Context-Based Web Service Discovery Model

Amel Hannech, Hamid Mcheick
Computer science department
University of Québec at Chicoutimi
Chicoutimi (Québec) Canada
Amel.Hannech1@uqac.ca; Hamid_Mcheick@uqac.ca

Mehdi Adda
Mathematics, Computer science and Engineering department
University of Quebec at Rimouski
Rimouski (Quebec) Canada
Mehdi_adda@uqar.ca

***Abstract*—Web services offer a vast number of interoperable programs. The problem of web services is how to develop mechanisms to locate automatically the correct Web service in order to meet the user's requirements, that is appointed by the discovery of web services. Indeed, it is beyond the human's capability to manually analyse web services functionalities. This paper proposes an architectural model to assist the user by taking into account its constantly changing context. This model selects services based on the query semantics, which consist of preferences and context. These preferences may be digital, for example the price of a ticket when booking a flight or QoS desired.**

Keywords-component; Semantic Web service, ontologies, indexation, context, QoS.

I. INTRODUCTION

A service-oriented architecture (SOA) is the underlying structure supporting communication between services. SOA defines how two entities (programs) interact to enable one entity to perform a unit of work on behalf of another entity. Service interactions are defined using a description language. Each interaction is self-contained and loosely coupled, so that each interaction is independent of any other interaction [1].

The technology most commonly used and based on this architecture is that of Web Services [3]. According to [2] a Web Service can be defined as software designed to support interoperable machine interaction over a network. This interoperability is possible with the description language WSDL [4] and the communication protocol SAOP [5].

Architecture-based services meet the needs to satisfy flexibility and adaptability; it is flexible since a failed service can be replaced by another without changing the entire application. It is adaptable due to the fact that the selected service is chosen as the best in a given context.

Web 2.0 is a combination of technologies and tendencies where service-based architectures are widely used. This infrastructure allows naming new applications and appearances of Web such as Mashups (application whose content comes from a combination of sources), mobile applications, etc.

The concept of Web 2.0 can be enriched by the semantics of the domain [7, 8], the so-called Web 3.0 which aims to define and relate semantically resources and web services to simplify their use, discovery, integration and reuse in many applications [9]. This concept thus facilitates the search and selection of services in the SOA-based applications.

A. General problem

The main objective of the Service Oriented Architecture (SOA) is to reuse of offered services based on the fact that services are accessible on the web and may be used by a large number of users through a standard protocol. In order to be used, a service has to be previously described by its provider [10]. This is the second stage of web service life cycle, it is necessary for the service publication by its provider in a register and its subsequent selection by customers via this register. The general problem is how to develop mechanisms to automatically locate the correct Web service in order to meet user requirements, especially since it is beyond the capacity of humans to manually analyze Web services features; it is named by the Web services discovery.

Semantic web services and ontologies allow the sharing of web services to the context and use the concepts useful for search, communication and composition; we propose to base our architecture on the Semantic Web. But there is no established method for the acquisition of semantic Web service descriptions. The main topic of this paper is to explore the problem of acquiring a web service that best meets the user's request.

In the proposed approach we rely on the idea and assumption that one has to supplement web services with semantic description of the domain of interest and use context associated with our services using ontologies to facilitate their discovery and integration; we propose an architectural model to ease the selection of web services based on the query semantics that comprise use preferences and use context.

B. Contribution

The present study summarizes our proposition to represent the domain of interest and the context of use of a web service in the form of ontologies. Then, design these last ones as well as the system which is going to exploit them to answer the users' requests by taking into account their semantics. Thus, our work consists in representing in a first place the domain of interest and the context of use using the descriptive language OWL. This is done after choosing a construction method among the existing methods, then we shall pass to the design of

Identify applicable sponsor/s here. If no sponsors, delete this text box. (sponsors)

978-1-4673-1166-3/12/$31.00 ©2012 IEEE

the ontologie-based Services Search System. This paper proposes Web services classification that meets user needs according to a degree of satisfaction of the user.

C. Report organization

This paper is organized as follows. We discuss related work in section 2. We describe the proposed approach in section 3. A conclusion and future works are given in section 4.

II. RELATED WORKS

Web service discovery is a dynamic search field where various discovery mechanisms have been recently proposed in the literature. In [6], the authors defined discovery mechanism as "*the act of locating a web service description treatable by machine, not known before, and describing some functional criteria*".

Initially web service discovery was primarily syntactic. With the development of semantic Web technologies, the proposed techniques for web service discovery became essentially semantic (level of semantic similarity between query terms and semantic web service description).

The general principle of syntactic approach is to compare between the query syntax based on user's keywords and the syntactic Web Services description (WSDL).

In the approach proposed in [11], UDDI is used a central repository for publishing and discovering web services based on keywords. In the search phase, user or search program sends a query that consists of keywords; this query is compared with registry keywords. The search result is a set of web services descriptions; the user selects the web service that best meets its requirements.

The disadvantage of this method is that it may return a large number of results or, conversely, too few results. Another syntactic approach for discovering web services was proposed in [12] and is called AASDU. The AASDU (Agent Approach for Service Discovery and Utilization) is a multi-agent approach containing four components: a graphical user interface (GUI), an agent query analyzer (QAA), a system used to reference agents according to their expertise, where each agent has only knowledge of services related to their field of expertise, and the last component is the service module that offers to providers the capability to publish web services' descriptions. In this system, to answer a query Q, the user enters his search query as a string through the user interface. This request is sent to the QAA agent that extracts from this request relevant keywords. Agent QAA selects a set of expert agents. The selected agents transmit parameters of the services to which they are linked to the composition agent. This later, invokes a service according to user's choice.

Recent work has focused on semantic description web services and ontologies are mainly used to model the semantic service representation. It helps to establish semantic relations between concepts of the domain under consideration. We also have to mention that the OWL-S [13] approach that uses the ontology OWL-S to extend UDDI with semantic description of Web services.

It describes a web service using three classes, Service Profile is the class that provides the functional parameters for discovery such as enter expected, results produced, precondition and effects.

In this method, the discovery is based on a Matchmaking algorithm, which allows to find web services descriptions that have a semantic correspondence between functional parameters defined in the descriptions and those introduced in the search query.

Semantic correspondence between two concepts is based on the relationship between these ones in their respective OWL ontologies. The algorithm identifies four levels of semantic correspondence between two concepts, namely: Exact, Plug in, Subsume, and Disjoint.

At the last, web services are classified by semantic correspondence level between their output parameters and those cited in the query.

If two services have the same correspondence level with the request, a comparison on semantic correspondence level relative to the input parameters is performed. Another work adds the context parameter in the web services discovery process. In [14], the search for context-aware services is defined as the ability to use context information to find the most relevant services to the user.

The adaptation process is directly implemented in the mechanism of search services. This mechanism is based on the reference architecture of the service-based systems; the provider publishes its services on a server that is used by the user to send service requests. We summarize the steps of this work in the following steps: The provider must publish in the context manager the context in which the service can meet and conditions of use according to a description of the user and his/her environment. The web service description can be published on two levels: basic description expressed in a low-level language such as XML, or WSDL for Web services and semantic description (Semantic Service Description) expressed in OWL-S. The provider must publish in the register the two descriptions and the semantic service conditions of the use of the service, and the reference of its context. This information is stored in a server module called Service Provider – SP. The user must save its context (User Context) in the context manager (Context Manager) before he/she can make a request.

This request may be a basic query expressed in low-level search language (such as Service Location Protocol or UPnP62 - Universal Plug and Play) or query semantic: expressed as a semantic language query of high level (such as OWL-QL and RDQL). Then the user sends to the server a request and a pointer to his/her context. This information is stored in a server module named User. Once the user request is received, the search service enables the filtering engine service based on three filters: i)the filter base for selecting the category of service, ii)the semantic filter that returns a list of services that meet the exact specifications desired by the user, without the context information, and iii)context filter that refers to the list of services that match the context.

III. PROPOSED APPROACH

After giving an overview of some work around the problematic, we now may present the proposed architecture to perform a service discovery, in agreement with constraints given by the user in the form of requests. These constraints can be for example non-numeric values placed on the choice of departure and arrival cities when booking a plane ticket, or digital as constraints placed on the price ticket or the QoS desired (response time, security, etc. ..). It is also based on the use context and its changes. For constraints treatment, we have relied on the benefits of the web semantic (ontologies) and fuzzy logic.

This proposed architecture takes into account the user context and his/her preferences and offers him/her the ability to classify results. We start by presenting the domain of interest in the form of an OWL ontology. Then we present how this ontology is conceived from web services use context. To illustrate our architecture, we use a trip and tourism example.

A. Approach presentation

The goal through this example is to develop a service web discovery system based on domain ontology and use context. This system allows to index web services by calculating the satisfaction degree with regard to the various concepts that belongs to our ontology. To meet non numeric request constraints, the present proposal is based on the similarity degree calculation between two concepts in an ontology.

As for the digital constraints we are using fuzzy logic to its calculation. This part will be much detailed later. To better understand the topic we will present an example that consists on planning a trip. The first task consists at representing the domain of interst using an OWL ontology. To do so, we need to:

- Determine domain knowledge element and represent use context elements, to conceive and build so that the corresponding ontologies represented under OWL language
- Exploiting ontologies developed to index web services and thus be able later to locate relevant information
  in the list of web services related to interest domain
- Propose a classification of results according to user needs (preferences and use context)

B. User context definition and modeling

A widely accepted definition of context in the field of context-aware computing (context awareness) is: “context is any information that can be used to characterize the situation of an entity (person, place or object), considered relevant in the interaction of a user and an application” [15]. In the work presented in this paper, we are interesting in the use context as a characterization of the user himself and his access device to the system.

Use Context is associated with a session; a session is the act of connecting a user to the system. Use Context is associated with a session while a session can be associated with multiple use contexts. It is defined by a set of features that are related to either the user or the device used for a session and a feature is represented by a pair : attribute/value. The attribute specifies the name of the feature and its value is given by the facet of the same name. The user characteristics are: Static, Dynamic or Preference.

User static characteristics are recorded during the first session and remain unchanged. For example, static characteristic representing the user's first name is the same in each session. For instance, the personalized welcome message of the system to the user remains unchanged for all sessions.

Dynamic characteristics and user preferences are stored in the first session, but may change from one session to another or during the same session. If we consider an e-business system, if a user is under 25 years (dynamic characteristic representing user age) the system can offer preferential prices. The same user logs long later and he is now over 25 years; the system will not propose him the same preferential prices. By this example, we show that user context may change from one session to another. Another example of dynamic use context is user location. This dynamic feature may vary during a single session if the user uses a mobile device. Also, use context may change over time in the same session. Our representation of use context is based on the fact that it is dynamic and scalable since it may contain many characteristics. These characteristics are not fixed in advance and are defined by the system designer as needed.

C. System design

The proposed system consists of:

1) Four modules: indexing module, search module, enrichment ontologies module, and the satisfaction degrees computation and service classification module.

2) A set of domain ontologies {01, 02,…,0n} that represent different categories of service businesses.

3) To each domain ontology is associated a use context ontology.

4) Each of these ontologies a meta-database is assigned. The later contains synonyms or other words that car be used by the user.

5) RDF data representing all classes instances of our ontologies.

6) Ontologies and metadata base are possibly enriched by a domain expert.

7) A set semantic web service descriptions (OWL-S) stored in different registers and classified by domain of (business service categories). This classification takes advantage of a direct access to the sevrices.

8) Database indexing services with different ontological concepts.

*a) **Indexing module***

Ontologies improve considerably the relevance of the results in the search process; it is the reason why we opted for an indexing method by means of ontologies. The improvements one may expect are related to the fact that the indexing process takes into account the different concepts and relations between them supplied by the ontology. Consequently, contrary to methods based on the use of simple and static keywords matching when looking for services, the ontology-based indexing method takes into account the semantic relationships among query terms.

b) Concepts search and satisfaction degree computation

In this step we need the list of different concepts of our ontology of use context and interest domain corresponding to the domain of services to be processed. For this, we used a search engine called cores that can query an OWL ontology type and return a list of existing concepts. For every concept belonging to both ontologies we calculate its resemblance degree with service concepts to be indexed. This one is based on semantic similarity between two concepts. The computation process of the semantic similarity between two concepts is presented in Figure 1.

c) Updating index

In this phase, the index is implemented in a relational database. Each service is related to various concepts of our ontologies. Every concept is connected with all various interpretations in the meta-database. This information is later used when searching and selecting services. It is also used to calculate service satisfaction degree with regard to user complex requests. It is noteworthy that we designate by the *atomic request* a request that is composed of a single preference, and the term *complex request* is used to designate requests containing several preferences.

d) Search process

It represents the system interface with the user. Indeed, it is through the search process that the user expresses his needs by formulating requests and entering his/her preferences. A request is represented by a string. The display of results should be in a form that allows the user to easily exploit search results. Hence, the search process is completed by a service classification process. The search process is represented in the Figure 2.

This process consists of following stages:

Request lexical analysis: a graphical interface is available and is used to enter string queries. An entered query is lexically analyzed, This stage is necessary because a term in the query may have several forms in a text but its sense remains the same, thus it is enough to only use one of them to represent the concepts extracted from the query.

Use context identification: user session allows the identification of concepts related to the user context. These concepts are stored in the registry “context repository” and will be followed by the service “context manager” which is based on context comparison algorithm (current/predecessor) for detecting the change in state.

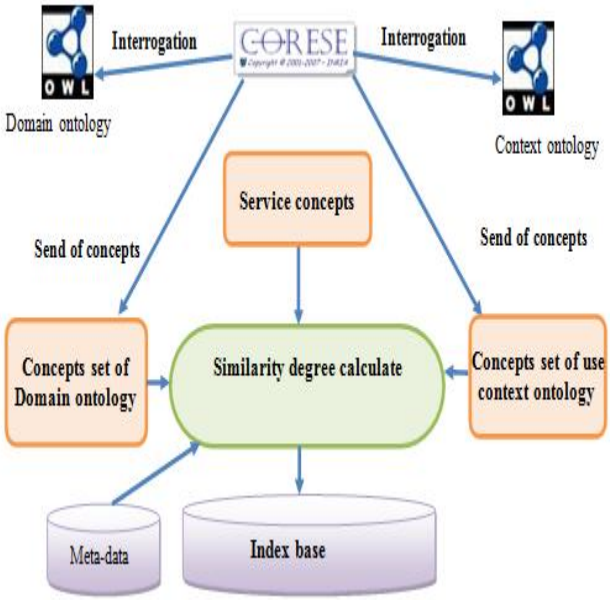


Figure 1. Indexing process

Generation of SPARQL query system: concepts extracted from resulting query in the previous stage are used to generate ASK SPARQL queries. The later allow the examination of the data sources to check the existence of the concepts.

Concepts search: search in our ontologies and metadata concepts related to query preference and context, this is done by calling semantic search engine CORESE that executes queries system generated in previous step.

Concepts display and classification: all concepts found are stored; a concept can be of type "class" or "instance". If a concept is of type instance is found, we add the associated class to the list of our list of found class concepts.

Sending unknown concepts to the enrichment module : concepts not found will be sent to the enrichment module for possible addition by the domain expert, it is done by calling CORESE system that executes a CONSTRUCT SPARQL query

Querying data sources: the first step consists at building SQL queries to query our index database using concepts of type "class" found earlier. Then, a list of services related to these concepts is returned. In the second stage, SPARQL queries are generated using the concepts of "instance" to select from the list returned by the first stage, services meeting these instances. This is done by querying our RDF data using two filters: user preferences and use context concepts.

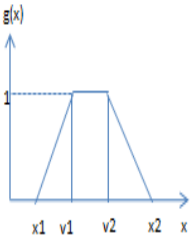
Calculating the query satisfaction degree: based on user request, we compute the degree of satisfaction of each service that is obtained from the filtering phase.

Processing the digital preferences: a user request may contain a numerical value of type: round (concept, v1), between (concept, v1, v2), max (concept, v1) or min (concept, v1). And to calculate service satisfaction degree towards a digital concept C_i, we need concept value (x) in the service process. Based on this this value, a satisfaction function g(x) is calculated. This function takes into account the value v_i required by the user.

Unlike Boolean logic, fuzzy logic allows a condition of being in a state other than true or false. There are degrees in the verification of condition. In our case it is used to calculate the value of g(x) that evaluates the degree to which a service web value (x) satisfies value (v) desired by user. G(x) is thus calculated according to its type (around, max, min …) and the value wished by the user. Inspired by the example given in Wikipedia [16] that explain how to consider the speed of cars, we give the following formulas:

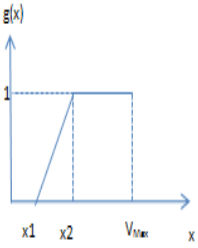
Between (Ci, v1, v2)

$$g(x) = \begin{cases} \frac{x - x1}{v1 - x1}, & x1 \le x \le v1 \\ 1, & x1 \le x \le x2 \\ \frac{x2 - x}{x2 - v2}, & v2 \le x \le x2 \\ 0, & x \le x1 \text{ et } x2 \le x \end{cases}$$



Max (Ci, V_{Max})

$$g(x) = \begin{cases} 0, & x \le x1 \\ 1, & x2 \le x \le Vmax \\ \frac{x - x1}{x2 - x1}, & x1 \le x \le x2 \end{cases}$$



With the same principle we can calculate g (x) of type Around (Ci, v1, v2), Min(C_i, V_{min})…etc.

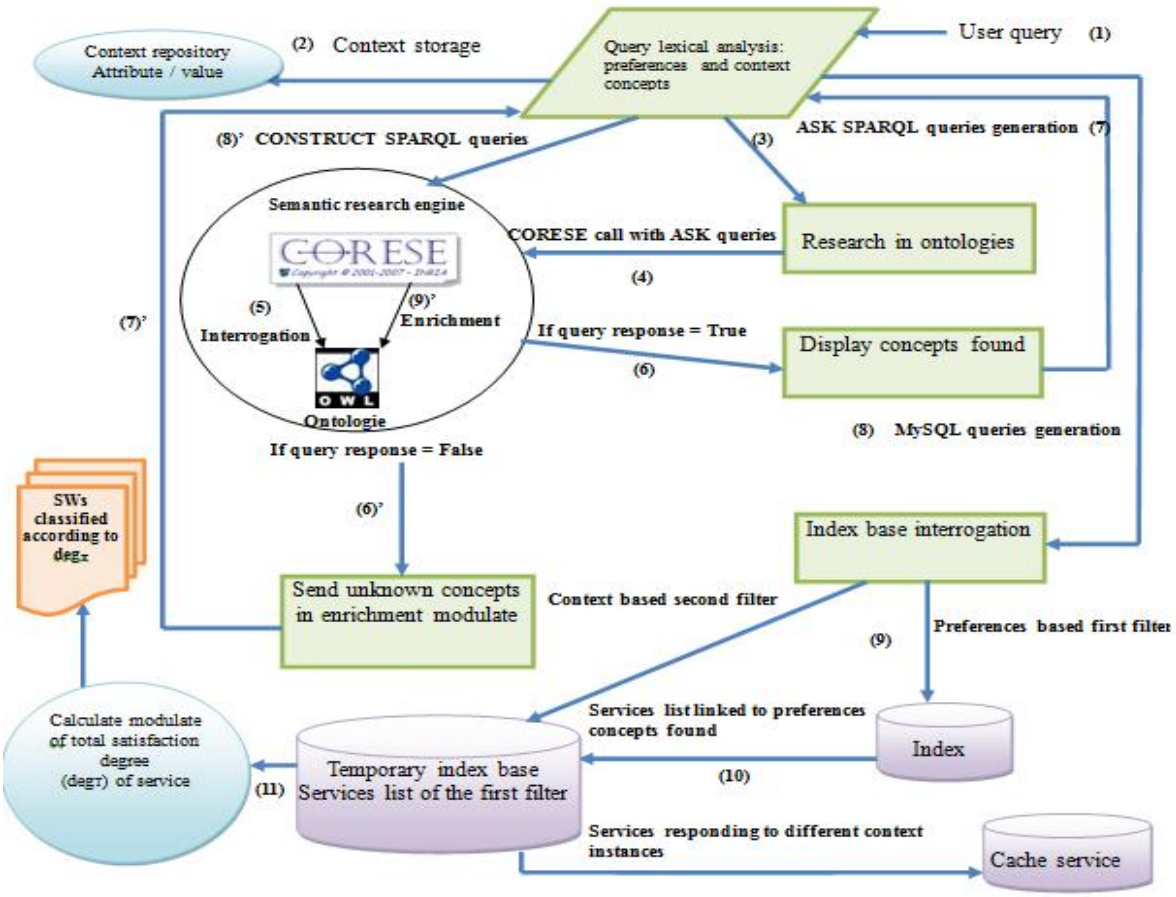


Figure 2. Search process

So in order to compute service satisfaction degree with regard to the user request, the user has to provide an ordered list of his/her preferences. An interest degree will be assigned to each element belonging to the list.

Given C_i a set of concepts extracted from the user request : $C_i = \{C_1, \ldots. C_i, \ldots. , C_n\}$, interest degree is assigned to each concept: $D_i = \{D_1, \ldots. D_i, \ldots., D_n\}$. To calculate concept interest degree, we ask the user to gather concepts that have the same level of importance and rank them. This is done using the two interpretations Pareto preference and Prioritized preference from preferences SQL language. Using Pareto preference we will have n subsets of concepts that have the same level of importance. These subsets will be ranked together with Prioritized preference. We get the ordered set C_i ': $C_i' = \{e_1, e_2, e_3, \ldots.., e_n\}$ when $e_1 = \{(C_1, C_2)\}$, $e_2 = \{(C_3)\}$, $e_3 = \{(C_4, C_5, C_6)\}$, $e_n = \{(C_i, \ldots. C_n)\}$.

The subset belonging level is equal to its ranking in C_i and is noted by Lev(e_i). The number *nb* represents the number of subsets e_i belonging to C_i'. Interest level of each subset ei belonging to Ci is equal to the division of its level of ownership in C_i on the total number of subsets belonging to C_i'.

$$Interest\ degree_{\ of\ concept} = \boldsymbol{Niv(e_i)\ /nb} \qquad \boldsymbol{(2)}$$

$$Request\ degree\ interest_{\ in\ service} = \sum_{i=0}^{N} \boldsymbol{(satisfaction\ degree_{\ concept\ i} * interest\ degree_{\ i})} \quad \boldsymbol{(3)}$$

Result classification: if the request is atomic (contains a single concept), the classification of services related to this concept is based on the recorded level of satisfaction of those services with regard to the single concept contained in the request. In the case where the query is a complex, the classification is done according to the satisfaction degree obtained from Equation (3).

e) Ontology enrichment module

The presented system needs the richest possible ontologies to have a better indexation and consequently make a better search. To that end, we propose a simple way to help domain experts enrich ontologies in addition to his own expertise on his/her field. Concepts not found in ontologies during search process are stored in a list and when domain expert connects for the enrichment he will find a list of these concepts and choose among them those suitable to the domain of interest. The effective enrichment of our ontologies is done using the SPARQL CONSTRUCT query system and the semantic search engine CORESE.

IV. CONCLUSION AND FUTURE WORKS

In this work, we presented the detailed steps we followed to model specific domain ontologies. Then, we showed how they may be integrated in a web service retrieval system. We opted for a modular architecture to build the system that consists of five main modules. The main idea was to solve the problem of web service discovery and be able to locate automatically the "correct" web service in order to meet user requirements. To that end, we proposed an architecture for a service selection system in accordance with user constraints and context by means of web semantic technologies. As of our perspectives, we are planning to evaluate the effectiveness of our approach using a real use case and real data. Also, we are planning to study how it may be generalized and extend its scope.

REFERENCES

[1] Papazoglou, M. P. Service-Oriented Computing: Concepts, Characteristics and Directions. In: Procs of the 4th International Conference on Web Information Systems Engineering (WISE 03), Dec. 2003, Washington, DC, USA, 2003. IEEE Computer Society, 2003, pp.3-12.

[2] Steve, J. Toward an Acceptable Definition of Service. IEEE Software, 2005,vol.22, n°3, pp.87-93.

[3] Erl, T. Service-Oriented Architecture: Concepts, Technology, and Design.Prentice Hall PTR, 2005, 760p.

[4] Chinnici, R., Moreau, J.-J., Ryman, A., Weerawarana, S. Web Services Description Language (WSDL) Version 2.0 Part 1: Core Language. W3C Recommendation [en ligne], 2007.

[5] Mitra, N., Lafon, Y. SOAP Version 1.2 Part 0: Primer (Second Edition). W3C Recommendation [en ligne], 2007.

[6] F.NewcomerE.ChampionM.FerrisC.Orchard : Booth, D.Haas. Web services architecture.

[7] Wahlster, W., Dengel, A. Web 3.0: Convergence of Web 2.0 and the Semantic Web. Technology Radar Feature Paper, Edition II/2006, Deutsche Telekom Laboratories, pp.1-23.

[8] Hendler, J.Web 3.0: Chicken Farms on the Semantic Web. Computer, 2008,vol. 41, n°1, pp.106-108.

[9] Berners-Lee, T., Hendler, J., Lassila, O. The Semantic Web. In: Scientific Americain, 2001.

[10] KaarthikSivashanmugam, KunalVerma, Amit P. Sheth, and John A. Miller. Adding semantics to web services standards. 2003, In *ICWS*, pages 395–401.

[11] Eric Newcomer. Understanding Web Services- XML, WSDL, SOAP and UDDI, chapter 5, Finding Web Services : UDDI Registry. Addison Wesley Professional, May.

[12] Paul Palathingal and Sandeep Chandra.Agent approach for service discovery and utilization.In *HICSS*, 2004.

[13] David Martin and al. Owl-s : Semantic markup for web services. Technical report, W3C, 2004.

[14] Vincenzo Suraci1, SilvanoMignanti, Anna Aiuto, University of Rome "Sapienza", Department of computer and system sciences, Context-aware SemanticService Discovery

[15] A. K. Dey, G. D. Abowd, and D. Salber. A Conceptual Framework and a Toolkit for Supporting the Rapid Prototyping of Context-Aware Applications. Human-Computer Interaction Journal, 16(1), 2001.

[16] http://fr.wikipedia.org/wiki/Logique_floue