Boosting Annotated Web Services in SAWSDL

Antonio J. Roa-Valverde, Jorge Martinez-Gil, and José F. Aldana-Montes

University of Málaga, Department of Computer Languages and Computing Sciences
Boulevard Louis Pasteur 35, 29071 Málaga, Spain
{roa,jorgemar,jfam}@lcc.uma.es
http://www.lcc.uma.es

Abstract. The W3C Recommendation for Semantic Annotations in WSDL and XML Schema (SAWSDL) defines an extension that can help to disambiguate the description of Web Services during automatic discovery and composition. In this way, SAWSDL is useful to facilitate the grounding stage when annotating Web Services. Despite SAWSDL does not specify a language to represent the semantic models for annotations, most of the times, ontologies are used to do it. In this work we propose a mechanism to automatically enrich SAWSDL annotations using concepts from different ontologies. As result, we provide a method for helping experts to annotate web services according to the SAWSDL recommendation.

Keywords: Automatic Grounding, Semantic Web Services, SAWSDL.

1 Introduction

Semantic Web Services (SWS) are getting more popular among IT companies and the research community as it can be noticed by the amount of ongoing projects related with this topic. The SWS approach is not a new technology, it is a fusion between the traditional SOA and Semantic Web technologies [1]. This initiative arose with the aim of solving the drawbacks of dealing with a big amount of services. In this way, SWS have demonstrated that applying Semantic Web technologies can be useful to automatize common tasks in the Web services life-cycle.

According to the grounding stage (i.e. the stage where the semantic extensions take contact with the underlying SOA technologies) efforts carried out resulted in the W3C Recommendation for Semantic Annotations in WSDL and XML Schema (SAWSDL) [2]. SAWSDL defines mechanisms using semantic annotations which can be added to WSDL resources. These annotations can help to disambiguate the description of Web services during automatic discovery and composition. SAWSDL does not specify a language to represent the semantic models for annotations, but ontologies are often used to annotate Web Services.

To achieve automatic Web Services annotation (commonly named automatic grounding) is a key challenge for researchers. The automatic grounding is strongly dependent of the underlaying semantic model. Still using a universally recognized model the process of annotating automatically a Web Service is conditioned to the WSDL descriptions. This problem would disappear if developers use the same identifiers when they make reference to the concepts of such semantic model. Despite this idea would facilitate semantic annotation tasks, to demand the use of a common set of descriptors for developing Web services is not doable. In this sense, the presence of an expert is required to curate the semantic annotations. At this point a question rise up, why are automatic grounding methods used for? To answer this question, the automatic grounding should be thought like a tool for helping experts and not for excluding them of this task. For example, there are domains such life sciences where the use of ontologies to deal with the enormous amount of information available is becoming more popular. In these domains, the information is many often processed via Web services [3]. This kind of contexts are characterized by a dynamic environment where new data and concepts arise in short periods of time. This issue is the responsible for the existence of different ontologies to describe the same domain and even for the changes produced on ontologies in use. The existence of an automatic tool suggesting new annotations to the expert would be so useful in this scenario.

In this work we propose a methodology to automatically enrich SAWSDL annotations using related ontologies. We use the notion of transitivity and automatic matching to adapt annotated WSDL files towards the new concepts. The rest of this work is structured as follows. Section 2 describes the state-of-the-art in relation to SAWSDL and Ontology Matching. Section 3 presents the technical preliminaries that compose the basis of our approach. Section 4 discusses our methodology to solve the problem. Section 5 describes an use case where the proposed methodology is applied. Finally, we remark the strengths of our proposal and discuss the future work in Section 6.

2 Related Work

This section introduces the related work to the SAWSDL Recommendation and advanced ontology matching techniques.

The Semantic Annotations for WSDL and XML Schema (SAWSDL) W3C Recommendation defines mechanisms using which semantic annotations can be added to WSDL components. SAWSDL does not specify a language for representing the semantic models, e.g. ontologies. Instead, it provides mechanisms by which concepts from the semantic models that are defined either within or outside the WSDL document can be referenced from within WSDL components as annotations. These semantics when expressed in formal languages can help to disambiguate the description of Web Services during automatic discovery and composition. Figure 1 illustrate the extensions provided by SAWSDL.

Based on member submission WSDL-S[1], the key design principles for SAWSDL are:

[1] http://www.w3.org/Submission/WSDL-S/

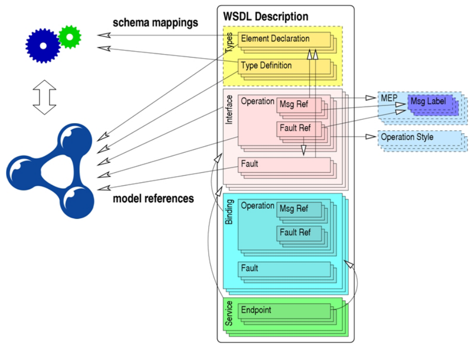

Fig. 1. Architecture of SAWSDL proposed by the W3C (Extracted from http://www.w3.org)

- The specification enables semantic annotations for Web Services using and building on the existing extensibility of WSDL.
- It is agnostic to semantic representation languages.
- It enables semantic annotations for Web Services not only for discovering Web Services but also for invoking them.

Based on these design principles, SAWSDL defines the following three new attributes for enabling semantic annotation of WSDL components:

- an extension attribute, named *modelReference*, to specify the association between a WSDL component and a concept in some semantic model. This attribute can be used especially to annotate XML Schema type definitions, elements and attributes declarations as well as WSDL interfaces, operations and faults.
- two extension attributes, *liftingSchemaMapping* and *loweringSchemaMapping*, that are added to XML Schema element declarations and type definitions for specifying mappings between semantic data and XML. These mappings can be used during service invocation to solve problems related to the data format.

On the other hand, related to Ontology Matching [5] there are a lot of techniques and tools for addressing it [6][7][8]. However, the complexity of the problem we are dealing with causes existing solutions are not fully satisfactory. Most of these strategies have proved their effectiveness when they are used with some kind of synthetic benchmarks like the one offered by the Ontology Alignment Evaluation Initiative (OAEI) [9]. However, when they process real ontologies they behave worse [10]. Nowadays, current trends to solve this problem consists of combining basic techniques.

The most outstanding basic techniques for Ontology Matching are: String normalization, String similarity, Data Type Comparison, Linguistic methods, Inheritance analysis, Data analysis, Graph-Mapping, Statistical analysis and Taxonomy analysis. A detailed explanation for each of these techniques is presented in [11].

For the matching tasks, we use an architecture for conceptual mediation, which allow users to infer information from the ontology-explicit knowledge, enabling them to discover related ontologies by means of Semantic Fields [12]. In this way, we can get a high degree of success when obtaining ontologies similar to an initial ontology. Semantic Fields is a generic infrastructure, so we propose to integrate in this infrastructure the matching functions obtained by using GOAL [11]. These functions have been optimized using the official benchmark from the Ontology Alignment Evaluation Initiative (OAEI) [8].

3 Technical Preliminaries

This section introduces the technical preliminaries needed to understand our approach.

Definition 1 (Similarity measure). *A similarity measure sim is a function* $sim : \mu_1 \times \mu_2 \mapsto \Re$ *that associates the similarity of two input ontology entities* μ_1 *and* μ_2 *to a similarity score* $sc \in \Re$ *in the range [0, 1].*

A similarity score of 0 stands for complete inequality and 1 for complete equality of the input ontology entities μ_1 and μ_2.

Definition 2 (Ontology Matching function). *An ontology matching om is a function* $om : O_1 \times O_2 \overset{sim}{\rightarrow} A$ *that associates two input ontologies* O_1 *and* O_2 *to an alignment A using a similarity measure.*

Property 1 (Transitivity in Ontology Matching functions). *Let* $c1 \in O_1, c2 \in O_2$ *and,* $c3 \in O_3$ *then*

$$sim(c1, c2) = \mu \wedge sim(c2, c3) = \lambda \rightarrow sim(c1, c3) = \mu \cdot \lambda$$

Example 1. *Calculate the similarity between plane and airplane knowing that the similarity between plane and aeroplane is 90 percent and between aeroplane and airplane is 90 percent too.*

$$\frac{sim(plane,aeroplane)=0.9 \wedge sim(aeroplane,airplane)=0.9}{sim(plane,airplane)=0.81}.$$

4 Methodology

In this section, we propose a methodology to annotate WSDL components automatically according to external ontologies. The methodology that we propose can be divided in five main steps as we show in Figure 2.

1. *Identify the pair WSDL-Ontology we want to extend.* This step involves obtaining each ontology referenced by the set of *modelRerence* attributes. Although the SAWSDL Recommendation says nothing about the semantic model to annotate WSDL our method has been designed to work with ontologies only.

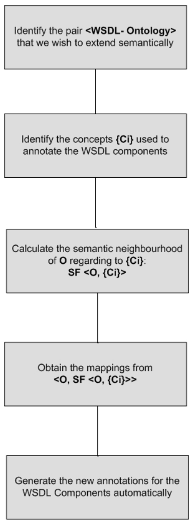
Identify the pair **<WSDL- Ontology>** that we wish to extend semantically
Identify the concepts **{Ci}** used to annotate the WSDL components
Calculate the semantic neighbourhood of **O** regarding to **{Ci}**: **SF <O, {Ci}>**
Obtain the mappings from **<O, SF <O, {Ci}>>**
Generate the new annotations for the WSDL Components automatically

Fig. 2. Methodology that we propose for adapting annotated web services in SAWSDL

2. *Identify the concepts used for annotating WSDL concepts.* In this step, it is necessary to identify the concepts which are used for annotating the components in the WSDL description. These concepts will be the input to look for related ontologies as we will see in the next step.
3. *Use Semantic Neighbourhoods to identify the new ontologies that will be used to annotate the WSDL components.* This is not a trivial task. If we choose an arbitrary ontology probably we may find too few correspondences with the original ontology. For this reason, we use the method described in [12] in order to locate related ontologies.
4. *Obtain the Semantic Correspondences from the previous step to apply transitivity.* This step involves to obtain the Semantic Correspondences (usually called mappings) between the existing concepts in the related ontologies. We propose using the algorithms described in [11] to do that. Moreover, these mappings could be pointing to different concepts from the same ontology. For example, a WSDL element can be associated with two concepts. SAWSDL Recommendation does not specify any relationship between these multiple annotations other than saying that they all apply. It is up to the consumers of these annotated WSDLs to use the ones that are relevant to them or to figure out the relationship between the concepts, if they so choose, by consulting the ontology that defines them.
5. *Automatic generation of links between the initial WSDL and new ontologies.* The different matchings obtained in the previous steps are used to add the new concepts discovered to the model reference set. Note that SAWSDL defines the *modelReference* attribute as a list of URIs, so there is not limitation in the number of references used.

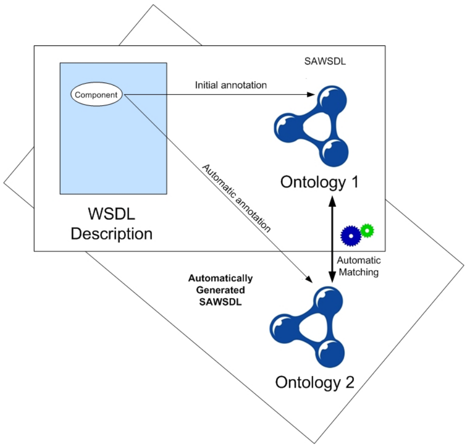

Fig. 3. Illustration of the automatic grounding in SAWSDL

In the Figure 3, we show how an initial SAWSDL annotation can generate new SAWSDL annotations automatically using the notion of automatic matching and transitivity.

5 Use Case

In order to test our method we have perform it using the same example described in [2]. This example consists of a purchase order Web Service interface described in WSDL and annotated semantically using an OWL ontology called *PurchaseOrder*. We have matched this ontology with other called *Order* by using matching functions obtained using GOAL [11]. The obtained results are showed in Table 1.

These results are used to annotate the WSDL components according to the *Order* ontology in an automatic way. For example, the simpleType named

Table 1. Experimental results obtained

PurchaseOrder.owl	**Order.owl**	**Confidence**
UPCCode	:PCode	0.70
Customer	:Customer	0.88
Unit	:Unit	0.79
Product	:Products	0.76
OrderRequest	:RequestOrder	0.55
CustomerID	:Customer	0.69
LineItem	:Item	0.50
OrderConfirmation	:Confirmation	0.70
Quantity	:Quantity	0.88

Confirmation points to the concept *OrderConfirmation* as it can be seen below.

```
<xs:simpleType name="confirmation" sawsdl:modelReference=
"http://www.w3.org/sawsdl/purchaseorder#OrderConfirmation">
```

According to our methodology, this WSDL component could point to the concept *Confirmation* which belongs to the second ontology. The reason is that the discovered correspondence (*OrderConfirmation*, *Confirmation*) has a high degree of confidence. Note that this semantic correspondence has been discovered automatically. The new concept is added like a new reference to the list.

```
<xs:simpleType name="confirmation" sawsdl:modelReference=
"http://www.w3.org/sawsdl/purchaseorder#OrderConfirmation",
"http://khaos.uma.es/order#Confirmation">
```

The process described above is the same for each *modelReference* attribute existing in the WSDL file. If a service discovery task is launched using a richer set of semantic annotations, the probability of discover the service is higher. In order to get the most suitable service, the list of references should be the most exact possible. This means that matchings with a low similarity measure must be refused. For this reason, our method uses a threshold value to decide when a reference should be added or discarded.

6 Conclusions and Future Work

In this work, we have presented a novel proposal for automatically enrich annotated Web Services in SAWSDL. The proposed methodology is based on two key assumptions: a) The semantic model chosen to annotate Web Services components is implemented as an OWL ontology and b) The fitness of this method relies on the semantic correspondences found by the matching function.

As result our proposal helps experts with the tedious task of annotating Web Services according to different ontologies. This first approach does not take into account the schema mappings used during the invocation phase. In this way, the developed method tries to boost the annotated Web Service adding new annotations to the set of *modelReference* in each WSDL component. An initial version of this tool has been configured at http://khaos.uma.es/sawsdl/.

As future work we consider several alternatives for the extension of the presented methodology: firstly, generate schema mappings using the information available within the model reference set and secondly, add semantic to not annotated WSDL components from the scratch.

On the first hand, we are reviewing the most suitable techniques on data mediation in order to take advantage of the schema mappings during Web Service invocation. We are studying how to use existing model references within the WSDL file to dynamically mediate the required data. A preliminary result shows that mediation techniques in conjunction with matching functions could be useful in this scenario. The presence of similar approaches in more complex models for Semantic Web Services like WSMO [13] depicts that we are going in the right direction.

On the second hand, we want to measure how good our method is when it is performed directly using the WSDL descriptors, i.e, we try to provide the expert with a first set of annotations without processing any initial semantic model. Note that this wished approach is closer to the concept of automatic annotation, however, as it has been stated, the presence of the expert is still required to validate the fitness of the annotations.

References

1. Berners-Lee, T., Hendler, J., Lassila, O.: The Semantic Web. Scientific American (May 2001)
2. Semantic Annotations for WSDL and XML Schema, `http://www.w3.org/TR/sawsdl/`
3. Wilkinson, M.D.: BioMOBY: an open-source biological web services proposal. Links, M. Briefings In Bioinformatics 3(4), 331–341 (2002)
4. Euzenat, J., Shvaiko, P.: Ontology Matching. Springer, Heidelberg (2007)
5. Do, H.H., Rahm, E.: COMA - A System for Flexible Combination of Schema Matching Approaches. In: VLDB 2002, pp. 610–621 (2002)
6. Aumueller, D., Do, H.H., Massmann, S., Rahm, E.: Schema and ontology matching with COMA++. In: SIGMOD Conference 2005, pp. 906–908 (2005)
7. Drumm, C., Schmitt, M., Do, H.H., Rahm, E.: Quickmig: automatic schema matching for data migration projects. In: CIKM 2007, pp. 107–116 (2007)
8. Ontology Alignment Evaluation Initiative (OAEI), `http://oaei.ontologymatching.org/2008` (last visit: January 29, 2009)
9. Shvaiko, P., Euzenat, J., Noy, N.F., Stuckenschmidt, H., Benjamins, V.R., Uschold, M.: Proceedings of the 1st International Workshop on Ontology Matching (OM 2006) Collocated with the 5th International Semantic Web Conference (ISWC 2006), Athens, Georgia, USA, November 5. CEUR-WS.org (2006)
10. Martinez-Gil, J., Navas-Delgado, I., Polo-Marquez, A., Aldana-Montes, J.F.: Comparison of Textual Renderings of Ontologies for Improving Their Alignment. In: CISIS 2008, pp. 871–876 (2008)
11. Martinez-Gil, J., Alba, E., Aldana-Montes, J.F.: Optimizing Ontology Alignments by Using Genetic Algorithms. In: NatuReS 2008 (2008)
12. Navas-Delgado, I., Sanz, I., Aldana-Montes, J.F., Berlanga Llavori, R.B.: Automatic Generation of Semantic Fields for Resource Discovery in the Semantic Web. In: Andersen, K.V., Debenham, J., Wagner, R. (eds.) DEXA 2005. LNCS, vol. 3588, pp. 706–715. Springer, Heidelberg (2005)
13. Roman, D., Lausen, H., Keller, U. (eds.): The Web Service Modeling Ontology WSMO, final version 1.1. WSMO Final Draft D2 (2005), `http://www.wsmo.org/TR/d2/v1.1/`