Management Tool for Semantic Annotations in WSDL

Nicolas Boissel-Dallier[1,2], Jean-Pierre Lorré[1], and Frédérick Benaben[2]

[1] EBM WebSourcing,
4 rue Amélie, 31000 Toulouse, France
{nicolas.boissel-dallier, jean-pierre.lorre}@ebmwebsourcing.com
[2] Ecole des Mines d'Albi-Carmaux,
Route de Teillet, 81000 Albi, France
frederick.benaben@mines-albi.fr

Abstract. Semantic Web Services add features to automate web services discovery and composition. A new standard called SAWSDL emerged recently as a W3C recommendation to add semantic annotations within web service descriptions (WSDL). In order to manipulate such information in Java program we need an XML parser. Two open-source libraries already exist (SAWSDL4J and Woden4SAWSDL) but they don't meet all our specific needs such as support for WSDL 1.1 and 2.0. This paper presents a new tool, called EasyWSDL, which is able to handle semantic annotations as well as to manage the full WSDL description thanks to a plug-in mechanism. This tool allows us to read/edit/create a WSDL description and related annotations thanks to a uniform API, in both 1.1 and 2.0 versions. This document compares these three libraries and presents its integration into Dragon the OW2 open-source SOA governance tool.

Keywords: Semantic Annotations, SAWSDL, Service Oriented Architecture, ontology, SOA governance.

1 Introduction

According to the World Wide Web Consortium (W3C), a web service is a software system designed to support interoperable machine-to-machine interaction over a network [1]. Web services use Web Services Description Languages (WSDL) to display what the service offers. Unfortunately, first, two versions of this W3C standard are already used (1.1 and 2.0) and, second, it is limited to syntactic (non-semantic) data.

In order to improve interoperability and reuse of code, especially in inter-enterprise collaboration contexts, information systems need to exchange machine understandable data and operations [2]. For that reason, we have to connect each web services with reference concepts which are defined between all partners. Toward this goal, the W3C published a new standard which allows to link web service description parts to semantic concepts, whatever the ontology language. This recommendation is called

R. Meersman, P. Herrero, and T. Dillon (Eds.): OTM 2009 Workshops, LNCS 5872, pp. 898–906, 2009.

Semantic Annotations for WSDL and XML Schema (SAWSDL)[1] and extends both WSDL standards and XML Schema specification[2]. This standard is an important step toward automating discovery, execution and composition of Web services. Moreover, in a concern of code reuse, it can be interesting to dispose of a SAWSDL management tool. This is the aim of EasyWSDL, presented in this article.

This work is based on an ongoing work supported both by the SemEUse ANR project[3] funded by the French government and by SOA4ALL[4], a NESSI strategic integrated project. EBM WebSourcing involvement in these projects is focusing on high scale service distribution and semantic for next generation service infrastructure and governance.

First, this paper presents WSDL standard and related open-source parsers (Section 2). Then, it introduces SAWSDL and associated tools (Section 3). The next part presents EasyWSDL, a new WSDL parser and its SAWSDL extension (Section 4), comparing to others. In order to illustrate previous points, it also presents the integration of a SAWSDL editor, based on EasyWSDL and embedded in Dragon, the SOA governance tool of the OW2 consortium (Section 5). Finally, the section 6 concludes this paper.

2 WSDL Standard and Associated Tools

2.1 Web Service Description Language

WSDL is one of the three technologies which underlie Web Services (together with UDDI and SOAP). It is an XML-based language that provides a model for describing web services. It contains abstract information about operations and exchanged messages but also concrete data like network access parameters or binding protocol.

Two versions of WSDL recommendation already exist: the 1.1[5] version (which was a W3C Submission but is still used in almost all existing systems) and the 2.0[6] version (which is a W3C Standard intended to replace 1.1). Both versions are functionally quite similar but have substantial differences in XML structure (renamed and removed tags).

2.2 WSDL Parsers

Two aspects are important for semantics handling in a programming language: a reliable WSDL management tool and a full implementation of SAWSDL recommendation. In the next section, we present the two main WSDL handlers Java libraries which both have SAWSDL management extension namely WSDL4J and Woden. The section 4 gives more technical information and presents a benchmark.

[1] http://www.w3.org/TR/sawsdl/
[2] http://www.w3.org/TR/xmlschema-0/
[3] http://www.semeuse.org/
[4] http://www.soa4all.eu/
[5] http://www.w3.org/TR/wsdl
[6] http://www.w3.org/TR/wsdl20/

Web Services Description Language for Java Toolkit (WSDL4J)[7] is an open-source Java library developed by IBM under CPL License and supported by SourceForge. It is the reference implementation of JSR110, the Java APIs for WSDL [3] and it is based on a WSDL 1.1 object model.

WSDL4J enables to read, write and create every WSDL 1.1 tags and automatically imports required documents. Unfortunately, it doesn't handle the XML Schema part (treated as DOM tree) and it doesn't allow managing of WSDL 2.0 compliant documents.

Woden[8] is an open-source Java library which implements the W3C WSDL 2.0 recommendation. It's an Apache project under Apache License.

Woden is based on a full WSDL 2.0 object model. It enables to read, write and create WSDL 2.0 document and can translate WSDL 1.1 file into 2.0 one, in order to parse it after. Unfortunately, this additional step brings data loss in the file such as message part types (links to XML Schema). Like WSDL4J does, it also imports automatically external parts of WSDL. It doesn't handle XML Schema but return an Apache XmlSchema object, manageable by an open-source library also available from the Apache Foundation.

Woden is already used in Apache Axis2[9], a Web service framework, and ServiceMix[10], a JBI-based [4] Enterprise Service Bus.

3 SAWSDL and Tools

3.1 Semantic Services

According to [5], the goal of Semantically-enabled Service Oriented Architecture is to add feature that can provide mechanisms to automate tasks which currently require human intervention. To make Semantic Web Services (SWS) possible, some solutions have been standardized. They are proposed by various working groups, like the W3C or the European Semantic Systems Initiative (ESSI).

On one hand, WSMO[11] and OWL-S[12] adopt a top-down approach to model services. They define complete frameworks to describe semantics for services without taking care of existing SOA technologies such as WSDL. This method is very powerful for the use of semantic in Web Services because the meta-model was made to support dynamic discovery and composition. However, this approach complicates changes on existing system because it sets rewriting of all service descriptions.

On the other hand, SAWSDL and WSMO-Lite [6] adopt a bottom-up approach for service modeling: they add small increments on top of WSDL in order to keep most of existing deployed technologies. SAWSDL allows users to add semantic annotations in WSDL file from any reference ontology in WSDL files. WSMO-Lite is an evolution of SAWSDL, filling the semantic annotations with a specific service ontology based on a simplified version of WSMO.

[7] http://wsdl4j.sourceforge.net/
[8] http://ws.apache.org/woden/
[9] http://ws.apache.org/axis2/
[10] http://incubator.apache.org/servicemix/
[11] http://www.wsmo.org/2004/d2/v1.0/
[12] http://www.w3.org/Submission/OWL-S/

The bottom-up approach seems to be a relevant choice to develop an industrial implementation of Semantic Web Services. The W3C Standard SAWSDL is one of the pillars of SWS and is compatible with WSMO-Lite submission. However, to use it in concrete projects, it is useful to dispose of an implementation of SAWSDL, which allows us to manage semantic annotations.

3.2 SAWSDL Parsers

Let's turn our attention to available SAWSDL extensions parsers. Both are developed by university research groups.

SAWSDL4J[13] is an open-source WSDL4J extension developed by Knoesis, a research center from Wright State University, specialized in semantic e-science.

This implementation changes some interfaces of WSDL4J object model and add methods to extract and set model references and schema mappings. Semantic annotations are cleanly manageable on port types, operations, messages and parts. Others can be handled by XPath[14] system (for elements and types) or with WSDL4J attribute management method (for faults and other tags). Then, SAWSDL4J handles semantic annotations with different methods as well as only manages WSDL 1.1 because of WSDL4J library.

Woden4SAWSDL[15] is a part of the METEOR-S[16] project from LSDIS, a University of Georgia research center. It is the Woden extension for semantics management.

This implementation adds a utilitarian class which extracts semantic annotations from Woden object passed in argument. The whole SAWSDL specification is covered but with read-only methods. No writing or creating method is expected for now.

3.3 Incomplete Solutions

Those two implementations of the SAWSDL standard represent a first step in semantic handling but they do not meet all users' needs.

First, they manage only one version of WSDL whereas both can be used in current or future systems. In order to covers both WSDL versions, libraries and treat services should be combined differently.

Next, some important functionalities are missing. For example, WSDL4J does not directly manage XML Schema while Woden4SAWSDL cannot add or edit semantic annotations. These functionalities are yet necessaries to facilitate SWS.

Then, handling of descriptions and semantic annotations is not uniform in those libraries. SAWSDL4J embed semantics in its object model for main tags, use XPath for XML Schema and a third method to others tags. Woden4SAWSDL add an utilitarian class whereas all other attributes are handled with an object model. A uniform API is not essential but it can facilitate development and code comprehension.

---

[13] http://knoesis.wright.edu/opensource/sawsdl4j/
[14] http://www.w3.org/TR/xpath20/
[15] http://lsdis.cs.uga.edu/projects/meteor-s/opensource/woden4sawsdl/
[16] http://lsdis.cs.uga.edu/projects/meteor-s/

Finally, the choice of the open-source licenses (CPL and Apache) can be incompatible with some projects due to the specificities of any type of licenses.

4 EasyWSDL and Its SAWSDL Extension

EasyWSDL[17] is a new WSDL open-source Java implementation developed by EBM WebSourcing for the OW2 Consortium under BSD License. It was initially developed to fulfill requirements that others did not meet, especially for WSDL management in PEtALS[18], the Enterprise Service Bus of the OW2 consortium and Dragon[19], the SOA governance tool. Both programs are under LGPL license which is compatible with BSD.

4.1 A Uniform Management of WSDL

EasyWSDL recognize both WSDL versions. Then, it enables to read, write and create WSDL 1.1 or 2.0, with a uniform API. Figure 1 gives a representation of the three proposed approaches (WSDL4J, Woden, and EasyWSDL).

As depicted by the next figure, EasyWSDL keep WSDL 2.0 tag names if possible. His full object model also follows the API and the W3C recommendations. For example, the root tag in WSDL 1.1 is `definition` against `description` in WSDL 2.0. Both API and object model in EasyWSDL use `description` as name.

Such a native handling of both WSDL versions avoid conversion phase (existing in Woden for WSDL 1.1 treatment) which produce lost of data.

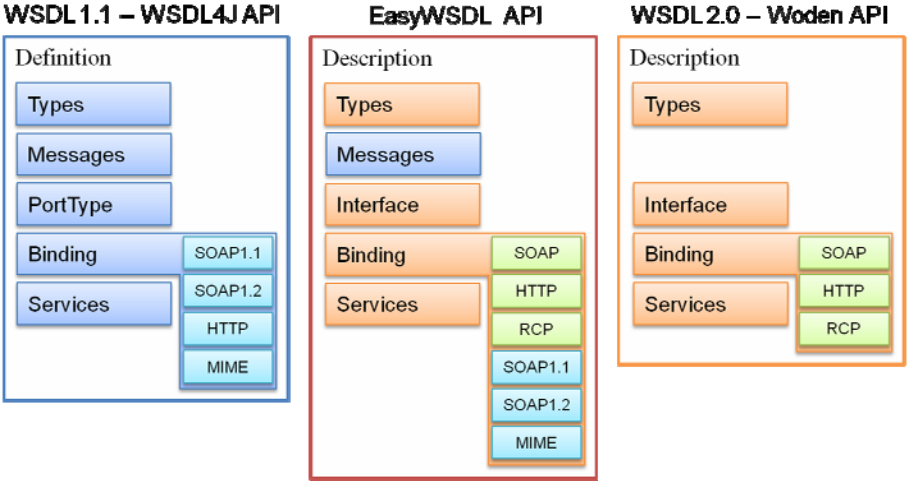

Fig. 1. Overview of the three API

[17] http://easywsdl.ow2.org/
[18] http://petals.ow2.org/
[19] http://dragon.ow2.org

EasyWSDL includes also an XML Schema management with its embedded object model. It enables to parse `types` tag in depth and bind those with the rest of the description (input, output, fault…) thanks to the object models. This implementation is also compliant with the recommendation. Schema handling is useful to manage input and output then enables communication between services.

4.2 The Extension System

EasyWSDL was designed according to a plug-in architecture. Each model's object is associated with a decorator pattern. This extendable layer allows developers to add properties and methods to classes, like any simple extension, but also allows users to combine them. Three extensions are already available:

- SAWSDL: It makes possible the handling of semantic annotations. For more information, see Section 4.4.
- WSDL4ComplexWSDL: Enables to work in distributed environments. It allows EasyWSDL to merge all imports with the main description, solving location problems.
- WSDL4BPEL: Add recognition of `partnerLink` tags which is essential to use WSDL documents in a BPEL process.

Thanks to the decorator pattern, developers may combine extensions as they need and develop new ones. For example, if a user want to manage WSDL file, taking account both semantic annotations and `partnerLink` tags, he just have to activate SAWSDL and WSDL4BPEL extensions.

4.3 Performances Comparisons

EasyWSDL seems to meet all users' needs but this library is only viable if it proposes an equivalent (or better) quality of services. In order to compare the three implementations of WSDL, we benched all with a large-scaled test. We used the Seekda[20] database which contains more than 40000 WSDL documents. All tests were carried out under the same conditions: we created a reader object for each Java library then all WSDL file were parsed. Reading times and errors were noticed and sum up in the following table.

After the first test batch (See Table 1), we can notice that WSLD4J is faster than EasyWSDL (about 20% faster). In compensation, EasyWSDL seems to be more robust (no memory leak and less parsing errors) and treat types as object model instead of a simple DOM tree.

All documents from the database are in version 1.1 and Woden only accepts 2.0 descriptions. So, we passed all files in an XSLT process to make them compliant with WSDL 2.0 standard. This transformation can't be realized on all documents: some invalid 1.1 descriptions didn't pass the XSLT engine (1880/40257 i.e. about 4.7%).

The second test shows that Woden is 53% slower than EasyWSDL with closed functionalities. Both exception managements are hardly comparable: for example,

[20] http://www.seekda.com/

Table 1. Performance tests with WSDL 1.1

Java library	WSDL4J	EasyWSDL
Tested files	**40377**	**40377**
Errors[21]	51	**9**
% of success	98.87	**99.98**
Avg. time (ms)	**24**	30
Memory leak[22]	Yes	**No**

Table 2. Performance tests with WSDL 2.0

Java library	Woden	EasyWSDL
Tested files	**38377**	**38377**
Avg. time (ms)	**46**	30
Memory leak	**No**	**No**

Woden logs import errors while continuing parsing whereas EasyWSDL throws exception and stop handling. This is a relevant choice but produce irrelevant error-rate comparison. Furthermore, most of incompliant descriptions were stopped during 1.1 to 2.0 conversions while those files bring most of parsing problems.

4.4 SAWSDL Management

As seen above, EasyWSDL provides its SAWSDL extension. It was developed under BSD license too. This implementation enables to read, write and create any semantic annotation. It respects W3C recommendation and handles semantic annotations directly from the object model, even for the WSDL 1.1 operations. It manages `SchemaMapping` attributes in any elements or types (simple and complex) and `modelReference` attributes in all WSDL tags, including XML Schema. The uniform API enables integration in any external project, without taking care of the WSDL version.

Tests were performed with the SAWSDL-Test Collection[23] a service retrieval test collection created to test SWS algorithms and make benchmarks. It counts 894 SAWSDL descriptions with associated OWL ontologies and XSLT schema mapping. Those tests show that semantic annotations parsing do not have any influence on performances. Other unit tests were realized to check returned values: results are similar for the three Java libraries. Then, performances of SAWSDL libraries only depend on WSDL parsers.

[21] Number of errors doesn't take into account exceptions thrown because of inaccessible files. We consider these exceptions are expected because some information is missing.

[22] All parsing were initially carried out with one reader. If the resources used by the unit test increase until a memory fail, we consider the reader contains memory leak.

[23] http://www.semwebcentral.org/projects/sawsdl-tc/

5 Development of a SAWSDL Editor within the Dragon SOA Governance Tool

5.1 Presentation of Dragon

SOA governance is the ability to organize, enforce and re-configure service interactions in a SOA. Linked to this definition, we can identify two main phases for SOA Governance called design time and runtime. Ability to organize appends at design time with the registry/repository concepts. Ability to enforce and reconfigure appends at runtime with the service platform interface between the service runtime layer and the registry layer.

Dragon[24] is the SOA governance tool developed by EBM WebSourcing in the OW2 consortium. This open source software is designed to manage services (including lifecycle, dependencies, versioning…), WS-* policies, Service Level Agreement (SLA) and runtime. Dragon plans to manage, to stock and to use semantics to make dynamic composition of web services possible.

Dragon is composed of two main components; the Registry and the Service Platform Interface. The Registry provides classical functionalities like those provided by UDDI based registry but also include some enhanced governance features like SLA contract, policy management, lifecycle management, dependency management and versioning of services and services artifacts. The Service Platform Interface is the communication layer between the Enterprise Service Bus and Dragon registry. It allows taking into account necessary policies enforcement like those specified in SLA contract or enterprise wide policies.

5.2 Integration of EasyWSDL

EasyWSDL was initially developed to fulfill requirements of Dragon in terms of WSDL description and semantic annotations parsing. Dragon already included EasyWSDL but only used non-semantic functionalities.

The first step to improve semantic functionalities in Dragon is to allow users to handle semantic annotations in service descriptions which are stocked in Dragon repository. This feature has to be available directly from the web-based graphic interface to enable users to manage WSDL file from any workstation. According to this objective, we create a SAWSDL editor embedded in Dragon, using its graphic interface and its repository.

This editor allows users to manipulate any service description from Dragon database in order to add or edit semantic annotations. It uses an ergonomic tree system which allows to manage tags in depth. All WSDL and XML Schema tag are reachable and user can add, edit or delete semantic annotation for each, whatever the ontology language (including WSMO-Lite).

Management of semantic annotations was also added in all Dragon's pages which concern web services descriptions. For example, new fields were added in service management pages. This double annotation handling avoids separation of syntactic and semantic in the registry.

[24] http://dragon.ow2.org/

6 Conclusion and Future Work

SAWSDL standard has a new open-source Java library with EasyWSDL and its extension. Even if this one lack of maturity due to a recent development, it seems to be a good alternative to existing solutions. Indeed, it enables to handle both WSDL versions, including XML Schema, the whole with a full object model. Moreover, thanks to its extension system, it allows users to manage semantic annotation in any part of the document, following W3C recommendation. Finally, flexibility of attribute's values in SAWSDL enables to refer to concepts from any ontology language. WSMO-Lite is compatible with this library and limit annotation types for an efficient industrial using. This new implementation meets needs that others did not and it could be a good base for new SWS projects.

EasyWSDL is already used in Dragon and semantic annotation management is in progress. Development of the SAWSDL editor is the first step of semantic use in Dragon. Eventually, Dragon plans to provide such functionalities as service matchmaking and dynamic composition of services.

References

1. W3C, Web Services Glossary, http://www.w3.org/TR/ws-gloss/
2. Hendler, J.: Agents and the Semantic Web. IEEE Intelligent System 16, 30–37 (2001)
3. Fermantle, P., Duftler, M.: IBM Corporation: JSR110 - Java APIs for WSDL (2006), http://jcp.org/en/jsr/detail?id=110
4. Julson, E., Hapner, M.: Sun Microsystems Inc.: JSR208 - Java business Integration (JBI) 1.0 (2005), http://jcp.org/en/jsr/detail?id=208
5. OASIS SEE TC: Reference Ontology for Semantic Services Oriented Architectures, http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=semantic-ex
6. Vitvar, T., Kopecký, J., Viskova, J., Fensel, D.: WSMO-Lite Annotations for Web Services. In: Bechhofer, S., Hauswirth, M., Hoffmann, J., Koubarakis, M. (eds.) ESWC 2008. LNCS, vol. 5021, pp. 674–689. Springer, Heidelberg (2008)