAData	3321,4	89	1,4	96	73	78	80	2,6	96	100	90,72
logon	308	93	12,9	98	78	78	89	1	11	60,30	89,23
TNM_Gazetteer_Service	83,33	98	2,1	100	80	78	87	1,33	66	73,18	87,51
DataEnhancement	158,8	98	20,1	100	67	100	82	1,2	95	68,20	86,02
TNM_Gazetteer_Service	69,33	91	1,7	97	80	78	87	2	66	78,79	84,28
USNG	117,5	89	3,2	96	73	78	84	0,5	89	72,91	82,03
DataTypes	150,5	92	16,2	97	73	78	80	1	12	61,70	81,01
Service1	80	99	36,3	100	80	100	87	13	3	52,44	79,51
MedicareData	51,5	100	1,8	100	73	78	80	2	32	74,53	79,15
EquipmentData	124,33	96	7,7	99	80	89	65	43	97	64,43	76,42
IClientDataservice	315,85	90	9,9	97	72	89	86	1,65	4	62,56	71,79
redataService	383,2	98	2,1	100	73	100	84	6,2	39	68,69	71,38

Table 14 Example of 15 services resulted from the ranking of the mail services.

	Att 1	Att 2	Att 3	Att 4	Att 5	Att 6	Att 7	Att 8	Att 9	Qsim	Ranking approach (score)
SendEmailService	198	96	22,4	99	73	100	84	2	94	72,99	38,00
DOTSEmailValidate	105,6	100	19,2	100	73	78	84	0,6	95	69,22	36,03
EmailVerification	152	96	4,9	99	60	100	79	51	73	100	36,01
XWebEmailValidation	130	89	24,5	96	67	100	82	1	95	73,36	30,95
EmailVerify	126	99	2,4	100	73	78	84	2	94	73,78	28,10
MailingList	338,2	99	5,6	100	73	100	80	46	4	77,61	24,05
PegsTourService	212,5	100	13	100	78	78	91	6	6	62,34	23,14
ValidateEmail	149	94	16,7	98	73	78	84	36	42	78,31	22,79
EmailVerify	575,5	86	6,4	95	67	100	82	1	88	73,36	21,22
EmailVerification	539	95	3,6	98	60	100	79	42	85	85,32	20,67
MacamMailer	620,5	96	8,7	99	73	100	80	398	10	64,85	18,20
SendWM2Mail	227	88	21,2	96	73	100	84	23	34	71,59	18,06
EmailValidation	136	83	21,4	84	83	78	91	6	11	60,39	17,14
GD_VSDB_WMService	111	72	15	72	83	89	91	6,5	8	57,89	16,56
EmailVerNoTestEmail	95,38	62	13,7	62	73	78	80	55,88	88	73,31	13,45

Table 15 Example of 15 services resulted from the ranking of the SMS services.

	Att 1	Att 2	Att 3	Att 4	Att 5	Att 6	Att 7	Att 8	Att 9	Qsim	Ranking approach (score)
SmsApiService	158	100	23,3	100	80	89	87	1	10	61,29	57,28
SmsGatewayService	724,82	99	4,7	100	60	89	69	13,53	5	100	47,93
TextAnywhere_SMS	138,25	98	14,5	100	73	100	84	19,92	39	64,43	45,08
SendSmsService	206	98	2,1	100	67	89	82	10,25	59	69,39	41,10
SMSMessagingService	64,4	95	4,1	99	73	100	80	9	39	69,19	39,35
ATTSMS	138	85	33	95	73	100	84	13	12	66,99	37,43
sms	361,33	63	7,6	63	83	89	91	11,66	39	64,53	37,19
SOAPSend	152	96	24,2	99	73	78	80	12	5	75,93	35,24
SendSMSWorld	252	92	13,1	97	80	78	87	1	39	57,63	34,54
RandomBushismService	260	83	15,5	84	73	89	84	10	11	68,41	33,76
EnvoiSMS	164,33	95	15,2	98	73	78	84	13,33	9	72,45	32,89
SMSMessagingService	102,2	90	3,3	97	73	100	80	2,2	41	61,52	32,53
SOAPSend	154	90	23,9	97	73	78	80	9	9	67,32	29,64
SMS	106	95	5,5	99	73	100	84	28	1	68,04	29,40

Table 16 Example of 15 services resulted from the ranking of the geo-localization services.

	Att 1	Att 2	Att 3	Att 4	Att 5	Att 6	Att 7	Att 8	Att 9	Qsim	Ranking approach (score)
DOTSGeoCoder	107,45	97	13,3	99	73	78	80	1,36	94	78,08	71,18
DOTSGeoPhone	102	90	18,6	97	73	78	80	1	92	76,60	70,55
GeoAccess	191,11	87	9,6	95	67	78	77	30,85	89	100	65,65
GeoAddress	306,2	95	9,8	99	73	78	80	21,2	90	89,38	61,78
DOTSGeoPhone	107,4	85	19	95	73	78	80	0,8	92	77,06	60,35
DOTSGeoPinPoint	138,5	100	2	100	73	78	84	2	95	73,42	59,29
geoservices	65,07	88	16,7	96	67	100	77	1,28	3	63,27	57,10
DOTSGeoCensus	98	94	0,7	98	73	78	80	3	96	71,17	56,47
Geolizer	214	100	26,7	100	73	78	88	2	1	65,37	48,35
GeoIPService	172	98	6,1	99	73	78	84	1	37	73,74	46,97
GWSRequest	153,44	87	1,7	96	73	100	80	0,77	41	67,73	46,26
GFCore	148,5	98	11,4	99	73	89	62	1	5	67,76	45,65
IOremGeocoderservice	117	83	23,3	84	67	89	82	4	5	64,56	35,36
QueryWebserviceService	144	83	16,9	84	67	89	82	10	4	69,83	34,13
geographyRequests	125	86	9,4	86	73	78	84	3	7	71,99	33,42

To evaluate the performance of the proposed approach with QSim, we consider the following parameters:

– Number of times we get better results with the ranking approach;
– Number of times we get better results with the QSim approach;
– Number of proposed services when ranking approach gives better results;
– Response time (execution time) of the two approaches.

In some cases, the proposed approach gives better results, but some services have fewer attributes than required (in most cases, no more than two attributes and with relatively close values) that is why the fuzzy logic was included.

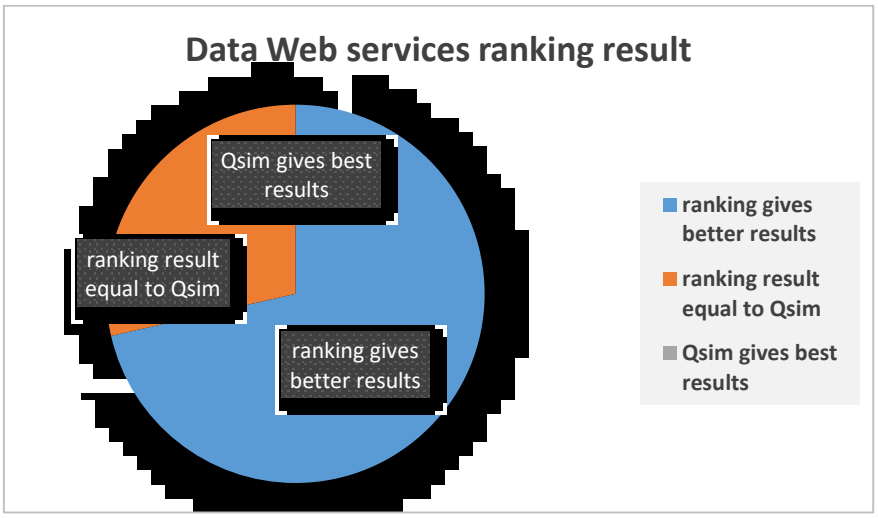

Figure 45 Comparison of QSim and ranking results for DATA request.

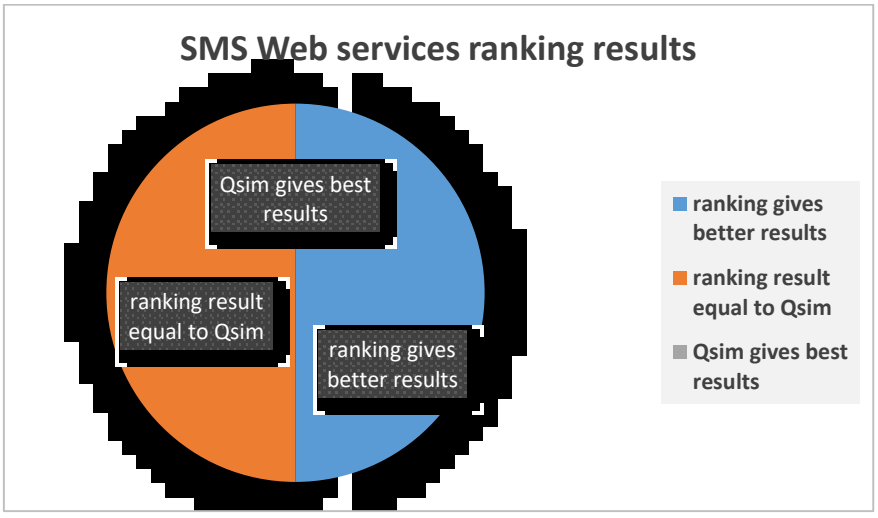

Figure 46 Comparison of QSim and ranking results for SMS request.

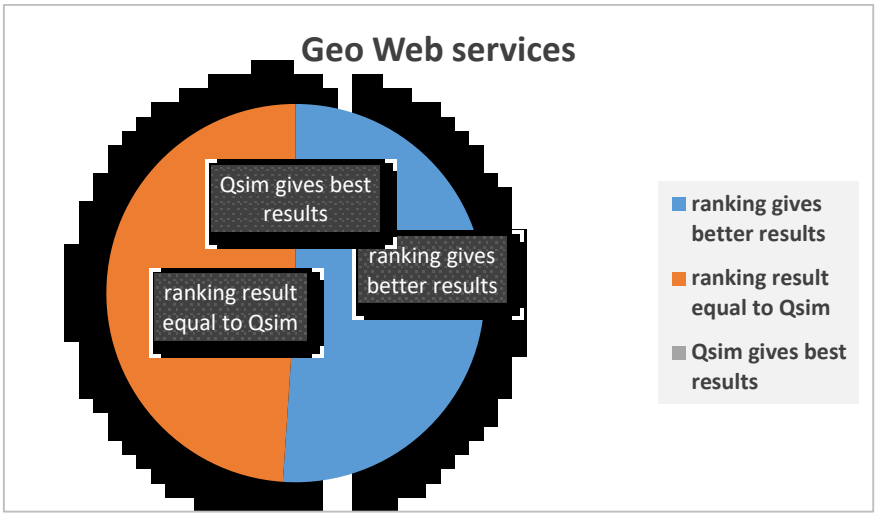

Figure 47 Comparison of QSim and ranking results for GEO request.

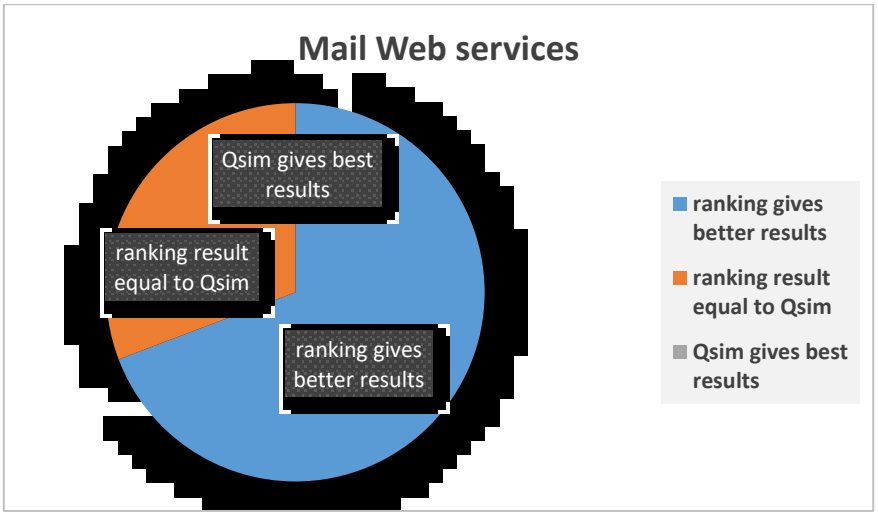

Figure 48 Comparison of QSim and ranking results for Mail request.

Figure 49 and Figure 50 shows the number of services selected with the ranking approach when it gives better results.

The result changes depending on the reference point and the qi and pj parameters, extracted from the list of services preselected during the functional discovery step. The fuzzy ranking approach detects profiles that are similar or nearly similar to the required profile.

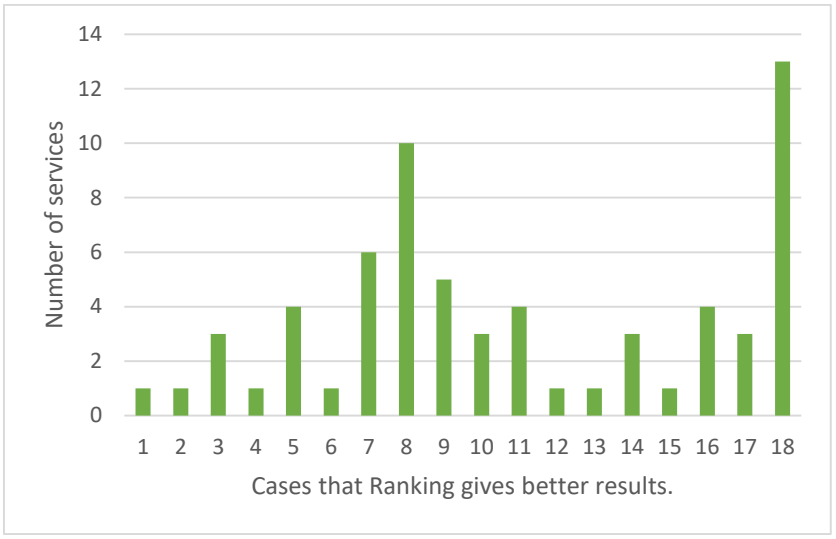

Figure 49 Number of services for each case that ranking approach gives better results for the SMS request.

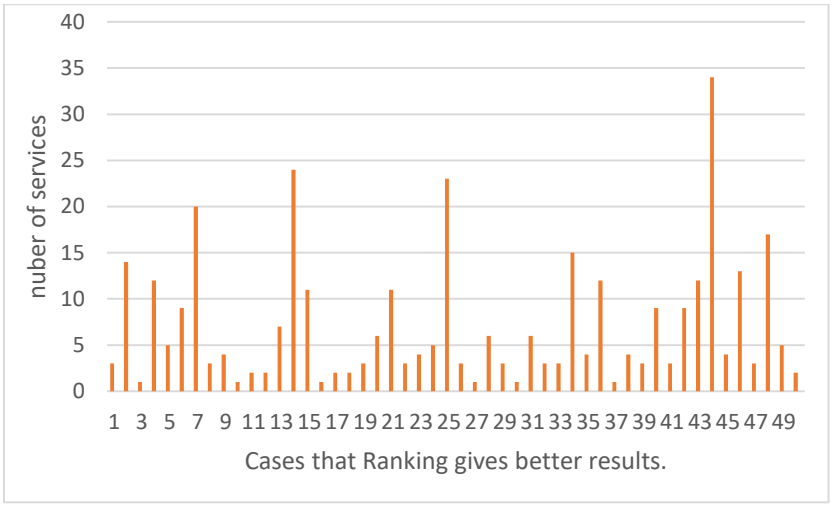

Figure 50 Number of services for each case that ranking approach gives better results for the DATA request.

Table 17 Example shows the efficiency of the fuzzy logic in the ranking

	Att1	Att2	Att3	Att4	Att5	Att6	Att7	Att8	Att9	Qsim	Time ex(µ)	Ranking approach (score)	Time ex(µ)
FullerData_	115,4	96	18,7	99	80	78	87	1,4	93	70,04	16	99,35	671
DataEnhanc-	158,8	98	20,1	100	67	100	82	1,2	95	66,73	21	98,90	636
TNM_Elevation	174,5	99	2,5	100	80	78	87	1	96	71,83	18	96,10	674
TNM_Gazetteer	83,33	98	2,1	100	80	78	87	1,33	66	81,20	50	92,68	842
DataEnhanc	63,8	99	18,1	100	73	78	84	1,8	95	73,50	15	90,44	732
Data-Generat	49,5	85	1,8	95	67	78	72	1,5	34	100	168	89,00	793

The example in table 17 shows the case when the service profile is almost similar to the required one (reference point), but it may be penalized by one or more attributes having a close value to the required profile. In fact, the way that we calculate the indifference coefficient qj and pj is very efficient using the list of pre-selected services from the functional discovery that gives the coefficients a dynamicity in order to adapt whenever a service is requested. However, the previous approach does not consider the case when the service has an attribute value bigger than the required profile (line one in the table rate with QSim is 70.04% when it is the best choice regarding the value of the first attribute which are relatively close).

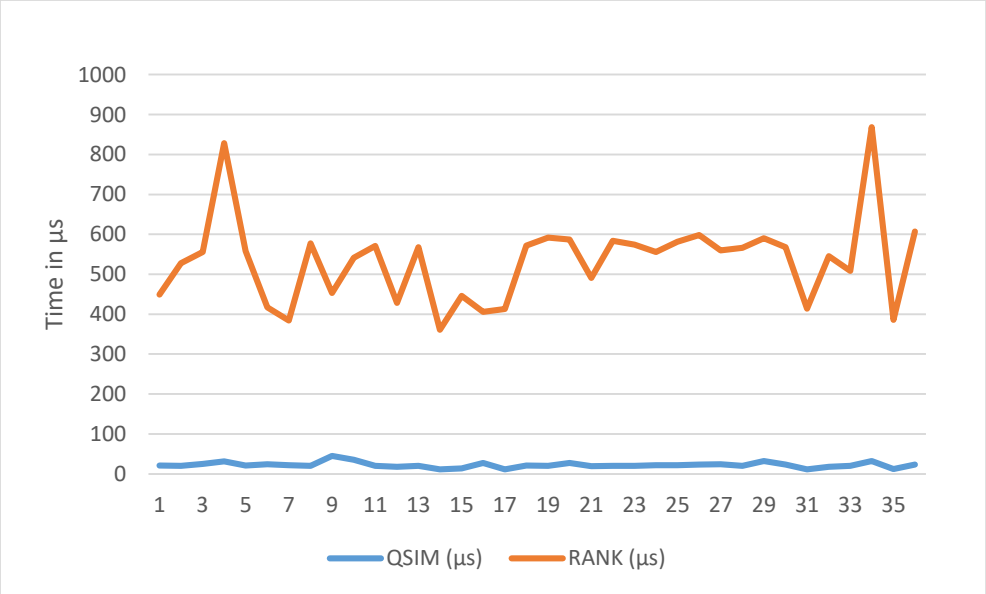

Figure 51 Response time for QSim and the ranking approaches.

For the execution time (see Figure 50), the ranking approach needs more processing, hence the response time of the ranking approach is more important. However, it still under 1 ms in worst case, which is acceptable if considering the accuracy and precision of the proposed approach.

Although the proposed approach has a $O(e^n)$ complexity, the advantage is that it addresses a limited list of services resulting from the functional discovery step, so the execution time will not necessarily be significant.

6.8. Conclusion

Web services technology is a very promising concept regarding its potential, including loosely coupled programing, interoperability and reusability. The discovery process must be able to increase the relevance of the results in order to be as close as possible to the needs of the user. Thus, to improve the discovery process, we propose a hybrid architecture with contextual fuzzy ranking. We aim to solve some of weaknesses in the discovery process such as latency, overload failure and network congestion by designing an architecture that avoids those problems.
The main goal of this work is grouping services and distributing them in hierarchically organized registries based on domain and subdomain specialty, for locating the right services easily. On the other hand, from the technical perspective, this work provides a unique user interface for the infrastructure that enable accessing multiple registries domain with a multi-agent platform in order to enhance service discovery mechanism.

The proposed architecture is based on a distributed network using a single access point (Gateway). The registry architecture is a super-peer architecture. It consists of three types of peers, namely gateway peer, index peer and registry peer. This solution able to manage the increasing number of services using multi-level domain service distribution and multi-agent platform that manages the different tasks from the publishing, the discovery and the context and QoS filtering.

As a second contribution, we improved the contextual fuzzy ranking approach. This approach allows choosing the service offering the best level of quality parameters between services with the same functionalities. We use it for contextual discovery, which shows a very promising result.

Chapter 7: Conclusion and Perspectives

Web services are now one of the most adopted technologies for the interoperability of information systems. They are characterized by their independence from platforms and operating systems. Over the years, Web services have become the most adopted technologies by organizations and enterprises to provide their service through the Web. Thus, the number of these services has become very large, and the task of discovering them has become a task that requires powerful discovery systems.

The objective of this thesis is to propose solutions for the service discovery challenges in order to enhance the service based technology users' experience.

After we had proposed state-of-the-art that discusses Web service major concepts. We study in detail most of the existing repositories solutions in several types of architecture and witch treat user request in deferent way as syntactic and semantic discovery as well as context-aware discovery.

We have presented a discovery approach in a hybrid architecture to encompass the advantages of both centralized and distributed architectures. Supported by a multi-agent system where tasks will be distributed over several agents instead of a single program, which will increase performance.

We have taken into consideration the context of the services and the user profiles in this approach to get better results. We have implemented a multi-criteria similarity measure in the discovery system, and we have compared the results obtained with other similarity measures such as QSIM.

In the future, we intend to improve our system by adding and integrating other approaches such as the semantic approach. The semantics can be used to classify automatically services during the publication step also in the discovery process to enhance the search results. The context ranking works only on quantitative attributes rather than qualitative, the semantics are very useful to resolve this limitation.

- Adding a QoS updater agent into the context discovery layer to have a consistent service selection based on QoS parameters.
- Studying the performance and stability of the platform and implementing security measures.
- The composition of services represents, in addition, a very important perspective in the case where the discovered services do not satisfy the user. The construction of new Web services from two or more Web services already present and published on the Web represent a solution to such problems.

Bibliography

[1] T. Pilioura and A. Tsalgatidou, “E-services: Current technology and open issues,” in International Workshop on Technologies for E-Services, 2001, pp. 1–15.

[2] G. Wiederhold, “Mediators in the Architecture of Future Information Systems Gio Wiederhold,” no. September 1991, 1992.

[3] Web services architecture, https://www.w3.org/TR/ws-arch/ (accessed Feb. 15, 2016).

[4] Web services architecture, https://www.w3.org/TR/ws-arch/ (accessed Feb. 01, 2016).

[5] Web services architecture, https://www.w3.org/TR/ws-arch/ (accessed Jul. 16, 2023).

[6] E. Newcomer, Understanding Web Services-XML, WSDL, SOAP and UDDI. .

[7] Simple Object Access Protocol (SOAP) 1.1, https://www.w3.org/TR/2000/NOTE-SOAP-20000508/. (Accessed: 06-Jan-2016).

[8] K. Decker, K. Sycara, and M. Williamson, “Middle-Agents for the Internet.” Proc. 15th., Nagoya, Japan., 1997, pp. 578–583.

[9] D. Booth et al., “Web Services Architecture. W3C WG Note.” [Online]. Available: http://www.w3.org/TR/ws-arch/. [Accessed: 03-Feb-2016].

[10] M. Malaimalavathani and R. Gowri, “A Survey on Semantic Web Service Discovery.”

[11] M. Paolucci, T. Kawamura, T. R. Payne, and K. Sycara, “semantic matching of web services capabilities.” in the first International Semantic Web Conference on The Semantic Web, 2002.

[12] D. Mukhopadhyay and A. Chougule, “A Survey on Web Service Discovery Approaches,” pp. 1001–1012, 2012.

[13] P. Palathingal and S. Chandra, “Agent Approach for Service Discovery and Utilization,” vol. 00, no. C, pp. 1–9, 2004.

[14] T. A. Letsche and M. W. Berry, “Large-scale information retrieval with latent semantic indexing,” Inf. Sci. (Ny)., vol. 100, no. 1–4, pp. 105–137, Aug. 1997.

[15] Klein M. and K.-R. B., “Coupled Signature and Specification Matching for Automatic Service Binding.” in the European Conference on Web Services (ECOWS 2004), 2004, pp. 183–197.

[16] K. U. and K.-R. B.:, “Semantic Service Discovery with DIANE Service Descriptions.,” in In WIIATW ’07: Proc. of the 2007 IEEE/WIC/International Conferences on Web Intelligence and Intelligent Agent Technology workshops, 2007, pp. 152–156.

[17] U. Küster, B. König-Ries, M. Klein, and M. Stern., “DIANE - A Matchmaking-Centered Framework for Automated Service Discovery, Composition, Binding and Invocation.” in 16th International World Wide Web Conference (WWW2007).

[18] I. El Bitar, F. Belouadha, and O. Roudies, “Semantic Web Service Discovery Approaches : Overview And Limitations,” vol. 5, no. 4, pp. 21–36, 2014.

[19] T. B. Lee, J. Hendler, O. Lassila, and others, “The semantic web,” Sci. Am., vol. 284, no. 5, pp. 34–43, 2001.

[20] E. Miller, “An introduction to the resource description framework.,” Am. Soc. Inf. Sci. Technol., vol. 25, no. 1, pp. 15-19., 1998.

[21] Owl-S 1.1 release, http://www.daml.org/services/owl-s/1.1/. (accessed Aug. 1, 2016).

[22] M. C. Jaeger, G. Rojec-Goldmann, C. Liebetruth, G. Mühl, and K. Geihs, “Ranked matching for service descriptions using OWL-S,” Inform. aktuell, vol. 2, pp. 91–102, 2005.

[23] L.-H. Vu, M. Hauswirth, and K. Aberer, “Towards P2P-based semantic Web service discovery with QoS support,” Bus. Process Manag. Work, no. 507483, pp. 18–31, 2006.

[24] C. Doulkeridis, N. Loutas, and M. Vazirgiannis, “A system architecture for context-aware service discovery,” Electron. Notes Theor. Comput. Sci., vol. 146, no. 1 SPEC. ISS, pp. 101–116, 2006.

[25] K. Arabshian and H. Schulzrinne, “Distributed Context-aware Agent Architecture for Global Service Discovery.”

[26] A. K. Dey, “Understanding and Using Context,” pp. 4–7, 2001.

[27] A. G. de Prado and G. Ortiz, “Context-aware services: A survey on current proposals,” Serv. Comput., pp. 104–109, 2011.

[28] S. Araban and L. Sterling, “Measuring quality of service for contract aware web-services,” in First Australian Workshop on Engineering Service-Oriented Systems, 2004, pp. 54–56.

[29] H.-L. Truong and S. Dustdar, “A survey on context-aware web service systems,” Int. J. Web Inf. Syst., vol. 5, no. 1, pp. 5–31, 2009.

[30] R. Reichle et al., “A comprehensive context modeling framework for pervasive computing systems,” Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics), vol. 5053 LNCS, pp. 281–295, 2008.

[31] T. Strang and C. Linnhoff-popien, “A Context Modeling Survey.”

[32] A. Ferscha, M. Hechinger, M. Riener, A. Schmitzberger, H. Franz, and M. d. Rocha, “Context-Aware Profiles.,” in International Conference on Autonomic and Autonomous Systems (ICAS 2006), 2006.

[33] C. Dorn and S. Dustdar, “Sharing hierarchical context for mobile web services,” no. December 2006, pp. 85–111, 2007.

[34] A. Held, S. Buchholz, and A. Schill, “Modeling of context information for pervasive computing applications,” Procceding World …, pp. 1–6, 2002.

[35] K. Henricksen, J. Indulska, and A. Rakotonirainy, “Modeling context information in pervasive computing systems,” in 1st International Conference on Pervasive Computing (Pervasive), 2002.

[36] K. Henricksen and J. Indulska, “A Software Engineering Framework for Context-Aware Pervasive Computing.,” in Second IEEE International Conference on Per- vasive Computing and Communications, 2004, pp. 77–86.

[37] T. A. Halpin, “Information Modeling and Relational Databases: From Conceptual Analysis to Logical Design,” in Morgan Kaufman, 2001.

[38] H. Chen, T. Finin, and A. Joshi, “An ontology for context-aware pervasive computing environments.” Knowl. Eng. Rev., vol. 18(3), pp. 197–207, 2003.

[39] B. Soukkarieh and F. Sèdes, “Dynamic services adaptation to the user’s context,” Proc. 2009 4th Int. Conf. Internet Web Appl. Serv. ICIW 2009, no. iii, pp. 223–228, 2009.

[40] S. K. Mostéfaoui, A. Tafat-Bouzid, B. Hirsbrunner, and others, “Using context information for service discovery and composition,” IiWAS, no. March 2003.

[41] F. M Bouyakoub, A. Belkhir, and M. A. Mellal, “A multi-agent system for Web Services Discovery in a UDDI cloud,” International Journal of Web Engineering and Technology, vol. 9, no. 3, p. 203, 2014. doi:10.1504/ijwet.2014.065866

[42] A. Mousa and J. Bentahar, “An Efficient QoS-aware Web Services Selection using Social Spider Algorithm,” Procedia - Procedia Comput. Sci., vol. 94, no. MobiSPC, pp. 176–182, 2016.

[43] S. Youcef, “Méthodes et outils d’évaluation de performances des services web,” http://www.theses.fr, Jan. 2009.

[44] S. Y. Chou, Y. H. Chang, and C. Y. Shen, “A fuzzy simple additive weighting system under group decision-making for facility location selection with objective/subjective attributes,” Eur. J. Oper. Res., vol. 189, no. 1, pp. 132–145, Aug. 2008.

[45] “QoS-based Selection of Services: The Implementation of a Genetic Algorithm | VDE Conference Publication | IEEE Xplore.” [Online]. Available: https://ieeexplore.ieee.org/abstract/document/5755506. (Accessed: 16-Jul-2023).

[46] E.Motta, J.Domingue, L.Cabral, and M.Gaspari, “A framework and infrastructure for semantic web services.” in International Semantic Web Conference, 2003.

[47] B. Soukkarieh, “On Using Generic Profiles and Views for Dynamic Web Services Adaptation,” vol. 2, pp. 18–23, 2013.

[48] W. Hoschek, “The web service discovery architecture,” ACM/IEEE 2002 Conf. Supercomput., vol. 00, no. SC, pp. 38–38, 2002.

[49] N. Griffiths and K.-M. (Kuo-M. Chao, Agent-based service-oriented computing. Springer, 2010.

[50] B. Sapkota, D. Roman, S. R. Kruk, and D. Fensel, “Distributed Web service discovery architecture,” Proc. Adv. Int. Conf. Telecommun. Int. Conf. Internet Web Appl. Serv. AICT/ICIW’06, vol. 2006, no. Fp6 004617, p. 136, 2006.

[51] I. Filali, F. Bongiovanni, F. Huet, and F. Baude, “A survey of structured p2p systems for rdf data storage and retrieval,” … Knowledge-Centered Syst. …, pp. 20–55, 2011.

[52] J. Zhou and Z. Shi, “Unstructured P2P-enabled service discovery in the cloud environment,” IFIP Adv. Inf. Commun. Technol., vol. 340 AICT, pp. 173–182, 2010.

[53] I. Stoica et al., “Chord : A Scalable Peer-to-Peer Lookup Protocol for Internet Applications,” vol. 11, no. 1, pp. 17–32, 2003.

[54] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Shenker, “A scalable content-addressable network.” ACM SIGCOM, 2001.

[55] A. Rowstron and P. Druschel, “Pastry: Scalable, Decentralized Object Location, and Routing for Large-Scale Peer-to-Peer Systems.” Springer Berlin Heidelb., 2001.

[56] S. I and E. Al, “Chord: a scalable peer-to-peer lookup service for internet applications.” in the 2001 conference on Applications, technologies, architectures, and protocols for computer communications, 2001, pp. 149–160.

[57] E. Meshkova, J. Riihijärvi, M. Petrova, and P. Mähönen, “A survey on resource discovery mechanisms peer-to-peer and service discovery frameworks,” vol. 52, pp. 2097–2128, 2008.

[58] Z. Yun, S. Huayou, Q. Hengnian, and N. Yulin, “An approach to discover semantic web services in distributed environment based on Chord,” Proc. 2010 IEEE Int. Conf. Intell. Syst. Knowl. Eng. ISKE 2010, no. 2009, pp. 401–405, 2010.

[59] Z. Zhengdong, H. Yahong, L. Ronggui, W. Weiguo, and L. Zengzhi, “A P2P-based Semantic Web Services Composition Architecture *,” vol. 1, pp. 403–408, 2009.

[60] N. Jafari Navimipour and F. Sharifi Milani, “A comprehensive study of the resource discovery techniques in Peer-to-Peer networks,” Peer-to-Peer Netw. Appl., vol. 8, no. 3, pp. 474–492, 2015.

[61] W. Zhang, S. Zhang, F. Qi, and M. Cai, “Self-Organized P2P Approach to Manufacturing Service Discovery for Cross-Enterprise Collaboration,” IEEE Trans. Syst. Man Cybern. Syst., vol. 44, no. 3, pp. 263–276, 2014.

[62] A. Xiang, L. Liu, and Q. Luo, “VPeers: A peer-to-peer service discovery framework for Virtual Manufacturing Organizations,” Comput. Ind., vol. 59, no. 5, pp. 411–419, 2008.

[63] V. Sahota, M. Li, M. Baker, and N. Antonopoulos, “A grouped P2P network for scalable grid information services,” Peer-to-Peer Netw. Appl., vol. 2, no. 1, pp. 3–12, 2009.

[64] A. Padmanabhan, S. Ghosh, and S. Wang, “A self-organized grouping (SOG) framework for efficient Grid resource discovery,” J. Grid Comput., vol. 8, no. 3, pp. 365–389, 2010.

[65] G. Di Modica, O. Tomarchio, and L. Vita, “Resource and service discovery in SOAs: A P2P oriented semantic approach,” Int. J. Appl. Math. Comput. Sci., vol. 21, no. 2, pp. 285–294, 2011.

[66] B. Yuan, L. Liu, and N. Antonopoulos, “A self-organized architecture for efficient service discovery in future peer-to-peer online social networks,” Proc. - 2016 IEEE Symp. Serv. Syst. Eng. SOSE 2016, pp. 415–422, 2016.

[67] P. Trunfio et al., “Peer-to-Peer resource discovery in Grids : Models and systems 6,” vol. 23, pp. 864–878, 2007.

[68] E. K. Lua, J. Crowcroft, M. Pias, R. Sharma, and S. Lim, “A survey and comparison of peer-to-peer overlay network schemes,” IEEE Commun. Surv. Tutorials, vol. 7, no. 2, pp. 72–93, 2005.

[69] Y. Zhang, Y. Qu, H. Huang, D. Yang, and H. Zhang, “An ontology and peer-to-peer based data and service unified discovery system,” Expert Syst. Appl., vol. 36, no. 3 PART 1, pp. 5436–5444, 2009.

[70] A. Furno and E. Zimeo, “Self-scaling cooperative discovery of service compositions in unstructured P2P networks,” J. Parallel Distrib. Comput., vol. 74, no. 10, pp. 2994–3025, 2014.

[71] A. Boukhadra, K. Benatchba, and A. Balla, “Ranked Matching of OWL-S Process Model for Distributed Discovery of SWs in P2P Systems,” 2014 17th Int. Conf. Network-Based Inf. Syst., pp. 106–113, 2014.

[72] K. Wu and C. Wu, “State-based search strategy in unstructured P2P,” Futur. Gener. Comput. Syst., vol. 29, no. 1, pp. 381–386, 2013.

[73] B. Yang and H. Garcia-molina, “Designing a Super-Peer Network,” 19th Int. Conf. on. IEEE, pp. 1–25, 2003.

[74] E. Ayorak and A. B. Bener, “Super peer web service discovery architecture,” Proc. - Int. Conf. Data Eng., pp. 1360–1364, 2007.

[75] K. Verma, “METEOR-S WSDI : A Scalable P2P Infrastructure of Registries for Semantic Publication and Discovery of Web Services,” vol. 6, pp. 17–39, 2005.

[76] K. Sivashanmugam and J. Miller, “Adding Semantics to Web Services Standards,” 2003.

[77] H. Madani and M. Author, “Toward a Better Automation of the Distributed Discovery Mechanism for Semantic Web Services,” pp. 88–93, 2010.

[78] A. Boukhadra, K. Benatchba, and A. Balla, “HPS5DSWS: A hybrid P2P strategy of the distributed discovery mechanism for semantic Web services,” Proc. - 2013 8th Int. Conf. P2P, Parallel, Grid, Cloud Internet Comput. 3PGCIC 2013, pp. 29–36, 2013.

[79] Y. Aklouf and H. M. Meghazi, “Wsmxdiscocast: A P2P approach for a better automation of the Discovery Mechanism for Web Service Execution Environment,” International Journal of Internet Protocol Technology, vol. 6, no. 1/2, p. 96, 2011. doi:10.1504/ijipt.2011.040618

[80] B. Soukkarieh, “Technique de l’internet et ses langages: vers un syst{è}me d’information web restituant des services web sensibles au contexte,” Universit{é} de Toulouse, Universit{é} Toulouse III-Paul Sabatier, 2010.

[81] F. M. Bouyakoub and A. Belkhir, “A service discovery approach based on a quantitative similarity measure for M-tourism platforms,” Communications and Network, vol. 04, no. 03, pp. 227–239, 2012. doi:10.4236/cn.2012.43027

[82] F. M. Bouyakoub and A. Belkhir, “A service discovery approach based on a quantitative similarity measure for M-tourism platforms,” Communications and Network, vol. 04, no. 03, pp. 227–239, 2012. doi:10.4236/cn.2012.43027

[83] E. Negre, “Comparaison de textes: quelques approches...,” Apr. 2013.

[84] E. Herrera-Viedma, F. Crestani, and G. Pasi, Soft computing for Web information retrieval. 2006.

[85] F. M. Bouyakoub and A. Belkhir, “A similarity measure for the negotiation in web services,” Multimed. Tools Appl., vol. 50, no. 2, pp. 279–312, 2010.

[86] A. Rolland, “Reference-based preferences aggregation procedures in multi-criteria decision making,” Eur. J. Oper. Res., vol. 225, no. 3, pp. 479–486, 2013.

[87] R. L. Keeney and H. Raiffa, Decisions with multiple objectives : preferences and value tradeoffs. Cambridge University Press, 1993.

[88] A. Rolland, “Aide à la décision multicritère et orientation des étudiants en IUT,” pp. 1–8, 2009.

[89] B. Roy, “Classement Et Choix En Presence De Points De Vue Multiples,” Rev. française d’informatique Rech. opérationnelle, vol. 2, no. 8, pp. 57–75, 1968.

[90] HENRIET, Laurent. Multi-criteria evaluation and classification systems for decision support: Model building and assignment procedures. 2000. PhD thesis. University of Paris Dauphine-Paris IX.

[91] D. E. Abdelli, F. M. Bouyakoub, T. A. Sifaqes, and Y. Hamza, “Evaluation tool for contextual similarity measures in web services discovery approaches,” 2017 International Conference on Mathematics and Information Technology (ICMIT), 2017. doi:10.1109/mathit.2017.8259712

[92] D. E. ABDELLI and F. M. BOUYAKOUB, “Improved multicriteria ranking based Web Service Discovery Approach,” 2018 3rd International Conference on Pattern Analysis and Intelligent Systems (PAIS), 2018. doi:10.1109/pais.2018.8598539

[93] P. Perny and B. Roy, “The use of fuzzy outranking relations in preference modelling,” Fuzzy Sets Syst., vol. 49, no. 1, pp. 33–53, 1992.

[94] L. Henriet. Systèmes d'évaluation et de classification multicritères pour l'aide à la décision. Construction de modèles et procédures d'affectation. thèse de doctorat, Université Paris Dauphine, France, 2000.

[95] [1] E. Al-Masri and Q. H. Mahmoud, “Crawling multiple uddi business registries,” Proceedings of the 16th international conference on World Wide Web, 2007. doi:10.1145/1242572.1242794