Journal of Computer Science 10 (8): 1418-1422, 2014
ISSN: 1549-3636

doi:10.3844/jcssp.2014.1418.1422 Published Online 10 (8) 2014 (http://www.thescipub.com/jcs.toc)

SEMANTIC ENHANCED UDDI USING OWL-S PROFILE ONTOLOGY FOR THE AUTOMATIC DISCOVERY OF WEB SERVICES IN THE DOMAIN OF TELECOMMUNICATION

Lakshmana Kumar, R. and M.S. Irfan Ahmed

Department of Computer Applications, Hindusthan College of Engineering and Technology, Coimbatore, India

Received 2013-07-31; Revised 2014-02-04; Accepted 2014-03-20

ABSTRACT

The current web services which are evolved in the telecom domain such as payment web services, Yellow pages web services, operator web services, weather web services are failed to bring down the semantic as they used to prove its syntactic description. The reason for bringing down the semantic description into already existing web services will invoke certain operations like automatic discovery of web services, automatic composition of the necessary services, automatic invocation of web services and automatic monitoring of the execution process. At present the web services in the domain of telecommunication is following the parlay X standard. The parlay X has given a set of standard web service API's for the telecom. The each of the services will have its own interface, services and types In this study in order to bring down the semantic representation we have proposed an idea to enable the semantic through the upper ontology like OWL-S and then how to map OWL-S to UDDI registry and also we have discussed some of the issues that we have faced while mapping OWL-S into UDDI registry. So the approach which we are going to propose improves the accuracy of the telecommunication network services description, discovery and matching, unifies the semantic representation of telecommunications network and Internet services.

Keywords: ParlayX, OWL-S, UDDI, Upper Ontology

1. INTRODUCTION

Web services have become the main resources for managing the universal Connectivity and interoperability of heterogeneous applications and services. Their growing popularity and usage, however, triggers the problem of finding and composing relevant services since thousands of such services already exist within the public domain. In order to be able to discover and compose suitable services, one must search through the set of abstract descriptions that are made available, since web services provide functionality without showing any organization-specific implementation details . The lack of any machine interpretable semantics requires human intervention for the discovery, composition, invocation and monitoring of services, which prevents the use of services in complex business contexts, where the automation of these processes is necessary. Enabling Semantic web services relax this restriction by annotating services with semantic descriptions provided by ontologies (Paolucci *et al*., 2002). In this study we have discussed how to use the upper ontology like OWL-S into the existing web services. Next we have proposed our own approach for mapping from OWL-S to UDDI. The mapping is done with OWL-S profile Information into the UDDI registry. While mapping we have made discussions on UDDI tModels. And finally we have also discussed some of the issues of mapping OWL-S Profile Information into UDDI Registry.

2. IN NEED OF OWL-S

2.1. OWL

This OWL-S is an upper ontology for services. OWL-S defines classes and properties that can be used to describe the main things in the service such as:

Corresponding Author: Lakshmana Kumar, R., Department of Computer Applications, Hindusthan College of Engineering and Technology, Coimbatore, India

- What does the service do?
- How does the service work?
- How is the service Invoked?

In order to answer these questions the Upper ontology helps us a lot such as OWL-S upper ontology contains a sub-ontology called as.

Profile ontology (Profile.owl) to define the classes and properties. This sub ontology is mainly used to advertise the service, thereby enabling a service requester to determine whether the given service meets the needs or not OWL-S upper ontology's second sub ontology, process ontology (process.owl) defines all the terms that we need to describe how the service works. Moreover, we describe how the service works by describing the procedures necessary to interact with the service from a client's point of view.

OWL-S upper ontology's third sub ontology named as Grounding ontology (grounding.owl) is included by OWL-S to provide terms that one can use to describe how the service can be accessed technically. This in-cludes the terms to describe the supported protocol and the exchanged message formats and other related low-level information. The sun ontologies are the main com-ponents of OWL-S. In addition to these ontologies there is high level ontology known as service ontology (ser-vice.owl) which helps to club these three components work together to describe a web service (Duke *et al*., 2005).

3. HOW DOES OWL-S MEET EXPECTATIONS?

The purpose of describing the profile of a given service is to provide enough information so that a soft agent can decide whether the given service meets the requester's need (Griffin and Pesch, 2007). The OWL-S upper ontology's profile.owl doc-ument provides terms that can be used to accomplish this goal. With the help of profile.owl we can achieve things like:

- Service information and contact information(name of the service, descriptions and the organization that provides the service)
- Functional description: In-put/output/Precondition/effect(IOPE) of the service
- Non-Functional information of the service(including information such as Quality of service)

The profile document has been written for our service known as:

- **Figure 1** Parlayx_sms_notification_interface
- **Figure 2** Profile document has been created for this service

In the above process while bringing down the Profile document IOPE terms has not been mentioned and the IOPE Terms has been defined in the **Fig. 3** process.owl.

Clearly from the profile ontology it is possible for us to add semantics to our document. For our simple SmsNoti-fication service, if we were to use the WSDL document to describe this service, we would only know that this service takes an xsd: String as input and returns a xsd: Double as ouput and clearly there are no semantics at all, there might be hundred of web services out there that take exactly the same input and return exactly the same output. Therefore, there will be no way to select this par-ticular service correctly and automatically from all the candidates having a similar look and feel (Bond *et al*., 2009).

But now we have used the OWL-S profile sub ontology to describe this service. These are exactly the semantics we are looking for. To make matters clearer, the follow-ing discussion shows how these semantics will enable us to automatically find this service OSA, 2008. If we want to find a service that takes a SMSNotification model as an input and returns a string, we have followed these steps:

Step 1: Created a service request file to express what we are looking for by using the concepts from the right on-tology

Step 2: Submit the request file to some smart agent that is capable of reading the OWL-S profile document of each candidate service and using a match making engine to conduct semantics matching

Step 3: The agent will compare the request document against each profile document it has located and included the matched service as a potential candidate

4. MAPPING OWL-S PROFILE INFORMATION INTO THE UDDI REGISTRY

The mapping form OWL-S profile document to the UD-DI registry is done on one-to-one basics, which we have given in the **Table 1** mapping profile into registry. Here the left column contains the elements from OWL-S and the right column contains the elements from UDDI data structures. Any row shows the mapping relation-ship. If for a given row the mapping does not exist, the corresponding cell in the lest or the right column is left blank. ALsi, if an OWL-S profile element has a corres-ponding UDDI element, the mapping is a direct connec-tion between two elements. For OWL-S profile element with no corresponding UDDI elements, tModel-based mapping is used. The basic

idea is to create specialized UDDI tModels for each unmapped element in the OWL-S profile, such as OWL-S input, Output and so on. These tModels have to be first created and registered with the UDDI before the mapping can use them. These specia-lized tModels are in fact used as namespaces in the map-ping process (Srinivasan *et al*., 2004). **Figure 4** UDDI's input_tModel Shows the description of UDDI's input_tModel, which is created and registered to represent the input element form, the OWL-S profile document.

Table 1. Mapping profile into registry

OWL S profile elements	UDDI elements
Name	ContactInformation:Name
BusinessEntity:Name	BusinessEntity:Contact:Person name
ContactInformation:Phone	BusinessEntity:Contact:Phone
ContactInformation:Email	BusinessEntity:Contact:Email
ContactInformation:PhysicialAddress	BusinessEntity:Contact:Address
ContactInformation:WebURL	BusinessEntity:Discovery URL's
Service name	BusinessService:Name
Text description	BusinessService :Description
Has process	BusinessService:CategoryBag:Hasprocess_tModel
Service category	BusinessService:CategoryBag:Service Category _tModel
Serviceparamenter	BusinessService:CategoryBag:Serviceparameter_tModel
Quality rating	BusinessService:CategoryBag:QulaityRating_tModel
Input	BusinessService:CategoryBag:Input_tModel
Output	BusinessService:CategoryBag:Output_tModel
Precondition	BusinessService:CategoryBag:Precondition_tModel
Effects	BusinessService:CategoryBag:Effect_tModel
serviceproduct	BusinessService:CategoryBag:ServiceProduct_tModel
Service classification	BusinessService:CategoryBag:Classification_tModel

```
<? Xml version="1.0" encoding="UTF-8"?>
<!—July 28, 2012-->
<wsdl: definitions
name="parlayx_sms_notification_interface"
targetNamespace="http://www.csapi.org/wsdl/parlayx/sms/notification/v1_0/interface"
xmlns="http://schemas.xmlsoap.org/wsdl/"
xmlns:wsdl="http://schemas.xmlsoap.org/wsdl"xmlns:xsd=http://www.w3.org/2001/XMLSchema
xmlns:parlayx_sms_notification="http://www.csapi.org/wsdl/parlayx/sms/notification/v1_0/interface"
xmlns:parlayx_sms_faults="http://www.csapi.org/wsdl/parlayx/sms/v1_0/faults"
xmlns:parlayx_common_faults="http://www.csapi.org/wsdl/parlayx/common/v1_0/faults"
xmlns:parlayx_sms_xsd="http://www.csapi.org/schema/parlayx/sms/v1_0"
xmlns:parlayx_common_xsd="http://www.csapi.org/schema/parlayx/common/v1_0">
<wsdl: import namespace="http://www.csapi.org/wsdl/parlayx/sms/v1_0/faults"
location="parlayx_sms_faults.wsdl"/>
<wsdl: import namespace="http://www.csapi.org/wsdl/parlayx/common/v1_0/faults"
location="parlayx_common_faults.wsdl"/>
<Wsdl: types>
<xsd: schema elementFormDefault="qualified">
<xsd: import namespace="http://www.csapi.org/schema/parlayx/sms/v1_0"
schemaLocation="parlayx_sms_types.xsd"/>
<xsd: import namespace="http://www.csapi.org/schema/parlayx/common/v1_0"
schemaLocation="parlayx_common_types.xsd"/>
</xsd: schema>
</wsdl: types>
<wsdl: message name="SmsNotification_notifySmsReceptionRequest">
<wsdl: part name="registrationIdentifier" type="xsd: string"/>
<wsdl: part name="smsService.ActivationNumber" type="xsd: string"/>
<wsdl: part name="senderAddress" type="parlayx_common_xsd: EndUserIdentifier"/>
<wsdl: part name="message" type="xsd: string"/>
</wsdl: message>
<wsdl:messagename="SmsNotification_notifySmsReceptionResponse"/>
<wsdl: portType name="SmsNotification">
<wsdl: operation name="notifySmsReception">
<Wsdl: input m essage="parlayx_sms_notification:SmsNotification_notifySmsReceptionRequest"/>
<wsdl: output message="parlayx_sms_notification:SmsNotification_notifySmsReceptionResponse"/>
<wsdl: fault name="Application Exception" message="parlayx_common_faults: ApplicationException"/>
</wsdl: operation>
</wsdl: portType>
</wsdl: definitions>
```

Fig. 1. Parlayx_sms_notification_interface

```
<?xml version='1.0' encoding='ISO-8859-1'?>
<?DOCTYPE uridef [
<!ENTITY rdf "http://www.w3.org/1999/02/22-rdf-syntax-ns#"><!ENTITY rdfs
"http://www.w3.org/2000/01/-rdf-schema">
<!ENTITY owl "http://www.w3.org/2002/07/owl#">
<!ENTITY service "http://www.daml.org/services/owl-s/1.1/Service.owl">
<!ENTITY profile "http://www.daml.org/services/owl-s/1.1/Profile.owl">
<!ENTITY process "http://www.daml.org/services/owl-s/1.1/Process.owl">
<!ENTITY actor "http://www.daml.org/services/owl-s/1.1/ActorDefault.owl">
<!ENTITY xsd "http://www.w3.org/2001/XMLS chema">
<!ENTITY DEFAULT "http://localhost/GetSMSNotification.owl#">
]>
<rdf:RDF
xmlns: rdf = "&rdf;#"
xmlns: rdfs = "&rdfs;#"
xmlns: owl = "&owl;#"
xmlns: actor = "&actor;#"
xmlns: profile = "&profile;#"
xmlns: xsd = "&xsd ;#"
xmlns: = "&DEFAULT;#"
xmlns: base = "&DEFAULT;#"
>
<owl:Ontology about="">
<owl:imports rdf:resource= "& service;"/>
<owl:imports rdf:resource= "& profile;"/>
<owl:imports rdf:resource= "& process;"/>
<owl:imports rdf:resource= "& actor;"/>
</owl:Ontology>
<profile:profile>
<profile:serviceName>getS MSNotification
</profile:serviceNa me>
<profile:textDescription>
Text describtion of the service for human read
</profile:textDescription>
?<profile:contactInformation>
<actor:Actor rdf: ID="getSMSNotification_provider">
<actor:name>name of the service representative
</actor:name>
<actor:title>person's title</actor:title>
<actor:phone>(area)+phonenumber</actor:phone>
</actor:Actor>
</profile:contactInfor mation>
```

Fig. 2. Profile document

```
<owl:ObjectProperty rdf:ID="hasParameter">
<rdfs:domain rdf:resource="#Profile"/>
<rdfs:range rdf:resource="&process; #Parameter"/>
</owl:ObjectProperty>

<owl:ObjectProperty rdf:ID="hasInput">
<rdfs:subPropertyOf rdf : resource="#hasParameter"/>
<rdfs:range rdf:resource="&process:#Input"/>
</owl:ObjectProperty>

<owl:ObjectProperty rdf:ID="hasOutput">
<rdfs:subPropertyOf rdf : resource="#hasParameter"/>
<rdfs:range rdf:resource="&process:# Output"/>
</owl:ObjectProperty>

<owl:ObjectProperty rdf:ID="hasPrecondition">
<rdfs:domain rdf : resource="#profile"/>
<rdfs:range rdf:resource="&expr;:#Condition"/>
</owl:ObjectProperty>

<owl:ObjectProperty rdf:ID="hasResult">
<rdfs:domain rdf : resource="#profile"/>
<rdfs:range rdf:resource="&expr;:#Result"/>
</owl:ObjectProperty>
```

Fig. 3. Process.owl

```
Name  : input_tModel
Key   : uuid_of_input_tModel
Technical Model Descriptions : This preregistered tModel represents the input element in an OWL-S
profile document
Overview URL : some other document describing this tModel
```

Fig. 4. UDDI's input_tModel

Using our getSMSNotification service as example, its UDDI entity can be updated as shown in the **Fig. 2** Profile document. We have created 15 tModels to accomplish this. However, the final result is a semantically enhanced UDDI.

5. ISSUES OF MAPPING OWL-S PROFILE INFORMATION INTO UDDI REGISTRY

The first issue is, how to inform a soft agent that a given service advertisement has an OWL-S profile representation. UDDI is essentially a huge database holding a vast amount of service advertisements. If a given UDDI registry is semantically enhance, some of these advertisements must have used the predefined tModels (Martin *et al*., 2007; Aguilera *et al*., 2007; Srinivasan *et al*., 2004; MohanRam and Balasubramanian, 2013; Colgrave and Januszewski, 2004). For each service advertisement, the agent has to first see whether it is semantically marked-up advertisement, if not, the agent will simply skip it. For that issue, we have created a single flag for the agent to read. We have created and registered another tModel called as OWL-S_tModel, which has a special meaning, it states that the service using this tModel has been semantically marked up and furthermore, its value cane be the URL of the OWL-S profile document. Therefore, not only have the elements in the OWL-S profile document been mapped into some UDDI service entity, but the service entity also has a link pointing back to its original OWL-S document.

Another issue which we have faced is related to the specialized tModels. UDDI itself provided several tModels, one such example is the getSMSNotification tModel used for the categorization. UDDI has made these tModels well known to the users, so anyone who wants to add categorization information to his service advertisement can use these predefined tModels freely. Similarly all the tModels representing the OWL-S elements should also be made well known to the users. Currently however it would be unclear how this could be accomplished. Therefore, different users may register their own input_tModel with UDDI. We have addressed this problem by the solution is to always look for specialized tModels before we create your own. For instance, when we are implementing the mapping from OWL-S to UDDI structure, you should first search for the input_tModel, the output_tModel, or any other specialized tModel.

6. CONCLUSION

In this study we have taken a syntactic web service called as parlayx_sms_notification_interface from the parlay X model which acts as a framework for the telecommunication domain. We have added semantics to the service using an upper ontology OWL-S. After adding the semantics we have mapped OWL-S with UDDI registry and finally we have also discussed some of the issues we have faced while mapping OWL-S into UDDI registry. In this broad research, we have given an idea that how a semantic web service can be discovered automatically. Particularly in the domain of telecommunications it provides higher efficiency and performance.

7. REFERENCES

Aguilera, U., J. Abaitua, J. Díaz, D. Buján and D.L. De Ipiña, 2007. A semantic matching algorithm for discovery in UDDI. Proceedings of the International Conference on Semantic Computing, Sep. 17-19, IEEE Xplore Press, Irvine, CA., pp: 751-758. DOI: 10.1109/ICSC.2007.43

Bond, G., E. Cheung, I. Fikouras and R. Levenshteyn, 2009. Unified telecom and web services composition: Problem definition and future directions. Proceedings of the 3rd International Conference on Principles, Systems and Applications of IP Telecommunications, Jul. 07-08, ACM New York. DOI: 10.1145/1595637.1595654

Colgrave, J. and K. Januszewski, 2004. Using WSDL in a UDDI registry. OASIS.

Duke, A., M. Richardson, S. Watkins and M. Roberts, 2005. Towards B2B integration in telecommunications with semantic web services. Proceedings of the 2nd European Conference on The Semantic Web: Research and Applications, May 29-Jun. 1, Springer Berlin Heidelberg, Crete, Greece, pp: 710-724. DOI: 10.1007/11431053_48

Griffin, D. and D. Pesch, 2007. A survey on web services in telecommunications. IEEE Commun. Magaz., 45L: 28-35. DOI: 10.1109/MCOM.2007.382657

Martin, D., M. Burstein, D. McDermott, S. McIlraith and M. Paolucci et al., 2007. Bringing semantics to web services with OWL-S. World Wide Web, 10: 243-277. DOI: 10.1007/s11280-007-0033-x

MohanRam, B.R. and S. Balasubramanian, 2013. Automating business processes of telecom service providers using BPM and web services for NGOSS. Infosys Technologies Limited.

Paolucci, M., T. Kawamura,T.R. Payne1 and K. Sycara, 2002. Importing the Semantic Web in UDDI. In: Verlag Ber-lin Heidelberg, Bussler, C., (Ed.), WES LNCS, pp: 225-236.

Srinivasan, N., M. Paolucci and K. Sycara, 2004. An efficient algorithm for OWL-S based semantic search in UDDI. Proceedings of the 1st International Conference on Semantic Web Services and Web Process Composition, Jul. 6-6, Springer Berlin Heidelberg, San Diego, CA, USA, pp: 96-110. DOI: 10.1007/978-3-540-30581-1_9