The International Conference on Advanced Wireless, Information, and Communication Technologies (AWICT 2015)

Shared-Repository based approach for storing and discovering web Services

Moussa KAOUAN¹, Djelloul BOUCHIHA² and Sidi Mohamed BENSLIMANE³

¹,² Ctr Univ Naama, Inst. Sciences and Technologies, Dept. Mathematics and Computer Science, Algeria
³ EEDIS labo, University of Sidi Bel-Abbes, Algeria
{kaouan.moussa; bouchiha.dj}@gmail.com
benslimane@univ-sba.dz

Abstract

The Web services technology uses a set of standards, namely SOAP, WSDL and UDDI. In particular, UDDI is a platform for storing and retrieving Web services. Thus, the problem of Web service discovery is often attached to the UDDI. As Web services are developed through Internet, users must be able to effectively access and share these services. The production and interoperability of a large number of Web services has led to the emergence of new standards of publication, discovery and use of Web services. Therefore, mechanisms to distribute services are necessary for efficient selection of the appropriate Web service to the customer needs. In this paper, we propose a shared repository (ontology), which can be exploited by the UDDI for the storage and the discovery of a Web service. A matching algorithm is also proposed to show how to perform this operation. A prototype tool has been developed and experiments have been carried out to evaluate and show the efficiency of our approach.

Keywords: Web Service; UDDI, Shared Repository; Ontology; Service Discovery; Service Storage;
1. Introduction

The UDDI\(^1\) (Universal Description, Discovery and Integration) plays a very important role in the Web service environment. It can be used to publish and find services. Most of the current UDDI models are centralized. Therefore, their performance decreases if there are too many services listed or interviewed. Centralized structure is less robust, and expresses poor interoperability. The current UDDI is a passive "directory service"; this makes difficult the service change detection. The future UDDI will attempt to reduce disadvantages of the centralized approach. In this context, several solutions have been proposed. One solution is to replicate all the information to put them in different registries. Although this solution improves temporarily the UDDI performance, it implies a high cost in deployment and maintenance. The UDDI data replication is a non-progressive and inconsistent approach. The use of distributed technologies (P2P systems) seems a more efficient solution. However, with the lack of a common model between nodes, interoperability becomes a difficult goal. The distribution of information on different sites requires the existence of a shared model that can be used as a scheme to distribute or reconstruct information; absent trick in centralized systems. The old architecture of service discovery can not be extended in distributed systems. The success of the distribution of Web service directories is collateralized by the construction of a new architecture with a shareable repository model. All the works that have addressed the discovery issue are limited by the fact that Web services that provide the same functionalities may have different interfaces and operations. Inversely, the services that have the same interfaces can provide completely different functionalities [11, 21]. This is a major flaw that affects in particular the Web service discovery. Thus, interfaces and operations can be ambiguous sources for the discovery. For this, we have to annotate interfaces or use semantic descriptions of services, such as OWL-S and WSMO. OWL-S and WSMO provide a language or ontology for describing services without providing a model that can be shared between multiple sites. In addition, if we use an ontology (OWL-S or WSMO) to describe a single service in two different sites, these two definitions may not be consistent, given the detail level of conceptualization and perception used by both sites. The creation of a coherent model that can be shared as distribution and composition schema is necessary for such distributed environment of service registries. In this paper, we propose a reference model that can be shared by different UDDIs. This reference model (ontology) has been designed to assist the UDDI to store and discover Web services. Using a shared repository can unify the spirit of Web service discovery between UDDIs. Matching algorithm has been implemented to allow the storage and the discovery of Web services. The remainder of this paper is organized as follows: The following section summarizes similar works. Section 3 presents the proposed ontology and matching algorithm. Section 4 describes the tool supporting our work. Section 5 explains the followed steps to evaluate the implemented matching algorithm. In Section 6, we conclude our work and give our perspectives.

2. Related Work

The distributed UDDI strategy is adopted to overcome the degradation caused by the overload of resources (network & registry). It also overcomes the problem of failure of a centralized UDDI node. Implementing registries in a distributed environment gives rise to many problems, including concurrency, query processing, replication, reliability, scalability, etc. [16]. These problems can be solved using distributed technologies and P2P. Furthermore, there is not a standard distributed architecture for a distributed UDDI. METEOR-S Web Service Discovery Infrastructure (MWSDI) provides transparent access to public and private registries of Web services [20]. Canturk and Senkul describe a new mechanism for service discovery based on P2P architecture [8]. The architecture adopted by Sellami et al. consists in structuring registries of Web services into groups with the same ontology in a P2P network [19]. Sapkota et al. propose to use the notion of shared space for Web services [18]. Bianchini et al. propose a discovery mechanism based on P2P architecture, and based on interoperability between nodes [5]. Ying al. propose Chord4S, a decentralized approach for the service discovery based on P2P. The famous Chord P2P protocol is improved to increase data availability. Chord4S supports QoS upon discovery. In addition, the Chord routing

\(^{1}\) http://www.uddi.org/
protocol is extended to support efficient discovery of several services with a single query. In [6], distributed Discovery is based on the OWL-S ontology matching in different machines. DA5DCSWS, which implements the automatic discovery and composition, contains four modules: User Interface, Semantic Description of WS, Automatic Discovery of WS, and Automatic Composition of WS. Folino et al. propose an efficient model for discovering data mining Web services. This model proposes a Chord DHT extension that takes into account taxonomy of data mining services [9]. Boukhadra et al. propose an evolutionary and distributed approach in P2P network for the discovery of semantic Web services [7]. They propose to use OWL-S ontology matching, and collaboration between different pairs in P2P network. This allows creating a collaborative space, where each pair can use the experiences of others, to reduce the search space. Benghida and Boufaida propose a process to discover mobile Web services, based on three concepts: P2P Networks, Mobile Web Service, and Clustering [4]. The proposed approach is well suited to mobile devices and network, and it is characterized by the multicast communication. Lin proposes a Web service discovery model that combines the two technologies, semantic Web service and P2P [13]. The approach consists in creating a registry infrastructure for publishing and discovering Semantic Web services (OWL-S). The idea is to use specific domain ontology for each register. The following table summarizes approaches that have addressed the Web service discovery issue:

<table>
<thead>
<tr>
<th>Approach</th>
<th>Goal</th>
<th>Model</th>
<th>Tool</th>
<th>Architecture</th>
<th>Storage Method</th>
<th>Advantages</th>
<th>Disadvantages</th>
</tr>
</thead>
<tbody>
<tr>
<td>[9]</td>
<td>Efficient discovery of data mining services</td>
<td>WS-Chord</td>
<td>DHT</td>
<td>Shared</td>
<td>Distributed</td>
<td>Efficient discovery in large-scale networks</td>
<td>Specific Taxonomy* Model to data mining services</td>
</tr>
<tr>
<td>[7]</td>
<td>Distributed discovery</td>
<td>OWL-S ontology matching</td>
<td>OWL-S</td>
<td>P2P</td>
<td>Distributed</td>
<td>Collaborative work between nodes architecture purely distributed and heterogeneous</td>
<td></td>
</tr>
<tr>
<td>[13]</td>
<td>Distributed discovery</td>
<td>Domain Ontology + P2P</td>
<td>OWL-S</td>
<td>P2P</td>
<td>Distributed</td>
<td>Scalable method for accessing registers</td>
<td>The WS description is represented in WSDL &quot;one flat description&quot;</td>
</tr>
<tr>
<td>[4]</td>
<td>Discovering Web services in a mobile environment</td>
<td>Clustering +P2P + mobile web services.</td>
<td>JXTA</td>
<td>P2P</td>
<td>Distributed</td>
<td>Mobile network environment</td>
<td></td>
</tr>
<tr>
<td>[22]</td>
<td>High availability equilibrate the charge between registers</td>
<td>Chord4S OWL-QoS</td>
<td>P2P-Chord</td>
<td>P2P</td>
<td>Distributed</td>
<td>Improved Chord to support QoS high availability the discovery of several similar services</td>
<td></td>
</tr>
<tr>
<td>[6]</td>
<td>Composition and distributed discovery</td>
<td>OWL-S ontology matching</td>
<td>OWL-S</td>
<td>P2P</td>
<td>Distributed</td>
<td>Collaborative work between nodes architecture purely distributed and heterogeneous</td>
<td></td>
</tr>
<tr>
<td>[8]</td>
<td>Reduce the search time, Domain ontology</td>
<td>P2P</td>
<td>Hybrid</td>
<td></td>
<td></td>
<td>Single Interface, update the QOS.</td>
<td>High construction</td>
</tr>
</tbody>
</table>
quality of result, balance the load on the registries.

[19]
Reduce the search time, balance the load on the registries. Domain ontology + recommendation system JXTA YASA4W SD platform P2P Hybrid Reasonable response time Less relevant results

[18]
Reliable, flexible, and scalable discovery, synchronous and asynchronous communications. Shared space RDF Shared Distributed New strategy for the discovery and the composition Security Issues

[5]
Semantic Interoperability between peers Semantic links between peers (intra-peer and inter-peer links) P2P-SDSD VS Gnutella P2P Hybrid Relevant results as all registries are queried Difficulty to define semantic links Flexibility problems, requiring to establish mediation

[20]
Semantic interoperability between peers Extended ontology + federation OWL P2P + federation Distributed Single Interface

The considered criteria in Table 1 are: (1) the Goal of the approach in question, (2) the used Model, (3) whether the approach is supported by a Tool or not, (4) the Architecture, (5) the Storage method, (6) the Advantages and (7) the Disadvantages of each approach.

3. The proposed approach

The service discovery issue is very similar to the information retrieval issue. Behind it lie two sub-issues:

- Identification of needs: what we want to find, is it well expressed? Is it well interpreted?
- Identification of information: what is available; is it well presented? Is it well classified to be found?

The conventional information retrieval issue is solved as a classification issue (index, block). Thus, in our context, we assume that: if we better classify services, we distinguish them and we can better discover them. The classification is one of the solutions adopted to solve artificial intelligence problems. The problem of discovery is seen as an artificial intelligence problem, where it is possible to use classification as a solution. To enable efficient and flexible classification of services, we have built a shared repository, modeled by an ontology of services that has been deployed on a distributed architecture to better store Web services and thus reduce the time of the service discovery.

3.1. The Service Representation Model

An ontology is created to allow the knowledge sharing and reuse [12]. Thus, designing an ontology for services can lead to a shared service architecture. To address the problem of registries distribution, the use of service ontology is set out in several works, including OWL-S [14], WSMO [17] and WSDLs [1]. In our approach, we propose a 4-level pyramidal model for the representation of Web services (see Fig. 1).
Enterprise level: where each Enterprise can express several Domains.
Domain level: where each Domain is identified by a key D and contains several Abstract Services.
Abstract Web Service level: where each Abstract Web Service is identified by a key C and contains several published Web Services.
Web Service level: where each Web Service is identified by the couple <D, C>.
The layers, Domain and Abstract Web Service, represent the shared repository (ontology of services) for knowledge representation in Web services. The same repository will be used for publication, storage and discovery.

3.2. The functioning principle

The functional architecture of our Web services publishing (storage) and discovery system is based on three layers (see Fig. 2): the service layer, the knowledge layer and the storage/discovery layer. This layered representation will allow us to easily design the system, reduce the complexity of development and ensure the independent development of each layer.
3.2.1. The Service layer:

The purpose of this layer is to prepare services (to store or discover) to be assigned to a specific domain [3].

3.2.2. The Knowledge layer:

Based on the proposed ontology of services (SO), this layer provides a key \(<D, C>\) after matching the Web service (WS) with the ontology SO. The ontology of services SO is defined as a triplet \(<D, SD, AS>\), where:

- D (domain) is the name of an area of interest for enterprises and users (e.g. tourism, education, etc.)
- SD (sub-domain) is a sub-space which reduces the search space of a domain D (e.g. the domain tourism has as sub-domains: transport, restaurants, hotels, etc.)
- AS (abstract service) is a classification of semantically similar services.

3.2.3. The Storage/Discovery layer:

Regardless of the two upper layers, this layer selects the storage/discovery units to satisfy a publication or discovery request using the key \(<D, C>\) provided by the knowledge layer.

4. Implementation

The implementation of the approach is performed in three phases: constructing the ontology of services, implementing the matching algorithm, and finally, defining the structure of the UDDI.

4.1. Constructing the ontology

To implement a semantic Web service, several proposals can be adopted, notably WSDL-S, OWL-S and WSMO. In our case, we choose OWL-S. Three reasons have motivated our choice, namely standardization of OWL-S by W3C, the description of the interfaces necessary in our approach (see Section 3.2), and the abstract description of a part of service. The following is an excerpt of the SO OWL ontology, manually built, for storing and discovering Web services:

```xml
...<owl:Ontology> ...
<owl:Class rdf:ID="grocerystore_breadorbiscuitquantity_service.owls">  
  <rdfs:subClassOf>  
    <owl:Class rdf:ID="Food"/>
  </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="queryParserLocation.owls">  
  <rdfs:subClassOf>  
    <owl:Class rdf:ID="Geography"/>
  </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="_fodder_USimportservice.owls">  
  <rdfs:subClassOf>  
    <owl:Class rdf:about="#Food"/>
  </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="Hotel">  
  <rdfs:subClassOf>  
    <owl:Class rdf:ID="hebergement"/>
  </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="getMapOfAddress.owls"> <rdfs:subClassOf> 
  <owl:Class rdf:about="#Geography"/>
</owl:subClassOf>
</owl:Class>
...```

Listing.1. An OWL excerpt of the ontology of services OS
4.2. The matching Algorithm

The main idea of our proposal is that the Matching is performed not on all stored Web services, as is the case in conventional UDDIs, but on the ontology SO (see Algorithm 1). The ontology shows a simplified view of the stored services. This will reduce the search space, and consequently will reduce the discovery time that is a critical factor in a distributed system.

Algorithm 1: Matching

Input: Request $R=(DR, IR, OR, PR, ER);$ /* The request */
Ontology SO(D, SD, AS) where
- AS (DA, IA, OA, PA, EA, CA) /* Abstract Service */
- SD (N, T) /* Sub-domain */
- D (N, T) /* Domain */

Output: Key C;

Begin
F=Ø; C=Ø;
Browse SO For each SD do
  If(Compare(1(D, N) >= Threshold)) then F=F+SD; end_if;
End_for;
For each AS $\in F$ do
  If(Compare2(AS, R) >= Threshold) then C=C+CA; end_if;
End_for;
End.

Compare1() calculates $(DR \text{ MES} N)$, which is the similarity value between the domain name DR of the request, and the domain names N recorded in the SO. Compare2() estimates the similarity between the request R and the abstract services AS of the selected domains in F. The similarity is calculated via:

$$\left(\sum \text{Max}\left(\{\text{IR}\}_{Y} \text{ MES} \{\text{IA}\}_{J}\right) + \sum \text{Max}\left(\{\text{OR}\}_{Y'} \text{ MES} \{\text{OA}\}_{J'}\right) \right) \div 2$$

Where $\{\text{IR}\}_{Y}$ are the Y Inputs of the Request R, $\{\text{IA}\}_{J}$ are the J Inputs of the Abstract service AS, $\{\text{OR}\}_{Y'}$ are the Y’ Outputs of the Request R, and $\{\text{OA}\}_{J'}$ are the J’ Outputs of the Abstract service AS. The similarity measures MES, used and evaluated in Section 5, aim to quantify how much two entities are semantically similar. In this context, we used similarity measures based on WordNet [15]. The Threshold is a value between 0 and 1. The value 1 indicates that there is a complete semantic equivalence between the two entities.

4.3. Storage/Discovery in the UDDI

The result of the matching algorithm (the key <D, C>) will be used to store or discover a Web service according to the client's request. This will ensure a number of advantages, including availability, reducing the loads of storage and routing, no degradation of the discovery quality with the increasing number of the stored Web services, etc. The main role of a registry is the storage of a large number of published services. This task may not be easy without the existence of an organizational representation to navigate this large number of services, hence the need to use the ontology SO. The proposed UDDI model adapts the structure of the ontology SO. Each UDDI registry includes a finite number of nodes. Each node corresponds to a specific (sub) domain. This structure can be used both for the storage and the discovery of services.
5. Evaluation

The evaluation section aims to choose the most efficient similarity to be used in the matching algorithm (Compare1() and Compare2()). For this, we studied four WordNet similarity measures: PATH, PIRRO_SECO, RESNIK_METRIC, JIANG_METRIC and LIN [10]. To perform this test, we first obtained a corpus of 1090 Semantic Web Services (OWL-S)\(^2\). Our initial intention was to test our algorithm on the whole corpus. Since there is no pre-classification in the corpus, we limited our experiment on only one subset (geographical domain). Then, we have manually classified a set of OWL-S semantic Web services (see Table 2 and Table 3).

<table>
<thead>
<tr>
<th>Semantic Web Service</th>
<th>Domain</th>
<th>Class in SO «Domain ; keys &gt;</th>
</tr>
</thead>
<tbody>
<tr>
<td>calculateDistanceInMiles.owls</td>
<td>Geography</td>
<td>«Geography ;1&gt;</td>
</tr>
<tr>
<td>getDistanceBetweenPlaces.owls</td>
<td>Geography</td>
<td>« Geography ;1&gt;</td>
</tr>
<tr>
<td>getLocationOfAddress.owls</td>
<td>Geography</td>
<td>« Geography ;11 ;12 ;6&gt;</td>
</tr>
<tr>
<td>getLocationOfAddressYahooMaps.owls</td>
<td>Geography</td>
<td>« Geography ;11,9&gt;</td>
</tr>
<tr>
<td>getPlaceOfAddress.owls</td>
<td>Geography</td>
<td>« Geography ;12&gt;</td>
</tr>
<tr>
<td>getDistanceBetweenLocations.owls</td>
<td>Geography</td>
<td>« Geography ;1&gt;</td>
</tr>
<tr>
<td>renderMapService.owls</td>
<td>Geography</td>
<td>« Geography ;8&gt;</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Domain</th>
<th>Class number « Key »</th>
<th>Name of SA</th>
</tr>
</thead>
<tbody>
<tr>
<td>Geography</td>
<td>&lt;1&gt;</td>
<td>addressDistanceCalculator</td>
</tr>
<tr>
<td>Geography</td>
<td>&lt;6&gt;</td>
<td>getCoordinatesOfAddress</td>
</tr>
<tr>
<td>Geography</td>
<td>&lt;8&gt;</td>
<td>getDrivingDirections</td>
</tr>
<tr>
<td>Geography</td>
<td>&lt;9&gt;</td>
<td>getLocationOfAddress</td>
</tr>
<tr>
<td>Geography</td>
<td>&lt;11&gt;</td>
<td>getMapOfAddress</td>
</tr>
<tr>
<td>Geography</td>
<td>&lt;12&gt;</td>
<td>getPlaceOfAddress</td>
</tr>
</tbody>
</table>

To evaluate our proposal, we implemented a prototype tool that supports our algorithms. Then we performed our tests on pre-classified Web services listed in Table 2. The Web services discovery issue is similar to that of information retrieval [2]. Thus, we used two metrics used in the last one, including Precision and Recall\(^3\), to evaluate the results of our algorithm. Now, we define the following assistant parameters:

- **CN**: number of correct Web services (OWL-S documents) that should be assigned to the considered domain;
- **EN**: number of Web services assigned by the tool;
- **CEN**: number of the correct Web services assigned by the tool.

Then:

- **Recall (R)**: proportion of the correctly assigned Web services of all the Web services that should be assigned. It can be presented as “\(R = \frac{CEN}{CN}\)”.
- **Precision (P)**: proportion of the correctly assigned Web services of all the Web services that have been assigned. It can be presented as “\(P = \frac{CEN}{EN}\)”.

\(^2\) Corpus Link
\(^3\) [http://en.wikipedia.org/wiki/Precision_and_recall](http://en.wikipedia.org/wiki/Precision_and_recall)
Usually, Precision and Recall scores are combined into a single measure, called F-measure defined as follows: 
\[ F\text{-measure} = \frac{2 \times \text{recall} \times \text{precision}}{\text{recall} + \text{precision}}. \]

The graphs above shows the Precision, Recall and F-measure statistics obtained by applying our matching algorithm on the set of pre-classified Web services (see Table 2), for different threshold values. We expected that the Precision for the PATH measure reduces proportionally with the threshold. However, this was not the case in the graph. This can be explained by the fact that the two semantic Web services `getDistanceBetweenPlaces.owls` and `renderMapService.owls` were recognized with a threshold less than 0.5. Therefore, the PATH measure expresses the bad results for our algorithm, since it sets minimum values of Precision, Recall and F-measure. For RESNIK_METRIC measure, classes have appeared with a threshold less than 0.9. The other similarity measures have expressed almost identical Recall values. So it is the Precision value which will make difference between these methods. Finally, we can say that to have best results, we must choose one of the two similarity measures, PIRRO_SECO_METRIC or Lin, with a threshold of 0.8. We may also use the RESNIK_METRIC measure with a threshold of 0.7.

6. Conclusion and perspectives

The UDDI is a directory for storing and discovering Web services. Most of the current UDDI models are centralized. Centralized structure is less robust and expresses poor interoperability. In this paper, we presented a storage/discovery architecture designed to be distributed, scalable, and address the centralization issue of an UDDI directory. Our work is based on the classification of Web services according to an ontology of services, also used as a shared model for interoperability between registries. Our ultimate goal is to improve the Web service Storage and Discovery, based on a classification process, using SO service ontology. This organizational classification will
reduce the workspace. As a result, we will improve the discovery in terms of response time, and improve the storage in terms of space, in order to overcome the UDDI centralization issue. To implement our approach we developed a prototype tool. In its current version, the tool allows the storage and discovery of a Web service in an UDDI using a shared ontology. Our work is also extended to deal with other problems, including the Web services composition. Finally, a comparative study should be done to show the effectiveness of our approach compared to existing works. Our future work is also to complete the prototype tool, in particularly the user interface. We will also propose a solution for automatically building the shared repository used in our approach. To do this, we will surely introduce ontology learning techniques.

References