Towards WSMO Ontology Specification From Existing Web Services

Houda EL BOUHISSI[1,2], Mimoun Malki[1,3], Djelloul Bouchiha[1,4],

[1] EEDIS Laboratory, Departement of computer sciences, Sidi-Bel-Abbes University, Algeria
[2] Houda.elbouhissi@gmail.com, [3] Malki@univ-sba.dz, [4] Bouchiha.dj@gmail.com,

Abstract. Semantic Web Services (SWSs) aim to improve the possibilities for automated discovery, composition and invocation of Web Services by providing ontology-based service descriptions expressed in a formal language. Several approaches have been driving the development of Semantic Web Service frameworks such as OWL-S (Ontology Web Language for Services), WSMO (Web Service Modeling Ontology) and WSDL-S (Semantic Annotation of Web Service Description Language descriptions). This paper focuses on WSMO; it aims to give a service ontology specification according to WSMO and using the reverse engineering techniques.

Keywords: Web Service, Semantic Web Services, WSDL, Ontology, Reverse engineering.

1 Introduction

Semantic Web Services aim to reduce the human effort required to build Service Oriented Architectures by enabling machines to understand the function and interfaces of Web services through the addition of semantics; However to enable the adoption of Semantic Web Service technologies by business it is crucial that adequate tool support exists which supports the engineer of Semantic Web Services and Semantically enabled Service Oriented Architectures through the full life cycle.

This paper describes a reverse engineering process [1], a software engineering technique that consists of extracting useful information from a WSDL file of an existing Web Service in order to build web service ontology according to the WSMO conceptual model.

A prototype is implemented which assists the process in the stages. The produced ontology is stored in WSML file, which provides a mean for the tasks such as discovery, composition and execution of Web Services.

The remainder of this paper is structured as follows. Section 2 gives a general overview of Web services; in section 3 we provide an overview of the Semantic Web and ontology, in section 4, we describe Semantic Web Services and section 5, discusses related works according to SWS. In section 6, we describe our approach and present how our tool has been used to map the WSDL file of Google Search Web

Service into WSMO ontology. Finally section 7 concludes the paper and gives future directives of the project.

2 Web Services

The W3C Web Services Architecture Working Group defines a Web service as: "A software application identified by an URI, whose interfaces and bindings are capable of being defined, described and discovered as XML artifacts. Web Service supports direct interactions with other software agents using XML-based messages exchanged via Internet-based protocols [2]. To define the infrastructure of the Service Web, a stack of interrelated standards, such as SOAP, UDDI and, WSDL are presented:

The Simple Object Access Protocol (SOAP[1]) is a specification for interactions among Web Services across the Internet. SOAP uses XML to exchange structured and typed information.

WSDL[2] (Web Service Description Language) is an XML-based language for describing Web services and how to access them. WSDL is also used to locate Web services.

As services become available, they may be registered with a UDDI[3] registry (Universal Description, Discovery and Integration) which can subsequently be browsed and queried by other users, services and applications.

3 Semantic Web

The goal of the Semantic Web is to solve the current limitations of the Web by augmenting Web information with a formal (i.e., machine processable) representation of its meaning [3]. A direct benefit of this machine processable semantics would be the enhancement and automation of several information management tasks, such as search or data integration. The Semantic Web s success and proliferation depends on quickly and cheaply constructing domain specific ontologies.

Ontologies [4] serve as metadata schemas, providing a controlled vocabulary of concepts, each with explicitly defined and machine processable semantics. By defining shared and common domain theories, ontologies help people and machines to communicate concisely supporting semantics exchange, not just syntax.

4 Semantic Web Services

A key problem with the use of standards for Web Service description (e.g. WSDL) and publishing (e.g. UDDI) is that the syntactic definitions used in these descriptions do not completely describe the capability of a service and cannot be understood by software programs. It requires a human to interpret the meaning of inputs, outputs and applicable constraints as well as the context in which services can be used. To make

1 http://www.w3.org/TR/soap12
2 http://www/w3.org/TR/wsdl
3 http://www.uddi.org

proper use of the Web services advertised by WSDL documents, programmers have to know in advance the intended meaning of the custom tags that specify the input and output schemes as well as the names of services that a Web service provides.

Semantic Web Services (SWSs) are the result of the evolution of the syntactic definition of Web Services and the semantic Web. Semantic Web Services will allow the semi-automatic and automatic annotation, advertisement; discovery, selection, composition, and execution of inter-organization business logic, making the Internet a global common platform.

5 SWSs Approaches

The Major initiatives in the area of SWSs are documented by recent W3C member submissions: OWL-S [5], WSMO [6] and WSDL-S [7]. The proposals differ in scope, modeling approach and the concrete logical languages used.

The first approach uses OWL-S, a description language that semantically describes Web Services using OWL ontologies. OWL-S services are mapped to WSDL operations, and inputs and outputs of OWL-S are mapped to WSDL messages.

The second approach, the Web Services Modeling Ontology, WSMO, provides ontological specifications for the description of semantic Web services. One of the main objectives of WSMO is to give a solution to application integration problems for Web services by providing a conceptual framework and a formal language for semantically describing all relevant aspects of Web Services.

WSDL-S is a third approach of creating semantic Web services is by mapping concepts in a Web service description (WSDL specification) to ontological concepts. The WSDL elements that can be marked up with metadata are operations, messages, preconditions and effects, since all the elements are explicitly declared in a WSDL description.

These approaches are complementary in many ways. Each initiative can be characterized in terms of (1) a conceptual model describing the underlying principles and assumptions; and (2) a language or a set of languages that provide the means to realize the model.

The WSDL-S takes a bottom-up approach by annotating existing standards with metadata using domain ontologies, the other approaches takes a top down approach according to a conceptual model and a specific language.

In this paper, we focus on the WSMO approach; we aim to generate the WSMO ontology semi-automatically using an implemented tool.

6 Reverse Engineering Process

The research contribution of this paper is the description of a reverse engineering process, a software engineering technique that consists of extracting useful information from a WSDL file of an existing Web Service in order to build web service ontology according to the WSMO conceptual model.

It mainly aims at moving from web service application where service s descriptions are described in a syntactic manner to semantic web service in order to automate discovery, composition and invocation of Web Services.

The reverse engineering also reduces the efforts and the cost to build new web service applications by reengineering.

We give first a short overview of WSMO and in the next sections; we describe our approach in details.

6.1 WSMO Overview

The WSMO initiative aims at providing an overarching framework for handling Semantic Web services (SWSs).

WSMO identifies four main top-level elements:

1. Ontologies that provide the terminology used by other elements;
2. Goals that state the intentions that should be solved by Web Services;
3. Web Services descriptions which describe various aspects of a service;
4. Mediators: to resolve interoperability problems.

Each of these WSMO Top Level Elements can be described with non-functional properties like creator, creation date, format, language, owner, rights, source, type; etc. WSMO comprises the WSMO conceptual model, as an upper level ontology for SWS, the WSML [8] language and the WSMX [9] execution environment.

The Web Service Modeling Language (WSML) is a formalization of the WSMO ontology, providing a language within which the properties of Semantic Web Services can be described.

WSMX provides an architecture including discovery, mediation, selection, and invocation and has been designed including all required supporting components enabling an exchange of messages between requesters and the providers of services.

6.2 WSMO Ontology Approach

The planned research can be split into the following sequence of high-level steps as presented in figure 1:

1. Extraction phase: extraction of information from the XML schema of the WSDL file.
2. Analyze Phase: definition of a mapping from the XML Schema Conceptual Model to the WSMO Ontology.
3. Translation phase: Translation of useful information into WSML specification.

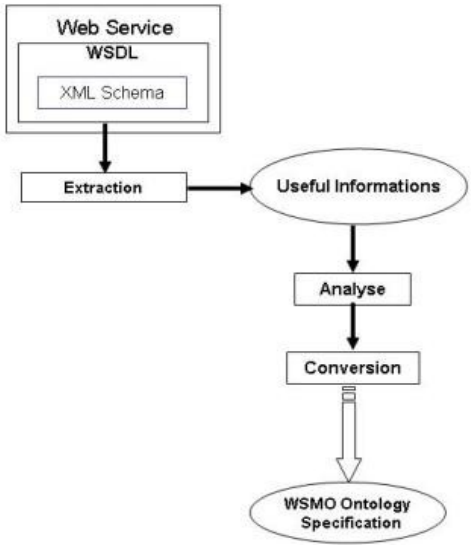

Fig. 1. Phases for WSMO Ontology Generation.

6.1.1 Extraction phase

We use the content of the WSDL file to derive WSMO ontology. However, we focus on the main part that provides us with information about the structure of valid XML document which is the XML schema definition.

Useful information is: Simple definition types, complex definition types and attributes.

6.1.2 Analysis phase

In this phase, we use the information extracted from the XML schema part of the WSDL file to define a mapping from XML elements to terms used in a WSMO Ontology. Figure 2 shows the definition of this mapping on the design level. This mapping is done automatically by the analyzer module and the results at least are stored in a simple XML file.

We propose simple mappings for each of the components extracted at the previews phase. The mapping produced is roughly based on the following rules:

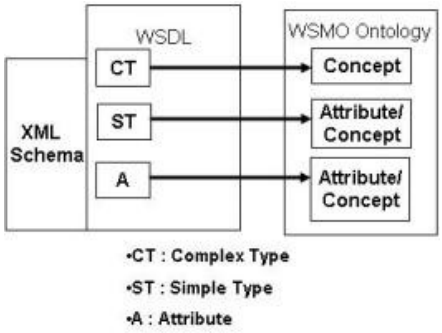

Fig. 2. Analysis Phase.

1. *Rule 1 : Simple type definition:*

If a simple type is used to create a new type based on restricting a built-in type, we create a new concept with the same built-in type and an axiom to define the restriction. For example the following simple type definition:

```
<xsd :element name=  age  >
<xsd :simpleType>
<xsd :restriction base=  xsd :positiveInteger  >
<xsd :maxExclusive value=  35  >
</xsd :restriction>
</xsd :simpleType>
</xsd :element>
```

The type age is mapped to the WSMO ontology concept with its build-type positiveinteger.

We deduce also a new axiom which is the restriction of the type: age>=35.

2. *Rule 2 : Complex Type Definition*

Complex type definitions can obtain sub-components that are a mixture of simple elements, attributes and other complex type definitions.

We propose to map each complex type to a concept in WSMO Ontology. Sub-components with simple type built-in are mapped to attributes with the same built-in type and attributes are mapped to attributes with the same built-in type.

If sub-component itself is a complex type, here we proceed in a recursive manner, we create first the corresponding concept , then the sub-component are mapped to attributes with the build-in type.

For example, a complex type definition:

```
<xsd :complexType name=  employe  >
  <xsd :element name=  name   type=  xsd :string  >
```

```
   <xsd :element name=  age  >
      <xsd :simpleType>
           <xsd :restriction base=  xsd :positiveInteger  >
              <xsd :maxExclusive value=  35  >
            </xsd :restriction>
       </xsd :simpleType>
    </xsd :element>
</xsd complexType>
```

The type employe is mapped to a concept with the corresponding attributes name and age and their built-in types.

Finally, the complex type embedded in another complex type is mapped at one hand to sub-concept of the complex type and at the other hand to a concept.

3. Rule 3 : Attributes

An attribute may be associated directly to the root or embedded in a simple or complex type.

If an attribute depends to the root, we propose to create a new concept with the built-in type.

If the attribute is embedded in a simple or a complex type, it is mapped to an attribute of the concept of the complex or the simple type.

For example the following declaration:

```
<xsd :complexType name=  job   type=  jobDesc  >
<xsd :attribute name=  jobid   type=  xsd :string  />
</xsd :complexType>
```

The type job is mapped to a concept embedding an attribute jobid with the build-in type string.

Furthermore, later than this phase, the list of the terms used for the ontology is stored in an XML file.

6.1.3 Translation phase

The goal of this phase is to translate the list of terms stored previous to in WSML specification.

This translation generates an ontology expressed in WSML language according to the WSMO conceptual model.

This phase proceeds with the help of a WSML template file stored in the local folder. The WSML template has several constants that are replaced with specific information extracted from the XML file produced in the analyze phase. Once the specific data is replaced, the new created WSML file is stored to be used next.

6.2. Implementation

In order to validate our approach, we create a prototype with eclipse platform and java language.

WSDL2WSMO (Web Service Modeling Language to Web Service Modeling Ontology), a semi automatic tool for the translation of the WSDL file to an incomplete WSMO ontology.

The Translator engine takes as input a WSDL specification and it returns as output a partial WSMO ontology description of the Web Service expressed in the WSML language.

Upon loading a WSDL file (Figure 3), WSDL2WSMO parses the WSDL file and extracts the XSD definitions defined between the WSDL type tags.

The extracted WSD definitions are mapped to terms used by the ontology following a set of mapping rules. The mapping rules produce a list of terms which is stored in an XML file.

Finally, WSDL2WSMO translates the terms into WSML specification and generates a WSML file.

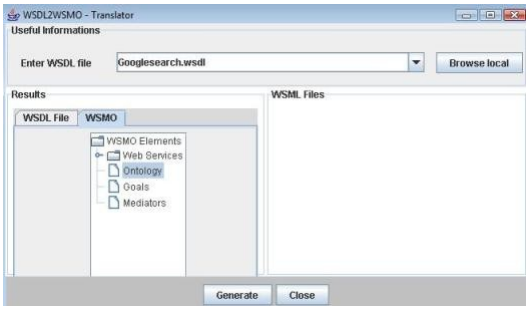

Fig. 3. Using the tool.

6.3. Case study: Google search

Google.com has exposed a Web service [10] that allows to put Google Search area in web pages.

The user can embed a simple, dynamic search box to display search results in his web pages or use the results in innovative, programmatic ways.

The WSDL description of Google search contains 20 types. Using the WSDL2WSMO tool, we were able to compile the WSDL specification into the corresponding WSMO ontology specification.

After this translation, the programmer is left with a WSML ontology file and three tasks to complete the WSMO specification (Web Services, Goals and Mediators).

7 Conclusion & Future Work

Summarizing, Semantic Web Services are an emerging area of research and currently all the supporting technologies are still far from the final product.

We have described the current main approaches of Semantic Web Services and we have proposed a new approach, reverse engineering process based to specify WSMO Ontology from WSDL. Nevertheless, there are still a number of issues concerning Semantic Web Services being investigated in a number of initiatives.

These issues will have the attention of industry and academia for the next few years.

Finally the present research has attempted to give a partial specification of the WSMO ontology with the information extracted from the XML schema part of the WSDL file.

The work described in this paper may be generalized to include UDDI registry in order to obtain complete specification of WSMO ontology and make the process completely automatic.

8 References

[1] Chikofsky E.J., Cross II J.H.: Reverse engineering and design recovery: A taxonomy. IEEE Software, 13, 1990.

[2] Web services architecture requirements, W3C Web Services Architecture Working Draft, Available online at: http://www.w3.org/TR/2002/ WD-wsa-reqs-20021114.

[3] Berners-Lee, T., Hendler, J., and Lassila, O. (2001). The Semantic Web. Scientific American, 284(5):34 - 43.

[4] Gruber,T.(1993). A Translation Approach to Portable Ontology Specifications. Knowledge Acquisition, 5(2):199 –220.

[5] OWL Services Coalition, OWL-S: Semantic Markup for Web Services, Dec 2003, http://www.daml.org/services/owls/1.0/owl-s.html.

[6] Dumitru Roman, Holger Lausen, and Uwe Keller. Web service modeling ontology (WSMO). Final Draft D2v1.3, WSMO, 2006. Available from: http://www.wsmo.org/TR/d2/v1.3/.

[7] Akkiraju, R., Farrell, J., Miller, J., Nagarajan, M., Schmidt, M., Sheth, A., & Verma, K. (2005), Web Service Semantics - WSDL-S, W3C Member Submission 7 November 2005, Retrieved April 4, 2006, from http://www.w3.org/Submission/2005/SUBM -WSDL-S-20051107/

[8] H. Lausen, J. de Bruijn, A. Polleres, and D. Fensel. WSML - A Language Framework for Semantic Web Services. In Proc. of the W3C Workshop on Rule Languages for Interoperability, 2005.

[9] A. Haller, E. Cimpian, A. Mocan, E. Oren, and C. Bussler. WSMX - A Semantic Service-

Oriented Architecture. In Proceedings of the International Conference on Web Service (ICWS 2005), 2005.

[10] WSDL description of the Google web APIs, available from : http://api.google.com/GooleSearch.wsdl.