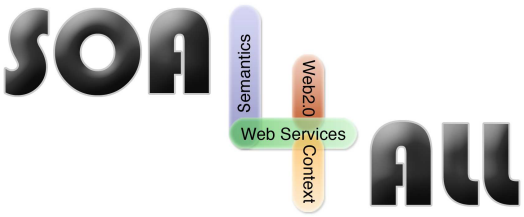

Project Number: **215219**

Project Acronym: **SOA4All**

Project Title: **Service Oriented Architectures for All**

Instrument: **Integrated Project**

Thematic Priority: **Information and Communication Technologies**

D2.1.2 Service Modelling Tools Design

Activity:	Activity 1 - Fundamental & Integration Activities	
Work Package:	WP2 – SOA4All Studio	
Due Date:		M6
Submission Date:		29/08/2008 Resubmission: 12/03/2009
Start Date of Project:		01/03/2008
Duration of Project:		36 Months
Organisation Responsible of Deliverable:		iSOCO
Revision:		2.6
Authors:	Guillermo Álvaro Rey iSOCO Luchesar Cekov SIRMA Nikolay Mehandjiev UNIMAN Lai Xu SAP Sven Abels TIE Juergen Vogel SAP Alex Simov SIRMA Maria Maleshkova OU	
Reviewers:	Jean-Pierre Lorre EBM WS Bernhard Schreder HANIVAL	

Project co-funded by the European Commission within the Seventh Framework Programme (2007-2013)		
Dissemination Level		
PU	Public	**X**

Version History

Version	Date	Comments, Changes, Status	Authors, contributors, reviewers
0.1	27/05/2008	First proposal of ToC	Guillermo Álvaro Rey
0.2	24/06/2008	Initial draft version with contributions for 2nd Plenary Meeting (Nice)	Luchesar Cekov, Guillermo Álvaro Rey
0.3	08/07/2008	Merged additional changes in new template	Guillermo Álvaro Rey, Luchesar Cekov, Nikolay Mehandjiev
0.4	24/07/2008	New contributions	Guillermo Álvaro Rey, Lai Xu
0.5	04/08/2008	First complete version	Guillermo Álvaro Rey, Luchesar Cekov, Nikolay Mehandjiev, Lai Xu, Sven Abels
0.6	06/08/2008	General corrections	Guillermo Álvaro Rey, Luchesar Cekov, Nikolay Mehandjiev, Lai Xu, Sven Abels
1.0	08/08/2008	Version for internal reviewers	Guillermo Álvaro Rey, Luchesar Cekov, Nikolay Mehandjiev, Lai Xu, Sven Abels
1.1	26/08/2008	Addressed first issues alter revisions by Matteo Villa and Carlos Pedrinaci.	Guillermo Álvaro Rey, Matteo Villa, Carlos Pedrinaci
1.2	29/08/2008	More issues corrected. Preliminary draft.	Guillermo Álvaro Rey, Luchesar Cekov, Nikolay Mehandjiev, Lai Xu, Sven Abels, Juergen Vogel
1.3	12/09/2008	Further corrections	Guillermo Álvaro Rey, Luchesar Cekov, Nikolay Mehandjiev, Lai Xu, Sven Abels, Juergen Vogel
2.1	12/01/2009	Proposed skeleton for resubmission	Guillermo Álvaro Rey
2.5	11/02/2009	Contributions	Alex Simov, Maria Maleshkova, Guillermo Álvaro Rey

-	23/02/2009	Internal Review	Reviewer: Jean-Pierre Lorre (EBM WS)
-	23/02/2009	Internal Review	Reviewer: Bernhard Schreder (HANIVAL)
2.6	27/02/2009	Addressed comments by reviewers	Guillermo Álvaro Rey, Alex Simov, Maria Maleshkova
Final	11/03/2009	Overall format and quality revision	Malena Donato (ATOS)

Table of Contents

List of Figures and Tables

List of Figures

List of Tables

Glossary of Acronyms

Acronym	Definition
AJAX	Asynchronous JavaScript And XML
API	Application Programming Interface
BPEL	Business Process Execution Language
BPM	Business Process Management
BPMN	Business Process Modeling Notation
D	Deliverable
DRM	Digital Rights Management
DSB	Distributed Service Bus
EC	European Commission
GWT	Google Web Toolkit
GUI	Graphical User Interface
IPR	Intellectual Property Rights
JMX	Java Management eXtensions
HTTP	HyperText Transfer Protocol
OKBC	Open Knowledge Base Connectivity
QoS	Quality of Service
RIA	Rich Internet Application
RDF	Resource Definition Framework
REST	Representational State Transfer
SaaS	Software as a service
SAWSDL	Semantic Annotations for WSDL and XML Schema
SCA	Service Component Architecture
SEE	Semantic Execution Environment
SOA	Service Oriented Architecture
SOAP	Simple Object Access Protocol
STP	SOA Tools Platform
SWS	Semantic Web Service
T	Task
UI	User Interface
URI	Uniform Resource Identifier
W3C	World Wide Web Consortium
WP	Work Package
WS	Web Services

Acronym	Definition
WSDL	Web Services Description Language
WSML	Web Service Modeling Language
WSMO	Web Service Modeling Ontology
WSMX	Web Service Execution Environment
WWW	World Wide Web
XML	eXtended Markup Language
XPDL	XML Process Definition Language

Executive summary

The present deliverable complements and is heavily interrelated with deliverable D2.1.1, where the Service Provisioning Platform is described. We focus here on the tools that we will use in order to semantically annotate services, grouped in the so-called “Simple Semantic Web Services Editing Framework”.

The main outcomes of this framework are two editors that will allow users to enrich traditional WSDL-based and RESTful services with semantic annotations: The WSMO-Lite Editor and the MicroWSMO Editor, respectively.

We have identified several characteristics that these tools will need to have in order to achieve the ambitious objective of the project of having an extremely large number of services available for consumption. Concretely, our tools will be lightweight and web-based, useable by both expert and non-expert users, and enabling a community approach towards modelling. We consider these characteristics are key for reaching a scenario of many services being deployed.

The detailed design of these tools is then provided, highlighting the functionalities they will cover and the interactions with other architectural components of SOA4All.

It is worth noting that the current deliverable does not address the Composite Semantic Web Services Editing Framework, which is treated separately by another task (T2.6, SOA4All Process Editor).

1. Introduction

This deliverable covers the design of the Service Modelling Tools that will be used in SOA4All in order to enrich service descriptions with semantic annotations. These tools are necessary in order to enable a world where billions of services are available, which is one of the main objectives of the project. Hence, the characteristics of these tools will be focused on the idea that they have to promote the creation of an extremely large number of services.

It is worth noting the close relationship of the tools described herein with the Service Provisioning Platform to which they belong, described in D2.1.1 [23]. In this deliverable we explicitly address the two main outcomes of the Platform, i.e., the WSMO-Lite Editor and the MicroWSMO Editor, which are the tools that will be used to semantically annotate traditional WSDL-based [2] services and RESTful [10] services, respectively.

1.1 Purpose and Scope

Creating semantic descriptions of services, as envisaged by SOA4All, should be a lightweight process available for all different kinds of users, in contrast to the previously more convoluted process that implied applications that were only accessible by experts. Besides, this process should also permit the gathering of information from communities of users, resulting in a collaborative modelling and annotating of services.

The scope of the Service Modelling Tools addressed in this deliverable is to enable the kind of lightweight modelling envisioned by the project. These tools will cover two new levels of semantics that come from WSMO [4], namely WSMO Lite [8] and MicroWSMO [9], as each level of semantics is foreseen to require different ways to capture and represent knowledge.

The purpose of this deliverable is to present the major design characteristics of the WSMO-Lite Editor and the MicroWSMO Editor, in order to allow their development in next steps of the project.

1.2 Structure of the document

This document is structured as follows:

- This section addresses the purpose and scope, and structure of the document. Additionally we put the deliverable in context with the rest of the architectural components of the project and in relation to the Use Cases.
- Section 2 provides a short review on the State of the Art in different flavours of WSMO, and Service Annotation tools.
- Section 3 gives an overview of our vision on the tools to be developed.
- Section 4 describes the design of the tools, addressing the functionalities of the Simple SWS Editing Framework and its two major outcomes: The WSMO-Lite Editor and the MicroWSMO Editor.
- Finally, Section 5 collects the main conclusions of this document.

1.3 Deliverable relation with the architecture of the project

The Service Modelling Tools addressed in this deliverable are a very important outcome of the Service Provisioning Platform, which, in turn, is part of the SOA4All Studio. In a nutshell, the SOA4All Studio is the gateway for the user to SOA4All, as depicted in Figure 1. The tools described in this deliverable are thus the entry point for SOA4All users when enriching service descriptions with semantic annotations.

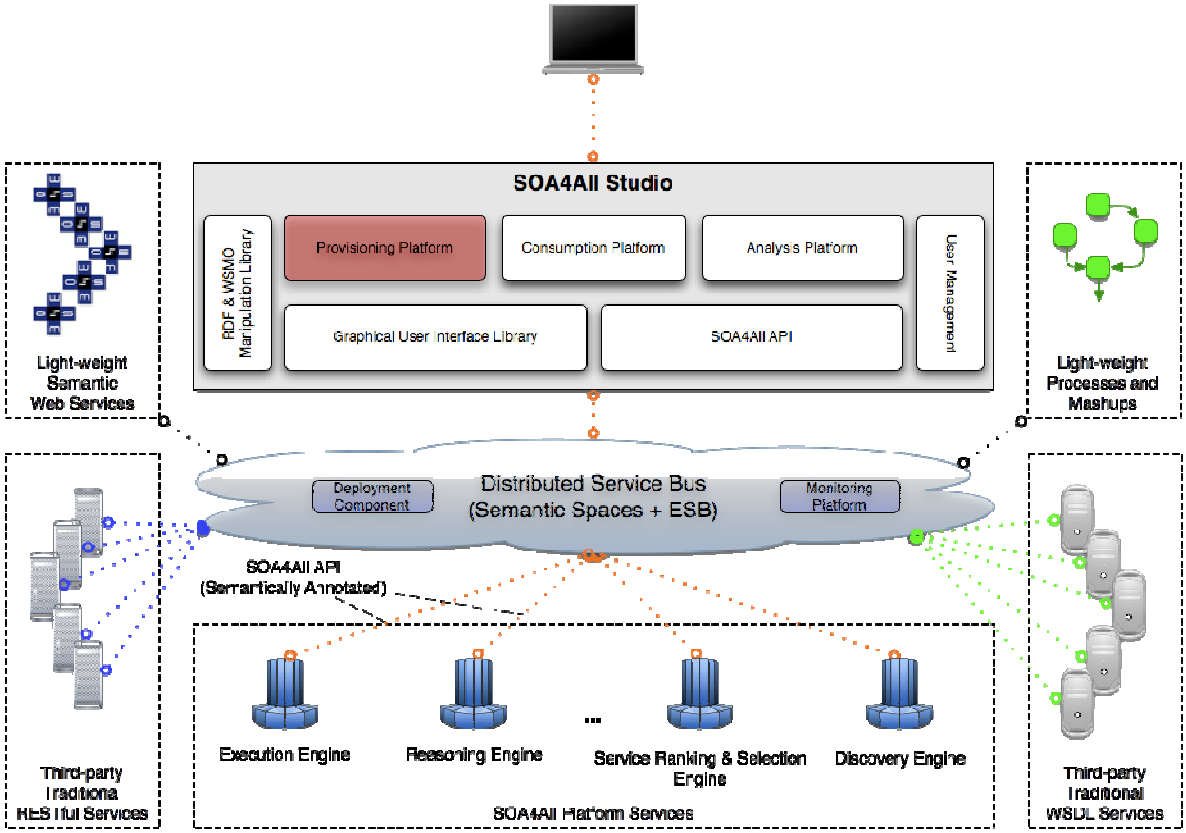

Figure 1: SOA4All architecture overview

Being the Service Provisioning Platform part of the SOA4All Studio, it will make use of the Infrastructure Services and UI Components that the SOA4All Studio will provide [31]. These characteristics are treated with more detail in the Service Provisioning Platform deliverable [23].

Regarding the tools to be covered by this deliverable, we will focus on the Simple SWS Editing Framework. It is worth noting that the Composite SWS Editing Framework is addressed in a different task (T2.6 *SOA4All Process Editor*, [32]), while we cover here the enrichment of service descriptions with semantic annotations. The two major outcomes in this sense are (i) the WSMO-Lite Editor, which enables the semantic annotations over WSDL Services, and (ii) the MicroWSMO Editor, which permits annotating RESTful services.

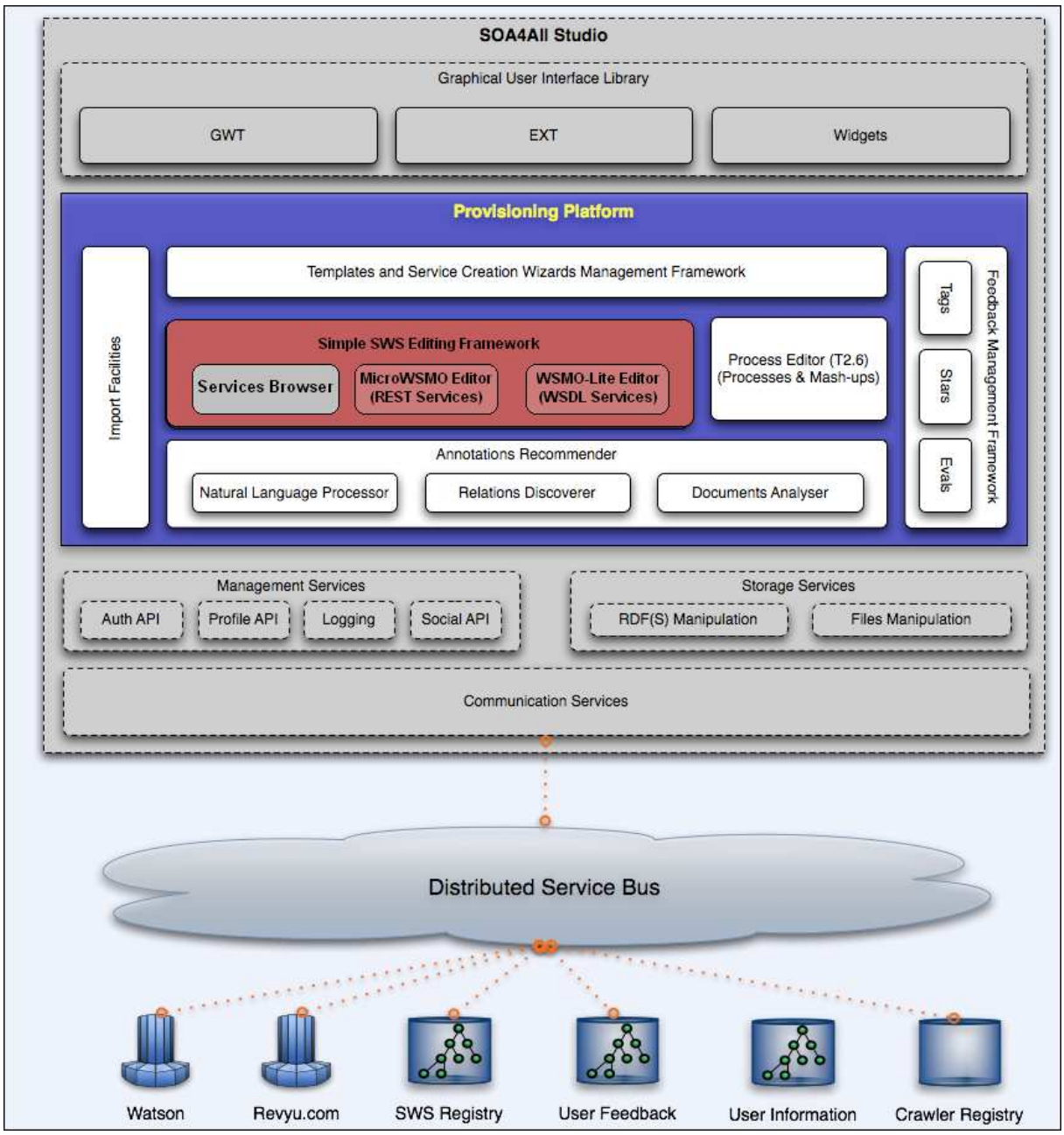

Figure 2: Service Provisioning Platform architecture overview

Figure 2 shows the Simple SWS Editing Framework with its two Editors (note that the Services Browser component is addressed in D2.1.1 [23], Section 4.3.1), in the context of the Service Provisioning Platform. It is important to point out the presence of the Framework on top of the Annotations Recommender, which will be used in order to support the process of adding annotations.

1.4 Deliverable relation with the use cases

Due to the strong interrelation between this deliverable and the one that addresses the design of the whole platform, we refer here to deliverable D2.1.1 [23], Section 2.3, “Alignment with the use cases”, where the relation of the three use case work packages with the Service Provisioning Platform, and tools such as the ones described herein, is explained.

However, we would like to highlight here the importance of the particular tools addressed in the present deliverable for the whole project. In order to further align the design of these, we

have asked the respective leaders of the case study work packages to identify the likelihood of these editors -as described in this deliverable- being used. Both editors were assigned at least a “mid” likelihood by each of the use case work packages, being particularly relevant the importance of the editor for WS-based services. We summarize in Table 1 these envisaged connections.

Use Case WP	WSMO-Lite Editor	MicroWSMO Editor
WP7	High	Mid
WP8	High	High
WP9	Sure	Mid

Table 1: Likelihood of the Editors being used by the Use Case WPs

2. State of the art

In this section, we provide a brief overview on the state of the art in two particular fields which are relevant for the deliverable. Particularly, we begin by covering the WSMO variants that will be covered by the tools, namely WSMO-Lite and MicroWSMO. We also address the state of the art in tools for the semantic annotation of services, as the purpose of the tools is to annotate existing services with semantic information to enable better and quicker discovery, orchestration and mediation.

2.1 WSMO Flavours

Although some efforts have been taken in the direction of enriching traditional Web Services (based in SOAP [1], WSDL [2] and UDDI [3]) with semantic annotations, like SAWSDL (Semantic Annotations for Web Services Description Language, [6]), and RESTful services with microformats, they are far from being a complete approach to fulfil our needs in SOA4All. Hence, for our purposes in the project, in order to make the desired lightweight modelling of services possible, the WSMO family of ontologies has expanded to cover two additional levels of semantics, addressed respectively by WSMO Lite (which is based in SAWSDL) and MicroWSMO (microformat to annotate RESTful [10] services), which will enable different methodologies and representations of services

Work package 3 (Service Annotation and Reasoning) will work on these two new WSMO variants, in order to satisfy the needs of the project. Outside SOA4All –but in close relation to this project– , the Conceptual Models for Services Working Group[1] (CMS WG) is leading the efforts of building the new WSMO variants on top of WSMO.

We will review now the main characteristics of the three different flavours of WSMO, putting them in relation to the particular requirements of our project. WSMO Lite and MicroWSMO will be thoroughly defined and developed in WP3, so it is not our intention to extensively cover them, but just to give an overview on what do we have to take into account related to these languages and concerning our tools that deal with them.

2.1.1 WSMO

WSMO (Web Services Modelling Ontology, [4]) is an ontology for describing various aspects related to Semantic Web services. It is based on the Web Service Modelling Framework (WSMF, [7]) and refines it through a formal ontology and language (WSML, Web Service Modelling Language [22]).

WSMO deals with four different main elements for describing semantic Web services, represented in Figure 3:

1. **Ontologies**, to provide the terminology used by other elements,
2. **Goals**, to state the intentions that should be solved by Web services,
3. **Web services**, to define functionalities and behaviour, and
4. **Mediators**, to resolve interoperability problems.

[1] http://cms-wg.sti2.org/home/

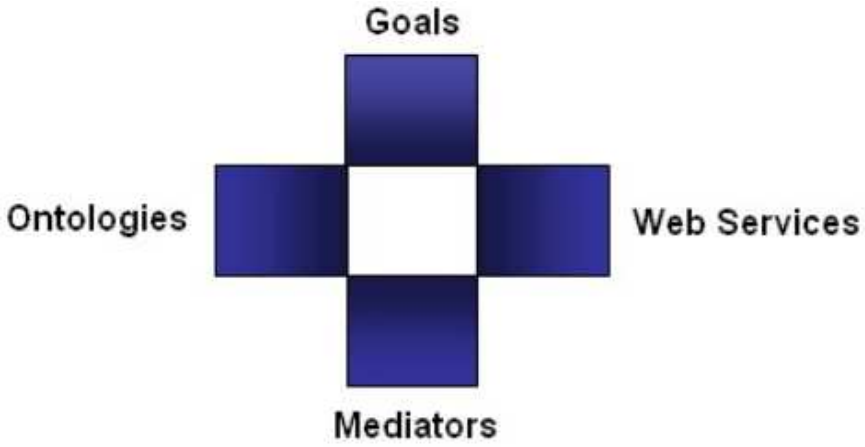

Figure 3: Top-level elements of WSMO

WSMO has already been successfully used during several EU funded projects such as DIP[2] (Data, Information, and Process Integration with Semantic Web Services) or SUPER[3] (Semantics Utilised for Process Management within and between Enterprises), and serves its purpose of enabling the modelling, execution and monitoring of Semantic Web services.

Regarding SOA4All, the main concerns related to WSMO in conjunction with enabling the creation of a web of billions of services are in terms of its weight and ease of use, which would not enable a scenario featuring a widespread creation of services. Consequently, the need for other lighter versions that could support a service revolution is explained.

2.1.2 WSMO-Lite

Service specifications that exist today formalised in WSDL describe their functionality and the way in which users can interact with those services. As the number of services exposing their interfaces in WSDL rises up, a proper automation will be essential to facilitate a reasonable service discovery. Thus, existing service specifications need to be augmented with semantic descriptions.

From the need of enriching service specifications, SAWSDL [6] came up, being a W3C Recommendation since 2007. It provides a bottom-up approach for service modelling, by supporting the idea of adding small increments on top of WSDL. Hence, SAWSDL is independent of any particular semantic technology, as it does not define any types, forms or languages for semantic descriptions.

WSMO-Lite is envisioned as the next evolutionary step after SAWSDL, filling the SAWSDL annotations with concrete semantic descriptions, and thus embodying the semantic layer of the Semantic Service Stack [8].

From the point of view of SOA4All, this version of WSMO will help us embrace in a lightweight fashion a great quantity of services. The modelling tools that we describe in this document will have to deal with this language, allowing users to easily enrich service specifications with semantic descriptions.

2.1.3 MicroWSMO

As we have already pointed out, WSMO Lite addresses the issues concerning the enrichment of traditional Web Services that expose their interfaces using WSDL, in the often

[2] http://dip.semanticweb.org/

[3] http://www.ip-super.org/

called WS-* set of specifications, which use the messaging paradigm and are mostly deployed within enterprises.

In contrast to this approach, and following a direction that relies on the architectural style of the World Wide Web, we can find REST (Representational State Transfer [10]) technologies, which consider Web services as sets of resources accessible through HTTP uniform interfaces.

MicroWSMO consists of a service ontology for RESTful Web services, and a method for annotating descriptions of them. In the context of SOA4All, MicroWSMO will be very important in order to embrace the large quantity of services that, within the Web 2.0, do not expose their interfaces in WSDL, e.g., mash-ups, gadgets, pipes, etc. MicroWSMO is a microformat that will enable a lightweight annotation of those resources, hence favouring the discovery of these kind of services.

Regarding our project, the fact that we will be semantically annotating RESTful services will be quite beneficial, as it will increase the chances of reaching a world of billions of services enormously, due to the fact that there is a very important growth in the number of RESTful services taking place [35].

2.2 Service semantic annotation

We cover here existing state of the art (desktop-based) tools that enable the semantic annotation of services. In SOA4All we need similar tools exposed like Web 2.0 user interface, able to deal with the new WSMO variants previously addressed.

2.2.1 Protégé

Protégé[4] (Figure 4) is a free, open-source platform that provides a growing user community with a suite of tools to construct domain models and knowledge-based applications with ontologies. At its core, Protégé implements a rich set of knowledge-modeling structures and actions that support the creation, visualization, and manipulation of ontologies in various representation formats. Protégé can be customized to provide domain-friendly support for creating knowledge models and entering data. Further, Protégé can be extended by way of a plugin architecture and a Java-based Application Programming Interface (API) for building knowledge-based tools and applications.

[4] http://protege.stanford.edu/

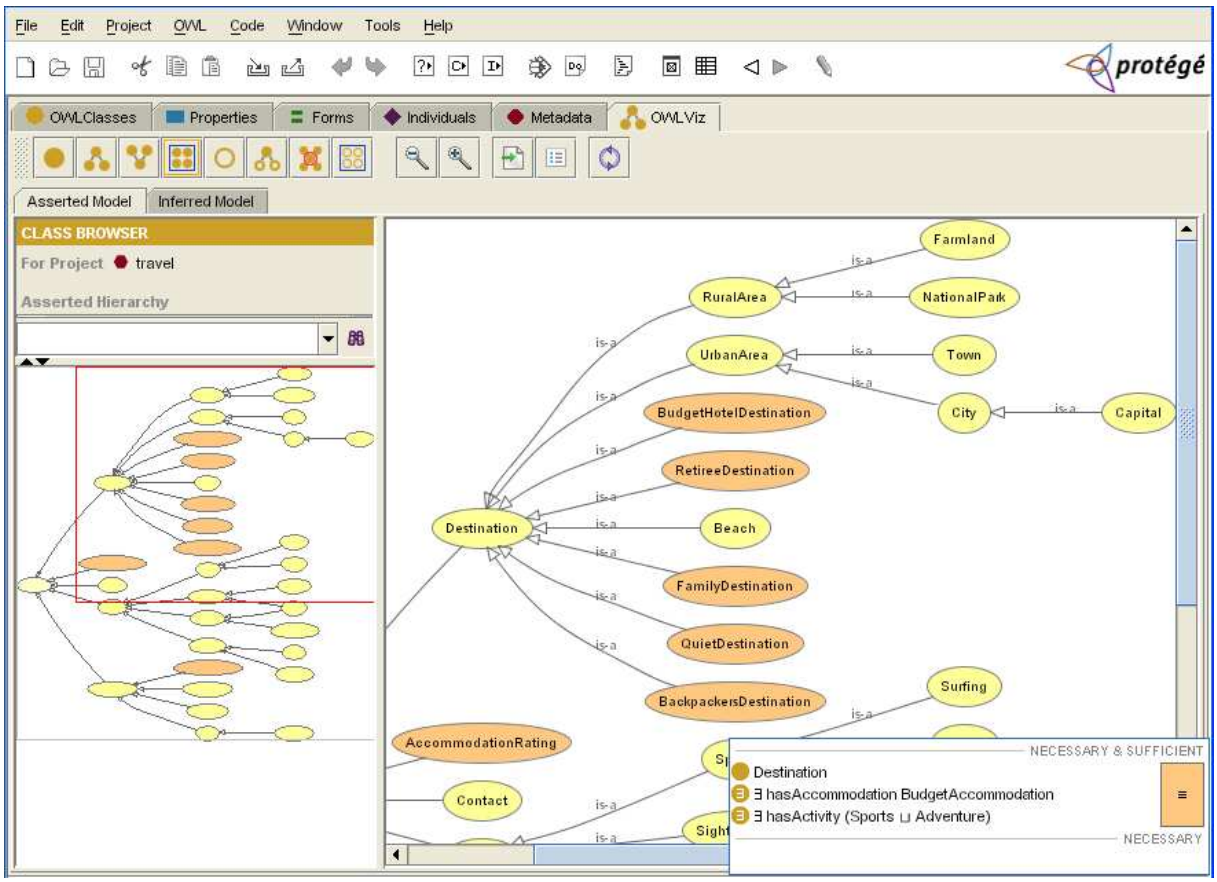

Figure 4: Protégé-OWL editor

The Protégé platform supports two main ways of modeling ontologies:

- The Protégé-Frames editor enables users to build and populate ontologies that are frame-based, in accordance with the Open Knowledge Base Connectivity protocol (OKBC) [20]. In this model, an ontology consists of a set of classes organized in a subsumption hierarchy to represent a domain's salient concepts, a set of slots associated to classes to describe their properties and relationships, and a set of instances of those classes - individual exemplars of the concepts that hold specific values for their properties.
- The Protégé-OWL editor enables users to build ontologies for the Semantic Web, in particular in the W3C's Web Ontology Language (OWL) [5].

2.2.2 WSMO Studio

WSMO Studio [19] (Figure 5) is an open source Semantic Web Service and Semantic Business Process modelling environment for the Web Service Modelling Ontology. WSMO Studio aims to providing a GUI tool that assists the users working in the WSMO domain with tasks related to semantic web service annotation. It is also an extensible tool and architecture that will allow third parties to integrate and extend WSMO Studio functionality.

WSMO Studio is an Eclipse-based application (and hence a desktop application as well), and supplies the following functionality:

- Creating WSMO descriptions of ontologies, goals, web services and mediators
- Export and import of the WSMO descriptions (supported languages and formats are WSML, WSML-XML and OWL-DL)

- Front-end to service, goal, mediator and ontology repositories (such as IRS-III [36] or WSMX [37])
- SAWSDL editor for adding semantic annotations to WSDL documents
- Front-end to service discovery components providing goal based semantic service discovery

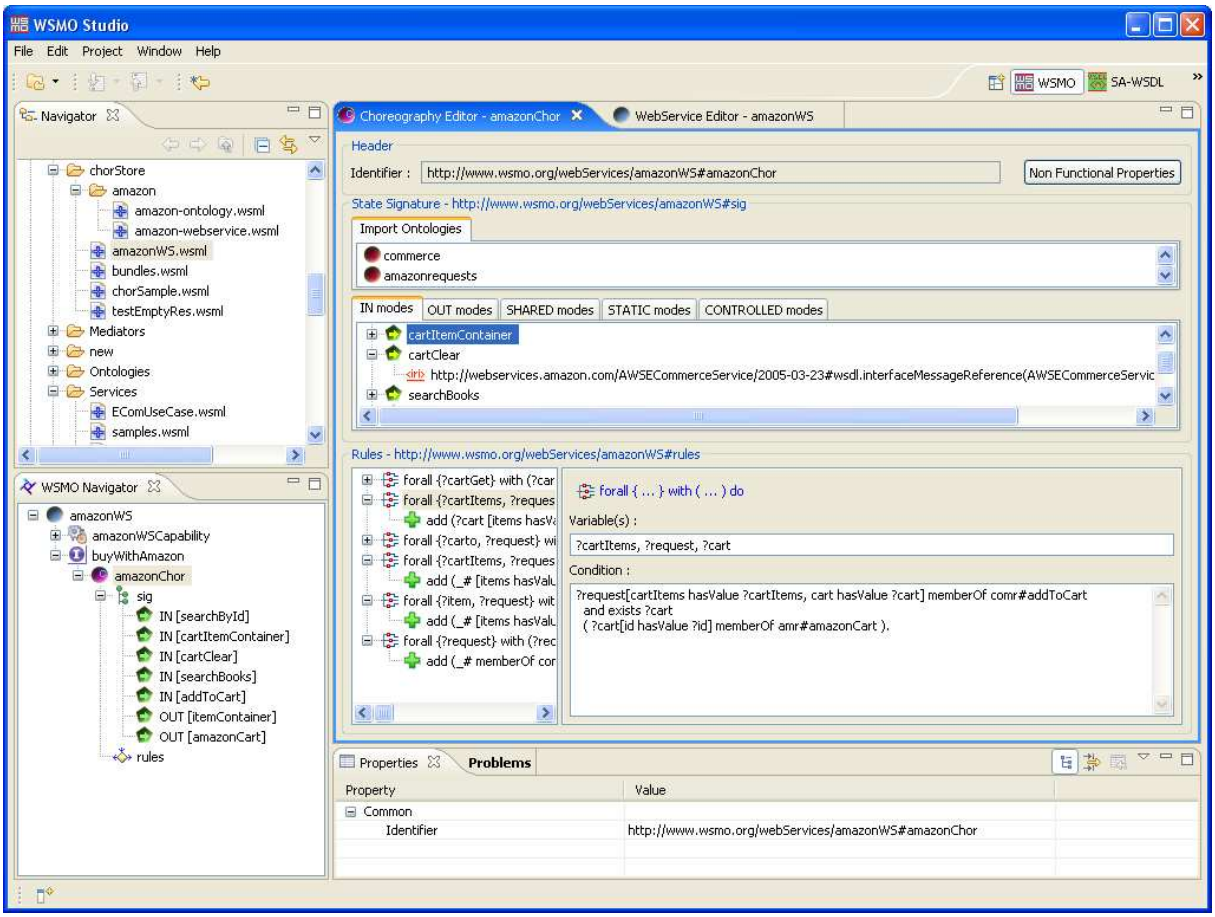

Figure 5: WSMO Studio SWS Choreography Editor

2.2.3 WSMT

The Web Service Modeling Toolkit[5] (WSMT) (Figure 6) is a collection of tools for Semantic Web Services intended for use with the Web Service Modeling Ontology (WSMO), the Web Service Modeling Language (WSML) and the Web Service Execution Environment (WSMX).

[5] http://sourceforge.net/projects/wsmt

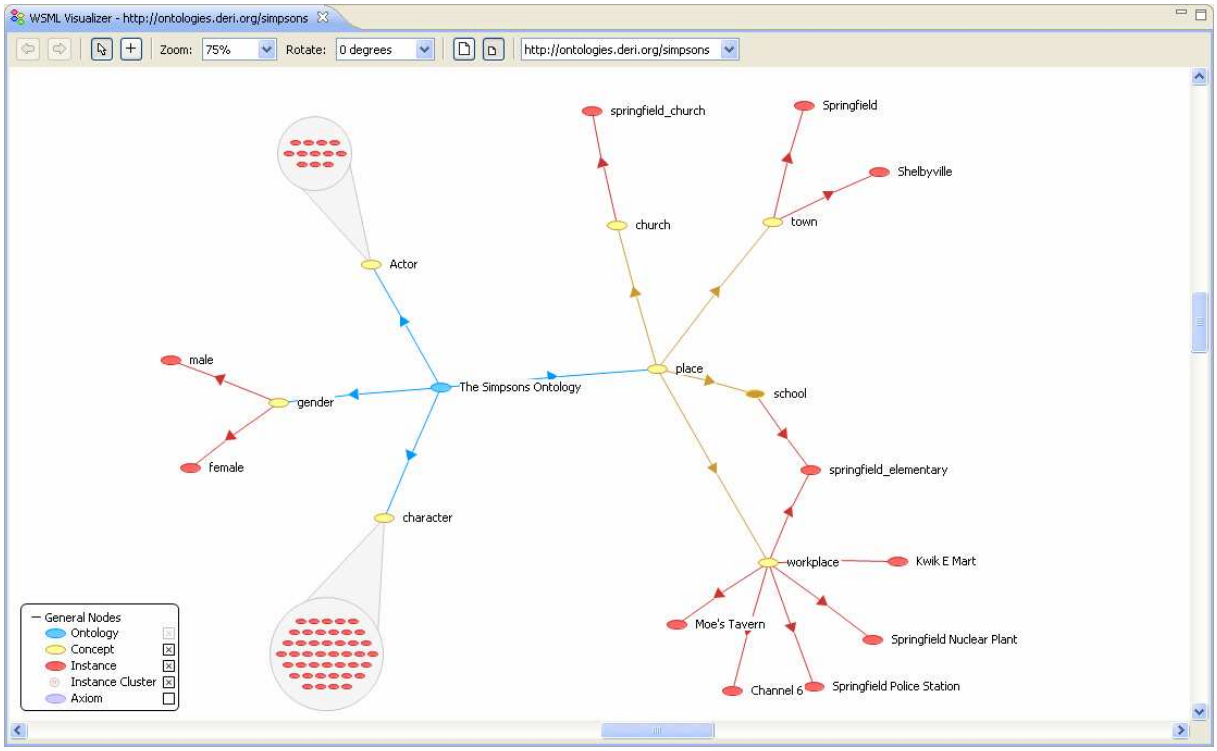

Figure 6: The Web Service Modeling Toolkit: WSML Visualizer

It is also Eclipse-based and therefore a desktop application. WSMT functionalities include:

- Collaboration with Semantic Execution Environment (SEE) [21]
- WSML visualizer
- WSML reasoner
- Semantic Highlighting
- WSMX Data Mediation Mapping Tool
- Management of WSMX via JMX

3. Service Modelling Tools: Overall Vision

Enriching Web Services with semantic annotations is a convoluted process that only experts in the field can perform. In order to fulfil the objective of SOA4All, “enabling the SOA revolution on a world-wide scale”, thus enabling to realize a web of services interconnecting billions of them, we need to enormously lighten the process of semantically annotating services. By developing efficient service modelling tools that many users will be able to deal with, we will be favouring the creation of a truly large number of semantic web services.

What SOA4All is promoting is a new service-modelling paradigm, based in the new versions of WSMO, where different kinds of users will be able to model the semantics of different types of services in a lightweight fashion. The scope of this deliverable is to design these tools so that they can really fulfil the requirements of a web-scale service revolution.

Furthermore, the actual paradigm involves expert users modelling services in desktop-based applications, but there is a need to enable different types of users, namely experts and non-experts, in the process of modelling services. In the service-modelling paradigm that we are promoting, the importance of the users will be highlighted by lowering the entry barrier in order to permit a straightforward semantic annotation of services, quite beneficial for our objectives in the project of enabling billions of them.

We will approach the definition of the characteristics of the tools by identifying trends in other fields that are relevant for our work. In particular, in the software sector, we have identified a trend towards offering software solutions more as a service, accessible via Internet, rather than as a product. This approach is the Software as a service (SaaS, [26]) model, where services are provided to customers across the Internet, and which is being widely adopted. In this line, Cusamano [27] points out the tendency among software vendors that are shifting from product revenues to embracing services. This actually means that software that allows users to perform different activities does not necessarily need to be installed in the user’s computer, but accessible by the web via a browser instead. Hence, SaaS alleviates the customer's burden of software maintenance, ongoing operation, and support.

This will be the case for our service modelling tools, which will not be a product that a user has to install, but a service itself to which the user will connect through the Internet. This is especially consistent with the way the information flows in the Internet nowadays, as we have noticed how distinction between providers and consumers has blurred so much that the term *prosumer* (coined by Alvin Toffler, [28]) is being increasingly used. In the Web2.0, users have no longer a passive role, but they are generating content in many different ways. Following the same approach as the one with content, within WP2 we want to promote a new role within the service world: the “*service prosumer*”, which is a central role that the users will play during the overall life-cycle of services (that covers provisioning [23], consumption [25] and analysis [24]). To make that possible, we will need to stress Web2.0 characteristics, such as different kinds of users being able to contribute new services in a community-driven fashion.

Bearing these aspects in mind, we can highlight the characteristics that the tools will have, in order to fulfil our requirements of the project:

- We will need **lightweight tools** ubiquitously available.
- Not only expert, but also **non-expert users** will be able to use these tools in a simple fashion.
- The new tools will enable a **community-oriented** approach to modelling.

The following Table 2 roughly depicts the actual situation of these characteristics in relation to service modelling, and the general tendencies that can be found in Internet today. In SOA4All, we will combine the power of service modelling with the advantages from the web tendency approach.

Characteristic	In Service Modelling	Web Tendency
Weight of tools (regarding the user)	Desktop-based heavyweight tools.	Web-based lightweight tools.
Type of users	Expert.	Expert and non-expert.
Individual vs. community	Individual.	Individual and community-based.

Table 2: Service modelling and web tendency characteristics

It is important to note that when we refer to the weight of the tools, we are doing it from the perspective of the user. In fact, while web-based applications appear to be lightweight for the user, they actually are heavier than their desktop counterparts, just considering the backend infrastructure required for dealing with a huge number of users, etc. However, what we are concerned about in this particular work package is with the experience of the user, so despite being complex applications with a strong backend, we consider them lightweight regarding the interactions of SOA4All users.

Additionally, when we talk about expert and non-expert users, we are of course referring to their experience within the particular field being discussed. For example, a user might be an expert in biomedicine, but have no experience in service-modelling, and hence he would fall into the non-experts group when catalogued based on that particular field. Therefore, when we say that non-expert users will be able to model services, it means that users that have not modelled services before will be able to do it with our new tools.

4. Modelling Tools Design

The Service Modelling Tools described in this deliverable will allow SOA4All users to annotate services, both WSDL-based and REST-based, in the Service Provisioning Platform. Please note that the composition of services is out of scope of this task, and it is currently addressed within T2.6 (*SOA4All Process Editor*, [32]).

In this Section, we will explain first the main characteristics of the Simple (as opposed to Composed) Semantic Web Services Editing Framework (Section 4.1). Then we will cover the characteristics of the two main outcomes of this Framework: The WSMO-Lite Editor (Section 4.2), and the MicroWSMO Editor (Section 4.3), which provide environments for providing semantic annotations over traditional Web Services based in WSDL and over RESTful services, respectively.

4.1 Simple SWS Editing Framework

Apart from the Services Browser component addressed in D2.1.1 ([23], Section 4.3.1), the Simple SWS Editing Framework consists of two main components, that we will review in the next two subsections: the WSMO-Lite Editor and the MicroWSMO Editor. However, it is worth noting that from the technical point of view and regarding their inclusion within the SOA4All Studio, and the Service Provisioning Platform in particular, these two editors share some characteristics that we address here.

First, the tools described in this deliverable will be able to make use of the underlying SOA4All Studio Infrastructure Services and UI Components developed in T2.4 [31], leveraging their services for storage, communication, user management, etc., as well as some useful UI widgets.

Additionally, the inclusion of these tools as part of the SOA4All Studio will place some technical requirements on them. Deliverable DX-UI [18] analyses different Rich Internet Applications (RIAs) and justifies the election of Google Web Toolkit (GWT) as the framework to be used in order to support the various functionalities that SOA4All will give (with additional functionalities from the Ext-GWT framework [31]). In the words of their own developers [30], GWT's mission is "*to radically improve the web experience for users by enabling developers to use existing Java tools to build no-compromise AJAX for any modern browser*". This means that we will use the Java programming language to build web applications (in our case, the Service Modelling Tools), as GWT cross-compiles the code into optimized JavaScript that automatically works across all major browsers.

The following picture (Figure 7) depicts the types of communications within SOA4All and shows how the editors of the Simple SWS Editing Framework will communicate with other architectural components of the project. Components in the SOA4All Studio such as the aforementioned ones of T2.4 will be accessed via direct Java calls, while the communications with external components will be done through DSB normalised messages.

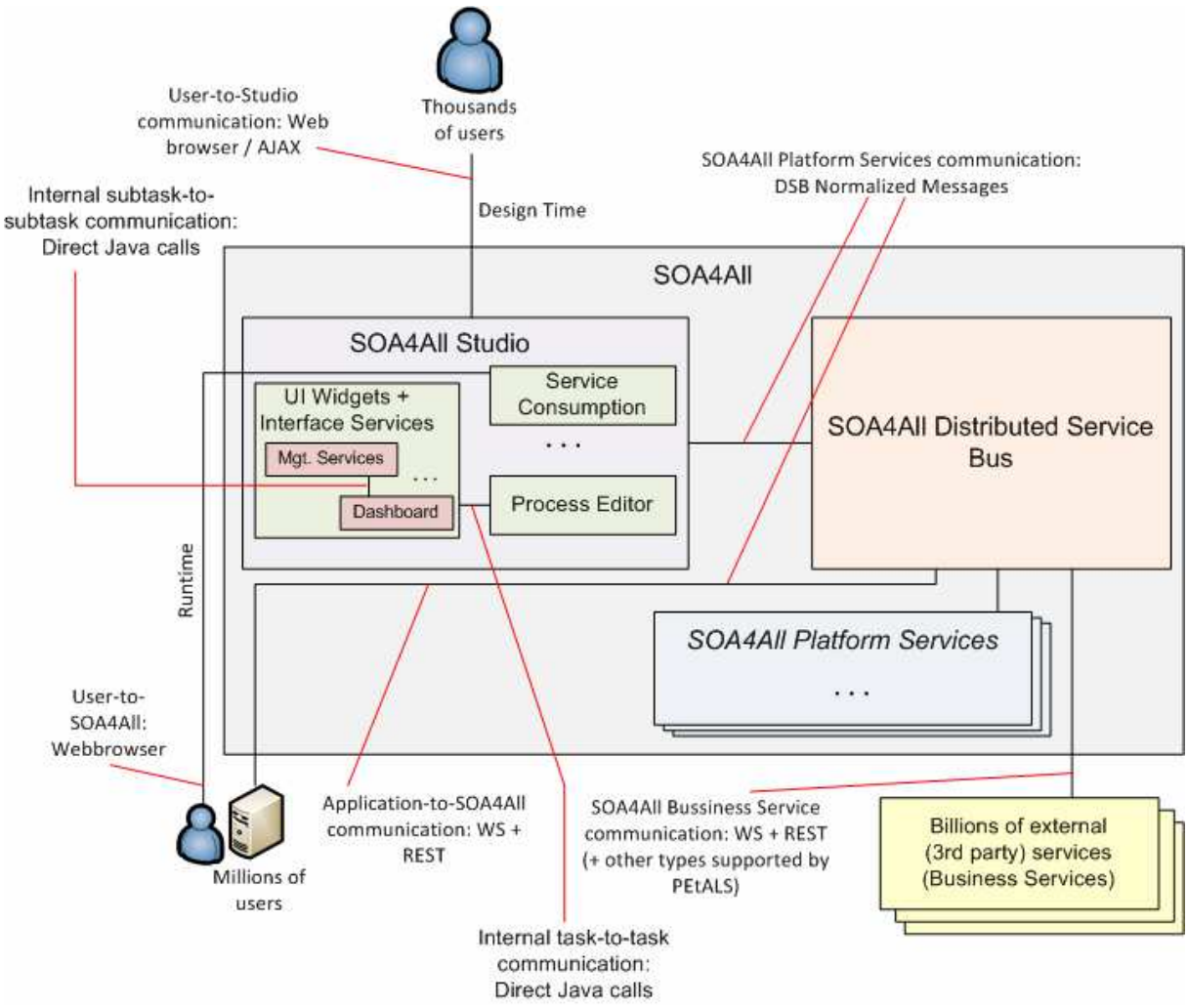

Figure 7: Communication Types in SOA4All

4.2 WSMO-Lite Editor

This section describes in detail the WSMO-Lite Editor, part of the SOA4All Studio. The main function of the component is to provide flexible and user-friendly visual environment for creating semantic web service descriptions starting from classical WSDL services. Being targeted to non-technical expert users, the tool relies on traditional UI components (forms, trees, context menus, etc.) to build comprehensive model representation. The implementation will make use of the rich UI widget library provided by D2.4.1 [31] for building the representation layer as well as the infrastructural services to support the data exchange process.

4.2.1 WSMO-Lite Editor Requirements

Here we identify several important functional requirements for this component. First of all, it should provide instruments for creating semantic annotations on plain services and at the same time it must be capable to detect pre-existing semantic annotations of services and to support any further editing operations. Thus, the WSMO-Lite Editor must support two types of input artefacts – WSDL service descriptions and SAWSDL annotations over WSDL. These two types of descriptions are retrieved from the DSB with the assistance of the Annotations Recommender (D2.1.1 [23], Section 4.3.2). The outcome of the editor represents fully or partially semantically annotated web service descriptions (in SAWSDL). The result is stored in the SWS Library accessible through the DSB.

Another important aspect in the requirements for the editor component is building comprehensive representation model of the editing objects. Going beyond the toy examples, the service descriptions contain a lot of information with many technical details, which makes them hard to read/explore. On the other hand, not all of it is relevant for the annotation task and might be ignored. To cover this requirement the editor must be capable to filter out the content restricting the representation to the minimal set of relevant elements. Then on user demand, more detailed information can be visualized.

The management of domain ontology resources (use for annotation) is also an important issue. The UI must provide clear and intuitive taxonomy representation to facilitate the user in locating the desired information. Additionally, there should be an easy way to gain access to ontological information depending on the user access rights.

The following table summarizes all requirements that have been described along with their priority.

ID	Name	Priority (1=high, 10=low)
WLE-1	Annotation of plain services	1
WLE-2	Editing existing annotations	1
WLE-3	Adequate visual service representation	2
WLE-4	Comprehensive domain model visual representation	4

Table 3: Summary of WSMO-Lite Editor Requirements

The focus of the following sections is visual design and functional specification of the WSMO-Lite Editor.

4.2.2 WSMO-Lite Editor Use Cases

The following diagram (Figure 8) summarizes the major aspects of the WSMO-Lite Editor's functionality described in next sections.

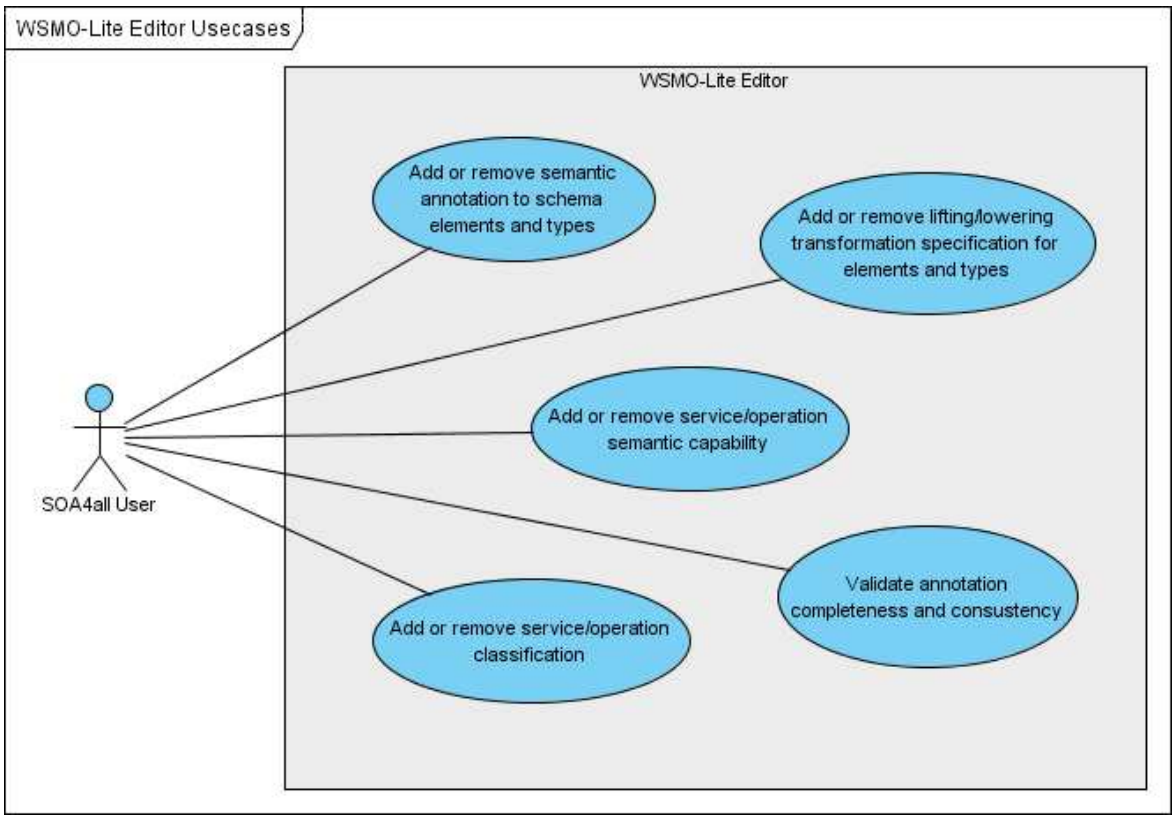

Figure 8: WSMO-Lite Editor usecases

4.2.3 Functional Specification and Graphical User Interface of the WSMO-Lite Editor

The WSMO-Lite editor is implemented in a light-weight Web 2.0 style, on top of the visual component provided by D2.4. For service descriptions and ontology representation, the *Taxonomy Selector* widget (D2.4, section 3.5.5) will be deployed. Other widgets like *Search & Result Handler*, *Fault Handler*, *Help System* will further support the annotation process.

Figure 9 outlines the visual appearance of the editor component with its supporting views.

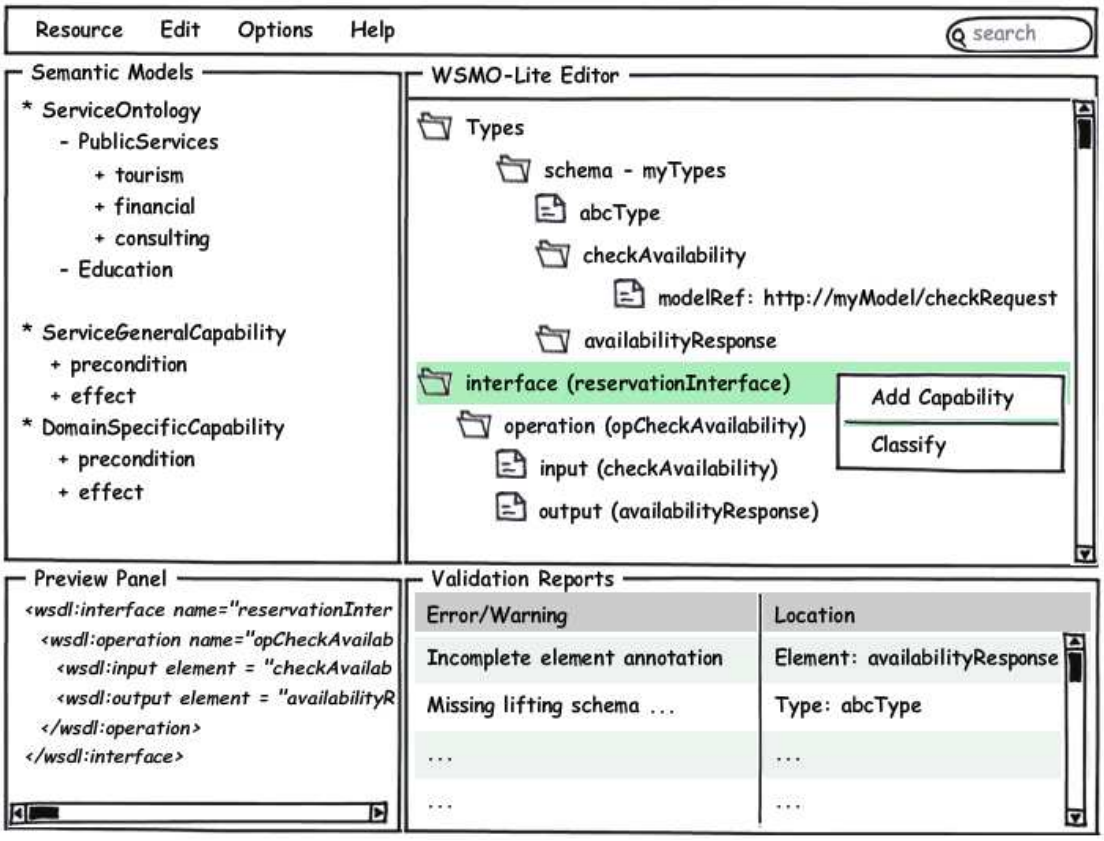

Figure 9: WSMO-Lite Editor

The editor area represents the service description content in a tree style filtering the irrelevant details (for the annotation process). The nodes of the tree correspond to various parts of the description like data-model types and elements, interfaces, operations and corresponding input and output specifications. Apart from the pure WSDL elements, the view reveals any semantic annotations already created for the service. This includes links from schema types and elements to ontology elements (including lifting and lowering specifications), service and/or operation classification, assignment of semantic capabilities and others.

There are several fundamental editing operations which are supported here:

- Adding references from schema types and elements to ontology elements
- Adding lifting and lowering transformation references for schema types and elements
- Adding categorization annotation for interfaces and operations (pointing to ontology elements)
- Adding semantic capability annotation for interfaces and operations (referring to reusable precondition and effect definitions in a semantic model)
- Removing any kind of semantic annotations and transformation specifications

These operations are realized in the GUI by context menu actions on certain selection sensitive elements. The environment offers all applicable operations depending on the usage context. Alternatively, *Drag and Drop* techniques might be utilized for adding annotations.

An important sub-component for the annotation process is the **Semantic Models** view. It provides a front-end view for the ontologies used for annotation. The representation reveals the ontologies content in tree-like structures, facilitating the user to locate and use certain semantic model elements. The elements of this view can be dragged directly to the editor

area and dropped on certain elements establishing model references. The basic management operations over ontologies in this view are retrieval and deletion. The content of the view is preserved between the working sessions of each user.

During the process of annotation, the WSMO-Lite Editor provides basic completeness and consistency validation support. A reflection of this functionality is a **Validation Reports** view containing a (possibly empty) list of problematic issues report. This view and the editor component are interconnected assisting the user in locating/identifying problematic spots in the document.

The rest of the UI components more or less contribute to the annotation process, like simple editing operations (copy / paste / undo / search), annotation result preview in the *Preview Panel*, additional resources management (open / close).

4.3 MicroWSMO Editor

The MicroWSMO Editor is a user interface component and is a part of the Provisioning Platform. Its main functionality is to enable the user to create, edit, and delete MicroWSMO service annotations by retrieving and visualizing data from the Annotations Recommender (D2.1.1 [23] Section 4.3.2) and the Distributed Service Bus (DSB). The following sections describe the editor's architecture, use cases and suggested user interface.

The MicroWSMO Editor consists of three main components, as seen in Figure 10. The Visualization component implements functionalities, necessary for the proper graphical representation of the annotations, including colour-schemes, visualization patterns, representation of the toolbar and service-property representation rules. This component is not shared with the WSMO-Lite Editor because the visualization requirements for a RESTful service differ greatly from those for a WSDL service. RESTful services require that the HTML descriptions of the APIs are pre-processed, in order to identify and highlight service operations and properties. WSDL descriptions, on the other hand, require that a tree view of the XML elements is build.

The Interaction component controls data type and information flow between the editor and other components, such as the Annotations Recommender and the SWS Repository, accessible through the DSB. It processes user requests, retrieves the necessary data and pre-processes it for the visualization component. The Navigation component, on the other hand, controls the sequence of activities, which the user can perform by specifying requirements and effects of a given user action. For example, the user cannot save an annotation, without assigning a domain to the service, and the result has to be a saved service annotation.

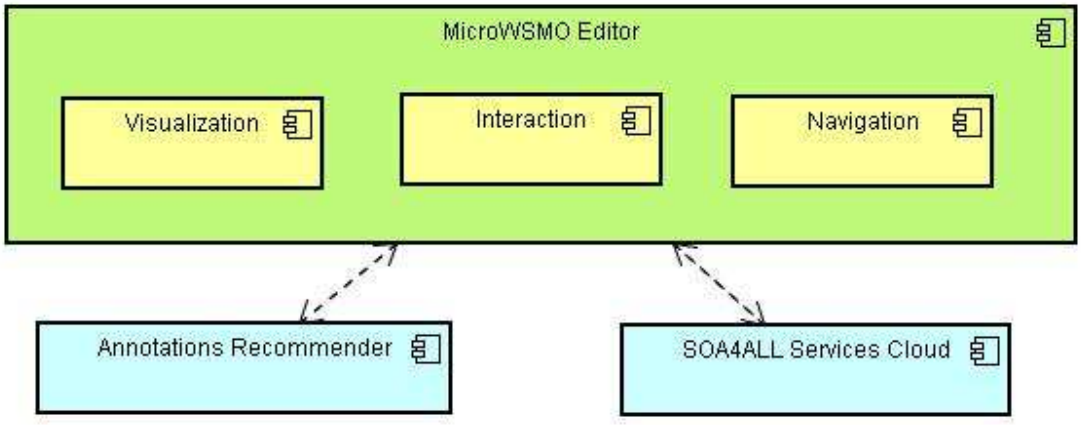

Figure 10: Components of the MicroWSMO Editor

4.3.1 MicroWSMO Editor Requirements

The here described design of the MicroWSMO Editor is based on a number of requirements and restrictions, which determine its main components and functionalities. First, the editor has to enable the user to manipulate all MicroWSMO service properties. These include the REST service URI, all its operations, with corresponding addresses, HTTP methods, parameters and input/output data formats and labels. Also, the user has to be able to create new MicroWSMO annotations, to edit them and, if necessary, to delete them.

Considering the data flow and its processing, each semantic service description should have a reference to the user, who created it, and to the service, WSDL or REST, on which it is based, and vice versa. This enables the usage of three separate distributed repositories for storing the SWS, the Crawled Data and the User Profiles, as opposed to having only one, heavy-weight repository

In addition, the chosen technology also imposes restrictions on the design of the editor. The MicroWSMO Editor will reuse some of the technology of Magpie [33][34], which is a Semantic Web browser that enhances the browsed text with semantic information. In particular, Magpie's communication model and infrastructure will be reused, which impose some additional restrictions on the implementation of the editor. All of these requirements are reflected in the architecture, the functionality and the user interface of the component.

The following table summarizes all requirements that have been described along with their priority.

ID	Name	Priority (1=high, 10=low)
MWE-1	Indentify MicroWSMO Properties	1
MWE-2	Create a new MicroWSMO Annotation (equivalent to edit existing MicroWSMO annotation)	1
MWE-3	Delete existing MicroWSMO Annotation	4
MWE-4	Cross-reference between the semantic annotation, the service description and the user	1
MWE-5	Adaptation of Magpie technical requirements	2
MWE-6	Visual service representation of HTML and hREST	2
MWE-7	Visual representation of the RESTful service properties	2
MWE-8	Validation of annotation's completeness	4

Table 4: Summary of MicroWSMO Editor Requirements

4.3.2 MicroWSMO Editor Use Cases

The component architecture of the MicroWSMO Editor implements three main use cases, necessary for the manipulation of MicroWSMO service annotations. These use cases are described in the UML diagram in Figure 11.

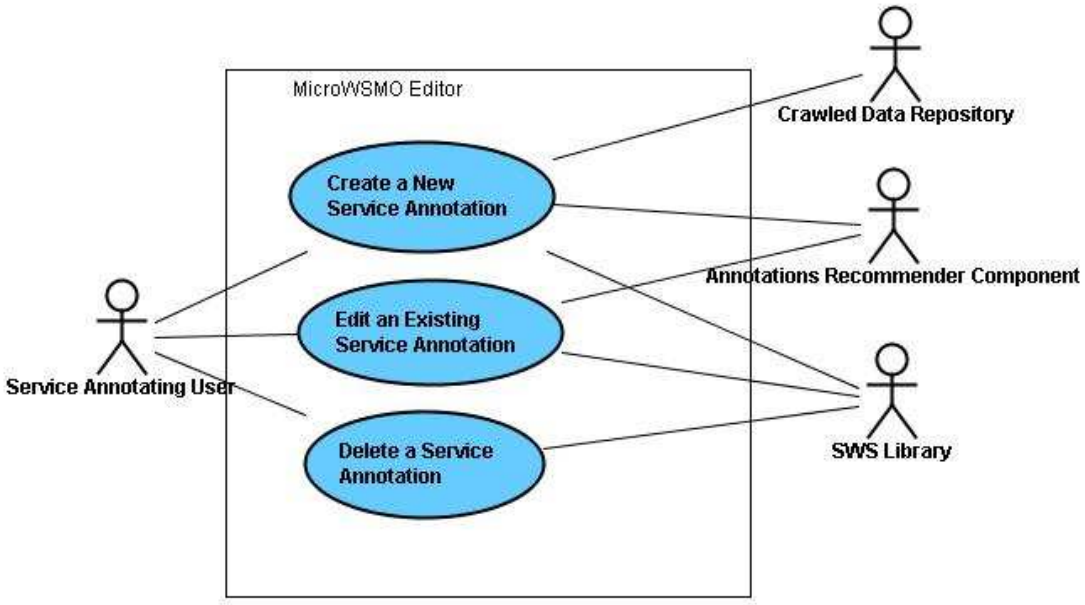

Figure 11: MicroWSMO Use Cases

First, the user can *Create a New Service Annotation*. While browsing through the services, the user may find a service without any metadata and may choose to annotate it. Second, the user may find errors or inaccuracies in existing annotations, which can be corrected during the *Edit an Existing Service Annotation* use case. Finally, some service annotations become obsolete with time because the functionality of the service undergoes major changes or it does not exist anymore. In these cases, it is necessary to be able *to Delete a Service Annotation*. These three main use cases require access to the Annotations Recommender Component (detailed description in D2.1.1 [23] Section 4.3.2), which assists the user by suggesting possible annotations, to the SWS Library and to the Crawled Data, which are facilitated by the DSB.

4.3.2.1 Create New Service Annotation

Create New Service Annotation is performed when a user wants to add semantic information to a service without any previous metadata. This requires that the MicroWSMO Editor retrieves the REST service description and related documents, such as service API text description, implementation recommendations and blog entries, from the Crawled Data (D5.1.2) repository and visualizes them. The visualization is done in a RESTful-specific manner, as opposed to a WSDL-specific one, by presenting the data based on the API description. In addition, the pre-processed data from the Annotations Recommender also has to be retrieved and represented using colour- and font-schemes, which assist the user in recognising service properties. For example, parameter names and method names should be highlighted, so that they can be easily visually identified. After this, data retrieval and visualization step is completed, the user has to validate the domain and classification automatically assigned to the service during the Annotations Recommender Preprocessing phase. If one of the descriptions is inaccurate, the user can change it by choosing from a precompiled rated list (Annotations Recommender Preprocessing phase) of the top five possible suggestions. Once this is completed, the user can choose one of the domain ontologies suggested by the Annotations Recommender (For details on the domain ontology recommendation process, refer to D2.1.1 Section 4.3.2). The list of domain ontologies is rated and includes a short description, as well as a summary of its main concepts. This is necessary in order to ensure that the user has enough information to pick the suitable domain ontology. Figure 12 visualizes the main activities involved in the *Create New Service Annotation* use case.

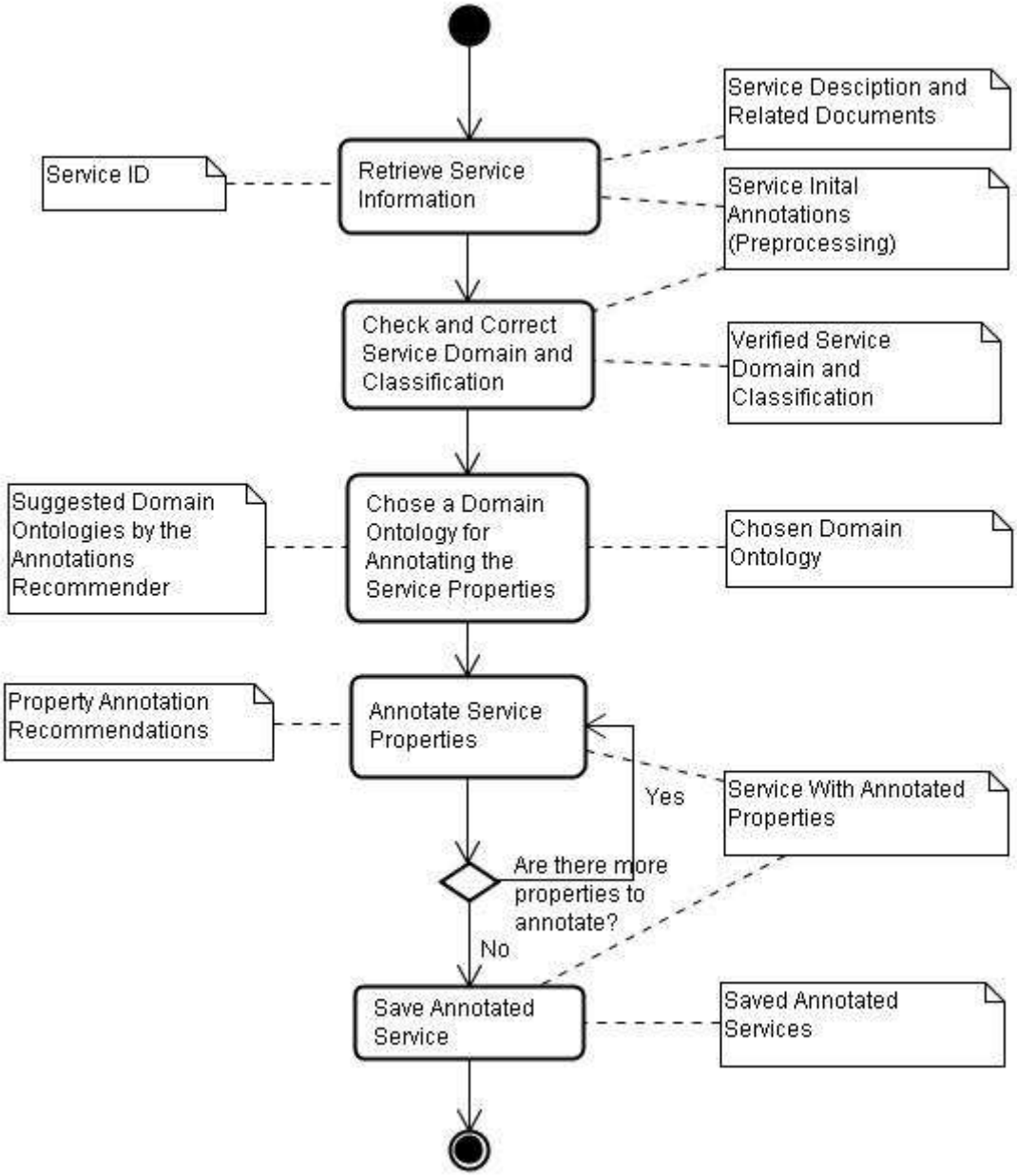

Figure 12: Activity Diagram "Create New Service Annotation"

After the user chooses a domain ontology, each of the service properties can be annotated based on this ontology's concepts. The service properties, including the name of the service, its address, operation names, parameter names, input and output types are identified by the Annotations Recommender during a service data pre-processing step. However, some of the properties could have been missed or falsely identified. Therefore, during this activity, the user has the option to change them, by making a text selection and choosing from the context (right-mouse click) menu whether the text is a service property or not, and what type. This then results in automatically updating the hREST [35] mark-up of the text, which is the basis for the resulting MicroWSMO annotation.

If all service properties can be annotated by using the chosen domain ontology, then the user may do so and save the resulting annotated service. However, in some cases a service property cannot be described by any of the concepts of the domain ontology. If this occurs, the user can 'locally' extend the ontology by, for example, introducing a new concept, which is stored together with the service annotation and does not really modify the chosen domain

ontology. Naturally, if the user has the access right, he can modify the ontology to better be able to describe the service and then use it for the actual annotation.

A summary of input and output artefacts in the *Create New Service Annotation* is given in Table 5. The only required input information is the Service ID and the User ID. The use case results is a Saved Annotated Service, which is confirmed to the user by displaying the SWSId, after the "save" button is pushed. It is important to point out, that similarly to the domain ontology not being able to describe all service properties, in some cases the classification taxonomies and the ontology of service domains need to be modified and extended. The user can edit the taxonomies and the ontology used for assigning a domain to a given service, by downloading them, editing them on his computer and uploading them to the Provisioning Platform.

Use Case	Input Information	Output Information	Preconditions and Effects	Comments
Create New Service Annotation	-Service ID -User ID (from the active user)	-Saved Annotated Service, which is represented by a SWSId	Access to the Semantic Web Services Registry and to the Crawled Data is required. Access to the Annotations Recommender is required.	The Annotate Service Properties, as well as the validations of the service domain and classification, include options for providing annotations not included in the recommendation list and options for extending the suggested ontologies/taxonomies.

Table 5: Artefacts in the "Create New Service Annotation" Use Case

4.3.2.2 Edit Existing Service Annotation

The *Edit Existing Service Annotation* is performed in a very similar way to the *Create New Service Annotation*. Based on the lifecycle of semantic web services, specified in D2.1.1 Section 4.2, one service (WSDL or REST) has a multitude of corresponding semantic descriptions, created by different users. In addition, the editing of a MicroWSMO description is equivalent to creating a new semantic description with the user as owner[6]. This is done in order to prevent the case, in which a SWS is already used in processes, after which the SWS description is modified, causing the processes no longer to be executable.

4.3.2.3 Delete Service Annotation

In some cases, semantic service descriptions become obsolete. In order to facilitate the discovery of up-to-date services, based on accurate annotations, the user has to be able to delete semantic descriptions of services, which no longer exist or whose functionality has drastically changed. The deletion of a service semantic description requires the SWS ID and the user ID as input, since users are allowed to delete only annotations, of which they are the owner and which are not used in any service compositions. Table 6 includes a short overview of the use case's artefacts, preconditions and effects.

Use Case	Input Information	Output Information	Preconditions and Effects	Comments
Delete Service Annotation	-Service Annotation ID	- Confirmation for Deleted Annotation	There needs to be a method for verifying which user is allowed to delete which annotations. Initial	The deletion of an annotation involves not only the removal from the Semantic Web Services

[6] A user is the owner of a semantic annotation if he/she created it.

	- User ID		approach: a user is allowed to delete only his own annotations.	Registry but also the deletion from the document holding the service to annotations relations, since one service can have multiple annotations.

Table 6: Artefacts in the "Delete Service Annotation" Use Case

A new functionality, which has to be implemented in this use case, is the verification whether a user is allowed to delete a semantic service description or not. This includes a user verification functionality, since a user is allowed to delete only services, of which he is the owner. In addition, a function that checks if a service description is used in any composite process also needs to be provided. As shown in Figure 13, if the user has no permission to delete the annotation, an error message is displayed. On the other hand, if deletion is possible, the user first has to confirm that he is certain that the annotation should be permanently removed, after which the deletion is performed.

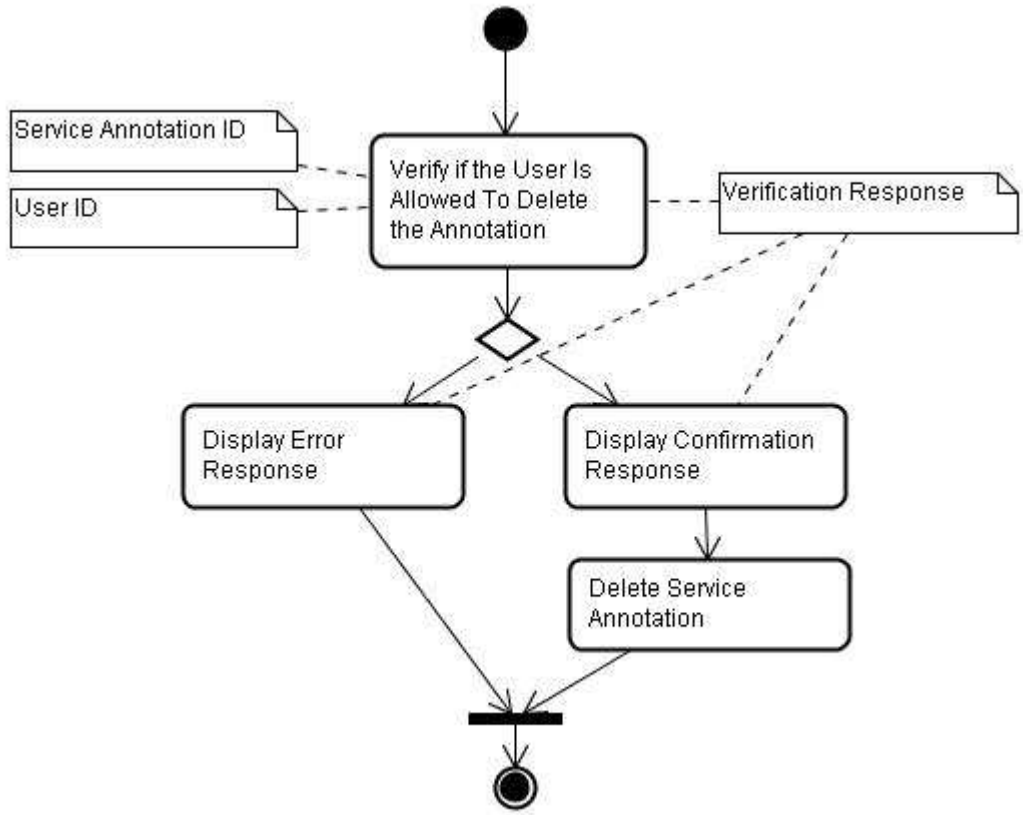

Figure 13: Activity Diagram "Delete Service Annotation"

The following section identifies the main MicroWSMO Editor visualization elements and shows some graphical mockups of the editor.

4.3.3 Functional Specification and Graphical User Interface of the MicroWSMO Editor

The MicroWSMO Editor will be implemented in an extra free-flowing widget, which will be dockable in the main browser window of the SOA4All Studio. In this way, the user can position and resize it accordingly. The MicroWSMO Editor components are somewhat different from the WSMO-Lite ones because of the specific nature of the services, which are to be annotated. First, RESTful services are usually represented by a HTML description of

the API. This requires that HTML is visualized and that service properties and operations are highlighted for the user. In contrast, WSDL service visualization is done by building an XML tree of the service's properties. Therefore, the visualization of a RESTful service could be compared to a web browser, while the visualization of a WSDL service to an XML editor. Second, while in WSDL service properties are clearly identified, the annotation of RESTful services requires an additional initial step for identifying service properties and operations.

The Semantic Models window will consist of four expandable/collapsible sections for displaying the service domain, classification, domain ontology and annotated properties. In addition, the editor implements a context menu for marking service properties or removing falsely identified ones. Each service property will be graphically highlighted (colour and font) and will include a drop-down menu for direct annotation. The service properties' annotations can also be viewed and edited in the main MicroWSMO Editor window, in the Properties section. Similarly to the WSMO-Lite Editor, the MicroWSMO Editor includes a Preview Panel, which shows the actual HTML and the inserted hREST tags, and a Validation Reports panel. It is important to point out that the validation for completeness of the service annotations can be done only on the basis of the identified properties. If the service description contains some additional properties and these are not marked as such, then they will also not be considered in the completeness validation process. Figure 14 provides an overall mockup visualization of the MicroWSMO editor.

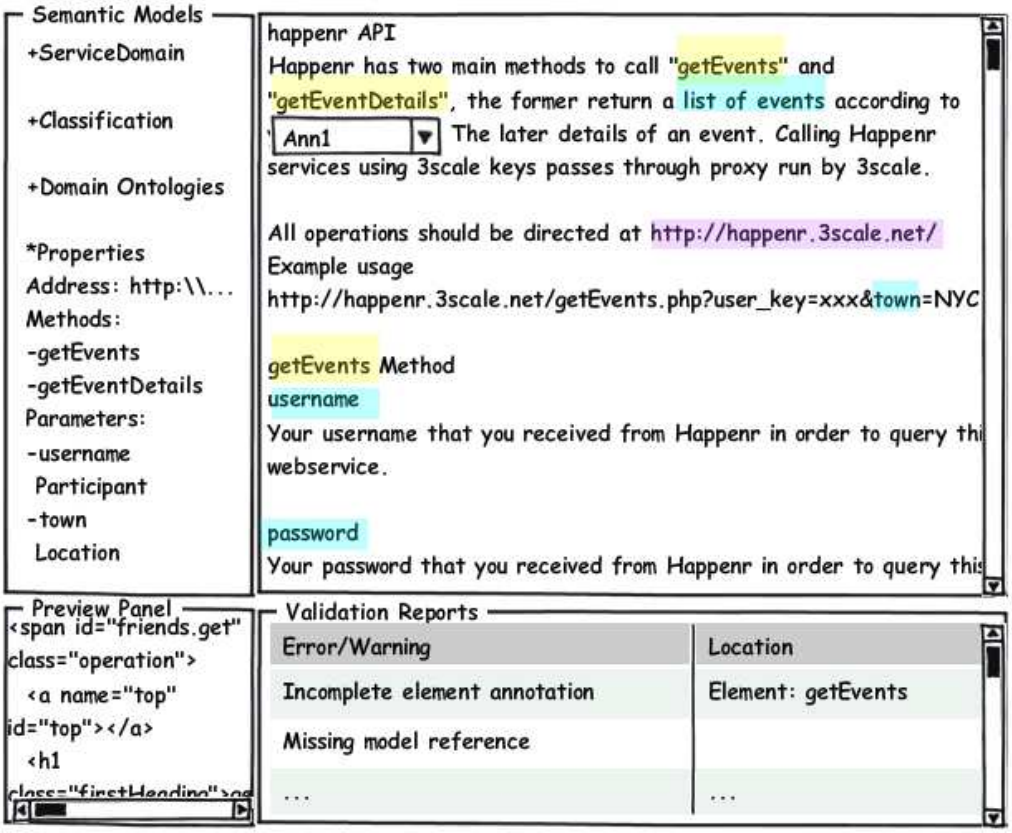

Figure 14: MicroWSMO Editor

The following figures (Figure 15, Figure 16, Figure 17) visualize the process of creating a new semantic service description. First, the user has to verify the automatically assigned domain and classification to the service (Telecommunication and Localization Taxonomy). If these are not accurate, they can be changed by choosing from the list of suggestions (-Marketing, Entertainment, City Guide, -Favorites Taxonomy).

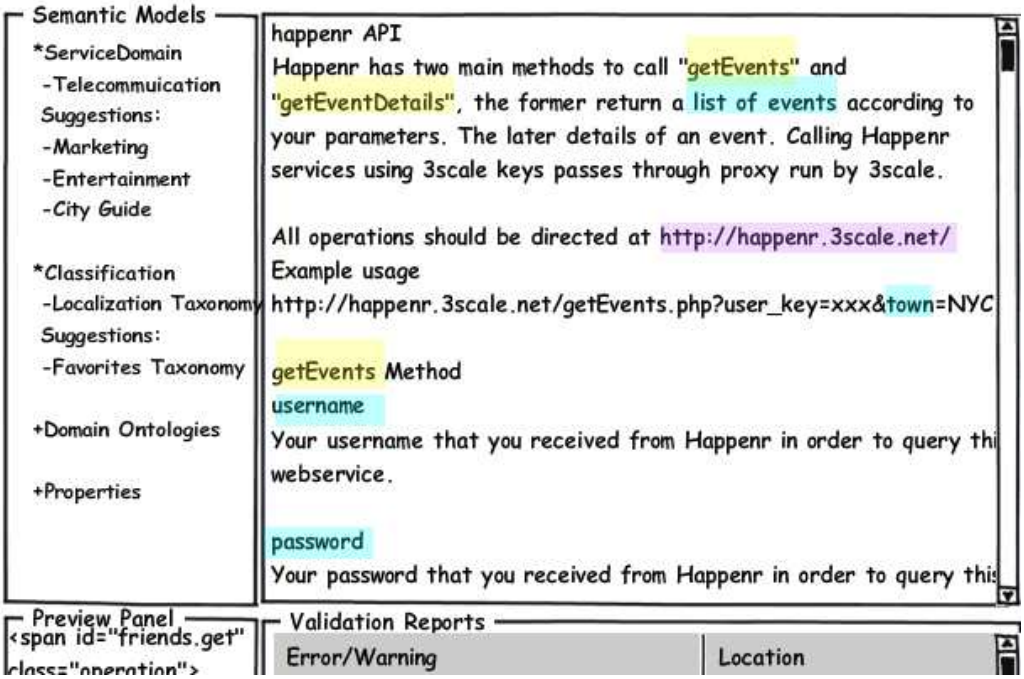

Figure 15: Verifying the Service's Domain And Classification

Figure 16 shows how the user can choose a domain ontology and use it to annotate the highlighted service properties. For example, an Event Ontology can be used to annotate the username parameter as a participant.

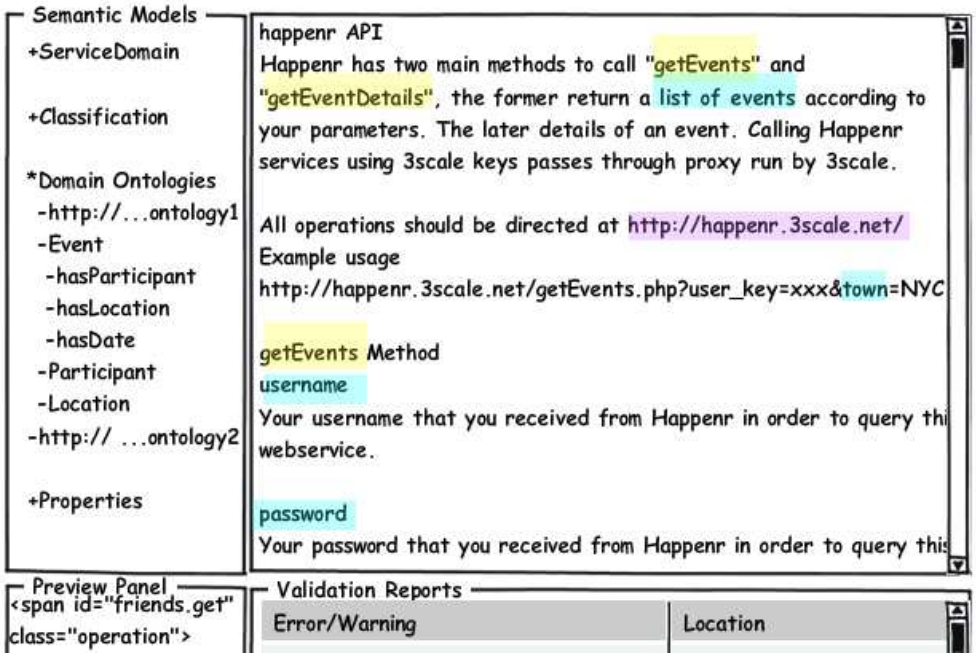

Figure 16: Choosing a Domain Ontology

The user can view and edit already annotated properties, in the Properties section of the MicroWSMO Editor (Figure 17).

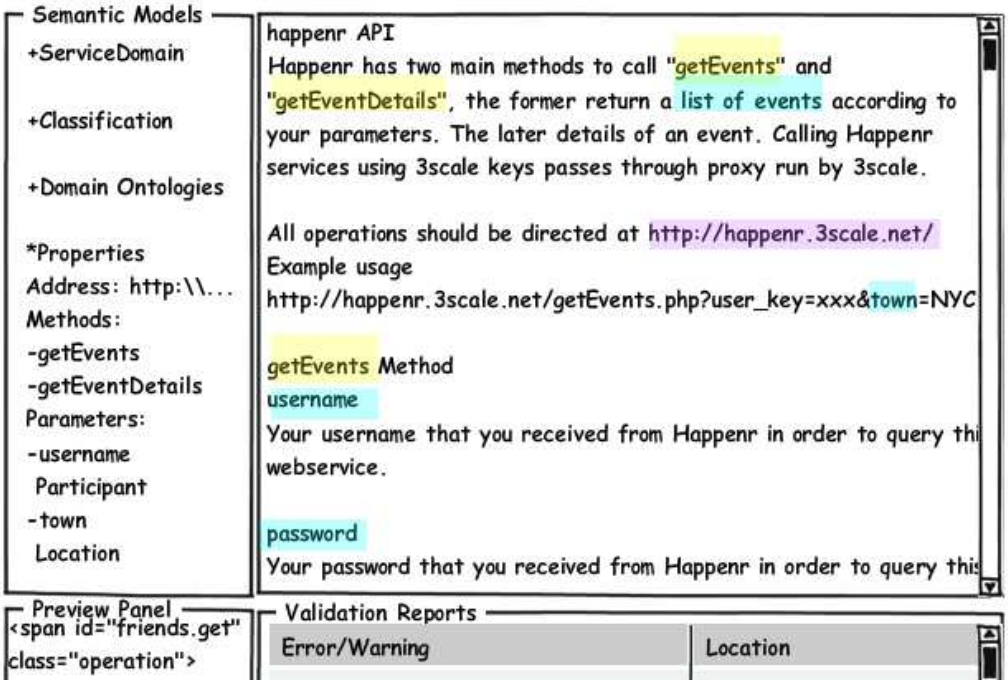

Figure 17: Annotating Service Properties

5. Conclusions

In this deliverable, we have addressed the general design characteristics for the Service Modelling Tools used in SOA4All within the context of the Simple Semantic Web Services Editing Framework of the Service Provisioning Platform.

In order to reach an efficient design of the tools, we have begun by sketching our vision on what characteristics should the tools have in order to satisfy the envisaged use cases, namely that they should be lightweight and web-based tools, useable by different kinds of users, both experts and non-experts, and enabling a community-oriented approach towards modelling.

Finally, we have specified our first general design characteristics for these tools, addressing the general Framework and the two main outcomes, the WSMO-Lite Editor and the MicroWSMO Editor.

6. References

[1] World Wide Web Consortium, W3C: Simple Object Access Protocol, SOAP, Version 1.2 Part 0: Primer, (2003). Web site: http://www.w3.org/TR/soap12-part0/).

[2] World Wide Web Consortium, W3C: WSDL: Web services Description Language (WSDL) 1.1, (2001). Web site: http://www.w3.org/TR/2001/NOTE-wsdl-20010315.

[3] World Wide Web Consortium, W3C: Universal Description, Discovery and Integration: UDDI Spec Technical Committee Specification v. 3.0, (2003). Web site: http://uddi.org/pubs/uddi-v3.0.1-20031014.htm.

[4] WSMO Working Group, D2v1.0: Web Service Modeling Ontology (WSMO). WSMO Working Draft, 2004. Web site: http://www.wsmo.org/2004/d2/v1.0/

[5] OWL-S Coalition: OWL-S 1.1 release. (2004). http://www.daml.org/services/owl-s/1.1/

[6] World Wide Web Consortium, W3C: Semantic Annotations for WSDL and XML Schema. W3C Recommendation (August 2007). Web site: http://www.w3.org/TR/sawsdl/

[7] D. Fensel and C. Bussler: The Web Service Modeling Framework WSMF, Electronic Commerce Research and Applications, 1(2), 2002.

[8] T. Vitvar, J. Kopecký, M. Zaremba, and D. Fensel. WSMO-Lite: Lightweight Descriptions of Services on the Web. In Proceedings of the IEEE European Conference on Web Services, Halle (Saale), Germany, 11 2007. IEEE Computer Society.

[9] J. Kopecký, T. Vitvar, and D. Fensel. MicroWSMO: Semantic Description of RESTful Services. WSMO Working Draft (February 2008).

[10] R. T. Fielding. Architectural styles and the design of network-based software architectures. PhD thesis, University of California, Irvine, 2000. Chair-Richard N. Taylor.

[11] G. Alonso, F. Casati, H. Kuno, and V. Machiraju, (2003). Web services: Concepts, architectures and applications. Springer Verlag.

[12] J. Vogel, L. Xu, F. Schnabel, R. Krummenacher, S. Abels, R. Gonzalez, A. Mos, N. Mehandjiev, L. Cekov, G. Ripa, and M. Villa: State of the Art Report and Requirements for Service Construction, Project Deliverable D6.1.1, EU FP7 SOA4ALL project, August 2008.

[13] Object Management Group: Business Process Modeling Notation Specification (2006)

[14] Workflow Management Coalition: XML Process Definition Language

[15] D. Jordan et. al., Web Services Business Process Execution Language Version 2.0, OASIS Standard (2007)

[16] A. Arkin et. al., Web Service Choreography Interface (WSCI) 1.0, W3C Note (2002)

[17] D. Beckett, B. McBride, RDF/XML Syntax Specification, W3C Recommendation (2004)

[18] S. Abels, L. Chekov, N. Mehandjiev, G. Álvaro, and C. Ruiz, Holistic User Interface Design, Project Deliverable DX-UI, Holistic User Interface, EU FP7 SOA4ALL project, August 2008.

[19] M. Dimitrov, A. Simov, M. Konstantinov, L. Cekov, V, Momtchev, WSMO Studio Users Guide, July 2007. http://www.wsmostudio.org/

[20] V. K. Chaudhri, A. Farquhar, R. Fikes, P. Karp, J. Rice, Open Knowledge Base Connectivity, April 1998, http://www.ai.sri.com/~okbc/spec.html

[21] A. Mocan, E. Cimpian, M. Moran, E. Della Valle, Semantic Execution Environment, April 2006.

[22] N. Steinmetz, I. Toma. Web Service Modelling Language (WSML) Reference. Final Draft (August 2008).

[23] S. Dietze, C. Pedrinaci, A. Gugliotta, P. Merle, I. Martínez, G. Álvaro, C. Ruiz, M. Villa, S. Abels: Service Provisioning Platform Design, Project Deliverable D2.1.1, EU FP7 SOA4ALL project, August 2008.

[24] R. González-Cabero, C. Pedrinaci, J.M. Gómez, C. Ruiz, C. Hamerling, A. Mos, S. Abel: Service Monitoring and Management Tool Suite Design, Project Deliverable D2.3.1, EU FP7 SOA4ALL project, August 2008.

[25] G. Álvaro, S. Abels, N. Mehandjiev and M. Villa: Service Consumption Platform Design, Project Deliverable D2.2.1, EU FP7 SOA4ALL project, August 2008.

[26] K. Bennett, P. Layzell, D. Budgen, P. Brereton, L. Macaulay, M. Munro: Service-Based Software: The Future for Flexible Software. (1999)

[27] M. A. Cusamano: The Changing Software Business: Moving from Products to Services, Computer, vol. 41, no. 1, pp. 20-27. (January 2008)

[28] A. Toffler: The Third Wave. Bantam Books ISBN 0-553-24698-4. (1980)

[29] http://www.eclipse.org/stp/

[30] http://code.google.com/webtoolkit

[31] S. Abels et al. SOA4All Studio First demonstrator + Interface Specification, D2.4.1, EU FP7 SOA4ALL project, February 2009

[32] J. Vogel et al., Specification of the SOA4All Process Editor, D2.6.1, EU FP7 SOA4ALL project, February 2009

[33] M. Dzbor, E. Motta, J. Domingue.: Magpie: Experiences in supporting Semantic Web browsing. Web Semantics: Science, Services and Agents on the World Wide Web 5 (2007) 204—222

[34] M. d'Aquin, E. Motta, M. Dzbor, L. Gridinoc, T. Heath, M. Sabou.: Collaborative Semantic Authoring. IEEE Intelligent Systems 23 (2008) 80—83

[35] J. Kopecky, K. Gomadam, and T. Vitvar, hRESTS: An HTML Microformat for Describing RESTful Web Services, Kno.e.sis tech. report, Wright State Univ., 2008.

[36] J. Domingue, L. Cabral, S. Galizia, V. Tanasescu, A. Gugliotta, B. Norton, and C. Pedrinaci. 2008. IRS-III: A broker-based approach to semantic Web services. Web Semant. 6, 2 (Apr. 2008), 109-132.

[37] A. Haller, E. Cimpian, A. Mocan, E. Oren, and C. Bussler. 2005. WSMX - A Semantic Service-Oriented Architecture. In Proceedings of the IEEE international Conference on Web Services (July 11 - 15, 2005). ICWS. IEEE Computer Society, Washington, DC, 321-328.

