

Finding Web Services

Holger Lausen and Thomas Haselwanter

Aleph Web Services GbR

firstname.secondname@aleph-webservices.com

April 18, 2007

Abstract

The Web is moving from a collection of static documents to a collection of services. For realizing service interchange the service-oriented architecture together with the Web Services technology are widely seen as the most promising fundament. As a result, considerable attention has been given, both in research and in industry, to Web Services and related technologies. While a significant number of papers have already been published in the area of semantic Web Service discovery, most of them are more concerned with a logical framework rather than providing a contribution that meets existing practical realities. In this paper, we take a different approach: first we provide an empirical analysis of approaches towards Web Service discovery that are in use and then we describe our methodology that is derived from current actualities in the area of publicly available Web Services. Finally we present our Web Service Search Engine prototype. By employing automated methods to obtain rich service descriptions it provides high quality search results on top of the largest pool of Web Services known so far.

1 Introduction

Web Service Technologies provide a set of standards to access computational components over the Internet, the acceptance by all major industrial player has lead to the fact that individuals and companies do now offer more and more of its business functionalities as Web Services. Similar to the way the standard Web technologies have extended the boundaries of a single enterprise, allowing individuals to access functionality using HTML Forms, the use of Web Services allows automatic processes to interact across the boundaries of a single company. This way business functionalities can be utilized without human intervention.

Web Service Technologies and in particular the Web Service Description Language (WSDL)[3] provide means to describe the properties of the technical interfaces as an XML document, however they do not provide support for all tasks during the Web Service life cycle: A Web Service must be published, discovered, details of the invocation sequence determined, security and monitoring issues addressed, etc. Currently those issues are completely handled by application designers and software engineers. Within this paper we focus on how to discover services. Within the literature there are numerous proposals on how to extend the existing WSDL based interface descriptions with semantics: [6, 8, 9, 10]. However rather than on taking current constraints and foreseeable evolvement of services into account the approaches focus purely on some logical

framework that could potentially make retrieval more accurate, without discussing in detail the underlying assumptions, such as from where the detailed annotation stems from.

In this paper we take a different approach. Based on previous work [1] we analyze existing approaches for Web Service Discovery, then we present our approach and implementation that is based on a completely automated process: we gather all available data on services, integrate it in one coherent model to allow discovery on an up-to-date set of services. In our model service descriptions are mainly created by automatically analyzing processes and community feedback rather than by relying on experienced knowledge engineers.

2 Existing Approaches for Web Service Discovery

As we can not make any statements about private respectively intra-net usage of Web services by its definition, all empirical results presented are based on publicly available Web services. Based on the sources for Web Service discovery mentioned in previous work [5, 7, 4, 1] and our own observation, we have identified three major approaches for discovering publicly available Web Services:

- UDDI (Universal Description, Discovery, and Integration) [2] is a standard for centralized repositories. The first UDDI Business Registry (UBR) nodes were run by IBM, Microsoft, SAP and NTT Com.
- Service directories (or portals) which gather services using focused crawlers or by manual registration and offer a search functionality via a HTML interface.
- Standard Web search engines which are able to restrict the search in some way to retrieve WSDL descriptions. Although this is no guarantee to find services this possibility provides the biggest coverage.

When looking at these approaches we focus on two aspects: (1) the number of services, (2) the quality and quantity of information that is associated with them.

Number of Services As first step we have investigated in more detail the number of services that can be found using the various means. In the case of UDDI we had to realize that the public repositories have been shut down in the beginning of 2006¹, such that we could not do an analysis ourselves, however earlier work [7] (2004) reported that only one third of the 1200 registered services contained references to valid WSDL files. To examine the number of services available in the various Web portals (see Table 1 for a complete list) we have extracted the number of unique WSDL files referenced from each site and subsequently verified the documents were still accessible. To determine the number of services available via search engines like Google² or Yahoo³ we

¹<http://www.uddi.org/find.html>

²<http://google.com>

³<http://yahoo.com>

Table 1: Survey Results

Repository	No. WSDL	retrieved WSDL	Liveliness	Categorization
UBR	1200	300	n	y
RemoteMethods	319	205	n	y
StrikeIron	638	508	y	y
Woogle	751	312	y	y
XMethods	505	460	n	n
programmableWeb	80	77	y	n
Google	262000	n/a	n	n
Yahoo	61800	n/a	n	n
Alexa	30846	3630	n	n

encountered 2 problems. First there are no means to retrieve all results⁴. The Second major problem is that there are no means to formulate a query that only returns Web Service descriptions. The numbers are obtained by querying for all resources having the keyword “wsdl” in its URL. Of course there is no guarantee that the document retrieved from those URLs eventually includes a WSDL document, however we could only verify this for the results we obtained from Alexa, where about 12% of the URLs did resolve to WSDL documents.

Available Information As we did not have access to the UDDI repositories, we could not evaluate the quality of information. However [7] reported very low quality with respect to the data. Search engines like Google don’t collect specific service related information: The ranking is based on the number of incoming links and the matching is done on a keyword level without considering the WSDL structure, moreover no availability or other service related information are given. The Web Service directories provide more information. This includes pricing data, scoring and review systems, provider information, textual descriptions and links to online documentation. Only Woogle [4] operates solely on the information given with the WSDL document. ProgrammableWeb⁵ as one of the portals considered has the most information fields (30), however many services only have filled out a small subset of those. Moreover it is a generic repositories for Web APIs and not only for Web Services that use the WSDL technology, in fact only 20% of the 400 registered services use WSDL.

Table 1 summarizes our observations. The first column gives the number of services that are reported by the respective means of discovery, the second column is the number of WSDL documents we could retrieve and the 3rd and 4th column indicate if a particular information is available.

Conclusions One has to conclude that for publicly available Web Services the UDDI based approach has failed and been discontinued. Using normal search engines one can achieve the broadest coverage, however one has to spent considerable time in browsing through results, since one can not filter those for Web Services efficiently. The various dedicated portal pages provide are at present the most convenient way to find

⁴Both Google and Yahoo show only the first 1000 results per Query.

⁵<http://www.programmableweb.com/>

Web Services. Except Woogles they rely on manually maintained repositories and have only a limited amount of services. We also included programmable Web as repository, however only a small portion of the APIs registered are using WSDL technology.

3 Methodology for a Scalable Service Search Engine

Previous work in the area of Semantic Web Services [6, 8, 9, 10] has focused mainly on provide means to describe the functionality of a Web Service in the most accurate way and to allow a very expressive language for querying services. However none of the papers discusses in depth how the description are created, but assume that the burden is on the service publisher. While in an Intranet setting it might be possible to enforce some formalism for publicly available services this seems unrealistic. For this reason we do not focus on generating semantic descriptions for Web Services as means to share a good characterization of a given service, but use them internally in our prototype in order to integrate the information from the various sources into a coherent semantic model.

Based on the empirical studies described in the previous section, we have developed a Web Service search engine. It is based on the observations that centrally maintained repositories can not scale in a Web context and that naive keyword search on a document level does not allow an efficient service discovery. Thus we base our approach on (1) a high level of automation and (2) the extraction of semantics from existing information.

In order to achieve this we perform the following steps:

- A focused crawl of the Web aiming to obtain service descriptions.
- Analysis of the WSDL documents, especially features around the endpoint.
- Correlate related Information with each service.
- Analysis explicit and implicit user behavior gathered while interacting with the search engine.

We start of by crawling the Web for WSDL files. Although a WSDL document is only a technical interface specification it is the basis for our analysis. From our experience more then 40% of all the WSDLs include textual documentation. And even if a WSDL file has no textual documentation it still contains very valuable information, e.g. method names such as "SendSMSToIndia" with inputs like "FromEmailAddress" "MobileNumber". However one needs to consider the particularities of a WSDL document. First words in different positions of the document have very different meaning. If "fax" is part of the service name (or port type) it has a different relevance as if it occurs only somewhere within the XML schema as part of the input information.

There is more information implicitly hidden: If a particular host has more then just one services this increases the chances that it is a professional provider and probably more relevant then others. Then of course the availability and response times of a particular endpoint is an important information that can be added as a ranking criteria. Moreover knowing the IP address of a service we can deduce its location using a

IP2Geo database. While this is no guarantee for reliable information about the actual geographic coverage of a service it can at least provide a hint. With a similar analysis we can generate a semantic annotation about the nature of a service (e.g. if it is commercial or academic can be deduced based on the owner organization of a particular IP range).

This way we can automatically generate initial models for the services in a completely automated fashion. In a next step we are refining these models using the implicit and explicit feedback from users of the search engine. As opposed to many rich semantic approaches which take the premise that service selection will be done automatically, we believe for years to come service selection will be a task carried out by a human. Thus we believe in an interface similar to current search engines. Given this we can observe the behavior of users. E.g if within one session a search for "fax" and "messaging" and in another "payment", "creditcard", "fraud" is performed, we can group those words, use them to broaden search request as well as to cluster services and users.

4 Implementation

We have implemented the methodology described above. Our prototype is available at: search.aleph-webservices.com.

Within our initial experiments we could gather more then 8000 services. Of which for many we could already create descriptions going beyond the features offered only by WSDL.

Crawling For performing a focused crawl we have adopted the Heritrix Web crawler⁶ by adding special scoping rules limiting the crawl to relevant portions of the Internet. With our initial set of seed URLs as base we could gather around 13000 WSDL files. The seed URLs itself have been collected in a semi automatic process that involved screening well known sites mentioned in the section on existing discovery approaches and in utilizing the Alexa Web Search platform⁷.

Analysis In the analysis step we removed duplicates and WSDLs that did not include any valid endpoint definition. Given the 13000 URLs that resolved to valid WSDL documents only 8307 distinct services could be identified. Considering interface (porttype) together with specific endpoint URL as differentiating criteria.

For allowing in a first stage keyword based search, we implemented our tokenizer and lexical analyzer on top of the tsearch2 library⁸. For example we added to the tokenizer specific routines for taking care of the CamelCasing often used by developers (such as "WeatherForecastService", that is split into it tokens). Then words have been reduced to lexemes by their respective dictionaries. That is e.g. the token "faxes" is reduced to its linguistic stem, i.e. "fax". In order to provide ranked results for a particular keyword we assigned each keyword one of four different weights depending on its position within the document.

⁶<http://crawler.archive.org/>

⁷<http://www.alexa.com/>

⁸<http://www.sai.msu.su/~megera/postgres/gist/tsearch/V2/>

Regarding the technical properties we so far monitored the response time, and if a particular endpoint does adhere to the protocols it claims to support. Moreover we have included the results of an IP to geographic database mapping. A further ranking criteria that we have included is the number of known Web pages that point to a service description. While this characteristic has in our opinion not the same relevance as for static content it still provides a reasonable indicator for relevance. Figure 1 shows a screen shot of our search engine.



Figure 1: Search Engine Screenshot

Future Work Since we just started we have only limited means to capture user behavior. However we record search terms together with which services have been looked at and which providers are selected. We will extend this with more community features in future that also allow direct ranking and tagging by the users.

On the analyzing site we will extend our work to include history information of a service and also more related information. Such as pricing information present on the provider Web sites or comments with Blog or Forums.

5 Conclusion

The standard model of syntactic search engines like Google alone is not well suited for Web service discovery. Neither the identification of potential services through key word extraction nor the relevance ranking based on hyperlinks provide much of a use in a Web service scenario. At the same time approaches that have aimed at a complete automation respectively focused only on a machine2machine interface such as

UDDI or several works in the area of Semantic Web Services are also not suitable for a heterogeneous and open Web environment.

Our approach is to post process interface description of services (WSDL documents) and to build up semantic models of services by applying heuristic analysis that are verified in a community process. While we are at the beginning of our work our prototype shows that existing solutions can not meet the already achieved combination of broad coverage together with a relative accuracy in the retrieval.

Acknowledgments The work is supported by the European Commission under the projects Knowledge Web and DIP; by the FIT-IT (Forschung, Innovation, Technologie - Informationstechnologie) under the projects RW2 and TSC.

References

- [1] Daniel Bachlechner, Katharina Siorpaes, Holger Lausen, and Dieter Fensel. Web service discovery a reality check. In *3rd European Semantic Web Conference*, 2006.
- [2] T. Bellwood, L. Clément, D. Ehnebuske, A. Hately, Maryann Hondo, Y.L. Husband, K. Januszewski, S. Lee, B. McKee, J. Munter, and C. von Riegen. Uddi version 3.0. <http://uddi.org/pubs/uddi-v3.00-published-20020719.htm>, July 2002.
- [3] E. Christensen, F. Curbera, G. Meredith, and S. Weerawarana. Web services description language (wsdl) 1.1. <http://www.w3.org/TR/wsdl>, March 2001.
- [4] Xin Dong, Alon Y. Halevy, Jayant Madhavan, Ema Nemes, and Jun Zhang. Similarity search for web services. In *VLDB*, pages 372–383, 2004.
- [5] Jianchun Fan and Subbarao Kambhampati. A snapshot of public web services. *SIGMOD Rec.*, 34(1):24–32, 2005.
- [6] J. Gonzalez-Castillo, D. Trastour, and C. Bartolini. Description logics for match-making of services. In *KI-2001 Workshop on Applications of Description Logics*, September 2001.
- [7] Su Myeon Kim and Marcel-Catalin Rosu. A survey of public web services. August 2004.
- [8] L. Li and I Horrocks. A software framework for matchmaking based on semantic web technology. In *12th International Conference on the World Wide Web*, Budapest, Hungary, May 2003.
- [9] M. Paolucci, T. Kawamura, T. Payne, and K. Sycara. Semantic matching of web services capabilities. In *1st Int. Semantic Web Conference (ISWC)*, pages 333–347. Springer Verlag, 2002.
- [10] K. Sycara, S. Widoff, M. Klusch, and J. Lu. Larks: Dynamic matchmaking among heterogeneous software agents in cyberspace. In *Autonomous Agents and Multi-Agent Systems*, pages 173–203, 2002.