Semantic Web Service Discovery Based on Agents and Ontologies

Rohallah Benaboud, Ramdane Maamri, and Zaïdi Sahnoun

***Abstract*—Web services are useless if they cannot be discovered. So, discovery is the most important task in the Web service model.Recent researchers have focused on performing semantic matching to enhance the accuracy of Web service discovery.In this paper we present a framework for Web services discovery and selection based on intelligent software agents, OWL-S and domain ontologies. With the help of software agents, information provided by Web services can be made more efficient and more dynamic.With semantics provided by OWL-S and domain concepts, match and discovery engine can return the most relevant services.**

***Index Terms*—web service discovery, QoS, agents, ontologies.**

I. Introduction

Service-oriented computing (SOC) is an interdisciplinary paradigm that revolutionizes the very fabric of distributed software development Applications that adopt service-oriented architectures (SOA) can evolve during their lifespan and adapt to changing or unpredictable environments more easily [1]. SOA is built around the concept of Web Services. Web Services are new forms of Internet software which can be invoked using standard Internet protocols. Web Services, as it is defined by the World Wide Web Consortium (W3C), is a software system designed to support interoperable machine-to-machine interaction over a network. Web services interact with each other, fulfilling tasks and requests that, in turn, carry out parts of complex transactions or workflows.

With the rapid development of web services technologies, discovering web services is becoming the most urgent problem to be resolved [2]. Discovery is the process of finding Web services Provider locations which satisfy specific requirements. Web services are useless if they cannot be discovered. So, discovery is the most important task in the Web service model [3]. There are two challenges facing the practicality of Web services discovery [4]: (a) efficient location of the Web service registries that contain the requested Web services and (b) efficient retrieval of the requested services from these registries with high quality of service (QoS). With ever increasing number of available web services it is problematic to find a service with required functionality and appropriate quality characteristics [5]. The main reason for this problem is that current web services technology is not semantic-oriented. Several approaches have been proposed to add semantics to Web Services descriptions to facilitate discovery and selection of relevant Web services (e.g. DAML-S [6], WSDL-S [7], WSML [8], OWL-S [9]). These so-called Semantic Web services can better capture and disambiguate the service functionality, allowing a logic-based matchmaking to infer relationships between requested and provided service parameters [10]. Recently, we have seen an explosion of interest in ontologies as artefacts to represent human knowledge and as a critical component in several applications; among these the web services [11]. Ontologies have been developed to provide a machine-processable semantics of information sources that can be communicated between different applications. They are also essential to the development and use of intelligent systems, particularly for the interoperation of heterogeneous systems. Ontologies are, thus, responsible for informing about the domain vocabulary and explaining the meaning that interacting systems attribute to terms [12]. In our work, we propose the use of ontologies to describe semantically the various parameters and characteristic of the customer request and the Web services.

Even the use of ontologies, machines or programs is still not efficient enough to take the automatic and dynamic decision for semantic Web service discovery. With the help of software agents, information provided by Web services can be made more efficient and more dynamic. Actually agents are intelligent enough to take the decision according to the changing environment and changing level of available information which cannot be expected from the traditional system[13]. Our work takes the multi-agent approach in which a team of agents, each with local information, collaborates to satisfy Web services discovery objective. The overall behavior of the system emerges through the dynamic interactions between agents.

If the discovery engine returned multiple candidate Web services provide the same functionality, then Quality of Service (QoS) is becoming an important criterion for selection of the best available Web service. The consumers have to pay enough attention to find the service provider who can satisfy their QoS requirements. If they cannot find a provider satisfying all their QoS requirements, they usually have to give up all the candidates or make some tradeoff [14]. Our work aims to provide a more "consumer-centric" approach simplifying service discovery using semantics while satisfying QoS requirements. A major problem in using QoS for service discovery is the specification and storage of the QoS information [15]. In this paper, we propose an ontology-based OWL-S extension to adding QoS to Web service descriptions.

Service consumers have different preferences. For

Manuscript received May 1, 2012; revised June 19, 2012.

R. Benaboud is with the Department of Mathematicsand Computer Science, University of Tebessa, Tebessa 12000, Algeria (e-mail: r_benaboud@ yahoo.fr).

R. Maamri and Z. Sahnoun are with LIRE Laboratory, Mentouri University ofConstantine, Ain ElBey 25000, Algeria(e-mail: rmaamri@yahoo.fr; sahnounz@yahoo.fr)

example, a service consumer may want a service that offers the fastest response time while for another execution price could be his most important parameter. For this reason, we propose that service consumer gives different weights for different QoS attributes when he formulates his request.

The rest of this paper is organized as follows: In section 2, we outline related research, Section 3 presents our framework. In Section 4, we present request and service description. Section 5 describes how to calculate the degree of similarity between request and Web servicesand section 6 offers concluding remarks and future directions.

II. Related Work

The problems pertaining to Web service discovery have long been taking attention of both academia and industry. Many researches have investigated the discovery of semantic web services, QoS-aware discovery or the use of agents for semantic web services discovery. We provide an overview of some of this work as a context for the research discussed in the remainder of the paper.

A. Semantic Web Services Discovery

Most current approaches for web service discovery cater to semantic web services, i.e., web services that have associated semantic descriptions. The work presented in [16] proposes a DAML-S matchmaking algorithm which is used to match a requested service with a set of advertised ones. This matching algorithm compares the input and output concepts of user request to the service description in registry and defines four levels of matching: Exact, Plug in, Subsumes, Fail. In our work, we don't use only the subsumption relationships between concepts to calculate their similarity but we also take into account common properties between them.

Reference [17] presents an approach for web service discovery that combines semantic and statistical association metrics. Semantic metrics are based on the semantic aspects of relevant ontology. Statistical association metrics are based on the association aspects of web services instances (their inputs and outputs). The proposed approach exploits semantic relationship ranking for establishing semantic relevance, and a hyperclique pattern discovery method for grouping web service parameters into meaningful associations. These associations combined by the semantic relevance are then leveraged to discover and rank web services.

B. QoS and Web Services Discovery

The QoS requirements for Web Services have become vital for both service providers and consumers as several Web Services offers similar functionality [18].

Kokash [19] proposes a QoS-aware discovery and subscription approach. The core idea of this approach is to build up a "virtual service" grouping function similar services together (called service pool) and dispatching customer requests to the proper service in terms of QoS requirements. After investigating the structure and the underlying semantic similarity of Web services, it employs a similarity matching algorithm to cluster the function-similar services and generate a virtual WSDL so that the service pool can be accessed as a Web service. Assuming consumers' QoS requirements are compliant with Web Service Quality Model (WSQM) [20], it designs algorithms to automatically finish the QoS negotiation between consumers and providers. However, using traditional standards such as UDDI and WSDL in this approach is insufficient, since the underlying semantics of Web services are not exploited enough. Also, the user requirements are not described explicitly and consistently.

The work described in [21] refers to the need for an extensible QoS model that contains domain-specific QoS criteria. It sustains that QoS must be represented to users according to user preferences and users should express accurately their preferences with this QoS model without resorting to complex coding of user profiles. It also suggests that QoS computation must be fair and open for providers and requesters. Then it proposes an extensible QoS model.

C. Agents and Web Services Discovery

Towards incorporating Web Service into agents has been the subject of many research projects. Reference [22] presents an agent based method for Web service selection, from the information that is given in the WSDL file by the Web service provider. Data mining is done on those data that are collected from WSDL files and feedback taken from the Web service users, by the agent to discover some interesting patterns for further users of the Web service. The author of this work doesn't provide the details of the matching mechanism and his impact on service selection.

In [23], Zhang et al. propose a multiagent approach for a distributed information retrieval task. In their work, each agent has a view of its environment called agent view. The agent-view structure of an agent contains information about the language models of documents owned by each agent. An agent-view reorganization algorithm is run to dynamically reorganize the underlying agent-view topology. Zhang et al.'s protocol does not use ontologies during information retrieval.

III. Agent-Based Framework for Service Discovery

Our proposed framework has two types of agents are devised namely, Consumer Agent and Provider Agent (Fig. 1). Like in [24], we use a central base of OWL Ontologies as a reference to develop the various local ontologies and semantic descriptions of the different Web services. Each Provider Agent implements a number of Web services described semantically with OWL-S enhanced with QoS attributes.The central ontologies base contains the different concepts used in diverse fields of proposed Web services and will be consulted periodically by Provider Agents to develop or enrich local ontologies.

When a service consumer wants to insert his request, An Ontology-Guided Interface is offered by the Consumer Agent. In order to input the request, service consumer must select the desired terms they want to use in his request from the list of terms provided by the interface in a pop-up. This list of terms is generated by Provider Agents using terms in local ontologies andsent to the Consumer Agent.

When receiving the request from Consumer Agent, each Provider Agent matches the request to the services in the Register Depository using OWL-S description and OWL

local ontologies. Then, Provider Agent returns to Consumer Agent a set of candidate services.

When receiving all responses from Provider Agents, the Consumer Agent sorts all candidate services according to the degree of functional similarityand the QoS score. Then, it returns to the service consumer the services that have the highest overall score.

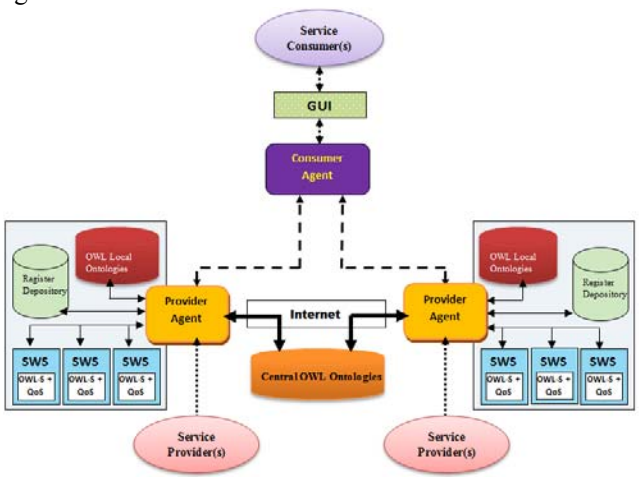

Fig. 1. Agent-based framework for WS discovery.

IV. Request and Web Service Description

Request description includes functional and non-functional requirements. The former describes the functional characteristic of the service demand, such as name, textual description, inputs and outputs. The latter mainly focuses on the customer's preferences, namely quality of service (QoS). In our work, service consumer doesn't have to give the value of each desired QoS attribute but he should get the means to specify that a QoS attribute is more important than another one. Indeed, he gives a weight for each QoS attribute. Weights range from 1 to 5 where higher weights represent greater importance.Fig. 2 shows an example of a user request interface with weights of some QoS attributes.

Typically, Web services are described using functional and non-functional properties. Functional properties contain Service Name, Textual description, a set of Inputs and a set of Outputs. Non-functional properties represent the description of the service characteristics (e.g. QoS).We use OWL-S service profile as a model for semantic matchmaking of service descriptions. However, OWL-S mainly focuses on describing functional aspects of a Web service. Based on works presented in [25] [26], we propose an ontology-based OWL-S extension to adding non-functional description, referring to as QoS, to Web service description. In OWL-S service profile we can use a set of ServiceParameter which has name "serviceParameterName" and a value "sParameter" (Fig. 3). For the connection of OWL-S and QoS ontology, the QoSProperty is a subclass of OWL-S ServiceParameter. And QoSParameterName and qosParameter are subproperties of OWL-S ServiceParmaerterName and sParameter property.

V. Degree of Similarity between Request and Web Services

In this section we present our matching mechanism, which evaluates the similarity between request and Web service advertisements. This is done using functional match and the calculation of QoS score.

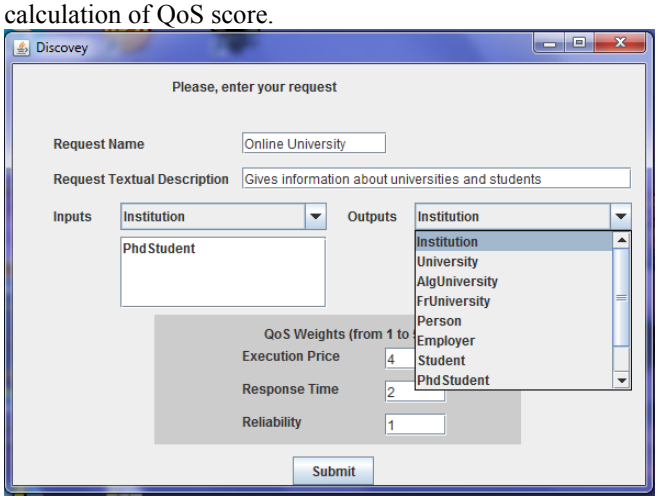

Fig. 2. Consumer request interface.

```
<rdf:RDFxml:base="http://www.daml.org/services/owl-s/1.1/Profile.owl>
……
<profile:serviceName>Universities_Service</profile:serviceName>
<profile:textDescription>
    This  service provide Information about universities and students……
</profile:textDescription>
……
<profile:serviceParameter> ……..
<profile:serviceParameterName> ……</profile:serviceParameterName>
    <profile:sParameterrdf:resource="……">
</profile:serviceParameter>
……
<profile:hasInputrdf:resource=""/>
……
<profile:hasOutputrdf:resource=""/>
……
</rdf:RDF>
```

Fig. 3. OWL-S profile example.

A. Functional Match

In this discovery step, name and textual description of request and services are matched using syntactic similarity function whereas inputs and outputs are matched based on conceptual similarity function.

1) Syntactic Similarity

Syntactic similarity function *between a request R and a* service S is calculated using the equation bellow:

SyntacticSim(R,S)=

$$\frac{SyntacticSim(R.Name,\ S.Name) + SyntacticSim(R.TextDescription,\ S.TextDescription)}{2}$$

To calculate the syntactic similarity of tow strings, we use the algorithm 1, where the function Subset(String) takes as input a string and devises it into sequences of three characters. The function card(L) returns the number of elements in the set L. In line 4, we normalize the SyntacticSim value to have a value in the range of 0 to 1.

Algorithm 1 SyntacticSim(String1, String2)

L1, L2, L3: sets of string;

Begin

1: L1 = Subset(String1);

2: L2 = Subset(String2);

3: L3 = L1 ∩L2;

4: SyntacticSim = $\frac{2*card(L3)}{card(L1)+card(L2)}$;

5: **return**SyntacticSim;

End

For example, to calculate SyntacticSim("FindAlgUniversity","FindAlgerianUniversity"):

L1={fin, ind, nda, dal, alg, lgu, gun, uni, niv, ive, ver, ers, rsi, sit, ity}.

L2= { fin, ind, nda, dal, alg, lge, ger, eri, ria, ian, anu, nun, uni, niv, ive, ver, ers, rsi, sit, ity}.

L3 = { fin, ind, nda, dal, alg, uni, niv, ive, ver, ers, rsi, sit, ity}.

$$\text{SyntacticSim} = \frac{2*13}{15+20} = 0{,}743.$$

2) Conceptual Similarity

An output in the request must not be consider as similar to a more generic output in the advertised service, while a request input could be consider as similar to a more generic advertised input [27]. We think also that an input in the advertised service must not be consider as similar to a more generic input in the request, while an output in the advertised service could be consider as similar to a more generic output in the request.

University ≡ Institution ∩(∀hasID.UniversityID) ∩ (=1hasID) ∩(∀hasName.Name) ∩(=1hasName) ∩ (∀hasPostcode.Postcode) ∩ (=1hasPostcode) ∩ (∀hasCourse.Course) ∩ (=1hasCourse)

AlgUniversity≡ University ∩(∀hasPostcode.AlgPostcode) ∩ (=1hasPostcode)

Person ≡ (∀hasAdress.Adress) ∩ (≥1hasAdress) ∩ (∀hasFirstName.Name) ∩ (=1hasFirstName)∩(∀hasLastName.Name) ∩(=1hasLastName)

Employer ≡ Person ∩ (∀hasEmployerID.EmployerID) ∩ (=1hasEmployerID)

Student ≡ Person ∩ (∀hasStudentID.StudentID) ∩ (=1hasStudentID)

PhdStudent≡ Student ∩ (∀hasThesisID.ThesisID) ∩ (=1hasThesisID)

GeographicArea≡ (∀hasGoName.Name) ∩ (=1 hasGoName) ∩ (∀hasCountryName.Name)∩ (=1 hasCountryName)

Location ≡GeographicArea∩ (∀hasAltitude. Altitude) ∩ (=1 hasAltitude) ∩ (∀hasLatitude. Latitude) ∩ (=1 hasLatitude) ∩ (∀hasLongitude. Longitude) ∩ (=1 hasLongitude)

Fig. 4. Part of sample AL ontology.

ConceptSim(A, B) function matches a concept A(A ∈ request inputs or outputs) against a concept B (B∈ service inputs or outputs) and returns the conceptual similarity of the two concepts. For illustration, let us take the example shown in Fig.5. All inputs and outputs refer to concepts of domain ontology, an example portion of which is shown in Fig.4in a logic description notions. The function nbprop(A) denotes the number of properties of the concept A.To calculate the ConceptSim(A,B) function, we distinguish several scenarios:

Case 01: **if**(A and B are same or they declared as equivalent classes) **then***ConceptSim(A,B)= 1*;

Example: ConceptSim(University, University) = 1.

Case 02:**if**(A and B are inputs **and** A is subclass of the concept B directly or indirectly) **then**

ConceptSim(A,B)= 1;

Example: Fig 5(a): ConceptSim(PhdStudent, Person) = 1.

Case 03:**if(**A and B are Outputs **and** A is subclass of the concept B directly or indirectly) **then**

$$\text{ConceptSim(A,B)} = \frac{\text{nbprop(B)}}{\text{nbprop(A)}};$$

Example: Fig.5(a): ConceptSim(AlgUniversity, University) = 0,80.

Case 04:**if**(A and B are outputs **and** B is subclass of the concept A directly or indirectly) **then**

ConceptSim(A,B)= 1;

Example: Fig.5(b): ConceptSim(University, AlgUniversity) = 1.

Case 05:**if**(A and B are inputs **and** B is subclass of the concept A directly or indirectly) **then**

$$\text{ConceptSim(A,B)} = \frac{\text{nbprop(A)}}{\text{nbprop(B)}};$$

Example: Fig.5(b): ConceptSim(Person, PhdStudent) = 0,60.

Case 06:**if**(A does not have a parent/child relationship with B, but both concepts have a parent concept C in common) **then**

$$\text{ConceptSim(A,B)} = \frac{\text{nbprop(A}\cap\text{B)}}{\text{nbprop(A}\cup\text{B)}};$$

Example: ConceptSim(PhdStudent, Employer) = 0,5.

Case 07:**if**(otherwise) **then**ConceptSim(A,B) = 0;

Example: ConceptSim(Person, University) = 0.

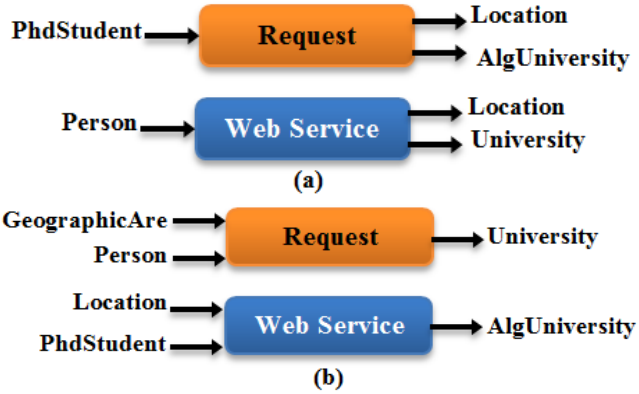

Fig. 5.Example of request and web service.

Algorithm 2performs an inputs matching.Where R.Inputs and S.Inputs denote the set of inputs in the request R and the set of inputs in the service S respectively, Card(E) denotes the cardinality of the set E, Sort(A) allows to sort the elements of the array A in descending order. In lines 1,2, 3 and 4, the algorithm matches each request input against all Web service inputs, and keeps the best mapping for each request input. If the number of request inputs is less than the number of service inputs, then we have a miss of information; thereforeInputsSim value is decreased (line 10).

The outputs similarity, given by OutputsSimfunction, is also calculated in the same way as inputs similarity. But when the number of service outputs is less than the number of request outputs, the value of OutputsSim is decreased. Therefore we inverse line 10 with 12 and perform changes in variable names in the algorithm 2.

Algorithm 2 InputsSim(R.Inputs, S.Inputs)

InSim: array of float;
Begin
1: **foreach** e_1 in R.Inputs**do**
2: **foreach** e_2 in S.Inputs**do**
3: $InSim_i$ =Max($InSim_i$,ConceptSim(e_1, e_2));
4: **end for**
5: i = i + 1;
6: **end for**
7: *Sort*(InSim);
8: *m* = Card(R.Inputs) – Card(S.Inputs);
9: **if** m<0 **then**
10: InputsSim = $\frac{\sum_{j=1}^{Card(R.Inputs)} InSim_i}{Card(R.Inputs)}/(|m|+1)$
11: **else**
12: InputsSim = $\frac{\sum_{j=1}^{Card(S.Inputs)} InSim_i}{Card(S.Inputs)}$
13: **end if**
14: **return**InputsSim
End

Let us calculate the Inputs and Outputs similarity between request and service shown in Fig5(a).

InputsSim = ConceptSim(PhdStudent, Person) = 1.

OutputsSim=

$$\frac{\text{ConceptSim(Location, Location)} + \text{ConceptSim(AlgUniversity, University)}}{2}$$

$$= \frac{1+0,8}{2} = 0,9.$$

InputsOutputs similarity is calculated based on InputsSim and OutputsSim functions as follows:

$$InputsOutputsSim(R,S) = \frac{InputsSim(R.Inputs,\ S.Inputs)+OutputsSim(R.Outputs,\ S.Outputs)}{2}$$

Functional similarity can be calculated using the equation bellow. Where weights w1 and w2 are real values between 0 and 1 and must sum to 1; they indicate the degree of confidence that the service consumer has in the syntactic similarity and inputs and outputs similarity.

$$FunctionalSim(R,S) = w1 * SyntacticSim(R,S) + w2 * InputsOutputsSim(R,S)$$

B. QoS Score

Each QoS value needs to be normalized to have a value in the range of 0 to 1. A QoS attribute can be monotonically increasing or decreasing. A monotonically increasing QoS attribute means increases in the value reflects improvements in the quality (ex. Reliability), while monotonically decreasing means decreases in the value reflects improvements in the quality (ex. Execution Price and Response Time). Monotonically increasing QoS attribute are normalized by Equations (1) and monotonically decreasing QoS attribute are normalized by Equations (2). In addition, qos.max value and qos.min value show the maximum and minimum value of the QoS attribute between all candidate services.

NoramizedValue(qos)=

$$\begin{cases} 1 - \frac{qos.\max - qos}{qos.\max - qos.min} if(qos.\max \neq qos.\min) \\ 1 \qquad\qquad\qquad\qquad if(qos.\max = qos.\min) \end{cases} \quad (1)$$

$$\begin{cases} 1 - \frac{qos - qos.min}{qos.\max - qos.min} if(qos.\max \neq qos.\min) \\ 1 \qquad\qquad\qquad\qquad if(qos.\max = qos.\min) \end{cases} \quad (2)$$

To calculate the overall QoS score of the service S, each normalized QoSattribute, qos is multiplied the corresponding weight, w, given by a service consumer as shown by Equation bellow:

$$QoSScore(S) = \frac{\sum qos * w}{\sum w}$$

VI. SUMMARIES

Web service discovery is a hot topic in past a few years. Seeking the right service based on user's search criteria in which the user may be interested is still a problem[28]. This paper has introduced a framework for Web services discovery based on intelligent software agents and ontologies. An advanced feature of our framework is that we perform the service discovery, selection and ranking based on the matching level of service advertisements to user requests both in terms of functionality and QoS. With the use of agents, information provided by Web services can be made more efficient and more dynamic. With the use of OWL-S and domain ontologies, match and discovery engine can return the most relevant services.

To implement our framework, we have used JADE platform [29]. JADE is a very powerful middleware framework built with Java to design a MultiAgent Systems based architecture. Consumer Agent and different Provider Agents are created with JADE and inherent "Agent" JADE class. Each agent has a "match module" which is realized using Jena APIs which provides plenty of methods to access ontology files. Registers Depository of our system are stored in Oracle 10g.

In our future research, we will have to incorporate the Web services composition into our proposed framework, by doing that, we hope that our based agent framework can be further improved and become more practical in real-world applications.

REFERENCES

[1] M. Junghans, S. Agarwal, and R. Studer, "Towards practical semantic web service discovery," *In Proc. the 7th International Conference on The Semantic*, pp. 15-29, 2010.

[2] Ye. L and B. Zhang. "Discovering web services based on functional semantics," In *Proc. the 2006 IEEE Asia-Pacific Conf. on Services Computing*, pp. 348-355, 2006.

[3] V. Sharma and M. Kumar, "Web service discovery research: A study of existing approaches," *Int. Journal on Recent Trends in Engineering and Technology*, vol. 1, pp.106-109, 2011.

[4] H. Wang, Y. Zhang, and R. Sunderraman, "Extensible soft semantic web services agent," *Soft Comput*, vol. 10, pp. 1021–1029 ,November 2005.

[5] N. Kokash, A. Birukou, and V. D'Andrea, "Web service discovery based on past user experience," *In Proc. the 10th international conference on Business information systems*, pp. 95-107, 2007.

[6] A. Ankolekar et al, "DAML-S: Web service description for the semantic web," *In Proc. 1st International Semantic Web Conf.*, Italy, pp. 348-363, 2002.

[7] W3C, Web Services Semantics -- WSDL-S. [Online]. Available: http://www.w3.org/Submission/WSDL-S/

[8] J. de Bruijn, Holger Lausen, A. Polleres, and D. Fensel, "The web service modeling language WSML: An overview,"*In Proc. 3rd European Semantic Web Conf.*, Budva, Montenegro, pp. 590-604, 2006.

[9] D. Martin et al, "Bringing semantics to web services: The OWL-S approach,"*In Proc. of 1st International Workshop on Semantic Web Services and Web Process Composition conf*., USA, pp. 26-42, 2004.

[10] A. Averbakh, D. Krause, and D. Skoutas, "Exploiting user feedback to improve semantic web service discovery," In *Proc. 8th International Semantic Web Conf.*, pp. 33-48, 2009.

[11] B. Benatallah, M. Hacid, A. Leger, C. Rey, and F. Toumani, "On automating web services discovery," *The International Journal on Very Large Data Bases*, vol. 14, pp. 84 – 96, March 2005.

[12] A. Malucelli, "Ontology-based services for agents interoperability," Phd thesis. University of Porto. 2006.

[13] U. Marjit, A. Sarkar, S. Santra, and U. Biswas, "A goal driven framework for service discovery in service-oriented architecture: A multiagent based approach," *International Journal of Computer and Communication Technology*, vol. 1, pp. 251-256, August 2010.

[14] X. Liu, L. Zhou, G. Huang, and H. Mei, "Consumer-centric web services discovery and subscription," *In Proc. the IEEE International Conference on e-Business Engineering*, Hong Kong, pp. 24-26, 2007.

[15] Z. Xu, P. Martin, W. Powley, and F, Zulkernine, "Reputation-enhanced QoS-based web service discovery," *In Proc. the International Conference on Web Services*, pp. 249- 256, 2007.

[16] M. Paolucci, T. Kawamura, T. Payne, and K. Sycara, "Semantic matching of web services capabilities," *In Proc. the 1st International Semantic Web Conf*., pp. 333-347, 2002.

[17] A.V. Paliwal, N. R. Adam, H. Xiong, and C. Bornhovd, "Web service discovery via semantic association ranking and hyperclique pattern discovery," *In Proc. the 2006 IEEE/WIC/ACM International Conf. on Web Intelligence*, pp. 649-652, 2006.

[18] T. Rajendran and P. Balasubramanie, "An efficient multi-agent-based architecture for web service registration and discovery with QoS,"*European Journal of Scientific Research,* vol. 3, pp.421-432, 2011.

[19] N. Kokash, "Engineering service-oriented systems: Modeling, discovery and quality," Phd thesis, International Doctorate School in Information and Communication Technologies, DIT - University of Trento, February 2008.

[20] OASIS: Web Services Quality Model (WSQM) TC. [Online]. Available: http://www.webkorea.or.kr/pds/data/pds1/WSQMver-0.3_20050909143621.doc

[21] Y. Liu, A. H. H. Ngu, and L. Zeng, "Qos computation and policing in dynamic web service selection," *In Proc. the 13th international World Wide Web conf. on Alternate track Papers and Posters*, New York , pp. 66-73, 2004.

[22] S. Susila, "Agent based discovery of web service to enhance the quality of web service selection," *IJCSNS International Journal of Computer Science and Network Security,* vol. 2, pp. 159-163, 2011.

[23] H. Zhang, W.B. Croft, B. Levine, and V. Lesser, "A multi-agent approach for peer-to-peer based information retrieval system," *In Proc. Third Int'l Joint Conf. Autonomous Agents and Multi-Agent Systems*, pp. 456-463, 2004.

[24] M. Gharzouli and M. Boufaida, "PM4SWS: A P2P model for semantic web services discovery and composition," *Journal of Advances in Information Technology*, vol. 2, pp. 15-26, 2011.

[25] L. Lin, S. Kai, and S. Sen, "Ontology-based QoS-aware support for semantic web services," *Technical Report at Beijing University of Posts and Telecommunications*, 2008.

[26] Y. Zhang, H. Huang, D. Yang, H. Zhang, H. Chao, and Y. Huang, "Bring QoS to P2P-based semantic service discovery for the universal network," *Journal Personal and Ubiquitous Computing,* vol. 7, pp. 471–477, 2009.

[27] N. Ouldahmed and S. Tata, "How to consider requester's preferences to enhance web service discovery," *In Proc. the second International Conf. on Internet and Web Applications and Services*, pp. 59-64, 2007.

[28] V. Sharma and M. Kumar, "Web service discovery research: A study of existing approaches," *Int. J. on Recent Trends in Engineering and Technology*, vol. 05, pp. 106-109, Mar 2011.

[29] F. Bellifemine, G. Caire, T. Trucco, and G. Rimassa, (2003). Jade Programmer's Guide. [Online]. Available: http://sharon.cselt.it/projects/jade/.