Available online at www.sciencedirect.com
ScienceDirect
Web Semantics: Science, Services and Agents on the World Wide Web xxx (2007) xxx–xxx
JOURNAL OF Web Semantics
www.elsevier.com/locate/websem

Towards semantically enhanced Web service repositories

Marta Sabou [a,*], Jeff Pan [b]

[a] *Knowledge Media Institute & Centre for Research in Computing, The Open University, UK*
[b] *Department of Computing Science, University of Aberdeen, UK*

Received 7 June 2006; accepted 10 November 2006

Abstract

The success of the Web services technology has brought topics 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 online 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 state of the art Web service repositories and as a solution, we report on ongoing work and ideas on how to use techniques developed in the context of the Semantic Web (ontology learning, matching, metadata based presentation) to improve the current situation.

Keywords: Web services repositories; Semantic Web; Ontology learning; Ontology mapping; Reasoning; Metadata presentation

The Web service technology allows uniform access via Web standards to software components residing on various platforms and written in different programming languages. As a result, software components providing a variety of functionalities (ranging from currency conversion to flight booking) are now accessible via the Web. Uniform access is ensured by the use of a stack of XML based standards (WSDL,[1] SOAP[2]) and leads to an increased interoperability between heterogeneous software components. As such, Web services technology has enabled reuse and composition between different software modules. Indeed, Web service technology has introduced a new abstraction layer over and a radically new architecture for software. From a business perspective, Web services often correspond to business services and thus the compositionality paradigm that underlies the Web service technology allows composing existing business services into new and more complex services.

A prerequisite to reusing and composing Web services is the ability to find the right services. However, Web service discovery is becoming problematic with the increasing number of Web services to several hundreds (e.g., there are already 1000 Web services in bioinformatics [9]). Current research efforts investigate the possibility to automate Web service tasks (such as discovery and composition) by augmenting services with their formal semantic description[3] [11]. While this advanced technology is under development, the state of the art solution for finding Web services on the Web is inspecting online repositories of such services.

In this paper we investigate the state of the art technology employed by online ontology repositories and explore possible uses of Semantic Web methods to provide possible enhancements. In particular, we investigate two research questions:

- Which are the problematic aspects of Web service repositories? To answer this question we perform a survey of existing online Web service repositories and conclude on some of their major drawbacks (Section 1). We find that metadata acquisition and its meaningful presentation are two major problems that underly all negative aspects of these repositories.
- Which Semantic Web techniques could be used and how? Metadata acquisition and its presentation are core issues for the Semantic Web. In this paper, we discuss how methods used in ontology learning (Section 2), ontology matching (Section 3) and metadata based presentation (Section 4) could be useful to enhance the Web service repositories.

* Corresponding author. Tel.: +44 1908 653800; fax: +44 1908 653169.
E-mail address: r.m.sabou@open.ac.uk (M. Sabou).
[1] Web Service Description Language, http://www.w3.org/2002/ws/desc/.
[2] Simple Object Access Protocol, http://www.w3.org/TR/soap/.
[3] See http://www.daml.org/services/ for a number of initiatives.

1570-8268/$ – see front matter © 2007 Elsevier B.V. All rights reserved.
doi:10.1016/j.websem.2006.11.004

The primary goal of this article is to explore a possible synergy between Semantic Web and (a particular aspect of) software engineering. As a result, we provide a broad range of suggestions on how Semantic Web related methods could be used for the analyzed problem. Inherently from the broad scope of our analysis, some of our suggestions are at an idea level. However, many of them rely on a set of experiments that are conducted in the context of current work to enhance Web service repositories.

1. Web service repositories: state of the art

In this section we tackle the first research question. For this (1) we summarize some major lessons learned from research on software libraries, (2) we perform an overview of online Web service repositories and (3) conclude on the major limitations of these repositories in comparison with software library standards.

1.1. Lessons learned from software libraries

Storage and retrieval methods for software assets have been studied for almost three decades. However, a major survey of software reuse libraries concludes 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*" [12]. The practically viable approaches offer a good ratio between ease (and low cost) of implementation on one hand and a reasonable performance 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 in practice, as it will be detailed later in this text. Other methods are more complex and therefore not used in practice. They rely on more formal descriptions of software components than the *Information Retrieval* and *Descriptive* methods. For example, *operational semantics* methods use the executability of software components as a basis for their selection and rely on a formal interface and behavioral specifications of the components. The *denotational semantics* methods rely on signature matching to identify the right components. *Topological* methods take into considerations not just functional but also a structural comparison between components. Finally, *structural* methods compare components based on their structures (e.g., quicksort and selection sort have the same sorting function but they rely on a completely different internal structure).

Information retrieval methods regard software assets (source code, comments, design artifacts) as documents and adapt indexing techniques to these collections. The methods that are used largely vary across the approaches depending of the nature of documents that are considered (e.g., source code versus design documents). Information retrieval methods are easy to implement as several tools performing such task already exist and the actual principles are simple and well understood. These methods are also easy to use because users can phrase their query in natural language. However, the typically low precision and high recall of these methods delegates much of the selection task to the user.

Descriptive methods represent a link between the information retrieval methods that are agnostic to the special characteristics of software assets and the more formal software oriented approaches, which, unfortunately are too complex to be applied in practice. The underlying principle is to classify software assets in terms of a list of (predefined) keywords. The most well known descriptive method is that of *faceted classification*, introduced by Pietro-Diaz [17]. 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 concludes that descriptive methods provide a better performance (in terms of precision and recall) and are easier to use. However, one of their drawbacks is that the acquisition of the right keywords and the classification of the assets according to these keywords increases the cost of their implementation [12].

1.2. An overview of online Web service repositories

In this section we overview seven Web services repositories. For each of them we describe the facilities that they offer to access the available services and point out problematic aspects when it is the case.

1. UDDI[4] is a cross-industry effort driven by major platform and software providers to establish an industry standard business registry which aims to facilitate Universal Description, Discovery and Integration of businesses and services. Different vendors (Microsoft, IBM, SAP) offer interfaces to this large repository. Hereby we present our findings for UDDI IBM and Microsoft. UDDI-Microsoft[5] allows both searching and browsing facilities. Browsing can be done according to several categorization schemes describing industry sectors (three versions of the North American Industry Classification System—NAICS[6]), product catalogs (three versions of the United Nations Standard Products and Services Code—UNSPSC[7]), geographic information (microsoft-com:geoweb:2000, ubr-uddi-org:iso-ch:3166-2003) and a small Web service classification scheme. This scheme contains 19 terms denoting domains (e.g., Health, Weather) and functionality types (e.g., Search, Printing). The search functionality is rather limited as it only allows searching for services whose name starts with a given string. UDDI-IBM[8] provides a form based search on the name of the services (both for businesses and services) and a locator in one of the categorization schemes.
2. Bindingpoint[9] is a repository of XML-based Web services. This site offers both search and browse facilities. Searching for a keyword will return any Web service which contains that keyword as a substring of the strings denoting the Web services name and description. This lack of tokenization leads to undesired effects. For example, when searching for

[4] http://www.uddi.org.
[5] http://uddi.microsoft.com/.
[6] http://www.census.gov/epcd/www/naics.html.
[7] http://www.unspsc.org/.
[8] https://uddi.ibm.com/beta/find.
[9] http://www.bindingpoint.com/.

Please cite this article in press as: M. Sabou, J. Pan, Towards semantically enhanced Web service repositories, Web Semantics: Sci. Services Agents World Wide Web (2007), doi:10.1016/j.websem.2006.11.004

"date", any services that contain words such as "vali*date*" or "up*date*" (which are clearly not related to dates) are returned.

There is some ambiguity involved also in the presentation of the results. All resulting services are shown as well as the categories where a Web service matching the search criteria was found. One would expect that when accessing these categories only those services which match the search criteria would be shown. This would be similar to performing a compound query, e.g., search for all the *Calendar* type services that mention *date*. However, this is not the case—a click on the categories in this search context reveals all their members.

Browsing the available services can be done via two classification schemes. The BindingPoint scheme amounts to eight top categories which are further specialized up to two levels. The Visual Studio scheme consists of 15 categories without further specialization. Note, however, that even if some of these categories have the same name there is a considerable mismatch in their content. For example, the *Calendar* category has three instances in one classification scheme and 20 in the other.

3. NET XML Web Services Repertory[10] offers a simple keyword based search on UDDI data using BindingPoint technology on both service names and descriptions. The same unsolicited results are obtain as in the case of BindingPoint.
4. WebserviceX.NET[11] is a Web service provider that currently offers about 70 services. These services are grouped in seven categories which form the basic browsing mechanism.
5. Web Service List[12] provides 17 categories for browsing the available services (estimated 200). 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, but this functionality is not very helpful when one does not know the exact name of the service he is looking for. Further, the site offers a search facility which searches the name and description of Web services. Unlike the BindingPoint technology this search works on correct tokenizations (i.e., matches search terms to whole words only and not to substrings in the given text).
6. Xmethods[13] is one of the largest Web service repositories containing already several hundred services.[14] However, this site provides only a long list of services. It has no support for browsing nor does it provide any search facilities.
7. SalCental[15] is a Web service repository which aggregates services published in other repositories (a meta-repository). It offers both search and a faceted classification based browsing. Searching is only performed on WSDL method names and textual service descriptions. However, search does not keep account of naming conventions of the composed method names. By considering each name as a string and simply performing substring search leads to several problematic cases. For example, searching for "text" retrieves: "GetGeoIPCon*text*", Ge*tExt*endedRealQuote.

This repository is the only one that attempts a multi-facet based browsing. Services are classified according to six facets: the name of the method, country, toolkit, domain, hosting server, suffix. Browsing is only possible on a single facet once, one cannot impose filters by selecting values from different facets. Since the values of the last four facets (toolkit, domain, hosting server, suffix) can be determined automatically they do not present any anomalies. However, we have several observations related to the first two facets.

The "by method name" facet offers a list of keywords that frequently occur in the names of the methods offered by Web services. These keywords are:

- Accounts, Address, Airport, Audio, Bill, Category, City, Credit, Client, Country, Currency, Customer, Database, Date, Domain, Email, Fax, File, Flight, Historical, Invoice, Location, Message, News, Postcode, Quote, Shop, Search, Sms, State, Supplier, Tax, Time, Town, User, Validate, Weather, Zipcode.

It is not clear how these terms were derived, whether they have been simply manually selected or their selection involved some automatic analysis of the available services. There are several flaws in this categorization:

- Incorrect instances. Each category denoted by a keyword contains a set of Web services characterized by that keyword. However, this instantiation of categories with Web services is often incorrect—any Web service is a member of a category if the denoting keyword is contained in the name of the service. For example, when browsing the "Date" category we find instance services such as "vali*date*EmailAddress" or "up*date*AccountInfo" which clearly should not belong to this category. This is a direct consequence of the employed search algorithm.
- Incomplete keyword set. Several keywords have only a few instances (e.g., four instances for *Flight*). Nevertheless, terms that appear more often are missing from the offered keyword list. For example, searching for "text" returns four pages of results (about eighty hits) and searching for "phone" returns about 40 hits. This fact suggests that there is a mismatch between the terms frequently used by the collected services and those that are offered for browsing.
- Lack of abstractions. Finally, many of these keywords are interrelated in a way that would allow grouping them in more generic (abstract) classes and building a deeper hierarchy to support browsing.

The "by country" facet offers a list of countries. The membership of a Web service to a country category is deduced based on the country extension of the URL or by using the manually added location information available in UDDI. Note, however,

[10] http://www.xmlwebservices.cc/.
[11] http://www.webservicex.net.
[12] http://www.webservicelist.com/.
[13] http://www.xmethods.com.
[14] 425 on 19.07.2005.
[15] http://www.salcentral.com.

Please cite this article in press as: M. Sabou, J. Pan, Towards semantically enhanced Web service repositories, Web Semantics: Sci. Services Agents World Wide Web (2007), doi:10.1016/j.websem.2006.11.004

Table 1
Overview of access methods in Web service repositories

Repository	Search	Browse	List
UDDI	Substring search	Product catalogs One facet classification	
Bindingpoint	Substring search	One facet classification	
NET XML Web services repertory	Substring search		
WebserviceX.NET		One facet classification	
Web service list	Keyword search	One facet classification	
Xmethods			Yes
SalCentral	Substring search	Six facet classification	

that the country extension of the URL does not always reflect the activity range of the service (we provide an example about this in Section 2.2).

1.3. 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 in [12]. In particular:

- Simple techniques are used. We encountered three 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 (this is similar to the information retrieval methods implemented for software libraries). We encountered one case of searching where matching is done at token level (*keyword search*) and four cases where matching is done at substring level (*substring search*). Note that, substring search leads to many hits that have no content relevance for the search key.

  The second extensively used access method in Web service repositories is browsing based on different categorization schemes (similar to descriptive techniques used in software libraries). There are two types of schemes employed. On the one hand, large industry standard thesauri such as UNSPCSC and NAICS are used. These schemes are often under-populated (i.e., several of their categories describe no or few services) and it is not always obvious which path to take to find what one needs. On the other hand, lightweight Web service specific classification schemes are also used. Finally, one repository that we analyzed uses no metadata to support browsing but simply presents all the available services as a large list.
- Browsing relies on few and low quality metadata. Current Web service classification schemes are lightweight. Unlike the industry standard schemes, they have only a few top categories (maximum 20) which, in most cases, are not further specialized. These schemas contain information about a single facet (except the case of SalCentral). 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 Web services to be categorized and that covered by the categories. As a result, many categories are overpopulated 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 identically named categories are populated completely differently from the same set of services (see our previous example about the *Calendar* category in the BindingPoint portal).
- 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 only inspecting one facet at a time.

Searching for Web services is a complex issue. Their domain of activity is just one of the many criteria that characterize them. Especially when searching for services with the goal to reuse them in other applications it is important to know the functionality they offer, the type of input they require, the type of output that they produce, the restrictions that may apply on them.

Web service repositories seldom use multi-faceted metadata. Our hypothesis is that the major cause for this is the cost of acquiring these metadata. High cost is mainly caused by two factors. First, Web service libraries can be quite large. They often contain a couple of hundred services. Building an extensive and balanced categorization scheme for these services requires a considerable manual effort. The second factor, aggravating the first one is that these repositories are changing almost on a daily basis. Therefore, the problem is not so much building the categorization scheme but rather maintaining it up to date over time.

In the next two sections we investigate how techniques from the fields of ontology learning and matching could be used to (semi-)automatically acquire and enhance metadata for Web service repositories. In Section 4 we overview a few techniques that allow an intuitive presentation of faceted metadata. We also show that some metadata we derived from our preliminary experiments can be successfully used as a basis for intuitive visualisations.

2. Ontology learning

Ontology learning deals with developing methods for (semi-)automatically deriving ontologies from unstructured, semi-structured and structured data sets [10]. The stringent need of acquiring ontologies imposed by the development of the Semantic Web lead to the development of a large variety of approaches to this problem and already several tools that implement diverse ontology learning algorithms [16]. In previous work we successfully experimented with adapting existing

Please cite this article in press as: M. Sabou, J. Pan, Towards semantically enhanced Web service repositories, Web Semantics: Sci. Services Agents World Wide Web (2007), doi:10.1016/j.websem.2006.11.004

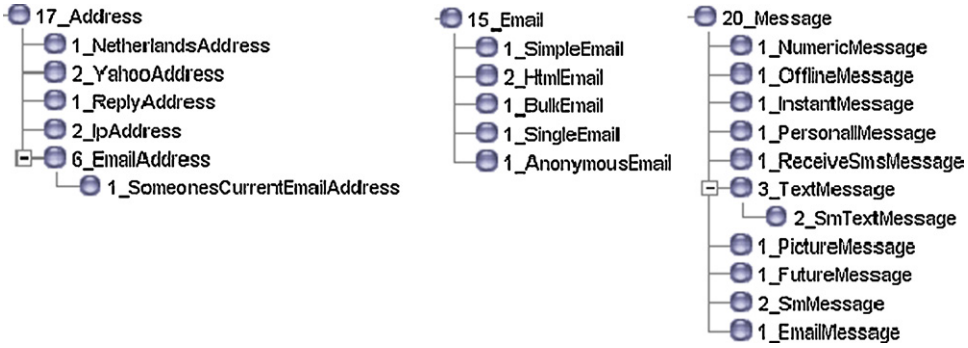

Fig. 1. Extracted specialization hierarchies.

ontology learning methods to deriving ontologies from textual Web service descriptions [18,19]. 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 some preliminary experimental results to strengthen the viability of some of our ideas.

2.1. Evolve 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. We experienced this phenomena during our survey of Web service repositories.

Our previous work showed that it is difficult for a domain expert to identify the terms that best describe a given collection of services ([18–20]). The reason is that human experts do not perform a meticulous investigation of all available descriptions but rather rely on their own view of the domain to define the best terms. Our experiments showed that ontology learning techniques can support domain experts to identify the most frequent terms used by the community. Our proposal is to use concept identification to extend existing classification schemes and thus ensure that they truly reflect the content of the underlying repository.

For example, we ran our ontology learning module on a collection of services extracted from SalCentral and identified a set of terms (see bellow) that would extend the set of existing keywords in the "*by name*" category. We tested the relevance of these terms for the collection of Web services by searching for them in the collection. Several terms covered tens of services, thus proving their relevance for the collection.

- text, temperature, stock, status, chart, company, word, price, payment, article, distance, language, find, convert, verify, simulate, play, create, store, check, track, translate, calculate, validate.

Besides terms for broadening the existing scheme, our experiments also provided terms that would specialize existing keywords (i.e., "deepen" the scheme). Fig. 1 depicts a few examples of specialization hierarchies that we learned. Inevitably our automatic methods introduce a couple of errors. However, what is interesting to note here is that they also discover several new terms. These terms help to specialize the existing terms in the manually built categories. Our previous experiments in applying these algorithms to two separate domains have shown that on average 50% of the learned terms are domain relevant [20]. Therefore, this technique can be used to (semi-)automatically extend existing categorization schemas so that they reflect the underlying service collection.

2.2. Learn new facets

In Section 1.3 we stressed the importance of using faceted classification schemes when describing services. We also observed that these faceted information is almost absent in current repositories probably due to its 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 documentation of the services or inspecting their WSDL documentation. In this section we demonstrate our ideas about deducing operational features (input, output, functionality) and restrictions.

2.2.1. Inputs, outputs, functionalities

The type of input and output parameters as well as the action performed by a Web service are in many cases enough to identify the needed service. While none of the analyzed repositories allow searching on these features, they can 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 very strong syntactic characteristics (they use a sublanguage [6]) and that this allows extracting the desired information by employing a few 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.

Please cite this article in press as: M. Sabou, J. Pan, Towards semantically enhanced Web service repositories, Web Semantics: Sci. Services Agents World Wide Web (2007), doi:10.1016/j.websem.2006.11.004

- 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 names of the WSDL methods and messages. From preliminary investigations it seems that WSDL files are often more accurate in providing the right terminology to describe a service than the global textual descriptions of services. Our idea is that a combination of both sources should give the best results. This topic is also subject to current work.

2.2.2. Restrictions

Besides operational features, such as inputs and outputs, other features can be important when choosing a service. In particular the geographic area where the service is active is an important consideration. Current repositories 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.[16] Conversely, a Web service whose URL contains a certain country identifier (e.g., France[17]) might perform a service that is independent of geographic constraints (e.g., in the case of the example service—cipher/decipher).

An alternative solution to determining geographic constraints for a service is to use Named Entity Recognition (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% Precision and Recall for systems that are tuned to the needs of particular domains [3].

- 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.

[16] http://ws.strikeiron.com/IndianAddressVerification?WSDL.
[17] http://www.quisque.com/fr/chasses/crypto/cesar.asmx?WSDL.

2.3. 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 [2] four different techniques are combined to determine a subsumption hierarchy:

(1) Hearst style lexico-syntactic patterns are matched against large corpora [7]. For example, the text snippet *carnivores such as lions, tigers* matches the pattern *NP0 such as NP1*, *NP2*,... $\Rightarrow$ *isA*(*NP1*, *NP0*), *isA*(*NP2*, *NP0*) and results in determining that lions and tigers are kinds of carnivores.
(2) A similar approach is used to determine subsumption relations by taking advantage of the large amount of data offered by the World Wide Web. Given two terms, a set of Hearst like patterns are built up with them. The occurrences of these patterns on the Web are counted and then normalized to determine the most likely relations.
(3) WordNet is inquired for hypernymy information for the analyzed terms.
(4) Vertical relations, as described in [22], are identified. This approach regards a term *t1* obtained by adding an extra modifier to a term *t2* as more specific than *t2*. For example, the term "*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, by analyzing a collection of Web service from SalCentral we learned hierarchies as those depicted in Fig. 1. The advantage of this simple algorithm is that it performs well in terms of Precision (the majority of so identified subsumption relations are valid). The drawback is that 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.

3. Ontology reasoning and matching

In the previous section we presented a set of techniques to acquire metadata for characterizing Web services. In this section, we briefly discuss how to reuse some existing ontologies and make use of them in Web service repositories, with the help of ontology reasoning and matching.

Besides learning ontologies from existing data sets, we can also reuse existing ontologies available from the Web. The first

Please cite this article in press as: M. Sabou, J. Pan, Towards semantically enhanced Web service repositories, Web Semantics: Sci. Services Agents World Wide Web (2007), doi:10.1016/j.websem.2006.11.004

step is to get the candidates by using ontology search tools like OntoSearch [24]. Now the *question* is: how can ordinary users, who may not know OWL at all, decide which ontology suits their application best?

A good solution [14] is to combine ontology reasoning services and natural language generation to provide human-readable presentation of parts of ontologies. Natural language is well equipped to express the complex logical structures that arise in modern ontologies. Thus the point is trying to find ways of exploiting this medium. An ontology takes the form of a set of logical axioms, and so the challenge is to present the material of these axioms in comprehensible way using a language such as English. However, it is important to take on board the fact that the axioms may not come in a form ready for direct realisation in English. The axioms represent one possible way that the material could have been expressed, but there are many other possible ways that this could have been done equally well. For the ontology writer, the choice is arbitrary—because one can rely on reasoning services for ontologies, it is not necessary to worry about which of the many logically equivalent methods of expression to use. This means however that reasoning services must also be used in natural language generation. By making use of ontology classification service, one can ask questions such as "What is X" and "What are the differences between X and Y?" w.r.t. a certain ontology. For example, suppose that one wants to construct an entertainment ontology, and with the help of OntoSearch, he gets 50 candidate ontologies. By using some natural language presentation toolkit, they can ask questions like "What are movies?" and "What are the differences between cinemas and theaters?", so as to see which ontologies out of the 50 match their understanding of important concepts in their entertainment ontology.

Now suppose users successfully locate some candidate ontologies, with the help of natural language presentation techniques mentioned above, and reuse some parts of them. However, the only primitive for ontology reuse available in the Semantic Web is owl:imports [15], which allows one to "copy-and-paste" an ontology into another one. We can call this operation *syntactic import* as the operation is defined on the set of axioms of an ontology, rather than the knowledge encoded in the ontology itself. Another kind of import is *semantic imports* [21]. The main idea of semantic import is to make it possible for an ontology to import the "semantics/meaning" of a symbol defined in another ontology, rather than a set of axioms. Semantic imports allow one to declare their agreement on the meaning (interpretations) of some classes, properties and individuals from another ontology, so that one could get all the knowledge (from the original ontology) about them, being free to disagree with the meaning of other vocabulary. Serafini and Pan [21] provide a semantics for the "semantic import" primitive and a basic (only theoretical) distributed algorithm to compute the effects of semantic imports.

With the help of the above techniques, ontology users can construct their domain ontologies by reusing (parts of) existing ontologies. Domain ontologies are crucial in tasks such as reasoning-based searches in Web service repositories. The basic idea is that the capabilities of Web services can be described by terms, in the domain ontology, that can be published, for example, by a UDDI registry. When clients search the repository, they can specify the capability of their desired Web services with a query term. The result of a search contains the Web services that are classified (by ontology reasoners such as Instance Store [1]) as instances of the query term. For example, [9] describes an application of reasoning-based search within bioinformatics.

However, there could be multiple domain ontologies related to a set of Web service repositories. One of the challenges would be to harmonize these ontologies by building mappings among their terms. There is a variety of instance based matching techniques that could be used for this purpose (see [13] for an overview). A well studied task for ontology matching is to translate instances in the source ontology to instances of the target ontology, based on forward-chaining reasoning. With the help of this ontology matching task, a Web service repository can be extended by the increase of not only local registered Web services, but also ones in its federated Web service repositories.

4. Metadata based presentation

While important, the acquisition of metadata is just the first part of solving the problems of current Web service repositories. The intuitive presentation of this metadata is crucial to truly take advantage of its value. Faceted browsing and visual techniques are two frequently used ways to perform metadata based presentation.

4.1. 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 to browse large image collections in the Flamenco project[18] [23] or to inspect museum item collections in the MuseumFinland project[19] [8]. This technology is reaching maturity as software vendors offer commercial products that automatically generate faceted interfaces from adequate metadata. A highly relevant example is the semantics-based Spectacle tool-suite offered by the Dutch company Aduna.[20]

The open source software repository, Sourceforge,[21] 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 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.

Naturally, faceted browsing based portals can be easily built when the required metadata about Web services is in place.

[18] http://bailando.sims.berkeley.edu/flamenco.html.
[19] http://museosuomi.cs.helsinki.fi.
[20] http://aduna.biz/.
[21] http://sourceforge.net/.

Please cite this article in press as: M. Sabou, J. Pan, Towards semantically enhanced Web service repositories, Web Semantics: Sci. Services Agents World Wide Web (2007), doi:10.1016/j.websem.2006.11.004

4.2. Visualisation

Another way to present metadata is through visualisation techniques. Our previous work has shown that visualisation of faceted metadata can support several user tasks such as analysis, comparison and search [4]. We used Cluster Map [4], a visual technique developed by Aduna which is already integrated in several Semantic Web applications [5]. This technique visualises *instances* of a set of *classes* according to their classification into these classes.

In this section we give an example of using Cluster Map to support the task of searching for Web services based on the automatically derived faceted metadata (with the previously described methods). Our current methods allow extracting two facets of the analyzed services: the types of their parameters and the functionality that they offer. These two facets are enough to answer queries that supply a functionality and a parameter type. For example, imagine that a user needs a service that finds addresses.

The Cluster Map technique is embedded in an interactive GUI as depicted in Fig. 2. The left 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 chose 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 users 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. There is a third Web service that is relevant which, besides mentioning *Address* and *Finding* also mentions *Searching* (see the cluster formed at the intersection of these three terms). 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, he might be interested to see what other services provide *Find* functionalities, or to inspect the one service that allows finding zip codes. Further, using the specialization hierarchy in the left pane he can refine the query, for example by choosing more specialized terms (in our case, he might refine his query to search for email addresses rather than any kind of addresses). Note that refining queries in such a flexible way is currently not supported in any online Web service repository.

5. Summary

In this paper we investigated the use of various Semantic Web related techniques to enhance current online Web service repositories.

Our overview of Web service repositories 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

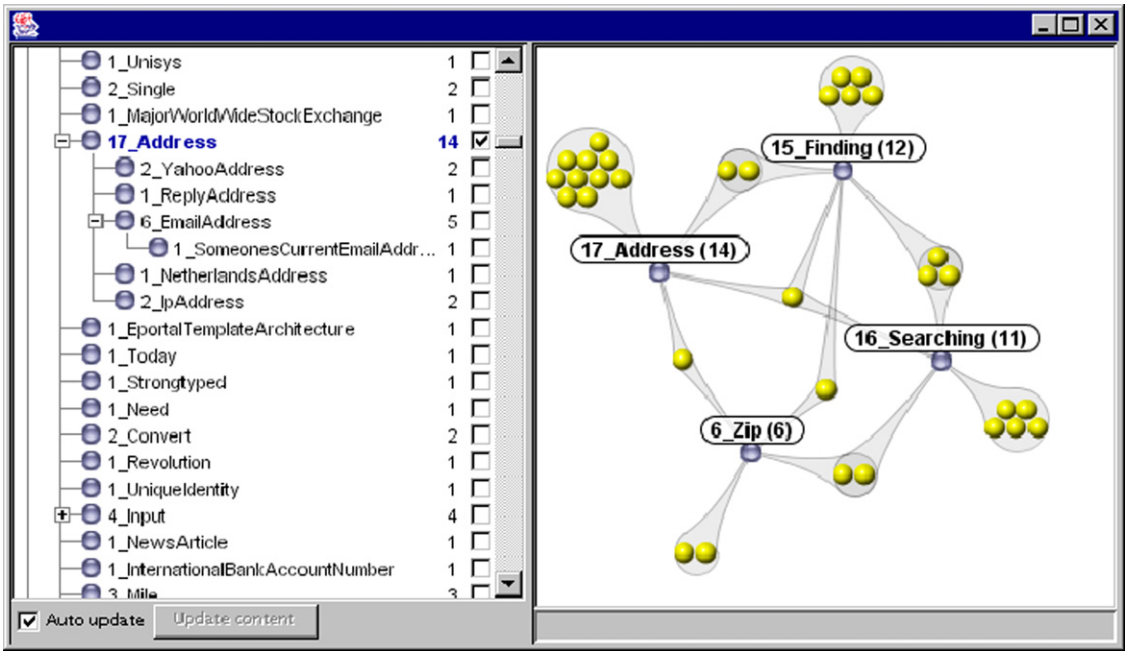

Fig. 2. An interface for visual search.

Please cite this article in press as: M. Sabou, J. Pan, Towards semantically enhanced Web service repositories, Web Semantics: Sci. Services Agents World Wide Web (2007), doi:10.1016/j.websem.2006.11.004

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 the ontology learning techniques. Ontology matching techniques could be used to harmonize existing schemes and to allow the construction of meta-schemas. Finally, rich faceted metadata can be exploited for building intuitive browsing interfaces. We exemplified that even the lightweight metadata that we derived automatically can significantly enhance the search for Web services when coupled with visualisation techniques.

Encouraged by our results so far, we prepare to implement all these ideas in a prototype system that would collect available Web services, automatically extract metadata describing different facets of these services and then would use this metadata to build an intuitive search/browse interface. Further, we believe that a minimal user intervention would be enough to "clean" the automatically derived metadata so that it can be used as a basis for (simple) reasoning tasks. This would bring the state of the art a step closer to the ultimate vision of Semantic Web services where semantics is added in a bottom-up fashion (learned from available sources) rather then being imposed in a top-down approach (requiring costly manual annotations).

References

[1] S. Bechhofer, I. Horrocks, D. Turi, The OWL instance store: system description, in: Proceedings of the 20th International Conference on Automated Deduction (CADE-20), Springer, 2005, pp. 177–181, Lecture Notes in Artificial Intelligence.

[2] P. Cimiano, A. Pivk, L. Schmidt-Thieme, S. Staab, Learning taxonomic relations from heterogeneous evidence, in: Proceedings of the ECAIWorkshop on Ontology Learning and Population, Valencia, Spain, 2004.

[3] H. Cunningham, D. Maynard, K. Bontcheva, V. Tablan, GATE: a framework and graphical development environment for robust NLP tools and applications, in: Proceedings of the 40th Anniversary Meeting of the Association for Computational Linguistics, Philadelphia, July 2002.

[4] C. Fluit, M. Sabou, F. van Harmelen, Ontology-based information visualisation, in: V. Geroimenko (Ed.), Visualising the Semantic Web, Springer Verlag, 2002.

[5] C. Fluit, M. Sabou, F. van Harmelen, Ontology-based information visualization: towards Semantic Web applications, in: V Geroimenko (Ed.), Visualising the Semantic Web, 2nd ed., Springer Verlag, 2005.

[6] R. Grishman, R. Kittredge (Eds.), Analyzing Language in Restricted Domains: Sublanguage Description and Processing, Lawrence Erlbaum Assoc., Hillsdale, NJ, 1986.

[7] M.A. Hearst, Automatic acquisition of hyponyms in large text corpora, in: Proceedings of the Fourteenth International Conference on Computational Linguistics, 1992.

[8] E. Hyvonen, E. Makela, M. Salminen, A. Valo, K. Viljanen, S. Saarela, M. Junnila, S. Kettula, MuseumFinland—Finnish Museums on the Semantic Web, J. Web Semantics 3 (4) (2005).

[9] P.W. Lord, P. Alper, C. Wroe, C.A. Goble, Feta: a light-weight architecture for user oriented semantic service discovery, in: A. Gomez-Perez, J. Euzenat (Eds.), The Semantic Web: Research and Applications. Proceedings of the Second European Semantic Web Conference, ESWC'05, Springer-Verlag, Heraklion, Crete, Greece, 2005, pp. 17–31, volume 3532 of LNCS.

[10] A. Maedche, S. Staab, Ontology learning for the Semantic Web, IEEE Intell. Syst. 16 (2) (2001) 72–79.

[11] S. McIlraith, T.C. Son, H. Zeng, Semantic Web services, IEEE Intell. Syst. Special Issue Semantic Web 16 (2) (2001) 46–53.

[12] A. Mili, R. Mili, R.T. Mittermeir, A survey of software reuse libraries, Ann. Softw. Eng. 5 (1998) 349–414.

[13] N.F. Noy, Semantic integration: a survey of ontology-based approaches, ACM SIGMOD Record 33 (4) (2004) 65–70.

[14] J.Z. Pan, C. Mellish, Supporting semi-automatic semantic annotation of multimedia resources, in: Proceedings of the 3rd IFIP Conference on Artificial Intelligence Applications & Innovations (AIAI 2006), pp. 609–617.

[15] P.F. Patel-Schneider, P. Hayes, I. Horrocks, OWL Web Ontology Language Semantics and Abstract Syntax, Technical report, W3C, February 2004. W3C Recommendation, URL http://www.w3.org/TR/2004/REC-owl-semantics-20040210/.

[16] A. Gomez Perez, D. Manzano Mancho, A Survey of Ontology Learning Methods and Techniques. OntoWeb Delieverable 1.5, May 2003.

[17] R. Pietro-Diaz, Implementing faceted classification for software reuse, Commun. ACM. Special Issue Softw. Eng. 34 (5) (1991) 88–97.

[18] M. Sabou, From software APIs to Web service ontologies: a semi-automatic extraction method, in: Proceedings of the Third International Semantic Web Conference, ISWC, Hiroshima, Japan, November 2004.

[19] M. Sabou, C. Wroe, C. Goble, G. Mishne, Learning domain ontologies for Web service descriptions: an experiment in bioinformatics, in: Proceedings of the 14th International World Wide Web Conference, Chiba, Japan, May 2005.

[20] M. Sabou, C. Wroe, C. Goble, H. Stuckenschmidt, Learning domain ontologies for Semantic Web service descriptions, J. Web Semantics 3 (4) (2005).

[21] L. Serafini, J.Z. Pan, SemanticImport: a smooth import for OWL ontologies, submitted for publication.

[22] P. Velardi, M. Missikoff, P. Fabriani, Using text processing techniques to automatically enrich a domain ontology, in: Proceedings of the International Conference on Formal Ontology in Information Systems, ACM Press, 2001, pp. 270–284.

[23] K.-P. Yee, K. Swearingen, K. Li, M. Hearst, Faceted metadata for image search and browsing, in: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, ACM Press, Ft. Lauderdale, Florida, USA, 2003, pp. 401–408.

[24] Y. Zhang, W. Vasconcelos, D. Sleeman, OntoSearch: an ontology search engine, in: Proceedings of the Twenty-fourth SGAI International Conference on Innovative Techniques and Applications of Artificial Intelligence, 2004.

Please cite this article in press as: M. Sabou, J. Pan, Towards semantically enhanced Web service repositories, Web Semantics: Sci. Services Agents World Wide Web (2007), doi:10.1016/j.websem.2006.11.004