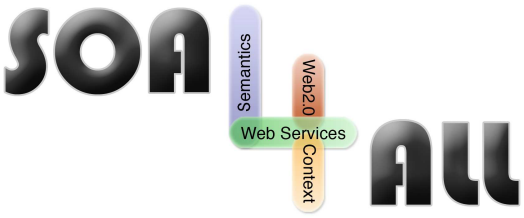

Project Number:	**215219**
Project Acronym:	**SOAAll**
Project Title:	**Service Oriented Architectures for All**
Instrument:	**Integrated Project**
Thematic Priority:	**Information and Communication Technologies**

D2.1.1 Service Provisioning Platform Design

Activity N:	Activity 1 – Fundamental and Integration Activities	
Work Package:	WP2 – SOAAll 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:		The Open University
Revision:		2.0
Author(s):	Carlos Pedrinaci OU Maria Maleshkova OU Guillermo Álvaro Rey ISOCO Carlos Ruiz Moreno ISOCO Stefan Dietze OU Alessio Gugliotta OU	
Reviewers:	Juergen Vogel SAP Jacek Kopecky UIBK	

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		Document structure, table of contents	Stefan Dietze, Carlos Pedrinaci, Alessio Gugliotta
0.2		Restructured. Vision described.	Stefan Dietze
0.3		Glossary, Introduction, Motivation, Architecture sections updated. Sections 5.2.5 and 6.5 added.	Philippe Merle (INRIA)
0.4		Architecture refined. Document further consolidated. Related work on emergent semantics added. Sections 5.2.3.2 and 5.2.4 added.	Stefan Dietze, Carlos Pedrinaci, Alessio Gugliotta.
0.5		Various typos. Figure renumbering. Various comments, Figure 5 page 25 updated. References updated (not all see comments). Check and complete glossary.	Philippe Merle (INRIA)
0.6		Executive summary, conclusions and subsection on annotation recommendation added. Final consolidation. Typos removed.	Stefan Dietze
0.7		Sections 2.1 - 2.6 added. Further consolidation. References improved and consolidated.	Matteo Villa, Philippe Merle, Stefan Dietze
1.0		Further consolidation, minor improvements (typos, grammar).	Stefan Dietze, Philippe Merle.
1.1		Revision regarding reviewer comments.	All authors.
1.2		Improvement suggestions base don reviewer comments and new structure. Add overall architectural view, lifecycle of services	Carlos Pedrinaci
1.3		Merge inputs on Annotations recommender and 4.2.4, 4.2.5 and 4.2.6	Maria Maleshkova, Guillermo Alvaro, Carlos Pedrinaci
1.4		Include content on Service Browser and Service Editing Frameworks	Carlos Pedrinaci

1.5		Updates to the annotation recommender	Maria Maleshkova
1.6		Final edits	Carlos Pedrinaci
1.7		Add newer version of 4.3.5-4.3.7	Guillermo Alvaro
1.8		Integrate last changes	Carlos Pedrinaci
1.9		Address reviewers comments	Carlos Pedrinaci, Maria Maleshkova, Guillermo Alvaro
2.0		Last editorial fixes	Carlos Pedrinaci
Final		Overall format and quality revisión	Malena Donato

Table of Contents

List of Figures

List of Tables

Glossary of Acronyms

Acronym	Full Name
AI	Artificial Intelligence
AJAX	Asynchronous JavaScript And XML
API	Application Programming Interface
BIC	Bayesian Information Criterion
BPEL	Business Process Execution Language
BPM	Business Process Modeling; Business Process Management
BT	British Telecom
BT 21CN	BT 21st Century Network
C2C	Consumer toŹ Consumer
CMS	Content Management System
DSB	Distributed Service Bus
EC	European Commission
eCommerce	Electronic Commerce
ESB	Enterprise Service Bus
eShop	Electronic Shop
EU	European Union
EXT GWT	Extended GWT
FOAF	Friend Of A Friend
FP	Framework Program
FP7	The 7th Framework Program
GUI	Graphical User Interface
GWT	Google Web Toolkit
hRESTS	HTML format for describing RESTful Services
HTML	HyperText Markup Language
HTTP	HyperText Transfer Protocol
IE	Information Extraction
IP	Integrated Project
ISP	Internet Service Provider
IST	Information Society Technology
IT	Information Technology
KIM	Knowledge and Information Management
ML	Machine Learning
NLP	Natural Language Processing
OASIS	Organization for the Advancement of Structured Information Standards
OBIE	Ontology-Based IE
OWL	Web Ontology Language
OWL-S	Web Ontology Language for Services
PANKOW	Pattern-based Annotation through Knowledge on the Web
RDF	Resource Description Framework
RDFa	RDF in Attributes
REST	Representational State Transfer
SAP	Systeme Anwendungen und Produkte
SA-REST	Semantic Annotations for RESTful Services
SAWSDL	Semantic Annotations for WSDL
SOA	Service-Oriented Architecture
SOA4All	Service-Oriented Architectures for All
SOAP	Simple Object Access Protocol
SWS	Semantic Web Service
T	Task

Acronym	Full Name
UDDI	Universal Description Discovery and Integration
UI	User Interface
UML	Unified Modeling Language
URI	Uniform Resource Identifier
URL	Uniform Resource Locator
W3C	World Wide Web Consortium
WP	Work Package
WS	Web Service
WSDL	Web Service Description Language
WSML	Web Service Modeling Language
WSMO	Web Service Modeling Ontology
WSMX	Web Service Execution Environment
WWW	World Wide Web
XHTML	Extensible HyperText Markup Language
XML	eXtended Markup Language

Executive summary

Traditional web services technologies suffer from a lack of automation for key activities within the lifecycle of web service-based applications such as the discovery or composition of services. To cater for this semantic web services technologies enhance services with semantic descriptions that are amenable to automated reasoning, thus paving the way for the application for knowledge-based algorithms to better support the automation of these tasks. Still, obtaining rich enough semantic descriptions that can enable this requires an important knowledge-acquisition effort which represents the main bottleneck for a wider and systematic application of semantic web services technologies.

The SOAAll Studio is a fully-fledged web-based user interface that acts as a gateway between users and the technologies developed within SOAAll. It is composed of three platforms which are in charge of supporting the three main phases within the lifecycle of services, namely their provisioning, consumption and analysis. The main goal of the Studio is therefore to assist users in the creation of semantic annotations over web services, as well as in their discovery, invocation and analysis.

In order to simplify the creation of semantic web services the Provisioning Platform described in this deliverable will provide a set of tools allowing users to i) find relevant services taking into account user profiles and previous history of service usage, ii) annotate them helped by a recommender system and iii) persist the resulting semantic web services so that they can be used by anyone. Therefore in this respect, the Provisioning Platform aims to leverage users as the main source of information using interchangeably direct user input and automated processing informed by prior user-provided information to simplify the annotation of services.

Based on the semantic web services modelled by users, the Provisioning Platform will support defining composite semantic web services. To this end, it includes a Process Editor with which users can put together existing semantic web services in novel forms giving rise to new and more complex services. The Process Editor will allow people to create mash-ups, that is composite services that have the control flow implicit in the data-flow. This kind of composite service will therefore allow any kind of user to create relatively complex services in an easy way in a similar vein to that of Yahoo! Pipes [1] for example. Additionally, the Process editor will also address the requirements of more advanced users that may need to create complex workflows, i.e., using explicit control-flow constructs.

Essential to the overall vision of the Provisioning Platform is the central role played by users during the overall life-cycle of services. Users are at the same time service providers, when they define or compose new services, service consumers, when they utilize services or compose new services out of existing ones, and knowledge providers and/or consumers when they annotate or simply use services. An important aspect of this vision is that it blurs the distinction between providers and consumers which rather coexist in a transparent way allowing, for instance during the composition of services, the same individual to play both roles so that the newly created service can automatically become available for others. As a result, the overall platform benefits from an ever-growing repository of services with increasingly richer annotations provided by users making the provisioning platform an extremely rich and dynamic semantic web services provisioning platform.

1. Introduction

In this section we describe the overall content, purpose and scope and structure of this deliverable and describe the alignment of the Provisioning Platform with the project architecture and the use-cases contemplated in the project.

1.1 Introductory Explanation of the Deliverable

Service-orientation is increasingly being adopted to allow applications to be created flexibly by linking loosely-coupled web services. Standards for describing web services currently use syntactic (XML-based) notations such as Web Services Description Language (WSDL) [2]. These descriptions are not amenable to applying automated reasoning, so specialist workers must be involved throughout the web service lifecycle. This causes numerous problems, the most significant of which is the lack of scalability. It simply isn't possible to maintain millions, let alone billions, of services to cope with environmental and context changes, discover new services, compose them or support their adaptation at runtime through human effort alone.

These limitations inherent to traditional web services technologies such as WSDL and UDDI [3] have limited a wider spread of service-orientation as initially predicted. Large organisations such as Verizon already have systems based on approximately 1,500 web services [4]. However, the web contains only around 27,000 web services based on WSDL [5]. In consequence, service-oriented architecture (SOA) is largely still an enterprise-specific solution exploited by, and located within, large corporations.

Researchers studying the semantic web, semantic web services and more traditional artificial intelligence have shown it is possible to automate some of these tasks to a significant extent (see, for instance, work in OWL-S [6], WSMO [7], WSDL-S [8], semantic execution environments [9, 10] or planning [11]). However, the complexity of the descriptions required represents a significant bottleneck and in fact the application of semantic web services is still largely limited to research environments where a set of experts provide the necessary descriptions. Therefore, in order to achieve a wider application of semantic web services technologies and in consequence of web services in general, we need to provide the adequate mechanisms able to reduce the overhead for providing expressive semantic descriptions for web services.

SOAAll proposes the use of lightweight semantic descriptions as a means to simplify the modelling of semantic web services at the same time that it reduces the computational complexity for performing certain activities automatically in a scalable manner. In this deliverable we provide the design and specification of the Provisioning Platform, a component of the so-called SOAAll Studio which aims to support users in the provisioning of semantic web services descriptions[1]. The Provisioning Platform is based upon Web 2.0 principles aimed at better supporting users in the collaborative yet seamless provisioning of semantic descriptions, assisted by a non intrusive set of technologies such as an annotations recommender or wizards.

The remainder of the document is organised as follows. In Section 2 we present the main requirements for the Provisioning Platform and, based on these, we describe the overall approach we have adopted. In Section 3 we present a detailed design of the Provisioning Platform, placing it in the context of the SOAAll Studio and in the project architecture in general. Finally we cover related work in the area and conclude the deliverable.

[1] In this document we shall refer to semantic web service as a particular self-contained semantic annotation over an existing web service.

1.2 Alignment with the Architecture

In this section we describe the alignment of the Service Provisioning Platform with the SOAAll architecture currently being devised in WP1. Given that the Service Provisioning Platform is a component of the SOAAll Studio, most of the details regarding its integration and alignment with the project vision are common to that of the SOAAll Studio. Still, where necessary we shall mention the specific details affecting the Service Provisioning Platform.

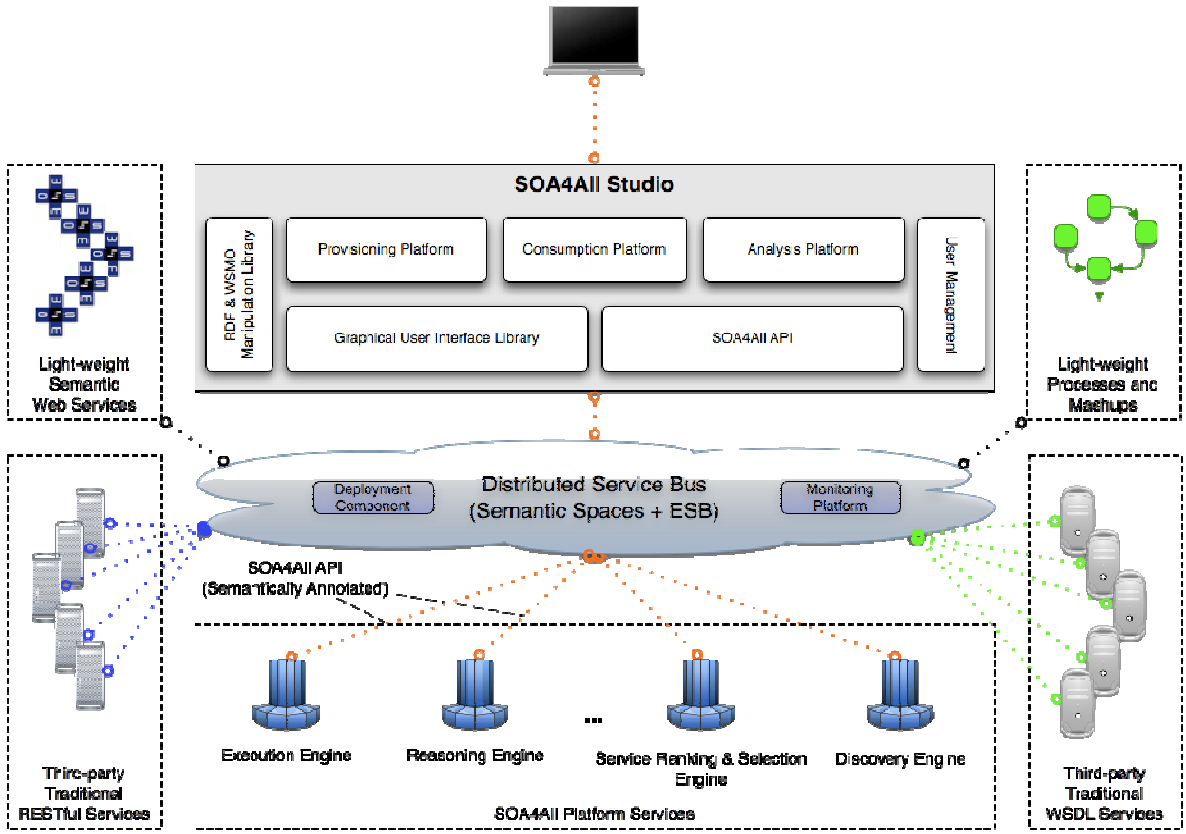

Figure 1. SOAAll Architecture.

Figure 1 provides a high-level view on the overall SOAAll architecture centered on the infrastructural components, the main artifacts manipulated and how the SOAAll Studio is integrated. Integration in SOAAll takes place through the so-called Distributed Service Bus (DSB) which integrates notions from Enterprise Service Buses and Semantic Spaces into a unified distributed infrastructure providing message-based and event-based communication, as well as a distributed RDF repository. From a client perspective, the DSB provides access to the platform services as well as the myriad of services provide by third parties. The services offered through the bus include, traditional third-party WSDL and RESTful services, light-weight semantic Web services based on annotations over traditional services, and infrastructural services supporting actions such that the discovery, ranking & selection and execution of services.

In a nutshell, the SOAAll Studio is the gateway for the user. It therefore provides a Web-based interface for creating or enhancing annotations, browsing them, discovering suitable services, invoking services, and finally analyzing their execution. These different functionalities are provided by three different platforms composing the Studio. In particular, the Service Provisioning Platform supports the user in providing annotations may them be, WSMO-Lite, MicroWSMO, tags or ratings. The Service Consumption Platform allows the

user to browse, discover and invoke existing services. Finally the Service Analysis Platform provides the means for users to analyze the execution of services either at runtime or post-execution.

The SOAAll Studio, like every other component is integrated into the overall architecture through the Distributed Service Bus. It is worth noting however that as opposed to platform services that provide infrastructural services, the Studio is mainly a client for these different components, allowing the user to interact with them in a seamless and transparent way. To support this, there is an internal component within the Studio in charge of supporting the interaction with the bus by making use of the appropriate messages and protocols [12]. The concrete formats and protocols are currently being established within WP1 but they will be based on Web and WS-* standards.

The DSB also provides a scalable RDF storage & querying system. In particular, every annotation provided by the user through the Service Provisioning Platform will ultimately be stored for future use in the DSB. Similarly, raw monitoring data concerning the execution of services, like the message exchanges, will be stored in the DSB for further reference. Additionally, in order to support monitoring the execution of services—simple and composite—the DSB provides a notification mechanisms such that applications can register themselves as observers [13] for certain events of interest. Whenever any of these events occur, the DSB—thanks to the templates binding mechanism provided by Semantic Spaces—notifies any application interested.

The Service Provisioning Platform aims to better support and simplify the definition of semantic web services by creating annotations over existing services or supporting the composition of existing semantic web services. To this end the Provisioning Platform includes a set of intuitive Web-based modeling tools and supporting components such as an Annotations Recommender. In order to achieve this goal the Provisioning Platform builds upon a number of technologies devised in other work packages. In particular, it uses the languages defined in WP3 for annotating existing services, it utilizes the language devised in WP6 for supporting the creation of composite services, it interacts with the semantic web services registry and the crawler registry for retrieving existing services, and it uses the runtime infrastructure for supporting the communication with platform services as for the scalable storage of RDF information. More details on how these interactions take place shall be given in Section 3.

1.3 Alignment with the use cases

In this section, we cover the different use cases contemplated within SOAAll and identify the main aspects where the Service Provisioning Platform will play an important role.

1.3.1 End-user Integrated Enterprise Service Delivery Platform

Public administrations nowadays have to deal with hundreds of different procedures (e.g., for handling a parking permit request) that are typically implemented in one or more legacy systems or even executed manually. WP7 envisions an open and flexible service delivery platform where administrative procedures are handled over a central Internet portal as single point of contact. Administrative procedures are composed of semantic web services, including so-called enterprise services that are provided by SAP and that offer business functionality such as the management of resources and customer orders. These services can be combined in different ways so that new procedures can be created or existing ones can be adapted easily. A key element for making this service delivery platform efficient and scalable is enabling end (or business) users, i.e., the vaste majority of employees in public administrations (and other organizations), to carry out these tasks. Such business users have a detailed understanding of the procedures in their field of expertise but lack the specific programming skills required nowadays to actively consume and compose web services. The SOAAll approach therefore is to provide the end users with simple web-based

tools on top of semantic web services so that they can search, model, annotate, modify, share, analyze, and execute administrative procedures on the basis of web services.

The main requirements of this use case are listed in detail in [14]. These requirements are divided into business-oriented and technology related. The Provisioning Platform shall play a crucial role in fulfilling many of these requirements. From a business perspective the role of the Provisioning Platform will be as follows:

- **"Providing single points of contact for the constituents with information transparency"**

  The Provisioning Platform as a component of the SOAAll Studio is part of the global user interface used for SOAAll technologies. As such it contributes from the modeling perspective towards having a unique entry point.

- **"Establishing electronic procedures and electronic document exchange with a high degree of automation"**

  The Provisioning Platform will support users in defining processes thus contributing to the further automation of the administrative procedures.

- **"Faster and cheaper administrative procedures"**

  Given that the Provisioning Platform is envisioned such that any user can utilize it, the number consultants to be contracted for modeling processes will be reduced which will in turn have an important impact in costs and agility.

- **"Better reuse of services and processes"**

  The Provisioning Platform allows users to create composite services out of existing ones in a simple way, thus promoting reuse.

- **"Usability of the electronic tools with a focus on end users with limited technical experience"**

  One of the main goals of the SOAAll Studio is to support all kinds of users.

From a technical perspective, the role of the Provisioning Platform will be as follows:

- **"Directory that lists existing processes and services and allows adding new ones. The directory should cope with semantic annotations and context information"**

  The Provisioning Platform provides an interface for retrieving existing services based on their annotations, tags, etc. Information about users will support listing information based on the potential relevance.

- **"Search interface allowing for the use of rich search parameters according to semantic annotations and context information"**

  The Provisioning Platform will provide a search interface by means of which users will retrieve services based on annotations, tags, ratings, etc.

- **"Graphical modeling tool for creating and adapting services and processes"**

  The Provisioning Platform provides graphical user interfaces for modeling both simple (WSMO Lite, MicroWSMO) and composite (processes and mash-ups) services.

- **"Textual editor for rating, commenting (free text), and semantically annotating (free text or following an existing taxonomy/ontology) processes and services"**

  The platform includes means for users to provide annotations, tags,

comments and ratings in such a way that the information can be used by the platform itself to improve its behaviour.

From a more technical perspective, Table 1 presents the relevance each of the functionalities described in this deliverable will have in the use case as reported by the use-case partners. In particular the table shows how likely it will be within WP7 to reuse the different use-cases devised in the Provisioning Platform.

Table 1. Alignment of the Provisioning Platform with WP7.

Use Case Name	Brief Description	Detailed Content	Likelihood of Being Used
Find Service By Query	Use simple textual search for retrieving relevant crawled services in order to annotate them eventually	See Section 3.3.1	Sure
Retrieve Services By Tag	Obtain a list of crawled web services that are related to a tag	See Section 3.3.1	Sure
Retrieve Services By Category	Obtain a list of crawled web services that are classified as relevant to a concrete category	See Section 3.3.1	Sure
Retrieve Service Information	Retrieve detailed information about a given service. The information retrieved is based on the next 5 use cases	See Section 3.3.1	Sure
Retrieve Service Definition	Retrieve the service definition (WSDL or REST) file	See Section 3.3.1	Mid
Retrieve Service Documentation	Retrieve the relevant documents crawled about the given service	See Section 3.3.1	High
Retrieve Service Statistics	Retrieve the statistics (performance, usage, etc) about a service	See Section 3.3.1	High
Retrieve Service Ratings	Retrieve ratings given by users to the service	See Section 3.3.1	Sure
Retrieve Service Reviews	Retrieve reviews by users on the given service	See Section 3.3.1	Sure
Recommend Service Annotations	Assist the user in annotating a service by recommending suitable annotations on different granularity levels (domain, classification and individual properties annotations)	See Section 3.3.3	Sure
Retrieve Service Tags	Retrieve tags associated to the service by users	See Section 3.3.5	Sure
Rate item (Service/Goal)	Assign a rating produced by a user to a Service or Goal	See Section 3.3.5	Sure
Review item (Service/Goal)	Assign a review produced by a user to a Service or Goal	See Section 3.3.5	Sure
Tag item (Service/Goal)	Assign a tag produced by a user to a Service or Goal	See Section 3.3.5	Sure
Use wizard to annotate Service	Use a wizard in order to create a Semantic Web Service	See Section 3.3.4	Sure

1.3.2 W21C BT Infrastructure

This use case will create a semantically enhanced and expanded version of BT's Web21c platform, which will result in a framework for the delivery of services, both by BT itself and third parties. This platform aims to provide user-friendly facilities for advanced service discovery, composition, consumption and monitoring using the existing Web21c infrastructure exposing core BT communication capabilities. The first scenario is about designing a simple application by composing existing services, and the second scenario is about enabling Bulk Resellers to implement innovative business ideas using unbranded services provided by BT, or to integrate said services into their business processes.

The main requirements of this use case are listed in detail in [15]. These requirements are divided into business-oriented and technology related. The Provisioning Platform shall play a crucial role in fulfilling many of these requirements. From a business perspective the role of the Provisioning Platform will be as follows:

- **"Encourage greater uptake of Web21c SDK by providing tools to enable easier use of the services"**

The SOAAll Studio aims to support any kind of user in modeling, consuming and analyzing services, independently form their background and expertise. The Provisioning Platform itself is in charge of supporting and simplifying the modeling of semantic web services.

- **“Provide tools with a focus on end users with limited technical experience”**

  As previously explained this is one of the main goals of the platform.

- **“Create a community of SDK users, to encourage collaboration and innovation in creating new Telco applications”**

  The SOAAll Studio will support the creation of communities around services, allowing users to collaboratively and opportunistically model, consume, enhance, and reuse services for novel purposes.

- **“Create an infrastructure to allow a third party business to resell BTs SDK services, providing support in design and management of the thirds party services”**

  Given the community-oriented aspect of the SOAAll, any individual (end-users or companies) can seamlessly create new added-value services our of existing ones.

- **“Increase overall use of the Web21c SDK, and hence increase revenue”**

  The previous aspects will contribute to fulfilling this requirement.

From a technical perspective, the role of the Provisioning Platform will be as follows:

- **“Usable without detailed knowledge of Ontologies, WSDL or programming languages”**

  As mentioned earlier the SOAAll Studio aims to support any kind of user in modeling, consuming and analyzing services, independently form their background and expertise.

- **“Web Browser Based, Drag and drop, easy to use interface”**

  The Provisioning Platform will make use of state-of-the-art technologies for creating highly dynamic Web-based user interfaces, see [12]

- **“Search on functional and non-functional aspects of service, and keywords”**

  The Provisioning Platform provides an interface for retrieving existing services based on their annotations, tags, etc.

- **“Semi automated assistance with creating service compositions, with facilities to link a GUIs”**

  The Provisioning Platform includes a Process Editor, see [16], and assistants for guiding users in the creation of composite services.

- **“Service ranking and help with design time selections based on user context”**

  The Provisioning Platform will support users in rating, commenting and tagging services. This information as well as user profiles will support listing information based on the potential relevance.

- **“Import and link to Industry standard ontologies”**

  The Provisioning Platform includes support for finding existing ontologies on the Web and using them for tagging. User-provided ontologies will also be usable.

- **"Import and mark-up third party Web Services"**

  The Provisioning Platform will provide access to the services found by the service crawler and will support users in annotating them.

From a more technical perspective, Table 2 presents the relevance each of the functionalities described in this deliverable will have in the use case as reported by the use-case partners. In particular the table shows how likely it will be within WP8 to reuse the different use-cases devised in the Provisioning Platform.

Table 2. Alignment of the Provisioning Platform with WP8.

Use Case Name	Brief Description	Detailed Content	Likelihood of Being Used
Find Service By Query	Use simple textual search for retrieving relevant crawled services in order to annotate them eventually	See Section 3.3.1	Sure
Retrieve Services By Tag	Obtain a list of crawled web services that are related to a tag	See Section 3.3.1	High
Retrieve Services By Category	Obtain a list of crawled web services that are classified as relevant to a concrete category	See Section 3.3.1	Sure
Retrieve Service Information	Retrieve detailed information about a given service. The information retrieved is based on the next 5 use cases	See Section 3.3.1	Sure
Retrieve Service Definition	Retrieve the service definition (WSDL or REST) file	See Section 3.3.1	Mid
Retrieve Service Documentation	Retrieve the relevant documents crawled about the given service	See Section 3.3.1	Mid
Retrieve Service Statistics	Retrieve the statistics (performance, usage, etc) about a service	See Section 3.3.1	Sure
Retrieve Service Ratings	Retrieve ratings given by users to the service	See Section 3.3.1	Sure
Retrieve Service Reviews	Retrieve reviews by users on the given service	See Section 3.3.1	Sure
Recommend Service Annotations	Assist the user in annotating a service by recommending suitable annotations on different granularity levels (domain, classification and individual properties annotations)	See Section 3.3.3	Sure
Retrieve Service Tags	Retrieve tags associated to the service by users	See Section 3.3.5	High
Rate item (Service/Goal)	Assign a rating produced by a user to a Service or Goal	See Section 3.3.5	High
Review item (Service/Goal)	Assign a review produced by a user to a Service or Goal	See Section 3.3.5	Mid
Tag item (Service/Goal)	Assign a tag produced by a user to a Service or Goal	See Section 3.3.5	High
Use wizard to annotate Service	Use a wizard in order to create a Semantic Web Service	See Section 3.3.4	Sure

1.3.3 C2C Service eCommerce

The WP9 use case focuses on one holistic and real-world oriented C2C eCommerce scenario, the development of an eCommerce framework for ISPs. It is entirely focused on providing an easy way for end users to use third party services offered through the framework, enabling them to build eCommerce applications. Potential users of such a C2C eCommerce framework, such as the customers of an ISP, will normally not search for individual services (e.g. a specific credit card approval service), but for complete solutions to address their business needs (e.g. services to provide a certain kind of payment functionality for the Web shop). That is why the platform must be capable to support very different application scenarios, while at the same time it should remain generic enough so that the same set of common components can be (re-)used to run each of such scenarios.

The main requirements of this use case are listed in detail in [17]. These requirements are divided into business-oriented and technology related. The Provisioning Platform shall play a crucial role in fulfilling many of these requirements. From a business perspective the role of

the Provisioning Platform will be as follows:

- **“Scalability”**

  The Provisioning Platform aims to support the SOAAll vision were billions of users create, invoke and analyse billions of services. Scalability is therefore one aspect that is considered of main relevance in the SOAAll Studio, see [12].

- **“Intuitive Composition of Services”**

  One of the main goals of the Provisioning Platform is to support users in creating composite services by means of the lightweight process modelling language. The Process Editor part of the platform will be in charge of this [16].

- **“Security”**

  The SOAAll Studio will integrate authentication mechanisms.

- **“Privacy and trust model”**

  Users profiles and history will remain private and shall only be used for suggesting annotations or services. Concerning trust, although this is not a central aspect of the platform, users will be able to rate and evaluate services which will generate models about how trustworthy services are in a way similar to that of eBay.

- **“Reliability”**

  The SOAAll Studio integrates support for analysing services. This information will be accessible to the Provisioning Platform when presenting services to the user, so that reliability, performance and other aspects can be taken into account by the user.

- **“Extendibility”**

  The main idea behind the SOAAll Studio is to support users to seamless collaborate in the gradual provisioning of new services, the enhancement of the annotations over existing ones, or the composition of new solutions out of existing services. The amount of services available and the quality of their annotations shall therefore be continuously increasing.

From a technical perspective, the role of the Provisioning Platform will be as follows:

- **“The end user frontend MUST feature similar functionalities as the commercial software Sitebuilder”**

  The Provisioning Platform will provide all the functionalities required for annotating services. The SOAAll Studio is however a modular tool-suite which has a core set of libraries for integration and user-interface. Third parties can extend the software as appropriate in order to deal with use case specific requirements.

- **“Templates for the core parts of eCommerce applications (of different complexity) MUST be available”**

  The Provisioning Platform makes use of the notion of processes and abstract processes within the Process Editor which cater for this.

- **“Templates SHOULD describe the parts of an eCommerce application in the form of typical workflows”**

  The graphical representation that will be used is strongly based in BPMN

which is a widely used workflow notation.

- **"Templates for layouting the eCommerce application SHOULD also be available"**

  The SOAAll Studio makes use of widgets and CSS for layout which allow very easily to customize the user interface. Third parties can therefore adapt it to their own needs and preferences.

- **"The frontend SHOULD allow for integration with the remainder of the web site generation tools used"**

  SOAAll Studio makes use of standard Web-based technologies and can therefore be integrated within other Web sites.

From a more technical perspective, Table 2 presents the relevance each of the functionalities described in this deliverable will have in the use case as reported by the use-case partners. In particular the table shows how likely it will be within WP8 to reuse the different use-cases devised in the Provisioning Platform.

Table 3. Alignment of the Provisioning Platform with WP9.

Use Case Name	Brief Description	Detailed Content	Likelihood of Being Used
Find Service By Query	Use simple textual search for retrieving relevant crawled services in order to annotate them eventually	See Section 3.3.1	High
Retrieve Services By Tag	Obtain a list of crawled web services that are related to a tag	See Section 3.3.1	High
Retrieve Services By Category	Obtain a list of crawled web services that are classified as relevant to a concrete category	See Section 3.3.1	High
Retrieve Service Information	Retrieve detailed information about a given service. The information retrieved is based on the next 5 use cases	See Section 3.3.1	High
Retrieve Service Definition	Retrieve the service definition (WSDL or REST) file	See Section 3.3.1	High
Retrieve Service Documentation	Retrieve the relevant documents crawled about the given service	See Section 3.3.1	Low
Retrieve Service Statistics	Retrieve the statistics (performance, usage, etc) about a service	See Section 3.3.1	High
Retrieve Service Ratings	Retrieve ratings given by users to the service	See Section 3.3.1	Medium
Retrieve Service Reviews	Retrieve reviews by users on the given service	See Section 3.3.1	Medium
Recommend Service Annotations	Assist the user in annotating a service by recommending suitable annotations on different granularity levels (domain, classification and individual properties annotations)	See Section 3.3.3	Low for Enduser / High for Service Provider or Service Broker
Retrieve Service Tags	Retrieve tags associated to the service by users	See Section 3.3.5	Medium
Rate item (Service/Goal)	Assign a rating produced by a user to a Service or Goal	See Section 3.3.5	Medium
Review item (Service/Goal)	Assign a review produced by a user to a Service or Goal	See Section 3.3.5	Medium
Tag item (Service/Goal)	Assign a tag produced by a user to a Service or Goal	See Section 3.3.5	Medium
Use wizard to annotate Service	Use a wizard in order to create a Semantic Web Service	See Section 3.3.4	Low for Enduser / High for Service Provider or Service Broker

2. Simplifying the Creation of Semantic Web Services: Requirements and Vision

In this section we briefly revisit the vision of the project in order to illustrate the main motivations behind the work described in this deliverable. Based on this motivation, we introduce the main requirements the Provisioning Platform should address and we finally present the overall vision on how the SOAAll Provisioning Platform will tackle them. Concrete details on how this shall be achieved will be introduced in Section 4.

2.1 Requirements

Service-Oriented Architecture (SOA) is commonly lauded as a silver bullet for Enterprise Application Integration, inter-organizational business processes implementation, and even as a general solution for the development of all complex applications. Although technology independent, SOA is typically implemented using Web services related technologies such as WSDL, SOAP and WS-BPEL [18]. However, despite the well-known benefits of SOA and web services technologies, currently, they are mostly applied within enterprises which limits to an important extent the benefits that can be obtained by their systematic application as well as it fails to achieve the economic and technological impact that was initially predicted.

The main limitation exhibited by traditional web services technologies is the low level of automation that can be achieved for key activities within the lifecycle of web service-based applications such as the discovery or composition of services. To cater for this semantic web services technologies enhance services with semantic descriptions that are amenable to automated reasoning, thus paving the way for the application of knowledge-based algorithms to better support the automation of service discovery, composition and execution. Still, obtaining rich enough semantic descriptions that can enable this requires an important knowledge-acquisition effort which represents the main bottleneck for a wider and systematic application of semantic web services technologies.

SOAAll proposes to use lightweight semantic descriptions of services in an attempt to reduce both the labour necessary for annotating services, as well as for reducing the computational complexity for processing these descriptions. Additionally, given the widespread use of RESTful services, SOAAll, as opposed to most research in the area of semantic web services, also advocates the integration of RESTful services provided through Web APIs in the services landscape. These two decisions underpinning SOAAll are crucial for a greater uptake of service-oriented technologies. On the one hand, this can increase the level of automation that can be achieved throughout the lifecycle of services and service-based applications while minimising the knowledge-acquisition bottleneck. On the other hand, this allows us to integrate new sources of services coming out of existing Web 2.0 solutions. Figure 1 shows the usage of each as reported by ProgrammableWeb [19], a website that maintains a comprehensive directory of web Application Programming Interfaces (APIs) [20] and the technologies(s) they support.

Although these are highly valuable and probably even necessary steps, it is still necessary to promote and better support the creation of new services in a sustainable way. This requires among other things, techniques for:

- Simplifying the creation of semantic web services out of existing services;
- Simplifying the creation of composite semantic web services by composing existing semantic web services; and
- Supporting the adaptation of services to unforeseen contexts

The first two will provide means for increasing the number of available semantic web services descriptions that can directly be discovered, used and composed by users and

applications. The third will increase the applicability of existing services and it will presumably improve the quality of service provided to end users which will in turn contribute to the wider uptake of service technologies. The Provisioning Platform described herein aims to address all three aspects taking into account use-case specific requirements as presented earlier. The first two will be covered by applying Web 2.0 principles that have significantly contributed to increasing the amount and quality of the information in the Web. The last one shall be covered by allowing users to make use of the adaptation mechanisms defined in WP6 as well as the context ontologies being devised in WP3 through an easy to use graphical user interface.

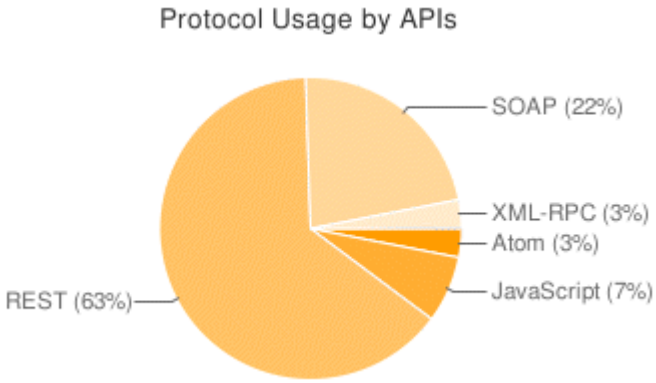

Figure 2. Technologies usage for Published Web APIs.

In addition to the functional requirements listed above, the Provisioning Platform needs address some technical ones derived from the functional ones as well as from global technical and architectural decisions. In particular the Provisioning Platform needs:

R1. Compliance with WSMO dialects defined in the project;

R2. Reuse of existing knowledge models;

R3. Persistent and holistic knowledge storage;

R4. Adaptability to distinct contexts;

One major decision made in the project is the use of WSMO-Lite and MicroWSMO for annotating services. In turn, both WMSO-Lite and MicroWSMO are part of the set of formalisms set out by WSMO which therefore represent the main technologies the Provisioning Platform needs to be compliant with. In particular, the platform shall be able to generate WSMO-Lite and MicroWSMO definition which will point to existing semantic models.

These semanic models will define classification hierarchies, functional and non-functional aspects of services using some ontology definition language. Although both WSMO-Lite and MicroWSMO annotations are language independent in this respect, WP3 will define a set of languages and the corresponding machinery for reasoning with them. The Provisioning Platform shall therefore be able to use these languages within the semantic web services descriptions.

Defining semantic web services requires among other things linking the inputs and outputs of the services to domain-specific ontologies. It will therefore be necessary to allow users to use existing knowledge models when defining their annotations. This will allow users to focus on the actual annotation of services, leaving the creation of domain models aside, and supporting the reuse of models defined by others and available on the Web. Doing so will on the one hand minimise the complexity and labour required for creating services and on the other hand it will presumably maximise the potential for integrating services developed

independently by using the same common domain ontology.

The Provisioning Platform should support the persistence of semantic annotations for services such that they can be retrieved and applied efficiently both by humans and machines. To this end the Provisioning Platform should provide a transparent interface over the storage infrastructure devised in SOAAll.

Given the broad vision of SOAAll, it is important that we support a wide variety of users. Semantic Web services as well as the Provisioning Platform should be adaptable to specific contexts so that the execution can take into account user preferences, location, etc, and the information presented can be organised in terms of its relevance and suitability. To this end the Provisioning Platform should leverage the technologies defined in other WPs (e.g., WP6).

2.2 Vision: Collaborative Provisioning of Services

In order to simplify the creation of semantic web services the Provisioning Platform will provide a set of tools allowing users to i) find relevant services taking into account user profiles and previous history of service usage, ii) annotate them helped by a recommender system and iii) persist the resulting semantic web services so that they can be used by anyone. Therefore in this respect, the Provisioning Platform aims to leverage users as the main source of information using interchangeably direct user input and automated processing informed by prior user-provided information to simplify the annotation of services.

In addition to the semantic annotations provided by users, the SOAAll Studio will gather information about users behaviour in order to create profiles for providing information that is presumably more relevant to the users. For instance, knowing that user has typically used telecommunication services, the Provisioning Platform can order the services available so that telecommunication services, or services that were used by other people with a similar profile, appear first. Similarly, user ratings, comments or tags will be gathered by the Provisioning Platform, thus enriching in a somewhat transparent way the information available about services.

Based on the semantic web services modelled by users, the Provisioning Platform will support users in defining composite semantic web services. A Web-based Process Editor will provided so that users can put together existing semantic web services in novel forms giving rise to new and more complex services. The Process Editor will allow users to create composite services that have a simple data-driven workflow which we refer to as mash-ups [21]. This kind of composite service will therefore allow any kind of user to create relatively complex services in an easy way in a similar vein to that of Yahoo! Pipes [1] for example. On the other hand the Process Editor will also address the requirements of more advanced users that may need to create complex workflows including control-flow constructs [21]. In a nutshell the editor will provide an intuitive user interface for generating process models supported by the lightweight process definition language devised in WP6 [22].

Essential to the overall vision of the Provisioning Platform is the central role played by users during the overall life-cycle of services. Users are at the same time service providers, when they define or compose new services, service consumers, when they utilize services or compose new services out of existing ones, and knowledge providers and/or consumers when they annotate or simply use services. An important aspect of this vision is that it blurs the distinction between service providers and service consumers which rather coexist in a transparent way allowing, for instance during the composition of services, the same individual to play both roles so that the newly created service can automatically become available for others. As a result, the overall platform benefits from an ever-growing repository of services with increasingly richer annotations provided by users making the provisioning platform an extremely rich and dynamic semantic web services modelling platform.

3. Service Provisioning Platform Design

In this section we shall present the design of the Service Provisioning Platform. We first present the architecture overview, placing the platform in context both within the SOAAll Studio and within the project. We then cover briefly the lifecycle of services in order to better illustrate some of the decisions that have been adopted within the platform and finally we describe each of the components in detail.

3.1 Architecture Overview

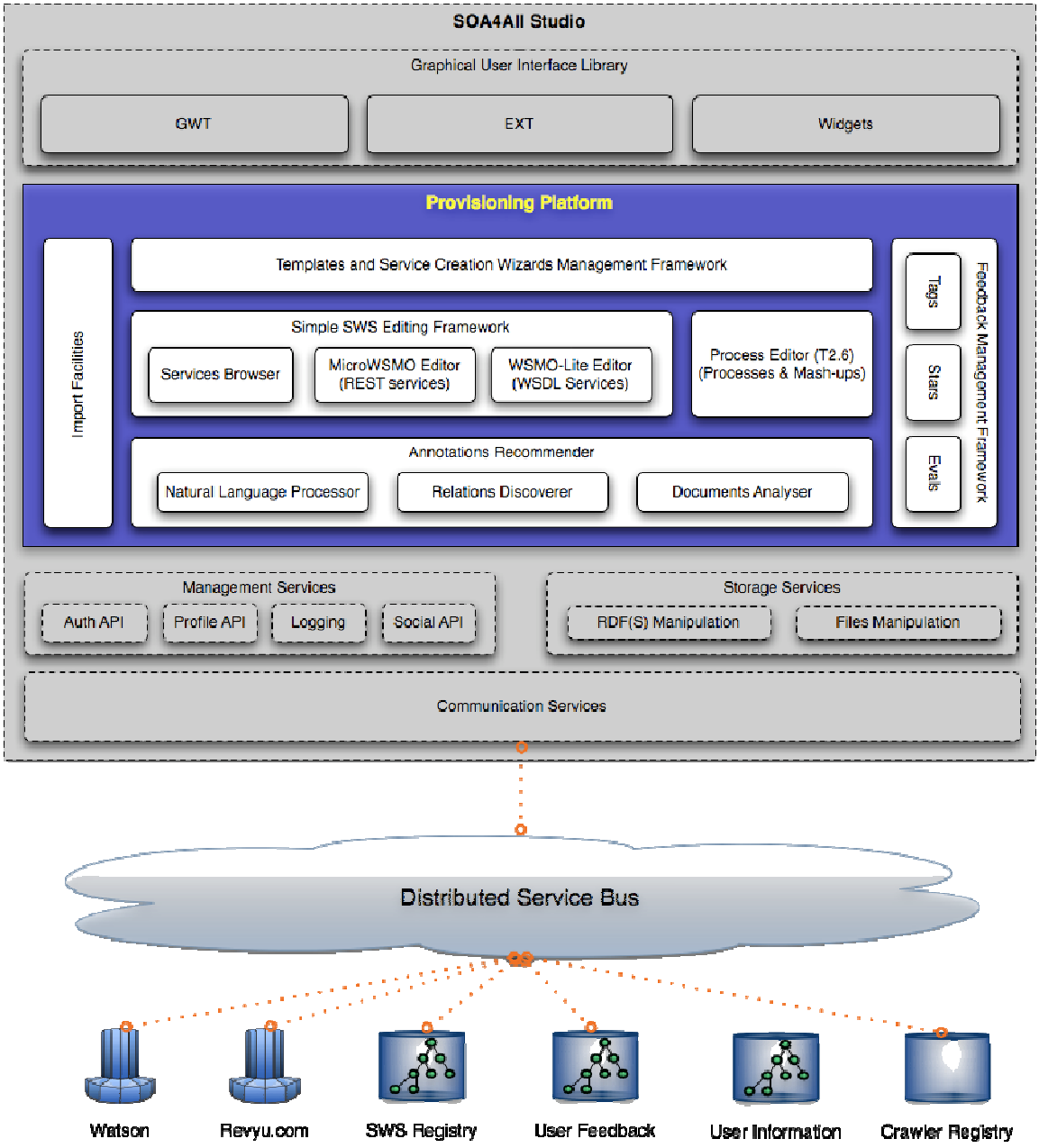

Figure 3. Service Provisioning Platform Architecture.

We previously introduced the overall vision of the project focussing particularly on those aspects that are of most relevance to the SOAAll Studio. The Studio is a holistic Web-based tool covering the whole life-cycle of services, i.e., from their definition to post-execution analysis. The Service Provisioning Platform is the component of the SOAAll Studio that

supports the user in creating compositions of semantic web services, providing semantic annotations for existing services using WSMO-Lite or MicroWSMO, and attaching lighter annotations such as tags, ratings and comments. It is therefore a key component for allowing the integration of users as the most valuable source of information, thus enabling the social and collaborative aspects of Web 2.0 technologies.

The architecture of the Service Provisioning Platform is depicted in Figure 3 including the platform itself (as a blue box), the underlying support provided by the SOAAll Studio infrastructure (as grey boxes), and finally external services and repositories that will be used by the platform (see components connected to the DSB). It is worth noting in this respect that although the repositories are depicted as separate logical components, in practice storage support will be provided globally by the Distributed Service Bus thanks to the use of Semantic Spaces. They are here depicted in this way to distinguish the main kinds of artefacts that will be manipulated and generated by the Service Provisioning Platform.

The core of the Service Provisioning Platform is concerned with supporting the creation of semantic Web services (SWS). The platform distinguishes between two main classes of semantic Web services: *simple* and *composite*. We refer as simple semantic Web services to those that are directly created by attaching semantic annotations to existing traditional services. We therefore contemplate in this group the annotation of WSDL-based services [23] and RESTful services [24]. WSDL-based services, typically employed within enterprises, are described in a WSDL file including information about their operations, inputs and outputs, etc. Annotations to this kind of services will be supported through a form-based editor by making use of SAWSDL [25] for extending WSDL files with annotations based on WSMO-Lite [26]. On the contrary, RESTful services, more frequently used by Web-based systems, are typically described in natural language in some Web page. The annotation of RESTful services will therefore be supported by an editor able to attach annotations to the Web pages themselves. To this end, the Service Provisioning Platform will extend PowerMagpie [27, 28] in order to support the annotation of web pages describing RESTful services using microformats, namely hRESTS [29] and MicroWSMO which will be defined in WP3.

The other kind of semantic Web services contemplated—*composite* semantic Web services—are those composed of two or more services. In this kind of services we contemplate lightweight processes or mash-ups defined using the lightweight process modelling language devised in WP6 [22]. The main difference between both kinds of services lies on the fact that mash-ups do not include any control flow other than the one implicit in the data flow, i.e., when the input data is available the service can be executed. Mash-ups are a concept introduced relatively recently in the context of Web 2.0 technologies by means of which people can make use of data available on the Web (often through RESTful services) and mash it up in order to produce new aggregated information and visualization services.

The Service Provisioning Platform will support the creation of semantic mash-ups and processes making particular emphasis on simplifying their creation and maximizing the level of automation and adaptability in their execution. Once these composite services are defined they will be deployed in the SOAAll runtime infrastructure becoming themselves brand-new services that can be discovered, invoked, analysed or even used within another composite semantic web service. This recursive nature of processes and mash-ups will therefore contribute to a big extent to supporting the proliferation of millions of services in a collaborative way based on users interactions.

The support for creating both simple and composite semantic Web services will try and maximise the uptake of our technologies by simplifying the creation of annotations and services compositions. Additionally, the Service Provisioning Platform provides an Annotation Recommender underlying the editing environment, and a component including means for managing service templates and wizards for guiding and simplifying whenever possible the creation of semantic Web services.

The Annotations Recommender will support the semi-automated annotation of services by accessing crawled data about services, related web pages, and documentation. The information will be processed in order to identify relevant concepts concerning their services, their inputs and outputs, etc. Using the information obtained, the Annotations Recommender will finally try and identify relevant concepts and properties from existing ontologies on the Web and suggest them as annotations to the user. In this process, the Annotations Recommender will make use of services provided by WP5, like the Crawler Registry, as well as external services such as Watson [28] for finding ontologies on the Web and possibly Scarlet [30] for determining relations.

The Service Provisioning Platform will also enable ways for users to perform the same tasks with additional help in the form of "Wizards". We believe that introducing helpful guidance to the new users, the use of the platform will be simplified. The Wizards framework will be completely integrated within the Service Provisioning Platform, so that every step performed within a wizard will directly trigger actions in the platform, allowing the user to perform semantic annotations independently from his or her expertise.

In addition to the support for creating semantic Web services, the Service Provisioning Platform will allow users to provide lighter annotations over services through tagging, rating, etc. Tagging will allow users to quickly attach some words that are considered relevant for describing the service, in a similar vein to that of Web 2.0 sites like Flickr [31] or Delicious [32]. Additionally users will be provided the means for providing feedback about services. We hereby consider therefore the possibility for users to comment on services and rate them using the Review vocabulary [33]. This will pave the way for profiling users based on their preferences and enhancing users' experience with more adequate services while using a vocabulary compatible with existing systems such as Revyu.com [34] and therefore ready to be integrated with the Linked Data initiative [35].

Finally, given the amount of ontologies, services and, to a lesser extent, semantic Web services descriptions available on the Web, the Provisioning Platform will also provide the means for importing existing data and if necessary translating the data into formats that can be processed by the SOAAll tools, e.g., RDF/RDFS, WSML, SAWSDL, WSMO-Lite, etc. To this end we shall rely on existing software libraries for parsing and transforming the documents, as well as on external services such as Watson [28] for locating and reusing existing ontologies.

3.2 On the lifecycle of Semantic Web Services

The vision outlined in this deliverable and pursued within the project is one where billions of users will be collaborating in providing and utilizing web services. Users, may they be companies or individuals, collectively contribute to the creation of web services, their annotation, their composition, their evaluation in a seamless and autonomous way. Conversely, users will also engage in the consumption of existing services by invoking existing semantic Web services. The execution of services will eventually provide further valuable information about their services either automatically, thanks to the analysis of the actual execution (e.g., execution time, reliability, etc), or indirectly, thanks to the feedback provided by users.

Crucial to this vision is the use of humans as the most valuable source of information. Users, supported by computers, will contribute to an ever-growing body of services leading to what we refer to as the Service Web [36]. In the short term this will lead a greater number of services that can better support the automation of their discovery, execution, composition, and analysis. In the mid and long term however this will naturally lead to a new set of challenges we need to address. Among these challenges scalability and evolution are of particular relevance for the Service Web. We shall herein present the overall approach we will follow in this respect from the perspective of the Service Provisioning Platform.

We previously introduced that for annotating services in a Web scale we will make use of lightweight semantics, namely WSMO-Lite and MicroWSMO, able to support scalable reasoning for computationally expensive tasks such as web service discovery. WSMO-Lite uses SAWSDL annotations over WSDL files describing web services. MicroWSMO uses microformats for including annotations on web pages describing RESTful services. Thus, in both cases, the annotations are to be included within documents most typically owned by a third party and stored in a server where we most often will not have write access. Additionally, given the collaborative dimension of the Service Web it is likely that, over time, different people will provide different annotations over the same underlying web services (e.g., using different data models). Similarly, these annotations will possibly evolve leading to different versions of a web service annotation being used within different processes, mash-ups, and third-party applications. Finally, it is likely that changes over the web services will occur causing certain disruptions on any other software relying on prior versions of these services.

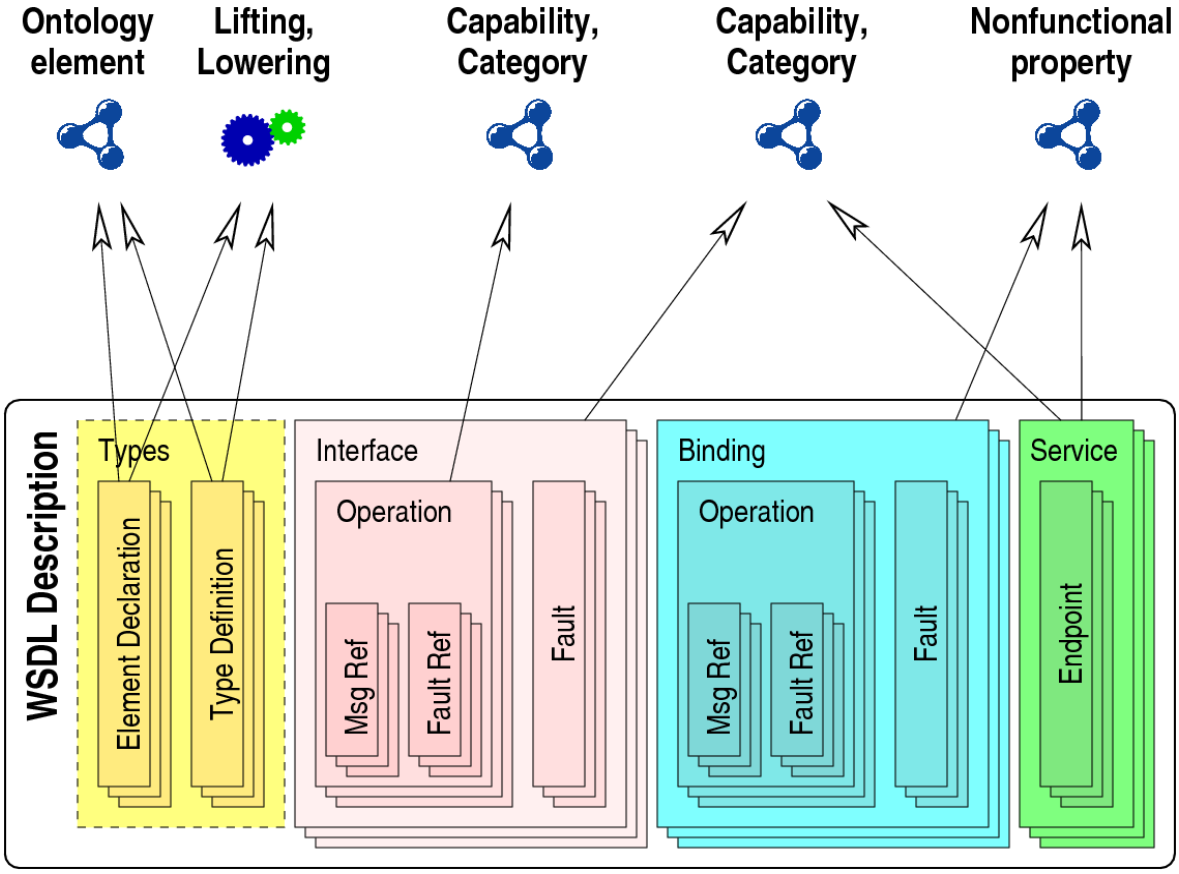

Figure 4. WSMO-Lite Annotations (see [26]).

In [26] the authors present a set of annotations and rules that can be supported by WSMO-Lite using SAWSDL hooks. The types of annotations have been represented in Figure 4. In addition, [26] defines a set of rules over these annotations in order to determine their consistency and completeness. In a nutshell, completeness is achieved when all the parts of a service are annotated. Consistency on the other hand is achieved when the annotations are not contradictory, e.g., transformation and ontological annotations are consistent. Similarly, MicroWSMO proposes the use of microformats for annotating RESTful services as described in HTML web pages. Despite the fact that the kind of document to be handled is different, the notions of consistency and completeness are still applicable.

As we previously highlighted, in the Service Web, a plethora of annotations could be defined

over the same web service. As a consequence, the notions of completeness and consistency become harder to track if all the annotations are kept jointly in the same document. Imagine for instance services with several annotations pointing to ontologies and their corresponding transformations; determining the consistency in this case requires the application of certain mechanisms, e.g., naming conventions, so that one can distinguish between pairs of ontology-transformation annotations. Furthermore, keeping the annotations within a unique service definition can grow the files to sizes where their manipulation will be extremely inefficient. For instance, trying to apply the desired set of annotations, despite the use of naming conventions, would require a time consuming process for retrieving the annotations from the document in the first place.

The lifecycle of semantic Web services exposed above therefore presents important technical issues that need to be addressed when supporting the provisioning of semantic Web services:

- Each WSDL service annotation[2] will have to be stored in the SWS Registry as a new semantic Web service in such a way that all the annotations attached can be retrieved. The details on how to store these annotations will be covered in WP5;
- Similarly, each MicroWSMO annotation will have to be stored in the SWS Registry as a new semantic Web service;
- Each SWS will have to be uniquely identified in a way that users and software can discover and utilise them; and
- The management of SWS should support the evolution of annotations based on users interactions so that they can be improved and still the execution of previously created semantic Web services can be ensured.

These requirements have mainly an impact on the backend used for storing and discovering the semantic Web services being devised in WP5. However, the Service Provisioning Platform needs to take these aspects into account when supporting users in the creation, modification, and deletion of annotations. In particular we shall deal with the lifecycle of services as follows:

- The creation of a new SWS (i.e., a complete annotation over some service) will generate a new document—WSDL with annotations or HTML with annotations—that will be sent for storage to the semantic Web services Registry;
- After storage the repository should return a unique identifier for the annotation created;
- The modification of any existing SWS will provoke the creation of a brand new annotation based on the duplication of the original one;
- The deletion of existing SWS will only be allowed for the creator; and
- To properly support the evolution of services and their annotations, there should be a bidirectional link between each SWS and the underlying service it is enriching with semantic annotations. In this way the impact of changes performed by third-parties over services they offer on the Web can be effectively determined.

In the remainder of this section we shall describe in more detail the different components of the Service Provisioning Platform, explaining how their concrete tasks are performed and

[2] We here refer to “annotation” as the set of coherent annotations of different kinds (e.g., lifting, ontology mapping) part of the same annotation effort. We also use interchangeably the term semantic web service to refer to a complete set of annotations over a service.

giving details on the artefacts manipulated and the interactions with other components.

3.3 Detailed Architecture

The following sections present more details about the different components of the architecture of the Service Provisioning Platform.

3.3.1 Simple SWS Editing Framework

One of the roles of the Service Provisioning Platform is to support providing annotations for existing services so that we can better support tasks like discovering suitable services, or composing new ones out of existing ones. To this end the Simple Semantic Web Services Editing Framework (see Figure 3) has been devised. This framework provides the necessary support for browsing existing services, annotating WSDL services via the so-called WSMO-Lite Editor and RESTful services with the MicroWSMO editor. In this section we shall present the Service Browser which allows users to browse the services that have been found by the Crawler. The editors, however, given their relevance and in order to maintain this document within a reasonable size, are covered in detail in [37].

3.3.1.1 Services Browser

Prior to annotating any service, users need to find them and realise that they are interested in using them even though they have not been annotated yet. Similarly, service providers may be interested in annotating their own services for increasing their uptake. In order to cover for this very first step towards adding semantics to existing services, we need to provide a user interface for users to search, browse and get information about existing services in a convenient way. To this end, the Service Provisioning Platform includes what we refer to as the Service Browser, see Figure 5 for a mock-up of the user interface.

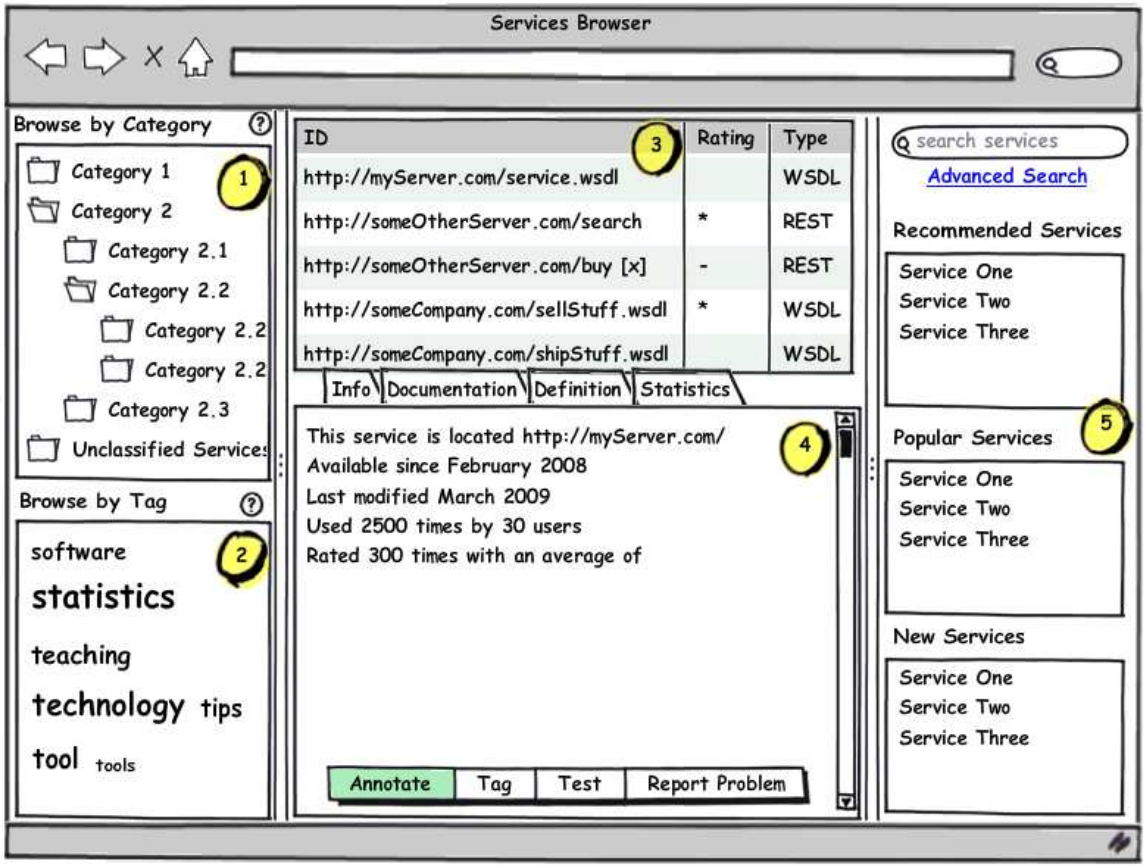

Figure 5. Mockup of the Service Browser User Interface.

The Service Browser provides a simple way for browsing existing services (WSDL-based or RESTful) through two main means. Services can be browsed by category or based on tags attributed by prior users. The former is supported thanks to a predefined taxonomy of domains that identifies in a broad manner the main domain to which services can belong, see area 1 in Figure 5. In particular, we shall reuse the Service Categories ontology [38] defined within Service Finder (FP7 INFSO-ICT-215876) [39]. It is therefore different from the taxonomies used by the Service Consumption Platform which are more specific, typically based around a concrete domain (e.g., eTOM for telecommunications [40]). Instead this global categorisation of services provides a high-level view over crawled services so that users can restrict their search but it is not supposed to provide fully-fledged refined search capabilities since this would require additional metadata about services. This taxonomy will be populated based on an automated high-level domain analysis of services and their documentation after they have been found by the service crawler [41]. The concrete techniques for performing this analysis are covered in more detail in Section 3.3.3 since it is performed by one of the main components of the Annotations Recommender. In order to enhance the overall classification of services, the Service Browser allows users to add additional classifications to services if they feel the automated results were not accurate enough. Browsing through tags is based on the feedback previously provided by users about services (see Section 3.3.5). The existing tags will be shown using the tag cloud widget defined in [12], see area 2 in Figure 5. Obtaining the tags to use will be based on querying the repository holding them, and similarly retrieving relevant services based on a tag selected will involve querying the Crawler Registry.

The area 3 in the user interface of the Service Browser will be populated with the appropriate list of services, based on the tag or category selected showing some brief information about

them. With this list users will be able to quickly browse through relevant services retrieved, and the Area 4 will show additional detailed information about the service selected including general data such as the server hosting the service, when it was crawled, or how many times users have rated the service. Additionally, users will be able to get further details about the service such as its technical description, performance data gathered about service or documentation available. All this information will be accordingly retrieved from the various repositories maintained in SOAAll (e.g., Crawled Registry, analysis results, etc).

Finally, in order to provide more flexibility for the user to retrieve services, the Service Browser will allow generating keyword-based queries over the crawler registry [41], and it will show a set of services for convenience based on their popularity, how recently they have been added to the registry, or even based on the likelihood for the service to be relevant to the user given his or her profile and previous behaviour from similar users.

The means for achieving the functionality described above will be detailed next, presenting first the main use cases and then moving into more concrete details on how they are to be performed.

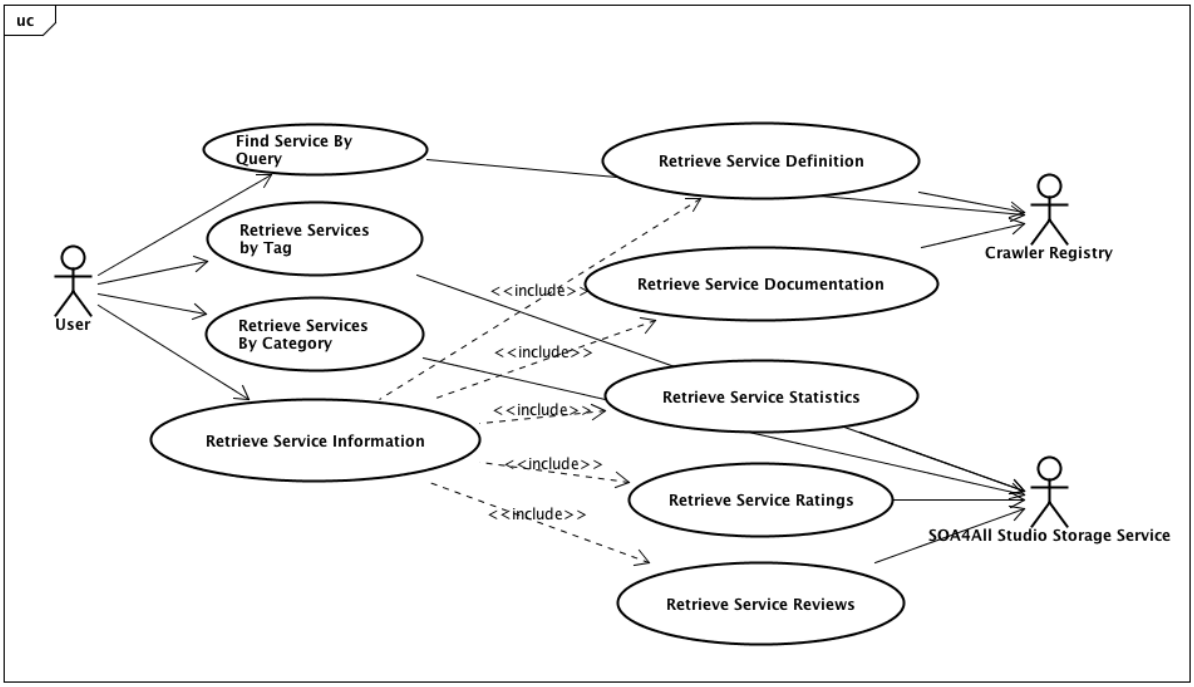

Figure 6. Service Browser Use Case Diagram.

The Service Browser will provide the functionality described above through 4 main use cases illustrated in Figure 6. *Find Service By Query* as indicated by its name allows users to find services based on queries. The Crawler Registry supports keyword-based queries as well as SPARQL queries based on the Service Ontology it uses [41]. *Retrieve Services By Tag* on the other hand allows to retrieve services that have been attributed a concrete tag by the users. *Retrieve Services By Category* allows users to find services that have been categorised with respect to the high-level services domains category [38]. Finally, *Retrieve Service Information* obtains all the known information about a given service. This use case is in turn supported by 5 other use cases that retrieve parts of the information, namely the service definition, related documentation, statistics, ratings, and reviews.

The implementation for most of these use cases will be rather similar, the main difference been the backend component to which the request is forwarded. *Find Service By Query*, *Retrieve Service Definition* and Retrieve Service Documentation contact the Crawler Registry in order to obtain the list of services matching a query or the information stored about one

concrete service. The rest of the use cases depicted in Figure 6 on the other hand delegate their query to SOAAll Studio Storage Service (see [12]) who will in turn send the query to the Distributed Service Bus. In order to illustrate the process in both cases, Figure 7 shows the sequence diagram for retrieving services based on a text-based query, and Figure 8 shows the sequence diagram for retrieving the detailed information about the service.

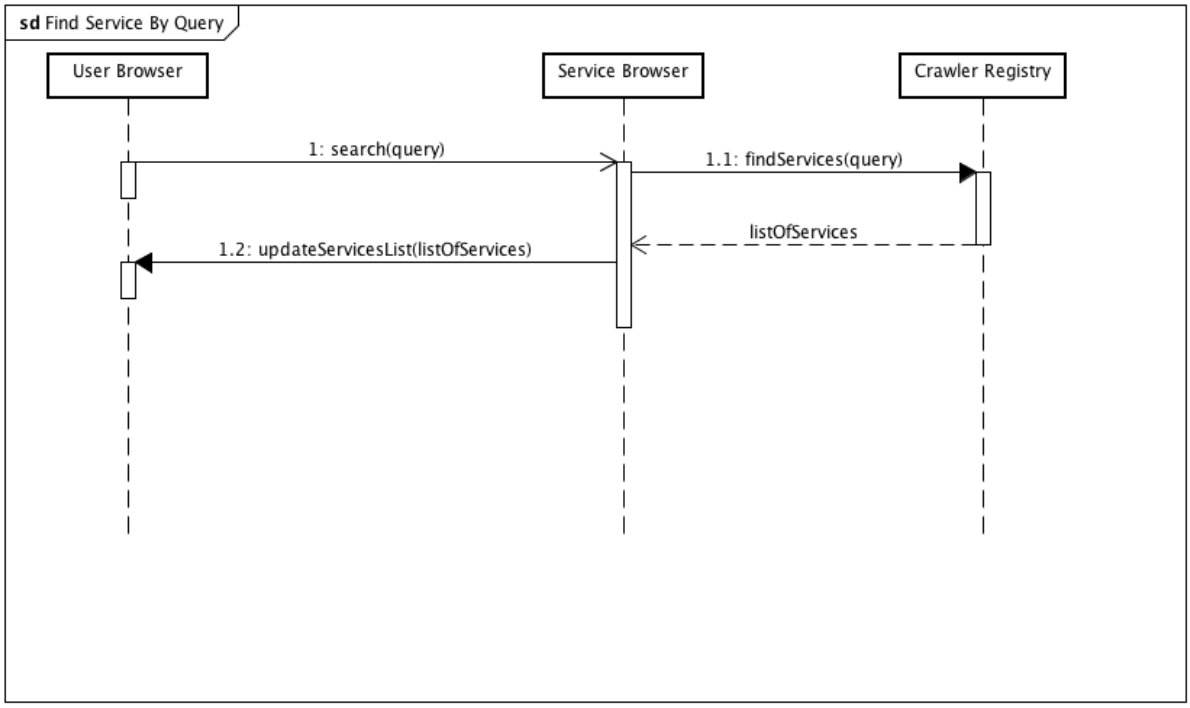

Figure 7. Find Service By Query Sequence Diagram.

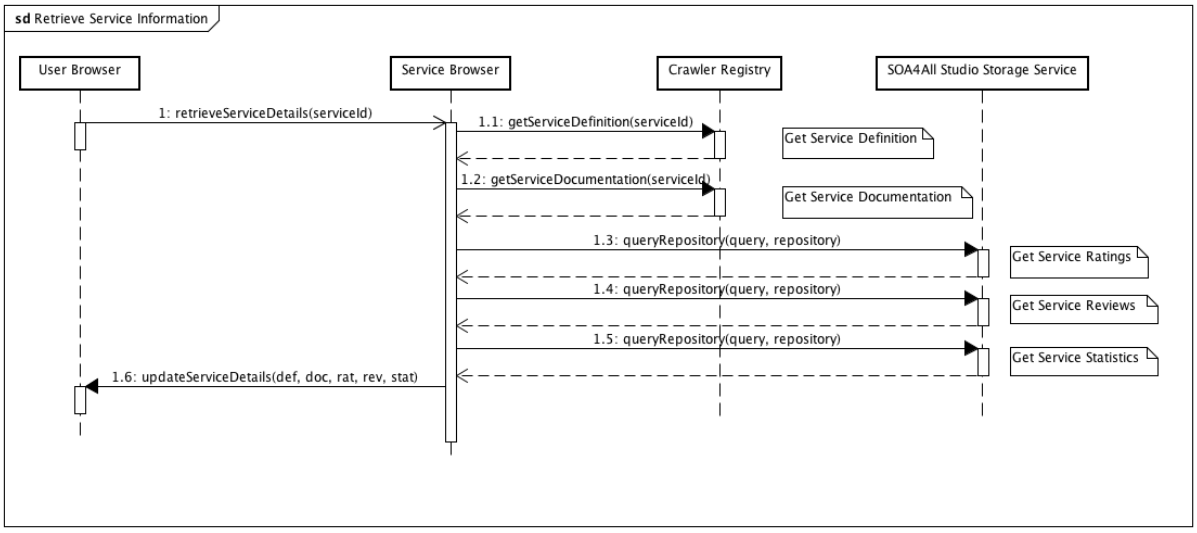

Figure 8. Retrieve Service Information Sequence Diagram.

3.3.2 Process Editor

In addition to providing annotations over existing services, the Service Provisioning Platform will support the creation of composite semantic Web services. We contemplate in particular

two kinds of composite services, namely processes and mash-ups [16, 21, 22]. The former refer to orchestrations of services that include a more or less complex control and data flow. The latter, following Web 2.0 practices, aims at supporting the creation of simple composite services and therefore limits the definitions to orchestrations whereby the control flow is implicit in the data flow. The creation of composite semantic Web services will be supported by the Process Editor which is part of the Service Provisioning Platform. Again this framework is described in more detail in a separate deliverable [16] in order to devote the level of detail it requires.

3.3.3 Annotations Recommender

The Annotations Recommender is one of the main components of the Provisioning platform. Its main function, as the name suggests, is to assist the user in annotating a service by recommending suitable annotations at different levels of granularity. The Annotations Recommender analyses the service description and related documents in the crawled data, and uses this information to suggest suitable annotations for the service as a whole (domain and classification annotations) and for its individual properties (for example, input/output data types and operations). The Annotations Recommender plays a key role for the semantic description of services, since it reduces the amount of necessary manual work and increases the accuracy of the resulting semantic service descriptions by actively involving the user. This section gives an overview of the Annotations Recommender architecture and design. Additional details, including activity diagrams and implementation requirements, are given in the Annex.

The recommendation approach used for the SOAAll Annotations Recommender is a hybrid one, combining both content-based and ontology-based recommendation [42]. The content-based recommendation is implemented by computing similarity measures, between the description of the new service to be annotated and previously annotated services. The computation of these similarity manures is done by determining text correlations and correspondence. For example, the k-Nearest Neighbour [43] approach determines which service description is the ‘closest’ one to the service to be annotated, by representing each service description as a term vector and comparing these term vectors. Other text-similarity measures for making content based recommendation include, lexical matching methods [44] and word-to-word similarity measures, using approaches that are either knowledge based [45], [46] or corpus-based [47]. [48] provides a concise overview of the most common text-similarity measures and their applications. The content based recommendation approach used in the Annotations Recommender is going to be based on the k-Nearest Neighbour approach, where the service descriptions are represented by computing the normalized term frequency[3] for each word and removing stop words[4] and words with very low frequency[5]. Previous work with k-Nearest Neighbour-based comparison and classification has show that it delivers good results [42] and there exist already performant implementations.

The ontology-based recommendation is used to compensate for the shortcomings of content-based approaches. While content-similarity approaches are successful to a certain degree, they cannot always identify the semantic similarity of texts. For instance, there is an obvious similarity between the text segments “I own a dog” and “I have an animal”, but most of the current text similarity metrics will fail in identifying any kind of connection between these texts. The Annotations Recommender will profit from the fact that service descriptions are

[3] Number of times a term occurs in the text divided by the total number of terms in the text.

[4] Stop words are very common words like “the” and “or”.

[5] Dimensionality reduction is common and necessary in information analysis [49].

semantically annotated and will identify semantic similarities by using ontology inference [42, 50]. The goal is to achieve more accurate annotation recommendations by combining the two recommendation approaches and compensating the weaknesses of text-similarity by using ontology inference.

Figure 9 depicts the main components of the Annotations Recommender, based on the hybrid recommendation approach. The Service Data Preprocessing component performs text analysis tasks including i) term frequency extraction; ii) natural language analysis [51] and; iii) keywords extraction [52]. These tasks do not require any user interaction and can be performed in a pre-processing step, preparing the service data for the actual annotation recommendation. The Relationship Analysis component implements the most important tasks for the Annotations Recommender. It performs k-Nearest Neighbour similarity analysis of the new service and previously annotated services, as well as it computes correlations for determining semantic-similarity. Based on the results of the Relationship Analysis component, a Set of Similar Services (SSS) is determined, whose annotations are used as the basis for recommendations for the new service. The Relationship Analysis component uses input data from the Service Data Preprocessing and from the Document Analysis, which performs tasks for identifying service properties in a given service description.

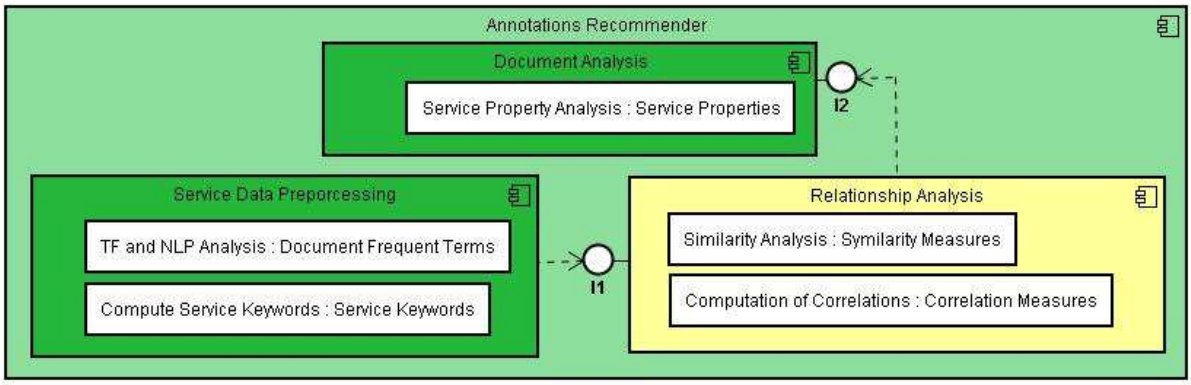

Figure 9. Annotations Recommender Components.

The design of the Annotations Recommender is based on one main use case: *Recommend Service Annotations*. Figure 10 visualizes this use case, and the included use cases *Recommend Service Annotations for WSDL Services* and *Recommend Service Annotations for RESTful Services* for making recommendations for WSMO-Lite and MicroWSMO annotations respectively. The Annotations Recommender has one Service Annotating User actor, who uses the Recommended Service Annotations results in order to complete a semantic service description. The Annotations Recommender does not include a user interface, but rather exposes its results through the Simple SWS Editing Framework described in more detail in [37]. In a similar way, the Recommender uses the SOAAll Distributed Service Bus (DSB) (see [53], [12]), which enables the access to Crawler Registry, external services and User Profiles repository.

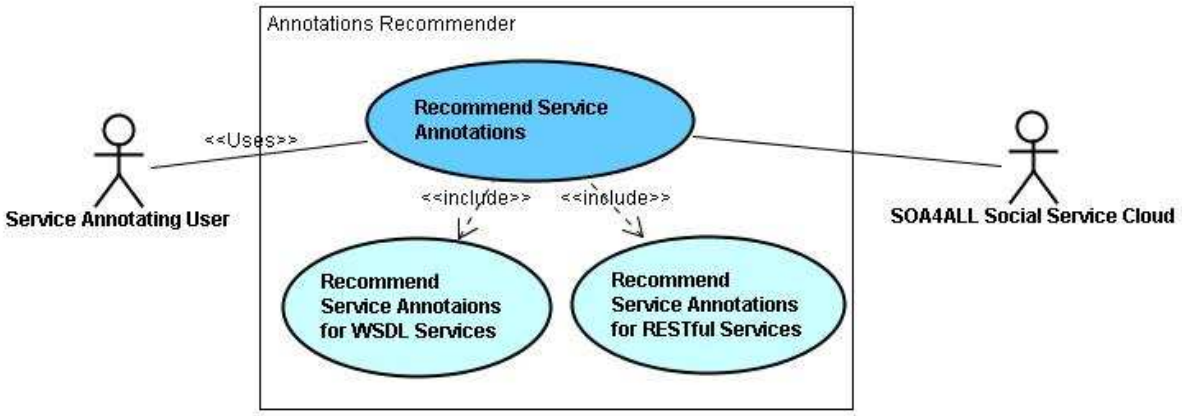

Figure 10. Use Case diagram "Recommend Service Annotations".

Table 4. Artefacts in the "Recommend Service Annotations" Use case.

Activity Name	Input Information	Output Information	Comments
1. Service Preprocessing	- Service Description -Service Related Documents	- List of Service Properties - Set of Similar Services (SSS) -Service Keywords	-Access to the Crawler Registry is required, which returns a Service and its Related Documents -The TF and Keywords extraction can be performed by using system-external services. A connection to these services would be a requirement. - The Preprocessing does not involve any user interaction and can be performed anytime previous to the user-guided annotations - The SSS can be an empty set, if there is not enough training data, or annotated services.
2.Suggest Service Domain	-Service Keywords -Service Properties -Ontology for Service Domains -SSS	- Service Domain, automatically assigned to the Service - List of Possible Service Domains -SSS, which is narrowed down by assigning a Domain to the Service	- The Domain Analysis does not involve any user interaction and can be performed anytime previous to the user-guided annotations. The Result is one Domain assigned to the Service, and a list of possible Domains, from which the user can choose, if he wants to make a change. -The generation of training data is a necessary and essential step. It must be done before the Annotation Recommender is in an operation state. -The SSS can be an empty set, if there is not enough training data, or annotated services.
3.Suggest Service Classification	-Service Domain -Domain\ Classification Dependencies -Classification Taxonomies -SSS -User Profile	- Service Classification, automatically assigned to the Service - List of Possible Service Classifications -SSS, which is narrowed down by assigning a Classification to the Service	- The Classification Analysis does not involve any user interaction and can be performed anytime previous to the user-guided annotations. Only the Classification based on User Profile requires that a user is logged onto the system (no user interaction is required). -The User Profile, can be used in the same manner as the SSS. By computing similarity of the current user to other users, the proper Service classification can be determined. In contrast to the SSS similarity, User Profile similarity has to be determined based on an active user.
4.Suggest Domain Ontology	-Service Classification -Service Domain -Domain Ontologies -Service Properties -Service Keywords	- List of Recommended Service Domain Ontologies, the user assigns one of them to the Service -SSS, which is narrowed down by assigning a Domain Ontology to the Service	- A connection to a service for searching for existing ontologies and semantic information (Watson [54], Sindice [55]) is required -This activity requires an active user to choose a Domain Ontology from the list with suggestions.
5.Suggest Annotations for Service Properties	-Domain ontology -Service Properties -SSS and Property Annotations	- List of Service Property Annotation Suggestions, from which the user has to choose	-In contrast to previous activities, this one requires continuous user interaction.

In summary, the *Recommend Service Annotations* use case performs the following tasks. First, as soon as new services without annotations are available, for example after being collected by a crawler, a pre-processing of the service data is performed, which includes term frequency and key words analysis. Based on these, similarity measures to already annotated services are computed, and a domain and a classification are initially assigned to the service. A ranked list of other possible domains and classifications is stored, so that the user can easily change the initial service annotations. After these computation steps are completed, the user initiates the next one by deciding to annotate a service, which thanks to

the pre-processing has already an identified domain and classification, as well as a set of keywords and term frequencies. When the user wants to annotate each of the properties of the service, the Annotation Recommender computes a list of domain ontology suggestions. Based on the chosen domain ontology, the user can annotate the service in detail. These steps which we have described from a high-level perpective are discussed in detail in the following sections, starting with the required artefacts and continuing with a description of each of the activities.

The *Recommend Service Annotations* use case comprises a number of activities, which perform the tasks that are necessary for the computation of annotation recommendations. Each of these activities has a set of requirements and effects, as well as a number of input and output artefacts which are processed. A detailed activity-based overview on the inputs and outputs is given in Table 4.

Considering the required input information and the resulting output, the Service Description and the Service Related Documents are the only initially required input. The Service Description is a WSDL file or a REST API text description, while the Service Related Documents include descriptions of implementation details (html, MSWord, pdf document formats), news and updates, user comments and feedback. The exact structure and content of the Service Related Documents and how they are determined are described in [41].

In addition, to the initial input information, the Annotations Recommender requires access to the semantic Web services Registry [21], the Crawler Registry [41], the User Profiles repository currently being devised in T2.7 and a number of external services, such as Watson[6] [54] and existing NLP methods implementations[7]. This access is facilitated by the DSB [12, 21, 53]. The overall result of the use case is an Annotated Service, while there are several important intermediary artefacts, such as the Service Keywords and Service Properties, which are cached and reused in several activities.

The Service Keywords are seen as the service's content representation and serve as the basis for determining similarity measures to other service descriptions. The Service Keywords are computed during the Service Preprocessing activity. During the pre-processing, term frequencies are computed, for all words present in the service description and its related documents. These term frequencies are normalized, and the terms appearing more often in the related documents, than in the description itself are assigned lower weights, in order to decrease their importance in the service representation. Following is a step, during which stop words and words with low frequencies are eliminated. The result is the Service Keywords, which are used for determining a Set of Similar Services (SSS), based on which annotation recommendations are made.

The Service Properties, on the other hand, are determined by directly analysing the WSDL file or REST description. These include service operations, input and output types, address, etc. and serve as the main elements for attaching semantic information. The Service Properties are used together with the Service Keywords, to determine possible service domain and classification, since the service operations, input and output carry indications for functionality of the service. In contrast to the Service Keywords, the Service Properties can be subsequently edited by the user, in order to ensure that all service properties are properly identified and can be annotated.

The suggesting of a service domain and service classification (2. and 3.) requires the use of

[6] Watson is an infrastructure component for the Semantic Web, a gateway that provides the necessary functions to support applications in using the Semantic Web. In particularly, it enables ontology keywords-based search.

[7] ClearForest NLP services http://sws.clearforest.com/Blog/?page_id=6

a predefined Ontology for Service Domains and Classification Taxonomies. A service Domain, as defined in the context of the Annotations Recommender represents is the general application area of the service (for example, telecommunications). The Classification, on the other had, described the taxonomy, based on which the service function can be described (for example, NGOSS-based taxonomy [40]). Naturally, there is a correlation between the domains and classifications, which is specified in the Domain\Classification Dependencies document. This document specifies which classification taxonomies can be assigned to which domains and vice versa. The Domain\Classification Dependencies document will be composed manually; however, it can be updated and extended by the users, by adding new domains and classification taxonomies.

As already mentioned, the suggesting of a service domain and service classification (2. and 3.) are based on the determined Similar Set of Service (SSS), which is computed by calculating similarity measures between already annotated services and the service keywords. However, no recommendations can be made, if the SSS contains no service, i.e. the set is empty. This requires the manual creation of an initial set of training data. The training data consist of manually annotated services, whereas, the goal is to cover as many domains and classifications as possible. The provisioning of training data has to be done by users who provide accurate annotations, with a certain level of expertise, because the following recommendation relies completely on these initial annotations. In addition, since the annotation of training data is a completely manual process, the MicroWSMO Editor and the WSMO-Lite Editor can be used to ease the work. The resulting training data annotations can than be used for determining similarity measures and adding new annotated service.

In addition, it must be pointed out, that a main point of weakness of all recommender systems is the new-system cold-start problem [42]. This is typical in new systems, which have only a few initial documents and input, on which similarity analysis can be based. As a result the computed recommendations have a fairly low accuracy. In order to tackle this problem, the Annotations Recommender will implement additional methods of probability correlation analysis [56], whenever the set of similar services is empty, in order to ensure that users can properly annotate services from the very beginning. The probability correlation approach reflects the existing dependencies between the domains, the classifications and the individual properties of a service and is computed by providing an initial probability table, which is modified based on the given training data. In this way, a service property annotations recommendation is based on the probability that it belongs to a given classification and to a given domain. For example, a property “book title” is more likely to occur in a book order service, than in a telephone registry service.

After discussing the most important artefacts involved in the *Recommend Service Annotations* use case, it is also important to describe the corresponding activities. There are two main types of activities. The first type does not require any user interaction and can be completed offline, in a batch-manner as soon as new services without annotations are available. Service Preprocessing, Suggest Service Domain, and Suggest Service Classification are such activities, which automatically determine frequent terms, service keywords, and assign a domain and a classification to the service. The second type of activities are user-bound and require an active user to choose a domain ontology from a list of suggested domain ontologies (Suggest Domain Ontology) and to choose an annotation for each of the service properties (Suggest Annotations for Service Properties). Figure 11 visualizes the activity flow and the artefact modifications involved in the *Recommend Service Annotations* use case.

The Service Preprocessing (Activity 1) computes the Service Keywords, an initial SSS and the Service Properties. It must be pointed out that depending on the number of already annotated services, the computation of the SSS, based only on the Service Keywords and without any additional constrains, could need a lot of resources and time. Therefore, the

computation of the SSS may have to start after a domain is assigned to the service, which narrows down the number of required comparison steps. After this, the SSS will be refined during each activity.

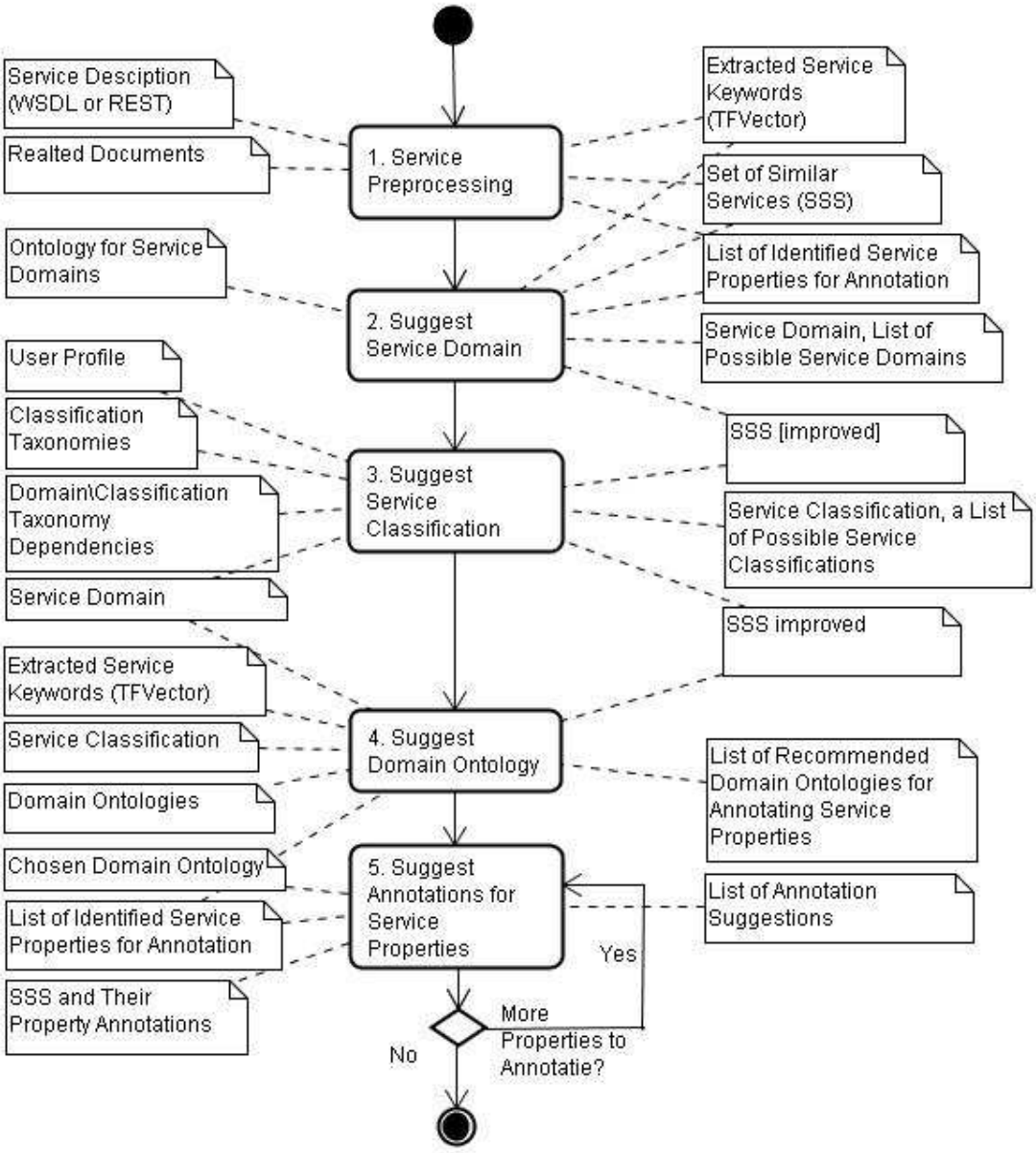

Figure 11. Activity Diagram: "Recommend Service Annotations".

This use case is cumulative for recommending annotations for both WSDL and RESTful services. However, due to the difference in the descriptions and the different resulting annotations (WSMO Lite and MicroWSMO) some of the activities include different subtask or additional artefacts. For example, the Service Preprocessing for RESTful services not only identifies service properties, but also generates a corresponding hREST [29] description.

The Annotations Recommender is based on past research and reuses some of the main information retrieval and recommender systems techniques. These include document representation by term frequency-based keywords, similarity measures computations and classification approaches. However, the Annotations Recommender will adapt these methods to the particular task, that is service description analysis. In addition, the service pre-processing, as well as the multi-level recommendation, including domain, classification

and ontology recommendation, pose additional challenges.

3.3.4 Templates and Service Creation Wizards Management Framework

Within the Service Provisioning Platform, we will provide wizards on top of the MicroWSMO and WSMO-Lite editors [37], guiding our users through the necessary steps to complete the annotation of services. The wizards shall trigger when necessary the invocation of the Annotations Recommender (see Section 3.3.3) in order to give suggestions to the user.

By having intuitive wizards, we will increase the chances of having different kinds of users interacting with the Service Provisioning Platform, for the wizards will be an easy way to begin to use the platform. Hence, the first contact of SOAAll users, when acting as service providers, will likely be easier through the use of wizards, and they will be initiated in service provisioning in a smoother way.

Our particular approach towards these wizards will take advantage of the characteristics of the Ext-GWT Framework that we will use in the components of the SOAAll Studio [12]. We will present the wizard as a draggable box on top of the rest of UI components of the platform, see Figure 12. Any actions taken by the user (e.g., answering to some question) will trigger the appropriate actions in the Provisioning Platform, thus assisting the user in the generation of semantic web services in a simple step-by-step way.

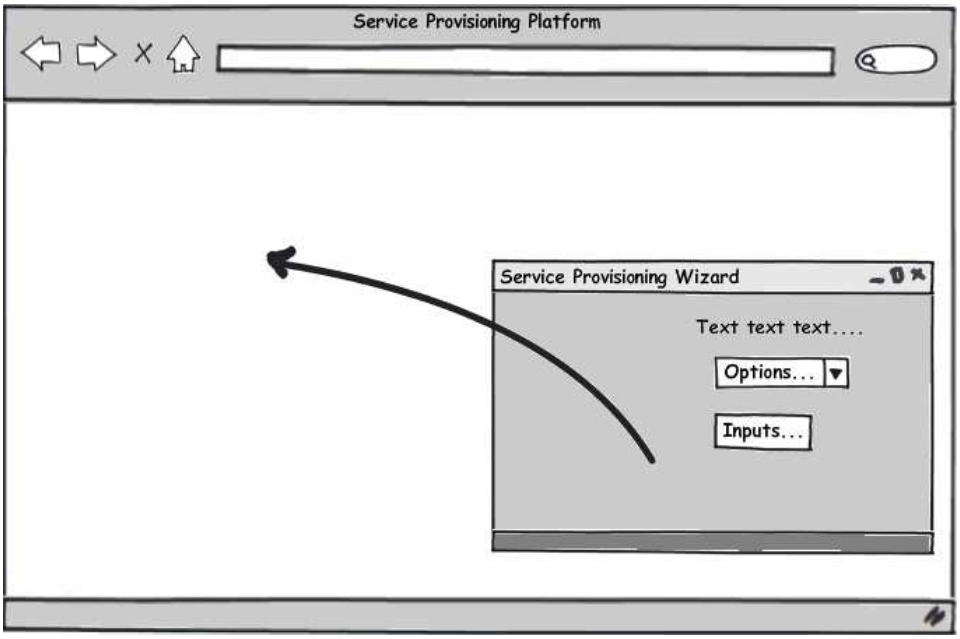

Figure 12. Service Provisioning Wizard interacting with the Platform

This way, while the wizards will simplify things for the non-expert user, abstracting the complex operations that take place below, at the same time we will be able to show the more advanced users what do their selections in the wizard imply in the platform, allowing them to understand what is going on, in which part of the overall process they are, and thus to learn how can they perform those actions without the help of the wizard in successive interactions with the platform.

From the technical point of view, wizard paths will be represented in the platform using XML. The Service Creation Wizards component will parse those XML files in order to present the relevant instructions to the user. The following Figure 13 is an example of the definition of a wizard and how it would translate into the graphical element shown to the user.

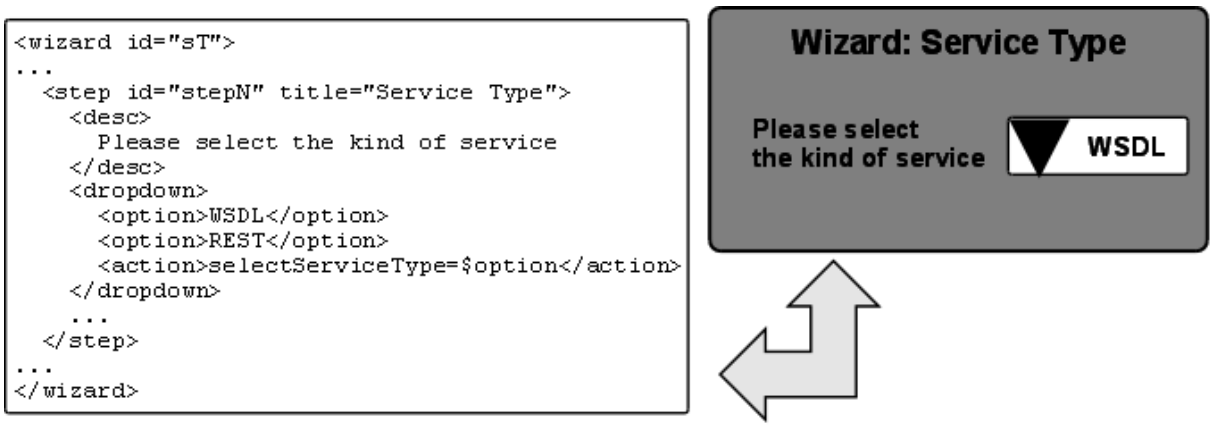

Figure 13. Wizard definition and translation to GUI example

3.3.5 Feedback Management Framework

SOAAll users (*service prosumers*) will be able to create new semantic web services within the Service Provisioning Platform, and they will also provide useful information about them. Our approach is similar to that of Web 2.0, where users are able to enrich the content by providing additional information about it, typically in these ways:

- By tagging: Associating lightweight keywords which are relevant for a particular item.
- By rating: Assigning a numeric value from a predefined set, typically from 1 to 5, to an item, indicating its subjective quality or degree of helpfulness.
- By evaluating: Writing textual reviews or comments about a particular item, which are useful for other users.

In SOAAll, we want to allow users to give their feedback about services (and Goals) in the aforementioned ways. While the Service Provisioning Platform mainly focuses on Services (strictly speaking, about semantic descriptions of services made in WSMO-Lite and MicroWSMO), the Service Consumption Platform [57] stresses the use of (WSMO) Goals (and as such, service consumers will be "Goal *prosumers*"), we will consider here any kind of feedback on the different items (Services and Goals), for it is information *provided* by the users to enrich their descriptions. We will refer to the identifiers of both types of items expressed as URIs in the following subsections as "idService" and "idGoal" respectively.

We enclose here the sequence diagram (Figure 14) envisioned for the Feedback Management Framework, including the three kinds of feedback previously addressed.

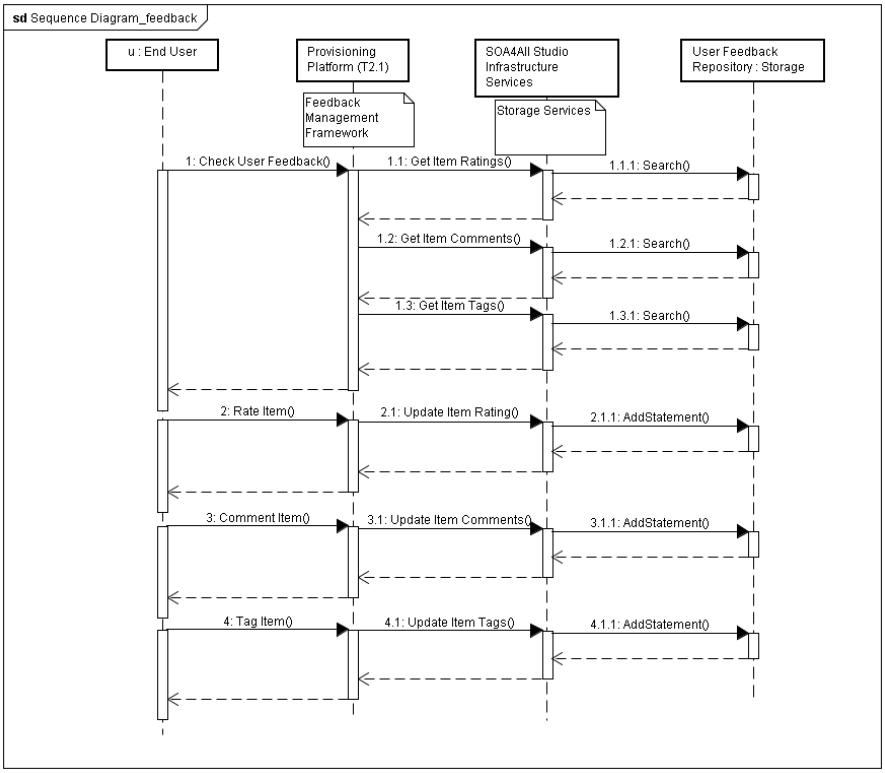

Figure 14. Feedback Management Framework sequence diagram

Regarding storage of feedback information, they all share a common approach: the relevant information is stored in the semantic spaces provided by WP1 through the use of the SOAAll Studio Infrastructure Services for Storage [12]. The sequence diagram depicts the interaction of users with the repositories through the invocation of the SOAAll Studio Storage Services.

We will also make use of other SOAAll Studio facilities for generating Tag Clouds. Additionally, with respect to ratings and comments, we will make use of the Review vocabulary [33], therefore ensuring compatibility with existing systems such as Revyu.com [34] and therefore ready to be integrated with the Linked Data initiative [35].

Another general guideline for the collection of feedback in SOAAll Studio is that we will require that the user providing feedback is logged-in, i.e., we won't permit anonymous feedback. We will make use of the Management Services provided as part of the SOAAll Studio Infrastructure Services [12] to easily map every feedback to the particular user providing it.

In the following subsections, we will cover each of the three components that will take care of the different kinds of feedback, and their technical details.

3.3.5.1 Tagging component

The Service Provisioning Platform will provide the user interface and underlying machinery for tagging Services. The Service Consumption Platform will additionally make use of the infrastructure devised here for allowing users to easily attach tags to Goals that are considered relevant for describing them. The approach is similar to the one taken by Web 2.0 sites like Flickr [31] or Delicious [32], which allow to tag content (photos and urls, respectively).

Technically, tags within this platform will be stored as RDF into the semantic spaces, through the use of the SOAAll Studio Storage Services. Retrieving the tags from the repository will be possible through the same gateway services. Additionally, this component will be able to

display “Tag Clouds”, making use of the Tag Cloud Widget provided by the SOAAll Studio UI Components [12].

The following listing expresses in a simplified and schematic way, using Turtle notation for informative purposes, our approach towards the vocabulary for storage of tags in the semantic spaces. Note that different resources (`rdfs:Resource`) such as services or Goals may have associations to tags. Each association can have a tag and a user who has associated the keyword to the resource.

```
@prefix foaf: <http://xmlns.com/foaf/0.1/> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix tag: <http://www.SOAAll.eu/tags#> .

:Association     a rdfs:Class;
    rdfs:comment "An association of a tag to an item";
    rdfs:label "Association";

:hasTag     a rdfs:Property;
    rdfs:comment "Used to specify the tag in an association";
    rdfs:domain tags:Association;
    rdfs:range tags:Tag;
    rdfs:label "hasTag";

:hasAssociation     a rdfs:Property;
    rdfs:comment "Used to specify the item in an association";
    rdfs:domain rdfs:Resource;
    rdfs:range tags:Association;
    rdfs:label "hasAssociation";

:Tag     a rdfs:Class;
    rdfs:comment "A keyword to describe items";
    rdfs:label "Tag";

:tagger     a rdf:Property;
    rdfs:comment "The user that tags the item";
    rdfs:label "tagger";
    rdfs:domain tags:Association;
    rdfs:range foaf:Agent;
```

The following Table 5 expresses the functionalities covered by this component, and the components that it makes use of.

Table 5. Tagging component summary

Functionalities covered	Tagging Services/Goals with relevant keywords Retrieving tags of a given Service/Goal Retrieving Services/Goals by selecting keywords
Components used	SOAAll Studio Infrastructure Services: Storage Services SOAAll Studio Infrastructure Services: Management Services SOAAll Studio UI Components: Tag Cloud Widget

The following functions will deal with tagging Goals and retrieving the tagging information:

1) Function `tagItem(idService or idGoal, tag, user)`

It permits a user assign a keyword (tag) to a Service/Goal.

It implies that both the list of tags assigned to the item and the list of items assigned to a tag in the repository are updated.

2) Function `getItemTags(idService or idGoal) and getItemTags(idService or idGoal, user): Tag[]`

For a given Service/Goal, it retrieves its tags. A user can also be provided to the function, in order to retrieve the particular tags that a user has chosen for the particular item.

3) Function: `getServices(tag[])` and `getServices(tag[], user): ServiceId[]` and `getGoals(tag[])` and `getGoals(tag[], user): GoalId[]`

For a given tag (or set of tags), this function returns the list of relevant Services/Goals. It also can be restricted to the annotations of a particular user.

4) Function: `getServiceTagCloud(number)` and `getGoalTagCloud(number)`

It creates a tag cloud with the specified number of tags, making use of the Tag Cloud Widget provided by the SOAAll Studio UI Components. A number of tags can be passed as an argument, as that widget accepts an “amount” as an argument. It returns the tag cloud as a graphical element.

3.3.5.2 Evaluations component

Informal evaluations of Services and Goals in the form of textual comments will be possible thanks to this component. These comments can help users understand better the purpose of the Service or Goal, and check previous experiences of users with them.

We will make use of the Review vocabulary in order to easily enable a way of storing evaluations, following an existing approach. In particular, in the `text` property of the `Review` class. The storage of the RDF-based Review instances will be again achieved through the use of the Storage Services. The following listing shows the relevant parts of the Review vocabulary for comments, expressed in Turtle:

```
:Review     a rdfs:Class;
    rdfs:comment "A review of an artistic work";
    rdfs:isDefinedBy <http://www.purl.org/stuff/rev>;
    rdfs:label "Review";
    vs:moreinfo "core term";
    vs:term_status "stable" .

:hasReview     a rdf:Property;
    rdfs:comment "Used to associate a work of art with a a review";
    rdfs:domain rdfs:Resource;
    rdfs:isDefinedBy <http://www.purl.org/stuff/rev>;
    rdfs:label "hasReview";
    rdfs:range <http://www.purl.org/stuff/rev#Review>;
    vs:moreinfo "core term";
    vs:term_status "stable" .

:text     a rdf:Property;
    rdfs:comment "The text of the review";
    rdfs:isDefinedBy <http://www.purl.org/stuff/rev>;
    rdfs:label "text";
    vs:moreinfo "core term";
    vs:term_status "stable" .
```

The following Table 6 expresses the functionalities covered and components used by this component, as not only writing evaluations and retrieving them will be supported, but also a full text search on those.

Table 6. Evaluations component summary.

Functionalities covered	Writing a comment on a Service/Goal Retrieving comments of a given Service/Goal Full text search of the comments made on Services and Goals
Components used	SOAAll Studio Infrastructure Services: Storage Services

These functions will permit to cover the functionalities:

1) Function `commentItem(idService or idGoal, comment, user)`

It stores a comment by a user on a Service/Goal, updating the list of comments assigned to it.

2) Function `getComments(idService or idGoal) and getComments(idService or idGoal, user): comment[]`

For a particular Service/Goal, it returns a list of comments. It can be invoked with the user as an extra parameter, allowing to retrieve specific comments of a user for a given Service/Goal.

3) Function `searchComments(string): idService[] or idGoal[]`

For a set of words provided by the user, it returns a list of services or goals that have reviews that match those words.

3.3.5.3 Ratings component

This component deals with the user-generated ratings of services and goals. It allows people to rate those items, assigning a numeric value amongst a predefined set to them, and retrieving those ratings conveniently.

As for the graphical part, we will rely on the Rating Widget provided by the SOAAll Studio UI Components [12], while this component will also take care of storing and retrieving the data, and about the logic used.

The Review vocabulary addressed in the previous section also contemplates ratings as part of a Review, so we will use it for our purposes. Like in the previous cases, storing and retrieving the instances from the semantic spaces repository through the SOAAll Studio Storage Services. In the Review vocabulary, the `Review` class includes two different properties: `positiveVotes` and `totalVotes` (integers), by which a rating by a particular user is defined.

The following listing shows these relevant parts of the Review vocabulary regarding ratings:

```
:Review     a rdfs:Class;
    rdfs:comment "A review of an artistic work";
    rdfs:isDefinedBy <http://www.purl.org/stuff/rev>;
    rdfs:label "Review";
    vs:moreinfo "core term";
    vs:term_status "stable" .
```

```
:hasReview     a rdf:Property;
    rdfs:comment "Used to associate a work of art with a a review";
    rdfs:domain rdfs:Resource;
    rdfs:isDefinedBy <http://www.purl.org/stuff/rev>;
    rdfs:label "hasReview";
    rdfs:range <http://www.purl.org/stuff/rev#Review>;
    vs:moreinfo "core term";
    vs:term_status "stable" .

:reviewer     a rdf:Property;
    rdfs:comment "The person that has written the review";
    rdfs:domain <http://www.purl.org/stuff/rev#Review>;
    rdfs:isDefinedBy <http://www.purl.org/stuff/rev>;
    rdfs:label "reviewer";
    rdfs:range foaf:Person;
    vs:moreinfo "core term";
    vs:term_status "stable" .

:positiveVotes     a rdf:Property;
    rdfs:comment "Number of positive usefulness votes (integer)";
    rdfs:domain <http://www.purl.org/stuff/rev#Review>;
    rdfs:isDefinedBy <http://www.purl.org/stuff/rev>;
    rdfs:label "positiveVotes";
    rdfs:range rdfs:Literal;
    vs:moreinfo "proposed by iterating.com";
    vs:term_status "testing" .

:totalVotes     a rdf:Property;
    rdfs:comment "Number of usefulness votes (integer)";
    rdfs:domain <http://www.purl.org/stuff/rev#Review>;
    rdfs:isDefinedBy <http://www.purl.org/stuff/rev>;
    rdfs:label "totalVotes";
    rdfs:range rdfs:Literal;
    vs:moreinfo "proposed by iterating.com";
    vs:term_status "testing" .
```

We have decided that the component will allow one rating per user and service, so when a particular user rates again the same service, he will be updating the data, but not storing a second rating. As each Review is performed by a “reviewer”, it will be easy to check if a user has rated the item before or not.

The following Table 7 shows the functionalities that this component covers, and the components that it uses to achieve so:

Table 7. Goal Ratings component summary.

Functionalities covered	Ability to rate a Service/Goal. Ability to retrieve the average rating of a Service/Goal. Ability to retrieve a particular rating of a user on a Service/Goal.
Components used	SOAAll Studio Infrastructure Services: Storage Services SOAAll Studio UI Components: Rating Widget

These functions will permit dealing with ratings:

1) Function: `rateItem(idService or idGoal, positive votes, total votes, user)`

It stores a new rating on a Service/Goal by a user, specifying the number of "positive votes" as well as the "total votes" (i.e., the maximum number that the user could have chosen). If the user had rated the same item before, the rating is updated, but not added.

2) Function `getRating(idService or idGoal): average rating, number of ratings` and `getRating(idService or idGoal, user): positive votes, total votes`

If it is invoked without specifying a particular user, it returns the average rating of a Service/Goal (taking into account all the positive and total votes of each rating on that item), as well as the total number of ratings. It can be invoked with a particular user as a parameter, in which case his rating for the item (expressed in positive votes versus total votes) is returned.

3.3.6 Import Facilities

We previously mentioned the main requirements that had to be fulfilled by the Provisioning Platform. Among these we highlighted the need for importing existing knowledge models in order to reduce the overhead for annotating services but also for eventually reducing the mismatches between different knowledge models, using instead common ontologies.

Given the scope of the SOAAll Studio, we address this requirement in a rather indirect way by allowing users to automatically retrieve and use existing ontologies from the Web. In fact, the current evolution of the Semantic Web has taken us to a situation where thousands of ontologies are available on the Web. In the light of this situation several researchers have worked on search engines able to find, index, and serve third-party ontologies to users and applications over the Web. The Provisioning Platform will make use of these systems for importing existing ontologies and using them directly when annotating services. Among the systems we intend to integrate it is worth mentioning Watson [28], developed at the Open University and a similar system called Sindice developed by the University of Galway [55]. We shall integrate them in this very order as the development progresses.

4. Related Work and Envisaged Progress

In this section, we provide an overview of the state of the art in the field touched by the Service Provisioning Platform as described in the previous sections.

4.1 Service Modelling and Tagging

Service Modeling can be defined as the task in charge of generating services—simple or composite—that can then be used by end-users or developers to achieve their objectives. Services definition is the first and foremost task involved in the development of any service-oriented system and it has therefore been subject of much development and research. Work in this area range from pure business analysis to the deepest technical details of creating Web services [58]. In this section we shall limit ourselves to the IT landscape. The reader is referred to [59] for a broader review of the state of the art.

Research in service modelling from an IT perspective has mostly been dominated by work in Service-Oriented Architecture and related technologies [60, 61], and on Business Process Management. The former aims to support the development of IT systems out reusable, platform independent and loosely-coupled Web services. The latter tries to better support execution of business processes within and between enterprises through IT systems such as ERP, CRM, and Workflow Engines [62]. From a service provisioning perspective most of the research in the area has focussed on the definition of methodologies or overall approaches [63-65], on the theory behind processes and workflows [62, 64, 66, 67], and on the definition of languages for modelling processes and services [23, 66, 68-74].

There have been a number of research and industry strength products developed that support users in creating web services out of legacy code—see for instance IBM's WebSphere toolsuite [75] or Apache Woden [76]—or that help creating more complex processes like a number of Eclipse plugins such as the BPMN Modeler [77] and the BPEL designer [78], or commercial products such as IBM's WebSphere toosuite or Intalio's BPMN editor [79]. For brevity we shall not perform an exhaustive listing of these tools since they all provide a similar set of functionalities.

The approaches mentioned so far are mostly thought as desktop-based technologies in the sense that the editors are essentially local heavyweight applications. More recent trends on the Web around so-called Web 2.0 technologies are promoting what is commonly referred to as Mash-ups. Mash-ups are basically extremely simple processes which merge several sources of information. Mash-ups typically build upon a systematic application of REST principles allowing to save resources on the server-side both at runtime and at design time while creating the machinery for handling them [24, 80]. They therefore allow the provisioning of services over the Web using technologies other than SOAP or WSDL which are more common within enterprise settings.

Currently many web sites and companies provide part of their services via Web APIs so that they can take part into innovative mash-ups, see [20] for an extensive list. The proliferation of mash-ups has provoked the creation of a number of editors able to support users via Web-based editors. See for instance Yahoo! Pipes [1], the Open Mashups Studio [81] or Microsoft Popfly [82].

Despite the extensive set of technologies and tools developed around services, the uptake is not as it was initially expected. Despite the appealing characteristics of service-orientation principles and technologies, their application remains largely limited to large corporations and, effectively, we are still far from truly benefiting from the promised simplicity for constructing agile and interoperable systems. To address this, semantic Web services (SWS) combine Web services with semantic technologies in order to better support the discovery, composition and execution of Web services. Among the main initiatives in this respect we can mention WSMO [7], OWL-S [6], WSDL-S [8] and SAWSDL [25]. Along these

languages for defining semantic web services there have also been a number of tools created. For instance, WSMO Studio [83], WSMT [84] or the IRS Broker [85] support users in the creation of WSMO compliant semantic web services. Similarly, WSDL-S and OWL-S services can be created with the corresponding editors [86, 87]. Finally, editors such as WSMO Studio or Radiant [88] can generate SAWSDL compliant annotations, and there exist engines that use SAWSDL annotations for discovery like Lumina [89].

Even though the semantic web services languages and tools previously exposed have been around for some time already, these tools are typically desktop-based editors targeted at experts. Their inherent complexity reduces the scalability of the techniques employed and presents too big a knowledge-acquisition overhead, which is limiting the spread of semantic web services technologies. The Provisioning Platform devised herein is, to the best of our knowledge, the first fully-fledged world-wide scale framework for creating semantic web services through a Web-based intuitive user interface adapted to users with various degrees of expertise in services and semantic technologies.

A final body of work which is worth mentioning regards the application of folksonomies [90, 91] for annotating SWS. The underlying idea is to give the user the possibility to tag the services, instead of adopting complex SWS annotations (e.g. no pre- and post-conditions are defined). The resulting folksonomy – which increases in complexity with the number of the user tagging the services - can be used to infer relationships between services, and, for example, drive the automatic service composition. The work described in [90] is at a very early stage of development, and therefore no concrete implementations are described or provided. To the best of our knowledge, no other approaches are currently using tags in the area of SWS. The Provisioning Platform will therefore be among the first approaches to apply research in folksonomies to the area of services.

4.2 Natural Language Processing Systems

Natural Language Processing is a large area, which includes topics like text understanding and machine learning. We have concentrated on a subset: Information Extraction, a term which has come to be applied to the activity of automatically extracting pre-specified sorts of information from short, natural language texts.

In this section we give a brief overview of state-of-the-art systems which apply traditional IE techniques for semantic Web applications such as annotating Web pages with metadata. We shall first cover traditional approaches to Information Extraction and secondly we will introduce the main systems that make use of ontology-based techniques to support the processing.

4.2.1 Information Extraction Approaches

AeroDAML [92] is an annotation tool created by Lockheed Martin which applies IE techniques to automatically generate DAML annotations from Web pages. The aim is to provide naive users with a simple tool to create basic annotations without having to learn about ontologies, in order to reduce time and effort and to encourage people to semantically annotate their documents. Aero-DAML links most proper nouns and common types of relations with classes and properties in a DAML ontology.

There are two versions of the tool: a Web-enabled version which uses a default generic ontology, and a client-server version which supports customised ontologies. In both cases, the user enters a URI (for the former) and a filename (for the latter) and the system returns the DAML annotation for the Webpage or document. It provides a drag-and-drop tool to create static (manual) ontology mappings, and also includes some mappings to predefined ontologies.

Amilcare [93] is an IE system which has been integrated in several different annotation tools for the Semantic Web. It uses machine learning (ML) to adapt to new domains and

applications using only a set of annotated texts (training data). It has been adapted for use in the Semantic Web by simply monitoring the kinds of annotations produced by the user in training, and learning how to reproduce them. The traditional version of Amilcare adds XML annotations to documents (inline markup); the Semantic Web version leaves the original text unchanged and produces the extracted information as triples of the form <annotation, startPosition, endPosition> (standoff markup).

MnM [94] is a semantic annotation tool which provides support for annotating Web pages with semantic metadata. This support is semi-automatic, in that the user must provide some initial training information by manually annotating documents before the IE system (Amilcare) can take over. It integrates a Web browser, an ontology editor, and tools for IE, and has been described as "*an early example of next-generation ontology editors*" [94], because it is Web based and provides facilities for large-scale semantic annotation of Web pages.

S-CREAM (Semi-automatic CREAtion of Metadata) [95] is a tool which provides a mechanism for automatically annotating texts, given a set of training data which must be manually created by the user. It uses a combination of two tools: Onto-O-Mat, a manual annotation tool which implements the CREAM framework for creating relational metadata [96], and Amilcare. As with MnM, S-CREAM is trainable for different domains, provided that the user creates the necessary training data. It essentially works by aligning conceptual markup (which defines relational metadata) provided by Ont-O-Mat with semantic markup provided by Amilcare.

Despite the very promising results obtained by these tools, we have to note that none of them have been applied for the annotation of services. Typically these tools are generic and do not allow the user to customise them for particular domains other than through training. As a consequence one typical criticism is that they do not provide the user with a way to customise the integrated language technology directly. While many users would not need or want such customisation facilities, users who already have ontologies with rich instance data will benefit if they can make this data available to the IE components. However, this is not possible when "traditional" IE methods like Amilcare are used, because they are not aware of the existence of the user's ontology.

4.2.2 Ontology-Based Information Extraction Approaches

Ontology-Based IE (OBIE) uses of a formal ontology as one of the system's resources and typically make use of some reasoning at runtime. As a result, as opposed to IE techniques, OBIE techniques does not just find the (most specific) type of the extracted entity, but it also identifies it, by linking it to its semantic description in the instance base. This allows entities to be traced across documents and their descriptions to be enriched through the IE process.

Magpie [27, 97] is a suite of tools which supports the interpretation of Web pages and "collaborative sense-making". It annotates WebPages with metadata in a fully automatic fashion and needs no manual intervention by matching the text against instances in the ontology. It automatically populates an ontology from relevant Web sources, and can be used with different ontologies. The principle behind it is that it uses an ontology to provide a very specific and personalised viewpoint of the WebPages the user wishes to browse. This is important because different users often have different degrees of knowledge and/or familiarity with the information presented, and have different browsing needs and objectives. Magpie's main limitation is that it does not perform automatic population of the ontology with new instances, i.e., it is restricted only to matching mentions of already existing instances.

The PANKOW system (Pattern-based Annotation through Knowledge on the Web) [98] exploits surface patterns and the redundancy on the Web to categorise automatically instances from text with respect to a given ontology. The patterns are phrases like: the <INSTANCE> <CONCEPT> (e.g., the Ritz hotel) and <INSTANCE> is a <CONCEPT> (e.g., Novotel is a hotel). The system constructs patterns by identifying all proper names in the text

(using a part-of-speech tagger) and combining each one of them with each of the 58 concepts from their tourism ontology into a hypothesis. Each hypothesis is then checked against the Web via Google queries and the number of hits is used as a measure of the likelihood of this pattern being correct.

The system's best performance on this task in fully automatic mode is 24.9% while the human performance is 62.09%. However, when the system is used in semiautomatic mode, i.e., it suggests the top five most likely concepts and the user chooses among them, then the performance goes up to 49.56%. PANKOW therefore illustrates that semi-automated techniques can outperform to great extent automated techniques (with results quite close to those obtained by humans alone) by requiring strategic guidance and support from users.

The SemTag system [99] performs large-scale semantic annotation with respect to the TAP ontology, which contains about 65,000 instances. It first performs a lookup phase annotating all possible mentions of instances from the TAP ontology. In the second, disambiguation phase, SemTag uses a vector-space model to assign the correct ontological class or determine that this mention does not correspond to a class in TAP. The disambiguation is carried out by comparing the context of the current mention with the contexts of instances in TAP with compatible aliases, using a window of 10 words either side of the mention.

The SemTag system is based on a high-performance parallel architecture –Seeker, where each node annotates about 200 documents per second. The demand for such parallelism comes from the big volumes of data that need to be dealt with in many applications and make automatic semantic annotation the only feasible option. In general, a parallel architecture of a similar kind is often an important ingredient of large-scale automatic annotation approaches, whereas semi-automated techniques can reduce the overhead to an important extent by "utilizing" the user as one of the main sources of information and knowledge.

The Knowledge and Information Management system (KIM) is a product of OntoText Lab [100]. KIM is an extensible platform for semantics-based knowledge management which offers IE-based facilities for metadata creation, storage, and conceptual search. The system has a server-based core that performs ontology based IE and stores results in a central knowledge base. This server platform can then be used by diverse applications as a service for annotating and querying document spaces. The ontology-based Information Extraction in KIM produces annotations linked both to the ontological class and to the exact individual in the instance base. The instance base of KIM has been pre-populated with 200,000 entities of general importance that occur frequently in documents.

Like in the previous case, there exist tools that provide good results and relatively mature and scalable solutions. However, to the best of our knowledge, none of the tools exposed above have been applied and adapted for the annotation of services. Therefore, the WSMO-Lite and MicroWSMO editors part of the platform supported by the Annotations Recommender, will be the first editors to appropriate address the semi-automated annotation of services over the Web.

4.3 User Profiling and Recommender Systems

Three different approaches are known in literature in order to derive models for representing Web users and identify their interests: *collaborative filtering, content-based analysis, browsing behaviour modelling*; they differ on the basis of the data source used.

1) People interacting with *collaborative filtering* based systems have to actively express an interest, rating the contents they are viewing. This allows the system to give "friendly suggestions" (filter) based on the opinions of other users belonging to the same community (from this the term "collaborative").

Collaborative filtering systems have been proposed in [101]. In this cases, it is required an

active and explicit participation from the user community: each user has to rate the content of Usenet news articles. A form of automation it is here introduced by applying a *k*-nearest neighbour algorithm to find groups with similar interests.

For a complete review of the implicit participation systems see [102].

In a recent work of Sugiyama [103] user's profiles are derived from the choices made after a query submission to a search engine and from the contents of the following selected pages. A modified collaborative filtering is then applied to a user-term matrix (instead of user-item matrix in classic collaborative filtering). Users' term vectors are then compared to find homogeneous communities.

2) Content based recommendation systems build a model of relevant source of information and compare it with the contents which are of interest for the user. Collaborative filtering is here implicit, in the sense that user's choices are helpful to state the relevance of similar items. The main techniques applied in this field can be grouped in clustering [104], bayesian networks [105] and rule-based systems [106].

In [42] a computer science ontology is used for bootstrapping the current user's interests, in order to achieve the "cold start" problem arising when the user is unknown to the system. Documents viewed by the user are associated to a topic by using a variant of the nearest neighbour algorithm. Collaborative filtering is then performed on a user-topic matrix.

Content personalized Web pages present different information to different users and diverge from link personalization, that only adapts the link anchor structure and leave unmodified the substantial information part. In My Yahoo! [107] user's preferences are collected from explicit indication or semi-automated inference from navigation activity, asking the user to choose from general areas to more specific topics.

3) The *browsing behaviour modelling* approach analyzes the interactions between the user and the Web. Web-log data provide information about activities performed by a user from the moment he enters a Web site to the moment he leaves it [108] and allows us to separate browsing sessions. Session clustering is useful for discovering both groups of users, exhibiting similar browsing patterns, and groups of pages, having related contents (pages are clustered on the basis of how often they appear together across navigation patterns). Algorithms for sessions clustering can be classified into two approaches: *similarity-based* and *model-based* (or *probabilistic*) [108].

Compared to *similarity-based* methods, which assign user to a cluster only on the base of a given session similarity measure, *model-based* methods offer better interpretability: each model directly characterizes the corresponding cluster. Model-based clustering techniques have been widely used and have shown promising results in many applications involving Web data [104, 108].

More specifically, in the *model-based* approach, users' session clusters are generated as follows: a user arrives at the Web site in a particular time and is assigned to a cluster with some probability. The number of clusters is determined by using several probabilistic methods, such as BIC (*Bayesian Information Criterion*), bayesian approximations, or bootstrap methods [109]. The behaviour of each cluster is governed by a statistical model and the user's behaviour is generated from this model. Each cluster has a data-generating model with different components. Clusters are defined by learning the parameters of each probability distribution function, used to assign people to the various clusters, and the number of components. The model structure can be determined by model selection techniques and parameters estimated using maximum likelihood algorithms, e.g. the EM (Expectation-Maximization) [98].

5. Conclusions

In this document we have introduced the motivations, requirements and the design of the Service Provisioning Platform. We have provided a detailed design of the components involved, leaving the editing frameworks for [16, 37] that provide detailed design of these key components.

In order to simplify the creation of semantic web services the Provisioning Platform described in this deliverable will provide a set of tools allowing users to i) find relevant services taking into account user profiles and previous history of service usage, ii) annotate them helped by a recommender system and iii) persist the resulting semantic web services so that they can be used by anyone. Therefore in this respect, the Provisioning Platform aims to leverage users as the main source of information using interchangeably direct user input and automated processing informed by prior user-provided information to simplify the annotation of services.

Based on the semantic web services modelled by users, the Provisioning Platform will support users in the definition of composite semantic web services. To this end, it includes both a Mash-up and a Process Editor with which users can put together existing semantic web services in novel forms giving rise to new and more complex services. The Mash-up editor will allow users to create composite services that have a simple workflow. This kind of composite service will therefore allow any kind of user to create relatively complex services in an easy way in a similar vein to that of Yahoo! Pipes [1] for example. The Process editor on the other hand will address the requirements of more advanced users that may need to create complex workflows.

Essential to the overall vision of the Provisioning Platform is the central role played by users during the overall life-cycle of services. Users are at the same time service providers, when they define or compose new services, service consumers, when they utilize services or compose new services out of existing ones, and knowledge providers and/or consumers when they annotate or simply use services. An important aspect of this vision is that it blurs the distinction between service providers and service consumers which rather coexist in a transparent way allowing, for instance during the composition of services, the same individual to play both roles so that the newly created service can automatically become available for others. As a result, the overall platform benefits from an ever growing repository of services with increasingly richer annotations provided by users making the provisioning platform an extremely rich and dynamic services marketplace.

The implementation of the Service Provisioning Platform outlined here will be elaborated within future work and will be described in upcoming deliverables D2.1.3-D2.1.6 as well as 2.6.2 and 2.6.3.

6. References

1. Yahoo!: Yahoo! Pipes. http://pipes.yahoo.com/ (2008)
2. Booth, D., Liu, C.K.: Web Services Description Language (WSDL) Version 2.0 Part 0: Primer. W3C (2007)
3. Clement, L., Hately, A., von Riegen T. Rogers, C.: UDDI Specification Version 3.0.2. OASIS (2004)
4. Stollberg, M.: Scalable Semantic Web Service Discovery for Goal-driven Service-Oriented Architectures. Austria (2008)
5. Seekda: Seekda home page. http://seekda.com (2008)
6. Martin, D., Burstein, M., J., H., Lassila, O., McDermott, D., McIlraith, S., Paolucci, M., Parsia, B., Payne, T., Sirin, E., Srinivasan, N., Sycara, K.: OWL-S: Semantic Markup for Web Services. http://www.daml.org/services/owl-s/1.0/owl-s.pdf (2004)
7. Fensel, D., Lausen, H., Polleres, A., de Bruijn, J., Stollberg, M., Roman, D., Domingue, J.: Enabling Semantic Web Services: The Web Service Modeling Ontology. Springer (2007)
8. Akkiraju, R., Farrell, J., Miller, J., Nagarajan, M., Schmidt, M.-T., Sheth, A., Verma, K.: Web Service Semantics - WSDL-S. http://www.w3.org/Submission/WSDL-S/ (2005)
9. Norton, B., Pedrinaci, C., Domingue, J., Zaremba, M.: Semantic Execution Environments for Semantics-Enabled SOA. it - Methods and Applications of Informatics and Information Technology **Special Issue in Service-Oriented Architectures** (2008) 118--121
10. Fensel, D., Kerrigan, M., Zaremba, M. (eds.): Implementing Semantic Web Services: The SESA Framework. Springer (2008)
11. Traverso, P., Pistore, M.: Automated Composition of Semantic Web Services into Executable Processes. 3rd International Semantic Web Conference (ISWC 2004) (2004) 380--394
12. Abels, S., Lombardi, J.-P., Delchev, I., Pariente, T., Álvaro, G., Dimitrov, M.: First Demonstrator \& Interface Specification. EU FP7 SOAALL project (2009)
13. Gamma, E., Helm, R., Johnson, R., Vlissides, J.: Design Patterns: Elements of Reusable Object-Oriented Software. Addison-Wesley (1994)
14. Vogel, J., Schnabel, F., Mehandjiev, N.: Requirements Specification. EU FP7 SOAALL project (2008)
15. Richardson, M., Martínez, I.: Web21c Requirements. EU FP7 SOAALL project (2008)
16. Delchev, I., Vogel, J., Abels, S., Puram, S.: Specification of the SOAAll Process Editor. EU FP7 SOAALL project (2009)
17. Schreder, B., Villa, M., Abels, S., Zaremba, M.: Future C2C eCommerce Requirements and Scenario Descriptions. EU FP7 SOAALL project (2008)
18. OASIS Web Services Business Process Execution Language (WSBPEL) TC: Web Services Business Process Execution Language Version 2.0 Committee Specification. http://docs.oasis-open.org/wsbpel/2.0/CS01/wsbpel-v2.0-CS01.pdf (2007)
19. Musser, J.: ProgrammableWeb: 1000 web APIs. http://blog.programmableweb.com/2008/11/03/1000-web-apis (2008)
20. ProgrammableWeb: Web 2.0 API directory. http://www.programmableweb.com/apis/directory (2008)
21. Krummenacher, R., Hamerling, C., Lorre, J.-P., Baude, F., Legrand, V., Merle, P., Ruz, C., Pedrinaci, C., Liu, D., Richardson, M., Pariente, T.: SOAAll Reference Architecture Specification. EU FP7 SOAALL project (2009)
22. Schnabel, F., Born, M., Xu, L., González-Cabero, R., Lecue, F., Mehandjiev, N.: First Specification Of Lightweight Process Modelling Language. EU FP7 SOAALL project (2009)
23. Christensen, E., Curbera, F., Meredith, G., Weerawarana, S.: Web Services Description Language (WSDL) 1.1. http://www.w3.org/TR/wsdl (2001)
24. Richardson, L., Ruby, S.: RESTful Web Services. O'Reilly Media, Inc. (2007)
25. Farrell, J., Lausen, H.: Semantic Annotations for WSDL and XML Schema. http://www.w3.org/TR/sawsdl/ (2007)
26. Vitvar, T., Kopecky, J., Viskova, J., Fensel, D.: WSMO-Lite Annotations for Web Services. In: Hauswirth, M., Koubarakis, M., Bechhofer, S. (eds.): Proceedings of the 5th European Semantic Web Conference. Springer Verlag, Berlin, Heidelberg (2008)
27. Dzbor, M., Motta, E., Domingue, J.: Magpie: Experiences in supporting Semantic Web browsing. Web Semantics: Science, Services and Agents on the World Wide Web **5** (2007) 204--222
28. d'Aquin, M., Motta, E., Dzbor, M., Gridinoc, L., Heath, T., Sabou, M.: Collaborative Semantic Authoring. IEEE Intelligent Systems **23** (2008) 80--83

29. Kopecky, J., Gomadam, K., Vitvar, T.: hRESTS: an HTML Microformat for Describing RESTful Web Services. The 2008 IEEE/WIC/ACM International Conference on Web Intelligence (WI2008). IEEE CS Press, Sydney, Australia (2008)
30. Sabou, M., d'Aquin, M., Motta, E.: Exploring the Semantic Web as Background Knowledge for Ontology Matching. Journal of Data Semantics (2008)
31. Yahoo!: Flickr. http://www.flickr.com/ (2008)
32. Yahoo!: Delicious. http://delicious.com/ (2008)
33. Ayers, D., Heath, T.: Review Vocabulary. http://vocab.org/review/review.rdf (2007)
34. Heath, T., Motta, E.: Revyu: Linking reviews and ratings into the Web of Data. Web Semant. **6** (2008) 266--273
35. Linked Data Community: Linked Data - Connect Distributed Data across the Web. http://linkeddata.org/ (2008)
36. Domingue, J., Fensel, D., Davies, J., González-Cabero, R., Pedrinaci, C.: The Service Web: a Web of Billions of Services. BT Technology Journal **26** (2009) (To appear)
37. Álvaro, G., Cekov, L., Mehandjiev, N., Xu, L., Abels, S., Vogel, J., Simov, A., Maleshkova, M.: Service Modelling Tools Design. EU FP7 SOAALL project (2008)
38. Service Finder Consortium: Service Finder Classification Ontology. http://www.service-finder.eu/ontologies/service-categories.rdfs (2008)
39. Service Finder Consortium: Service Finder Home Page. http://www.service-finder.eu/ (2008)
40. TMForum: NGOSS Overview. http://www.tmforum.org/Overview/1912/home.html (2008)
41. Brunner, M., Steinmetz, N., Fabre, O., Dimitrov, M.: First Crawler Prototype. EU FP7 SOAALL project (2009)
42. Middleton, S.E., Shadbolt, N.R., De Roure, D.C.: Ontological user profiling in recommender systems. ACM Transactions on Information Systems **22** (2004) 54--88
43. Song, Y., Huang, J., Zhou, D., Zha, H., Giles, C.L.: IKNN: Informative K-Nearest Neighbor Pattern Classification. PKDD 2007: Proceedings of the 11th European conference on Principles and Practice of Knowledge Discovery in Databases. Springer-Verlag, Berlin, Heidelberg (2007) 248--264
44. Salton, G., Buckley, C.: Term Weighting Approaches in Automatic Text Retrieval. Readings in Information Retrieval, Ithaca, NY, USA (1987)
45. Wu, Z.: Verb semantics and lexical selection. Annual Meeting of the Association for Computational Linguistics (1994) 133--138
46. Leacock, C., Chodorow, M.: Combining local context with WordNet similarity for word sense identification. WordNet: A Lexical Reference System and its Application (1998)
47. Turney, P.D.: Mining the Web for Synonyms: PMI-IR versus LSA on TOEFL. Twelfth European Conference on Machine Learning (ECML-2001) (2001)
48. Mihalcea, R., Corley, C.: Corpus-based and knowledge-based measures of text semantic similarity. In AAAI'06 (2006) 775--780
49. Sebastiani, F.: Machine learning in automated text categorization. ACM Comput. Surv. **34** (2002) 1--47
50. Maguitman, A.G., Menczer, F., Erdinc, F., Roinestad, H., Vespignani, A.: Algorithmic Computation and Approximation of Semantic Similarity. World Wide Web **9** (2006) 431--456
51. Yang, Y., Pedersen, J.O.: A Comparative Study on Feature Selection in Text Categorization. ICML '97: Proceedings of the Fourteenth International Conference on Machine Learning. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA (1997) 412--420
52. Yih, W.-t., Goodman, J., Carvalho, V.R.: Finding advertising keywords on web pages. WWW '06: Proceedings of the 15th international conference on World Wide Web. ACM, New York, NY, USA (2006) 213--222
53. Shafiq, O., Vitvar, T., Gorroñogoitia, Y., Pedrinaci, C., Abels, S., Filali, I., Villa, M., Richardson, M., Huet, F., Gonzalez-Cabero, R.: Semantic Spaces: A Unified Semantic Data Coordination Infrastructure. EU FP7 SOAALL project (2008)
54. d'Aquin, M., Sabou, M., Motta, E., Angeletou, S., Gridinoc, L., Lopez, V., Zablith, F.: What Can be Done with the Semantic Web? An Overview Watson-based Applications. In: Gangemi, A., Keizer, J., Presutti, V., Stoermer, H. (eds.): SWAP, Vol. 426. CEUR-WS.org, Rome, Italy (2008)
55. Oren, E., Delbru, R., Catasta, M., Cyganiak, R., Stenzhorn, H., Tummarello, G.: Sindice.com: a document-oriented lookup index for open linked data. IJMSO **3** (2008) 37-52
56. Heß, A., Kushmerick, N.: Learning to Attach Semantic Metadata to Web Services. The SemanticWeb - ISWC 2003 (2003) 258--273
57. Álvaro, G., Abels, S., Mehandjiev, N., Lecue, F., Villa, M.: Service Consumption Platform Design. EU FP7 SOAALL project (2008)

58. Baida, Z., Gordijn, J., Omelayenko, B., Akkermans, H.: A Shared Service Terminology for Online Service Provisioning. Proceedings of the Sixth International Conference on Electronic Commerce (ICEC04), Delft, The Netherlands (2004)
59. Krummenacher, R., Vitvar, T., Perdinaci, C., Grenon, P., Agarwal, S., Vogel, J.: SOAALL Baseline and State of the Art. EU FP7 SOAALL project (2008)
60. Keen, M., Acharya, A., Bishop, S., Hopkins, A., Milinski, S., Nott, C., Robinson, R., Adams, J., Verschueren, P.: Patterns: Implementing an SOA Using an Enterprise Service Bus. IBM (2004)
61. Hohpe, G., Woolf, B.: Enterprise Integration Patterns: Designing, Building, and Deploying Messaging Solutions. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA (2003)
62. van der Aalst, W.M.P., ter Hofstede, A.H.M., Weske, M.: Business Process Management: A Survey. In: van der Aalst, W.M.P., ter Hofstede, A.H.M., Weske, M. (eds.): Business Process Management, Vol. 2678. Springer (2003) 1-12
63. Dustdar, S.: Reconciling Knowledge Management and Workflow Management Systems: The Activity-Based Knowledge Management Approach. Journal of Universal Computer Science **11** (2005) 589--604
64. van der Aalst, W., van Hee, K.: Workflow Management: Models, Methods, and Systems, Vol. 1. The MIT Press (2004)
65. Pedrinaci, C., Brelage, C., van Lessen, T., Domingue, J., Karastoyanova, D., Leymann, F.: Semantic Business Process Management: Scaling up the Management of Business Processes. Proceedings of the 2nd IEEE International Conference on Semantic Computing (ICSC) 2008. IEEE Computer Society, Santa Clara, CA, USA (2008)
66. Peterson, J.L.: Petri Net Theory and the Modeling of Systems. Prentice Hall PTR, Upper Saddle River, NJ, USA (1981)
67. van der Aalst, W.M.P., ter Hofstede, A.H.M., Kiepuszewski, B., Barros, A.P.: Workflow Patterns. Distributed and Parallel Databases **14** (2003) 5--51
68. van der Aalst, W., Dumas, M., ter Hofstede, A.: Web service composition languages: Old wine in new bottles? : Proceedings of EUROMICRO'03. IEEE Computer Society (2003) 298--307
69. Nitzsche, J., van Lessen, T., Karastoyanova, D., Leymann, F.: BPEL for Semantic Web Services (BPEL4SWS). On the Move to Meaningful Internet Systems 2007: OTM 2007 Workshops (2007) 179--188
70. Maurice, Bucciarone, A., Gnesi, S.: A survey on Service Composition Approaches: From Industrial Standards to Formal Methods. Istituto di Scienza e Tecnologie dell'Informazione, Consiglio Nazionale delle Ricerche (2006)
71. Leymann, F., Roller, D.: Modeling business processes with BPEL4WS. Information Systems and E-Business Management (2006) 1--20
72. Andrews, T., Curbera, F., Dholakia, H., Goland, Y., Klein, J., Leymann, F., Liu, K., Roller, D., Smith, D., Thatte, S., Trickovic, I., Weerawarana, S.: Business Process Execution Language for Web Services Version 1.1. http://www-128.ibm.com/developerworks/library/specification/ws-bpel/ (2003)
73. Object Management Group: Business Process Modeling Notation Specification - Final Adopted Specification. Available at http://www.bpmn.org (2006)
74. Mitra, N.: SOAP Version 1.2 Part 0: Primer. W3C Recommendation. http://www.w3.org/TR/soap12-part0/ (2003)
75. IBM Corporation: WebSphere. http://www-01.ibm.com/software/websphere/ (2008)
76. Apache Software Foundation: Woden. http://ws.apache.org/woden/ (2008)
77. Eclipse: BPMN Modeller. http://www.eclipse.org/projects/project_summary.php?projectid=stp.bpmnmodeler (2008)
78. Eclipse: BPEL Designer. http://www.eclipse.org/projects/project_summary.php?projectid=technology.bpel (2008)
79. Intalio: Business Process Management Suite. http://bpms.intalio.com/ (2008)
80. Fielding, R.T.: Architectural Styles and the Design of Network-based Software Architectures. (2000)
81. Open Mashups: Open Mashups Studio. http://www.open-mashups.org/ (2008)
82. Corporation, M.: Microsoft Popfly. http://www.popfly.com/ (2008)
83. Dimitrov, M., Simov, A., Konstantinov, M., Momtchev, V.: WSMO Studio - a Semantic Web Services Modelling Environment for WSMO (System Description). In: E.~Franconi, M.~Kifer, W.~May (eds.): Proceedings of the 4th European Semantic Web Conference (ESWC), Innsbruck, Austria (2007) 749--758
84. Kerrigan, M., Mocan, A., Tanler, M., Fensel, D.: The Web Service Modeling Toolkit - An Integrated Development Environment for Semantic Web Services. ESWC '07: Proceedings of the 4th European conference on The Semantic Web. Springer-Verlag, Berlin, Heidelberg (2007) 789--798

85. Domingue, J., Cabral, L., Galizia, S., Tanasescu, V., Gugliotta, A., Norton, B., Pedrinaci, C.: IRS-III: A broker-based approach to semantic Web services. Web Semantics: Science, Services and Agents on the World Wide Web **6** (2008) 109--132
86. Patil, A.A., Oundhakar, S.A., Sheth, A.P., Verma, K.: METEOR-S web service annotation framework. WWW '04: Proceedings of the 13th international conference on World Wide Web. ACM Press, New York, NY, USA (2004) 553--562
87. Elenius, D., Denker, G., Martin, D., Gilham, F., Khouri, J., Sadaati, S., Senanayake, R.: The OWL-S Editor–A Development Tool for Semantic Web Services. The Semantic Web: Research and Applications (2005) 78--92
88. Large Scale Distributed Information Systems -- University of Georgia: Radiant. http://lsdis.cs.uga.edu/projects/meteor-s/SAWSDL/ (2008)
89. Large Scale Distributed Information Systems -- University of Georgia: Lumina. http://lsdis.cs.uga.edu/projects/meteor-s/SAWSDL/ (2008)
90. Mika, P.: Ontologies are us: A unified model of social networks and semantics. Web Semant. **5** (2007) 5--15
91. Meyer, H., Weske, M.: Light-Weight Semantic Service Annotations through Tagging. In: Dan, A., Lamersdorf, W. (eds.): Service-Oriented Computing - ICSOC 2006, Vol. 4294. Springer, Heidelberg (2006) 465--470
92. Kogut, P.: AeroDAML: Applying Information Extraction to Generate DAML Annotations from Web Pages. First International Conference on Knowledge Capture (K-CAP 2001). Workshop on Knowledge Markup and Semantic Annotation (2001)
93. Ciravegna, F., Dingli, A., Wilks, Y., Petrelli, D.: Adaptive information extraction for document annotation in amilcare. SIGIR '02: Proceedings of the 25th annual international ACM SIGIR conference on Research and development in information retrieval. ACM, New York, NY, USA (2002) 451--451
94. Vargas-Vera, M., Motta, E., Domingue, J., Lanzoni, M., Stutt, A., Ciravegna, F.: MnM: Ontology Driven Semi-automatic and Automatic Support for Semantic Markup. EKAW '02: Proceedings of the 13th International Conference on Knowledge Engineering and Knowledge Management. Ontologies and the Semantic Web. Springer-Verlag, London, UK (2002) 379--391
95. Handschuh, S., Staab, S., Ciravegna, F.: S-CREAM - Semi-automatic CREAtion of Metadata. EKAW '02: Proceedings of the 13th International Conference on Knowledge Engineering and Knowledge Management. Ontologies and the Semantic Web. Springer-Verlag, London, UK (2002) 358--372
96. Handschuh, S., Staab, S., Maedche, A.: CREAM: creating relational metadata with a component-based, ontology-driven annotation framework. K-CAP '01: Proceedings of the 1st international conference on Knowledge capture. ACM, New York, NY, USA (2001) 76--83
97. Domingue, J., Dzbor, M.: Magpie: supporting browsing and navigation on the semantic web. IUI '04: Proceedings of the 9th international conference on Intelligent user interfaces. ACM, New York, NY, USA (2004) 191--197
98. Cimiano, P., Handschuh, S., Staab, S.: Towards the self-annotating web. WWW '04: Proceedings of the 13th international conference on World Wide Web. ACM, New York, NY, USA (2004) 462--471
99. Dill, S., Eiron, N., Gibson, D., Gruhl, D., Guha, R., Jhingran, A., Kanungo, T., Mccurley, K.S., Rajagopalan, S., Tomkins, A., Tomlin, J.A., Zien, J.Y.: A Case for Automated Large Scale Semantic Annotations. Journal of Web Semantics **1** (2003) 115--132
100. Kiryakov, A., Popov, B., Ognyanoff, D., Manov, D., Goranov, K.M.: Semantic annotation, indexing, and retrieval. Journal of Web Semantics **2** (2004) 49--79
101. Konstan, J.A., Miller, B.N., Maltz, D., Herlocker, J.L., Gordon, L.R., Riedl, J.: GroupLens: applying collaborative filtering to Usenet news. Commun. ACM **40** (1997) 77--87
102. Kappel, G., Proll, B., Retschitzegger, W., Schwinger, W.: Customisation for ubiquitous web applications\&\#58; a comparison of approaches. Int. J. Web Eng. Technol. **1** (2003) 79--111
103. Sugiyama, K., Hatano, K., Yoshikawa, M.: Adaptive web search based on user profile constructed without any effort from users. WWW '04: Proceedings of the 13th international conference on World Wide Web. ACM, New York, NY, USA (2004) 675--684
104. Padhraic: Modeling the Internet and the Web: Probabilistic Methods and Algorithms. (July,2003)
105. Breese, J.S., Heckerman, D., Kadie, C.: Empirical Analysis of Predictive Algorithms for Collaborative Filtering. Morgan Kaufmann (1998) 43--52
106. Sarwar, B., Karypis, G., Konstan, J., Riedl, J.: Analysis of recommendation algorithms for e-commerce. EC '00: Proceedings of the 2nd ACM conference on Electronic commerce. ACM, New York, NY, USA (2000) 1
107. Manber, U., Patel, A., Robison, J.: Experience with personalization of Yahoo! Commun. ACM **43** (2000) 35--39

108. Cadez, I., Heckerman, D., Meek, C., Smyth, P., White, S.: Model-based clustering and visualization of navigation patterns on a web site. (2000)
109. Fraley, C., Raftery, A.E.: How many clusters? Which clustering method? Answers via model-based cluster analysis. The Computer Journal **41** (1998) 578--588

Annex A. Annotations Recommender

The following sections provide some additional details and information on the architecture and the design of the Annotations Recommender.

Figure 15 visualizes the architecture overview of the Annotations Recommender. The graphical user interface is realized by the Simple SWS Editing Framework, which includes editors for WSMO Lite and MicroWSMO descriptions. In this way, the Annotations Recommender has no user interface of its own and performs mainly computation functionalities. The access to required input data is facilitated through the SOAALL Social Service Cloud, which enables data flow to the SWS Library, the Crawled Data and to External Services (i.e. Watson).

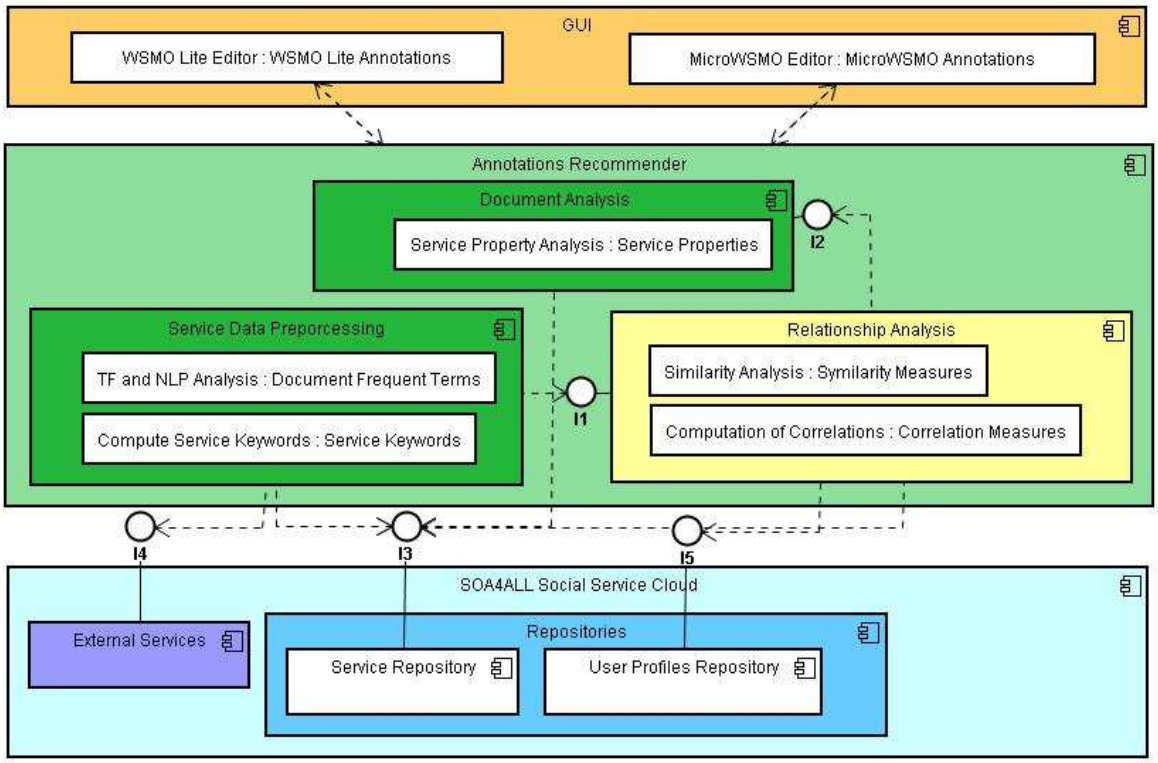

Figure 15. Annotations Recommender Components.

A description of the main use cases (Figure 16) was already given in the section 4.3.2 of this deliverable. The following activates diagrams, however, show in more detail the sub-activities includes within each main activity as well as the functionality differences in the implementations of the WSDL and RESTful service annotations recommendation.

Figure 17 shows an overall sequence of the activities involved in the Recommend Service Annotations use case. Each of the activities one through five are described in an extra activity diagram, which includes more detail on the functionality and the information flow.

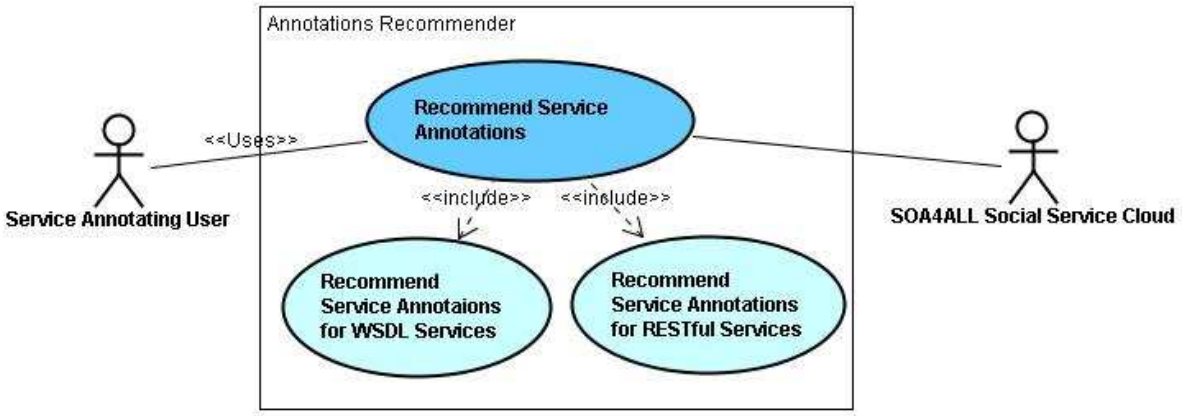

Figure 16. Use Case diagram "Recommend Service Annotations".

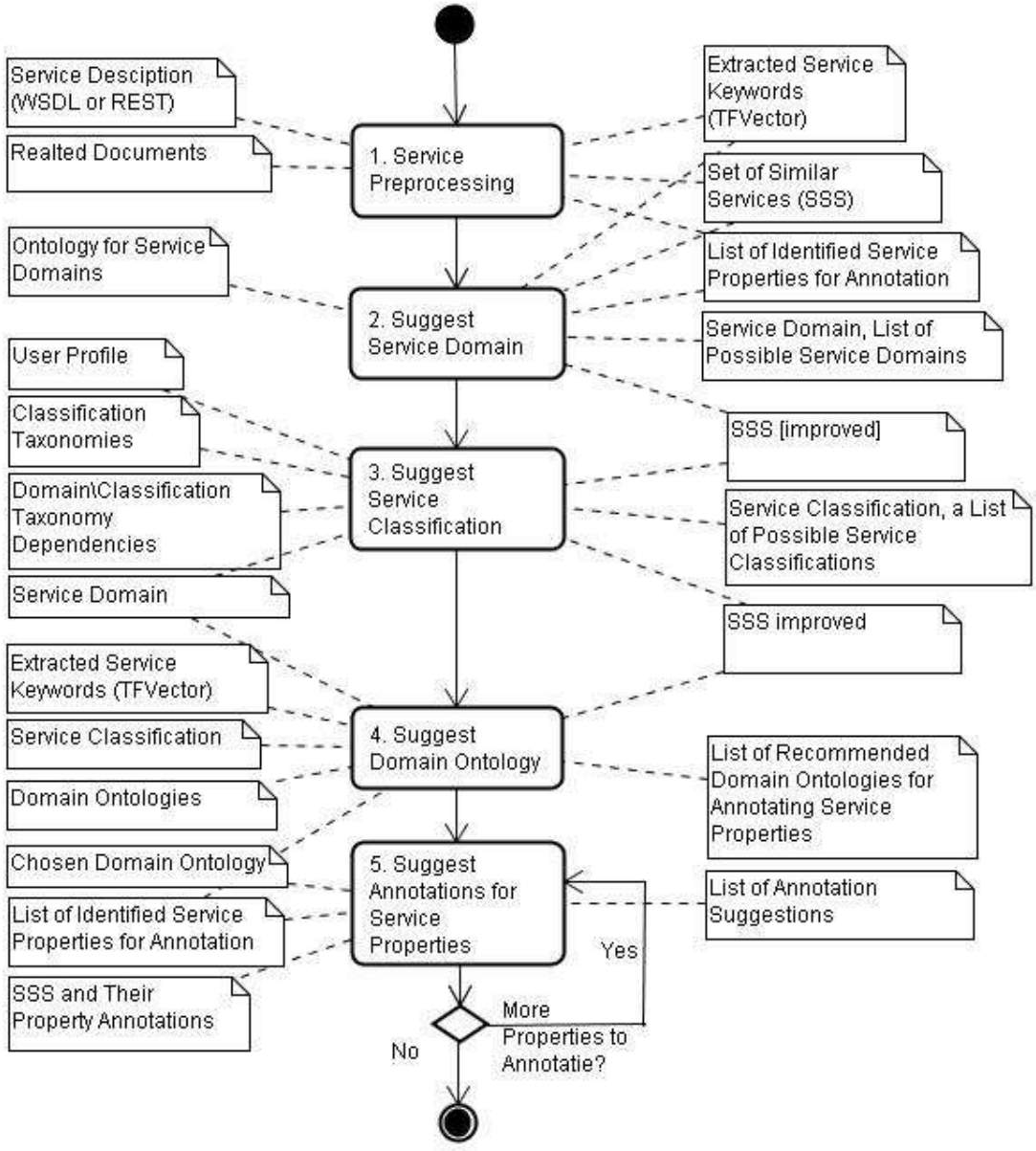

Figure 17. Activity Diagram: "Recommend Service Annotations".

Table 8. Service Preprocessing Artefacts.

Activity Name	Input	Output	Preconditions and Effects	Comments
1. Service Preprocessing	- Service Description -Service Related Documents	- List of Service Properties - Set of Similar Services (SSS) -Service Keywords	-Access to the Service Repository is required, which returns a Service and its Related Documents -The TF and Keywords extraction can be performed by using system-external services. A connection to these services would be a requirement.	- The Preprocessing does not involve any user interaction and can be performed anytime previous to the user-guided annotations - The SSS can be an empty set, if there is not enough training data, or annotated services.

Service Preprocessing (Figure 18, Table 8) includes activities for term frequency analysis, computation of service keywords and similarity analysis based on the k-Nearest Neighbour approach. Since, the service property analysis is not based on any of the results of the keywords analysis it can be preformed in parallel. The overall results of the activity are an initial SSS, the Service Keywords and the identified Service Properties.

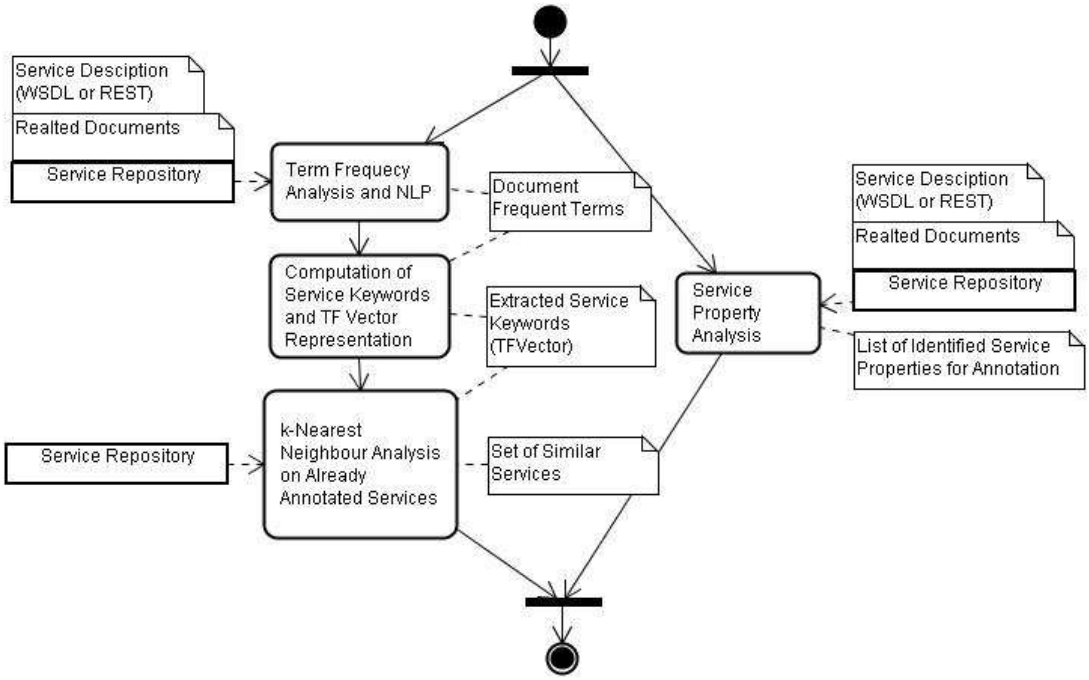

Figure 18. Activity Diagram: "Service Preprocessing".

Figure 19, in contrast to the general service preprocessing, shows additional activities, which have to be preformed specifically for RESTful services. After indentifying the service properties, these have to be marked with hREST mark-up, which is used as the basis for the following annotation in MicroWSMO format.

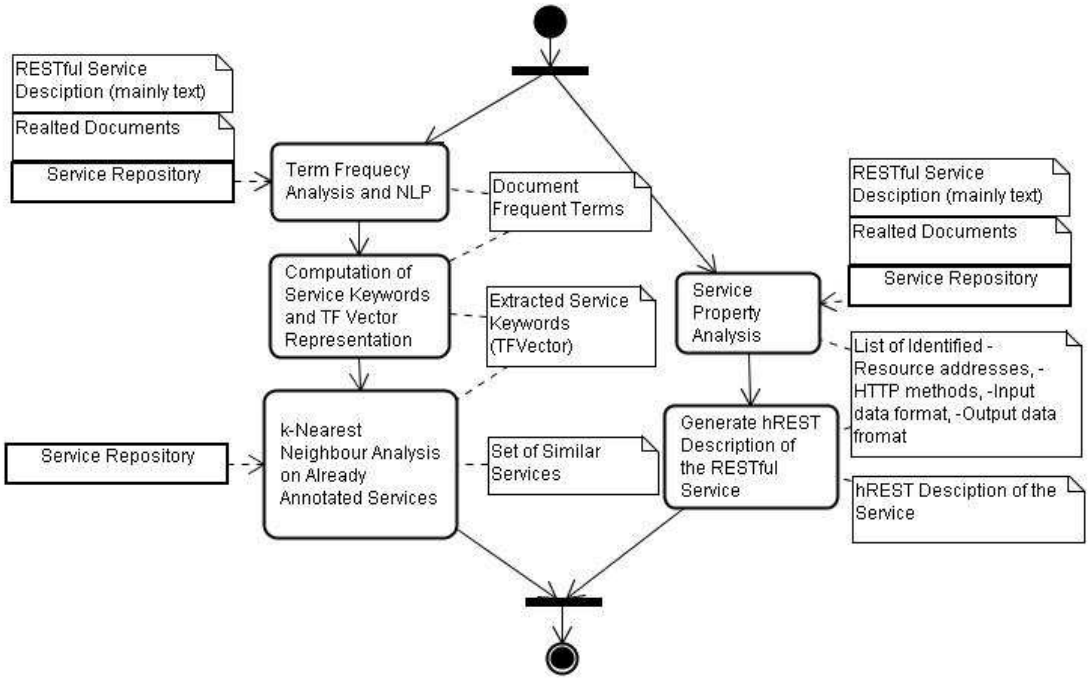

Figure 19. Activity Diagram: "Service Preprocessing for RESTful Services".

Table 9. Suggest Service Domain Artefacts.

Activity Name	Input	Output	Preconditions and Effects	Comments
2.Suggest Service Domain	-Service Keywords -Service Properties -Ontology for Service Domains -SSS	- Service Domain, automatically assigned to the Service - List of Possible Service Domains -SSS, which is narrowed down by assigning a Domain to the Service	- The Domain Analysis does not involve any user interaction and can be performed anytime previous to the user-guided annotations. The Result is one Domain assigned to the Service, and a list of possible Domains, from which the user can choose, if he wants to make a change.	-The generation of training data is a necessary and essential step. It must be done before the Annotation Recommender is in an operation state. -The SSS can be an empty set, if there is not enough training data, or annotated services.

The description of the Suggest Service Domain activity (Table 9, Figure 20) emphasizes the importance of providing training data and manually annotating service descriptions, in order to be able to properly compute similarity measures. In addition, it shows that additional measures have to be implemented in order to ensure a good accuracy of the annotations recommendation, if the SSS contains only a few entries. This can be done by performing a statistical analysis based on the correlations between service domain, classification and properties.

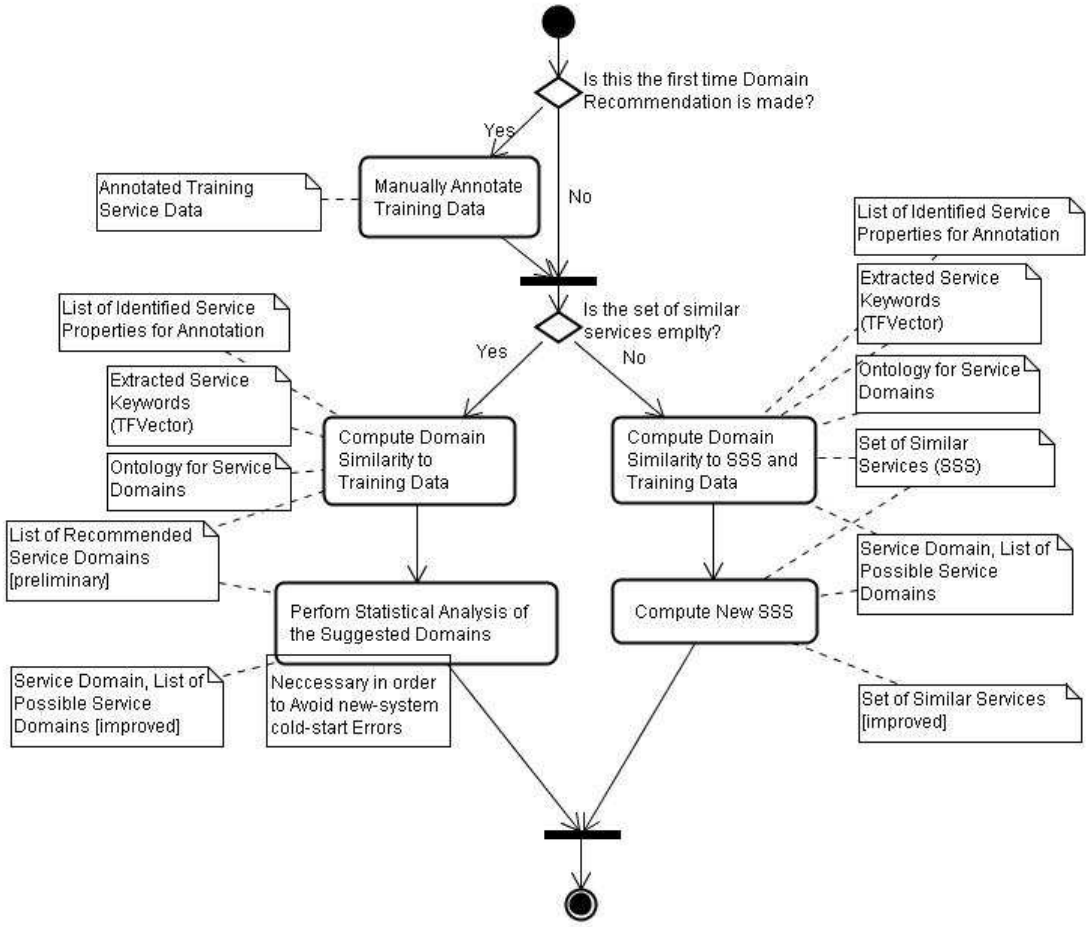

Figure 20. Activity Diagram: "Suggest Service Domain".

Table 10. Suggest Service Classification Artefacts.

Activity Name	Input	Output	Preconditions and Effects	Comments
3.Suggest Service Classification	-Service Domain -Domain\ Classification Dependencies -Classification Taxonomies -SSS -User Profile	- Service Classification, automatically assigned to the Service - List of Possible Service Classifications -SSS, which is narrowed down by assigning a Classification to the Service	- The Classification Analysis does not involve any user interaction and can be performed anytime previous to the user-guided annotations. Only the Classification based on User Profile requires that a user is logged onto the system (no user interaction is required).	-The User Profile, can be used in the same manner as the SSS. By computing similarity of the current user to other users, the proper Service classification can be determined. In contrast to the SSS similarity, User Profile similarity has to be determined based on an active user.

Once the service is assigned to a domain, a corresponding service classification can be computed (Figure 21, Table 10). If the SSS is empty, this is done on the basis of the Domain\Classification Taxonomy Dependencies document, which is manually composed, prior to the preprocessing process. However, if the SSS is not empty, it can be used to verify and improve the results based only on the correlation of domains and classification taxonomies.

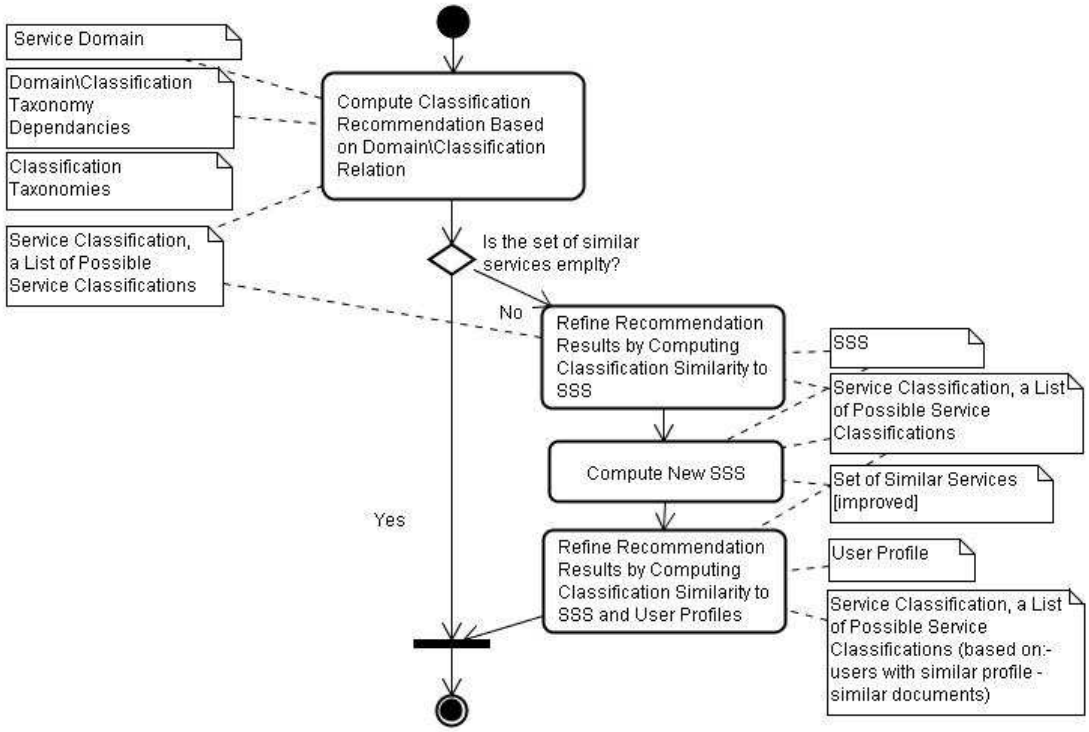

Figure 21. Activity Diagram: "Suggest Service Classification".

Table 11. Suggest Domain Ontology Artefacts.

Activity Name	Input	Output	Preconditions and Effects	Comments
4.Suggest Domain Ontology	-Service Classification -Service Domain -Domain Ontologies -Service Properties -Service Keywords	- List of Recommended Service Domain Ontologies, the user assigns one of them to the Service -SSS, which is narrowed down by assigning a Domain Ontology to the Service	- A connection to a service for searching for existing ontologies and semantic information (Watson, Sindice[8]) is required -This activity requires an active user to choose a Domain Ontology from the list with suggestions.	

The computation of suggestions for a domain ontology (Table 11, Figure 22) is based on querying Watson with the service keywords and adjusting the results based on the SSS. Recommendation based strictly on the SSS would be ineffective, since this would allow the user only to use ontologies, previously used and disable the introduction of new ones.

[8] Semantic Web search engine. http://sindice.com/

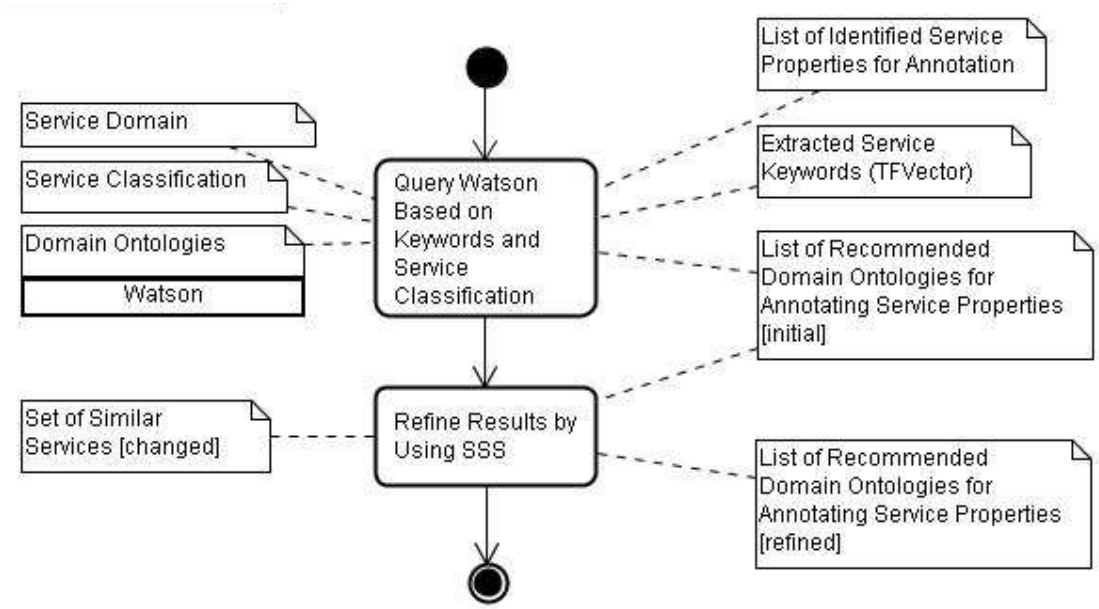

Figure 22. Activity Diagram: "Suggest Domain Ontology".

Table 12. Suggest Annotations for Service Properties Artefacts.

Activity Name	Input	Output	Preconditions and Effects	Comments
5.Suggest Annotations for Service Properties	-Domain ontology -Service Properties -SSS and Property Annotations	- List of Service Property Annotation Suggestions, from which the user has to choose	-In contrast to previous activities, this one requires continuous user interaction.	

The annotation of service properties is an iterative process, where the user annotates one property at a time. The list of annotations suggestions is computed based on the domain ontology and similarity measures to already annotated similar properties.

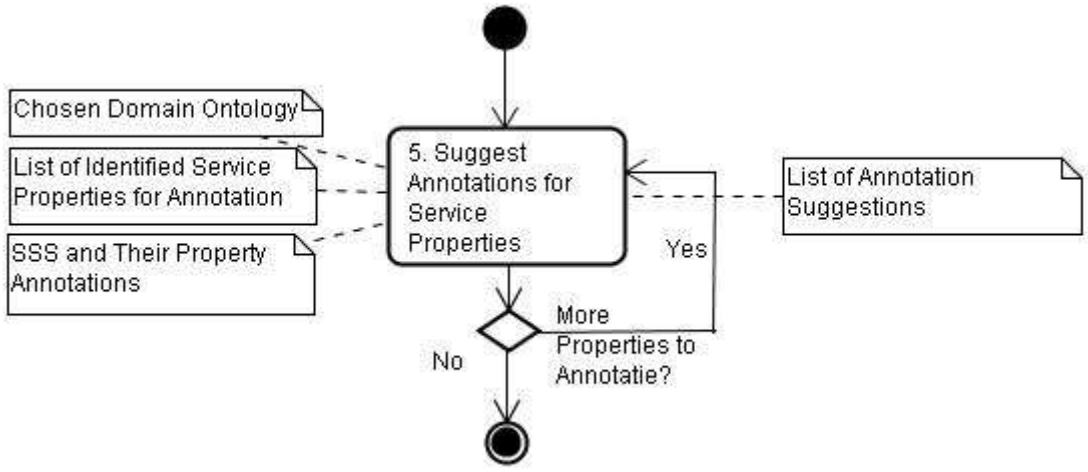

Figure 23. Activity Diagram: "Suggest annotations for Services".