SASCHA: Supporting the Italian Public Cooperation System with a Rich Internet Application for Semantic Web Services

Alessandro Adamou

Semantic Technology Lab, CNR-ISTC

Abstract. We present SASCHA, a web browser-based application for authoring semantic Web Service descriptions by the SAWSDL standard. Even though it was conceived in a specific real domain scenario, i.e. the Italian framework for service interoperability, it is in fact a general-purpose SAWSDL tool that showcases a singular feature set including, but not limited to, simplicity of use, availability as an infrastructural service and XML Schema support. Although upper-level ontologies are expected to be stored in an ad-hoc repository, SASCHA allows for OWL ontologies being stored anywhere locally or on the web.¹

1 Background

The Public Cooperation System (abbreviated SPCoOP) is the framework for Public Administration interoperability that the Italian government has been promoting since its inception in 2007. It is built on top of a large ad-hoc connectivity infrastructure called Public Connectivity System (SPC) and is a Service Oriented Architecture based on Semantic Web Services, which are exposed by each public organization and assembled as e-Government workflows. The adoption of semantic technologies in the scope of this architecture is aimed at facilitating service discovery, composability and reuse as well as reducing startup costs for new organizations intending to join the system [1].

SPCoOP conforms to the WS-I Basic Profile 1.0 [2], hence specifies that service descriptions be exposed as sets of formal specifications (called Service Agreements), which include XML documents written in WSDL (Web Service Description Language) version 1.1, a format that the newer WSDL 2.0 is only partially retro-compatible with. Concept-level service specifications are integrated with the SAWSDL extension for Semantic Web Services [3], allowing services to reference concepts describing their components and the information conveyed [4]. SAWSDL represents the state of the art in terms of semantic Web Service definition and exposure, as it allows immediate validation by matching service element annotations with related semantic entities. In SPCoOP these entities may

¹ This work is the result of a combined effort of the Italian National Agency for Digital Administration (CNIPA), the University of Rome “La Sapienza” - Department of Computer Science, and the National Research Council - Institute of Cognitive Sciences and Technologies (CNR-ISTC).

include, but are not limited to elements from a shared knowledge repository called Schema and Ontology Catalog.

The need for a simple, richly interactive tool for authoring SAWSDL model references on existing Service Agreement components, as is SASCHA, is due to a number of different factors. For one thing, the application suite developed specifically for supporting SPCOOP administration or participation therein relies upon third-party tools for annotating Service Agreements. These tools, which we will look into in the following section, do not fully cover the application of SAWSDL extensions to WSDL 1.1 documents and stand-alone XML Schemas. Also, it is not possible to presume any degree of knowledgeability from domain experts in each organization, with respect to knowledge modelling and ontology engineering. This is due to two reasons: first, semantic annotation of services in SPCOOP occurs when such services are published, which may happen anytime after the domain knowledge has been formalized and registered; then, multi-provider services may be managed by a neutral third party, which may have had no part in authoring domain knowledge. Finally, by following a Software as a Service approach, these functionalities could be provided as remote infrastructural services available anytime, anywhere, thus favoring the adoption of the Web application paradigm. Originally presented in [5], the AJAX application SASCHA was developed to address these issues.

2 Relation with Existing Work

SASCHA makes an extensive use of tried-and-true technologies which encompass several key components of semantic web applications, from the user interface to core functionalities. Most of the GUI was written using GWT-Ext, an open-source Google Web Toolkit wrap of the Ext widget library, which is already being employed for the *OwlSight* ontology browser² as well as the *WebProtege* editor³. Although additional parsing for XML syntax highlighting is performed on the client, all the business logic (namely serialization and parsing of WSDL and XSD, ontology loading and reasoning) is implemented server-side and made available as GWT servlets. The OWL API is employed for ontology management, and is backed by *Pellet* for additional reasoning tasks. Interpretation of WSDL 1.1 documents relies upon the *WSDL4J* library, to which SAWSDL model reference support is provided through a modified version of the *SAWSDL4J* library. Because their object model JWSDL does not offer direct support for stand-alone XML Schemas, a small plugin (*SAXSD4J*) was also developed ad-hoc.

Having conceived SASCHA in response to standards and features which fall short of coverage among existing Semantic Web Service tools, an evaluation of similarities and differences with the state of the art is appropriate. For comparison, we have focused on three Semantic Web Services tools, which all happen to be based on the Eclipse framework: the LSDIS *Radiant*⁴ plugin, *WSMO Studio*⁵

² <http://pellet.owldl.com/ontology-browser>

³ <http://protegewiki.stanford.edu/index.php/WebProtege>

⁴ <http://lsdis.cs.uga.edu/projects/meteor-s/downloads/index.php?page=1>

⁵ <http://www.wsmostudio.org>

and *Semantic Tools for Web Services*⁶ by IBM alphaWorks. These tools were tested with reference to the W3C SAWSDL implementation report⁷, against evaluation criteria including support for SAWSDL and version 1.1 of the WSDL standard, the ability to annotate non-embedded XML Schemas and the availability of a common view for representing and interacting with annotation source and target - what is called “Shared UI Perspective” in the following table.

Table 1. Comparison of semantic Web Service annotation tools

	SAWSDL Support	WSDL 1.1 Support	Stand-alone XSD Support	Shared UI Perspective
Radiant	yes	partial	yes	yes
WSMO Studio	yes	partial	no	no
Semantic Tools for WS	no	yes	no	N/A
Sascha	yes	yes	yes	yes

The Semantic Tools for Web Services were developed with support for the WSDL-S specification, which preceded the SAWSDL standard. As for the remaining candidates, Radiant has shown to be the only one to support stand-alone XML Schemas and to offer simultaneous display for ontologies and annotation targets, which can prove helpful for inexperienced users. However, it still shows minor interface quirks and limited WSDL support, which only encompasses common elements between versions 1.1 and 2.0 of the standard. On the other hand, WSMO Studio provides additional support for the `attrExtensions` extensibility element, which is required for annotating operations in WSDL 1.1. It does, however, allow blank `attrExtensions` elements, which is generally not a desirable feature.

3 Demonstration Scenario

In the demonstration session, visitors will learn how a WSDL description and an XML Schema can be annotated in SASCHA with references to entities from several sources. They will also learn how an annotation can be edited or deleted, as well as alternate paths to perform these actions across different resource views. A localized user interface in English will be provided for the purpose.

The example task is to load an annotated resource and replace its most generic model references with new ones pointing at appropriate, more specific entities (such as subclasses or instances of originally referenced classes). The demonstration will consist of the following steps:

1. Loading an annotation target (WSDL or XSD), at least one suitable ontology and their dependencies into SASCHA.

⁶ <http://www.alphaworks.ibm.com/tech/wssem>

⁷ <http://www.w3.org/2002/ws/sawSDL/CR>

2. Introducing visitors to the three available display methods - i.e. tree, stack and source code view - which are used for WSDL and XSD documents as well as OWL ontologies.
3. Removing existing model references that must be replaced. Technically, this is not strictly necessary, but it is useful for showing the context menus that are invoked on tree nodes representing one model reference each.
4. Performing SAWSDL annotations through drag-and-drop actions across both tree and stack views.
5. Verifying the impact of previous actions on the source code view, which is updated in real-time on every change in the model reference set.

When visitors are introduced to the SASCHA user interface, the functions of the lower tabbed panel, which is used for diagnostics, ontology status and dependency listing as well as entity slots, will also be displayed. An example of this setup is shown in Figure 1.

Fig. 1. Opening a WSDL document and a set of ontologies in SASCHA

At least one WSDL document and one XML Schema will be loaded as active items at different times. If these documents reference any other XML Schemas by means of **import**, **include** and **redefine** elements, the referenced schemas will be brought in as well. As for ontologies, they may or may not be bound by OWL **imports** declarations. Users have the option to instruct the reasoner so that it will infer a class hierarchy from the whole set of ontologies the system

is fed, rather than only the active one, which is always the last in the upload queue. Failure to load ontology dependencies is not fatal, as users are able to operate with the available set despite being warned about missing imports.

When annotating a single element, multiple semantic entities can be selected at a time if needed. If an entity belongs to a namespace that has not been bound to any prefix on the target document (which is going to occur at least once), a dialog will pop up to request a prefix for shorthand references. This action is optional, as the full entity URI will be used as a model reference if no prefix is entered.

4 Final Remarks and Future Work

Our demonstration will outline the features of a platform-independent, easy-to-use tool that provides a straightforward interface and several alternate paths for adding semantic value to Web Services descriptions and XML Schemas, and sets the ground for a full-fledged application for easily authoring Semantic Web Service descriptions from scratch.

Although tightening the bond with SPCOOP, by releasing modules for supporting Service Agreement navigation and Schema and Ontology Catalog querying, is still top-priority, the migration of SASCHA to a complete SAWSDL environment is part of the development plan. Some key development points are listed as follows:

1. Decision support by mapping WSDL and XSD schema definitions to mediator ontology patterns which can be best-matched against the active ontology;
2. Coverage of SAWSDL schema mappings applied to XSD elements;
3. Support for WSDL 2.0 and non-normative semantic annotations;
4. Natural language annotations through `wSDL:documentation` nodes.

At the time of writing, SASCHA is undergoing testing for its first public release, possibly as LGPL software.

References

- [1] Baldoni, R., Fuligni, S., Mecella, M., Tortorelli, F.: The Italian e-Government Service-Oriented Architecture. Strategic Vision and Technical Solutions (2007)
- [2] Ballinger, K., Ehnebuske, D., Gudgin, M., Nottingham, M., Yendluri, P.: Basic Profile Version 1.0, The Web Services-Interoperability Organization (2004), <http://www.ws-i.org/Profiles/BasicProfile-1.0-2004-04-16.html>
- [3] Farrell, J., Lausen, H.: Semantic Annotations for WSDL and XML Schema, W3C Recommendation (August 28, 2007), <http://www.w3.org/TR/sawSDL>
- [4] Sheth, A., Verma, K.: Semantically Annotating the Web Services. IEEE Internet Computing 11(2), 83–85 (2007)
- [5] Adamou, A.: SAScha: a RIA approach for supporting Semantic Web Services in the Italian interoperability framework. In: Proceedings of the 5th Workshop on Semantic Web Applications and Perspectives (SWAP 2008) (2008)