KNOWLEDGE MEDIA INSTITUTE

Open Services on the Web

Technical Report kmi-09-5
November 2009

Maria Maleshkova

Maleshkova, M., Kopecky, J., and Pedrinaci, C. (2009) Adapting SAWSDL for Semantic Annotations of RESTful Services, Workshop: Beyond SAWSDL at OnTheMove, OTM2009, Vilamoura, Portugal, OTM 2009 Workshops

Maleshkova, M., Pedrinaci, C., and Domingue, J. (2009) Supporting the Creation of Semantic RESTful Service Descriptions, Workshop: Service Matchmaking and Resource Retrieval in the Semantic Web (SMR2) at 8th International Semantic Web Conference, Proceedings of ISWC '09, Washington D.C., USA

Open Services on the Web

Maria Maleshkova

First Year Probation Report
Knowledge Media Institute, The Open University,
Walton Hall, MK7 6AA, Milton Keynes,
United Kingdom

m.maleshkova@open.ac.uk

November 2009

Contents

Chapter 1

Introduction

The goal of the here described research is to explore the possibilities of combining Semantic Web [1] technologies and fundamental Web principles, including URIs and HTTP, and to apply these on open services on the Web, in order to contribute to a Semantic Web, which is not only an extension of the current Web with more semantic descriptions of data but is rather more dynamic and seamlessly integrates services as sources of that data, which can be automatically discovered, composed and executed by the computer on behalf of its user. This document is the **first year probation report** of this PhD study on open services on the Web.

The World Wide Web [2] has undergone significant changes since it was first launched and became popular. It provides ubiquitous access to heterogeneous data residing anywhere on the Internet, however, it is more than a system of static interlinked hypertext documents. The Web has evolved and is now not only a significant source of information, media and entertainment but is also a platform for communication and doing business, where individuals, but also companies and devices, can consume and provide content and services alike. The initial practice of publishing static web pages in plain HTML has developed into a generation of dynamic web pages populated at runtime, based on user requests and queries. Moreover, the value of Web applications is augmented in providing content to consumers but also in exposing functionality through an increasing number of public APIs designed for machine consumption.

Currently, more and more Web applications and APIs expose functionalities for providing access to resources in different representation forms. Some of these Web applications and APIs conform to the REST (Representational State Transfer [3]) architectural style and are therefore commonly refereed to as RESTful Web services [4]. RESTful services are characterized by their relative simplicity and their natural suitability for the Web. However, a good portion of the currently existing Web application and APIs do not strictly follow the REST principles but rather loosely adopt these notions, in order to provide an entry point for interaction with application resources. Therefore we introduce the term **open services on the Web**, to refer to Web Applications and APIs, which use URIs (sometimes parameterized) for identifying resources and links between them, and HTTP for retrieving and manipulating representations of these resources.

As a result open services on the Web use the same solutions that allow users to interact with any Web site by providing a browser with the HTTP URI, and implement them in order to enable the transferring and manipulation of representations of resource through predefined access points, rather than operating directly on those resources. For example, popular Web 2.0 applications by Facebook, Google, Flickr and Twitter offer easy-to-use, resource-oriented APIs, which provide simple access to diverse resources and also enable combining heterogeneous data coming from diverse services, in order to create data-oriented service compositions called mashups.

Even though Web applications and APIs are already widely used, their potential and wider adoption is hindered by the fact that their discovery, composition and invocation have to be completed manually and significant human labour has to be devoted to locating existing APIs, understanding their documentation, and implementing software that uses them. This is mainly because 1) the majority of open services on the Web are described in HTML pages and commonly **lack machine-processable descriptions**, 2) API implementations, in general, follow the fundamental Web principles but frequently **differ in the particular implementation details** and 3) open service descriptions have **no semantic metadata**, which limits the automation of discovery, composition and invocation, possible through the adoption of existing Semantic Web Service technologies. Initial progress towards tackling these challenges has already been made. In particular, there is some research on the formal description of RESTful services [5], [6] and on the machine-processable tagging of service properties [7]. On the other hand, advances in the semantic annotation of RESTful services have resulted in annotation formalism [8], [9] for describing service metadata.

Although, there is some pioneer research in the area of the semantic description of Web applications and APIs (e.g. [8], [7], [9], [10]), there is still a lot to be done until open services on the Web can be seamlessly integrated into the Semantic Web and be automatically discovered, composed and invoked. Therefore, the here presented work explores the potential of combining Semantic Web technologies and fundamental Web principles for the creation of **semantic descriptions of open services on the Web** as means for simplifying and automating the Web API-based interaction with resources.

This report is structured as follows: Chapter 1 details the background and the motivation for this work. It also describes the identified research problem and research questions, which need to be addressed. Chapter 2 presents related work and unaddressed research issues in the area. Chapter 3 provides details on the approach and results of initial prototype work. The report is concluded by a roadmap and planned future work.

1.1 Web Services

"Classical" Web services, based on WSDL and SOAP, have for a long time dominated the service world, playing a major role in the interoperability within and among enterprises and serving as the basic construct for the rapid development of low-cost and easy-to-compose distributed applications in heterogeneous environments [11]. ***Web Service*** [12] technology enables the publishing and consuming of functionalities of existing applications, facilitating the development of systems based on decoupled and

distributed components. Therefore, Web Services can be seen as reusable building blocks for creating open distributed systems, which allow companies and individuals to make their digital assets available to the global community in a simple and effective manner.

Web services, or more precisely services based on the Web services technology stack [12], are an established technology and are commonly used. However, despite their popularity there are still **only a few services available** (28 000[1] in comparison to 112 million of registered domains[2] and over 182 million websites[3]) and this number is not significantly increasing. Moreover, services based on WS-* (SOAP, WSDL, WS-Addressing, WS-Messaging and WS-Security, ...) [13] are argued to have **high complexity** and require trained developers, relatively sophisticated infrastructure or extensive tooling.

The Web Service Description Language (WSDL) [5], [14] is used to describe the service access endpoint, providing a machine-processable description of the structure of the service, its operations and the request and response messages. In addition, SOAP [15] is used for specifying the messages exchanged between the service consumer and provider. Both WSDL and SOAP as well as WS-Addressing, WS-Messaging, WS-Security, etc., add complexity and this can often lead to implementation difficulties and interoperability problems.

Web services complexity can be handled by trained developers, backed up by good tool support. However, a more critical problem hindering the greater success of the Web service technology is the **low level of automation of common tasks** such as discovery, composition and invocation [16], [17]. Currently, developers need to manually search for services in repositories such as UDDI, subsequently design service compositions and develop software that is able to invoke and manipulate them. In order to address this problem, ***semantic Web services*** [16], that is Web services enhanced with semantic descriptions of both functional and non-functional properties, have been proposed. The main idea underlying semantic Web services is the use of formal descriptions of Web service characteristics as means for automating many common tasks involved in using Web services. In this way, discovery, negotiation, composition and invocation can have a higher level of automation, when Web services are supplemented with semantic descriptions of their properties.

In contrast to Web services, which profit from a range of formalisms and description forms as well as from developed approaches for discovery, matchmaking and composition, research in the area of Web applications and APIs is still relatively limited. However, Web services are a good starting point for analyzing some of the problems, common for both types of services, and enable the reusing of solutions provided by semantic Web service technologies as a basis for developing solutions in the Web API area.

[1] Source: `http://webservices.seekda.com/`

[2] Daily updated statistics at `http://www.domaintools.com/internet-statistics/`.

[3] Source: Netcraft `http://news.netcraft.com/archives/2008/10/29/october_2008_web_server_survey.html`

1.2 Open Service on the Web

Currently, there is an increasing importance and use of Web applications and APIs and sometimes it is even argued that they have displaced Web services [18]. This trend is strongly supported by the growing popularity and use of Web 2.0 technologies, because many Web 2.0 applications offer Web APIs as access points for interaction with application resources. Moreover, Web APIs enable combining heterogeneous data coming from diverse sources, in order to create data-oriented compositions called mashups [19].

Web APIs, which conform to the REST principles are commonly referred to as RESTful Web services. The representational state transfer (REST) paradigm was first described by Roy Fielding in his Ph.D. dissertation [4] denoting an architecture style of large-scale distributed systems. A RESTful web service is a simple web service based on REST principles and commonly implemented by using HTTP. RESTful services comprise a collection of resources and have the following main principles[4]:

- *Uniquely identified resources*: All resources are identified by URIs [20], which are unique and enable the addressing and identifying of resources in a global space. In the context of services, a RESTful service exposes a set of resources, which identify the endpoints for interaction with clients.

- *Fixed set of operations*: All resources can be manipulated by using only four fixed operations: GET, POST, PUT, DELETE. GET and POST retrieve or transfer the current state of a resource, correspondingly. In a similar way, PUT and DELETE create a new resource and delete a new resource. These operations result from the fact that RESTful services are commonly based on HTTP for transferring messages.

- *Stateless client/server architecture*: Every interaction with a resource is stateless and each request is complete and contains all information necessary for the server to process it. Response messages can contain states, which point to valid future states of interaction. However, the interaction itself does not modify the state of the process.

- *Resource representation decoupling*: Resource content can be accessed via different formats, since resources are not bound to one specific representation.

It is important to point out that currently, most available Web APIs do not follow all of these REST principles. Commonly, Web APIs use uniquely identified resources, enable resource representation decoupling and use a fixed set of operations. However, for example, often POST and GET operations are misused for deleting or modifying resources and requests are sent in a stateful manner. In addition, very often URI are defined in a parameterized manner, so that different resources are retrieved for different parameter values. In order to acknowledge the fact that not all Web APIs are strictly RESTful, we introduce the term "open service on the Web" to refer to Web applications and APIs, which are based on the fundamental Web technologies (URIs for identifying

[4] A complete and exhaustive list is available in [4].

entities and links between them, and HTTP) and provide access points for interaction with resources.

Open services on the Web have a number of advantages in comparison to WSDL [5]-based services. They are simpler to use because they are not burdened by the complexity of the WS technology stack and instead are based on HTTP, URIs and links between resources, which are also the underlying principles of the Web. Therefore, open service have a natural suitability for the Web, as opposed to WSDL-based services, which abuse HTTP for tunneling SOAP [21]. The simplicity of Web APIs at the technological level results in the fact that developers do not need extensive training and can create, invoke and test implementations without greater efforts. Open services on the Web can be described in simple HTML pages, they can be tested from an ordinary Web browser and their development is very similar to developing a dynamic web site. As a result, open services are very accessible and users with little training can create service descriptions, invoke services, and even make simple resource-oriented compositions.

1.3 Motivation and Research Problem

Despite their technological simplicity and ease of implementation in comparison to WSDL-based services, open services on the Web have one fundamental limitation: **the lack of formal and machine-processable descriptions**. While, WSDL provides a machine-processable description of the structure of Web services, most Web APIs are usually textually described in HTML pages and documents, following no strict guidelines or format. As a result, developers need to spend time and effort on reading and understanding service descriptions, which is a major hindrance for the automated processing.

In addition, the task of finding services also has to be completed manually by searching the very few existing Web API repositories, which contain services manually collected over time, or by using common search engines, which do no differentiate between Web API descriptions and normal HTML pages. Similarly, mashups have to be programatically created, even though, there are already some tools available, which provide developer support[5]. The resulting solutions are custom-tailored to fulfill the particular development goal and do not support the reusability of the included Web APIs.

Finally, similarly to the other service-related tasks, the invocation of individual services or service compositions has to be done manually as well. Invocation is particularly challenging because, even though, all open services on the Web are based on the same underlying principles, they frequently differ in the particular implementation details, which effect the invocation method. In addition, the HTML documentation often does not provide details about the service functionality, which is in accordance with the black-box service realization principles, but is extremely hindering if it influences the way in which the service needs to be invoked. User authentication is also an issue, which makes invocation more challenging, since the majority of the existing Web APIs

[5]Some available tools, which support the manual composition of a restricted number of services are Yahoo Pipes (`http://pipes.yahoo.com/pipes/`) and DERI Pipes `http://pipes.deri.org/`

require a user to have an API Key or to be registered to use the service. As a result, developers have to devote extra effort ensuring proper invocation and its automation is even more difficult.

Web APIs and applications have already had some initial success and are becoming increasingly adopted[6]. Open services on the Web have great potential because they are based on the technologies of the Web, which have proven to be scalable and lack the overhead introduced by the WS-* stack. In addition, they are easily accessible and can seamlessly be integrated in the Semantic Web, where the user does not have to differentiate whether he/she is getting information directly from a Web site or from a Web API or even a mashup. In order to overcome the current challenges faced by open services and to enable their wider adoption, we propose **semantic service descriptions of open services on the Web**, which are both machine-processable and, similarly to traditional SWS that improve the automation of WSDL-based solutions, bring automation to common tasks such as discovery, composition and invocation.

Therefore, the main research problem addressed in this PhD work is **How to create semantic service descriptions of open services on the Web?** This main research problem is tackled by exploring the answers of the following specific research questions:

- **RQ1:** How can **semantic descriptions of open services on the Web** be created? In particular, this requires the analysis of approaches for identifying service properties within existing HTML descriptions and enchanting these with semantic information. The annotation of services with semantic information can serve different purposes and can be used for completely different tasks. For example, enhancing services with information about their functionality is very helpful for finding services, however, it does not necessarily support their invocation. Therefore, it is necessary to define 1) the minimal required, but also sufficient, **semantic information**, which needs to be included in a semantic service description, in order to support the automation of the discovery, composition and invocation tasks; 2) the semantic **description form**, which supports the provisioning of these annotations.

- **RQ2:** How can the **creation of semantic open services on the Web be supported**? Provided with a formalism for the semantic description of open services on the Web, developers need to be supported by tools and guidelines, which make the creation of semantic descriptions easier. The goal is to hide formalism complexities by providing extensive tool support and to automate as much as possible the creation of semantic descriptions: based on already existing textual descriptions; based on support for finding ontologies necessary for the semantic annotation; and based on recommendation of suitable annotations.

The addressing of these two main research questions will directly contribute to integrating open services on the Web in the Semantic Web. The Semantic Web can serve as a valuable source of background knowledge, which will be used to assist the process of creating semantic descriptions. In addition, already existing services can be facilitated

[6]The number of Web APIs and existing mashups at `http://www.programmableweb.com` has constantly increased during the past year.

and integrated in the user support for semantically annotating services, and the implementation results of the research can be provided in the form of new semantic services that build upon the Web and provide new functionalities for others to exploit. In this way, the Semantic Web becomes a space for both retrieving and publishing semantic service-related data.

The existence of semantic data and common information models plays a major role for the creation of semantic descriptions of services. Initial research in the area of semantic Web services was forced to rely on information extraction and data analysis approaches, in combination with ontology learning, in order to acquire the necessary metadata and models for the semantic annotation [22], [23], [24]. Currently, this situation has changed and there exists a good basis of semantic data, which can be used to directly describe service properties or to serve as feedback information for improving the annotation process. This development can be further supported by an increased use and popularity of Linked Data.

The main underlying idea behind **Linked Data** [25] is that the same principles that enable the Web of documents to be so successful have to be applied on available structured data as well. This is achieved by following a set of best practices for publishing and connecting data on the Web, which lead to the enrichment of the Web with a global data space connecting data coming from diverse domains. The resulting **Web of Data** [25] can be used as basis for developing a wide variety of new applications, which can dynamically analyze and interpret the data, browse from one data source to another by following the links, or collect and query data related to particular topic of interest. More importantly, in the context of open service on the Web, linked data can be used as background information, used as support for suggesting service annotations, but also it already provides a set of ontologies, which can directly be used for semantically describing services. In addition, the resulting semantic descriptions of open services on the Web can be published by following the linked data principles and be directly integrated in the Web of data.

The following chapter provides an overview of the state of the art relevant to answering the here formulated research questions.

Chapter 2

State of the Art

This chapter focuses on current research related to the creation of semantic descriptions of open services on the Web. We divide the state of the art in three main sections. The first two sections focus on work related to RQ1. First, we provided an overview of the existing formalisms for the description of Web services and Web APIs and an analysis of research work in the area of semantic description for Web services. The following sections provides and introduction to Linked Data principles and address state of the art related to user support (RQ2). Since, the manual annotation of Web services is an effort- and time-consuming task, we consider existing research contributing to the automation of this task. This includes approaches for the recommendation of semantic annotations, service classification and finding suitable ontologies. Finally, existing tools for ontology visualization and service annotation are described.

2.1 Descriptions of Web Services and Open Services on the Web

Web services enable the publishing and consuming of functionalities of existing applications, facilitating the development of systems based on decoupled and distributed components. **WSDL** [5], [14] is an XML-based language for describing the interface of a Web service. A WSDL service description specifies: 1) the supported operations for consuming the Web service; 2) its transport protocol bindings; 3) the message exchange format; and 4) its physical location. In this way, the WSDL description contains all information necessary for invoking a service and since it is XML-based, conforming to the WSDL XML schema, it is also machine-processable.

WSDL. Similarly, to Web services, which provide access to the functionality of existing components, Web APIs and Web applications conforming to the REST paradigm, provide access to the state of existing resources, by using the WWW as an infrastructure platform. Based on this parallel between the two types of services, WSDL was extended to Version 2.0 [5], which can also be used for formally describing RESTful services. As a result WSDL is a machine-processable, platform and language independent form of describing Web services and RESTful services alike.

The difficulty of using WSDL for RESTful services, is that it was not especially designed for resource-oriented services and as a result, everything has to be described in an operation-based manner. In addition, WSDL introduces some difficulties with specifying the message exchange format and limits HTTP authentication methods. Moreover, the most important drawback is that it lacks support for simple links. There is no mechanism in WSDL 2.0 to describe new resources that are identified by links in other documents. However, one of the most important characteristic of RESTful services is that they consist of collections of interlinked resources.

WADL. In contrast to WSDL, the Web Application Description Language (WADL) [6] was especially designed for describing RESTful services in a machine-processable way. It is also XML-based and is platform and language independent. As opposed to WSDL, WADL models the resources provided by a service, and the relationships between them in the form of links. A service is described using a set of resource elements. Each resource contains descriptions of the inputs and method elements, including the request and response for the resource. In particular, the request element specifies how to represent the input, what types are required and any specific HTTP headers. The response describes the representation of the service's response, as well as any fault information, to deal with errors.

Currently, both WADL and WSDL for RESTful services are not widely used. A Google search for WADL files returns only 49 unique results[1], while from the popular Web 2.0 applications only delicious[2] and YahooSearch[3] have WADL descriptions. The main difficulty of using WADL descriptions is that they are complex, which requires that developers have a certain level of training and tool support that enables the application development with WADL. This complexity contradicts the simplicity of the open services on the Web principles, which enable the easy retrieving of resources only thought an HTTP request, directly in the Web browser. In addition, the majority of the Web API descriptions are usually given in textual form in Web pages. This description form is supported by the increasing popularity of Web 2.0 applications such as Facebook, Twitter, Last.fm, which are user-centric and provide access to their resources through Web APIs, which are described in **plain HTML**.

HTML-based Web API descriptions follow no standards or guidelines regarding the form or the content of the description. Therefore, it is up to the developer to decide what structure to use and what information to provide. As a result, everyone who is able to create a Web page is also able to create a Web API desciption. Probably, this is also one of the main reasons why this description form is so popular. However, plain text/HTML descriptions, in contrast to WSDL descriptions, are not machine-processable, which means that if a developer wants to use a particular service, he/she has to go to an existing description Web page, study it and write the application manually. Therefore, current research proposes the creation of machine-processable descriptions on top of existing HTML descriptions by using **microformats** [26]. Microformats offer means for annotating human-oriented Web pages in order to make key information machine-processable, without modifying the visualization or the content.

[1] Search done on October 5th, 2009

[2] http://delicious.com/

[3] http://search.yahoo.com/

hRESTS. One particular approach for creating machine-processable descriptions for RESTful services by using microformats is hRESTS (HTML for RESTful Services) [7]. hRESTS enables the marking of service properties including operations, inputs and outputs, HTTP methods and labels, by inserting HTML tags within the HTML. In this way, the user does not see any changes in the Web page, however, based on the tags the service can be automatically recognized by crawlers and the service properties can directly be extracted by applying a simple XSL transformation.

RDFa. An alternative to using hRESTS is offered by RDFa [27] that enables the embedding of RDF data in HTML. RDFa is similar to using microformats, but is somewhat more complex and offers more HTML marking options, as opposed to hRESTS. Approaches, based on making existing RESTful service descriptions machine-processable by using HTML tags are simpler and more lightweight as opposed to WSDL and WADL. In addition, they can be applied directly on already available descriptions, rather then creating new service descriptions from scratch. The adoption by developers is also easier, since the creation of a machine-processable RESTful service description is equivalent to Web content creation or modification.

Because of these characteristics and because of the fact that currently the majority of Web applications and APIs are described in HTML Web pages, we chose to use the lightweight hRESTS approach for out initial work on creating semantic RESTful service descriptions. Moreover, hRESTS provides a good basis for providing semantic annotations, which can be used to improve the automation of service discovery, composition and invocation tasks, while WADL does not support the addition of semantic metadata.

2.2 Semantic Descriptions of Web Services and Web APIs

This section focuses on existing research work related to the semantic description of open services on the Web, addressing in this way approaches related to RQ1.

Machine-processable service descriptions support the creation, publication and consumption of services. However, the service descriptions remain on a syntactic level with a strong technical focus. This requires that the identification of suitable Web services for a given task is performed through manual inspection. In oder to address this challenge, research on **semantic Web services** (SWS) has been devoted to reducing the extensive manual effort required for manipulating Web services [17], [16]. The main idea behind this research is that tasks such as the discovery, negotiation, composition and invocation of Web services can have a higher level of automation, when services are enhanced with semantic descriptions of their properties. In particular, SWS applies inference-based techniques on formalized semantic descriptions, in order to better support the analysis of Web services. Moreover, most SWS technologies use ontologies as the underlying data model, which provide means for tackling the interoperability problem at the semantic level and, more importantly, enable the integration of Web services within the Semantic Web.

Since open services on the Web face similar challenges as WSDL-based Web ser-

vices, including the lack of automation of common service tasks, research related to SWS descriptions and approaches should be carefully considered as a basis for semantically enchanting open services on the Web as well. In addition, if Web APIs can be semantically described by directly applying or by adapting existing SWS formalisms, this would support the reuse of already developed semantic-enabled discovery, composition and invocation approaches. Moreover, the integration of open services in the Semantic Web can be facilitated by the same SWS technologies, which are the basis for integration of Web services.

OWL-S. The foundation for the SWS research is laid by a number of standardized notations for the formal specification of Web services with semantically rich descriptions. One common approach is facilitated by OWL-S [28], which defines an upper ontology for semantically annotating Web services. OWL-S consist of three main parts: 1) the *Service Profile* for advertising and discovering services, which contains the name of the Web service, its provider, a natural language description, and a formal functional description that is defined in terms of inputs and outputs, preconditions and effects; 2) the *Process Model*, which gives a detailed description of a service's operation and describes how the Web service works; and 3) the *Service Grounding*, which provides details on how to access the service and how to interoperate with a service, via messages.

The service profile supports manual service discovery, based on the the natural language descriptions, as well as automatic discovery, based on the formal functional description. The service model provides information for determining whether the communication between a client and the Web service can be carried out successfully. Finally, the service grounding defines the mapping between the semantic descriptions and technological implementation in order to conduct the actual message exchange. The commonly pointed out weakness of OWL-S are related to conceptual insufficiencies, in particular, addressing the inadequacy of the process description language [29]. Related to open services on the Web services, OWL-S provides a good basis for identifying the semantic information, which needs to be included in the service description, in order to effectively support the discovery, composition and invocation tasks.

WSMO. Another SWS description formalism is the Web Service Modeling Ontology (WSMO) [30]. It defines four top-level notions including: 1) *Ontologies* that define formalized domain knowledge; 2) *Goals* that describe the client's objective in using the Web service; 3) semantic description of Web services; and 4) *Mediators* for enabling interoperability and handling heterogeneity. In contrast to other description formalisms, WSMO propagates a **goal-based** approach for semantic Web services. A client formulates requests in terms of goals, which formally describe the objective to be achieved while abstracting from technical details. The task of the system is to automatically discover and execute Web services, which satisfy the client's goal. In this way, Web services are not described only in terms of their capabilities but also in terms of problems that they solve. This approach is supported by the integrated mediators, which enable the handling of potentially occurring heterogeneities.

WSMO is more exhaustive than OWL-S and promotes a goal-driven approach, which goes beyond the idea of merely annotating Web services, aiming to provide means of expressing service offers and needs. In addition, WSMO defines an own specification language WSML [31] that covers all ontology languages that are consid-

ered for the Semantic Web, and provides reasoners for them. Moreover, there are implementations of execution environments for Semantic Web services, namely WSMX as the WSMO reference implementation (see www.wsmx.org) and IRS as a goal-based broker for Semantic Web services [32].

Similarly to OWL-S, WSMO provides details about the semantic information necessary for creating semantic descriptions of open services on the Web, which support the automation of service tasks. In addition, WSMO provides a new perspective to services by introducing the concept of goals. Maybe, this approach can successfully be applied on open services, by describing services in terms of resource or data objectives. Still, it is important to point out that both OWL-S and WSMO bring a certain level of complexity, which requires developers with good ontological knowledge and experience in using the formalisms. This somehow contradicts the current trend in Web APIs, where the majority of the descriptions are given in simple HTML and the technology is more accessible to less-trained developers.

SAWSDL. A more lightweight description formalism is the Semantic Annotations for WSDL and XML Schema (SAWSDL) [33], which currently is the only official W3C recommendation for semantic Web services. It defines a set of extension attributes for WSDL, which enable the linking of semantic entities to the service description. SAWSDL consists of two parts: 1) Mappings of XML schema definitions to ontology concepts, which specify the translation of data between the semantic layer and the layer of technical implementation; and 2) Semantic annotation of WSDL operations through references to ontology concepts.

Although SAWSDL service annotations are not as expressive as OWL-S or WSMO ones and adding of semantic information is limited to service properties associated with semantic concepts, several recent works build on this light-weight SWS approach [34], [8]. In the context of open services on the Web, the SAWSDL approach is very promising because it enables adding semantic information directly on top of existing service descriptions, by liking service keywords to ontology elements.

WSMO-Lite. SAWSDL supports the development of lightweight service description formats, in order to enable the creation of semantic Web service descriptions based on limited amounts of service information. Such a lightweight format is WSMO-Lite [34], which supports lightweight descriptions of WSDL services, by filling the SAWSDL annotations with concrete service semantics. WSMO-Lite defines four aspects of service semantics including: 1) *Information Model*, which defines the data model for input, output and fault messages; 2) *Functional Semantics*, which define the functionality, which the service offers to a client when it is invoked; 3) *Behavioral Semantics*, which define details specific to a service provider, or the service implementation or its running environment; and 4) *Nonfunctional Descriptions*, which define nonfunctional service properties such as quality of service or price. These service semantic are linked to a WSDL description by applying the SAWSDL annotation approach.

In particular, WSMO-Lite supports two types of annotations, namely *reference annotations* and *transformation annotations*. A reference annotation points from a WSDL component (XML Schema message or type definition, interface, operation, binding, service, or endpoint) to a WSMO-Lite semantic concept. A transformation annotation specifies a data transformation called *lifting* from a component of XML schema to an

element of ontology; and a reverse transformation (from ontology to XML) called *lowering*. A major advantage of the WSMO-Lite approach is that it provides means for describing service semantics without really being bound to a particular service description format (for example, WSDL). As a result, service properties, for example, marked with hRESTS, in an HTML description of a RESTful service can be used as annotation points for linking WSMO-Lite semantic concepts in the same way as using service properties described in WSDL.

MicroWSMO. In comparison to WSDL-based services, which have a somewhat longer history of research on semantic descriptions and annotation approaches, research in the area of semantic RESTful services is newer and therefore relatively limited. MicroWSMO [8] is a formalism for the semantic description of RESTful services, which is based on adapting the SAWSDL approach. MicroWSMO uses microformats for adding semantic information on top of HTML service documentation, by relying on hRESTS for marking service properties and making the descriptions machine-processable. It uses three main types of link relations: 1) **model**, which can be used on any service property to point to appropriate semantic concepts identified by URIs; 2) **lifting** and 3) **lowering**, which associate messages with appropriate transformations (also identified by URIs) between the underlying technical format such as XML and a semantic knowledge representation format such as RDF. Therefore, MicroWSMO, based on hRESTS, enables the semantic annotation of RESTful services in the same way in which SAWSL, based on WSDL, supports the annotation of Web services.

In addition, MicroWSMO can be complemented by the WSMO-Lite service ontology specifying the content of the semantic annotations. Since both RESTful and WSDL-based services can have WSMO-Lite annotations, this provides a basis for integrating the two types of services. Therefore, WSMO-Lite enables unified search over both WSDL-based and RESTful services and tasks such as discovery, composition and mediation can be performed based on WSMO-Lite, completely independently from the underlying Web service technology (WSDL/ SOAP or REST/HTTP).

SA-REST. Another formalism for the semantic description of RESTful services is SA-REST [9], which also applies the grounding principles of SAWSDL but instead of using hRESTS relies on RDFa [27] for marking service properties. Similarly to MicroWSMO, SA-REST enables the annotation of existing HTML service descriptions by defining the following service elements: input, output, operation, lifting, lowering, or fault and linking these to semantic entities. The main differences between the two approaches are not the underlying principles but rather the implementation techniques. In addition, MicroWSMO aims to add semantics as means for Web service automation, while SA-REST is more oriented towards enabling tool support in the context of service discovery and composition by mashup or smashup [35].

Even though, there is some initial research in the area of semantic RESTful services, which can be used as a foundation for further work, none of the here described formalism can be directly adopted for the creation of semantic descriptions, which enable the seamless integration of open services on the Web in the Semantic Web. The existing formats for creating machine-processable descriptions based on available HTML descriptions, in particular hRESTS, do not support the identification of all service elements usually provided within a Web API or Web application documentation.

For instance, most service descriptions include information about the necessary authentication, the response format, and examples. However, currently none of the available formats can represent this information.

Similarly, MicroWSMO and SA-REST, which are used specifically for the semantic annotation, also have some limitations. For example, descriptions provided in ether from very seldom provide enough information for automatic service invocation. Therefore there is the strong need to identify the service elements, which need to be included in the semantic service description, as well as the required semantic metadata.

2.3 Linked Data

Linked Data is used to refer to a set of principles describing the way for publishing and connecting structured data on the Web [25]. Following these principles, data providers can publish and link their data, contributing to the development of a global data space – the **Web of Data**. Linked Data suggest that the same principles that enabled the Web of documents to become so popular should be applies on data as well. In this way, data coming from diverse domains, which has previously been hidden away in databases or repositories, can be made available on the Web and interlinked with existing documents.

The Web of Data provides the basis for new types of applications, which can operate on it. For example, data can be explored with the help of a browser such as Tabulator [36], by following links to related data sources. In addition, it can be used for building Web applications [37] or it can be searched and queried for a particular topic of interest by simultaneously covering a number of linked sources.

By applying the underlying principles of the Web, Linked Data uses the Web to create links between data coming from different sources. The resulting published data is machine-readable, with explicitly defined meaning, and is linked to/from other external data sets. In oder to conform to these requirements, there are a set of rules[4] defined for publishing data on the Web in a way that it becomes part of a single global data space. These rules are summarized below:

1. All items should be identified by using URIs.
2. Use HTTP URIs to enable looking up the item identified by the URI.
3. When looking up an URI, it leads to more data.
4. Links to other URIs should be included, which enables the discovery of more data.

These rules are commonly referred to as the **"Linked Data Principles"** and provide the fundamental guidelines for using the Web infrastructure for publishing and connecting data. Therefore, these principles should be carefully evaluated, in the context of open services on the Web, since services can operate on and manipulate the data, in return providing output information, which can be linked to the Web of data as well.

The main technologies behind Linked Data includes HyperText Transfer Protocol (HTTP), Uniforms Resource Identifiers (URIs) and Resource Description Framework (RDF). HTTP and URI are essential technologies for the Web, where URI provides

[4] `http://www.w3.org/DesignIssues/LinkedData.html`

means for identifying documents and other entities and HTTP enables the retrieving of resources or entity descriptions based on the provided URI. These are supplemented by RDF, which provides a graph-based data model for structuring and linking data that describes things in the world.

By using these three underlying technologies and by following the linked data principles, data providers can add their data to the global data space, where it can be discovered and used by applications and users alike. There are three basis steps for **publishing data sets as Linked Data**:

1. Entities described by the data should be assigned URIs and these URIs, when retrieved over HTTP, should result in the corresponding RDF representations.
2. RDF links to other data sources on the Web should be defined.
3. The published data should be supplemented by metadata.

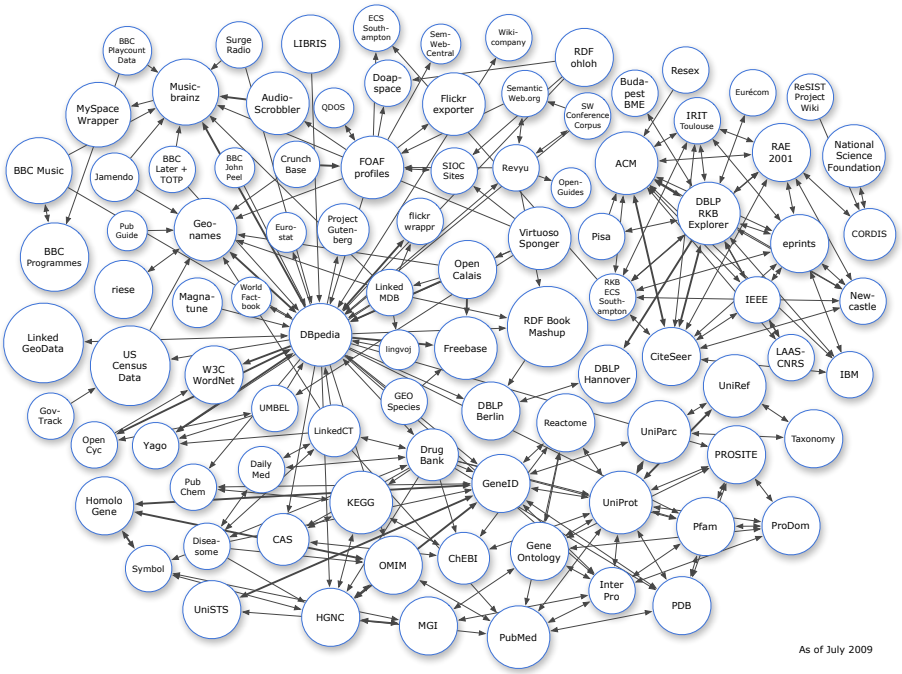

Figure 2.1: Linked Open Data Cloud

More detailed guidelines about how to publish linked data on the Web are available in [38]. By following these steps for adding data sets and adopting the Linked Data principles, the W3C's **Linking Open Data**[5] (LOD) project was developed. Figure 2.1 [6] illustrates the data sets participating in the LOD. This is currently the most advanced example of contributing to the Web of Data. The project aims to identify existing open

[5] `http://esw.w3.org/topic/SweoIG/TaskForces/CommunityProjects/LinkingOpenData`

[6] Source: by Richard Cyganiak at `http://richard.cyganiak.de/2007/10/lod/`

licensed data sets, to convert these to RDF and publish them on the Web. In this way, by starting with only a few initially added data sets, more and more data sets can be linked, leading to the bootstrapping of the Web of Data. Due to the opened nature of the project anyone can publish a data set by following the Linked Data principles. Currently, the cloud includes diverse information, including data about people, science publication, books, geographical locations, music, films, and many more.

The resulting Web of data can be used as basis for developing a wide variety of new applications, which can dynamically analyze and interpret the data, browse from one data source to another by following the links, or collect and query data related to particular topic of interest. More importantly, in the context of open service on the Web, linked data can be used as background information, used as support for suggesting service annotations, but also it already provides a set of ontologies, which can directly be used for semantically describing services. In addition, the resulting semantic descriptions of open services on the Web can be published by following the linked data principles and be directly integrated in the Web of data.

2.4 Supporting the Creation of Semantic Descriptions of Open Services on the Web

This section describes research work related to supporting the creation of semantic descriptions of open services on the Web (RQ2). The formalisms, which make HTML service descriptions machine-processable and enable the adding of semantic information, provide the means for creating semantic descriptions of open services on the Web. However, without supporting tools or guidelines, the developer would have to modify and enhance the descriptions manually by using a simple text/HTML editor. In addition, the complete annotation process would have to be manually completed, if the developer cannot rely on tools, which will help him/her to find suitable domain ontologies or to reuse annotations of previously semantically described services. In order to address these challenges and to facilitate the easier adoption of semantic description of open services by supporting users in their creation, we consider research related to 1) acquiring semantic Web service descriptions, to 2) annotation recommendation and finally, to 3) tools for visualizing and editing semantic data.

Acquisition of semantic Web service descriptions. There are a number of developed approaches aiming to support the acquisition of semantic Web service descriptions. Paolucci et al [22] offers a simple solution by addressing the problem of creating semantic metadata (in the form of OWL-S) from WSDL. The result is a syntactical transformation of the WSDL description, which contains no semantic information because WSDL contains no semantic information, in the first place. A challenge that is not addressed by this approach, is the mapping of WSDL-based service properties to classes or instance in an ontology. In addition, the difficulty of determining a suitable domain ontology for service annotation is also not discussed.

Sabou et al [23] tackles precisely this problem of creating suitable domain ontologies. She uses shallow natural language processing techniques to assist the user in creating an ontology based on software APIs. The approaches are presented based

on two ontology building processes in the context of two concrete research projects, revealing some of the major aspects necessary for building a Web service ontology.

Focusing on the Web service annotation task, Patil et al [39] applies graph similarity techniques to select a relevant domain ontology for a given WSDL file from a collection of ontologies. The presented approach involves also work on matching XML schemas to ontologies in the Web Services domain. They use a combination of lexical and structural similarity measures, based on the assumption that the user's goal is not to annotate similar services with one common ontology, but rather to use different ontologies. Therefore, they also address the problem of choosing the right domain ontology withing a set of ontologies.

Finally, Hess and Kushmerick [24] employ Naive Bayes and SVM machine learning methods to classify WSDL files in manually defined task hierarchies. In addition, there are a number of approaches, which use existing Web service repositories, in particular UDDI, and enhance them with complex semantic markup [40], [41].

Currently, there are no approaches developed for RESTful services, which lack structured descriptions such as WSDL files. Therefore the here described approaches, which rely on the available syntactical information cannot be directly applied on open services on the Web and some modifications are necessary in order to enable the providing of semantic annotations on top of existing HTML descriptions.

Annotation Recommendation. In general, the main aim of recommender systems is to assist users in making decisions among different alternatives, based on user preferences. Or more precisely, recommender systems support users by identifying interesting products and services in situations where the number and complexity of offers outstrips the user's capability to survey them and reach a decision [42]. In the case of semantic service descriptions, recommender systems can be effectively used to recommend suitable service annotations.

There are a number of recommender systems developed for the purpose of aiding users in providing semantic information and these systems are particularly relevant to our work on creating semantic descriptions of open services on the Web. TagAssist [43] is a system, which provides tag suggestions for new blog posts by utilizing existing tagged posts. Similarly, [44] uses collective intelligence extracted from collaborative tagging, in addition to word semantics, in order to learn the best set of tags to use for new blog entries. The approach presented in [45] is an adaptation of user-based collaborative recommendation and graph-based recommender, which suggests tags for different resources. All of these approaches use common recommender techniques, profiting from semantic information, for the purpose of making the process of adding semantic information on the Web easier, by suggesting tags and annotations.

Particularly in the context of semantic service descriptions, such annotation recommender systems can prove to be useful. There are different formats and methods for creating semantic service descriptions, however, there are some main tasks, which very commonly need to be addressed. These include the classification of the service based on its functionality, the assigning of a domain ontology and the annotation of the individual service properties. ASSAM [46], a tool, that assists a user in creating semantic metadata for Web Services, offers a solution for some of these issues. It automatically suggests semantic metadata based on two main machine learning algorithms. The first one, aims to semantically classify Web service, while the second one,

facilitates recommendation of semantic annotations by aggregating data returned by multiple semantically related Web services. A similar approach is taken in METEOR-S [39], a framework for semi-automatically marking up Web service descriptions with ontologies. The framework focuses on determining a relevant domain ontology and annotating WSDL files, based on it. Classification is also applied by using domain ontologies to categorize Web services into domains. Both systems show very promising experimental results.

METEOR-S addresses one of the main problems of creating semantic descriptions of open services on the Web, namely, the determining of suitable domain ontologies. Similarly to [23], some research on the semantic description of Web services focuses on learning domain ontologies for service annotation. Such approaches are effective because learnt ontologies are more service-specific and, therefore, more suitable for the annotation than general purpose ontologies. However, this means that each service or group of services has its own ontology and as a result, service discovery has to devote processing tasks related to ontology mapping and concept matching, which would not be the case if a set of common ontologies were used for the annotation. In addition, instead of reusing annotations from already semantically described services, service-based ontology learning adds additional complexity to the service annotation process.

Recently, with the growing use and popularity of the Semantic Web, there is an abundance of semantic data available and the Semantic Web represents a very valuable source of ontologies and background knowledge. This provides the basis for enchanting existing techniques for annotation recommendation to use this semantic data. In addition, there are tools such as Watson [47] and Sindice [48], which assist users in searching for semantic tags and ontologies. For the semantic annotation of open services on the Web this means that the Semantic Web provides the basis for the development of less complex approaches, based on reusing existing data.

In contras to most recommendation approaches, Billsus and Pazzani [49] address the problem in a different way. They suggest that the recommendation of alternatives based on user ratings can be solved by transforming the recommendation problem into a classification problem. Automatically assigning a Web service to a particular group of services with similar functionality or similar application domain, already provides semantic information, which directly contributers to the improved automation of service discovery and composition tasks. Moreover, in some approaches, the classification task can be equivalent to determining a suitable domain ontology for the annotation of the service [50], in other approaches it can at least simplify the process of determining a domain ontology and finally, sometimes [51] annotation recommendations are computed as part of the classification process. Therefore, most approaches for acquiring semantic Web service annotations rely on service classification. Commonly used classification approaches are the k-nearest neighbor, naive Bayes and Rocchio [51], while naive Bayes is the most commonly used one. Support vector machines based on document frequency values are also used in Web service classification and annotation [52]. A general survey of text-based classification is given by [53].

In summary, even though, the challenges of adding semantic information to Web services have already been addressed in a number of approaches, there is a lot to be done in the area of semantic descriptions of open services on the Web. In contrast to WSDL-based services, Web API descriptions are text/HTML based and, therefore,

require information extraction preprocessing before the recommendation analysis and the actual annotation can be performed. Still, some of the Web service classification or domain ontology assigning approaches can be adapted and used for the semantic description of open services on the Web.

Ontology Visualization and Annotation Tools. The process of semantic annotation can be supported by ontology visualization tools, which enable users to view and explore an ontology and to decide whether or not it can be used for the annotation of a particular service. A good overview of current ontology visualization tools is provided in [54], including common tools with explorer-based view of the ontology such as Protégé [55], OntoEdit [56], Kaon [57] and OntoRama [58].

Ontology visualization already provides some user support, however, functionalities such as adding or viewing annotations are even more relevant in the context of supporting the creation of semantic descriptions of open services on the Web. There are tools, which enable the annotation of Web content such as OntoMat-Annotizer and SMORE. OntoMat-Annotizer is a web page annotation tool [59], which supports users in creating and maintaining ontology-based OWL-markups. A similar tool is SMORE, which enables users to markup HTML documents in OWL by using Web ontologies. It includes some options for ontology editing and provides users with the possibility to create a new ontology based on terms from web documents. The results of both tools are a web page with attached markup.

PowerMagpie [60], [61] is a tool that uses semantic information in a quite different way. PowerMagpie was initially implemented as a tool using ontologies to markup web documents. Its current version has evolved into a semantically-enhanced web browser. While browsing, PowerMagpie identifies and provides any available semantic markup, which can be found on the Semantic Web. It is not restricted to one predefined ontology, but rather accesses the whole of the Semantic Web through Watson and selects and presents to the user relevant information. Tools such as PowerMagpie, lay the foundation for the development of a tool for the semi-automatic acquisition of semantic descriptions of open services on the Web.

In addition to web-content annotation tools, there are some desktop-based tools which are particularly useful for the creation of semantic Web service descriptions. These include WSMO Studio [62] and the Web Service Modeling Toolkit [63] (WSMT). WSMO Studio is developed for modeling and creating semantic Web services and semantic business processes. It provides a modeling and visualization framework for assisting users working in the WSMO domain with tasks related to the semantic Web service annotation. Similarly, the Web Service Modeling Toolkit (WSMT) comprises a number of tools for semantic Web services, based on WSMO, WSML and WSMX, including a Web Service Modeling Language visualizer, a WSML reasoner and a WSMX data mediation mapping tool.

Currently, the only tool particularly targeted at the semi-automatic creation of semantic Web service descriptions is ASSAM [46]. It includes a WSDL tree-based viewer, a category browser, as well as a data types display and a service browser. These components are common also for other annotation tools, however, the key feature of the WSDL annotator is the ability to suggest which ontological concept to use for annotating service elements. The result is a higher level of automation in comparison to other existing tools for creating semantic Web service descriptions.

Up to date, there are no tools developed for supporting the semantic annotation of open services on the Web. However, existing tools for the annotation of WSDL-based services can be analyzed and used for determining requirements and guidelines for the necessary computational and visualization components. Approaches for ontology visualization and browsing can be simplified and adopted from popular tools such as Protégé. Since the majority of the existing Web APIs are described in HTML Web pages, a necessary feature for a tool developed to support the creation of semantic descriptions of open services on the Web would be the to be able to directly manipulate the HTML content displayed in a Web browser, similarly to PowerMagpie.

2.5 Defining the Gap

Current research related to the creation of semantic descriptions of open services on the Web provides a foundation, which can be extended by approaches and tools particularly targeted at handling the challenges faced by open services. Still, there are a number of gaps, which need to be addressed in order to enabled the more automated use of open services on the Web and their seamless integration in the Semantic Web.

The following topics, which need to be targeted by future research, have been identified based on the analysis of the state of the art:
– There are no guidelines or formalisms, which specify the information, which needs to be provided in a HTML-based Web API description.
– There are no rules or guidelines for the form and structure of HTML-based Web API descriptions.
– Currently, there is no way of differentiating between Web API descriptions and common Web pages, which plays an important role for achieving higher automation of the service discovery task.
– Since the majority of the open services on the Web are described in plain HTML within normal Web pages, there is the need of a formalism for describing open services in a machine-processable, so that:

1. The formalism can be applied on existing Web API descriptions;
2. The resulting description can easily support adding of semantic annotations;
3. The resulting description preserves the lightweight characteristic of open services on the Web;
4. All service elements contained in the unstructured documentation can be reflected in the resulting formal description.

– There is no common repository (such as UDDI), where semantic descriptions of open services on the Web can be published, annotated and searched.
– It is necessary to define the minimum but sufficient semantic information required for supporting automation of open service discovery, composition and invocation.
– There is a need for a formalism for the semantic description of open services on the Web (this can incorporate the formalism for machine-procesable descriptions), which provides sufficient metadata for supporting the automation of service tasks.
– Currently there are no tools that support the creation and use of semantic open services on the Web.
– All tasks related to semantic open services on the Web have to be completed manu-

ally by the user.
– Developers need to be supported by tools and approaches, which reduce the manual effort required for creating semantic descriptions of open services on the Web.

Some of the here identified topics of future research are addressed in our work on the creation of semantic descriptions of open services on the Web. The following chapter described the followed research approach, initial work and results, as well as a roadmap of the planned future work.

Chapter 3

Open Services on the Web

The goal of the here described PhD work is to explore the possibilities of combining Semantic Web technologies and fundamental Web principles, including URIs and HTTP, and to apply these on open services on the Web, in order to contribute to a Semantic Web, which is not only an extension of the current Web with more semantic descriptions of data but is rather more dynamic and seamlessly integrates services as sources of that data, which can be automatically discovered, composed and executed by the computer on behalf of its user. In particular, this goal will be achieved by supporting the **creation of semantic descriptions of open services on the Web**, which are the basis for the automation of service discovery, compositions and invocation, by providing an access layer, which abstracts from the actual technical implementation and enables data manipulation based on semantic entities. This will enable a Semantic Web, where users can retrieve information without differentiating weather its source is a website, a Web API or even a mashup.

3.1 Approach and Research Contribution

The here proposed approach is to create a semantic layer on top of existing Web API descriptions, which enables the automation of service discovery, composition and invocation. The approach includes three main parts. First, a **formalism for the semantic description of open services on the Web** will be developed. This formalism is based on a service model, which is defined by analyzing common open service descriptions and annotation requirements. Second, a **methodology for the creation of semantic descriptions of open services on the Web** will be developed, which enables applying the designed formalism on existing HTML documentation. Third, **developer support** will be provided, including tools, which hide formalism complexities behind a user interface, and automated techniques, which will reduce the manual effort required for service annotation by suggesting suitable service metadata. In this way, the formal solutions can be directly put into practice by using the implemented tools.

The envisioned research work is divided into three main phases.

Phase 0: Analysis of current descriptions of open services on the web
This phase focuses on completing preliminary work necessary for the creation of semantic descriptions of open services on the Web. In particular, this includes:
– Analysis of current state of the art in areas directly related to open services on the Web.
– Completion of a literature review, based on this analysis.
– Analysis of common Web API descriptions and identification of description types and main description characteristics.
– Identifying a set of common service elements, contained in current Web API descriptions.
– Collection of requirements for the annotation of open services on the Web.
– Identifying use cases for a tool supporting developers in the creation of semantic descriptions of open services on the Web.
– Collection of tool requirements.

Phase 1: Manual creation of semantic descriptions of open services on the Web
This phase focuses on: 1) developing an initial methodology for the creation of semantic descriptions of open services on the Web; 2) drafting a first version of the description formalism; and 3) implementing supporting tool prototypes. In detail this includes:
– Defining an initial set of service elements, which need to be contained in the service description.
– Extending these with information required for service discovery, composition and invocation.
– Definition of a open service model based on the identified common service elements.
– Based on the service model, define a formalism for the semantic description of open services on the Web.
These first four tasks will not be completed in a sequential manner but rather in an iterative way, where experience gained through initial implementation work and methodology development is reflected though refinement in the service model and the developed formalism.
– Initial description of the process of creating semantic descriptions of open services on the Web.
– Initial draft of a methodology and guidelines for creating semantic descriptions of open services on the Web.
Similarly, these two tasks will also be performed in an iterative manner and will lead to the refinement of the results.
– Identification of sources of domain ontologies and possible approaches for searching for domain ontologies.
This step can be replaced by already existing ontology learning approaches, if the initial results show that reusing already available ontologies on the Web is not feasible. A hybrid solution is also possible.
– Initial design and architecture for supporting tools.
– Initial components implementation.
– Initial set of user guidelines for creating semantic descriptions of open services on the Web.
It is important to point out that some of these tasks can be performed in parallel, while

others are tightly related and result in valuable feedback, which can change the initial design. Therefore, it is necessary to cross-analyze the results and identify possible gaps, incompleteness or even conflicts.

Phase 2: Automating the creation of semantic descriptions of open services on the Web

This phase relies on the results of Phase 1, which provides complete support for the manual creation of semantic descriptions of open services on the Web. It identifies tasks, which have the potential to be automated, and develops solutions, which require less manual effort and user interaction. In particular, this includes:

– Parsing and processing of service descriptions in order to automatically identify service properties. (Developing of heuristics and language processing techniques especially targeted at identifying service properties.)

The service properties identified by this task will be visually highlighted in the service description, so that the user can easily recognize possible mistakes and validate the results. These service properties will serve as the basis for attaching semantic information.

The task of recommending suitable annotations for the service as a whole and for the separate service properties is a very challenging one. Therefore, an initial solution will be developed by reducing the complexity of the problem. This is achieved by focusing on groups of services belonging to a common functionality domain (for example, weather services or geographic services). In this way, a restricted number of domain ontologies can be used for annotating all services. It is important that the chosen domains contain enough service examples, so that the developed approach can be extended and generalized.

– Determining a set of fixed domains (3-4 domains), with a sufficient number of services. This task will be completed by manually searching for Web API descriptions in service repositories, such as programmableweb, or directly searching the Web by using a set of characteristic key words such as "Web API, REST web service, REST web service documentation, etc.". The goal is to identify three or four domains such as messaging, mapping, financial, photos and travel, each containing at least 40-50 services, which can be used as the basis for developing a recommendation approach and performing experiments. If initial results show that these numbers are insufficient, more services will be collected.

– Developing an annotation recommendation approach and implementation for these particular fixed domains.

– Refining the domain ontologies search methods defined in Phase 1.

– Developing an approach for service classification, which will automatically assign services to a particular domain.

– Extrapolating the domain-specific annotation recommendation approach to a general annotation recommendation approach. The service classification approach, in combination with the annotation recommendation approach, will provide support for automating the process of creating semantic descriptions of open services on the Web.

– Extending the tool support developed in Phase 1.

Similarly to Phase 1, these tasks will also be performed in an iterative manner, using results to improve already developed solutions and to refine the overall approach.

Initial work, following the here described approach, has already been completed and is described in the next section.

3.2 Completed Work and Initial Results

Following the approach described in the previous section and the tasks list for the different phases, in summary, Phase 0 is almost complete, while Phase 1 in mostly work in progress, and Phase 2 will shortly be intiated. The completed work and initial results are described in detail if the following sections.

As part of the tasks done for Phase 0, a literature review was completed, which was partially summarized in the state of the art section of this report. In addition, a manual analysis of Web API descriptions was performed, identifying: 1. elements which are common for most of the descriptions; 2. typical description structures; 3. common description types; and 4. description forms. This analysis resulted in a preliminary list of **common service elements** found in Web API descriptions and in the definition of three main types of services. These findings were published as part of two workshop papers at OnTheMove, Nov. 2009 [64], and ISWC, Oct. 2009 [65]. There is still some work to be done on determining a minimal service model, based on the semantic information required for supporting the automation of discovery, composition and invocation tasks.

In addition to doing initial work for defining a formalism for the semantic description of open services on the Web, some preparation for the implementation of tools was also done. In particular, **potential use cases** were determined based on the scenarios defined in the SOA4All[1] project and initial **set of requirements** were derived. These results were used for developing and **initial implementation and architecture design** of an overall solution, which is visualized in Figure 3.1. Some of the components such as the Document Analysis component were rather designed as a black-box component because at this stage the particular approach was not clear (to be completed in Phase 2). While other components such as the Visualization component and the Ontologies Access component were realized in actual implementations, shortly after the design was complete.

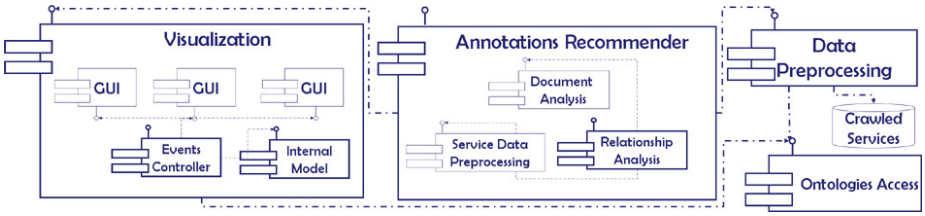

Figure 3.1: Initial Architecture

The work on the description formalism and on the supporting tool implementation was always done almost in parallel, integrating results on semantic description of open services on the Web into the prototype implementations and vice versa. This resulted in

[1] http://www.soa4all.eu/

the development of an **initial approach for the creation of semantic descriptions of open services on the Web** and **two tool implementations**, targeted at supporting users in two different use cases. The developed approach is very lightweight and is based on using the hRESTS microformat (for details see Section 2.1), which enables the creation of machine-processable Web API descriptions based on available HTML documentation. hRESTS is complemented by the MicroWSMO microformat (See Section 2.3), which introduces additional HTML classes, in order to enable the specification of a model reference. This approach was chosen because hRESTS and MicroWSMO are very lightweight and easy to use (as opposed to RDFa and SA-REST). In addition, they enable the creation of semantic descriptions on top of existing HTML documentation, which is an important feature, since the majority of the Web APIs are textually described in Web pages. In this way, hRESTS and MicroWSMO can be used as a basis for identifying gaps in the machine-processable description and lacking semantics, which are necessary for the automation of the discovery, composition and invocation tasks.

In order to be able to verify the applicability of this initial approach for formal description of semantic open services on the Web, two tool implementations of MicroWSMO editors were developed. As a result, the integrated creation of MicroWSMO-based semantic service descriptions is realized. The MicroWSMO editor, as the name suggests, supports users in creating MicroWSMO semantic annotations. We have decided to call it SWEET: Semantic Web sErvices Editing Tool and SWEET currently has a website[2] providing up to date information and a demo site[3], where users can directly test the tools.

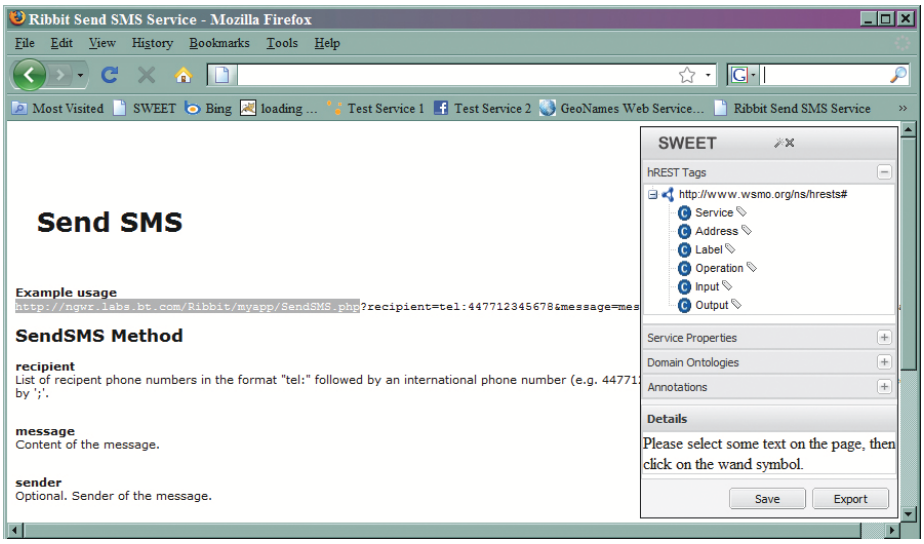

Figure 3.2: SWEET: Browser-based Version

[2] http://sweet.kmi.open.ac.uk/
[3] http://sweetdemo.kmi.open.ac.uk/

The MicroWSMO editor is implemented in two main versions. The first version takes the form of a vertical widget displayed within a web browser. It is very lightweight and can be used to directly annotate RESTful service descriptions visualized in the browser window. The second implementation is integrated in a fully-fledged platform for the creation, discovery, composition, invocation and monitoring of services, which is part of the SOA4All project. Both MicroWSMO editor implementations share common main functionalities:

- Creating machine-processable service descriptions by inserting hRESTS microformat tags in the HTML documentation, in order to mark service properties (service, operation, address, HTTP method, input, output and label).
- Integrated ontology search for linking semantic information to service properties by using Watson.
- Creating semantic service annotations by inserting MicroWSMO model reference tags, pointing to the associated semantic meaning of the service properties.
- Saving of semantically annotated HTML RESTful service description.
- Automatic extraction of RDF MicroWSMO service descriptions, based on the annotated HTML, and saving of the resulting RDF.

Each of the MicroWSMO editor implementations addresses different user needs and supports different use cases. The browser-based lightweight version (figure 3.2) is suitable for users who are browsing for RESTful services and want to directly annotate the currently displayed description. Whenever a user wants to create MicroWSMO annotations of the currently viewed service, he/she can simply click on the bookmarklet of the MicroWSMO editor, initiate it and directly use it.

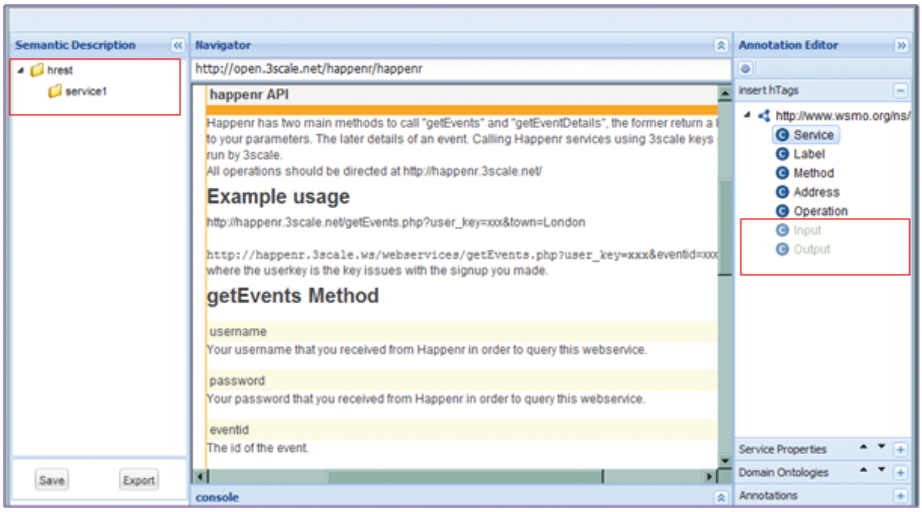

Figure 3.3: SWEET: Platform-integrated Version

On the other hand, the implementation of the MicroWSMO editor, integrated in the fully-fledged platform (Figure 3.3), is suitable in cases where the user performs multiple tasks, in addition to RESTful service annotation, and where a set of RESTful services are already known in the platform, for example through a crawler component. For instance, the user can annotate a multitude of services from the same domain and switch between the MicroWSMO editor and the WSMO-Lite editor, depending on the service type. After the services are annotated, he/she can use them in service compositions or directly execute them. Such more complex use cases are better facilitated by the platform-integrated MicroWSMO editor. Still both implementation can be used to support both use cases, even though one is more suitable than the other.

The challenge of finding appropriate domain ontologies for the service annotation was solved in an integrated way by using Watson [47]. Watson enables the search for ontologies based on keywords, which are matched against semantic entities' labels. In this way, the user can directly select a service property in the RESTful service description and simply by clicking on a button in the MicroWSMO editor, find ontologies that can directly be used for the annotation.

The results of the here described initial approach and implementation were published in the form of two posters at ESWC, May 2009 [66], and STI Offsite, May 2009 [67], one demo at ISWC, Oct. 2009 [68], two workshop papers at OnTheMove, Nov. 2009 [64], and ISWC, Oct. 2009 [65], and two project deliverables for SOA4All in Feb. 2009 [69], and Aug. 2009[70]. Future work in the short term will focus on collecting evaluation results of the initial approach and the two tool implementations. After this, work on the annotations recommendation will be initiated by defining three or four fixed service domains and developing a recommendation approach within these domains. The following section shortly describes the planned progress of the PhD work for the next two years.

3.3 Progress Plan

Table 3.1 summarizes the main activities planned for the second year of this PhD work.

The work will follow the already outlined approach, focusing both on the development of a formalism and a methodology for the semantic description of open services on the Web as well as on implementing the corresponding tool support. In particular, implementation work will be divided into two main parts, initially focusing on developing new tool functionalities and subsequently working on refining the resulting implementation, improving the usability and stability, and reacting to user feedback. In order to support the uptake and increase the impact of the research work, extra care will be devoted to dissemination activities, going beyond the presentation of results at conferences and workshops. Dissemination will be conducted through standardization channels and established community sites or bodies on the Web. These activities will also be continuously pursued during the third and final year.

Evaluation of the current results will be conducted periodically, at least every three months. For this purpose, during the first year, a concrete use case will be designed, which will serve as the basis for testing and measuring the progress and the completed work. In addition, analysis of existing mashups and other application areas of ser-

Table 3.1: Progress Plan for the Second Year

	M 1	M 2	M 3	M 4	M 5	M 6	M 7	M 8	M 9	M 10	M 11	M 12
Internal Seminar I	X											
Internal Seminar II									X			
Implementation Work	X	X	X	X	X							
Impl. Refinement						X	X	X	X	X	X	X
Dissemination Activities							X	X	X	X	X	X
Evaluation			X			X			X			X
Conference Submission	X	X	X	X	X	X	X		X		X	
Report Research School I	X											
Report Research School II												X
Thesis Overview								X	X			
2nd Year Report										X		
2nd Year Viva											X	

vices will be used to determine the adequacy and automation support provided by the developed semantic description for service on the Web.

Table 3.2: Progress Plan for the Third Year

	M 1	M 2	M 3	M 4	M 5	M 6	M 7	M 8	M 9	M 10	M 11	M 12
Internal Seminar III									X			
Impl. Refinement	X	X	X	X	X	X						
Impl. Maintenance							X	X	X	X	X	X
Dissemination Activities	X	X	X	X	X	X	X	X	X	X	X	X
Evaluation		X	X			X	X					
Conference Submission	X	X	X	X	X	X	X	X	X	X	X	
Report Research School III												X
Thesis Writing				X	X	X	X	X	X	X	X	
Thesis Submission												X
3nd Year Viva											X	

The third year will not focus so much on producing new results and implemented components but rather on making the achieved progress visible. This will be done through the intensive participation in conferences and workshops. The two main envisioned conferences are ESWC (submission deadline in December) and ISWC (submission deadline in June). Naturally, one of the main tasks, which has to be completed during this final year, is the documentation of the made contributions and results in the form of a thesis.

3.4 Conclusion

Currently, Web applications and APIs are becoming more and more popular, however, the general adoption of open services on the Web is hindered by the fact that they cannot be found, interpreted and invoked without the extensive user involvement and a multitude of manual tasks. This challenges can be addressed through the creation of machine-processable descriptions, which serve as the basis for automatically finding services, through crawlers and search engines, and processing them. Moreover, extended with semantic annotations, open services on the Web can even be discovered, composed and invoked automatically, following the principles of the SWS research. Therefore the goal of this PhD work is to explore the possibilities of combining Semantic Web technologies and fundamental Web principles, in order to contribute to a Semantic Web, which seamlessly integrates services.

This report provides an overview of the approach for the creation of semantic descriptions of open services on the Web. It also describes some initial results, including an integrated lightweight approach for formally describing RESTful services based on MicroWSMO. This initial approach uses two microformats: the hRESTS microformat that enables the tagging of key service properties and therefore supports the creation of machine-processable service descriptions; and the MicroWSMO microformat that enables the linking of semantic information to service properties. The approach is facilitated by two implementations of a MicroWSMO editor, which effectively supports users in creating semantic descriptions of open services on the Web, by providing functionalities for both the creation of machine-processable descriptions and the addition of semantic annotations based on hRESTS and MicroWSMO.

Future work will focus on developing a formalism for the semantic description of open services on the Web, based on the results of the initial MicroWSMO approach, which will be used to identify information gaps and missing semantics. In addition, work will be devoted to developing an initial annotation recommendation approach, in order to reduce the number of manual tasks that the user has to complete. Future work will also include the development of supplementary functionalities of the MicroWSMO editors, which will provide additional user support. Also, some work will be devoted to the automatic recognition of service properties such as operations and input parameters, so that the user only has to verify the pre-marked service properties.

Bibliography

[1] T. Berners-Lee, J. Hendler, and O. Lassila. *The semantic web*. 2001.

[2] T. Berners-Lee. *Weaving the Web*. Harpur, San Francisco, 1999.

[3] L. Richardson and S. Ruby. *RESTful Web Services*. OReilly, May 2007.

[4] R. T. Fielding. *Architectural styles and the design of network-based software architectures*. PhD thesis, University of California, 2000.

[5] W3C WS Description Working Group. Web service description language (WSDL) Version 2.0, W3C proposed recommendation. http://www.w3.org/TR/wsdl20-primer, May 2007.

[6] M. J. Hadley. Web application description language (WADL). Technical report, Sun Microsystems. https://wadl.dev.java.net, November 2006.

[7] Jacek Kopecky, Karthik Gomadam, and Tomas Vitvar. hRESTS: an HTML Microformat for Describing RESTful Web Services. *In Proceedings of the 2008 IEEE/WIC/ACM International Conference on Web Intelligence (WI-08)*, 2008.

[8] J. Kopecky and T. Vitvar. MicroWSMO. CMS Working Draft. http://www.wsmo.org/TR/d38/v0.1/20080219/, February 2008.

[9] A. P. Sheth, K. Gomadam, and J. Lathem. SA-REST: Semantically interoperable and easier-to-use services and mashups. *IEEE Internet Computing*, 11(6):91-94, 2007.

[10] Jonathan Douglas Lathem, John Miller, and Lakshmish Ramaswamy. SA-REST: Bring the power of semantics to REST-based web services. Master's thesis, University of Georgia, August 2007.

[11] M. Papazoglou, P. Traverso, S. Dustdar, and F. Leymann. Service-Oriented Computing: A research roadmap. *International Journal of Cooperative Information Systems*, 17(2), 223-255, 2008.

[12] G. Alonso, F. Casati, H. Kuno, and V. Machiraju. *Web Services*. Springer, 2003.

[13] G. Alonso, F. Casati, H. Kuno, and V. Machira ju. Web services: Concepts, architectures and applications. *Data-Centric Systems and Applications*, 2004.

[14] W3C WS Description Working Group. Web service description language (WSDL) Version 1.1, W3C Note. http://www.w3.org/TR/wsdl, March 2001.

[15] W3C WS XML Protocol Working Group. Simple Object Access Protocol (SOAP) Version 1.2, April 2007.

[16] S. McIlraith, T. Son, and H. Zeng. Semantic Web Services. *IEEE Intelligent Systems, Special Issue on the Semantic Web*, 16(2):46–53, 2001.

[17] D. Fensel, H. Lausen, A. Polleres, J. de Bruijn, M. Stollberg, D. Roman, and J. Domigue. *Enabling Semantic Web Services. The Web Service Modeling Ontology*. Springer, Berlin, Heidelberg, 2006.

[18] Alex Rodriguez. RESTful Web services: The basics. *IBM developerWorks*, November 2008.

[19] B. Worthen. Mashups sew data together: Mashup tools can cut costs, time for linking information sources. *The Wallstreet Journal*, 31 July, 2007.

[20] T. Berners-Lee, R. Fielding, and L. Masinter. RFC 3986, Uniform Resource Identifier (URI): Generic syntax. http://tools.ietf.org/html/rfc3986, 2005.

[21] Steve Vinoski. Putting the "web" into web services: Interaction models, part 2. *IEEE Internet Computing*, 6(4):90–92, 2002.

[22] M. Paolucci, N. Srinivasan, K. Sycara, and T. Nishimura. Towards a semantic choreography of web services: From WSDL to DAML-S. In *First International Conference on Web Services (ICWS'03)*, June 2003.

[23] M. Sabou, C. Wroe, C. Goble, and H. Stuckenschmidt. Learning domain ontologies for semantic web service descriptions. *Journal of Web Semantics*, 3(4), 2005.

[24] A. Hess and N. Kushmerick. Machine learning for annotating semantic web services. *In AAAI Spring Symposium on Semantic Web Services*, March 2004.

[25] Christian Bizer, Tom Heath, and Tim Berners-Lee. Linked data - the story so far. *International Journal on Semantic Web and Information Systems (IJSWIS)*, 2009.

[26] R. Khare and T. Celik. Microformats: a pragmatic path to the semantic web (poster). In *Proceedings of the 15th international conference on World Wide Web*, 2006.

[27] RDFa in XHTML: Syntax and Processing. Recommendation W3C. http://www.w3.org/TR/rdfa-syntax/, October 2008.

[28] D. Martin. OWL-S: Semantic Markup for Web Services. W3C member submission, 22 November 2004. http://www.w3.org/Submission/OWL-S/, 2004.

[29] R. Lara, D. Roman, A. Polleres, and D. Fensel. A conceptual comparison of WSMO and OWL-S. In *Proceedings of the European Conference on Web Services*, 2004.

[30] D. Roman, U. Keller, H. Lausen, J. Bruijn, R. Lara, M. Stollberg, A. Polleres, C. Feier, C. Bussler, and D. Fensel. Web Service Modeling Ontology. *Applied Ontology*, pages 1(1): 77 – 106, 2005.

[31] J. de Bruijn, H. Lausen, R. Krummenacher, A. Polleres, L. Predoiu, M. Kifer, and D. Fensel. The Web Service Modeling Language (WSML). http://www.wsmo.org/TR/d16/d16.1/v0.2/, 2005.

[32] J. Domingue, L. Cabral, S. Galizia, V. Tanasescu, A. Gugliotta, B. Norton, and C. Pedrinaci. IRS-III: A Broker-based Approach to Semantic Web Services. *Journal of Web Semantics*, 2008.

[33] J. Farrell and H. Lausen. Semantic Annotations for WSDL and XML Schema (SAWSDL). W3C recommendation. http://www.w3.org/TR/2007/REC-sawsdl-20070828/, August 2007.

[34] T. Vitvar, J. Kopecky, J. Viskova, and D. Fensel. WSMOLite annotations for web services. *5th European Semantic Web Conference, ESWC 2008*, In The Semantic Web: Research and Applications:Springer, 2008.

[35] A. Sheth, K. Verma, and K. Gomadam. Semantics to energize the full services spectrum. *Comm. ACM*, pages vol. 49, no. 7, pp.55–61, 2006.

[36] Tim Berners-lee, Yuhsin Chen, Lydia Chilton, Dan Connolly, Ruth Dhanaraj, James Hollenbach, Adam Lerer, and David Sheets. Tabulator: Exploring and analyzing linked data on the semantic web. In *In Proceedings of the 3rd International Semantic Web User Interaction Workshop*, 2006.

[37] Michael Hausenblas. Exploiting linked data to build web applications. *IEEE Internet Computing*, 13(4):68–73, 2009.

[38] Chris Bizer, Richard Cyganiak, and Tom Heath. How to publish linked data on the web. 2007.

[39] Abhijit Patil, Swapna Oundhakar, Amit Sheth, and Kunal Verma. METEOR-S web service annotation framework. pages 553–562. ACM Press, 2004.

[40] R. Akkiraju, R. Goodwin, P. Doshi, and S. Roeder. A method for semantically enhancing the service discovery capabilities of UDDI. *IIWeb*, pages 87–92, 2003.

[41] J. Luo, B. Montrose, and M. Kang. An approach for semantic query processing with UDDI. *OTM Workshops*, pages 89–98, 2005.

[42] Loren Terveen and Will Hill. Beyond recommender systems: Helping people help each other. In *HCI in the New Millennium*, pages 487–509. Addison-Wesley, 2001.

[43] Sanjay C. Sood and Kristian J. Hammond. TagAssist: Automatic tag suggestion for blog posts. In *In International Conference on Weblogs and Social*, 2007.

[44] Sigma On Kee Lee and Andy Hon Wai Chun. Automatic tag recommendation for the web 2.0 blogosphere using collaborative tagging and hybrid ANN semantic structures. In *ACOS'07: Proceedings of the 6th Conference on WSEAS International Conference on Applied Computer Science*, pages 88–93, 2007.

[45] Robert Jaeschke, Ro Marinho, Andreas Hotho, Lars Schmidt-Thieme, and Gerd Stumme. Tag recommendations in folksonomies. In *PKDD*, pages 506–514. Springer, 2007.

[46] A. Hess, E. Johnston, and N. Kushmerick. ASSAM: A tool for semi-automatically annotating semantic web services. In *Proceedings of the 3rd International Semantic Web Conference (ISWC)*, 2004.

[47] M. d'Aquin, M. Sabou, E. Motta, S. Angeletou, L. Gridinoc, V. Lopez, and F. Zablith. What can be done with the Semantic Web? an overview of Watson-based applications. In *5th Workshop on Semantic Web Applications and Perspectives*, 2008.

[48] Giovanni Tummarello, Renaud Delbru, and Eyal Oren. Sindice.com: Weaving the open linked data. In *Proceedings of the International Semantic Web Conference (ISWC)*, 2007.

[49] Michael J. Pazzani and Daniel Billsus. Content-based recommendation systems. In *The Adaptive Web: Methods and Strategies of Web Personalization. Volume 4321 of Lecture Notes in Computer Science*, pages 325–341. Springer-Verlag, 2007.

[50] Andreas Hess and Nicholas Kushmerick. Automatically attaching semantic metadata to web services. In *Proceedings of IIWeb*, pages 111–116, 2003.

[51] Nicole Oldham, Christopher Thomas, Amit Sheth, and Kunal Verma. METEOR-S web service annotation framework with machine learning classification. In *Proc. of the 1 st Int. Workshop on Semantic Web Services and Web Process Composition (SWSWPC)*, 2004.

[52] Marcello Bruno, Gerardo Canfora, Massimiliano Di Penta, and Rita Scognamiglio. An approach to support web service classification and annotation. In *Proc. of IEEE International Conference on e-Technology, e-Commerce and e-Service*, pages 138–143. IEEE Press, 2005.

[53] Yiming Yang and Jan O. Pedersen. A comparative study on feature selection in text categorization. In *Proceedings of the Fourteenth International Conference on Machine Learning table of contents*, 1997.

[54] Akrivi Katifori, Constantin Halatsis, George Lepouras, Costas Vassilakis, and Eugenia Giannopoulou. Ontology visualization methods—a survey. *ACM Comput. Surv.*, 39(4):10, 2007.

[55] N.F. Noy, R.W. Fergerson, and M.A. Musen. The knowledge model of Protégé-2000: Combining interoperability and flexibility. *Lecture Notes in Computer Science*, 1937:69–82, 2000.

[56] York Sure, J"urgen Angele, and Steffen Staab. OntoEdit: Guiding ontology development by methodology and inferencing. In *CoopIS/DOA/ODBASE*, pages 1205–1222, 2002.

[57] KAON. http://kaon.semanticweb.org/.

[58] Peter Werner Eklund, Nataliya Roberts, and Steve P. Green. OntoRama: Browsing an RDF ontology using a hyperbolic-like browser. In *The First International Symposium on CyberWorlds (CW2002)*, Theory and Practices, pages 405–411. IEEE press, 2002.

[59] Siegfried Handschuh and Steffen Staab. Authoring and annotation of web pages in CREAM. In *WWW '02: Proceedings of the 11th international conference on World Wide Web*, pages 462–473, New York, NY, USA, 2002. ACM.

[60] M. Dzbor, E. Motta, and J. Domingue. Magpie: Experiences in supporting Semantic Web browsing. *Web Semantics: Science, Services and Agents on the WWW*, 2007.

[61] Mathieu d'Aquin, Enrico Motta, Martin Dzbor, Laurian Gridinoc, Tom Heath, and Marta Sabou. Collaborative semantic authoring. *IEEE Intelligent Systems*, 23(3):80–83, 2008.

[62] M. Dimitrov, A. Simov, M. Konstantinov, L. Cekov, and Momtchev V. WSMO Studio - a semantic web services modelling environment for WSMO (system description). In *Proceedings of the 4th European Semantic Web Conference (ESWC). Number 4519 in LNCS*, 2007.

[63] Mick Kerrigan, Adrian Mocan, Martin Tanler, and Dieter Fensel. The Web Service Modeling Toolkit - An integrated development environment for semantic web services (system description). In *Proc. of the 4th European Semantic Web Conference (ESWC)*, pages 789–798, 2007.

[64] M. Maleshkova, J. Kopecky, and C. Pedrinaci. Adapting SAWSDL for semantic annotations of RESTful services. In *Beyond SAWSDL at OnTheMove Federated Conferences and Workshops*, 2009.

[65] M. Maleshkova, C. Pedrinaci, and J. Domingue. Supporting the creation of semantic RESTful service descriptions. In *Service Matchmaking and Resource Retrieval in the Semantic Web (SMR2) at 8th International Semantic Web Conference*, 2009.

[66] M. Maleshkova, L. Gridinoc, C. Pedrinaci, and J Domingue. Supporting the semi-automatic acquisition of semantic RESTful service descriptions. In *Poster session of the European Semantic Web Conference, ESWC*, 2009.

[67] M. Maleshkova, L. Gridinoc, C. Pedrinaci, and J Domingue. Semi-automatic acquisition of semantic RESTful service descriptions. In *Poster session of 2nd STI International Offsite*, 2009.

[68] M. Maleshkova, C. Pedrinaci, and J. Domingue. Semantically annotating RESTful services with SWEET. In *Demo session at 8th International Semantic Web Conference ISWC*, 2009.

[69] M. Maleshkova, G. A. Rey, and A. Simov. D2.1.3 Service provisioning platform first prototype documentation. In *SOA4All Deliverables*, August 2009.

[70] C. Pedrinaci, M. Maleshkova, G. A. Rey, C. R. Moreno, S. Dietze, and A. Gugliotta. D2.1.1 Service provisioning platform design. In *SOA4All Deliverables*, March 2009.