SAscha: A RIA Approach for Supporting Semantic Web Services in the Italian Interoperability Framework

Alessandro Adamou

Università degli Studi di Roma "La Sapienza"

Abstract. SPCoop is a nationwide framework for supporting service interoperability in the exercise and application of Italian government policies. The very core of this framework lies in the use of Semantic Web Services as a means to execute even complex administrative processes in a semi-automatic way, hence the need for added semantic value in the definition of such services. Because each participant is responsible for both authoring its domain knowledge representation and defining the services provided, and these tasks may be performed at different times by different individuals who may or may not belong to the same domain, the problem of heterogeneous individual knowledgeability arises. Also, the set of standards that were adopted in SPCoop includes specifications that have since been replaced by newer versions and show little coverage by existing implementations. SAscha is a Web tool intended to address such issues, offering an essentially simple WYSIWYM approach to annotating Web Service descriptors with SAWSDL references to concepts within domain ontologies which are stored in an *ad-hoc* repository. SAscha is provided as a Rich Internet Application (RIA) so as to be deployed as an infrastructural service itself for immediate use, without any need to resort to potentially complex client applications.[1]

1 Introduction

The ongoing trend of re-addressing responsibility for government processes towards local and autonomous administrative units, so as to follow a quasi-federal model, has raised a number of issues concerning the sustainability of such an organisational model in terms of scalability and continuous integration of governance processes. In the light of the current shift of focus towards an e-Government perspective over the last decade, such issues must be re-thought with respect to application interoperability and an effort to push the bulk of administrative information over the Internet. A response to these issues by the Italian institutional establishment is to be found within its proposal for a National interoper-

[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).

ability framework based on the Service-Oriented Architecture paradigm, called SPCoop.

This framework, whose name roughly translates to Public Cooperation System, is essentially an SOA operating on top of a dedicated connectivity infrastructure called Public Connectivity System. This architecture, often shorted as SPC, sprung to life in December 2007 as part of a full-scale normative plan, called the Digital Administration Code, to enforce e-Government policies in Italy. SPC is aimed at ensuring a secure, trusted and efficient means of delivering not only the information that is exchanged between services built on top of SPCoop, but all of the Public Administration-related network traffic. Several key principles of the SOA doctrine are satisfied by the SPCoop core components, most significantly service *loose-coupling*, *reuse*, *encapsulation*, *composability*, *contract* and *discovery*. All aspects will be looked into when defining these core components.

Figure 1[2] represents the approach taken by SPCoop on managing a service-based transaction between two peers, which may be single administrations by their own right as well as subjects operating on behalf of complex domains where several administrations concur in defining a composite procedure to achieve a common goal. Each administrative domain exposes a service endpoint, the *Domain Gatweway*, by which all SPCoop-related traffic is exchanged. Assuming peer A has already discovered a suitable service S as being provided by peer B, its message exchange patterns, naming and structure, as well as implementation-related information are provided as a set of documents, forming what is called a *Service Agreement*, which is stored in an *ad-hoc* registry loosely based on UDDI. An object of this kind is defined per service and <provider, consumer> couple and is split in two major components: one, the *common part*, is defined for the sake of reusability and contains a conceptual and logical description of the service stored as a set of WSDL (Web Service Definition Language) 1.1 descriptors, along with QoS and security specifications and (optional) natural language documents; another, the *specific part*, addresses implementation-related issues by providing further WSDL descriptors complete with binding specifications. Once both parties retrieve the corresponding pre-existing Service Agreement, the requested service is made available for consumption.

2 Semantics in the SPCoop Framework

Since semi-automation and discovery are among SPCoop's key goals in a vast, heterogeneous and, to a certain extent, open world as is the Italian Public Administration, the injection of a solid yet essential semantic component into its building blocks is paramount. To achieve this, SPCoop proposes a method for defining a semantic layer on top of its service infrastructure, which comprises two components described as follows:

1. a shared knowledge base which all participating systems may query as well as contribute to, by registering custom semantic models describing their own application domains;

[2] Edited from [1], page 17

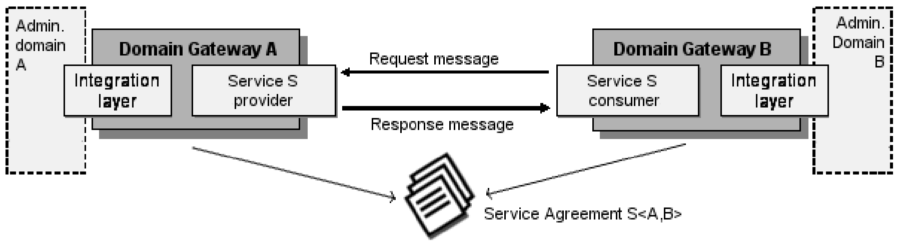


Fig. 1. Example of service exchange between administrative domains

2. a standard for the semantic annotation of Service Agreements, so that each of them includes appropriate references to existing semantic entities within the aforementioned knowledge base.

As shown on Figure 2[3], the first component is made available as a shared repository, named *Schema and Ontology Catalog*, whose functionalities are served by the same terms as any other application service provided by administrations that adhere to SPCoop. This repository is invisible to subjects that do not participate in the interoperability framework and it is designed to store mainly OWL ontologies and XML Schema definitions, yet in principle it may be adapted as to allow further knowledge representation formats such as UML class diagrams and so on. The Catalog, along with the Service Agreement registry mentioned in the introduction, is part of an infrastructural layer (shorted as SICA) that is essential for accessing the interoperability framework itself, and includes a number of other services regarding security levels, peer indexing and transaction monitoring.

It is worth noting that the principle of reusability also holds for the ontology and schema pool pertaining to the Catalog. Each participating administration is responsible for defining its operative context and is encouraged to author its own domain-specific ontology, or set thereof, and register it with the Catalog; however, it is not advisable that said ontologies be self-contained i.e. each redesigned from the ground up, whereas an alternative bottom-up approach that re-employs readily-available abstraction layers is generally preferred.

The annotation mechanism employed for Service Agreements is the SAWSDL standard, a W3C recommendation since June 2007 that is possibly the state-of-the-art method for semantic integration of Web Services. SAWSDL exploits the extensibility features of each WSDL syntactic element that is intended to convey information regarding the significance of the service being provided. Its range spans across those WSDL descriptors, and their type-defining XML Schemas, which are included solely in the common, reusable part of a Service

[3] Fuligni, S., *Use of the semantics for interoperability in the Italian Public Administrations*, 2008

Agreement. Of the three types of attributes designed for the SAWSDL specification, only `modelReference` is taken into consideration in the scope of SPCoop. Post-discovery data formatting issues, which would require the employment of the `liftingSchemaMapping` and `loweringSchemaMapping` attributes (used for matching the semantic model with the structure of input and output), are not expected in this context.

An additional requirement for SPCoop is the ability to introduce cyclic semantic references within the Catalog, in that conceptual XML Schemas can be retrieved from the Catalog and annotated *per se* using the SAWSDL indications for annotating XML Schema documents. This feature can then be exploited to support stand-alone annotation for XML Schema documents that are imported by WSDL descriptors as message type definitions.

Only the knowledge which is stored in the Schema and Ontology catalog is eligible for annotating WSDL 1.1 documents from the common part of Service Agreements, therefore an additional requirement is that the set of model references in a Service Agreement be consistent with the Catalog. All resource identifiers within model reference values must, when checked against the Catalog, point to a semantic entity that can be resolved by the Catalog itself, given that its inner structure maps namespaces to physical URIs that can be resolved to a resource containing the definition of the required concept.

Though it may often be the case that the individual(s) entrusted with annotating the Service Agreements of a domain coincide with the same ontology engineer(s) who modelled its knowledge in the first place, this condition is not guaranteed to always hold. For one thing, semantic annotation exclusively occurs in that phase of the service's life-cycle where its Agreement is published, whereas the formalisation of the semantics for that domain may have been previously performed by a specialised operator. Also, in scenarios where a service is intended for reuse by a multitude of providers, the corresponding Service Agreement may be managed by a neutral third party that does not serve as an endpoint for the service itself. Thus, it is not possible to presume any degree of competence in ontology engineering for the end-user who is given the task of annotating a service description, though we can make reasonable assumptions about them being sufficiently knowledgeable with regard to common modelling standards, like the Entity-Relationship paradigm and UML class and use case modelling which, compounded with reasonably-presumed domain knowledge, experience and common sense, makes them eligible for this task. It was therefore necessary to present end-users with a tool which enabled them to efficiently perform this task whilst hiding all the underlying complex logic from them.

Furthermore, the need for such an application was triggered by the lack of a proper third-party support for the 1.1 version of the WSDL specification, which was adopted by SPCoop long before version 2.0 became a W3C recommendation and cannot be upgraded due to institutional constraints. Although some tools and APIs like *Woden* are scheduled to fully support for WSDL 1.1 in the future, most existing tools currently offer exclusive support for those elements that are common to both versions of the specification.

The SAscha tool was developed to address these issues, and is offered as a Rich Internet Application with an option to deploy it as an infrastructure-supportive service on the Web available anytime, anywhere.

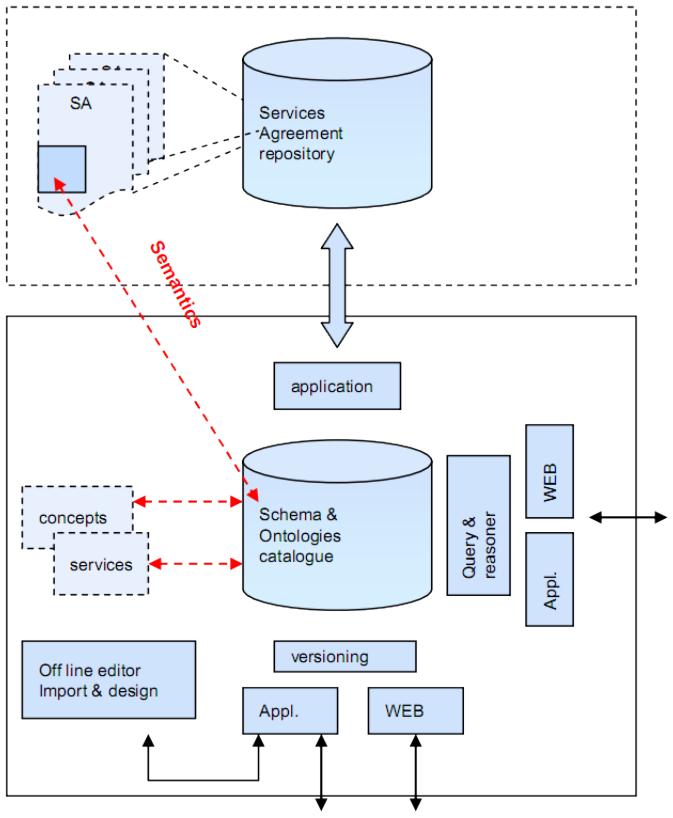

SA
Services Agreement repository
Semantics
application
concepts
services
Schema & Ontologies catalogue
Query & reasoner
WEB
Appl.
Off line editor Import & design
versioning
Appl.
WEB

Fig. 2. Overview of semantic integration mechanisms in SPCoop

3 The SAscha Component Model

Despite being presented as an AJAX application, SAscha, its two initials indefinitely standing for Semantic Annotation or Service Agreement, is entirely written in Java. Its UI component, remote interfaces and transport layer are compiled into AJAX-compliant markup and scripts through the *Google Web Toolkit*, whereas the engine itself can be run through any Web Container with servlet support (Tomcat, JBoss...). This approach entails loose coupling between user interface and application logic, as depicted on the diagram in Figure 3. The server-side APIs which are currently utilised for parsing and rendering resources back and forth are: the SourceForge *OWL API* for ontologies, IBM *WSDL4J* for WSDL documents and embedded schemas, a modified version of the LSDIS *SAWSDL4J* API, plus a custom add-on to accommodate the extra SPCoop requirement for supporting stand-alone XML Schema documents, i.e. not imported by a WSDL descriptor. The OWL API, in turn, is a flexible library that allows plugging-in of several reasoning engines like Pellet and FaCT++.

Because Google Web Toolkit's Remote Procedure Call API enforces certain restrictions on the Java objects that can be passed across a RPC, it is not possible to directly exploit object models from the aforementioned APIs, therefore we provide an intermediate object model which satisfies GWT serialisation policies and at the same time conveys the essential information to allow all resources, be they ontologies, schemas or Web Service descriptors, to be presented to the user in a convenient manner. Transformations between intermediate and server-side object models are performed by three specialised Java objects called *engines*. Annotations can then be merged to an XML file whose physical location is then passed as an HTTP header for the client to download.

Currently, the system is fed its resources by means of resolving physical URIs or by the user's local file system with respect to ontologies, and only by the local file system with respect to Service Agreement components and stand-alone XML Schemas. As the availability of infrastructural services for querying the Schema and Ontology Catalog closes in, this scenario is bound for change in short time.

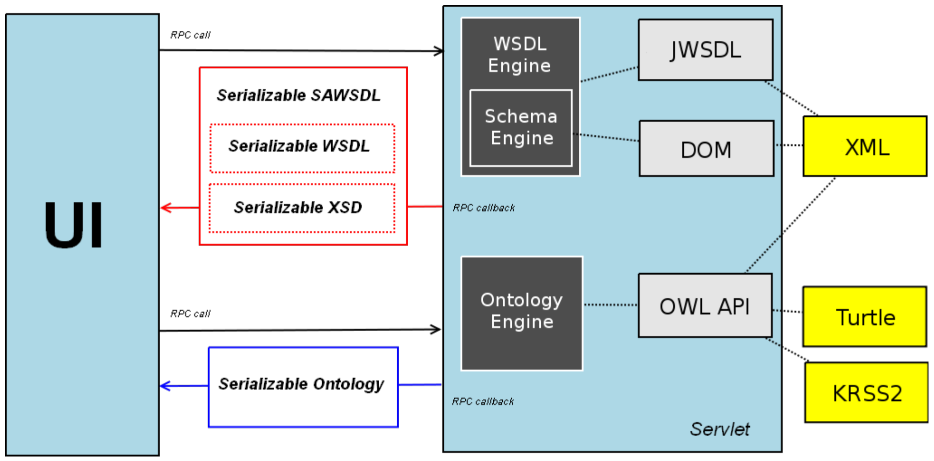


Fig. 3. Information flow between user interface and application engine in SAscha

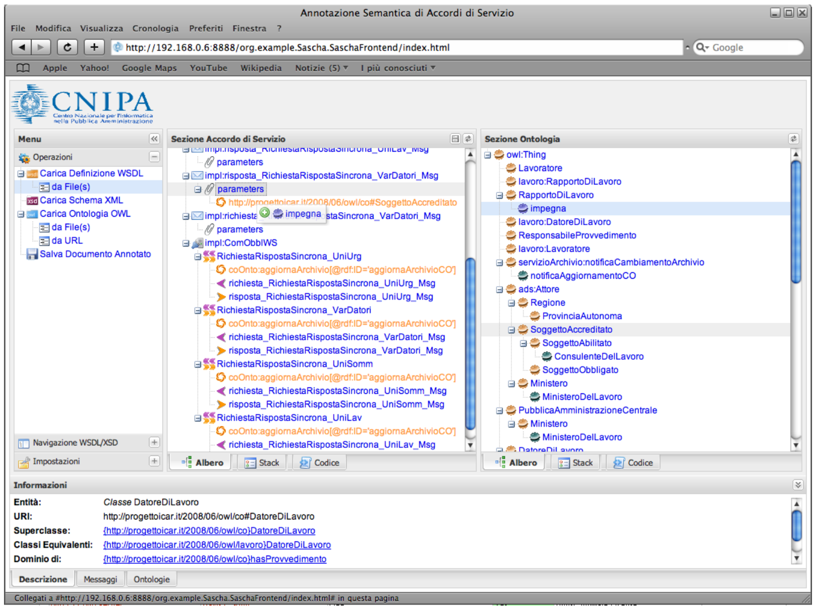
4 User Perspective

From an end-user standpoint, the SAscha tool offers a comparative, SAWSDL-versus-Ontology interface, displaying annotation sources i.e. ontologies on the right side, and annotation targets i.e. WSDL descriptors and XML Schemas on the left side. One key factor for the tool to appeal to end-users with reasonable modelling notions but litte to no knowledge of the Semantic Web is to closely follow the principle of least astonishment in terms of resource presentation. In accordance to such principles, SAscha tends to minimise conflicts by providing the same visualisation paradigms for both the annotation source and target. Each resource type can be viewed as a tree structure, a stack panel whose components are tables where information is grouped per item type (e.g. classes, properties and individuals for ontologies; messages, port types and operations for WSDL and so on), and a plain source code view. Annotations can be performed through mechanisms following a drag-and-drop metaphor, although the ability to directly edit the target source code is currently being considered. Users can highlight one or more concepts from the ontology tree or stack view and drag them to the node representing the item they wish to extend with a `modelReference` attribute, optionally binding an arbitrary prefix to each namespace. In a situation where such an attribute comprehends more URI references, each of them is treated as a tree node *per se*. These nodes can then be removed through a context menu action if need be.

Imports and external references are supported for both the annotation source and target. Typically, these imports are automatically resolved based on their declared physical locations, but users are given the ability to override such references by providing those resource themselves. Upon resolution failure, for each reference that was not resolved, users will be prompted for a manual override, which they might want to ignore, in which case the parsing/reasoning process will continue without. With respect to ontologies, users may also specify whether reasoning should be performed only on the active ontology, or the whole transitive imports closure, or even a given set of unrelated ontologies.

Tree-like representations for WSDL and XSD documents are quite straightforward, in that they mimic the nested structure from the corresponding XML syntax, with the exception of strictly implementation-related WSDL constructs (`binding`, `service` and `port` being the most frequent) which do not belong to the common portion of Service Agreements. As for ontology presentation, a number of factors were considered regarding which elements should be depicted as nodes. As with most common ontology authoring tools, the base for tree representation is the class hierarchy, as inferred by a server-side reasoning API. However, it is desirable to append individuals and properties to the very same class hierarchy, so as to offer a single view spanning across most of the concepts that can serve as SAWSDL annotation sources. In the light of this desired feature, individuals are appended as children of those class nodes which they are instances of. Datatype and object properties are also appended to nodes representing classes which feature them in their property domain declarations, with missing properties still being available in the corresponding tables on the stack view. Other information that does not fit the chosen representation is displayed in a collapsible diagnostics panel located at the lower portion of the UI.

These choices were the result of a number of technical and interaction-related considerations, which take into account the limitations of common JavaScript engines as well as the end-user impact of a potential overly redundant view. Providing the user with an option to prune both trees according to some degree of granularity is currently being considered. An insight to the SAscha client application running in a Web browser is given in Figure 4.


Annotazione Semantica di Accordi di Servizio
File Modifica Visualizza Cronologia Preferiti Finestra ?
http://192.168.0.6:8888/org.example.Sascha.SaschaFrontend/index.html
Google
Apple Yahoo! Google Maps YouTube Wikipedia Notizie (5) I più conosciuti
CNIPA
Centro Nazionale per l'Informatica nella Pubblica Amministrazione
Menu
Operazioni
Carica Definizione WSDL
da File(s)
Carica Schema XML
Carica Ontologia OWL
da File(s)
da URL
Salva Documento Annotato
Navigazione WSDL/XSD
Impostazioni
Sezione Accordo di Servizio
parameters
impl:risposta_RichiestaRispostaSincrona_VarDatori_Msg
parameters
http://progettoicar.it/2008/06/owl/co#SoggettoAccreditato
impl:richiesta
impegna
staSincrona_VarDatori_Msg
parameters
impl:ComObblWS
RichiestaRispostaSincrona_UniUrg
coOnto:aggiornaArchivio[@rdf:ID='aggiornaArchivioCO']
richiesta_RichiestaRispostaSincrona_UniUrg_Msg
risposta_RichiestaRispostaSincrona_UniUrg_Msg
RichiestaRispostaSincrona_VarDatori
coOnto:aggiornaArchivio[@rdf:ID='aggiornaArchivioCO']
richiesta_RichiestaRispostaSincrona_VarDatori_Msg
risposta_RichiestaRispostaSincrona_VarDatori_Msg
RichiestaRispostaSincrona_UniSomm
coOnto:aggiornaArchivio[@rdf:ID='aggiornaArchivioCO']
richiesta_RichiestaRispostaSincrona_UniSomm_Msg
risposta_RichiestaRispostaSincrona_UniSomm_Msg
RichiestaRispostaSincrona_UniLav
coOnto:aggiornaArchivio[@rdf:ID='aggiornaArchivioCO']
richiesta_RichiestaRispostaSincrona_UniLav_Msg
Albero Stack Codice
Sezione Ontologia
owl:Thing
Lavoratore
lavoro:RapportoDiLavoro
RapportoDiLavoro
impegna
lavoro:DatoreDiLavoro
ResponsabileProvvedimento
lavoro:Lavoratore
servizioArchivio:notificaCambiamentoArchivio
notificaAggiornamentoCO
ads:Attore
Regione
ProvinciaAutonoma
SoggettoAccreditato
SoggettoAbilitato
ConsulenteDelLavoro
SoggettoObbligato
Ministero
MinisteroDelLavoro
PubblicaAmministrazioneCentrale
Ministero
MinisteroDelLavoro
Albero Stack Codice
Informazioni
Descrizione Messaggi Ontologie

Entità:	*Classe* DatoreDiLavoro
URI:	http://progettoicar.it/2008/06/owl/co#DatoreDiLavoro
Superclasse:	{http://progettoicar.it/2008/06/owl/co}DatoreDiLavoro
Classi Equivalenti:	{http://progettoicar.it/2008/06/owl/lavoro}DatoreDiLavoro
Dominio di:	{http://progettoicar.it/2008/06/owl/co}hasProvvedimento

Collegati a #http://192.168.0.6:8888/org.example.Sascha.SaschaFrontend/index.html# in questa pagina

Fig. 4. Screenshot of the SAscha application running in Safari

5 Related Work

When mentioning semantics in Rich Internet Applications, a prior distinction is needed as to which phases of the approach involve the exploitation of semantic potential in the course of development. One approach is to promote the creation of semantic relationships between components of the software system itself, no matter what the ultimate task of the application. This technique enhances modelling, deployment and integration, yet at the time of writing, no practical approach regarding rapid RIA application development offers semantics at modelling time. On the other hand, there is a variety of hosted-mode tools which are focused in content towards the Semantic Web, employing technologies such as AJAX, OpenLaszlo, Silverlight and Flash. One good example is the media monitoring service *filtrbox*, a RIA developed in Adobe Flex for smart syndication, ranking and noise control over news, blogs, RSS and so on. As for ontology authoring, a prominent example is offered by the comprehensive semantic toolkit *TopBraid live*, combining Flex 2 technology with the strong-typed and object-oriented model offered by ActionScript 3.

On the side of existing Semantic Web Services tools there is none, to our knowledge, that supports all standard-specific requirements that Web Service descriptors must meet for the corresponding services to operate in SPCoop. Some systems are projected towards standards which could be seen as successors to the original, SAWSDL-enhanced WSDL 1.1 specification. Others opt for separate views with respect to WSDL outlines and ontology graphs, which is an acceptable, even desirable feature from a conceptual standpoint, but falls short of immediacy and does not lend itself to comparative decision support. For comparison, we have put aside Web Service discovery tools like *Lumina* and focused on state-of-the-art annotation technologies, further elaborating on the implementation report as in [5]. The LSDIS *Radiant* plugin for Eclipse, *WSMO Studio* and *Semantic Tools for Web Services* by IBM alphaWorks were picked as terms of comparison by which SAscha was measured:

Table 1. Comparison of semantic Web Service annotation tools

	SAWSDL Model References	**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 alphaWorks Semantic Tools for Web Services use annotations in Web Services Semantics (WSDL-S) format and were not updated to support its successor SAWSDL. As for the partial WSDL 1.1 support offered by Radiant and WSMO Studio, this is limited to those constructs that are common to both 1.1 and 2.0 versions, with WSMO offering additional support for the `attrExtensions` element, therefore WSDL 1.1 message parts are not eligible for annotation by either systems. Finally, only Radiant seemed to feature a common view for representing and interacting with annotation source and target - what is called “Shared UI Perspective” in the table.

6 Conclusion and Future Work

Despite still being in a pre-alpha status and covering a set of niche requirements for the Italian e-Government platform exclusively, several development plans are being taken into account. The adaptation of SAscha so that it would take advantage of all the features and services provided by the SPCoop framework, with regard to Schema and Ontology Catalog query services, was somehow partially hindered by the unavailability of such services and specifications at the time of development, and is planned for as soon as these services are up and running. A client-side and server-side extension to support the extraction, navigation and repackaging of Service Agreements (which essentially comprises two DEFLATE-compressed archives, one for each part) is also being developed.

Aside from the aforementioned SPCoop-specific issues, we have classified further development scenarios as follows:

1. Decision support, which can be provided by comparing SAWSDL annotations against a given ontology pattern and suggesting its best matches with the active ontology, if any;
2. Adding support for schema mappings;
3. Full WSDL 1.1 and 2.0 support, including non-normative semantic annotation e.g. on `binding` elements;
4. Enabling natural language comments on WSDL constructs by means of the `wsdl:documentation` node (currently under development).

From a general-purpose perspective, forking the application into two separate projects, one for full support of context-independent SAWSDL chores and one scaled down for SPCoop specifications, is a viable option. However, it is most likely that the public release will be a single package available under a permissive free-software license such as LGPL. This package would be stripped of its SPCoop-specific funtionalities, which may then be made available as separate modules.

References

[1] Armenia, S., Baldoni, R., Fuligni, S., Mecella, M., Raia, A., Tortorelli, F. : Sistema pubblico di cooperazione: Quadro Tecnico d'Insieme (2005)

[2] Erl, T. : SOA Principles: An introduction to the Service-Orientation paradigm (2005-2006)

[3] Fuligni, S., Gangemi, A., Tortorelli, F. et al. : Le ontologie come infrastruttura concettuale per lo sviluppo dei servizi assistiti da tecnologie semantiche nella Pubblica Amministrazione (2008)

[4] Sheth, A., Verma, K. : Semantically Annotating a Web Service (2007)

[5] SAWSDL Candidate Recommendation Implementation Report (2007) http://www.w3.org/2002/ws/sawsdl/CR/

[6] Radiant http://lsdis.cs.uga.edu/projects/meteor-s/downloads/index.php?page=1

[7] WSMO Studio http://www.wsmostudio.org

[8] Semantic Tools for Web Services http://www.alphaworks.ibm.com/tech/wssem

[9] Baldoni, R., Fuligni, S., Mecella, M., Tortorelli, F. : The Italian e-Government Service-Oriented Architecture. Strategic Vision and Technical Solutions (2007)

[10] Baldoni, R., Fuligni, S., Mecella, M., Tortorelli, F. : The Italian e-Government Enterprise Architecture: A Comprehensive Introduction with Focus on the SLA Issue (2008)

[11] Balkić, Z., Pešut, M., Jović, F. : Semantic Rich Internet Application (RIA) Modeling, Deployment and Integration (2007)

[12] Adamou, A. : Una Rich Internet Application per l'annotazione degli Accordi di Servizio nel Sistema Pubblico di Cooperazione (2008)