

Semantically Enabling Web Service Repositories

Marta Sabou, Maria Maleshkova, Jeff Pan

January 2011

### **Abstract**

The success of the Web service technology has brought topics such as software reuse and discovery once again on the agenda of software engineers. While there are several efforts towards automating Web service discovery and composition, many developers still search for services via online Web service repositories and then combine them manually. However, from our analysis of these repositories, it yields that, unlike traditional software libraries, they rely on little metadata to support service discovery. We believe that the major cause is the difficulty of automatically deriving metadata that would describe rapidly changing Web service collections. In this paper, we discuss the major shortcomings of current Web service repositories and, as a solution, we report on how to use techniques developed in the context of the Semantic Web (ontology learning, service annotation tools, automatic classifiers, metadata based presentation) to improve the current situation.

## 0.1 Introduction

Web service technology allows for uniform access via Web standards to software components residing on various platforms and written in different programming languages. At a technology level, we distinguish between WSDL<sup>1</sup>/ SOAP<sup>2</sup> services, which rely on a comprehensive stack of technology standards [1], and *Web APIs*, characterized by their relative simplicity and their suitability for the Web. Currently, many popular Web 2.0 applications such as Facebook, Flickr and Twitter offer easy-to-use, resource-oriented APIs, which not only provide simple access to different resources but also enable combining heterogeneous data coming from diverse services, in order to create data-oriented service compositions called mashups. As a result, Web APIs are becoming the trend for developing and using Web services. We provide more details about Web service technologies in Section 0.2.

A prerequisite to reusing and composing Web services is the ability to find the right service. However, Web service discovery is becoming problematic with the increased number of both WSDL-based services (around 28,000<sup>3</sup>) and Web APIs (around 1900<sup>4</sup>, but growing rapidly). Despite such large numbers of services, the state of the art solution for finding Web services is inspecting online service repositories or simply using Web search engines.

The first goal of this paper is to identify *Which are the problematic aspects of Web service repositories when facilitating access to their resources?* For this purpose, we perform a survey of Web service repositories and conclude on their limitations in terms of supporting search for Web services (Section 0.3). Our survey extends a similar analysis performed five years ago [2]. We find that many of the repositories investigated in 2005 are discontinued in 2010, and that newer repositories are still limited in their search functionalities by difficulties in metadata acquisition, reuse, and its meaningful presentation.

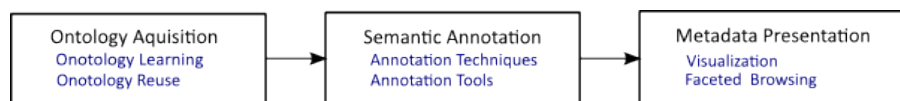


Figure 1: Main steps needed for semantically enhancing Web service repositories.

Our second goal is to understand *How can Semantic Web techniques improve search and browsing support in Web service repositories?* Recent research that combines Semantic Web and Web service technology is performed in the area of Semantic Web Services, where Web service tasks (including discovery, composition, and invocation) are automated by augmenting services with their formal semantic descriptions<sup>5</sup> [3]. This technology relies on complex service descriptions covering all aspects of the service (input/output parameters, internal working model, execution details) as well as dedi-

<sup>1</sup><http://www.w3.org/2002/ws/desc/>

<sup>2</sup><http://www.w3.org/TR/soap/>

<sup>3</sup><http://webservices.seekda.com/>

<sup>4</sup>Numbers from April 2010, <http://www.programmableweb.com/>

<sup>5</sup>See <http://www.daml.org/services/> for a number of initiatives.

cated tools that perform reasoning on these descriptions. While promising important benefits, this semantic web service technology is too heavy-weight for the purposes of improving search/browse functionalities in service repositories.

As an alternative, we propose using a light-weight semantic approach for semantically enhancing Web service repositories (see Figure 1). First, an ontology that formally describes the domains of the web services needs to be acquired. Ontology acquisition can be supported by ontology learning methods (Section 0.4). Second, the services contained within the repository need to be semantically annotated with metadata based on the terminology of the previously built ontology. This task can be achieved either by relying on (semi-) automatic methods or by using annotation tools (Section 0.5). The metadata attached to the services can be exploited to support semantically enhanced browsing functionalities (Section 0.6). We provide a summary in Section 0.7.

## 0.2 Technology Background

In this section we provide a brief overview of the types of Web services and the existing formalisms, which can be used for their semantic annotation.

### 0.2.1 Types of Web Services

We distinguish two types of Web services. On the one hand, “classical” Web services, based on WSDL and SOAP, play a major role in the interoperability within and among enterprises and serve as the basic construct for the rapid development of low-cost and easy-to-compose distributed applications in heterogeneous environments [4]. WSDL 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 is used for encoding the messages exchanged between the service consumer and provider. WSDL and SOAP are complemented by additional specifications such as WS-Addressing, WS-Messaging, WS-Security, etc., which build up the complete stack of technology standards.

On the other hand, Web APIs, which conform to the REST paradigm [5], are commonly referred to as RESTful services. RESTful services are centred around resources, which are interconnected by hyperlinks and grouped into collections, whose retrieval and manipulation is enabled through a fixed set of operations commonly implemented by using HTTP. In contrast to WSDL-based services, RESTful service are more lightweight in their technological stack, relying almost entirely on the use of URIs, for both resource identification and interaction, and HTTP for message transmission. In addition, RESTful services are characterised by resource-representation decoupling, so that resource content can be accessed via different formats. Finally, all resources of a RESTful service are modified with a fixed set of operations: GET (retrieve the current state of resource), POST (transfer the current state of resource), PUT (create new resource), DELETE (delete resource). However, many available Web APIs do not follow all of these REST principles.

## 0.2.2 Models and Frameworks for Capturing Service Metadata

The main idea underlying SWS [3] is the use of formal descriptions of Web service characteristics as means for facilitating their reusability and automating many common Web service tasks such as discovery, negotiation, composition and invocation. The main approaches for the development of semantic Web services can roughly be divided into two categories– top-down and bottom-up.

On one hand, *top-down approaches* such as WSMO [6] and OWLS [7] use high-level ontologies as frameworks for describing Web services. These approaches are based on the assumption that the service semantics, such as ontologies, functional, non-functional, and behavioral descriptions, are defined during the process of creating the service and specifying its invocation (WSDL) and communication (SOAP) implementation details.

On the other hand, *bottom-up models* adopt an incremental approach to adding semantics to existing Web service standards, adding specific extensions to WSDL that connect the syntactic definitions to their semantic annotations. Bottom-up approaches are especially suitable in the case of Web service repositories, where service definitions already exist and provide the basis for building up metadata, while at the same time enabling their automated processing. For example, the SAWSDL [8] defines a set of extension attributes for WSDL, which enable the linking of semantic entities to the WSDL service description.

In comparison to WSDL-based services, approaches for the creation of semantic descriptions of RESTful services are hampered because the majority of the Web APIs are described in textual form in Web pages and do not have WSDL-like machine-processable documentations. The lack of a common structured language for describing Web APIs is addressed by some initial work [9], [10], however, still the majority of the APIs are documented directly in HTML. Therefore, in order to overcome this difficulty, annotation approaches rely on marking and tagging the service properties, within the HTML, so that these can be subsequently enhanced with metadata [11]. MicroWSMO [12] and SA-REST [13] are two formalisms for the semantic description of RESTful services, which are based on adapting the SAWSDL approach.

## 0.3 Web Service Repositories: State of the Art

In this section we identify the limitations of online Web service repositories in terms of their search and browsing functionalities. We start by summarizing some major lessons learned from research on software libraries (Section 0.3.1). We then perform an overview of online Web service repositories in terms of their functionalities for accessing services. In particular, we focus on comparing 1) the search possibilities, 2) the availability of service metadata and 3) to what extent this metadata is employed to facilitate improved search functionalities (Sections 0.3.2 and 0.3.3). Finally, we conclude on the major limitations of Web service repositories (Section 0.3.4).

### 0.3.1 Lessons Learned from Software Libraries

Storage and retrieval methods for software assets have been studied for almost three decades. A major survey of software reuse libraries concludes in 1998 that, even if many sophisticated approaches exist to build and exploit such libraries, “*the practice is characterized by the use of ad-hoc, low-tech methods*” [14]. The practically viable approaches offer a good ratio between ease (and low cost) of implementation on the one hand and a reasonable performance coupled with ease of use on the other.

From the six major types of approaches discussed by the survey, the *Information Retrieval* and *Descriptive* methods are the most widely used. Information retrieval methods regard software assets (source code, comments, design artifacts) as documents and adapt indexing techniques to these collections. Descriptive methods classify software assets in terms of a list of (predefined) keywords. A popular descriptive method is that of *faceted classification*, introduced by Pietro-Diaz [15]. In his approach, the keywords that describe the assets are organized per (possibly orthogonal) facets, thus defining a multidimensional search space (where each facet corresponds to a dimension). The survey considers that descriptive methods provide a better performance (in terms of precision and recall) and are easier to use. One of their drawbacks is that the acquisition of the right keywords and the classification of the assets according to these keywords increases their cost [14].

### 0.3.2 WSDL-based Web Service Repositories

The landscape of WSDL-based web service repositories has changed substantially in the last five years. While in 2005 we overviewed seven such repositories [2], in 2010 we found that four of them have been discontinued (UDDI, BindingPoint, .NET XML Web Services Repertory, SalCentral) and two were stagnant (WebserviceX.NET, Xmethods). The only repository which is still operational and is being extended to also cover Web APIs is Web Service List (described in Section 0.3.3 where we focus on Web API repositories). Additionally to these earlier repositories, which we present here to give a complete picture of repository technologies and its evolution, in 2010, we can report on two new web service repositories which contain much more services and provide better search functionalities than their predecessors.

**1. Seekda! Services**<sup>6</sup> automatically crawls and indexes WSDL-based services, now containing around 28,500 services. The services can be accessed based on keyword search mechanisms as well as a form based search in terms of the service provider, hosting country and associated tags. Browsing is currently done by using a Tagcloud. For each service, besides general details, Seekda! also maintains user ratings and service availability history.

**2. Service-Finder**<sup>7</sup> allows access to more than 20,000 web services by extending and improving Seekda! technology. The site offers keyword based search and browsing in terms of a tag cloud or a classification scheme. The scheme has eight top level categories which are further specialized to two-levels of depth.

---

<sup>6</sup><http://webservices.seekda.com/>

<sup>7</sup><http://demo.service-finder.eu/index>

**3. WebserviceX.NET**<sup>8</sup> is a Web service provider that offers about 70 services grouped in seven categories which form the basic browsing mechanism. For each service, it provides a summary, an endpoint as well as details about the structure of the SOAP messages and the WSDL file. It is also possible to invoke the services from this site. This provider has the same number of services and browsing mechanisms as in 2005.

**4. Xmethods**<sup>9</sup> is one of the largest Web service repositories containing several hundred services<sup>10</sup>. However, this site only provides a list of services and no support for browsing or search facilities of any kind.

The following repositories are discontinued in 2010.

**5. UDDI**<sup>11</sup> was a cross-industry effort driven by major platform and software providers to establish an industry standard business registry that would facilitate Universal Description, Discovery and Integration of businesses and services. Different vendors (Microsoft, IBM, SAP) offered interfaces to this repository. **UDDI-Microsoft** allowed both searching and browsing facilities. Browsing could be done according to several categorization schemes describing industry sectors (three versions of the North American Industry Classification System - NAICS), product catalogs (three versions of the United Nations Standard Products and Services Code - UNSPSC), geographic information (microsoft-com:geoweb:2000, ubr-uddi-org:iso-ch:3166-2003) and a small Web service classification scheme. This scheme contained 19 terms denoting domains (e.g., Health, Weather) and functionality types (e.g., Search, Printing). The search functionality was limited to searching for services whose name started with a given string. **UDDI-IBM** provided a form based search (both for businesses and services), on the name of the services and a locator in one of the categorization schemes. IBM, Microsoft and SAP have discontinued their support for the UDDI registry, starting from 2006<sup>12</sup>.

**6. BindingPoint**<sup>13</sup> was a Web service repository offering both search and browse facilities. Searching for a keyword would return any service containing that keyword as a substring of its name or description. This lack of tokenization lead to undesired effects, e.g., searching for “date” returned services containing “*validate*” or “*update*” (clearly not related to dates).

Browsing the services was done via two classification schemes. Firstly, the BindingPoint scheme contained eight top categories which were further specialized up to two levels. Secondly, the Visual Studio scheme consisted of 15 categories without further specialization. Note, however, that although some of the categories in the two schemes had the same name there was a considerable mismatch in their content. For example, the *Calendar* category had three instances in one classification scheme and twenty in the other.

---

<sup>8</sup><http://www.websvcicex.net>

<sup>9</sup><http://www.xmethods.com>

<sup>10</sup>425 on 19.07.2005, and only 386 on 16.02.2010

<sup>11</sup><http://www.uddi.org>, discontinued from 2006.

<sup>12</sup><http://uddi.microsoft.com/about/FAQshutdown.htm>

<sup>13</sup><http://www.bindingpoint.com/>, discontinued in early 2010.

7. **.NET XML Web Services Repertory**<sup>14</sup> offered a simple keyword based search on UDDI data using BindingPoint technology on both service names and descriptions.

8. **SalCentral**<sup>15</sup> aggregated services published in other repositories (a meta-repository), offering both search and faceted classification based browsing. Searching was only performed on WSDL method names and textual service descriptions. However, search did not take into account naming conventions of the composed method names. By considering each name as a string and simply performing substring search lead to several problematic cases, e.g., searching for “text” retrieved *GetGeoIPContext* and *GetExtendedRealQuote*.

This repository was the only one providing a multi-facet based browsing. Services were classified into six facets: the name of the method, country, toolkit, domain, hosting server, suffix. Browsing was only possible on a single facet once, one could not impose filters by selecting values from different facets. Since the values of the last four facets could be determined automatically they did not present any anomalies. However, we had several observations related to the first facet. The “**by method name**” facet contained a list of keywords that frequently occur in the names of Web service methods, e.g.:

Accounts, Address, Airport, Bill, Category, City, Credit, Country, Currency, Customer, Database, Date, Domain, Email, Fax, File, Flight, Invoice, Location, Message, News, Postcode, Quote, Shop, Search, Sms, State, Tax, Time, Town, Validate

It is unclear whether these terms have been manually identified or their selection involved some automatic analysis of the available services. There are several flaws in the categorization of services under these keywords:

**Incorrect instances.** The instantiation of categories with Web services was often incorrect - any Web service was a member of a category if the denoting keyword was contained in its name. For example, the “Date” category inappropriately contained services such as “*validateEmailAddress*” or “*updateAccountInfo*”.

**Incomplete categories.** Several categories only had a few instances (e.g., four for *Flight*). Nevertheless, frequently mentioned terms were missing from the keyword list. For example, searching for “text” returned four pages of results (about eighty hits) and searching for “phone” returned about 40 hits, although none of these terms appeared as keywords of the “by method name” facet. Therefore, there is a mismatch between the terms mentioned by the collected services and those usable for browsing.

**Lack of abstractions.** Finally, many of these keywords were interrelated in a way that would have allowed grouping them into more generic (abstract) classes and building a deeper hierarchy to support browsing.

---

<sup>14</sup><http://www.xmlwebservices.cc/>, discontinued.

<sup>15</sup><http://www.salcentral.com/>, discontinued.

### 0.3.3 Web API Repositories

The task of discovering Web APIs is more challenging than the search for WSDL-based Web services, as there is no widely accepted structured language for describing Web APIs and RESTful services (although initial work in this area has started [16], [11]). As a consequence, search engines and crawlers, cannot directly differentiate between common Web sites and Web API descriptions. Therefore, the location of Web APIs is mainly a task which has to be completed manually and which could be improved by semantic techniques.

**9. ProgrammableWeb**<sup>16</sup> is a rapidly growing directory currently containing 1900 Web APIs and 4700 mashups<sup>17</sup>. In terms of access mechanisms it provides (a) a keyword search functionality; (b) filtering based on service category, provider company, implementation protocols and styles, data format, and date of registry; (c) sorting based on name, date, popularity and category; and (d) tagcloud. Although each entry has a detailed description (e.g., authentication mechanisms, output formats, service endpoint) this additional metadata is not exploited by the search and filtering mechanisms.

**10. iServe**<sup>18</sup> is a hybrid repository for publishing and discovering semantic Web services (around 1800 at the time of writing). It supports service annotations of both RESTful and WSDL-based services, expressed in a variety of forms (e.g., SAWSDL, WSMO-Lite, MicroWSMO, OWLS), by transforming these into a common Minimal Service Model (MSM). MSM enables uniform search over the different service types and forms stored in the repository by automatically generating the corresponding RDF statements. In this way, all the available service metadata can be directly used in the service search.

iServe provides access to its data through a Web-based application for browsing services, a SPARQL endpoint for querying and a RESTful API. The Web-based service browser includes functionalities for service classification based on predefined taxonomies or taxonomies loaded by the user.

**11. APiFinder**<sup>19</sup> provides around 1100 Web APIs, which can be accessed through searching or browsing functionalities. The search functionality relies on substring search over the names and descriptions of the services (e.g., searching for “date” returns services that mention “up-to-date”). Browsing can be done in terms of two facets: the main domain of the service (ten domains including, for example, E-commerce, Financial, Graphics/Games) or its implementation details in terms of programming language/operating systems. The services within a given category can be sorted on the alphabetical order of their names, by the date when they were added and by their provider.

**12. Webmashup.com**<sup>20</sup> contains around 1800 Web APIs and 3100 mashups. In terms of access mechanisms it offers (a) substring search on service names and descriptions; (b) browsing in terms of 50 categories denoting the general domains of the services, such as Sports, Retail, Music; (c) tagcloud generated from the tags added to

<sup>16</sup><http://www.programmableweb.com>

<sup>17</sup>Numbers from survey done in April 2010.

<sup>18</sup><http://iserve.kmi.open.ac.uk>

<sup>19</sup><http://www.apifinder.com>

<sup>20</sup><http://www.webmashup.com>

each service; (d) alphabetical search on service names. There are no additional sorting or filtering functionalities. We also note that, although each service is described in terms of a variety of details (e.g., provider, fees, implementation details), these are not exploited for providing more advanced search functionalities.

**13. Web Service List**<sup>21</sup> provides 22 categories for browsing an estimate of 200 Web APIs and Web services. These categories denote the domain of Web services (e.g., Multimedia, Healthcare, Business/Finance) or a certain functionality they provide (e.g., Conversion, Search/Finders, Calculators). Besides, Web services can be browsed alphabetically. Further, the site offers a substring-based search facility over the description of Web services. This site also allows rating services, but does not maintain any written reviews, and it is also unclear how new ratings are incorporated with the current score. The website contains many banners and commercial advertisements, which makes the location of relevant information challenging at first.

**14. WSFinder**<sup>22</sup> is a blog aiming to collect existing Web APIs, currently 230. The APIs are listed in a number of blog posts, where each Web API entry has a name, a description and a link to the provider's documentation. There are no sorting or search functionalities and the Web APIs are shown in a list.

**15. WebAPI.org**<sup>23</sup> includes around 40 popular Web APIs, which can be browsed according to six general categories or based on the provider's name. There are no sorting or search functionalities.

### 0.3.4 Conclusions

Based on the previously presented overview we conclude that the situation of Web service repositories is similar to that of software reuse libraries depicted a decade ago [14]. In particular:

**A. Simple techniques are used.** We encountered four simple ways of accessing the content of Web service repositories (see Table 1 for an overview). First, **search** is performed on (various combinations of) the textual sources attached to the Web services, such as their name, description or the names of the WSDL operations. In the case of Web API repositories, search is performed over the detailed API description. Search is similar to the information retrieval methods implemented for software libraries. We encountered three cases of searching where matching is done at token level (*keyword search*) and eight cases where matching is done at substring level (*substring search*). Note that, substring search leads to many low relevance results.

**Browsing** based on different categorization schemes (corresponding to descriptive techniques used in software libraries) is extensively used in Web service repositories. There are two types of schemes employed. First, large industry standard thesauri such as UNSPCSC and NAICS are used. These schemes are often under-populated and it is not always obvious which path to take to find what one needs. On the other hand, light-weight Web service specific classification schemes are also used. Some repositories

<sup>21</sup><http://www.webservicelist.com/>

<sup>22</sup><http://wsfinder.typepad.com>

<sup>23</sup><http://www.webapi.org/webapi-directory/>

Repository	Search	Browse	Other
1. Seekda! Services	keyword search	2-facet classification	Tagcloud
2. Service-Finder	keyword search	1-facet classification	Tagcloud
3. WebserviceX.NET		1-facet classification	
4. Xmethods			List
5. UDDI	substring search	product catalogs 1-facet classification	
6. BindingPoint	substring search	1-facet classification 2 classification schemes	
7. .NET XML Web Services Repertory	substring search		
8. SalCentral	substring search	6-facet classification	
9. ProgrammableWeb	keyword search	5-facet classification	Tagcloud
10. iServe	substring search in service properties	multiple classification schemes	
11. APIFinder	substring search	2-facet classification	
12. Webmashup.com	substring search	1-facet classification	Tagcloud
13. Web Service List	substring search	1-facet classification	
14. WSFinder			List
15. WebAPI.org		2-facet classification	

Table 1: Overview of retrieval methods in Web service repositories.

also offer, browsing based on the Web API's names. Finally, two repositories present all their services as a **list**. **Tagclouds** are increasingly used for content access.

**B. Browsing relies on few and low quality metadata.** Current Web service classification schemes are light-weight. Unlike the industry standard schemes, they have only a few top categories (max. 20) which, in most cases, are not further specialized. Besides their reduced size and scope, Web service schemes are also qualitatively poor. For example, there is a high level of ambiguity of their scope since their categories often correspond to different facets. Some describe domains of activity (e.g., Health, Multimedia) while others name functionality types (e.g., Search, Find). Further, there is a mismatch between the content provided by the existing services and that covered by the categories. As a result, many categories are over-populated with instances and there is a need to extend the set of categories with new terms as the underlying data set evolves. Finally, it is often unclear how the categories are populated. In some cases, two identical categories are populated completely differently from the same set of services (e.g., BindingPoint).

**C. The metadata is not fully exploited for presentation.** We found that sites, which possessed richer metadata did not fully exploit this semantics for presentation. In particular, one of the advantages of faceted classifications is that they allow browsing on multiple facets at the same time (similar to a multi keyword search). However, current repositories allow inspecting only one facet at a time.

In the next sections we investigate a set of Semantic Web techniques that would foster the acquisition of high-quality, multi-faceted metadata in order to improve the browsing functionalities offered by Web service repositories (Sections 0.4 and 0.5). We also describe a few techniques that allow an intuitive presentation of faceted metadata (Section 0.6).

## 0.4 Ontology Acquisition

The first step in semantically enhancing service repositories is the acquisition of an ontology that would describe the main concepts relevant for the indexed services by using ontology learning methods. Ontology learning deals with developing methods for (semi-)automatically deriving ontologies from unstructured, semi-structured and structured data sets [17]. The stringent need of acquiring ontologies imposed by the development of the Semantic Web lead to a variety of approaches to this problem and several tools that implement diverse ontology learning algorithms [18]. In previous work we successfully experimented with adapting existing ontology learning methods to deriving ontologies from textual Web service descriptions [19], [20]. In this section we show how ontology learning methods can be used to evolve Web service classification schemes or to extend them with new facets. We rely on preliminary results to strengthen the viability of some of our ideas.

### 0.4.1 Evolving Existing Classification Schemes

The fast development of the Web services technology leads to a rapid growth in the number of available services. As a result many Web service categorization schemes lag behind the actual content needs of a dynamically changing collection of Web services (e.g., SalCentral). This is often due to the fact that domain experts have difficulties in identifying the terms that best describe a collection of services ([19], [20]), since they do not perform a meticulous investigation of the available service descriptions but rather rely on their own view of the domain to define the best terms. We experimentally demonstrated that ontology learning techniques can support domain experts to identify the most frequent terms used by the community. Therefore, we propose to use concept identification to extend classification schemes in a way that they better reflect the content of a service repository.

To test our proposal, we ran our ontology learning approach on a subset of SalCentral services and identified several terms (see below) that would extend the keywords of the “*by name*” facet. When used as search keywords, most of these terms returned tens of services thus proving their relevance for the underlying collection.

text, temperature, stock, status, chart, company, price,  
payment, article, distance, language,

find, convert, verify, simulate, play, create, store, check,  
track, translate, calculate, validate.

Besides these terms, we also extracted terms that would specialize existing keywords (i.e., “deepen” the scheme) as shown in Figure 2.

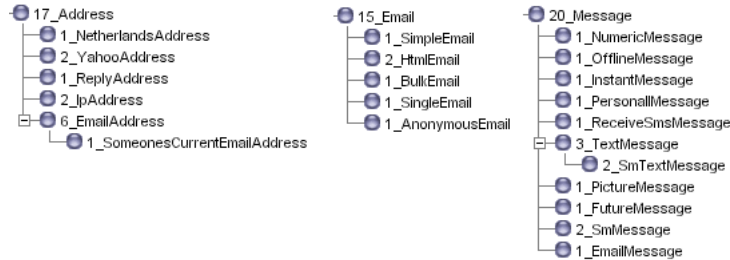


Figure 2: Extracted specialization hierarchies.

## 0.4.2 Learning New Facets

In Section 0.3.4 we observed that faceted metadata is almost absent in current repositories probably due to its high acquisition cost. During our experiments we found that the values for some of the basic facets can be easily identified by using simple pattern matching techniques on the textual description of the services or by inspecting their WSDL documentation. In this section we demonstrate our ideas about learning facets related to operational features (input, output, functionality) and restrictions.

**Inputs, Outputs, Functionalities.** The type of input, output parameters and the action performed by a Web service are often enough to identify the right service. While none of the analyzed repositories allow searching on these features, these could easily be identified with (semi-)automatic techniques.

First, the *textual descriptions attached to Web services* often contain this information. In fact, in our previous work we found that these texts exhibit strong syntactic characteristics (they use a sublanguage [21]) and that this allows extracting the desired information by employing pattern based extraction rules. In particular, we observed that most of the noun phrases in these texts denote the parameters of the service while verbs indicate the functionality of the service. For example, in the following Web service descriptions the noun phrases *image*, *url address*, *hyperlink*, *web site*, *contact information*, *global address* denote the parameters of the service. The verbs *extract*, *validate* and *enhance* indicate the functionality of the service.

- \* Extracts images from a given url address.
- \* Extracts hyperlinks from a given web site.
- \* Validate and enhance contact information for any global address.

Note that the above heuristics do not determine precisely, which are the inputs and outputs of the service. For this, more refined rules can be defined. For example, “*given*” in front of a noun phrase indicates that it plays the role of an input. We are currently working on identifying such heuristics.

The second source of information for determining inputs, outputs and functionalities are *the WSDL files that describe Web services*, in particular, the method and message names they contain. Preliminary investigations show that WSDL files are often more accurate than textual descriptions and that a combination of both sources should give the best results.

**Restrictions.** Besides operational features, such as inputs and outputs, other features can be important when choosing a service, e.g., the geographic area where the service is active. Some repositories (e.g., SalCentral) try to deduce this feature from the country extension of the URL where the service description is published. However, this seldom indicates the geographic region, for which the service was built. For example, a Web service that “*validates and enhances contact information for any address in India*” can be published at a *.com* address<sup>24</sup>. Conversely, a Web service whose URL contains a certain country identifier (e.g., France<sup>25</sup>) might perform a service that is independent of geographic constraints (e.g., cipher/decipher).

An alternative solution to determining geographic constraints for a service is to use NER systems. Such systems automatically identify geographic entities, persons and organizations in free text. NER technology matured in the previous decades to reach performances of 80-90% Precision and Recall for a generic system (such as ANNIE) and 90-95% for systems that are tuned to the needs of particular domains [22].

- \* Search through all Swedish telephone subscribers.
- \* Search UK Index.
- \* This webservice return longitude, latitude and height from a given city. Only for France.
- \* Lookup ATM Locations by Zip Code (US Only).

For example, for the Web service descriptions above, our experiments show that the ANNIE NER system recognizes *Swedish, UK, US, France* as references to the corresponding countries. We observed that, in some cases, the restriction is strengthened by the use of “*only*” in constructions such as “*only for/in country*” or “*country only*”. These constructs can be easily identified using a regular expression based rule mechanism.

### 0.4.3 Identifying Abstractions

The methods we presented so far identify important information about Web services. However, to be useful for browsing or even reasoning, these terms should be placed into subsumption hierarchies. There are several methods used to deduce subsumption relations. For example, in [23] four different techniques are combined to determine a subsumption hierarchy:

1. Hearst style lexico-syntactic patterns are matched against large corpora [24]. For example, the text snippet *carnivores such as lions, tigers* matches the pattern *NP0 such as NP1, NP2 .. ⇒ isA(NP1, NP0), isA(NP2, NP0)* and derives that lions and tigers are kinds of carnivores.

<sup>24</sup><http://ws.strikeiron.com/IndianAddressVerification?WSDL>

<sup>25</sup><http://www.quisque.com/fr/chasses/crypto/cesar.asm?WSDL>

2. A similar approach is used to determine subsumption relations by taking advantage of large scale web data. Given two terms, a set of Hearst like patterns are built up with them. The occurrences of these patterns on the Web are analyzed to determine the most likely relations.
3. WordNet is inquired for hypernymy information for the analyzed terms.
4. Vertical relations, as described in [25], are identified. This approach regards a term  $t1$  obtained by adding an extra modifier to a term  $t2$  as more specific than  $t2$ , e.g., “XML string” is more specific than “string”.

Our ontology learning approach uses a vertical relations based algorithm to derive hierarchies of concepts that serve as parameters for the analyzed Web services. For example, from a subset of SalCentral services we learned hierarchies as those in Figure 2. While this algorithm performs well in terms of Precision (the majority of identified relations are valid), it can only learn subsumptions indicated by compositionality (this results in a low Recall).

We also experimented with Hearst based patterns but these are rare in the textual sources attached to Web services. For example, when analyzing around 450 descriptions, only 10 contained subsumption information identifiable with Hearst patterns. We will further explore the use of WordNet and the Web for hierarchy learning in this domain.

## 0.5 Acquiring Semantic Annotations

A core step when semantically enhancing repositories is that of attaching semantic metadata to its services, i.e., to describe the service in terms of semantic concepts defined in a domain ontology. This ontology can either be learned from service descriptions, as shown in the previous section, or can be re-used from ontologies published online by relying on ontology search engines such as Watson [26] and Sindice [27]. In this section we describe tools that support users when annotating services (Section 0.5.1) and methods that automatically annotate services (Section 0.5.2).

### 0.5.1 Web Service Annotation Tools

The ASSAM [28] tool automatically suggests semantic metadata based on two machine learning algorithms. The first one, performs iterative relational classification for semantically classifying Web Services, their operations, and input and output messages. The second algorithm aggregates the data returned by multiple semantically related Web Services by facilitating schema mapping based on an ensemble of string distance metrics. ASSAM includes a panel-based user interface, which displays the service WSDL file, the classification ontology, and the recommended annotation.

Similarly to ASSAM, the METEOR-S Web service Annotation Framework [29] enables the semi-automatic marking up of Web service descriptions with ontologies by using predefined domain ontologies for categorizing Web services into domains. The

tool applies graph similarity techniques to select a relevant domain ontology for a given WSDL file from a collection of ontologies.

SWEET<sup>26</sup> supports the creation of semantic descriptions for Web APIs rather than WSDL-based services [30]. SWEET is a Web application, which takes as input an HTML Web page describing a Web API and offers functionalities for annotating the service properties with semantic information. The tool uses hRESTS [11] for marking service properties within textual Web API descriptions and making the descriptions machine-processable. In addition, it uses the MicroWSMO microformat for linking semantic annotations to the identified service properties. The results are a semantically annotated HTML description of the RESTful service and an RDF MicroWSMO description. These can either be saved locally or be directly posted to the iServe repository.



Figure 3: Using the SWEET interface to search for ontologies that are suitable for semantically annotating a Web API.

Figure 3 shows SWEET’s user interface and its support for searching for suitable domain ontologies by accessing Watson in an integrated way.

## 0.5.2 Automatic Methods for Annotating Services

Automatically assigning a Web service to a particular group of services with similar functionality or similar application domain, already provides semantic information. Most approaches rely mainly on the information provided in the service WSDL files to complete important tasks such as determining the relevant domain ontology or determining the type of service.

An approach focusing on the Web service annotation task is that of Patil et al [29], who apply graph similarity techniques to select a relevant domain ontology for a given

<sup>26</sup><http://sweet.kmi.open.ac.uk/>; <http://sweetdemo.kmi.open.ac.uk/war/MicroWSMOeditor.html>

WSDL file from a collection of ontologies. This approach relies on matching XML schemas to ontologies suitable for describing Web Services. The authors 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 various ontologies when annotating services. Therefore, they also address the problem of choosing the right domain ontology within a set of ontologies.

One main task in the context of service annotation is the task of Web service classification. In some approaches, the classification task can be equivalent to determining a suitable domain ontology for the annotation of the service [31], while in other approaches it can at least simplify the process of determining a domain ontology and finally, sometimes [32] 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 [32], 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 [33].

A largely unexplored technology for supporting Web service annotation is that of recommender systems, i.e., systems which exploit already existing semantic data to suggest possible annotation alternatives to users. Content based recommendations could rely on the similarity between semantically annotated services and the current Web service to recommend annotations. Or, collaborative recommender approaches could allow users with similar profile characteristics to use the same domain ontologies or ontological entities for annotating services.

## 0.6 Metadata Based Presentation

While important, the acquisition of metadata is just the first step towards improving Web service repositories. The intuitive presentation of this metadata is crucial to truly taking advantage of its value. Faceted browsing and visual techniques are frequently used to perform metadata based presentation.

### 0.6.1 Faceted Browsing

Several application domains have shown that (rich) faceted metadata provides a good basis for powering faceted browsing. For examples, faceted browsing interfaces were built for large image collections in the Flamenco project<sup>27</sup> [34] or to inspect museum item collections in the MuseumFinland project<sup>28</sup> [35]. This technology is reaching maturity as commercial vendors offer products that automatically generate faceted interfaces from adequate metadata.

The open source software repository, Sourceforge<sup>29</sup>, allows a faceted based browsing of the available applications. One can gradually narrow his search by imposing filters on the values of the available features. In the analyzed Web service repositories

---

<sup>27</sup><http://bailando.sims.berkeley.edu/flamenco.html>

<sup>28</sup><http://museosuomi.cs.helsinki.fi>

<sup>29</sup><http://sourceforge.net/>

only SalCentral allows accessing different facets. However, there is no interaction between these facets, i.e. one cannot restrict the value of several facets at the same time. Faceted browsing based portals could be easily built once the required Web service metadata is in place.

## 0.6.2 Visualisation

Another way to present metadata is through visualisation techniques. Previously, we showed that visualisation of faceted metadata can support user tasks such as analysis, comparison and search [36]. We used Cluster Map, a visual technique integrated in several applications [37], to visualize *instances* of a set of *classes* according to their classification into these classes.

The Cluster Map could support the task of searching for Web services based on the automatically derived faceted metadata (using the previously described methods). Our current methods allow extracting two facets of the analyzed services: the types of their parameters and their functionality. These two facets are enough to answer queries that supply a functionality and a parameter type, e.g., a service that finds addresses.

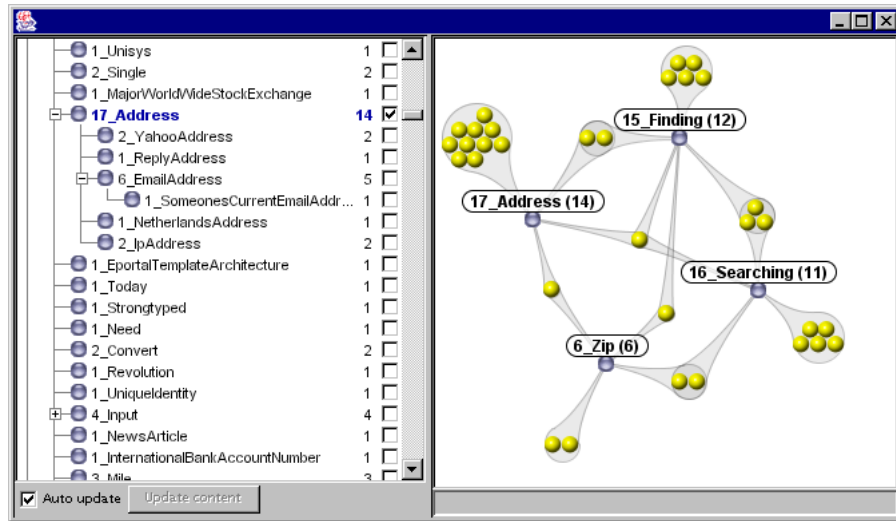


Figure 4: An interface for visual search.

The Cluster Map technique is embedded in an interactive GUI as depicted in Figure 4. The right pane displays the hierarchy of terms. In this example, the hierarchy was automatically derived. The user of the interface can browse the hierarchy and select the terms that define his query. In the case of our query, the user might choose to see all services that offer *search* or *find* functionalities (from the *functionalities* facet). Also, he wants to see services that have parameters of type *address* and *zip* (these are values from the *parameters* facet). Note that, by displaying all the domain relevant terms, we offer support for formulating the user's query in terms that are actually used within the service collection.

The selected terms are visualised in the right pane with their name and cardinality stated in a rounded rectangle. Balloon-shaped edges connect instances (small yellow spheres) to their most specific class(es). In this case the instances of a term are all the Web services that are described by that term. Instances with the same class membership are grouped in clusters (similar to Venn Diagrams). In our example, there are several clusters formed, one of them showing the intersection between *Address* and *Finding*. This cluster contains two Web services, which have a parameter of type *Address* and perform the action of *Finding* - thus they represent the answer to our example query. The instances in a cluster can be accessed with a mouse click.

This visualisation allows the user to explore the service collection. For example, in the example scenario, one might be interested to see what other services provide find functionalities, or to inspect the service that allows finding zip codes. Further, by using the specialization hierarchy in the left pane, the query can be refined, for example, by choosing more specialized terms (in our case, the user might actually be interested in email addresses).

## 0.7 Summary

In this chapter we investigated the use of various Semantic Web related techniques to enhance current online Web service repositories. Our overview of Web service repositories, both in 2005 and 2010, yielded that they rely on little and qualitatively poor metadata. As a consequence they offer only limited support for performing manual Web service discovery. Inspired by the lessons learned from traditional software libraries, we believe that the use of rich faceted metadata would be required. However, we are aware that acquiring such metadata, especially for describing Web service collections that are changing on a daily bases, is prohibitively expensive.

As a solution, we think that techniques developed for the Semantic Web, which are concerned with metadata acquisition and presentation, have a great potential in solving the current problems of Web service repositories. In particular, ontology learning techniques can be adapted for extending and keeping up to date the current service classification schemes. They can also be used to derive the information for several facets that describe services or to arrange the extracted terms in meaningful subsumption hierarchies. We already presented some encouraging results when using ontology learning techniques. Service annotation tools and automatic classification methods can then be used to attach semantic metadata to services, either in terms of a given domain ontology or by reusing concepts from various, online available ontologies. Finally, rich faceted metadata can be exploited for building intuitive browsing interfaces. We exemplified that even the light-weight metadata, which we derived automatically, can enhance the search for Web services when coupled with visualisation techniques.

# Bibliography

- [1] Gustavo Alonso, Fabio Casati, Harumi Kuno, and Vijay Machiraju. *Web Services*. Springer, 2003.
- [2] Marta Sabou and Jeff Pan. Towards Semantically Enhanced Web Service Repositories. *Journal of Web Semantics*, 2007.
- [3] Sheila McIlraith, Tran Cao Son, and Honglei Zeng. Semantic Web Services. *IEEE Intelligent Systems. Special Issue on the Semantic Web*, 16(2):46–53, March/April 2001.
- [4] Michael Papazoglou, Paolo Traverso, Schahram Dustdar, and Frank Leymann. Service-Oriented Computing: A research roadmap. *International Journal of Cooperative Information Systems*, 17(2), 223-255, 2008.
- [5] Roy Thomas Fielding. *Architectural Styles and the Design of Network-based Software Architectures*. PhD thesis, University of California, 2000.
- [6] Dumitru Roman, U. Keller, Holger Lausen, Jos de Bruijn, Ruben Lara, Michael Stollberg, Axel Polleres, Cristina Feier, Christoph Bussler, and Dieter Fensel. Web Service Modeling Ontology. *Applied Ontology*, pages 1(1): 77 – 106, 2005.
- [7] David Martin. OWL-S: Semantic Markup for Web Services. W3C member submission, 22 November 2004. <http://www.w3.org/Submission/OWL-S/>, 2004.
- [8] Joel Farrell and Holger Lausen. Semantic Annotations for WSDL and XML Schema (SAWSDL). W3C recommendation. <http://www.w3.org/TR/2007/REC-sawSDL-20070828/>, August 2007.
- [9] Marc Hardley. Web Application Description Language (WADL). Technical report, Sun Microsystems, November 2006.
- [10] Roberto Chinnici, Jean-Jacques Moreau, Arthur Ryman, and Sanjiva Weerawarana. Web Services Description Language (WSDL) Version 2.0). W3C Recommendation, June 2007.
- [11] Jacek Kopecký, Karthik Gomadam, and Tomas Vitvar. hRESTS: An HTML Microformat for Describing RESTful Web Services. In *WI-IAT '08: Proc. of the IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology*, pages 619–625. IEEE Computer Society, 2008.

- [12] Jacek Kopecký and Tomas Vitvar. MicroWSMO. CMS Working Draft. <http://www.wsmo.org/TR/d38/v0.1/20080219/>, February 2008.
- [13] Amit Sheth, Karthik Gomadam, and Jon Lathem. SA-REST: Semantically interoperable and easier-to-use services and mashups. *IEEE Internet Computing*, 11(6):91-94, 2007.
- [14] Ali Mili, Rym Mili, and Roland Mittermeir. A survey of software reuse libraries. *Annals of Software Engineering*, 5:349 – 414, 1998.
- [15] Ruben Pietro-Diaz. Implementing Faceted Classification for Software Reuse. *Communications of the ACM. Special issue on software engineering.*, 34(5):88 – 97, May 1991.
- [16] RDFa in XHTML: Syntax and Processing. Recommendation W3C. <http://www.w3.org/TR/rdfa-syntax/>, October 2008.
- [17] Alexandre Maedche and Steffen Staab. Ontology Learning for the Semantic Web. *IEEE Intelligent Systems*, 16(2):72–79, March/April 2001.
- [18] Asuncion Gomez Perez and David Manzano Mancho. A Survey of Ontology Learning Methods and Techniques. OntoWeb Deliverable 1.5, May 2003.
- [19] Marta Sabou. From Software APIs to Web Service Ontologies: a Semi-Automatic Extraction Method. In *Proc. of the Third International Semantic Web Conference, ISWC*, Hiroshima, Japan, November 2004.
- [20] Marta Sabou, Chris Wroe, Carol Goble, and Gilad Mishne. Learning Domain Ontologies for Web Service Descriptions: an Experiment in Bioinformatics. In *Proc. of the 14th International World Wide Web Conference*, Chiba, Japan, May 2005.
- [21] Ralph Grishman and Richard Kittredge, editors. *Analyzing Language in Restricted Domains: Sublanguage Description and Processing*. Lawrence Erlbaum Assoc., Hillsdale, NJ, 1986.
- [22] Hamish Cunningham, Diana Maynard, Kalina Bontcheva, and Valentin Tablan. GATE: A Framework and Graphical Development Environment for Robust NLP Tools and Applications. In *Proc. of the 40th Anniversary Meeting of the Association for Computational Linguistics*, Philadelphia, July 2002.
- [23] Philipp Cimiano, Aleksander Pivk, Lars Schmidt-Thieme, and Steffen Staab. Learning Taxonomic Relations from Heterogeneous Evidence. In *Proc. of the ECAI04 Workshop on Ontology Learning and Population*, Valencia, Spain, 2004.
- [24] Marti A. Hearst. Automatic Acquisition of Hyponyms in Large Text Corpora. In *Proc. of the 14th International Conference on Computational Linguistics*, 1992.

- [25] Paola Velardi, Michele Missikoff, and Paola Fabriani. Using Text Processing Techniques to Automatically enrich a Domain Ontology. In *Proc. of the International Conference on Formal Ontology in Information Systems*, pages 270–284. ACM Press, 2001.
- [26] 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.
- [27] Giovanni Tummarello, Renaud Delbru, and Eyal Oren. Sindice.com: Weaving the open linked data. In *Proc. of the International Semantic Web Conference (ISWC)*, 2007.
- [28] Andreas Hess, Eddie Johnston, and Nicholas Kushmerick. ASSAM: A tool for semi-automatically annotating semantic web services. In *Proc. of the 3rd International Semantic Web Conference (ISWC)*, 2004.
- [29] Abhijit Patil, Swapna Oundhakar, Amit Sheth, and Kunal Verma. METEOR-S web service annotation framework. pages 553–562. ACM Press, 2004.
- [30] Maria Maleshkova, Carlos Pedrinaci, and John Domingue. Supporting the creation of semantic RESTful service descriptions. In *Service Matchmaking and Resource Retrieval in the Semantic Web Workshop (SMR2) at ISWC*, 2009.
- [31] Andreas Hess and Nicholas Kushmerick. Learning to attach semantic metadata to web services. In *Proc. of the 2nd International Semantic Web Conference (ISWC)*, pages 258–273, 2003.
- [32] Nicole Oldham, Christopher Thomas, Amit Sheth, and Kumal Verma. METEOR-S web service annotation framework with machine learning classification. In *Proc. of the 1st Int. Workshop on Semantic Web Services and Web Process Composition (SWSWPC)*, 2004.
- [33] Marcello Bruno, Gerardo Canfora, Massimiliano Di Penta, and Rita Scognamiglio. An Approach to support Web Service Classification and Annotation. In *EEE ’05: Proc. of the 2005 IEEE International Conference on e-Technology, e-Commerce and e-Service (EEE’05)*, pages 138–143, Washington, DC, USA, 2005. IEEE Computer Society.
- [34] Ka-Ping Yee, Kirsten Swearingen, Kevin Li, and Marti Hearst. Faceted Metadata for Image Search and Browsing. In *Proc. of the SIGCHI conference on Human factors in computing systems*, pages 401 – 408, Ft. Lauderdale, Florida, USA, 2003. ACM Press.
- [35] Eero Hyvönen, Eetu Mäkelä, Mirva Salminen, Arttu Valo, Kim Viljanen, Samppa Saarela, Miikka Junnila, and Suvi Kettula. MuseumFinland – Finnish Museums on the Semantic Web. *Journal of Web Semantics*, 3(2):224–241, 2005.

- [36] Chritiaan Fluit, Marta Sabou, and Frank van Harmelen. Ontology-based Information Visualisation. In Vladimir Geroimenko, editor, *Visualising the Semantic Web*. Springer Verlag, 2002.
- [37] Chritiaan Fluit, Marta Sabou, and Frank van Harmelen. Ontology-based Information Visualization: Towards Semantic Web Applications. In Vladimir Geroimenko, editor, *Visualising the Semantic Web, Second Edition*. Springer Verlag, 2005.