A Semantic Matching Algorithm for Discovery in UDDI

Unai Aguilera
Tecnológico – Fundación Deusto
Avda. Universidades, 24
48007 Bilbao, Spain
uaguiler@tecnologico.deusto.es

Joseba Abaitua, Josuka Díaz,
David Buján, Diego López de Ipiña
Universidad de Deusto
Avda. Universidades, 24
48007 Bilbao, Spain
abaitua@fil.deusto.es
{josuka, dbujan, dipina}@eside.deusto.es

Abstract

*One of the key objectives of web service technology is to construct processes that enable service providers to inter-connect with their clients. The industry has developed the UDDI standard that allows providers to register their services and make them available to their clients. However, the search mechanism supported by UDDI is limited; it only enables the user to search using keywords. Keyword-based searches frequently return no results because they are not able to recognize the similarities and differences between the concepts used by the providers to describe their web services and the concepts used by the clients during the search process. This paper presents a semantic registry which extends the functionality of an UDDI registry adding semantic reasoning. These functionalities are introduced by means of semantic descriptions of the information stored in the registry. Previous work done on the combination of the UDDI registry with semantic information is extended in two ways: **f**irstly, we propose a generic semantic discovery algorithm that is not restricted only to the inputs and outputs of the web services or to the OWL-S ontology. This matching algorithm can be applied to the discovery process using different ontologies because it is not bound to any specific ontology or concepts from it. Secondly, through the construction of an ontology which represents the information associated with a business entity in UDDI, and which enables users to register and search business information semantically using the former algorithm.*

1. Introduction

In recent years, the web service technologies have considerably grown in their application in the enterprise world. The goal of web services is to enable heterogeneous applications to intercommunicate easily. The application of this technology is very adequate for the B2B (Business to Business) solutions where providers and clients, with very different characteristics, need to communicate among themselves to perform various tasks. In this scope, where a client can select between many different providers, a process is necessary to help in the selection of the services that are more adequate to perform the desired task. The industry (guided by OASIS Consortium [1]) has developed a standard that permits providers to register their services and make them available to their clients. This system is called UDDI (Universal Description Discovery and Integration) [2]. UDDI is similar to a "yellow pages" where providers advertise themselves and which are used by the clients to search services that can fulfill their needs. When the provider performs a register the service is described using some keywords. During the search process the client specifies some keywords that describe the service he is searching for, and a string base comparison is performed to retrieve the services that meet this requirements.

However, the possibilities of this approach are limited. The UDDI string-based search is not adequate in most cases because it only permits keywords to be used in the search process combined with a classification system [3]. The keyword based search frequently returns no results because it is not able to recognize the similarities and differences between the concepts used to describe the web services. During recent years some efforts have been aimed at the solution to this problem introducing semantics into the registration and search processes in the

web services registry thanks to the use of shared ontologies to represent the concepts and the relations between them.

In this paper we present the SemB-UDDI project which extends the functionality of an UDDI registry adding semantic capabilities. These new functionalities are introduced by means of the semantic description of the information stored in the registry. The goal of this approach is to avoid the limitations that exist in the string-based search of the UDDI registry. The introduction of semantics can add new possibilities to the matching process. These semantics are introduced by means of the use of ontologies. An ontology is a conceptualization of a specific domain that is shared among all the participants. This way ontologies enable the participants to know the significance and relation between the terms and concepts being used. In the case of a UDDI registry, the use of ontologies enables the matching algorithm to discover the differences and similarities between the concepts used by the clients in their searches and the concepts used by the service providers when they described their services. This way, in their searches, users can employ concepts that are not exactly equal to the concepts used by providers in their descriptions but are related to them thanks to the ontology.

In addition, an UDDI registry not only stores information about the services, but it stores other information related to them (e.g. Business Entity). For these reason, we think that a completely semantic registry must semantically manage all the entities involved in the process. So, we have constructed an ontology to represent the information related with a Business Entity, enabling the semantic register and search of these entities.

In SemB-UDDI, when a user wants to search for a service or business entity he must construct a semantic description of the entity he wants to find. This description is matched against the descriptions of the services and business entities registered by the providers. We have developed a semantic matching algorithm that enables the matching between user request and registered descriptions to be performed in a general form and without knowing the specific ontology being compared. This way, this algorithm can be used to match descriptions of different ontologies, so it is applied to the search processes of service and business entities.

The remainder of the paper is structured as follows: Section 2 presents the related technologies and previous work done in the area of adding semantic capabilities to UDDI registries. Section 3 presents the SemB-UDDI architecture explaining the connection between the elements. Section 4 discusses the new algorithm proposed to perform the semantic matching process. Section 5 presents the ontologies used in the register including the ontology developed to represent the Business Entity information. Finally, Section 6 summarizes and concludes the paper.

2. Background

2.1 UDDI

The Universal Description Discovery and Integration is an open industry initiative guided by OASIS that enables businesses to publish their own services and discover other services. When the services are registered in the UDDI registry the clients can search them by name, description, the business offering them and other related information. This search mechanism supported by UDDI is limited [3]: it only enables the user to search using keywords. The user normally does not know the words that the service provider used to describe the service when it was registered. When a user starts a search in UDDI he needs to have some knowledge of the words that were used to describe the services or businesses. In most cases, this knowledge is only partially available to the client, or is not available at all. For this reason, it is very likely that in his search, the client uses words that are not present in the description of the service and, as result, the keyword based search will discard many results that might be useful for the client. This is because the search algorithm is not able to recognize similarities or differences between the significance of the keywords used by the client and the keywords used by the provider when the service was registered. In other words, the search algorithm does not know if the client and the service provider are using a common semantic when referring to the same kind of services. UDDI only works well when the client knows some information about the services, but it fails as a discovery strategy in other cases.

2.2 OWL-S

OWL-S [6] is an ontology for web services constructed using OWL [4] and [5] and it has been developed to enable the following tasks: automatic service discovery, automatic service invocation and automatic service composition. The service discovery is improved using ontologies because the information needed to perform this task is expressed using a machine-processable form. A computer can access the description of a web service and it can know exactly what the service does thanks to the shared concepts contained in the ontologies used in the description.

A service described using OWL-S provides three types of knowledge: Service Profile, Process Model, Service Grounding. The ServiceProfile describes what the service does, including functional information such as inputs, outputs, and other non-functional information (category,

classification). It is normally used during the automatic discovery of web services. The Process Model describes how the service works; it is an abstract vision of the service operation. Finally, the ServiceGrounding tells how to access the service; it contains all the information related to the real implementation of the service and is used to invoke it automatically.

2.3 Related work

In the last few years, some work has been done in the area of semantic web services discovery. In this section we present a short review of these works.

One proposal for enhancing UDDI with semantic information is made in [7] and in [3]. In these papers the authors present an architecture to augment UDDI registries with additional semantic information. They add a new layer to the UDDI architecture that performs the semantic matching between service records. When an advertisement of a service containing semantic information is received, the information is extracted and stored in the registry. The services are described using OWL-S, and only the ServiceProfile is used to perform the discovery process. They use a one-to-one mapping if the information contained in the OWL-S profile has an equivalent in the UDDI registry. For those OWL-S profile elements that do not have a correspondence with UDDI registry elements a T-model based mapping is used. The same authors present a matching algorithm which they argue is efficient for semantic web service discovery [8] and is an evolution of the algorithm presented in [9]. The authors define four degrees of match for the result of the discovery process: *exact*, *plugin*, *subsume* and *fail*, ordered from the best to the worst result. The matching algorithm contains some optimizations, such as the indexation of the registered services, to improve the discovery process. Another solution is presented in [10], where the authors propose a similar solution to the former, but they use a filtering mechanism that progressively reduces the set of registered services being matched to improve the matching algorithm. The filtering mechanism used is similar to that developed in [11].

Another approach for a semantic UDDI registry is presented in [12] and is based on the proposal made in [13], but it extends the UDDI API to support the semantic based inquires and a planning algorithm to help users in web service composition is introduced.

Another work related to union of UDDI and the semantic web is presented in [14]. It presents an external matching mechanism to enhance the search in a service registry, and to permit the integration of multiple external engines. Another algorithm is presented in [18] where the authors propose a more grained ranking of results for the semantic matching adding more possible results than these that can be obtained performing only a subsume matching. The proposed algorithm performs a matching of the ServiceProfile as a whole, taking into account the service classes in addition to the inputs and outputs of the service.

In this paper we present an enhanced UDDI registry that uses some of the algorithms explained here, like the roots of the algorithm developed for the SemB-UDDI registry. We have developed a more general algorithm that not only performs semantic matching between inputs and outputs, but also the semantic matching between other concepts presents in the OWL ontology. So it can be used to match other concepts present in the OWL-S profile or other ontologies that represent other semantic information. The proposed algorithm is explained later in the section about the semantic matching algorithm.

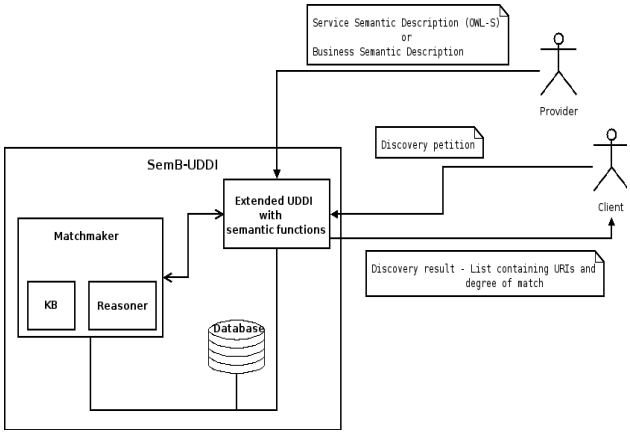

Figure 1: SemB-UDDI architecture

3. SemB-UDDI architecture

The architecture of the SemB-UDDI registry is shown in Figure 1. It extends the functionality of the standard UDDI registry by adding semantic capabilities. It adds new functions that enable the clients to register advertisements and search among them using the new semantic capabilities of the register. These new functions use the matchmaker module to perform the semantic process. The providers add the semantic descriptions by means of these functions. These descriptions are processed by the extended UDDI registry and sent to the matchmaker in order to store them in the knowledge base. When the client wants to discover a service or business entity he must construct a semantic description of the entity he wants to find, using the OWL-S ontology for semantic service or the developed ontology for business. This description is sent to the SemB-UDDI register and processed by the extended UDDI that uses the matchmaker to perform the matching process. The result of the matching is returned to the user in a list containing

the URI of the discovered descriptions that have a relationship with the request sent by the client, and the degree of match obtained in the matching process.

The matchmaker module performs the semantic matching functions. It has a functionality to register the semantic descriptions in its knowledge base, to compare them with a user request and to unregister them when necessary. It contains the algorithm used to perform the matching process. This algorithm, which will be explained later, performs the semantic comparison between the user request and the registered descriptions and obtains a result that expresses the degree of semantic similarity.

Internally, the matchmaker has various components that contribute to obtaining the functionality: a reasoner that performs the logic reasoning over the information contained in the ontologies stored in the knowledge base. It is used by the semantic matching algorithm to perform the comparison between user request and the registered descriptions. In the current implementation of the matchmaker module we are using Pellet as the reasoner for the matching operations [15]. The reasoner also contains a knowledge base that stores the ontologies and the related information about them. When a new instance (service description or business description) is stored the related ontologies are also downloaded and stored in the knowledge base to allow the reasoner to work with them. The knowledge base of the reasoner is not persistent so the ontologies and instances need to be reloaded every time the reasoner is restarted. For this reason, the matchmaker module includes a database that is used in order to maintain persistence. Whenever a new ontology or instance is stored in the matchmaker module the database stores the same information to provide persistence. When the semantic matching module is restarted this information is retrieved from the database and loaded enabling the reasoner to start with the information stored in previous sessions.

3.1 Extending UDDI with semantics

The SemB-UDDI project extends the UDDI standard API to enable the registration and search of registered descriptions (services and business). These functions have been added by means of modifying an existing implementation of UDDI called jUDDI [16]. In order to enable the clients to use these new functions we have extended an implementation of a UDDI client called UDDI4J [17]. This way a user can access the new capabilities of the semantic register. The new functions added to the SemB-UDDI registry are the following:

save_semantic_service: this function enables a semantic web service description to be added to the register. The service description consists of an OWL-S Profile. When the description is added to the registry all the syntactic information is extracted from the description and stored using the internal UDDI functions. This is done in order to maintain compatibility with the traditional UDDI registry. The OWL-S profile will be stored in the matchmaker module in order to enable semantic matching.

find_semantic_service: this function enables to search a service using the semantic capabilities of the SemB-UDDI register. The user needs to specify the description of the service that he wants to find and the registry will use the semantic matching module to obtain the degree of match between the user request and the descriptions stored in the knowledge base. The user request is an OWL-S Profile describing the service being sought. The correct execution of this function returns a list of the results obtained in the matching process. Only the services whose results are not *fail* are returned to the user. The returned list is a list of pairs containing the URL of the ServiceProfile and the degree of match obtained in the matching process with the user request.

save_semantic_business: this function enables business descriptions to be stored in the registry. According to the UDDI specifications a Business Entity is an entity that offers one or more services. In this case, the user sends a semantic description of the business which he wants to register. This description must use the Business ontology explained above. When the business description is sent to the registry, it is stored in the semantic matching module to permit semantic reasoning, and in the UDDI registry to maintain compatibility with the UDDI standard.

find_semantic_business: this function enables a business entity to be searched using the semantic capabilities of the registry.

There are also functions to remove the stored service descriptions and the business descriptions named delete_semantic_service and delete_semantic_business respectively.

4. Matchmaker algorithm

The semantic matching process has been implemented using an algorithm based on the idea proposed in [7] and [18]. In the first paper the authors propose a solution for semantic service discovery where the matching process is only performed between the inputs and outputs of the services being matched. In the second, an algorithm that matches the Service Profile as a whole is proposed, but it is restricted to OWL-S ontology. We propose an algorithm where the matching is performed with all the concepts included in the user request. The goal of the proposed algorithm is to compare instances (registered descriptions and user request) in a generic form without specifically knowing the ontologies and concepts being compared. That is, the proposed algorithm is able to match instances of any class and it is not only restricted to

OWL-S or other ontologies. This way, it can be used to match other ontologies like the business ontology that has been developed to represent the UDDI Business Entity information.

In the SemB-UDDI registry all the advertisements (services and businesses) are stored in the knowledge base of the matchmaker module. The semantic advertisements consist of a description constructed using OWL-S. In the case of a semantic web service this description is an OWL-S Service Profile. As explained above, the service profile contains all the information needed to perform the discovery of a web service, so this is the information used to perform the discovery of the web services in the registry. In the other case, when registering business entities, these are described using the ontology constructed for this objective. Whenever a client wants to search the registry for a semantic web service or a business entity he must construct an instance, using the corresponding ontology, to describe the service he is looking for. For example, if the user wants to search a semantic web service he must construct a service profile to describe the service that he wants to discover. Or, if he wants to discover a business entity he must construct a description using the business ontology. This description is matched against the stored descriptions in the knowledge base to search the registered instances that match the user's request. During the process of semantic matching the instances, classes, properties and values of the registered advertisements and the user request are compared to determine the degree of match between them. The results of the match are ranked from the best to the worst possible result in ascending order. The following table summarizes the values assigned to each of the possible results of the match. This rank will be used by the algorithm to compute the global result of the matching process.

Table 1: Ranking of match results

Result	Rank
Exact	0
Plugin	1
Subsume	2
Fail	3

4.1 Algorithm main loop

The matching algorithm is a generic algorithm for the comparison of instances that performs matching without knowing the ontologies and concepts being compared. The algorithm starts retrieving from the knowledge base the instances that have some semantic relationship with the instance contained in the user request. First, it obtains the class of the user request and then it retrieves from the knowledge base those instances whose classes are related with it. For example, if the user sends a search request that is an instance of an OWL-S ServiceProfile, the algorithm will retrieve the instances whose class is a super-class, a sub-class or is equal to that of the user request. Each of these retrieved instances is compared with the user request in order to obtain the degree of similarity between them. If the global result computed during the matching process is different from fail, the result is added to a list of results and returned to the user. The main loop of the matching algorithm is presented in the following figure.

```
computeMatch(request) {
 matchlist = []
 advertisements = getRelatedInstances(request)
 for all advert in advertisements {
  gResult = matchInstance(advert,request)
   if (gResult != fail)
    matchlist.add(matchResult)
  }
  return matchlist
}
```

Figure 1. Matching algorithm main loop

The *matchInstance* function computes the degree of similarity between two instances. This similarity is obtained in the following way: first, the relationship between the classes is obtained, and second, the properties of the instance are matched. If the match of the classes returns fail, the matching process stops because the user request cannot be satisfied. To obtain the final result of the process we compute the worst result of the two semantic comparisons, that is, the result with the highest rank.

```
matchInstance(advert, request) {
  result = matchClasses(advert, request)
  if result != fail {
   matchResult = matchProperties(advert, request)
   if matchResult > result
                result = matchResult
  }
  return result
}
```

Figure 2. Matching instances

4.2 Matching classes

The classes of the instances being compared are matched in the following way: if the class from the request and the class from the advertised instance are equal the result is exact; if the class of the request subsumes the class of the advertised instance, the result is *plugin*, because the user is searching for instances of a more generic class and it can be satisfied by any of its subclasses; if the registered instance subsumes the class of the user request the result is *subsume*, which is a worse result than the former because the user is searching for a more specific class than the one being compared. If none of these cases occurs the result is *fail*.

```
matchClasses(advert, request) {
   aClass = advert.getClass()
   rClass = request.getClass()
  if aClass = rClass return exact
  if aClass subsumes rClass return plugin
  if rClass subsumes aClass return subsume
  otherwise return fail
}
```

Figure 3. Matching classes

4.3 Matching properties

After computing the relation between the classes of the instances, and only when a result different from *fail* is obtained, the algorithm will proceed to perform a comparison between the properties from the user request and the registered instance.

```
matchProperties(advert, request) {
   properties = request.getProperties()
  forall reqProp in properties {
   advProp = advert.getProperty()
   if isDataProperty(prop)
     matchResult = matchDataProperties(advProp, prop)
    if isObjectProperty(prop)
      matchResult = matchObjectProperties(advProp, prop)
  }
   if matchResult > result
    result = matchResult
}
```

Figure 4. Matching properties

This comparison is performed by obtaining each of the properties contained in the user request and comparing them with the properties of the same type that exist in the advertised instance. There are two cases when comparing properties from instances in OWL: data properties, which have simple values, and object properties which have values that are instances.

When comparing literal values the matching is done by the *matchDataProperties* function, and it is performed in the following way: the associated values of the property are obtained from the user request and the registered instance. If the values are not concepts from an ontology, a literal comparison occurs. This literal comparison can only return two possible results - *exact* or *fai* -, that correspond to the case when the two values are exactly equal or different, respectively.

If the literal values of the properties being compared are concepts from an ontology, a semantic matching is performed to obtain the degree of similarity between them. In this case, the result of the comparison is one of the following:

Table 2: Matching concepts

Result	Rank	Explanation
Exact	0	Request concept and the registered concept are exactly equal.
Plugin	1	Registered concept subsumes the concept of the user request.
Subsume	2	Request concept subsumes the concept of the registered service.

Fail	3	None of the previous cases.

When comparing object properties, the match is performed re-using the same algorithm explained in this section. The instances are retrieved and compared using the *matchInstance* function again in a recursive manner. This way, it is possible to compare all the user request with the registered instances without knowing the concepts being compared.

The matching process is always performed taking the user request as a description that contains all the minimum requirements that a semantic description registered in the knowledge base must satisfy in order to be returned to the user. If the user request contains a property with a specific value only those instances that have the same property with the same value can be returned to the user. If the instance does not have an associated value with this property the result of the matching will be *fail*, because the user is searching for an instance with specific characteristics that cannot be completely satisfied with the registered instance.

The final result of all these semantic comparisons of instances, classes, properties and their associated values are combined using the worst result. The result obtained by the user after performing a matching request is the worst of all the results obtained when comparing the classes, properties and instances. The reason for this is that the user request contains all requirements that must be satisfied by the instance in order to be returned to the user. The instances that obtain a *fail* result are not returned to the user because they are not able to satisfy the user request.

5. SemB-UDDI ontologies

The information stored in the SemB-UDDI registry is described using various ontologies, some of them re-used from existing ones while others have been constructed expressly to represent specific entities required in the semantic registry. The description of the web services, used in the register and discovery processes is performed with the ServiceProfile of the OWL-S ontology. OWL-S is a very general ontology created for the description of any kind of web service and, for this reason, it only introduces some concepts and relations that are common and useful for all web services without considering their specific domain of application. When OWL-S is used to describe a web service another ontology is needed to capture the particular information that is related to that domain of application. Those domain ontologies are used with OWL-S to represent the concepts for the inputs, outputs and other information necessary to describe the web service semantically.

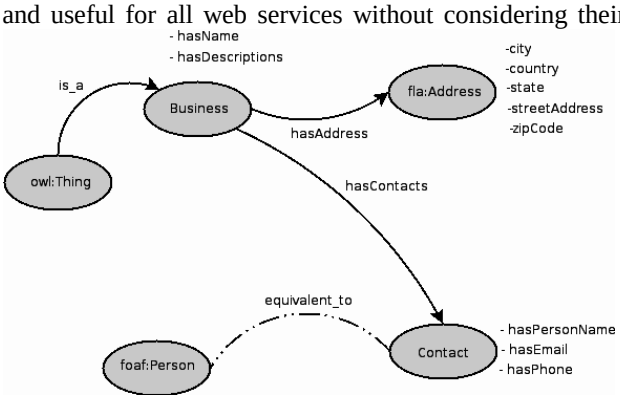

Figure 5. SemB-UDDI Business Ontology

As explained before we consider that a semantic registry must semantically manage all the information of the entities that are being stored. The UDDI standard defines the entities, the relations between them, and how they are stored in the registry. Those specifications include web services and some other elements as business entities. The business entity, as explained in the UDDI standard specification, represents the business that offers the services. We think that the semantic description of this entity is also needed to achieve the goal of a complete semantic registry for web services. This way a user can search all the data contained in the registry semantically without the need to consider the distinction between what information is semantically described and what not. So, for this reason, we think that an ontology to describe the information that corresponds to a business entity is needed. We have created an ontology to represent this information based on the specification of a business entity contained in the UDDI standard. We have represented this information because we want the SemB-UDDI registry to be able to maintain compatibility with a standard UDDI registry. This ontology enables the minimum information associated with a business entity to be captured while it enables the possibilities of the semantic web to be applied. Using the proposed ontology to represent the data we are able to capture all this information adding semantic searching capabilities at the same time. The Business ontology is depicted in Figure 5.

6. Conclusions and future work

In this article we have shown how a UDDI registry can be extended to allow the registry and semantic search of web services and related business entities. We have proposed a generic matching algorithm that allows the discovery of the registered entities to be made. This algorithm makes a comparison between all the concepts that appear in the user's request allowing a greater flexibility in the searches. In addition, this algorithm is not only related to a specific ontology and, therefore, the same algorithm can be used to make the semantic discovery of web services and business entities. This algorithm has been implemented completely and it is being tested. In addition, we have proposed an ontology for the description of the business entities, which allow basic

information about business presented in the UDDI standard to be captured and at the same time enables to support the possibilities of the semantic web.

In future work we plan to test the proposed algorithm in discovery tasks based on other parts of the OWL-S ontology, like the ServiceModel sub-ontology. The ServiceModel ontology describes how to use the service, and it contains useful information for service composition tasks. So, we will try to prove the usefulness of the proposed algorithm in discovery oriented towards automatic composition of semantic web services.

7. References

[1] OASIS - Organization for the Advancement of Structured Information Standards. http://www.oasis-open.org/

[2] OASIS, UDDI Committee Specification [Julio 2002] UDDI Version 2.04 API, Specification, http://uddi.org/pubs/ProgrammersAPI-V2.04-Published-20020719.htm

[3] Naveen Srinivasan, Massimo Paolucci, and Katia Sycara. "Adding OWL-S to UDDI: implementation and throughput". In Proc. 1st Intl. Workshop on Semantic Web Services and Web Process Composition (SWSWPC 2004), pages 6–9, 2004.

[4] W3C, World Wide Web Consortium [Febrero 2004] OWL Web Ontology Language Overview – W3C Recommendation, (http://www.w3.org/TR/owl-features/).

[5] W3C, World Wide Web Consortium [Febrero 2004] OWL Web Ontology Language Reference – W3C Recommendation, (http://www.w3.org/TR/owl-ref/).

[6] W3C, World Wide Web Consortium [Noviembre 2004] OWL-S Semantic Markup for Web Services – W3C Member Submission, (http://www.w3.org/Submission/2004/SUBM-OWL-S-20041122/).

[7] M. Paolucci, T. Kawamura, T. Payne, and K. Sycara. Importing the Semantic Web in UDDI. In Proceedings of the Workshop of Web Services, E-Business, and the Semantic Web, 2002.

[8] Naveen Srinivasan, Massimo Paolucci and Katia Sycara, "An Efficient Algorithm for OWL-S based Semantic Search in UDDI" First International Workshop on Semantic Web Services and Web Process Composition (SWSWPC 2004) 6-9, 2004, San Diego, California, USA.

[9] Paolucci, M., Kawamura, T., Payne, T.R. and Sycara, K. "Semantic Matching of Web Services Capabilities". In Proceedings of the 1st International Semantic Web Conference (ISWC2002)

[10] T. Kawamura, J. D. Blasio, T. Hasegawa, M. Paolucci, K. Sycara, "Public Deployment of Semantic Service Matchmaker with UDDI Business Registry", Proceedings of 3rd International Semantic Web Conference (ISWC 2004), LNCS 3298, pp. 752-766, 2004.

[11] K. Sycara, S. Widoff, M. Klusch, and J. Lu. "Larks: Dynamic matchmaking among heterogeneus software agents in cyberspace" Autonomous agents and multi-agent systems, 5:173–203, 2002.

[12] Rama Akkiraju, Richard Goodwin, Prashant Doshi, and Sascha Roeder. "A method for semantically enhancing the service discovery capabilities of UDDI". In In Proceedings of the Workshop on Information Integration on the Web, pages 87–92, August 2003.

[13] M. Paolucci, T. Kawmura, T. Payne, and K. Sycara. "Semantic Matching of Web Services Capabilities". In The First International Semantic Web Conference, 2002.

[14] J. Colgrave, R. Akkiraju, and R. Goodwin. "External matching in UDDI". In Intl. Conference on Web Services, 2004.

[15] Bijan Parsia and Evren Sirin. Pellet: An OWL DL Reasoner. Poster In Third International Semantic Web Conference (ISWC2004), Hiroshima, Japan, November 2004.

[16] jUDDI: http://ws.apache.org/juddi/

[17] UDDI4J (http://uddi4j.sourceforge.net/).

[18] Yonglei Yao, Sen Su, Fangchun Yang. "Service Matching Based on Semantic Descriptions", Advanced International Conference on Telecommunications and International Conference on Internet and Web Applications and Services (AICT-ICIW'06), February 2006, pp. 126.