Working Papers

Working Paper No. 5

Web Service Discovery – Reality Check 2.0

Stephan Hagemann, Carolin Letz, Gottfried Vossen

ISSN 1614-7448

cite as: Hagemann, S.; Letz, C.; Vossen, G.: Web Service Discovery – Reality Check 2.0. In: Working Papers, European Research Center for Information Systems No. 5. Eds.: Becker, J. et al. Münster 2007.

Table of Contents

List of Figures

List of Tables

Working Paper Sketch

Type

Research Report

Title

Web Service Discovery – Reality Check 2.0

Authors

Gottfried Vossen is professor of Information Systems and Computer Science at the University of Muenster in Germany. He is the European Editor-in-Chief of Elseviers *Information Systems - An International Journal*. Carolin Letz and Stephan Hagemann are PhD students in Prof. Vossen's research group focused on Web services and Web technologies. Contact via email {vossen, carolin.letz, stephan.hagemann}@uni-muenster.de.

Abstract

In practice the ability to find the right Web service decides between a functionality being implemented anew and at least the possibility of executing it via a service. This report evaluates existing public portals for Web service discovery with respect to their characteristics and their acceptance by developers. For this, we distinguish different possible settings and use cases and evaluate how these are supported in practice. Only few of the publicly available Web service registries are growing in size and importance, with the use case best supported being the pre-programming phase of evaluation of the service landscape.

Keywords

Web services, Web service discovery, UDDI, public service directories, Web 2.0

1 Introduction

Web services (WS) and their standards have matured over the last couple of years. Their importance as realizing components for service-oriented architectures (SOAs) has increased as well [DJ+05, WCL+05]. A key driver for SOA implementations is the hope to save development time and costs through a higher degree of component reuse in the form of readily implemented services [LLG+03, BH06]. To achieve this goal it is necessary, among other things, to make Web services discoverable. This paper presents a reality check on service discovery which extends a previous such check undertaken in [BSF+06].

Different theoretical approaches, ranging from artificial intelligence and automated reasoning to automata theory, have focused on certain aspects of reasoning about Web service discovery, comparing the expressiveness of Web service models and examining the complexity of service matching [e. g., GTB01, PKP+02, LH03, DSV04]. These are in contrast to the current practical state of Web service discovery on the public Internet where much less sophisticated techniques are employed, an apparent clash that has been observed before.

At the end of 2005 three groups of approaches to Web service discovery were identified and analyzed in [BSF+06]. The comparison includes:

- the Web service standard approach, the *UDDI business registry*,
- specialized *Web service search engines and portals*, and
- *general-purpose search engines*.

Features taken into account include the search functionality itself, the number of available services, the availability of discovered services, the choice of interfaces, the availability of a service rating, of test and demo functionality, of a WSDL parser, and of information about service costs.

The year 2006 has seen a dramatic rise of public interest in *Web 2.0* applications [VH07]. A common interpretation of this buzz-word is the shift from the Web of *information* towards the Web as a *platform* [MO07]. This trend was not noticed in [BSF+06]. Therefore, just a year later, there is a need to redo this type of study. However, we do not merely take a snapshot of the current state of Web service discovery on the public Internet but also uncover trends and tendencies in the events of the past one and a half years. Consequently, the criteria for comparison are extended to put a special emphasis on the opportunities Web 2.0 developments have to offer. Especially the explicit support for communities with eased user feedback and interaction mechanisms, end user orientation, and simple categorization approaches based on tagging are taken into account.

The paper is organized as follows: In Section 2, the foundations of Web services and their discovery are summarized, where the focus is on practical techniques for discovery as relevant for this study. Theoretical models are not taken into account. Furthermore, the foundations of Web 2.0 and the support for information retrieval in Web 2.0 are presented. In Section 3, different practical approaches towards Web service discovery on the public Internet are examined and updates compared to the findings in [BSF+06] are given. In Section 4, the different approaches are compared and opportunities for Web 2.0 techniques are pointed out, that help to enhance Web service search engines. In Section 5, the paper concludes with an outlook on future research opportunities that combine Web service discovery and Web 2.0.

2 Fundamentals

In this section the foundations of Web service search and discovery as well as of Web 2.0 are briefly introduced as far as they are relevant for the following analysis.

Web service standards have emerged in a bottom-up fashion from low level communication protocols to higher level standards, such as quality of service descriptions [Vos06]. From the very beginning, the emergence of Web services has been accompanied by the attempt to establish standards that are accepted by all major software vendors to ensure interoperability of possibly world-wide available Web services. A key success factor for this idea is the discoverability of Web services.

A registry approach has been chosen as solution. Before introducing the current standard UDDI [CHR+04] that has been invented for such a Web service registry and discussing possible alternatives in Section 2.3 we will take a step backwards to examine the different scopes for which a Web service registry might be used in Section 2.1 and to look at different use cases for Web services registries in SOAs in general in Section 2.2.

2.1 Web Service Registry Scenario Alternatives

Different scenarios for Web service registries have been described by the OASIS standardization committee [TSG02], by software vendors such as IBM [Gra01] or Microsoft [MSC03a, MSC03b], and by software designers and researchers [DJ+05]. The following summarizes and consolidates these scenarios. The scenarios differ in their scope of application, which we will introduce here. This will later serve as a guideline for evaluating which scopes are supported by publicly available Web service registries.

2.1.1 Universal Business Registry (UBR)

The first and most general use case is to operate a registry as a global domain independent business registry, the UBR. In this scenario it is used as public registry service, open to every service provider and consumer, and free of charge. It functions as registry and as search engine for establishing B2B contacts in an automated way [DJ+05, TSG02, Gra01]. A few such central registry nodes have been operated world-wide by a few international companies, these were interconnected as depicted in Figure 1.

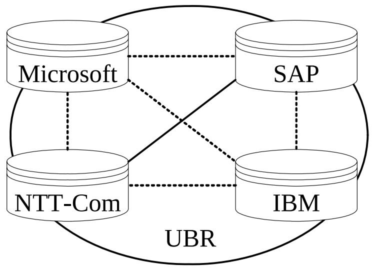

Figure 1: Schema of domain-specific UBR according to [DJ+05].

2.1.2 Domain Specific Business Registry

Instead of offering a single global registry for every domain it is also possible to establish specialized registries that contain entries related to only one industry branch [Gra01]. Such registries need to establish usage rules that must be adhered to by all users to ensure the quality of the entries and to avoid unwanted entries that are not domain related. Therefore, such a registry needs an administration that supervises the users [DJ+05]. This usage scenario is depicted in Figure 2.

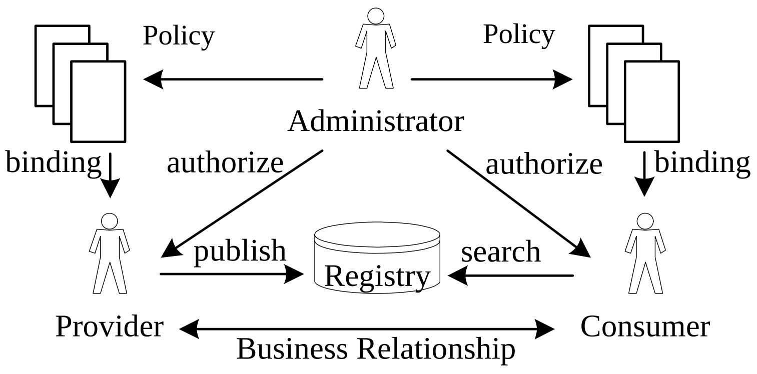

Figure 2 Schema of domain-specific Web Service registry according to [DJ+05].

In such a domain specific environment it is also possible to claim a membership fee, to rate the services and to use established domain specific taxonomies for service search.

2.1.3 Intranet Service Registry

Registries are also set up within the Intranet of enterprises [DJ+05, Gra01]. Figure 3 summarizes this scenario. Services of one department can be offered to all organizational units. Application designers use the registry to search for available services and as reference list that does not only contain the services but that also helps finding the related technical documentation [MSC03a]. Further, services within the registry can be approved of by the quality manager to be used company-wide. This approval process may also be extended to include external services that the application developers are allowed to use. Thus, the internal Web service registry helps to manage the software development process and the quality of services [DJ+05].

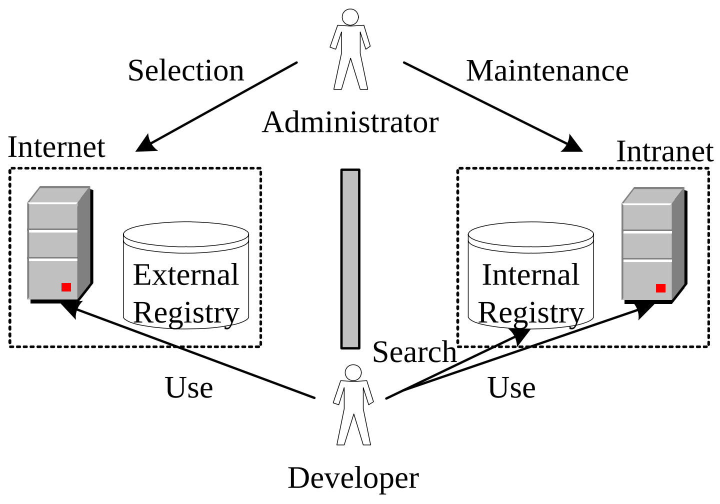

Figure 3: Schema of internal Web Service registry according to [DJ+05].

2.1.4 B2B and EAI Service Registry

The Intranet usage scenario can be extended such that the company in question does not only use its own and external services, but for some services also acts as an external service provider towards trading partners and customers [TSG02, MSC03a, Gra01]. In this scenario it is necessary to ensure that only applicable services available on the Intranet registry are made public. Only a selection of services is suitable for external use and must be carefully chosen by an administrator [DJ+05]. This is depicted in Figure 4.

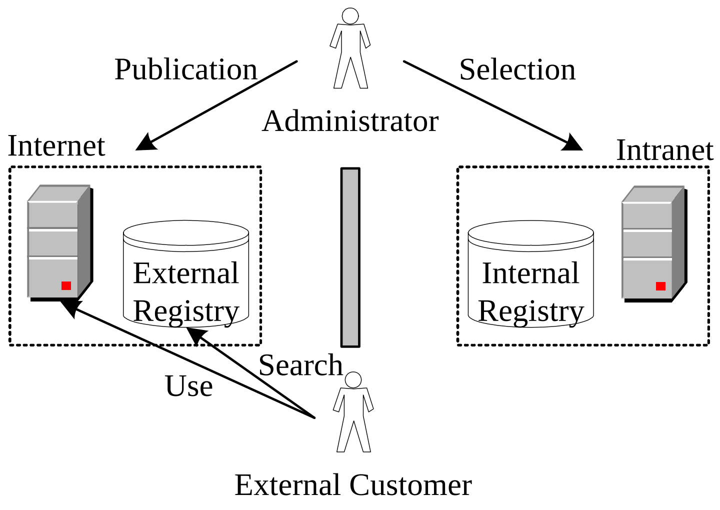

Figure 4: Schema of a B2B UDDI according to [DJ+05].

Note that only in the first scenario a globally known access point to the registry is assumed. All the other use cases rely on domain specific or enterprise specific access points that need not be published in a global directory. Instead, it is assumed that users know where the registry is located and that the access to it is restricted.

2.2 Web Service Search Use Cases

A Web service registry may need to support various search patterns, the search by the human developer at design time of an application and the search for services of an application at run time.

2.2.1 Design Time Search

At design time, a Web service registry can be used in a combination of browse and drill-down search patterns. The application designer calls a *find*-function, e. g., *find_service* of the registry inquiry API and gets a result list back. This list can then be inspected manually or refined through a more specific search. To inspect a particular entry in the result list, a *get_Detail*-function is needed, e. g., *get_ServiceDetail*. Finally, the designer can invoke the service. This sequence of browse, drill-down, and invocation patterns as supported by the UDDI standard is depicted in Figure 5, which we will cover extensively in Section 2.3.1. Most often, development tools support this exploratory search pattern with a GUI. The results of the search at design time can be used to bind the service invocation statically into the application.

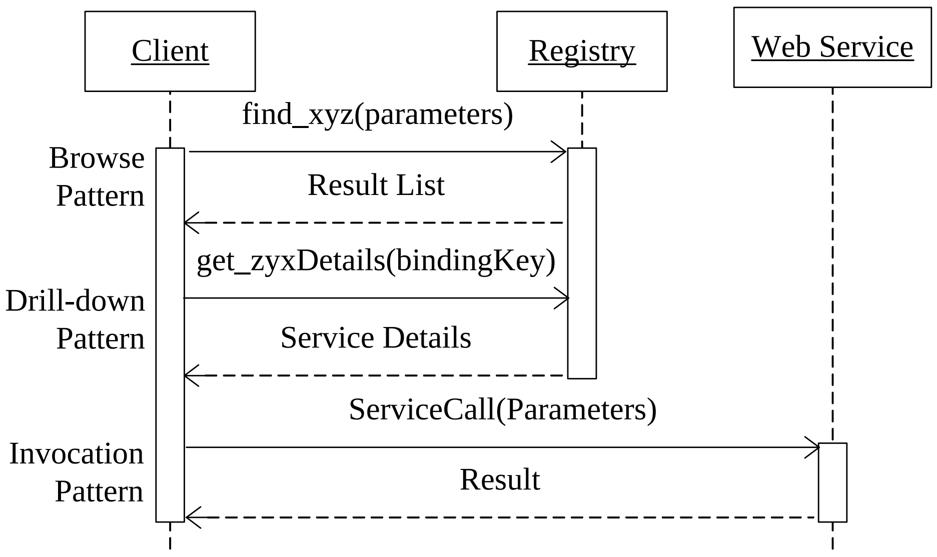

Figure 5: Patterns supported by a registry inquiry API [DJ+05].

2.2.2 Run Time Search

Besides static binding the search functionality of a registry can also be used for dynamic binding. We will speak of *dynamic binding* if the client application calls a service interface as defined by an existing template at run time. Such a template is called *tModel* in the UDDI standard. The service template can be implemented by more than one Web service. Thus, the interface and the binding details are known to the requestor a priori. Only the exact service and its provider are chosen at run time as shown in Figure 6. [ACK+04] consider this to be a realistic scenario for development of service-oriented applications.

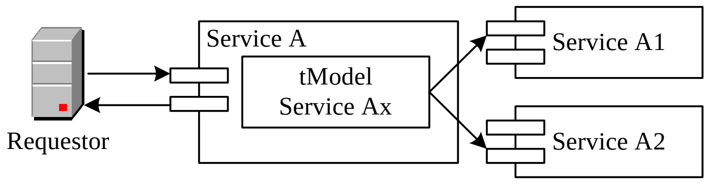

Figure 6: Dynamic service call via tModel.

A variation of dynamic binding is suggested by [Ley03, WCL+05]. It uses a specialized infrastructure, a *service bus*, for service selection. The service requestor sends a search request to the service bus and describes the needed service by defining business goals and policies. All matching services are considered to be equivalent from the requestor's point of view. They are represented by a *virtual service*. The service bus chooses an appropriate service, binds to it, invokes it, and delivers the result to the requestor. The choice the service bus makes is guided by further policies such as execution costs or availability. This scenario is depicted in Figure 7. The service bus is an infrastructure component that is delivered by software vendors, for example, IBM WebSphere[1] contains a service bus.

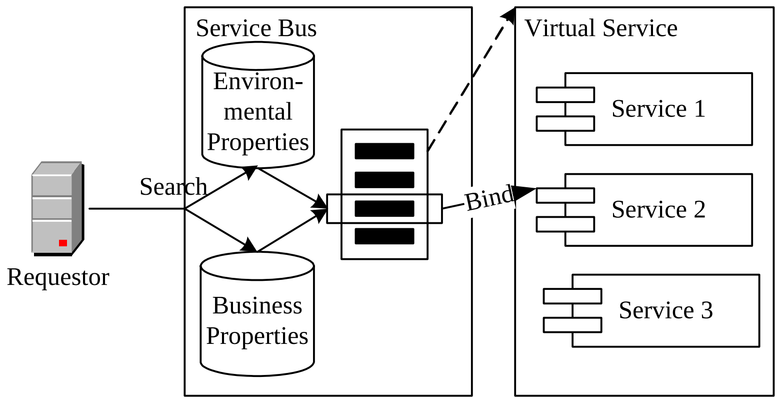

Figure 7: Dynamic service selection via service bus according to [Ley03].

With an even higher degree of dynamics at run time the application searches for a service it has never heard of before that is offered by a provider that is unknown to the consuming application. The application chooses a suitable service among the results returned by the registry and binds to it on the fly. This admittedly visionary scenario is currently considered to be unrealistic for service-oriented application development. The behavior of such an application would be unpredictable. It is unknown which parameters need to be passed. The application cannot be thoroughly tested, it is undefined how to react if an exception is returned, and the service consumer does not even have a proper service level agreement with the service provider. [ACK+04] doubt that application designers will ever have use for such a degree of dynamics.

2.3 Implementations for Web Service Registries

So far, Web service registry usage scenarios and application use cases have been considered. Next, different implementations are presented. One of the core Web service standards is UDDI

1 http://www-306.ibm.com/software/websphere/

[CHR+04]. It describes how to implement and interact with a service registry. The standard, its possible use cases, interaction patterns and alternative standardizations efforts are presented in the following sections.

2.3.1 The UDDI Standard

UDDI[2] is an XML-based Web service standard that describes how to implement and interact with a service registry. It is a framework that describes which data structures and APIs a Web service registry must implement and offer to support Web service publication and search.

The information that UDDI offers is often compared to the *white* and *yellow pages* of a telephone directory. White pages list companies and organizations, their contact information, the services they offer and a short human readable description. Yellow pages categorize service providers and services according to classification schemes or taxonomies. Additionally, there are the *green pages* that contain technical information on services and their invocation.

To store these different kinds of information, UDDI uses data structures that are schematically depicted in Figure 8. Each information type, *businessEntity*, *businessService*, *bindingTemplate* and *tModel*, has a unique key. A *businessEntity* can contain one or more *businessServices*, a *businessService* comprises a list of one or more *bindingTemplates*. All three information types may contain references to one or more *tModels*. *tModels* may be referenced by more than one UDDI entry.

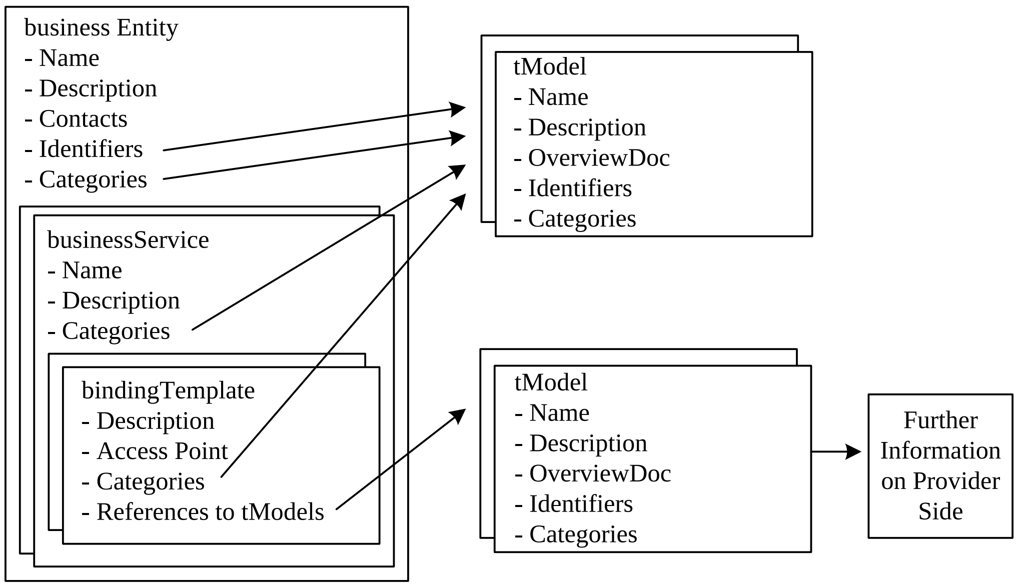

Figure 8: Schematic overview of UDDI data structure according to [ACK+04].

- The **businessEntity** (white pages) contains information about the service provider such as the name of the business, a short textual description, contact information (contact person name, telephone number or e-mail address), business identifiers such as a tax ID or an ID

2 At the time of writing the current version of UDDI is 3.0.2. The current version can always be found at http://www.oasis-open.org/committees/uddi-spec/doc/tcspecs.htm.

based on another commonly accepted identification scheme, and a list of categories that describe aspects of the business entity such as industry branch, product category or geographic region. To define which kind of identification scheme or category system is used pointers to further technical documentations, the *tModels*, can be included.

- The **businessService** (yellow pages) groups related Web services of a *businessEntity*. Usually, it will only contain one Web service that is available under different addresses, in different versions or under different protocol bindings. The *businessService* is described by its name, a short textual description, and a list of categories that can be further explained by references to *tModels*.

- The **bindingTemplate** (green pages) discloses the technical details to use a particular Web service. This is essentially the access point to the Web service, a short textual description and references to *tModels* that contain further technical information.

- The **tModel** is a generic container for any kind of technical model, e. g., a WSDL file or a taxonomy definition. They can be referenced by multiple Web services and, thus, be reused. A *tModel* consists of a name, a human readable description, references to further documentation (e. g., text documents), identifiers that conform to an identification scheme, and categories that follow a given classification.

UDDI itself offers different APIs that are themselves described as Web service interfaces. Therefore, a UDDI is a special kind of Web service. The UDDI specification defines the following APIs:

- The **Inquiry API** allows finding registry entries that fulfill given search criteria (*find_business*, *find_service*, *find_binding*, *find_tModel*) and getting details about a specific entry (*get_businessDetail*, *get_ServiceDetail*, *get_bindingDetail*, *get_tModelDetail*).

- The **Publisher API** is intended for service providers who want to add, or modify (*save_business*, *save_service*, *save_binding*, *save_tModel*), and delete (*delete_business*, *delete_service*, *delete_binding*, *delete_tModel*) registry entries.

- The **Security API** provides functionality to authenticate and authorize users based on security tokens.

- The **Custody and Ownership API** enables the UDDI to transfer the custody and ownership to another publisher.

- The **Subscription API** offers functionality to track changes within the registry, e. g., new, modified, and deleted entries.

- The **Replication API** supports the synchronization of distributed UDDIs.

Every UDDI provider must offer the Inquiry API and the Publisher API. All other APIs are optional.

2.3.2 Additional Standards

There are additional standards that address the Web service search problem in absence of a central UDDI.

WS-Inspection [BBM+01] is an alternative specification in addition to UDDI. The UDDI standard builds on centralized service registries that are in general open to every kind of business. The UBR (see Section 2.1.1) as operated by IBM, Microsoft, SAP and NTT-Communications has shown that this is possible. In parallel, other ways of publishing Web services have emerged, e. g., Amazon Web services are published directly on the provider's Web side. WS-Inspection addresses how to publish Web services in a decentralized but still structured manner. Thus, clients can search the Web services of providers they trust. An overview of the different strategies of UDDI and WS-Inspection is given in Figure 9.

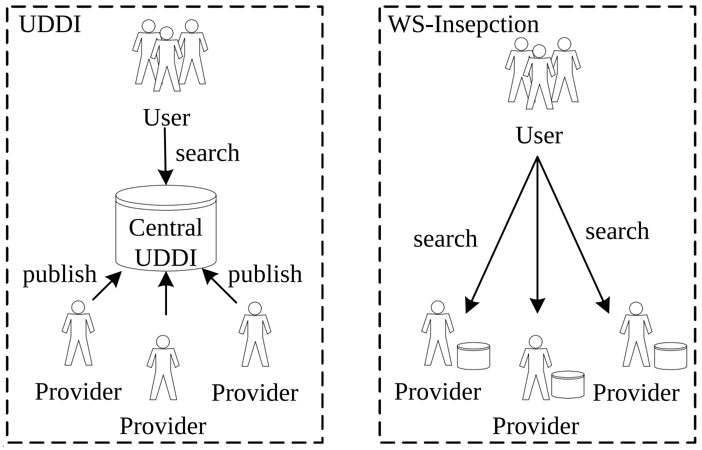

Figure 9: UDDI versus WS-Inspection according to [DJ+05].

WS-Inspection is document-based: The service provider publishes a document with the name “inspection.wsil” in the home directory of the Web server. This document contains a short human-readable abstract, an arbitrary number of service elements and an arbitrary number of link elements. Service elements consist of a short human-readable abstract, a name and a description that contains technical information such as the namespace and the location of the WSDL document that defines the service interface. Link elements consist of short human-readable abstract and technical information, e. g., the namespace and the location of another WS-Inspection document.

The service consumer reads the “inspection.wsil” document via HTTP and gets a list of all available service and their WSDL files. The WS-Inspection standard is extensible, and there are already pre-defined extensions to include UDDI and WSDL. The link elements permit to build hierarchies of WS-Inspection documents.

WS-Dynamic Discovery [BKK+05] is a specification that defines a multicast discovery protocol to find services in ad-hoc networks where Domain Name Services or directory services do not exist. The protocol is not intended for Internet wide usage. It does not define any data model nor any queries over rich meta-data. If a service consumer wants to find a service on the network it sends a probe message to a multicast group. Matching services respond directly to the service consumer. The service consumer then sends a resolution request message to the same group and the matching service replies directly to the service consumer. If new services join the network they send an announcement to the group to minimize polling. If a discovery proxy joins the network the multicast discovery protocol is automatically abandoned to use the proxy specific protocol. This protocol is suggested to be used for, e. g., discovery of low level resources such as printers within a network.

Electronic business XML (ebXML)[3] is a standardization initiative from the e-business domain that started in 1999 to promote an open XML-based infrastructure for world-wide, interoperable, secure and consistent exchange of information about electronic business. It supports trade between companies of different sizes and from different geographic areas in a cheap and efficient way by offering open XML standards for business-to-business (B2B) and business-to-customer (B2C) transactions that replace proprietary, vertical solutions by a horizontal solution spanning industry and trade. The standard consists of six specifications as depicted in Figure 10.

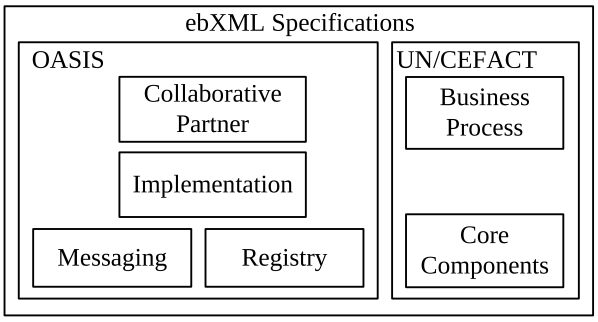

Figure 10: ebXML specifications.

These specifications are compatible with the basic Web service standards WSDL, SOAP, and UDDI and define the following standards for e-business [OAS06]:

- The **Messaging Service** defines a communication-protocol neutral method to exchange e-business messages over SOAP 1.1. It is compliant with WS-Security, supports reliable messaging and digital signatures.

- The **Collaboration Protocol Profile and Agreement** defines a bridge between technical capabilities and partner expectations for business collaborations. It contains technical capabilities of a business partner and the capabilities and preferences of protocol aspects and properties for specific roles in component services and activities used in processes. It enables monitoring of sessions and verification of delivery channel features. The agreements contain data to configure shared aspects of business collaboration protocols.

- The **Business Process Specification Schema** defines a standard language to describe interoperable business processes. It allows defining transaction patterns, partner roles, documents and it helps to compose and monitor processes. The business process can also involve Web services.

---

3 www.ebxml.org

- The **Registry Service** supports registering, locating, and accessing information resources in a distributed environment.
- The **Core Components Technical Specification** presents a methodology for developing semantic building blocks to represent the general business data types that are frequently used. This enables reusable and commonly understandable concepts and data models. It defines a fixed set of data types and naming conventions for consistent business representation.

These standards are especially helpful for Web service detection in two ways: First, ebXML registries can be regarded as Web services and can therefore be registered in UDDI registries, thus, becoming available like any other Web service [ebX01]. Second, ebXML provides a framework to define core components for e-business that can be re-used in any context, such as the naming of data types in a WSDL document. Thus, the core components help to establish naming conventions and semantics for WSDL documents and UDDI entries [WPA+01].

2.3.3 Web 2.0 Information Sharing

The interest in Web 2.0 [VH07] has risen ever since the term got wide attention after an article by Tim O'Reilly [Ore05]. Its core has been defined as the read/write Web that is used by people to create greater benefit for all [Hin06], also subsumed under "Harnessing Collective Intelligence" [MuOr07]. A strong convergence is seen between principles from SOA and Web 2.0 [Hin07]. The differences between the two are important to understand the different attitudes these approaches to service-orientation have.

SOA places much emphasis on a controlled environment, employing heavy-weight standards, of which we have discussed UDDI in depth in the previous sections, but which has led to what is called the Web services stack [Vos06]. This stack places standards and protocols first, and aims at reusing code in the form of services. It does not in the first place deal with the organization, the developer, or the user. Web 2.0, on the contrary, seems to neglect most of these standards and instead employs light-weight models based on HTTP and the REST architecture style [Fie00]. It is inherently user-oriented, as its values and patterns are said to be "the basis for the next generation of the Internet" [MuOr07], which implies that they are steering the way applications on the Web are developed.

An example of a technology pattern that is commonly applied in the Web 2.0 context is Ajax [CPJ06]. Ajax leverages existing Web technologies, namely XML, JavaScript, and XMLHttpRequest, by combining them in a meaningful way that can make Web pages with Ajax more responsive, interaction with them more fluid and logical than those without. It marks a point of departure from the page-based Web without (necessarily) breaking the structure of the Web. Thus, Ajax enables Web sites to become application providers, delivering software as a service, without making these unusable, because of the limitations of the browser. Essentially, the range of tasks that the Web can efficiently support has grown with Ajax. Examples, include online mail clients (e. g., mail.google.com and mail.yahoo.com), Web office suites (e. g. Zoho and ThinkFree), and mapping applications (e. g., maps.google.com and maps.yahoo.com).

At another end, many Web pages have started to add community features to their service portfolio, with the successful sites employing this for leveraging the value of the entire site. The most prominent examples include Amazon, Delicious, Flickr, and MySpace. While for some of these the community is everything, for others it is an essential feature, around some service. A common way to exploit the community on a Web site is through tagging. Different characteristics of tagging systems have been identified in the literature [MNBD06], which share that they allow users to classify resources by freely chosen keywords, known as tags. The important difference when compared to a hierarchical classification scheme is that no a priori consensus is necessary; at best consensus on certain tags emerges over time. Then tags form a flat namespace called a *folksonomy*. A folksonomy can be searched, thus supporting user-created access-patterns to the resources of a site. Users of such sites have different motivations for providing tags. One source of motivation stems from the ability to organize the resources for personal use. The community aspect of tagging thus comes as a byproduct for free.

Tagging represents a way in which users of Web sites share information, which allows reuse by others. This is similar in spirit to Web sites providing access to their data and services (the prominent examples being the same as above). This also allows for reuse by others, which is precisely the original SOA idea. In the context of Web 2.0, this has seen a growing interest under the name of *mash-up*. These combine typically freely available services to generate an added-value. Of course, mash-ups are similar to composed applications, which is the term used in SOA.

From our discussion above, it is fair to say that Web 2.0 and SOA are essentially concerned with the same topic, namely information and software reuse. As we will see, they are, however, working at different ends of the spectrum.

3 Service Discovery on the Public Internet - State of the Art

One and a half years after the survey reported in [BSF+06] has been published the various sites are revisited in this section to compare them anew. In the following we will only mention URLs if they deviate from the pattern “www.name-of-service.com”. The comparison includes: the UBR, Xmethods (www.xmethods.net), WebserviceX (www.webservicex.net), WebserviceList, StrikeIron, RemoteMethods, Woogle [DHM+04], WSDLicious (wsdlicio.us), ProgrammableWeb, as well as general purpose search engines.

All services have been visited between the May 5th and 8th, 2007. If they were unavailable during a first attempt they were revisited again later. If they were still unavailable they were considered to be obsolete.

The list of comparison criteria as presented in Section 3.1 is enriched to include the latest Web 2.0 aspects. The universal business registry is presented in Section 3.2. Specialized search engines and catalogues are presented in Section 3.3. Especially in this section the changes that have taken place since the survey of [BSF+06] become apparent. General purpose search engines are examined in Section 3.4.

3.1 Criteria for Comparison

In this section comparison categories are introduced that are later used to compare the different Web service discovery approaches. The first group of criteria is constituted of basic features that are needed for searching and gathering information about the available services in general. The second group of criteria consists of features that describe the quality of service and ease of use that the Web service search offers. Such criteria often influence the user decision and bind the user to a specific search implementation. Groups one and two resume the comparison in [BSF+06] to reveal trends and tendencies.

The third group of features newly includes Web 2.0 aspects that are considered to be useful for Web service search but which may not always be employed yet. This group of features is intended to reveal new opportunities that Web 2.0 has to offer for Web service discovery. This group of features is not only derived by observation of current Web service search engine implementations but also by the analysis of other popular Web 2.0 applications that support searching for different types of information.

Basic features of Web service discovery include the search functionality itself, the number of services available through a specific search approach, the availability of status information and the possibility to discover different kinds of technical interfaces to one service.

- **Registered information**: Most search engines allow registering the service name, the provider name and a description of the service. Other search engines also include information about individual operations and user comments.

- **Search functionality**: This is the central functionality offered by a Web service search engine or catalogue. Catalogue browsing and keyword search are the most common types of search support that sustain a browse and drill-down search pattern. The search functionality is also enhanced by service descriptions and a documentation of the individual service operations.

- **Number of Services**: This is an indicator for the popularity of the search engine and the likelihood of finding services. Scarcely populated catalogues are less likely to contain the service sought than large catalogues.

- **Service availability**: If the search engine does not only contain the link to the service but is also able to provide the information if the service is available this helps to filter out service that are down or that are no longer maintained.

- **Interface support**: WSDL interfaces are the standard interfaces that Web service search engines provide. Support for additional formats such as RSS or WS-Inspection documents enhance the search capability.

Quality of service features of Web service discovery encompass additional functionality such as test or demo functions and a WSDL parser as well as information about service costs. These additional features help the user in making a choice.

- **Test functionality**: An additional feature is the availability of a test function that allows calling the Web service operation with demo or dummy values to evaluate the behavior by observation. This helps the programmer to determine availability and compatibility with application needs without having to implement the interface.

- **WSDL parser**: A WSDL parser allows browsing the WSDL interface description in a user friendly and comfortable way to find out more about the Web service.

- **Information on terms of usage**: Additional information that is usually not used for searching in the first place is the terms of usage, including service costs. This is also an important criterion for the end user to decide which service to choose if several are available that offer similar functionality.

Under **Web 2.0 features for Web service discovery** we subsume those aspects that have of late been attributed to the Web 2.0 approaches and techniques. We consider all forms of community support, as well as information on service usage.

- **Community support**: This can come in many forms, which all have in common that they support some form of social interaction. User feedback comes in the form of comments and ratings. User interaction is supported in forums and wikis.

- **Service usage**: The combined use of services has come into focus in the context of Web 2.0 under the name of *mash-ups*. As an aspect of a service repository this is interesting as it puts the usage of a service into focus. If a portal shows mash-ups next to services, these can be considered as running usage examples, popularity measurements, and technology showcases.

3.2 Universal Business Registry

The universal business registry implemented the use case of a UDDI as global registry until January 2006. It was intended as a proof of concept by the initiating companies of the UDDI standard, IBM, Microsoft, SAP and NTT-Communications. They operated a global UDDI distributed onto four nodes for five years as shown in Figure 1. The UBR was shut down in January 2006 because all participating companies regarded the proof of concept as successful and the UDDI standard version 3 as stable. Since then, companies have started operating separate UDDIs, e. g., uddi.sap.com.

This registry requires a user account for service registration and viewing WSDL files. The search functionality is available without registration. It allows keyword based multi-language search with wildcards for tModels, Business Entities, Business Services and Binding Templates. Further, the same elements can be searched using a categorization-based browsing functionality.

There is neither test nor user feed-back functionality available. Further, the user does not get any information about the service costs or terms of use. The user is only able to see the information as defined by the UDDI data structures. During the survey, the SAP UDDI sometimes did not respond.

Although the UBR existed for several years, the idea of a global business registry has been abandoned. Therefore, the UBR is considered to be obsolete.

3.3 Specialized Search Engines and Catalogues

A variety of search engines and catalogues have emerged that collect available Web services from other sides or that rely on manual registration by service providers. Eight currently available search engines and catalogues are briefly described to show which of the above comparison criteria they are able to fulfill.

3.3.1 XMethods

XMethods (www.xmethods.net) is an online Web service catalogue maintained by XMethods, Inc. that currently lists around 500 Web services.

Entries in this portal correspond to one SOAP binding defined in the WSDL file of a service provider. Thus, several entries can be present for a single service. As an approximation, we have nevertheless counted all entries as separate services for our comparison. The details provided for an entry cover a link to the WSDL file and the possibility to analyze the WSDL, which currently only shows how many operations are present. Attempts to drill-down to more detailed information seem to only result in links to the original WSDL file. Additional information includes the unique key, the owner with contact information, a link for more information, a short and a detailed description, and usage notes.

There are two features that stand out, the first being the *contributed clients* section, which may list programming libraries that access the service and function as proxies for specific programming languages, full applications, and example code. Anyone can add clients after registration. However, the number of services actually containing such clients is very low. The other feature is the possibility to try out a service. For this the SOAPscope service of Mindereef (www.mindreef.net) is used, which allows building requests, sending them, and reviewing the intermittent documents.

The data of XMethods can be accessed through a variety of means in addition to the Web page. There are SOAP, UDDI, WS-Inspection, DISCO, and RSS feed documents available. While the first four of these allow accessing the entire database, the RSS feed always lists the last ten entries added.

3.3.2 WebserviceX

WebserviceX (www.webservicex.net) is an online Web service catalogue maintained by Generic Objects Technologies Ltd and lists currently round about 70 Web services.

Every service is described with a brief summary; the endpoint is given as well as the URI of the DISCO and the WSDL document. Further it is possible to display the WSDL file. Users can invoke the Web service in a test mode and get a sample description of the exchanged HTTP GET, POST and SOAP messages. However, not all services provide such a test mode. Those that do offer a test mode are not always straight forward to use, e. g., a power conversion service that converts power values from one unit system to another does not tell the user how to specify the power systems in the input fields for testing.

WebserviceX supports search by category browsing. A keyword based search is not possible. Recently published services and some top Web services are listed as short cuts. It is not clear what constitutes a top Web service, though. There is not any public usage statistic for individual services.

The user can see in a separate list if a particular Web service is available and can send a bug report to WebserviceX. This functionality is not directly connected to the information about the individual Web services. WebserviceX offers a WSDL analyzer that shows the operations, request and response messages of a WSDL file. It does not validate the WSDL file against the given standard. Service costs and terms of usage are not provided.

3.3.3 WebserviceList

WebserviceList is an offer of IT Netix Inc. and currently lists around 280 Web services.

A Web service is described briefly and the time of the last update is given. Further, the provider and its homepage, if available, are listed. The WSDL file is linked but status information is not given. If the link to the WSDL file is broken this is not visible. Visitors can rate the Web service and are prevented from rating a Web service several times. However, even if the Web service is unavailable this is not necessarily reflected in the visitors’ rating.

WebserviceList supports key-word based search as well as category browsing. The keyword search takes the Web service name and the textual service description into account.

Service costs, terms of usage, information on availability, a WSDL parser or validator, or alternative interfaces are not supported. WebserviceList provides a link for help information or test functionality which is more often than not misused for adding another link to the provider's homepage or to the WSDL file.

3.3.4 StrikeIron

StrikeIron is a marketplace for Web services maintained by StrikeIron, Inc. It currently lists round about 70 Web services. The company's service is to handle authentication, billing and accounting, usage reporting, user tracking, security, uptime monitoring, and other productivity monitoring capabilities for their clients who offer their Web services through StrikeIron.

For every Web service they list the service name, the version, the provider with a link to the provider's home page, and the availability of the service. A brief service description is given and the WSDL file is linked. On a separate page more details concerning the terms of usage and the service costs are given. Commercial benefits, the target audience of the Web service, and other marketing advertisements are also added.

The WSDL file of the service can be displayed. Further, the file can be analyzed in a GUI that also allows invoking individual Web service operations for testing. Every operation is also described briefly. If the user enters invalid input data for testing an error message is displayed that describes the expected input format.

StrikeIron supports category browsing and keyword based search. Bug reporting or user feed-back is not supported.

3.3.5 RemoteMethods

RemoteMethods is a Web service directory by InfoGenius Inc. that lists round about 370 Web services of various providers.

For every Web service, remote methods lists its name, a short description, the time of the last update, the price, the user rating, the number of hits during the day and the number of overall hits for the service. Additional pricing information is displayed on a separate page; the WSDL file is linked and can be displayed as plain XML files.

The users can review the Web services writing an anonymous or an authenticated comment. Further, they can give a score to the Web service. It is also possible to report bugs. Registered users can also maintain their own list of favorite Web services with RemoteMethods.

RemoteMethods supports category browsing and keyword based search. Results can be sorted by name, price, rating, version, hits, and date. Alternative interfaces are not supported. Test functionality is not available.

3.3.6 Woogle

An alternative to Web service catalogs are search engines, specialized on Web services. A research prototype called Woogle (data.cs.washington.edu/webService/) has been implemented to augment keyword based search with template search and composition search as documented in [DHM+04]. It collects Web services from SalCentral, WebserviceList, XMethods, and RemoteMethods. The project supporting the development has run out and Woogle is no longer online. We nevertheless present some aspects of this application as found in an earlier visit here, as these are of interest due to the different characteristics of Web service search.

Woogle provides the services' name and provider as well as a description. Additionally the services status is indicated and the WSDL file is linked. A test feature is also available for many services. The WSDL file can be viewed and validated.

Service costs, terms of usage, usage statistics, bug reporting, user feed-back or rating mechanisms, and alternative interfaces are not supported.

The template search is the distinguishing search feature of Woogle. The user can specify a Web service template by defining the names of the desired functionality as well as the names of the input and output parameters. This template is then used to search for Web services with similar names. Further, this algorithm can also be used to search for Web services that are suitable to be composed with a given service. In this case, composition is limited to concatenation. The search returns Web services whose input parameters match the output parameters of the search template or vice versa.

The underlying algorithm is a clustering algorithm. As similarity measure Term Frequency/Inverse Document Frequency (TF/IDF) [SB88] is used. To compute the similarity between two Web services the similarity of their operations and of their input and output parameters is computed. The computed similarities are combined linearly according to chosen weights.

A Web service operation *op* is associated with two vectors $i = (p_i, c_i, op)$ and $o = (p_o, c_o, op)$ consisting of the input and output parameter names p_i, p_o and the concepts c_i, c_o they belong to. In a preprocessing step, parameter names are normalized, stop words are removed and artificial names such as, e.g., ``LocalTimeByZipCode'' are broken into individual terms. This turns the parameter list into a bag of words. Then the TF/IDF between the two input bags and the two output bags is computed. In a next step, an agglomerative clustering algorithm is applied to cluster the terms in each bag into semantically meaningful concepts. Then, the terms are replaced by the concepts they represent and the TF/IDF is computed for the input and output concepts respectively.

A Web service operation is identified by the textual description of the Web service it belongs to, by the textual description of the operation itself and by the input and output parameters. The similarity of the operation description is computed by taking the tokenized operation name, the WSDL documentation and the input and output terms, computing the TF/IDF on the two bags of words. The similarity between web service descriptions is computed similarly. Here, the bag of words is computed using the tokenized web service name, the WSDL document, the UDDI description, the tokenized operation names and the input and output terms. More details are provided in [DHM+04].

3.3.7 WSDLicious

This site (wsdlicio.us), presents a list of WSDL files in a feature-reduced del.icio.us manner. The site currently lists 63 entries, which seem to come from less than five people.

After registering one can add a link to a WSDL through the site or by adding a Bookmarklet to one's browser. To upload a WSDL one enters its address, a description, and freely chosen tags. All this information is displayed in the main list. Additionally, the operations provided by the services are shown. One can navigate through the list of services, by using tags, usernames, or operation names as categories.

WSDLicious provides an additional endpoint for each service, through which it can be called with WS-Reliable Messaging, thus guaranteeing delivery, but this function is broken at the time of writing. The last entry has been added to WSDLicious on August 28, 2006.

3.3.8 ProgrammableWeb

ProgrammableWeb, run by John Musser, is a site that aims at collecting information on and covering news about mash-ups and Web 2.0 APIs. It currently lists 429 services. The site stands out from the others in this category as it equally focuses on APIs and what (publicly) results from them, namely mash-ups.

Regarding services the information offered is the basics, such as name, address, and description, but it also has more advanced information such as information on protocols, functionality, security, support, and licensing. Moreover, there is a rating, comments, related links, and there is a list of mash-ups that have been built on top of this service. In addition to this information that is directly available on the details page, blog entries from the ProgrammableWeb blog are linked, which cover the respective service.

Just as service, mash-ups can be rated and commented. Developers who have registered their mash-ups are also listed in a developers section of the page. Both services and mash-ups can be searched, and they are categorized using tags.

The ProgrammableWeb blog covers the news about mash-ups and Web 2.0 services. It too is categorized via tags, entries frequently link to mash-ups and APIs covered in the other sections, and individual entries can be commented by the readers.

3.3.9 Obsolete Registries

Since the [BSF+06] report, some of the registries mentioned have shut down due to limited success. The list of recently failed privately managed Web service registries includes Bindingpoint, Salcentral, and eSynaps. Bindingpoint officially admits its shut-down. Salcentral has shrunk to become an advertisement place for books, seminars, and IT-service providers for Web services. eSynaps does still list a few WSDL files, but its search functionality returns arbitrary Web pages not limited in any way to Web services or related topics. It looks like this page was edited last in 2004.

3.4 Search Engines

Instead of using specialized search engines and manually maintained catalogues an alternative way of discovering Web services makes use of general purpose Web service search engines. To highlight the characteristics of this approach, we take Google Search as an example.

Without using any special functions that the search engine may provide one way to search for services, is of course to search for the keywords, one expects to be contained in pages describing desired services. Using “service” as a keyword is bound to yield a lot of superfluous results due to the broad sense of the term. Using more specific terms such as “WSDL” or “Web service” is probably going to produce more adequate results. The important question is then how to distinguish between pages that only talk about a service, but do not provide any and those that do provide something.

On way this distinction could be possible is by restricting the results to resources that are WSDL documents. In the case of Google, this can be done by adding “filetype:wsdl” to the keywords one wants to use to find the service (this approach of course also works for WSIL files). Using only the aforementioned term one can retrieve the number of WSDL files Google’s index contains. At the time of writing this number was about 34 100.

One has to mention that the number of WSDL files found through such a search does not translate to the same number of services: a WSDL file could be present several times, it could be just an example file pointing to invalid endpoints, or the endpoints could be out of date. It is beyond the scope of this report to conduct a thorough analysis of the search results. To nevertheless get a feel for the possible pitfalls we searched for “filetype:wsdl” and looked at the first ten results. Five of these were in fact WSDL documents, the rest were plain Web pages. Of the five WSDL files, we were able to call one service successfully, the other four resulted in various errors: the links in the WSDL were local (2x), server error (1x), connection timeout (1x). We expect this tendency to hold also on the larger scale.

4 Evaluation

In this section the results of the above analysis are summarized and the different Web service search approaches are compared according to the criteria developed before.

Table 1 presents a summary of our analysis. For every service, two rows are included: the first, row represents the results of [BSF+06] (plus historical information for the new criteria where these were accessible), the second row summarizes the results of the current survey. We have not listed the Salcentral and eSynaps, since these proved to be unreachable or irrelevant in [BSF+06] already, and we consider them obsolete.

The columns represent the evaluation criteria from the three groups listed in Section 3.1. The column “Search” takes one or more of the following entries: *search* (keyword search), *list* (listing of all services), *categories* (browsable categories), and *template* (template-based search). The column “Number of services” contains the absolute numbers and also shows the change that has taken place in the number of services since the first survey of [BSF+06]. The numbers should be taken with a grain of salt, as some numbers from [BSF+06] seem to have been only estimates, and the number of services is not accurately countable for pages that do not have a full list. “Alternative interfaces” contains additional possibilities for accessing the registry. Note however, that interfaces may have different capabilities although the protocol is the same. “Community support” lists which mechanisms the site provides that can foster the building of a community. It does not give information on if and how these possibilities are actually used. All other columns should be self-explanatory. Since “Service usage” was not a criterion in [BSF+06] there are entries were this item is unknown.

Overall, the number of publicly available Web service registries as well as the number of available services has decreased. Except for StrikeIron and ProgrammableWeb, all registries make a neglected impression. The interest in this topic seems to remain nevertheless, as the increased number of results on Google suggests when searching for WSDL files.

The UBR registry is the only approach that has tried to implement the UBR usage scenario as described in Section 2.1.1. The other registries are intended to be open to a large community but they are not synchronized and connected with each other. Therefore, they support the UBR usage scenario in a restricted way. A business specific marketplace as described in Section 0 is not among the public search catalogues. Only StrikeIron might be regarded as an attempt to realize this application scenario domain independently because it handles quality control and administration for their service providers and their service consumers, which is why StrikeIron can be considered a service broker.

Most service registries contain Web services that are only poorly described, links to provider home pages are broken, the services are not available and links to WSDL files are broken, too. This indicates that the quality and availability of registered services is hard to guarantee. As long as the registry provider does not care about these quality aspects the registry does not inspire trust. Also test and demo functionality as well as user feed-back mechanisms do not seem to ameliorate the quality problems. Public quality control only works in those places were a great public is attracted, as projects such as Wikipedia show.

Furthermore, many services offer functionality that is not of interest to a large community or that is too trivial, such as most conversion services. If the services are not of interest, the Web service registry degenerates to a playground. Services of interest do not need a central registry to become known. They are connected with the name of the service provider such as the Amazon E-Commerce services. Here, e. g., web shop owners can access sales statistics via remote procedure call. Such a service provides added value to a large community. It can be regarded as an example for the B2B usage scenario as described in Section 2.1.4. If the services offered by a registry do not provide added value the registry does not attract customers.

Most registries are intended to be used by users who already know that they will find the service they are looking for under a given category or with the right keyword. Consumers who seek information on how to solve a design problem at hand or who are not sure if they want to use a Web service at all do not find help in most registries. ProgrammableWeb is an exception that reveals the typical Web 2.0 features of social interaction. Working programming examples, the mash-ups, and the possibility to share experiences in discussion forums are important additional features that are not included in standards such as UDDI.

Overall, Web service registries on the public Internet suffers from a lack of acceptance. It can be seen that technical standards are not as important as providing additional functionality around such registries. Technical standards alone are only the backbone for the realization of a Web service registry. They describe the minimum of functionality that must be available, the least common denominator. Additional services could include quality control, administration and payment transactions as offered by StrikeIron or community specific knowledge and experience sharing as offered by ProgrammableWeb.

However, these findings only cover the public part of Web service registries. This survey does not allow drawing conclusions about the usage of Web service registries in company-internal SOAs.

Table 1: Overview of Web service discovery resources (as of May 8, 2007).

Name	Search	Number of services/ change	Availability information	Alternative interfaces	Test/ demo	WSDL parser	Terms of use/ price	Community support	Service usage (mash-ups)
UDDI Business Registry	search, categories	-	no	SOAP	no	no	no	-	no
	offline								
XMethods	list	3 124	no	SOAP, UDDI, WS-Inspection, DISCO, RSS	yes	yes	no	-	yes
	~	500 (-84%)	~	~	~	~	~	~	~
Binding-Point	search, categories	4 056	imprecise	-	yes	yes	yes	ratings, reviews	?
	offline								
WebserviceX	search, categories	70	no	RSS	example SOAP docs	yes	no	-	no
	~	71 (+1%)	~	~	~	~	~	~	~
Web Service List	search, categories	1 000	no	RSS	no	no	no	-	no
	~	280 (-68%)	~	~	partial	~	~	~	~
StrikeIron	search, categories	1 000	yes	SOAP	yes	yes	yes	-	no
	~	70 (-93%)	~	-	~	~	~	~	~
Woogle	search, categories, templates	885	yes	-	partial	imprecise	no	-	?
	offline								
Remote-Methods	search, categories	-	no	-	no	no	no	ratings, reviews	no
	~	372 (-)	~	~	~	~	~	~	~
WSDLicious	list, categories	45	no	-	no	no	no	-	no
	~	63 (+40%)	~	~	~	~	~	~	~
Programmable Web	search, list, categories	86	no	RSS	no	no	yes	blog, ratings, comments, forum, developer list	yes
	~	429 (+500%)	~	~	~	~	~		~
Google	search	25 600	imprecise	SOAP	no	no	no	-	no
	~	34 100 (+25%)	~	Ajax	~	~	~	~	~

5 Conclusion and Outlook

The above discussion and its results show that three aspects must be taken into consideration for public Web service registries: the type of service offered, the quality of the offer, and the way the services are presented to support customer decisions.

Type of service A central Web service registry is not a success factor per se. The central reachability alone is not decisive. Far more, it is important that the Web services themselves provide added value for business. There is no need for many different unit conversion Web services. They do not represent added value but are easy to program and are already contained in many standard software tools.

Information is a value in itself and access to crucial business information is a service in itself. A convenient way of delivery today is via the Internet, as it gets rid of the problem of updating local databases (that one has with information stores delivered via DVD for example), and allows a pay-per-use payment model. Therefore we see information providing services are good candidates for successful Web services, independent of their location in a central registry or on a provider page, as long as the offered information is of value.

Further, domain knowledge and highly domain specific functionality is another good of value that can either be bought with expensive software suites or on a function-by-function basis via the Internet. However, these are still niche services because domain specific functionality usually operates on domain specific and maybe even confidential data. Therefore such domain specific services are only scarcely found on the public Internet. For this type of service, the global usage scenario does not seem to be suitable. A B2B or Intranet scenario is more likely.

Quality of the offer For a registry to be attractive to developers, the services offered must be available, the registry provider must ensure their technical quality and the services must be described in such a way that they are easily integrated into other applications. Test functionality must be available in such a way that testing is possible without reading the technical interface description.

The attractiveness of a registry can be increased by offering additional services, as was seen with StrikeIron and ProgrammableWeb. However, even with high quality services and comfortable additional services for service providers and customers, a registry must first of all be able to offer services that are of interest. If this is not fulfilled even the best designed registry will fail.

Presentation As the above analysis has shown, programmers searching for Web services do not go straight to a Web service registry or to a service provider they know during the first stages of searching. In an early stage of a software development project, the make or buy decision must be made. On a smaller scale, this is the same with the decision for or against a Web service or Web 2.0 component. Web service registries do not support this early decision stage. As ProgrammableWeb shows the early decision stage needs to answer such questions as “Has anybody before used a Web service for the problem I need to solve?”, “Was this successful?”, “What do I have to be aware of?”, “Where do I find successfully running examples?”, “Which services exist?”, or “Which providers are there?” Only the last two questions can be answered through a Web services registry approach. The first questions seek experiences and advice from members of a peer-group of programmers. Sharing knowledge and experiences is the strong feature of Web 2.0. In this aspect, Web pages such as ProgrammableWeb are complementary to the Web service registry approach.

The analysis shows that there are currently no successful Web service registries that base their strategy on being an access point for service search alone. It seems necessary to add a value to that to be of lasting interest to developers. StrikeIron adds a broker service for providers, which may be interesting to the user also, since there is a third party which is interested in keeping the services in good shape. However, the small number of services offered limits the attractiveness of the site. ProgrammableWeb adds a community to the registry, which is maintained with incentives such as visibility on the site or the site’s blog. It remains to be seen whether this site can remain attractive and continue growing beyond the hype of Web 2.0.

References

[ACK+04] Alonso, G., Casati, F., Kuno, H., Machiraju, V.: Web Services. Concepts, Architectures and Applications. Springer-Verlag Berlin, Germany, 2004.

[BBM+01] Ballinger, K., Brittenham, P., Malhotra, A., Nagy, W. A., Pharies, S.: Web Services Inspection Language 1.0 (WS-Inspection). IBM, Microsoft, 2001. URL: http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnglobspec/html/ws-inspection.asp (2007/05/30).

[BH06] Baroudi, C., Halper, F.: Executive Survey: SOA Implementation Satisfaction. Hurwitz and Associates 2006. URL: http://www2.mindreef.com/_hurwitz.aspx?cid=aebc8bb4-8a15-48fa-94f3-473f9fc4c96c (2007/05/30).

[BKK+05] Beatty, J., Kakivaya, G., Kemp, D., Kuehnel, T., Lovering, B. et al.: Web Services Dynamic Discovery (WS-Discovery). Microsoft, 2005. URL: http://schemas.xmlsoap.org/ws/2004/10/discovery/ws-discovery.pdf (2007/05/30).

[BSF+06] Bachlechner, D., Siorpaes, K., Fensel, D., Toma, I.: Web Service Discovery - A Reality Check. Technical Report DERI-TR-2006-01-17, 2006.

[CPJ06] Crane, D., Pascarello, E., James, D. : Ajax in Action. Manning, Greenwhich, CT, 2006.

[CHR+04] Clement, L., Hately, A., von Riegen, C., Rogers, T.: UDDI Version 3.0.2, UDDI Spec. Technical Committee Draft, OASIS 2004. URL: http://uddi.org/pubs/uddi_v3.htm (2007-05-30).

[DHM+04] Dong, X., Halevy, A. Y., Madhavan, J., Nemes, E., Zhang, J.: Similarity Search for Web Services. In *Proc. of VLDB 2004*, pp. 372-383.

[DJ+05] Dostal, W., Jeckle, M., Melzer, I, Zengler, B.: Service-orientierte Architekturen mit Web Services. *Konzepte-Standards-Praxis*. Spektrum Akademischer Verlag, München, Germany, 2005.

[DSV04] Deutsch, A., Sui, L., Vianu, V.: Specification and Verification of Data-driven Web Services. In *Proc. of PODS 2004*, pp. 71-82.

[ebX01] ebXML Registry Project Team: Using UDDI to Find ebXML Reg/Reps. OASIS, UN/CEFACT 2001. URL: http://www.ebxml.org/specs/rrUDDI.pdf (2007/05/30).

[Fie00] Fielding, R.: Architectural Styles and the Design of Network-based Software Architectures, PhD Thesis, University of California, Irvine, 2000. URL: http://www.ics.uci.edu/~fielding/pubs/dissertation/top.htm (2007/05/22).

[FK05] Fan, J., Kambhampati, S.: A Snapshot of Public Web Services. *SIGMOD Record* 34(1), 2005, pp. 24-32.

[Gra01] Graham, S.: The Role of Private UDDI Nodes in Web Services, Part 1: Six Species of UDDI. IBM Technical Report 2001. URL: http://www-128.ibm.com/developerworks/webservices/library/ws-rpu1.html (2007/05/30).

[GTB01] Gonzlez-Castillo, J., Trastour, D. and Bartolini, C.: Description logics for matchmaking of services. In *KI-2001 Workshop on Applications of Description Logics*, September 2001.

[Hin06] Hinchcliffe, D.: A round of Web 2.0 reductionism. In Entreprise Web 2.0 Blog, 2006. URL: http://blogs.zdnet.com/Hinchcliffe/?p=41 (2007/05/22).

[Ley03] Leymann, F.: Web Services: Distributed Applications Without Limits. In *Proc. of BTW* 2003, pp. 2-23.

[LH03] Li, L. and Horrocks, I: A software framework for matchmaking based on semantic web technology. In *Proc. of the 12th International Conference on the World Wide Web*, Budapest, Hungary, May 2003.

[LLG+03] Latimer-Livingston, N. S., Graham, C., Correia, J. M., Schroder, N.: Survey Shows Why Firms Undertake Web Services Projects. Gartner Group 2003. URL: http://www.gartner.com/DisplayDocument?doc_cd=116069 (2007/05/30).

[MO07] Musser, J., O'Reilly, T.: Web 2.0. Principles and Best Practices. O'Reilly Media, Sebastopol, USA, 2007.

[MNBD06] Marlow, C., Naaman, M., Boyd, D., Davis, M.: HT06, tagging paper, taxonomy, Flickr, academic article, to read. In *Proc. of the 17th ACM Conference on Hypertext and Hypermedia*, Odense, Denmark, August 2006.

[MSC03] Microsoft Corporation: Enterprise UDDI Services: Three Usage Scenarios, 2003. URL: http://www.microsoft.com/windowsserver2003/techinfo/overview/ uddiscen.mspx (2007/05/30).

[MSC03b] Microsoft Corporation: UDDI Services: Qwest Technical Case Study. 2003. URL: http://www.microsoft.com/windowsserver2003/techinfo/overview/ qwest.mspx (2007/05/30).

[OAS06] The OASIS ebXML Joint Committee: The Framework for eBusiness. An OASIS White Paper, 2006. URL: http://www.oasis-open.org/committees/ download.php/17817/ebxmljc-WhitePaper-wd-r02-en.pdf (2007/05/30)

[Ore05] O'Reilly, T.: What is Web 2.0 – Design Patterns and Business Models for the Next Generation of Software. In *Tim O'Reilly (Blog)*, 2005. URL: http://www.oreillynet.com/pub/a/ oreilly/tim/news/2005/09/30/what-is-web-20.html (2007/05/22)

[PKP+02] Paolucci, M., Kawamura, T., Payne, T. and Sycara, K.: Semantic Matching of Web Services Capabilities. In I. Horrocks and J. Handler, editors, *1st Int. Semantic Web Conference (ISWC),* 2002, pages 333-347.

[SB88] Salton, G., Buckley, C.: Term-Weighting Approaches in Automatic Text Retrieval. *In Journal of Inf. Process. Manage.* 24(5) 1988, pp. 513-523.

[TSG02] The Stencil Group, Inc.: The Evolution of UDDI. UDDI.org White Paper, 2002. URL: http://uddi.org/pubs/the_evolution_of_uddi_20020719.pdf (2007/05/30).

[Vos06] Vossen, G.: Have Service-Oriented Architectures Taken a Wrong Turn Already? In *Proc. of the IFIP TC 8th International Conference on Research and Practical Issues of Enterprise Information Systems (CONFENIS)*, Vienna, Austria 2006.

[VH079] Vossen, G., Hagemann, St.: *Unleashing Web 2.0 – From Concepts to Creativity*. Morgan Kaufmann Publishers, San Francisco, 2007.

[WCL+05] Weerawarana, S., Curbera, F., Leymann, F., Storey, T., Ferguson, D. F.: Web Services Platform Architecture. *SOAP, WSDL, WS-Policy, WS-Addressing, WS-BPEL, WS-Reliable Messaging, and More*. Prentice Hall Upper Saddle River, NJ, 2005.

[WPA+01] Whittle, J., Probert, S., Adcock, M., Boxman, G., Becker, T.: Core Component Overview. Version 1.05 ebXML Core Components, OASIS, UN/CEFACT, 2001. URL: www.ebxml.org/specs/ccOVER.pdf (2007/05/30).

Working Papers, ERCIS